Grant Agreement No.: 258378

FIGARO

Future Internet Gateway-based Architecture of Residential Networks

Instrument**:** **Collaborative Project**

Thematic Priority: **THEME [ICT-2009.1.1] The Network of the Future**

D2.2 Design document of gateway-centric monitoring tools

Due date of deliverable: 30.09.2012

Actual submission date of *revised* version: 24.05.2013

Start date of project: October 1st 2010 Duration: 36 months

Project Manager: Henrik Lundgren, Technicolor R&D Paris

Abstract

This document describes the design of the FIGARO monitoring module. We describe the techniques to monitor the properties of the home network, the access link, and the link to neighbor gateways as well as the traffic traversing the gateway. In addition, we present the gateway-centric data collection architecture and the interface for other FIGARO modules to access monitoring data.

Project co-funded by the European Commission in the 7th Framework Programme (2007-2013)		
Dissemination Level		
PU	Public	✓
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Document Revision History

Version	Date	Description of change	Editor	Authors
V1.0	29.09.2012	Final version submitted to the EC	UPMC	UPMC, TNO, GUAVUS, TID, TRDP, POLITO, EURECOM
V1.1	24.05.2013	Revised version with all papers in the appendix included along with copyright notice.	UPMC	UPMC, TNO, GUAVUS, TID, TRDP, POLITO, EURECOM

Table of Contents

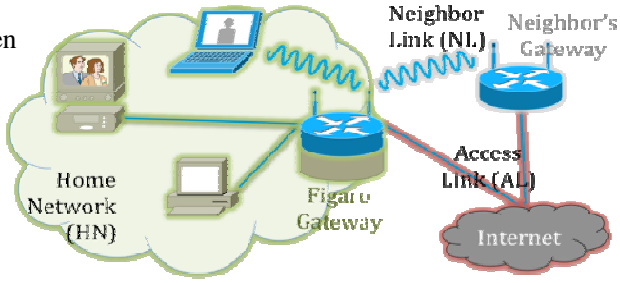
LIST OF ACRONYMS

AP	Access Point
API	Application Programming Interface
CPU	Central Processing Unit
DSL	Digital Subscriber Line
GUI	Graphical User Interface
HTTP	Hypertext Transfer Protocol
IP	Internet Protocol
ISP	Internet Service Provider
LAN	Local Area Network
LLDP	Link-Layer Discovery Protocol
LLTD	Link-Layer Topology Discovery
MAC	Medium Access Control
NOC	Network Operations Center
PDA	Personal Digital Assistant
RSSI	Received Signal Strength Indicator
RTT	Round Trip Time
SNR	Signal-to-Noise Ratio
WLAN	Wireless LAN

1 INTRODUCTION

This deliverable describes the components of the FIGARO monitoring module. Deliverable D2.1 identified the monitoring functionality that FIGARO requires and differentiated between three monitoring targets:

- Home Network (green, HN) often called LAN on home gateways,
- Internet Access Link (red, AL) typically DSL or Cable, and
- Link to a Neighbor (gray, NL), which is usually WLAN.



We design tools to monitor several properties of each of these targets as well as of the traffic traversing the home gateway. We group these monitoring tools into four categories:

- **Path performance and topology:** This category groups tools that use active probing to infer properties of network paths inside the home network and the access link. We focus on available bandwidth, latency, reachability, and topology (in particular the topology inside the home network).
- **Wireless link properties:** Many home networks today use 802.11 to interconnect devices inside the home and, in FIGARO, we also assume that a device in a home network will connect to a neighbor's gateway over WiFi. This group of tools will measure the WiFi links (both inside the home and to the neighbor). In particular, we collect the MAC layer goodput, achieved throughput, frame error rate; the wireless physical layer bit rate; the channel busy time; SNR and RSSI; and the IP layer packet error rate.
- **Traffic analyzer:** This category passively monitors the traffic that traverses the gateway to track active applications and their performance. The traffic analyzer maps packets into flows (using the typical 5-tuple to identify flows: source and destination IP, source and destination port, and protocol). Then, it extracts for each flow: the application that generated the flow, typical flow statistics; and performance metrics of the flow (for instance, achieved throughput). This module also generates alarms when application performance is poor.
- **Gateway status:** The last category samples the resource consumption (in particular, the available storage capacity and the CPU and memory utilization) and the status of the gateway (Internet access synchronization rate and the on-periods).

FIGARO gateways will provide an interface so that other FIGARO modules can query these measurements. Figure 1 presents the functional split of the FIGARO monitoring module. The four categories of monitoring tools feed data to a Collector that stores the data in a scalable fashion. Other FIGARO modules query on-gateway data from the monitoring module using the Data Export. A Central Collector (located outside of the home, possibly at a cloud or NOC) stores monitoring data long term (this Central Collector is part of the FIGARO Lookup Service described in Deliverable D2.1). The rest of this deliverable details the collection system the On-gateway Collector and Central Collector (Section 2) and the Data Export (Section 3) as well as the design of monitoring tools in each of the four categories (Section 4). We end in Section 5 with the IPFIX specifications for each monitoring module.

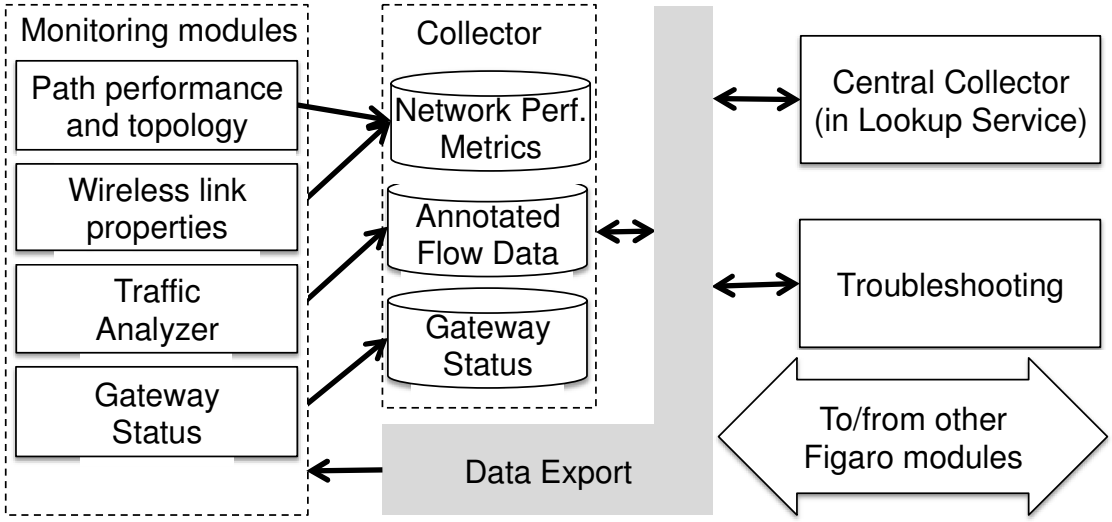


Figure 1: FIGARO monitoring functional graph.

2 COLLECTOR

The Collector subsystem is designed to collect, aggregate and store gateway performance, traffic and status data for troubleshooting and reporting as well as for other FIGARO modules to use.

2.1 Streaming architecture and processing

The Collector system is designed as a pipeline-based streaming architecture, which has four processing steps:

(1) Ingest raw data streams,

(2) Optionally: filter, join and time-bin the data,

(3) Optionally: aggregate data into summary views that are organized by time window,

(4) Archive the raw or the processed data in an on-gateway (or optionally, a cloud-based) database for SQL-based further interactive processing.

The pipeline architecture allows the data streams to be continuously processed first, and then inserted into a database, and in this manner is designed for low-latency processing.

The Collector system is designed to ingest data from different sources - both streaming or batched, and from socket or file interfaces. Different data formats are supported by a pluggable data adaptor architecture, although for FIGARO the primary adaptor is IPFIX (see Section 5). The Collector can be optionally configured to: (i) filter or subsample data, (ii) merge multiple streams of data by a user-defined operation into a new combined stream, (iii) optionally organize (or bin) the data along time boundaries - the time-binning is useful when records arrive out of order and need to be aligned on a common time boundary, and finally: (iv) the data stream can be aggregated into summary projections based on a priori interest in specific fields of the data. Such summary projections can be used to pre-process the data to be stored, and so are useful in scenarios where reducing the raw data volume to be stored is required.

2.2 Storage

For storage, the Collector relies on on-gateway storage, whose size is separately configured. Access to stored data is via the Data Export (described in Section 3). Data is stored in a circular disk buffer on-gateway so that old data is overwritten by fresh incoming data. The size of the circular buffer is configurable. However, because the on-gateway storage is limited, the Collector also sends old data records into an off-gateway data center / cloud (which we call the *Central Collector*), where the data will be archived and analyzed without the compute limitations of the gateway. The Central Collector is part of the Lookup Service described in Deliverable D1.2. The on-gateway Collector sends data to the Central Collector periodically, either when the circular buffer is full or after a configurable timeout, whatever happens first.

2.3 High availability

The Collector is also designed to operate in a highly available (1+1) mode. Such high availability may be needed for mission-critical environments (e.g., e-health) where requirements on data loss are stringent. High Availability is accomplished by creating a master and slave pipelines with periodic heartbeats. This way, should one data pipeline fail, the other pipeline takes over, minimizing any data loss.

2.4 Collector in a federated FIGARO network

In scenarios where multiple FIGARO gateways are deployed in federated manner, it is valuable to have a holistic view across gateways. Such a scenario can be supported as shown in Figure 2. Each gateway runs a local Collector pipeline, which ingests data streams, aggregates, and locally stores raw and processed data as before. In addition, the Collector also ships a copy of the processed data (and optionally, if transport is available, the raw data) to a Cloud Data Center where a Central Collector is running. This Central Collector aggregates the data from the multiple gateways and archives the data for the long term, and makes the results available via GUIs and SQL interfaces for further processing.

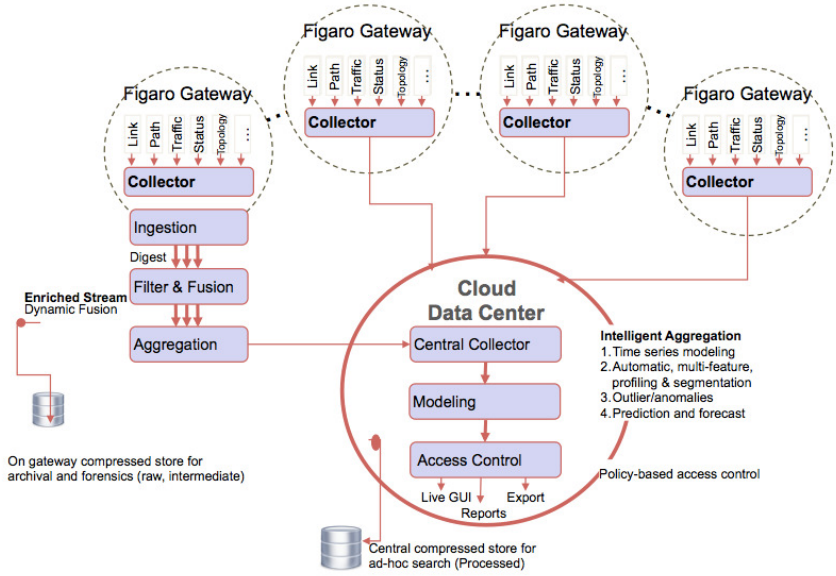


Figure 2. Collector in a Federated FIGARO Network Environment (note that all gateways support the ingestion; filter & fusion; and aggregation functions, we omit them for clarity)

2.5 Ongoing work

We are currently exploring a virtual machine-based design so that the basic functions of a Collector can be embedded as agent software in constrained resource environments like a gateway. We are also exploring how the Collector architecture can be extended to support an event-processing model to support active triggers, based on pattern-matching rules, to other system processes.

3 DATA EXPORT

FIGARO modules will require data collected by the home gateway, which is stored either on the gateway or on the Central Collector, in different scenarios. As detailed in Deliverable D2.1, some modules require just the most recent data, whereas other modules require historical data. Moreover, some modules require a continuous stream of data whereas others only need measurement results on-demand. Finally, some modules may want to access the raw data, whereas other modules may want projections or computed results.

The Data Export will interact with the on-gateway Collector and the Central Collector to provide all these different views to FIGARO modules. The Collector ingests all the IPFIX records on each gateway and sends a copy of these feeds to the Central Collector. The Data Export will receive this stream and export with two basic mechanisms:

1. Publish/subscribe: publish the raw stream continuously to any module that subscribes to it.
2. SQL access: data export will also allow modules to perform SQL queries on data stored in a given gateway or in the long-term storage in the Central Collector.

A FIGARO module can use the first mechanism to subscribe to the raw stream when it needs recent data or a continuous data stream, and unsubscribe to the stream, once it is complete. Of course a module can stay subscribed to the feed forever; for example, the Central Collector in the cloud is built by continuously subscribing to the raw stream from each gateway's Collector. Furthermore, a module can use the second mechanism when it needs a specific event or query a specific period of time in the past. For example, if a module needs to do a focused search on an IP address, it can run a SQL query against the local on-gateway store. These two mechanisms apply trivially to all datasets that are continuously monitored at gateways. Some measurements, however, are only done on-demand. In this case, the Data Export explicitly triggers the measurement and subscribes FIGARO module that requested the measurement to the resulting data stream. The measurement module will then starts sending the IPFIX feed to the Collector and the export will work with the usual publish/subscribe mechanism. When the FIGARO module receives all the required results, it will then unsubscribe. This design gives the flexibility to the modules to ingest data and keeps the Collector stateless and simple.

4 MONITORING MODULES

This section summarizes the measurement methods and tools used in FIGARO. Our goal is to give an overview of all metrics we are collecting. The details of the measurement methodology and their evaluation are explained in the referenced papers.

4.1 Path properties and topology

We design techniques to measure the available bandwidth, latency, and reachability of both the access link and the home network. In addition, we test techniques to infer the topology of the home network.

Measurements of access link properties: The capacity of residential access links is often asymmetric, hence we measure both downstream and upstream throughput (or available bandwidth). We assume control of the home gateway and of a server connected to the Internet to be able to distinguish upstream and downstream bandwidth. We also measure last-mile latency, which is the latency to the first hop inside the ISP network. Reachability is a special case of latency measurements when there is no response to probes. We describe the techniques to monitor downstream and upstream throughput and last-mile latency in the paper "Broadband Internet Performance: A View From the Gateway" in Proc. of ACM SIGCOMM 2012. There is an increasing interest in studying the performance of Internet access links. Because of many confounding factors in a home network or on end hosts, however, thoroughly understanding access network performance requires deploying measurement infrastructure in users' homes as gateway devices. Hence, the FIGARO setup offers new opportunities for studying access network performance. We test techniques to measure throughput and latency of access links from typical home gateways deployed in nearly 4,000 homes across 8 ISPs in the United States. This study was done in collaboration with the BISMark project in Georgia Tech and in conjunction with the Federal Communication Commission's study of broadband Internet access in the United States.

Measurements inside the home network: Home networks are becoming ever more heterogeneous. This means that a single network consists of many different physical- and link-layer technologies and topologies, interconnecting many different devices with each other and the Internet. We design techniques to estimate available bandwidth in the home network with access only to the home gateway. We describe Allbest, our tool to estimate available bandwidth in home networks in the paper "Real-Time Probing of Available Bandwidth in Home Networks" in IEEE Communications Magazine, 49(6), June 2011. In the paper, we refer to the home gateway as server for generality. For troubleshooting and content optimization, the home gateway also needs a precise map of the home network topology. Today, the two main protocols to discover the topology of a local area network are Link-Layer Topology Discovery (LLTD) protocol and the Link-Layer Discovery Protocol (LLDP). A performance evaluation of these protocols was presented in our paper "Performance Analysis of Home Network Topology Discovery Protocols" in Proc. of IEEE Conference on Consumer Communication & Networking Conference (CCNC), 2012. Our results show that none of the protocols fulfill all operators' needs, but LLTD comes closest. LLTD's performance is closest to the operators' requirements and LLTD outputs a complete topology. Unfortunately, LLTD is a Microsoft proprietary protocol and hence we cannot adapt it for FIGARO.

LLDP fits best in the overall FIGARO architecture, because of its high level of standardization and availability of open source code. But LLDP in itself does not generate a full topology as LLTD does. Usually, a network management system intelligently infers the topology from the low level information about links and neighbors it obtains from the Management Information Bases (MIBs) filled by the LLDP protocol. For FIGARO, we are building a monitoring module that deploys LLDP as well as intelligent reasoning to deduce the home network topology. This topology is then communicated to the Collector by means of IPFIX.

4.2 Wireless measurements

We design techniques to measure the behavior of the wireless interface and more generally the behavior of the wireless environment, as the wireless channels for WiFi communication are on a public band and require careful watch to ensure performance achievement.

Using the centralized position of the gateway as wireless Access Point, we can track different metrics related to the wireless link between the gateway and the wireless clients, that could be individual stations (such as computers, laptops, PDA, tablets, or smartphones) or remote federated gateways used for offloading or bandwidth aggregation.

In general our monitoring techniques are based on three different approaches to obtain the sought for metrics. Most wireless interface cards export some basic state information and performance indicators through more or less well-defined APIs. Thus, this information is typically available using open source tools, such as the Linux wireless tool *iw*, or by probing the /proc file system where some information is exported. However, although those tools are convenient and typically support a broad set of drivers, they have a few drawbacks and are not always usable for our purposes. First, the information available may not cover all required information. Second, for the information available there may be uncertainties regarding whether this is raw data or it has been processed (and in that case how), yielding poor accuracy. Therefore, a second complementary approach is to implement the monitoring of certain metrics directly in the wireless driver. This gives access to information that may not be exposed through its APIs and also eliminates data uncertainties. All information is not available even in the driver though, wherefore we in certain cases adopt a third complementary approach and directly capture and inspect the data traffic received by the wireless interface. There is a trade-off regarding the CPU load induced by the different approaches. Accessing the standard APIs is more lightweight than direct traffic measurements, wherefore the latter need to be carefully instrumented if the home gateway is resource constrained or currently heavily loaded.

By using the last approach, we obtain the information needed to compute the metrics of MAC-layer average frame payload size, MAC-layer frame error rate, IP packet error rate, MAC layer goodput, and IP layer achieved throughput. Wireless physical layer bit rate (PHY rate) and Received Signal Strength information (RSSI) is commonly available through standard APIs. However, this information is also available from traffic measurements (e.g., through the radiotap header). Channel busy time (or channel load) is rarely available in current wireless cards. However, this is a useful metric and we develop alternative ways to obtain information related to this metric.

Using self-loaded assessment through traffic measurements based on the defined metric, as presented in Bandwidth Monitoring in Multi-rate 802.11 WLANs with Elastic Traffic Awareness, in Proc. Of IEEE Globecom 2011, each gateway can classify their status as light, regular, or heavy through measurements. Wireless stations might be migrated from one gateway to another, a technique referred as offloading; offloading approaches tend to become mainstream thanks to the interest of cellular providers into neighboring WiFi networks, but only a federated approach might deliver expected performance.

In addition, we also monitor the contribution of the wireless neighborhood. In this case we refer to external contributor to the wireless band – i.e., other access point not in the federation. Still, due to the property of IEEE802.11 behind the WiFi standard, all the packets in wireless range are received by the gateway, which makes it an interesting observation point for remote diagnosis and problem mitigation. In “Fair WLAN Backhaul Aggregation”, in Proc. of ACM Mobicom’10 conference we introduced various estimators to get the relevant information for the alien wireless neighborhood, such as utilization rate, wireless capacity and so on. Also, this information can be used within the federation gateways to track in a distributed way the performance of the same wireless channel.

We have also developed an approach where we enhance the standard channel busy time metric. We first break it down to external (other home networks) and internal (the own home network), and then for the internal busy time, we break it down per client. The busy time is useful e.g., when troubleshooting poor home network performance. As previously mentioned, the contributions from neighboring networks can be estimated (both federated and alien networks). By further breaking it down to per client busy time inside our own network, one can also efficiently troubleshoot inside the home network. We discuss this in our "Wireless Home Network Monitoring", Technicolor Technical Report, CR-PRL-2012-09-001, September 2012. Furthermore, in our ongoing work we explore this further and show that busy time combined with wireless PHY rate can reveal the impact of the rate anomaly problem, i.e., the performance degradation impact caused by one or more clients operating on very low wireless PHY rates(s).

Finally, we have shown that by profiling the wireless cards it is possible to estimate the IP layer (UDP) throughput using SNR information. This can be used when CPU load is a concern, or when there is a need to probe throughput performance over a large number of combinations e.g., when searching for a better wireless configuration. We present this approach in "MIMO Wireless Networks with Directional Antennas in Indoor Environments, In Proc. of IEEE Infocom (mini-conference) 2012", where we use this approach to evaluate a large number of combinations of wireless rates and antenna sectors.

4.3 Traffic analyzer

The traffic analyzer passively observes packets traversing the gateway to track flow-level statistics and performance indicators as well as overall performance indicators for all traffic traversing the gateway. In addition, the traffic analyzer will generate alarms when the performance of a particular application becomes poor. We develop the traffic analyzer in two steps. First, we develop methods to passively monitor application performance and to predict when performance is poor using data collected at end hosts. Then, we adapt these methods so that they can run on home gateways.

Our end-host analysis uses data collected directly at the laptops and desktops of volunteer users with the HostView measurement tool, which we developed in our previous work. HostView not only collects network, application and machine level data, but also gathers feedback directly from users. We first analyzed the HostView data to characterize how the performance of networked applications varies across different networking environments in the paper "Characterizing end-host application performance across multiple networking environments", in Proc. of IEEE INFOCOM (mini-conference), 2012. We then worked on correlating network performance with user feedback. Our preliminary analysis (presented in the paper "Performance of Networked Applications: The Challenges in Capturing the User's Perception", in Proc. of ACM SIGCOMM Workshop on Measurements Up the Stack, 2011) showed that typical network performance metrics individually are not sufficient to predict user satisfaction. Our paper "Predicting User Dissatisfaction with Internet Application Performance at End-Hosts" recently submitted to IEEE INFOCOM 2013 developed predictors of user dissatisfaction with Internet application performance. We trained these predictors using HostView data. The main challenges of modeling user dissatisfaction with network performance come from the scarcity of user feedback and the fact that poor performance episodes are rare. We developed a methodology to build training sets in face of these challenges. Then, we showed that predictors based on non-linear support vector machine achieve higher true positive rates than predictors based on linear models. Therefore, in FIGARO we are developing predictors based on non-linear support vector machine to detect when application performance is poor.

Our study of network traffic from end-hosts allowed us to determine the set of relevant network performance metrics to track and the method to raise alarms when performance application is poor. Deploying these techniques in a home gateway, however, brings extra challenges. First, gateways have

limited CPU and memory. Second, at end-hosts we can directly query which application processes generate each traffic stream, but this information is not available at the gateway. Furthermore, typical payload analysis to match application signatures consumes too much CPU. Hence, we need to optimize tracking of application performance so that it can run efficiently on gateways and identify application in network traffic. Our paper "An end-host view on local traffic at home and work" in PAM 2012 studies traffic patterns in home networks using HostView's data. This analysis sheds light on the requirements of home network traffic and typical applications used in home networks. Given this application mix, we select an application identification technique that combines port-based classification with the analysis of destination names. Destination names are particularly useful to identify different services using HTTP. For HTTP, we also get the content-type of HTTP requests. When present, the content-type tells us, for instance, whether a given request is a video, an image, or simple text.

For integrating these techniques in home gateways, we are collaborating with Bismark project (http://projectbismark.net/) that already has home gateways modified to perform measurements. In particular, Bismark gateways have passive measurements of per-flow statistics (for instance, number of bytes and packets per flow) and already collect destination names. We are modifying the Bismark software to passively track application performance metrics as well (such as round-trip times, failed connections, losses). For each flow, we also collect the sequence number so that we can compute RTTs and jitter (using the same technique as in HostView) and the TCP flags so that we can compute failed connections and retransmissions. For HTTP connections, we also capture the content-type. The implementation and integration of the enhanced Bismark-passive firmware in the FIGARO gateway will be part of WP1 in the next year.

4.4 Gateway status

We developed simple modules to monitor the gateway status metrics. Most gateway metrics are system information maintained by the kernel and are available through standardized APIs or system calls.

For CPU, main memory, and swap memory usage we obtain the information by querying the Linux /proc virtual file system. We read the content of the files: "cpuacct.usage", "memory.usage_in_bytes", and "memory.memsw.usage_in_bytes", respectively. This gives the information about the CPU usage, the main memory and the swap in use on the gateway. The sampling rate is a configurable parameter of module. In our current implementation we use a granularity of five seconds between samples to ensure detailed statistics with a very minimal CPU overhead. The uptime metric can be obtained either by the command "uptime" or by reading /proc/uptime, which provides two integers, the first of them being the time since the machine has boot and is in seconds. We query this metric periodically or on demand. The metrics first time join, current online status and availability last months can be deduced from the historical uptime data. The access to Internet access synchronization rate depends on the DSL modem. It may not be directly available through driver API and our current workaround is to obtain this metric through a web-based access to the modem. The available storage space metric, as many other file system related statistics, can be obtained by using the statfs (or statfs64) function on any Linux system (other system have a similar system call). This system call returns a structure (buf) which contains among other information the block size (buf.f_bsize) and the number of free blocks (buf.f_bfree). Hence, the available storage space in bytes can be reported as buf->f_bfree * buf->f_bsize. To accommodate large filesystems, the statfs64 call should be preferred, and the reported available storage space should be encoded as a 64 bit integer.

5 APPENDIX: IPFIX SPECIFICATIONS

This appendix presents the IPFIX template for each of the measurement modules of FIGARO. We first list the attributes of these templates that are already defined in in RFC 5102 (Information Model for IP Flow Information Export). For each category of measurements, we present the IPFIX template and define the FIGARO-specific attributes. The numbering of FIGARO-specific attributes starts from an offset *X*, which we will define to respect pre-existing IPFIX templates.

5.1 Attributes from RFC 5102

The following table list the attributes defined in RFC 5102, which are also used by some of the FIGARO measurement modules.

Attribute	IPFIX Element ID	IPFIX Element Name	Group	Type
StartTime	150	flowStartSeconds	timestamp	dateTimeSeconds
EndTime	151	flowEndSeconds	timestamp	dateTimeSeconds
SourceIPv4	8	sourceIPv4Address	ipHeader	ipv4Address
SourceIPv6	27	sourceIPv6Address	ipHeader	ipv6Address
Source Port	7	sourceTransportPort	transportHeader	unsigned16
DestinationIPv4	12	destinationIPv4Address	ipHeader	ipv4Address
DestinationIPv6	28	destinationIPv6Address	ipHeader	Ipv6Address
Destination Port	11	destinationTransportPort	transportHeader	unsigned16
Time Stamp	258	collectionTimeMilliseconds	timestamp	dateTimeMilliseconds
VLAN name	243	Dot1qVlanId	identifier	Unsigned16

5.2 Path properties and topology

Access link properties

Attribute	Format	Purpose
Timestamp	UTC	Absolute time of measurement
Source IP Address	IP v4/6	WAN address of the HG
Access up throughput Instant	Numeric	Throughput measured by parallel TCP (Instantaneous)
Access up throughput Max	Numeric	Throughput measured by parallel TCP (Maximum)
Access up throughput Median	Numeric	Throughput measured by parallel TCP (Median)
Access up throughput Std.	Numeric	Throughput measured by parallel TCP (standard deviation)
Access down throughput Instant	Numeric	Throughput measured by parallel TCP (Instantaneous)
Access down throughput Max	Numeric	Throughput measured by parallel TCP (Maximum)
Access down throughput Median	Numeric	Median throughput measured by parallel TCP (Median)
Access down throughput Std.	Numeric	Throughput measured by parallel TCP (standard deviation)
Last-mile latency Instant	Numeric	RTT to the first hop after HG
Last-mile latency Median	Numeric	RTT to the first hop after HG (median)
Last-mile latency Std.	Numeric	RTT to the first hop after HG (standard deviation)
Reachability	Boolean	1= reachable; 0=unreachable

We define the following FIGARO-specific attributes.

Attribute	IPFIX Element ID	IPFIX Element Name	Type
Access up throughput Instant	X+1	accessUpThroughputInst	unsigned64
Access up throughput Max	X+2	accessUpThroughputMax	unsigned64
Access up throughput Median	X+3	accessUpThroughputMed	unsigned64
Access up throughput Std.	X+4	accessUpThroughputStd	unsigned64
Access down throughput	X+5	accessDownThroughputInst	unsigned64
Access down throughput	X+6	accessDownThroughputMax	unsigned64
Access down throughput	X+7	accessDownThroughputMed	unsigned64
Access down throughput	X+8	accessDownThroughputStd	unsigned64
Last-mile latency Instant	X+9	lastMileLatencyInst	unsigned64
Last-mile latency Median	X+10	lastMileLatencyMed	unsigned64
Last-mile latency Std	X+11	lastMileLatencyStd	unsigned64
Reachability	X+12	reachability	boolean

The following template is proposed to capture access link properties.

0								1								2								3							
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Version = 0x000a (10)																Length (148)															
Export Time (UTC)																															
Sequence Number																															
Observation Domain ID																															
Set ID (2) [Template]																Set Length (bytes)															
Template ID (256)																Field Count (23)															
0	collectionTimeMilliseconds (258)															Length (4)															
0	sourceIPv4Address (8)															Length (4)															
0	sourceIPv6Address (26)															Length (16)															
[Figaro Enterprise ID#]																															
1	accessUpThroughputInst (X+1)															Length (8)															
[Figaro Enterprise ID#]																															
1	accessUpThroughputMax (X+2)															Length (8)															
[Figaro Enterprise ID#]																															
1	accessUpThroughputMed (X+3)															Length (8)															
[Figaro Enterprise ID#]																															
1	accessUpThroughputStd (X+4)															Length (8)															
[Figaro Enterprise ID#]																															
1	accessDownThroughputInst (X+5)															Length (8)															
[Figaro Enterprise ID#]																															
1	accessDownThroughputMax (X+6)															Length (8)															
[Figaro Enterprise ID#]																															
1	accessDownThroughputMed (X+7)															Length (8)															
[Figaro Enterprise ID#]																															
1	accessDownThroughputStd (X+8)															Length (8)															
[Figaro Enterprise ID#]																															
1	lastMileLatencyInst (X+9)															Length (8)															
[Figaro Enterprise ID#]																															
1	lastMileLatencyMed (X+10)															Length (8)															
[Figaro Enterprise ID#]																															
1	lastMileLatencyStd (X+11)															Length (8)															
[Figaro Enterprise ID#]																															
1	reachability (X+12)															Length (1)															

Home network properties

Attribute	Format	Purpose
Timestamp	UTC	Absolute time of measurement
Prober IP Address	IP v4/6	LAN address of the HG
Probed IP Address	IP v4/6	LAN address of the device being probed
Probe Method	Boolean	HrPing (1) or UDP (0)
Path Capacity Instant	Numeric	Estimated path capacity by Allbest (Instantaneous)
Path Capacity Max	Numeric	Estimated path capacity by Allbest (Maximum)
Path Capacity Median	Numeric	Estimated path capacity by Allbest (Median)
Path Capacity Std.	Numeric	Estimated path capacity by Allbest (Standard deviation)
Path Available BW Instant	Numeric	Estimated available BW of the path by Allbest (Instantaneous)
Path Available BW Max	Numeric	Estimated available BW of the path by Allbest (Max)
Path Available BW Median	Numeric	Estimated available BW of the path by Allbest (Median)
Path Available BW Std.	Numeric	Estimated available BW of the path by Allbest (Standard deviation)

We define the following FIGARO-specific attributes.

Attribute	Enterprise Element ID	Enterprise Element Name	Type
Probe Method	X+13	probemethod	boolean
Path Capacity Instant	X+14	pathcapacityInst	unsigned64
Path Capacity Max	X+15	pathcapacityMax	unsigned64
Path Capacity Median	X+16	pathcapacityMed	unsigned64
Path Capacity Std.	X+17	pathcapacityStd	unsigned64
Path Available BW Instant	X+18	pathAvaBandwidthInst	unsigned64
Path Available BW Max	X+19	pathAvaBandwidthMax	unsigned64
Path Available BW Median	X+20	pathAvaBandwidthMed	unsigned64
Path Available BW Std.	X+21	pathAvaBandwidthStd	unsigned64

The following template is proposed to capture home network properties.

0								1								2								3							
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Version = 0x000a (10)																Length (148)															
Export Time (UTC)																															
Sequence Number																															
Observation Domain ID																															
Set ID (2) [Template]																Set Length (bytes)															
Template ID (256)																Field Count (23)															
0	collectionTimeMilliseconds (258)															Length (4)															
0	sourceIPv4Address (8)															Length (4)															
0	sourceIPv6Address (26)															Length (16)															
0	destinationIPv4Address (12)															Length (4)															
0	destinationIPv6Address (28)															Length (16)															
[Figaro Enterprise ID#]																															
1	Probemethod (X+13)															Length (1)															
[Figaro Enterprise ID#]																															
1	PathcapacityInst (X+14)															Length (8)															
[Figaro Enterprise ID#]																															
1	PathcapacityMax (X+15)															Length (8)															
[Figaro Enterprise ID#]																															
1	PathcapacityMed (X+16)															Length (8)															
[Figaro Enterprise ID#]																															
1	PathcapacityStd (X+17)															Length (8)															
[Figaro Enterprise ID#]																															
1	pathAvaBandwidthInst (X+18)															Length (8)															

[Figaro Enterprise ID#]		
1	pathAvaBandwidthMax (X+19)	Length (8)
[Figaro Enterprise ID#]		
1	pathAvaBandwidthMed (X+20)	Length (8)
[Figaro Enterprise ID#]		
1	pathAvaBandwidthStd (X+21)	Length (8)

Home network topology

FIGARO will use a monitoring module combining LLDP with some additional local intelligence to gather home network topology. The information that may be retrieved by this module is listed in the following table.

Attribute	Format	Purpose
Time Stamp	UTC	Absolute time at which a topology config is sent via IPFIX
Node ID	UUID	Universally unique ID for nodes in the LAN
Neighbor node ID	UUID	Universally unique ID for nodes in the LAN
Link type	string	Description of link types (Eth100/1000, Wlanb/g/n, IEEE1901, etc.)

UUID (universally unique identifier) is an identifier standard standardized by the OSF (Open Software Foundation) as part of the Distributed Computing Environment. A UUID is a 16-byte (128 bit) number. In its canonical form, a UUID is represented by 32 hex digits and is displayed in 5 groups separated by hyphens. There are about 3*1038 possible UUIDs. UUIDs are documented as part ITU-T Rec. X667, ISO/IEC 9834-8:2005. The equivalent of ITU-TX667 is IETF RFC 4122.

We define the following FIGARO-specific attributes.

Attribute	Enterprise Element ID	Enterprise Element Name	Type
Node ID	X+22	nodeID	UUID
Neighbor node ID	X+23	neighborID	UUID
Link type	X+24	linkType	String

The following template is proposed to capture home network topology.

0								1								2								3							
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Version = 0x000a (10)																Length (148)															
Export Time (UTC)																															
Sequence Number																															
Observation Domain ID																															
Set ID (2) [Template]																Set Length (bytes)															
Template ID (256)																Field Count (23)															
0	Dot1qVlanId (243)															Length (4)															
0	sourceIPv4Address (8)															Length (4)															
0	sourceIPv6Address (26)															Length (16)															
[Figaro Enterprise ID#]																															
1	nodeID_1 (X+22)																														
Length (128 bit)																															
...																															
[Figaro Enterprise ID#]																															
1	nodeID_N (X+22)																														
Length (128 bit)																															

[Figaro Enterprise ID#]		
1	neighborID_1_1 (X+23)	
Length (128 bit)		
...		
[Figaro Enterprise ID#]		
1	neighborID_1_K1 (X+23)	
Length (128 bit)		
1	neighborID_2_1 (X+23)	
Length (128 bit)		
...		
[Figaro Enterprise ID#]		
1	neighborID_2_K2 (X+23)	
Length (128 bit)		
...		
1	neighborID_N_1 (X+23)	
Length (128 bit)		
...		
[Figaro Enterprise ID#]		
1	neighborID_n_KN (X+23)	
Length (128 bit)		
[Figaro Enterprise ID#]		
1	linkType_1_1 (X+24)	Length (65535)
...		
[Figaro Enterprise ID#]		
1	linkType_1_K1 (X+24)	Length (65535)
[Figaro Enterprise ID#]		
1	linkType_2_1 (X+24)	Length (65535)
...		
[Figaro Enterprise ID#]		
1	linkType_2_K2 (X+24)	Length (65535)
...		
[Figaro Enterprise ID#]		
1	linkType_n_1 (X+24)	Length (65535)
...		
[Figaro Enterprise ID#]		
1	linkType_n_KN (X+24)	Length (65535)

For each node, its direct neighbors should be all listed with corresponding link type in the IPFIX template. Assuming the home network has N devices, device 1 has K1 neighbors with corresponding K1 direct links; device 2 then has up to K2 neighbors, therefore K2 direct links to the neighbors; a the Nth device has KN direct neighbors via KN links. On the other hand, there is a possibility that a link between two devices is not symmetric, in that the uplink/downlink can be of either different link layer technologies or different data rate of a same technology. In the template design above, each linkType entry in the template is single direction. It means the entry linkType_1_2 and linkType_2_1 represent uplink and downlink between device 1 and device 2.

5.3 Wireless measurements

We use RFC 5101 for references, 5102 for existing elements and RFC 5153 for implementation details.

MAC Layer goodput

Attribute	Format	Purpose
Timestamp	datetimeMicroseconds	Absolute time of the measurement
SourceMacAddress	macAddress	MAC address of the station considered as source
destinationMacAddress	macAddress	MAC address of the station considered as destination
elasticDownlinkGoodput	Unsigned64 with quantity semantic	TCP goodput on the downlink
nonElasticDownlinkGoodput	Unsigned64 with quantity semantic	UDP goodput on the downlink
elasticUplinkGoodput	Unsigned64 with quantity semantic	TCP goodput on the uplink
nonElasticUplinkGoodput	Unsigned64 with quantity semantic	UDP goodput on the uplink

MAC-layer Average Frame Payload size (P)

Attribute	Format	Purpose
Timestamp	datetimeMicroseconds	Absolute time of the measurement
bssID	macAddress	BSSid of network considered
payloadSize	Unsigned64 with quantity semantic	Average payload size

Achieved Throughput

Here we use IP in the description of the Achieved Throughput.

Attribute	Format	Purpose
Timestamp	datetimeMicroseconds	Absolute time of the measurement
sourceIPv4Address	ipv4Address	Source IP of the flow
sourceIPv6Address	ipv6Address	
destinationIPv4Address	ipv4Address	Destination IP of the flow
destinationIPv6Address	ipv6Address	
achievedThroughput	Unsigned64 with quantity semantic	bits

MAC-layer Frame Error Rate (FER)

Attribute	Format	Purpose
Timestamp	datetimeMicroseconds	Absolute time of the measurement
SourceMacAddress	macAddress	MAC address of the station considered as source
destinationMacAddress	macAddress	MAC address of the station considered as destination
frameErrorRate	float64 with quantity semantic	FER

Wireless Physical Layer Bit rate

Attribute	Format	Purpose
Timestamp	datetimeMicroseconds	Absolute time of the measurement
SourceMacAddress	macAddress	MAC address of the station considered as source
destinationMacAddress	macAddress	MAC address of the station considered as destination
phyRate	float64 with quantity semantic	Physical data rate of the transmission

Wireless Overall Average Physical Layer Bit rate

Attribute	Format	Purpose
Timestamp	datetimeMicroseconds	Absolute time of the measurement
bssID	macAddress	BSSid of network considered
overallPhysicalRate	Unsigned64 with quantity semantic	Overall Physical Rate based on the rate of all participant

Channel busy time

Attribute	Format	Purpose
Timestamp	datetimeMicroseconds	Absolute time of the measurement
bssID	macAddress	BSSid of network considered
busyTime	Unsigned64 with quantity semantic	Number of microseconds of busyness of the channel

Signal-to-Noise Ratio (SNR)

Attribute	Format	Purpose
Timestamp	datetimeMicroseconds	Absolute time of the measurement
SourceMacAddress	macAddress	MAC address of the station considered as source
destinationMacAddress	macAddress	MAC address of the station considered as destination
SNR	Unsigned8 with quantity semantic	Signal to noise ratio

Received Signal Strength Indicator (RSSI)

Attribute	Format	Purpose
Timestamp	datetimeMicroseconds	Absolute time of the measurement
SourceMacAddress	macAddress	MAC address of the station considered as source
destinationMacAddress	macAddress	MAC address of the station considered as destination
RSSI	Unsigned8 with quantity semantic	Received Signal Strength Indicator

IP layer packet error rate

Attribute	Format	Purpose
Timestamp	datetimeMicroseconds	Absolute time of the measurement
sourceIPv4Address	ipv4Address	Source IP of the flow
sourceIPv6Address	ipv6Address	
destinationIPv4Address	ipv4Address	Destination IP of the flow
destinationIPv6Address	ipv6Address	
PER	Float64 with quantity semantic	Packet error rate

As a result we have to introduce the following Figaro Enterprise ID:

Name	Enterprise element ID	Length (bytes)
elasticDownlinkGoodput	X+25	8
noneElasticDownlinkGoodput	X+26	8
elasticUplinkGoodput	X+27	8
noneElasticUplinkGoodput	X+28	8
achievedThroughput	X+29	8
frameErrorRate	X+30	8
phyRate	X+31	8
bssID	X+32	6
overallPhysicalRate	X+33	8
busyTime	X+34	8
SNR	X+35	1
RSSI	X+36	1
PER	X+37	8

The following template is proposed to capture wireless measurements.

0								1								2								3							
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Version = 0x000a (10)																Length (148)															
Export Time (UTC)																															
Sequence Number																															
Observation Domain ID																															
Set ID (2) [Template]																Set Length (bytes)															
Template ID (256)																Field Count (23)															
0	collectionTimeMilliseconds (258)															Length (4)															
0	sourceIPv4Address (8)															Length (4)															
0	sourceIPv6Address (26)															Length (16)															
0	destinationIPv4Address (8)															Length (4)															
0	destinationIPv6Address (26)															Length (16)															
0	SourceMacAddress (80)															Length (6)															
0	DestinationMacAddress (56)															Length (6)															
0	wlanChanneld (146)															Length (1)															
0	wlanSsid (147)															Length (32)															
[Figaro Enterprise ID#]																															
1	elasticDownlinkGoodput (X+25)															Length (8)															
[Figaro Enterprise ID#]																															
1	noneElasticDownlinkGoodput (X+26)															Length (8)															
[Figaro Enterprise ID#]																															
1	elasticUplinkGoodput (X+27)															Length (8)															
[Figaro Enterprise ID#]																															
1	noneElasticUplinkGoodput (X+28)															Length (8)															
[Figaro Enterprise ID#]																															
1	achievedThroughput (X+29)															Length (8)															
[Figaro Enterprise ID#]																															
1	frameErrorRate (X+30)															Length (8)															
[Figaro Enterprise ID#]																															
1	phyRate (X+31)															Length (8)															
[Figaro Enterprise ID#]																															
1	bssID (X+32)															Length (6)															
[Figaro Enterprise ID#]																															
1	overallPhysicalRate (X+33)															Length (8)															
[Figaro Enterprise ID#]																															
1	busyTime (X+34)															Length (8)															
[Figaro Enterprise ID#]																															
1	SNR (X+35)															Length (1)															
[Figaro Enterprise ID#]																															
1	RSSI (X+36)															Length (1)															
[Figaro Enterprise ID#]																															
1	PER (X+37)															Length (8)															

5.4 Traffic analyzer

Per-flow metrics

Attribute	Format
Start Time	UTC
End Time	UTC
Application Protocol	HTTP, FTP, etc.
Content Type	MIME
URL	String
Source IP	IP v4/6
Source Port	Numeric
Destination IP	IP v4/6
Destination Port	Numeric
UpLink bytes	Numeric
UpLink packets	Numeric
Downlink bytes	Numeric
Downlink packets	Numeric
UpLink data rate	Numeric
Downlink data rate	Numeric
Jitter average	Numeric
Jitter standard deviation	Numeric
Jitter median	Numeric
Jitter 95th%	Numeric
RTT average	Numeric
RTT standard deviation	Numeric
RTT median	Numeric
RTT 95th%	Numeric
Resets	Numeric
Retransmissions	Numeric

We define the following FIGARO-specific attributes.

Attribute	Enterprise Element ID	Enterprise Element Name	Type
Application Protocol	X+38	applicationProtocol	unsigned8
Content Type	X+39	contentType	string
URL	X+40	url	string
Uplink	X+41	uplink	unsigned64
Downlink	X+42	Downlink	unsigned64
UpLink data rate	X+43	uplinkdatarate	unsigned64
DownLink data rate	X+44	downlinkdatarate	unsigned64

Jitter average	X+45	jitterAvg	unsigned64
Jitter standard deviation	X+46	jitterSTD	unsigned64
Jitter median	X+47	jitterMed	unsigned64
Jitter 95th%	X+48	jitter95	unsigned64
RTT average	X+49	rttAvg	unsigned64
RTT standard deviation	X+50	rttSTD	unsigned64
RTT median	X+51	rttMed	unsigned64
RTT 95th%	X+52	rtt95	unsigned64
Resets	X+53	resets	unsigned64
Retransmissions	X+54	retransmissions	unsigned64

The following template is proposed to capture per-flow metrics.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Version = 0x000a (10)																Lemgth (148)															
Export Time (UTC)																															
Sequence Number																															
Observation Domain ID																															
Set ID (2) [Template]																Set Length (bytes)															
Template ID (256)																Field Count (20)															
0	flowStartSeconds (150)															Length (4)															
0	flowEndSeconds (151)															Length (4)															
0	sourceIPv4Address (8)															Length (4)															
0	sourceIPv6Address (26)															Length (16)															
0	sourceTransportPort (7)															Length (2)															
0	destinationIPv4Address (12)															Length (4)															
0	destinationIPv6Address (28)															Length (16)															
0	destinationTransportPort (11)															Length (2)															
[Figaro Enterprise ID#]																															
1	applicationProtocol (X+38)															Length (1)															
[Figaro Enterprise ID#]																															
1	contentType (X+39)															Length (65535)															
[Figaro Enterprise ID#]																															
1	url (X+40)															Length (65535)															
[Figaro Enterprise ID#]																															
1	uplink (X+41)															Length (8)															
[Figaro Enterprise ID#]																															
1	downlink (X+42)															Length (8)															
[Figaro Enterprise ID#]																															
1	uplink data rate(X+43)															Length (8)															
[Figaro Enterprise ID#]																															
1	downlink data rate(X+44)															Length (8)															
[Figaro Enterprise ID#]																															
1	jitterAvg (X+45)															Length (8)															
[Figaro Enterprise ID#]																															

1	jitterSTD (X+46)	Length (8)
[Figaro Enterprise ID#]		
1	jitterMed (X+47)	Length (8)
[Figaro Enterprise ID#]		
1	jitter95 (X+48)	Length (8)
[Figaro Enterprise ID#]		
1	rttAvg (X+49)	Length (8)
[Figaro Enterprise ID#]		
1	rttSTD (X+50)	Length (8)
[Figaro Enterprise ID#]		
1	rttMed (X+51)	Length (8)
[Figaro Enterprise ID#]		
1	rtt95(X+52)	Length (8)
[Figaro Enterprise ID#]		
1	resets (X+53)	Length (8)
[Figaro Enterprise ID#]		
1	retransmissions (X+54)	Length (8)

Per-gateway metrics

Attribute	Format
Start Time	UTC
End Time	UTC
Source IP	IP v4/6
UpLink achieved throughput	Numeric
Downlink achieved throughput	Numeric
Failed connections	Numeric

We define the following FIGARO-specific attributes.

Attribute	Enterprise Element ID	Enterprise Element Name	Type
Uplink achieved throughput	X+55	uplinkthroughput	unsigned64
Downlink achieved throughput	X+56	Downlinkthroughput	unsigned64
Failed connections	X+57	failedConnections	unsigned64

The following template is proposed to capture per-gateway traffic metrics.

0								1								2								3							
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Version = 0x000a (10)								Lemgth (148)																							
Export Time (UTC)																															
Sequence Number																															
Observation Domain ID																															
Set ID (2) [Template]																Set Length (bytes)															
Template ID (256)																Field Count (20)															

0	flowStartSeconds (150)	Length (4)
0	flowEndSeconds (151)	Length (4)
0	sourceIPv4Address (8)	Length (4)
0	sourceIPv6Address (26)	Length (16)
[Figaro Enterprise ID#]		
1	uplink achieved throughput(X+55)	Length (8)
[Figaro Enterprise ID#]		
1	downlink achieved throughput(X+56)	Length (8)
[Figaro Enterprise ID#]		
1	failed connections(X+57)	Length (8)

5.5 Gateway status

Attribute	Format	Purpose
Time Stamp	UTC	Absolute time at which a measurement is taken
CPU utilization	Numeric	Fraction or percentage of CPU used
Memory utilization	Numeric	Fraction or percentage of Memory used
Available storage capacity	Numeric	Absolute amount of Available storage
Internet Access Synchronization rate	Numeric	"sync rate" on the access link
Current online status	Boolean	Same as reachability in 3.2
UpTime	Numeric	Online since UpTime hours
Availability since last month	Numeric	Percentage
First Time Join	UTC	Absolute time of the first time the gateway joined the federation

We define the following FIGARO-specific attributes.

Attribute	Enterprise Element ID	Enterprise Element Name	Type
CPU Utilization	X+58	CpuUtilization	Float32
Memory Utilization	X+59	MemoryUtilization	Float32
Available Storage Capacity	X+60	AvailableStorageCapacity	Unsigned64
Internet Access Synchronization Rate	X+61	InternetAccessSyncRate	Unsigned64
Current Online Status	X+62	onlineState	Boolean
UpTime	X+63	upTime	Float32
Availability SinceLastMonth	X+64	availabilitySinceLastMonth	Float32
First Time Join	X+65	firstTimeJoin	dateTimeMilliseconds

The following template is proposed to capture the gateway status.

0								1								2								3							
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Version = 0x000a (10)																Length (148)															
Export Time (UTC)																															
Sequence Number																															
Observation Domain ID																															
Set ID (2) [Template]																Set Length (bytes)															
Template ID (256)																Field Count (23)															
0	collectionTimeMilliseconds (258)															Length (4)															
[Figaro Enterprise ID#]																															
1	cpuUtilization (X+58)															Length (4)															
[Figaro Enterprise ID#]																															
1	memoryUtilization (X+59)															Length (4)															
[Figaro Enterprise ID#]																															
1	AvailableStorageCapacity (X+60)															Length (8)															
[Figaro Enterprise ID#]																															
1	InternetAccessSyncRate (X+61)															Length (8)															
[Figaro Enterprise ID#]																															
1	CurrentOnlineStatus (X+62)															Length (8)															
[Figaro Enterprise ID#]																															
1	upTime (X+63)															Length (8)															
[Figaro Enterprise ID#]																															
1	availabilitySinceLastMonth (X+64)															Length (8)															
[Figaro Enterprise ID#]																															
1	firstTimeJoin (X+65)															Length (8)															

6 APPENDIX: INCLUDED PAPERS

S. Sundaresan, W. de Donato, N. Feamster, R. Teixeira, S. Crawford, and A. Pescape, *Broadband Internet Performance: A View From the Gateway*, in Proc. of ACM SIGCOMM, August 2011.

A. Delphinanto, T. Koonen, F. den Hartog, *Real-time Probing of Available Bandwidth in Home Networks*, IEEE Commun. Mag. 49, 6 (June 2011), pp. 134-140.

E. G. Diaz Castellanos, A. Delphinanto, F. den Hartog, *Performance analysis of home network topology discovery protocols*, IEEE Conference on Consumer Communication & Networking Conference (CCNC 2012), Las Vegas NV, US, January 2012.

Claudio Rossi, Claudio Casetti, Carla Fabiana Chiasserini, *Bandwidth Monitoring in Multi-rate 802.11 WLANs with Elastic Traffic Awareness*, IEEE Globecom 2011, Houston, TX, December 2011

D. Giustiniano, E. Goma, A. Lopez Toledo, Julian Morillo, Ian Dangerfield, Pablo Rodriguez, *Fair WLAN Backhaul Aggregation*, ACM Mobicom 2010, Chicago, September 2010.

N. Deng, A-K Pietilainen, H. Lundgren, *Wireless Home Network Monitoring*, Technicolor Technical Report, CR-PRL-2012-09-001, September 2012.

T. H. Kim, T. Salonidis, H. Lundgren, *MIMO Wireless Networks with Directional Antennas in Indoor Environments*, IEEE INFOCOM (mini-conference), Orlando, FL, March 2012.

D. Joumblatt, O. Goga, R. Teixeira, J. Chandrashekar, N. Taft, *Characterizing end-host application performance across multiple networking environments*, IEEE INFOCOM (mini-conference), Orlando, FL, March 2012.

D. Joumblatt, R. Teixeira, J. Chandrashekar, N. Taft, *Performance of Networked Applications: The Challenges in Capturing the User's Perception*, W-MUST 2011- ACM SIGCOMM workshop, Toronto, Canada, August 19, 2011

D. Joumblat, J. Chandrashekar, B. Kveton, Nina Tafty, Renata Teixeira, *Predicting User Dissatisfaction with Internet Application Performance at End-Hosts*, *submitted to* IEEE INFOCOM 2013.

A. Reggani, F. Schneider, R. Teixeira, *An end-host view on local traffic at home and work*, Passive and Active Measurement Conference, Zurich, Switzerland, March 2012.

Papers are included in this appendix with proper permissions. Specifically for IEEE papers, see the copyright notice below:

Broadband Internet Performance: A View From the Gateway

Srikanth Sundaresan
Georgia Tech
Atlanta, USA
srikanth@gatech.edu

Walter de Donato
University of Napoli Federico II
Napoli, Italy
walter.dedonato@unina.it

Nick Feamster
Georgia Tech
Atlanta, USA
feamster@cc.gatech.edu

Renata Teixeira
CNRS/UPMC Sorbonne Univ.
Paris, France
renata.teixeira@lip6.fr

Sam Crawford
SamKnows
London, UK
sam@samknows.com

Antonio Pescapè
University of Napoli Federico II
Napoli, Italy
pescape@unina.it

ABSTRACT

We present the first study of network access link performance measured directly from home gateway devices. Policymakers, ISPs, and users are increasingly interested in studying the performance of Internet access links. Because of many confounding factors in a home network or on end hosts, however, thoroughly understanding access network performance requires deploying measurement infrastructure in users' homes as gateway devices. In conjunction with the Federal Communication Commission's study of broadband Internet access in the United States, we study the throughput and latency of network access links using longitudinal measurements from nearly 4,000 gateway devices across 8 ISPs from a deployment of over 4,200 devices. We study the performance users achieve and how various factors ranging from the user's choice of modem to the ISP's traffic shaping policies can affect performance. Our study yields many important findings about the characteristics of existing access networks. Our findings also provide insights into the ways that access network performance should be measured and presented to users, which can help inform ongoing broader efforts to benchmark the performance of access networks.

Categories and Subject Descriptors

C.2.3 [**Computer-Communication Networks**]: Network Operations—*Network Management*; C.2.3 [**Computer-Communication Networks**]: Network Operations—*Network Operations*

General Terms

Management, Measurement, Performance

Keywords

Access Networks, Broadband Networks, BISMark, Benchmarking

1. INTRODUCTION

Of nearly two billion Internet users worldwide, about 500 million are residential broadband subscribers [19]. Broadband penetration is likely to increase further, with people relying on home connectivity for day-to-day and even critical activities. Accordingly, the Federal Communication Commission (FCC) is actively developing performance-testing metrics for access providers [5,15, 35]. Policymakers, home users, and Internet Service Providers (ISPs) are in search for better ways to benchmark home broadband Internet performance.

Benchmarking home Internet performance, however, is not as simple as running one-time "speed tests". There exist countless tools to measure Internet performance [7,14,29,32]. Previous work has studied the typical download and upload rates of home access networks [12, 24]; others have found that modems often have large buffers [24], and that DSL links often have high latency [26]. These studies have shed some light on access-link performance, but they have typically run one-time measurements either from an end-host inside the home (from the "inside out") or from a server on the wide-area Internet (from the "outside in"). Because these tools run from end-hosts, they cannot analyze the effects of confounding factors such as home network cross-traffic, the wireless network, or end-host configuration. Also, many of these tools run as one-time measurements. Without continual measurements of the same access link, these tools cannot establish a baseline performance level or observe how performance varies over time.

This paper measures and characterizes broadband Internet performance from home gateways. The home gateway connects the home network to the user's modem; taking measurements from this vantage point allows us to control the effects of many confounding factors, such as the home wireless network and load on the measurement host (Section 4). The home gateway is always on; it can conduct unobstructed measurements of the ISP's network and account for confounding factors in the home network. The drawback to measuring access performance from the gateway, of course, is that deploying gateways in many homes is incredibly difficult and expensive. Fortunately, we were able to take advantage of the ongoing FCC broadband study to have such a unique deployment.

We perform our measurements using two complementary deployments; the first is a large FCC-sponsored study, operated by SamKnows, that has installed gateways in over 4,200 homes across the United States, across many different ISPs. The second, BISMark, is deployed in 16 homes across three ISPs in Atlanta. The SamKnows deployment provides a large user base, as well as diversity in ISPs, service plans, and geographical locations. We designed BISMark to allow us to access the gateway remotely and run repeated experiments to investigate the effect of factors that we could not study in a larger "production" deployment. For example, to study the effect of modem choice on performance, we were able to install different modems in the same home and conduct experiments in a controlled setting. Both deployments run a comprehensive suite of measurement tools that periodically measure throughput, latency, packet loss, and jitter.

We characterize access network throughput (Section 5) and la-

SIGCOMM'11, August 15–19, 2011, Toronto, Ontario, Canada.

tency (Section 6) from the SamKnows and BISMark deployments. We explain how our throughput measurements differ from common "speed tests" and also propose several different latency metrics. When our measurements cannot fully explain the observed behavior, we model the access link and verify our hypotheses using controlled experiments. We find that the most significant sources of throughput variability are the access technology, ISPs' traffic shaping policies, and congestion during peak hours. On the other hand, latency is mostly affected by the quality of the access link, modem buffering, and cross-traffic within the home.

This study offers many insights into both access network performance and the appropriate measurement methods for benchmarking home broadband performance. Our study has three high-level lessons, which we expand on in Section 7:

- ISPs use different policies and traffic shaping behavior that can make it difficult to compare measurements across ISPs.
- There is no "best" ISP for all users. Different users may prefer different ISPs depending on their usage profiles and how those ISPs perform along performance dimensions that matter to them.
- A user's home network equipment and infrastructure can significantly affect performance.

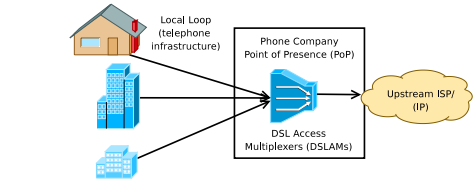
As the first in-depth analysis of home access network performance, our study offers insights for users, ISPs, and policymakers. Users and ISPs can better understand the performance of the access link, as measured directly from the gateway; ultimately, such a deployment could help an ISP differentiate performance problems within the home from those on the access link. Our study also informs policy by illustrating that a diverse set of network metrics ultimately affect the performance that a user experiences. The need for a benchmark is clear, and the results from this study can serve as a principled foundation for such an effort.

2. RELATED WORK

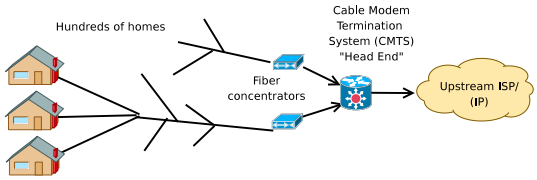
This section presents related work; where appropriate, we compare our results to these previous studies in Sections 5 and 6.

From access ISPs. Previous work characterizes access networks using passive traffic measurements from DSL provider networks in Japan [8], France [33], and Europe [26]. These studies mostly focus on traffic patterns and application usage, but they also infer the round-trip time and throughput of residential users. Without active measurements or a vantage point within the home network, however, it is not possible to measure the actual performance that users receive from their ISPs, because user traffic does not always saturate the user's access network connection. For example, Siekkinen *et al.* [33] show that applications (*e.g.*, peer-to-peer file sharing applications) often rate limit themselves, so performance observed through passive traffic analysis may reflect application rate limiting, as opposed to the performance of the access link.

From servers in the wide area. Other studies have characterized access network performance by probing access links from servers in the wide area [11, 12]. Active probing from a fixed set of servers can characterize many access links because each link can be measured from the same server. Unfortunately, because the server is often located far from the access network, the measurements may be inaccurate or inconsistent. Isolating the performance of the access network from the performance of the end-to-end path can be challenging, and dynamic IP addressing can make it difficult to determine whether repeated measurements of the same IP address are in fact measuring the same access link over time. A remote server also cannot isolate confounding factors, such as whether the user's own traffic is affecting the access-link performance.



(a) DSL.



(b) Cable.

Figure 1: Access network architectures.

From inside home networks. The Grenouille project in France [1] measures the performance of access links using a monitoring agent that runs from a user's machine inside the home network. Neti@Home [23] and BSense [4] also use this approach, although these projects have fewer users than Grenouille. PeerMetric [25] measured P2P performance from about 25 end hosts. Installing software at the end-host measures the access network from the user's perspective and can also gather continuous measurements of the same access link. Han *et al.* [18] measured access network performance from a laptop that searched for open wireless networks. This approach is convenient because it does not require user intervention, but it does not scale to a large number of access networks, cannot collect continuous measurements, and offers no insights into the specifics of the home network configuration.

Other studies have performed "one-time" measurements of access-link performance. These studies typically help users troubleshoot performance problems by asking the users to run tests from a Web site and running analysis based on these tests. Netalyzr [29] measures the performance of commonly used protocols using a Java applet that is launched from the client's browser. Network Diagnostic Tool (NDT) [7] and Network Path and Application Diagnostics (NPAD) [14] send active probes to detect issues with client performance. Glasnost performs active measurements to determine whether the user's ISP is actively blocking BitTorrent traffic [17]. Users typically run these tools only once (or, at most, a few times), so the resulting datasets cannot capture a longitudinal view of the performance of any single access link. In addition, any technique that measures performance from a device inside the home can be affected by factors such as load on the host or features of the home network (*e.g.*, cross-traffic, wireless signal strength). Finally, none of these studies measure the access link directly from the home network gateway.

3. ACCESS NETWORKS: BACKGROUND

We describe the two most common access technologies from our deployments: Digital Subscriber Line (DSL) and cable. Then, we explain how a user's choice of service plan and local configuration can affect performance. Although a few users in our deployments have fiber-to-the-node (FTTN), fiber-to-the-premises (FTTP), and WiMax, we do not have enough users to analyze these technologies.

DSL networks use telephone lines; subscribers have dedicated lines between their own DSL modems and the closest DSL Access Multiplexer (DSLAM). The DSLAM multiplexes data between the access modems and upstream networks, as shown in Figure 1a. The most common type of DSL access is asymmetric (ADSL), which provides different upload and download rates. In cable access networks, groups of users send data over a shared medium (typically coaxial cable); at a regional *headend*, a Cable Modem Termination System (CMTS) receives these signals and converts them to Ethernet, as shown in Figure 1b. The physical connection between a customer's home and the DSLAM or the CMTS is often referred to as the *local loop* or *last mile*. Users buy a service plan from a provider that typically offers some *maximum* capacity in both the upload and download directions.

ADSL capacity. The ITU-T standardization body establishes that the achievable rate for ADSL 1 [20] is 12 Mbps downstream and 1.8 Mbps upstream. The ADSL2+ specification [21] extends the capacity of ADSL links to at most 24 Mbps download and 3.5 Mbps upload. Although the ADSL technology is theoretically able to reach these speeds, there are many factors that limit the capacity in practice. An ADSL modem negotiates the operational rate with the DSLAM (often called the *sync rate*); this rate depends on the quality of the local loop, which is mainly determined by the distance to the DSLAM from the user's home and noise on the line. The maximum IP link capacity is lower than the sync rate because of the overhead of underlying protocols. The best service plan that an ADSL provider advertises usually represents the rate that customers can achieve if they have a good connection to the DSLAM. Providers also offer service plans with lower rates and can rate-limit a customer's traffic at the DSLAM.

Modem configuration can also affect performance. ADSL users or providers configure their modems to operate in either *fastpath* or *interleaved* mode. In *fastpath* mode, data is exchanged between the DSL modem and the DSLAM in the same order that they are received, which minimizes latency but prevents error correction from being applied across frames. Thus, ISPs typically configure fastpath only if the line has a low bit error rate. *Interleaving* increases robustness to line noise at the cost of increased latency by splitting data from each frame into multiple segments and interleaving those segments with one another before transmitting them.

Cable capacity. In cable networks, the most widely deployed version of the standard is Data Over Cable Service Interface Specification version 2 (DOCSIS 2.0) [22], which specifies download rates up to 42.88 Mbps and upload rates up to 30.72 Mbps in the United States. The latest standard, DOCSIS 3.0, allows for hundreds of megabits per second by bundling multiple channels. Cable providers often offer service plans with lower rates. The service plan rate limit is configured at the cable modem and is typically implemented using a token bucket rate shaper. Many cable providers offer *PowerBoost*, which allows users to download (and, in some cases, upload) at rates that are higher than the contracted ones, for an initial part of a transfer. The actual rate that a cable user receives will vary with the network utilization of other users connecting to the same headend. The CMTS controls the rate at which cable modems transmit. For instance, Comcast describes that when a CMTS's port becomes congested, it ensures fairness by scheduling heavy users on a lower priority queue [3].

4. MEASUREMENT INFRASTRUCTURE

We describe the measurement infrastructure that we deployed and the datasets that we collected. We first motivate the need for deploying measurement infrastructure directly at the gateway; then, we describe the *SamKnows* and *BISMark* (Broadband Internet Service benchMark) gateway deployments.

Factor	How we address it
Wireless Effects	Use a wired connection to modem.
Cross Traffic	Measure cross traffic and avoid it/account for it.
Load on gateway	Use a well-provisioned gateway.
Location of server	Choose a nearby server.
End-to-end path	Focus on characterizing the last mile.
Gateway configuration	Test configuration in practice and controlled settings.

Table 1: Confounding factors and how we address them.

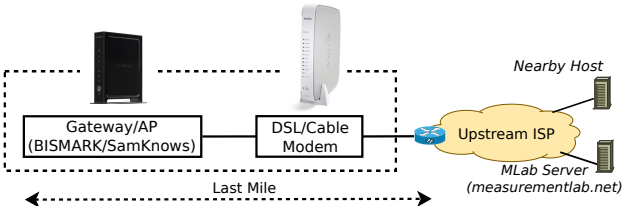


Figure 2: Our gateway device sits directly behind the modem in the home network. They take measurements both to the *last mile router* (first non-NAT IP hop on the path) and to wide area hosts.

4.1 Why a Gateway?

Deploying measurements at gateway devices offers the following advantages over the other techniques discussed in Section 2:

- *Direct measurement* of the ISP's access link: the gateway sits behind the modem; between the access link and all other devices at the home network as shown in Figure2. This allows us to isolate the effect of confounding factors such as wireless effects and cross traffic.
- *Continual/longitudinal measurements*, which allow us to meaningfully characterize performance of ISPs for individual users.
- *The ability to instrument a single home with different hardware and configurations*, which allows us to explore the effects of multiple factors on performance. In some deployments, we were even able to swap modems to study their effect on performance, holding all other conditions about the network setup equal.

Table 1 summarizes the challenges involved in conducting such a study, and how deploying gateways solves them. We now describe the two gateway deployments in our study.

4.2 Gateway Deployments

Our study uses two independent gateway deployments. The first, the FCC/SamKnows gateway deployment, collected data from over 4,200 users across different ISPs in the United States, as of January 2011. This deployment currently has over 10,000 users. Our goal in using the measurements from this deployment is to achieve *breadth*: we aim to classify a large set of users across a diverse set of ISPs and geographical locations. The second, the BISMark deployment, collects measurements from a smaller, focused group of users from different ISPs and service plans in Atlanta. Our goal with the measurements from this deployment is to achieve *depth*: this platform allows us to take measurements with detailed knowledge of how every gateway is deployed; we can also take repeated measurements and conduct specific experiments from the same deployment with different settings and configurations.

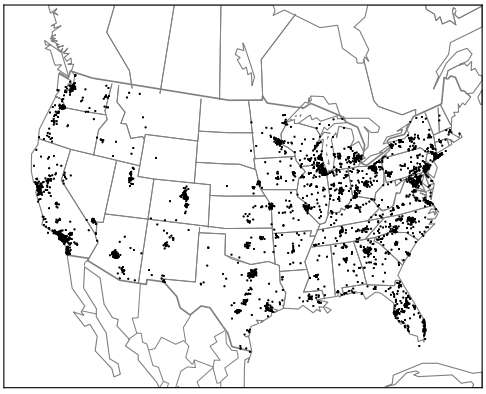


Figure 3: SamKnows deployment: 4,200 devices, 16 ISPs.

Gateway deployments entail significant challenges concerning the resource constraints of the gateway platform and the need to remotely maintain and manage the devices (especially because these devices are deployed in homes of "real users"); we omit discussion of these logistical challenges due to lack of space and instead focus on the details of the platforms and the measurements we collect.

4.2.1 SamKnows

SamKnows specializes in performance evaluation of access networks; it has studied access ISP performance in the United Kingdom and has now contracted with the FCC for a similar study in the United States. SamKnows deployed gateways in each participant's home either directly behind the home user's router or behind the home wireless router; the devices can be updated and managed remotely. The gateway is a Netgear WNR3500L RangeMax Wireless-N Gigabit router with a 480 MHz MIPS processor, 8 MB of flash storage, and 64 MB of RAM. We use active measurement data from the SamKnows study from December 14, 2010 to January 14, 2011. This dataset comprises measurements from 4,200 devices that are deployed across sixteen different ISPs and hundreds of cities in the United States. The volunteers for the study were recruited through `http://www.testmyisp.com`. Figure 3 shows a map of the deployment.

Table 2 lists the ISPs that we study, the number of gateways deployed in them, and the number of gateways that report more than 100 throughput measurements. Gateways are rolled out in phases. These devices perform measurements less aggressively when users are sending a lot of traffic. Therefore, not all gateways report data for the entire duration of the study. When we report averages and 95^{th} percentile values for some metric, we only consider gateways that have reported more than 100 measurements for that metric. We also only consider the eight ISPs with the most gateways.

Table 3 shows the active measurements that we use from the SamKnows deployment; some of these (*e.g.*, *last mile latency*) were inspired from our experience running them on BISMark. The gateways conduct upstream and downstream measurements to servers hosted at Measurement Lab [27] about once every two hours.

There are many ways to measure throughput, though there is no standard method. Bauer *et al.* list several notions of "broadband speed": *capacity* is the total carrying capacity of the link; and the *bulk transfer capacity* is the amount of data that can be transferred along a path with a congestion-aware protocol like TCP. In Section 5.1, we evaluate several methods for measuring these metrics.

ISP	Technology	SamKnows		BISMark
		Total	**Active**	Total
Comcast	Cable	864	560	4
AT&T	DSL/FTTN	787	335	10
TimeWarner	Cable	690	381	-
Verizon	DSL/FTTP	551	256	-
Cox	Cable	381	161	-
Qwest	DSL/FTTN	265	117	-
Charter	Cable	187	51	-
Cablevision	Cable	104	53	-

Table 2: The SamKnows and BISMark deployments. *Active* deployments are those that report more than 100 download throughput measurements over the course of our study.

The SamKnows gateways measure bulk transfer capacity using an HTTP client that spawns three parallel threads; this approach increases the likelihood of saturating the access link. The software first executes a "warmup" transfer until throughput is steady to ensure that the throughput measurements are not affected by TCP slow start. The download tests that follows use the same TCP connection to exploit the "warmed up" session. The tests last for about 30 seconds; the software reports snapshots of how many bytes were transferred for every five-second interval.

The gateways also measure different aspects of latency: (1) end-to-end latency; (2) latency to the first IP hop inside the ISP (*last mile latency*); and (3) latency coinciding with an upload or download (*latency under load*). They measure end-to-end latency in two ways: (1) Using a UDP client that sends about six hundred packets an hour to the servers and measures latency and packet loss, and (2) using ICMP ping to the same set of servers at the rate of five packets per hour. To measure latency under load, the gateway measures end-to-end latency during both the upload and the download measurements. They also measure jitter based on RFC 5481 [28] and the time to download the home page of ten popular websites. Before any test begins, the measurement software checks whether cross traffic on the outgoing interface exceeds 64 Kbits/s down or 32 Kbits/s up; if traffic exceeds this threshold, it aborts the test.

4.2.2 BISMark

BISMark comprises gateways in the home, a centralized management and data collection server, and several measurement servers. The gateway performs passive and active measurements and anonymizes the results before sending them back to the central repository for further analysis. This gateway also periodically "phones home" to allow the central repository to communicate back through network address translators, to update network configurations and to install software updates. The gateway is based on the NOX Box [31], a small-form-factor computer resembling an off-the-shelf home router/gateway. The NOX Box hardware is an ALIX 2D13 6-inch by 6-inch single board computer with a 500MHz AMD Geode processor, 256 MB of RAM and at least 2 GB of flash memory. The Nox Box runs Debian Linux.

Table 3 lists the measurements that BISMark collects.[1] We collect throughput, latency, packet loss, and jitter measurements.

BISMark measures *bulk transfer capacity* by performing an HTTP download and upload for 15 seconds using a single-threaded TCP connection once every 30 minutes, regardless of cross traffic. We do this to have more readings, and to account for cross-traffic, we count bytes transferred by reading directly from `/proc/net/dev`, and compute the "passive throughput" as the

[1]The data is available at `http://projectbismark.net/`.

Parameter	Type	Prot.	Freq.	Comments
SamKnows: 4,200 devices, 16 ISPs				
Latency	End-to-end	UDP	600 pkts/hr	MLab
	End-to-end	ICMP	5 pkts/hr	MLab
	Last-mile	ICMP	5 pkts/hr	First IP hop
	Upstream load	ICMP	2 hours	During upload
	Downstream load	ICMP	2 hours	During download
Loss	End-to-end	UDP	600 pkts/hr	MLab
Downstream Throughput	Multi-threaded HTTP	TCP	2 hours	MLab, idle link
Upstream Throughput	Multi-threaded HTTP	TCP	2 hours	MLab, idle link
Jitter	Bi-directional	UDP	1 hour	500pkts/30sec
Web GET	HTTP	TCP	1 hour	Alexa sites
BISMark: 17 devices, 3 ISPs				
Latency	End-to-end	ICMP	5 min	Host
	Last-mile	ICMP	5 min	First IP hop
	Upstream load	ICMP	30 min	During upload
	Downstream load	ICMP	30 min	During download
Packet loss	End-to-end	UDP	15 min	D-ITG
Jitter	End-to-end	UDP	15 min	D-ITG
Downstream Throughput	Single-thread HTTP	TCP	30 min	curlget to Host
	Passive throughput	N/A	30 min	/proc/net/dev
	Capacity	UDP	12 hrs	ShaperProbe
Upstream Throughput	Single-thread HTTP	TCP	30 min	curlput to Host
	Passive throughput	N/A	30 min	/proc/net/dev
	Capacity	UDP	12 hrs	ShaperProbe

Table 3: Active measurements periodically collected by the SamKnows and BISMark deployments.

byte count after the HTTP transfer minus the byte count before the transfer, divided by the transfer time. This yields the combined throughput of the HTTP transfer and the cross traffic. To measure *capacity*, we run ShaperProbe [32] once every twelve hours to measure UDP capacity. The gateways measure end-to-end latency to a nearby wide-area host, last-mile latency, and latency-under load to the last-mile router. They also measure packet loss and jitter using the D-ITG tool [6]. The gateways perform each measurement at the frequency presented in Table 3 regardless of cross traffic. All measurements are synchronized to avoid overlapping towards the same measurement server.

In January 2011, the BISMark deployment had 14 gateways across AT&T (DSL) and Comcast (Cable), and two more on Clear (WiMax). We recruited users from our research lab and other graduate students in the computer science department; some users were recruited using a recruiting firm.[2] We only study AT&T and Comcast using BISMark. The AT&T users form the most diverse set of users in the current deployment, with five distinct service plans. We use data from the same period as the SamKnows study.

5. UNDERSTANDING THROUGHPUT

We study throughput measurements from both the SamKnows and BISMark deployments. We first explore how different mechanisms for measuring throughput can generate different results and offer guidelines on how to interpret them. We then investigate the throughput users achieve on different access links, the consistency of throughput obtained by users, and the factors that affect it. Finally, we explore the effects of ISP traffic shaping and the implications it holds for throughput measurement.

5.1 Interpreting Throughput Measurements

Users of access networks are often interested in the throughput that they receive on uploads or downloads, yet the notion of "throughput" can vary depending on how, when, and who is measuring it. For example, a sample run of `www.speedtest.net` in an author's home, where the service plan was 6Mbits/s down and 512Kbits/s up, reported a downlink speed of 4.4 Mbits/s and an uplink speed of 140 Kbits/s. Netalyzr reported 4.8 Mbits/s and 430 Kbits/s. Long-term measurements (from the SamKnows gateway deployed in that author's home) paint a different picture: the user achieves 5.6 Mbits/s down and 452 Kbits/s up. Both `www.speedtest.net` and Netalyzr measurements reflect transient network conditions, as well as other confounding factors. Users cannot complain to their ISPs based solely on these measurements. Although measuring throughput may seem straightforward, our results in this section demonstrate the extent to which different measurement methods can produce different results and, hence, may result in different conclusions about the ISP's performance.

[2]All of our methods have been approved by Georgia Tech's institutional review board.

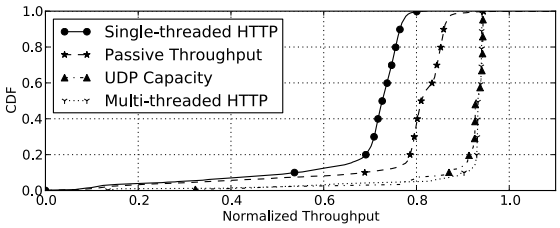


Figure 4: Comparison of various methods of measuring throughput. (SamKnows and BISMark)

We compare several methods for measuring upstream and downstream throughput from Table 3. We normalize the values of throughput by the service plan rates advertised by the ISP so that we can compare throughput across access links where users have different service plans.

Throughput measurement techniques—even commonly accepted ones—can yield variable results. We perform comparisons of throughput measurement techniques in two locations that have deployed both the SamKnows and BISMark gateways (we are restricted to two due to the logistical difficulty in deploying both gateways in the same location). In both cases, the ISP is AT&T, but the service plans are different (6 Mbits/s down and 512 Kbits/s up; and 3 Mbit/s down and 384 Kbits/s up). Figure 4 shows a CDF of the normalized throughput reported by the four methods we presented in Table 3. Each data point in the distribution represents a single throughput measurement by a client. A value of 1.0 on the x-axis indicates that the throughput matches the ISP's advertised rate. None of the four methods achieve that value. This could be due to many factors: the sync rate of the modem to the DSLAM; layer-2 framing overhead on the line; or overhead from the measurement techniques themselves. The throughput achieved by multiple parallel TCP sessions comes closer to achieving the advertised throughput. UDP measurements (obtained from ShaperProbe) also produce consistent measurements of throughput that are closer to the multi-threaded TCP measurement. A single-threaded TCP session may not be able to achieve the same throughput, but accounting for cross traffic with passive measurements can provide a better estimate of the actual achieved throughput.

The behavior of single-threaded TCP measurements varies for different access links. We compare the passive throughput for two BISMark users with the *same ISP and service plan* (AT&T; 3 Mbits/s down, 384 Kbits/s up) who live only a few blocks apart. Figure 5 shows that User 2 consistently sees nearly 20% more throughput—much closer to the advertised rate—than User 1. One possible explanation for this difference is the loss rates experienced by these two users; User 1 suffers more loss than User 2

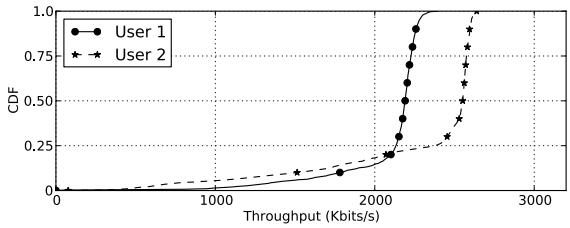


Figure 5: Users with the same service planbut different loss profiles see different performance. User 1 has higher loss and sees lower performance. (BISMark)

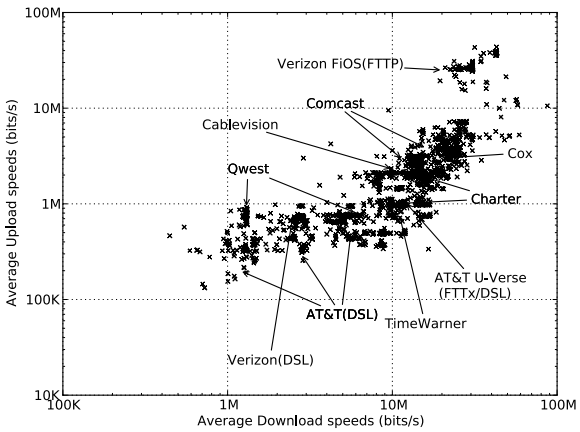


Figure 6: Average download rate versus the average upload rate obtained by individual users in the dataset. (SamKnows)

(0.78% vs. 0.20% on the downlink and 0.24% vs. 0.06% on the uplink). Their baseline latencies differ by about 16 milliseconds (8 ms vs. 24 ms). We confirmed from the respective modem portals that User 1 has interleaving disabled and that User 2 has interleaving enabled. Therefore, User 2 is able to recover from noisy access links that cause packet corruption or losses. Single-threaded downloads are more adversely affected by the loss rate on the access link than multi-threaded downloads (even when accounting for cross traffic); reducing the loss rate (*e.g.*, by interleaving) can improve the performance of a single-threaded download. For the rest of the paper, we consider only multi-threaded TCP throughput.

Takeaway: Different throughput measurement techniques capture different aspects of throughput. A single-threaded TCP session is sensitive to packet loss. Augmenting this measurement with passive usage measurements improves its accuracy. Multi-threaded TCP and the UDP capacity measurements measure the access link capacity more accurately and are more robust to loss.

5.2 Throughput Performance

We investigate the throughput obtained by users in the SamKnows deployment. We then study the consistency of their performance.

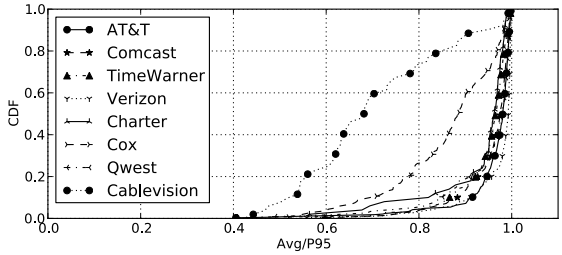
What performance do users achieve? Figure 6 shows the average download and upload speeds obtained by each user in the SamKnows dataset. Each point in the scatter plot shows the average performance obtained by a single user in the deployment. Clusters of points in the plot reveal common service plans of different ISPs, identified in the plot by labels. In general, these results agree with the findings from both Netalyzr [24] and Dischinger *et al.* [12], although our dataset also contains Verizon FiOS (FTTP) users that clearly stand out, as well as other more recent service offerings (*e.g.*, AT&T U-Verse). Although the statistics do show some noticeable clusters around various service plans, there appears to be considerable variation even within a single service plan. We seek to understand and characterize both the performance variations and their causes. We do not yet have access to the service plan information of each user, so we focus on how and why throughput performance varies, rather than whether the measured values actually match the rate corresponding to the service plan.

Do users achieve consistent performance? We analyze how consistently users in the SamKnows achieve their peak performance deployment using the $Avg/P95$ metric, which we define as the ratio of the average upload or download throughput obtained by a user to the 95^{th} percentile of the upload or download throughput value obtained by the same user. Higher values for these ratios reflect that users' upload and download rates that are more consistently close to the highest rates that they achieve; lower values indicate that user performance fluctuates.

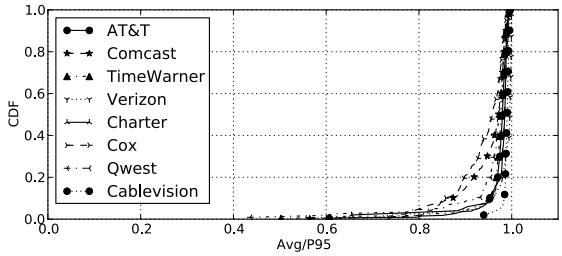
Figure 7a shows the CDF of the $\text{Avg}/P95$ metric for each user; Figure 7b shows the same metric for uploads. Most users obtain throughput that is close to their 95^{th} percentile value. Users of certain ISPs (*e.g.*, Cox, Cablevision) experience average download throughput that is significantly less than their 95th percentile. (Both ISPs have more than 50 active users in our data set; see Table 2). Upload throughput performance is more consistent across ISPs. The big difference between download rates and upload rates for popular service plans could account for the fact that upstream rates are more consistent than downstream rates. We also studied the $\text{Median}/P95$ performance; which is similar to $\text{Avg}/P95$, and so we do not show them. Our results suggest that upload and download throughput are more consistent than they were when Dischinger *et al.* performed a similar study few years ago [12], especially for some cable providers.

Why is performance sometimes inconsistent? One possible explanation for inconsistent download performance is that the access link may exhibit different performance characteristics depending on time of day. Figure 8a shows the $\text{Avg}/P95$ metric across the time of day. We obtain the average measurement reported by each user at that particular time of day and normalize it with the 95^{th} percentile value of that user over all reports. Cablevision users see, on average, a 40% drop in performance from early morning and evening time (when users are likely to be home). For Cox, this number is about 20%. As the figure shows, this effect exists for other ISPs to a lesser extent, confirming prior findings [12]. Because we do not know the service plan for each user, we cannot say whether the decrease in performance for Cox and Cablevision represents a drop below the service plans for those users (*e.g.*, these users might see rates higher than their plan during off-peak hours). Figure 8b shows how the standard deviation of normalized throughput varies depending on the time of day. Performance variability increases for *all* ISPs during peak hours. Figure 8c shows the loss behavior for different times of day; although most ISPs do not see an increase in loss rates during peak hours, Cox does. This behavior suggests that some access ISPs may be under-provisioned; those ISPs for which users experience poor performance during peak hours may be experiencing congestion, or they may be explicitly throttling user traffic during peak hours.

Takeaway: Although there is no significant decrease in performance during peak hours, there is significant variation. A one-time



(a) Download throughput is mostly consistent, with some exceptions.



(b) Upload throughput is consistent across ISPs.

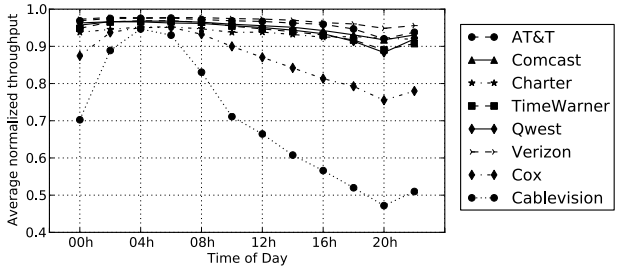
Figure 7: Consistency of throughput performance: The average throughput of each user is normalized by the 95^{th} percentile value obtained by that user. (SamKnows)

"speed test" measurement taken at the wrong time could likely report misleading numbers that do not have much bearing on the long-term performance.

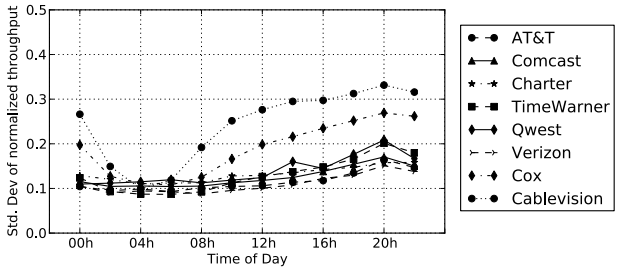
5.3 Effect of Traffic Shaping on Throughput

ISPs shape traffic in different ways, which makes it difficult to compare measurements across ISPs, and sometimes even across users within the same ISP. We study the effect of PowerBoost 3 across different ISPs, time, and users. We also explore how Comcast implements PowerBoost.

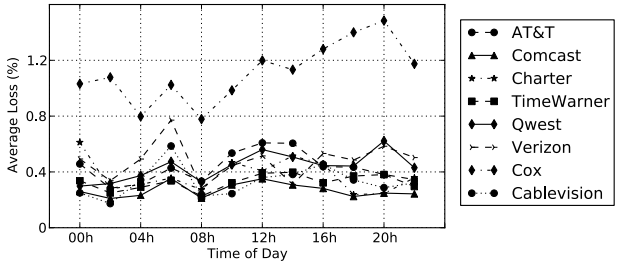
Which ISPs use PowerBoost, and how does it vary across ISPs? The SamKnows deployment performs throughput measurements once every two hours; each measurement lasts 30 seconds, and each report is divided into six snapshots at roughly 5-second intervals for the duration of the 30-second test (Section 4). This measurement approach allows us to see the progress of each throughput measurement over time; if PowerBoost is applied, then the throughput during the last snapshot will be less than the throughput during the first. For each report, we normalize the throughput in each period by the throughput reported for the first period. Without PowerBoost, we would expect that the normalized ratio would be close to one for all intervals. On the other hand, with PowerBoost, we expect the throughput in the last five seconds to be less than the throughput in the first five seconds (assuming that PowerBoost lasts less than 30 seconds, the duration of the test). Figure 9 shows the average progression of throughput over all users in an ISP: the average normalized throughput decreases steadily. We conclude that most cable ISPs provide some level of PowerBoost for less than 30 seconds, at a rate of about 50% more than the normal rate. Cablevision's line is flat; this suggests that either it does not provide PowerBoost, or it lasts well over 30 seconds consistently, in which case the throughput test would see only the PowerBoost effect. The



(a) The biggest difference between peak and worst performance is about 40%.



(b) The standard deviation of throughput measurements increases during peak hours, most significantly for ISPs that see lower throughputs at peak hours.



(c) Loss increases during peak hours for Cox. Other ISPs do not see this effect as much.

Figure 8: Time of day is significant: The average download throughput for Cablevision and Cox users drops significantly during the evening peak time. Throughput is also significantly more variable during peak time. (SamKnows)

gradual decrease, rather than an abrupt decrease, could be because PowerBoost durations vary across users or that the ISP changes PowerBoost parameters based on network state. From a similar analysis for uploads (not shown), we saw that only Comcast and Cox seem to provide PowerBoost for uploads; we observed a decrease in throughput of about 20%. Dischinger *et al.* [12] also reported PowerBoost effects, and we also see that it is widespread among cable ISPs. For the DSL ISPs (not shown), the lines are flat.

Takeaway: Many cable ISPs implement PowerBoost, which could distort speedtest-like measurements. While some people may be only interested in short-term burst rates, others may be more interested in long-term rates. Any throughput benchmark should aim to characterize both burst rates and steady-state throughput rates.

Do different users see different PowerBoost effects? Using BISMark, we study Comcast's use of PowerBoost in depth. According to Comcast [9], their implementation of PowerBoost provides

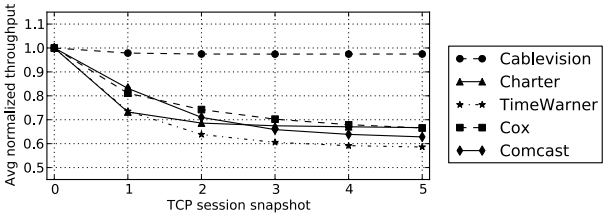
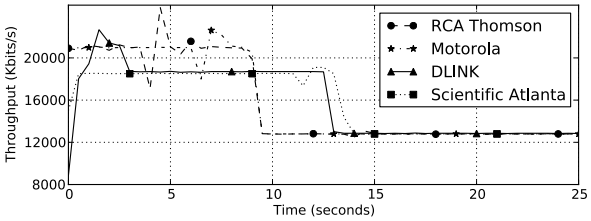


Figure 9: The average throughput obtained during the course of the measurement goes down significantly for the ISPs that enable PowerBoost. (SamKnows)

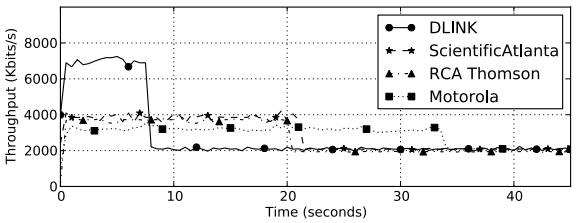
higher throughput for the first 10 MBytes of a download and the first 5 MBytes of an upload. We measure the shaped throughput for download and upload at the receiver using `tcpdump`. Because our tests are intrusive, we conducted them only a few times; however the results do not vary with choice of traffic generators or ports.

Figure 10 shows the observed throughput for four users for both download and uploads. All four users see PowerBoost effects, but, surprisingly, we see many different profiles even in such a small subset of users. Figure 10a shows download profiles for each user (identified by the modem they use; while the modem doesn't have an effect on burst rates, it does have an effect on buffering latencies as we show in Section 6). The user with a D-LINK modem sees a peak rate of about 21 Mbits/s for 3 seconds, 18.5 Mbits/s for a further ten seconds, and a steady-state rate of 12.5 Mbits/s. The Motorola user sees a peak rate of 21 Mbits/s for about 8 seconds. The PowerBoost technology [10] provides token buckets working on both packet and data rates; it also allows for dynamic bucket sizes. The D-LINK profile can be modeled as a cascaded filter with rates of 18.5 Mbits/s and 12.5 Mbits/s, and buffer sizes of 10MBytes and 1Mbyte respectively, with the line capacity being 21Mbits/s. We see varying profiles for uploads as well, although we only see evidence of single token buckets (Figure 10b). The D-LINK user sees about 7 Mbits/s for 8 seconds, Scientific Atlanta and Thomson users see about 4 Mbits/s for 20 seconds, and the Motorola user sees about 3.5Mbits/s for nearly 35 seconds. Because our results do not vary with respect to the packet size, we conclude that Comcast does not currently apply buckets based on packet rates.

Takeaway: Depending on how throughput measurements are conducted and how long they last, the measurements across users may vary considerably. Specifically, any speedtest measurement that lasts less than 35 seconds will *only* capture the effects of PowerBoost in some cases, and any short-term throughput measurement may be biased by PowerBoost rates.



(a) PowerBoost download behavior for 4 users.



(b) PowerBoost upload behavior for 4 users.

Figure 10: The level and duration of the burstiness is different for users with different modems, suggesting different shaping mechanisms or parameters. (BISMark)

6. UNDERSTANDING LATENCY

We show how latency can drastically affect performance, even on ISP service plans with high throughput. We then study how various factors ranging from the user's modem to ISP traffic shaping policies can affect latency.

6.1 How (and Why) to Measure Latency

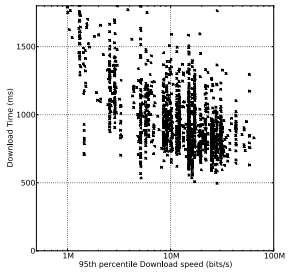
Latency affects the performance that users experience. It not only affects the throughput that users achieve, it also affects *perceived* performance: on a connection with high latency, various operations ranging from resolving DNS queries to rendering content may simply take longer.

Although latency appears to be a straightforward characteristic to measure, arriving at the appropriate metric is a subtle challenge because our goal is to isolate the performance of the access link from the performance of the end-to-end path. End-to-end latency between endpoints is a common metric in network measurement, but it reflects the delay that a user experiences along a wide-area path. We use two metrics that are more appropriate for access networks.

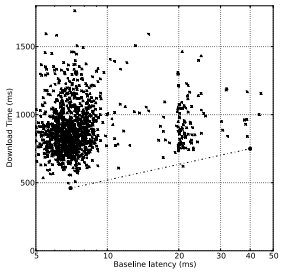
The first metric is the *last-mile latency*, which is the latency to the first hop inside the ISP's network. This metric captures the latency of the access link, which could affect gaming or short downloads. We measure last-mile latency in both of our deployments. As we show in this section, the last-mile latency is often a dominant factor in determining the end-user performance. The second metric we define is *latency under load*, which is the latency that a user experiences during an upload or download (*i.e.*, when the link is saturated in either direction). For BISMark, we measure the last-mile latency under load; on the SamKnows platform, we measure latency under load on the end-to-end path.

To investigate the effect of latency on performance, we measured how the time to download popular Web pages varies for users with different throughput and latency. Figure 11 shows the download time for `www.facebook.com` and how it varies by both the user's throughput and baseline last-mile latency. Figure 11a plots the 95^{th} percentile of each user's downstream throughput versus the average time it takes to download all objects from `www.facebook.com`. The average size of the download is 125 KByte. As expected, the download times decrease as throughput increases; interestingly, there is negligible improvement beyond a rate of 6 Mbits/s. Figure 11b plots download time against the baseline latency for all users whose downstream throughput (95^{th} percentile) exceeds 6 Mbits/s. Minimum download times increase by about 50% when baseline latencies increase from 10 ms to 40 ms. The fact that this effect is so pronounced, even for small downloads, underscores importance of baseline latency.

We investigate the effects of cable and DSL access-link technologies on last-mile latency, packet loss, and jitter. We also explore how different DSL modem configurations, such as whether the modem has interleaving enabled, affects last-mile latency and loss. Finally, we study the effect of modem hardware on perfor-



(a) Fetch time stabilizes above 6Mbits/s.



(b) Latency affects fetch times.

Figure 11: Effect of downstream throughput and baseline latency on fetch time from `facebook.com`. (SamKnows)

mance. Specifically, we investigate how oversized modem buffers that has recently received much attention from both operators and users [16]—affects interactivity and throughput.

ISP	Last mile latency		Loss	
	Average	Std. dev	Avg(%)	Std. dev
AT&T	25.23	33.47	0.48%	3.59
Comcast	10.36	14.49	0.27%	2.79
TimeWarner	11.87	25.18	0.33%	3.09
Verizon	12.41	20.60	0.51%	4.07
Charter	11.87	11.80	0.43%	3.29
Cox	13.88	28.02	1.11%	8.35
Qwest	39.42	32.27	0.33%	3.38
Cablevision	10.21	7.52	0.33%	3.14

Table 4: Last-mile latency and variation is significant; Variation in loss is high, suggesting bursty losses. (SamKnows)

ISP	Downstream		Upstream	
	Average	Std. dev	Average	Std. dev
AT&T	1.85	7.63	3.02	12.92
Comcast	1.15	6.37	3.24	6.60
TimeWarner	1.68	3.35	3.67	12.52
Verizon	1.71	5.01	1.97	4.82
Charter	1.17	1.66	2.66	7.48
Cox	1.18	1.89	4.27	7.10
Qwest	3.04	12.59	2.16	10.95
Cablevision	1.69	3.52	2.25	1.18

Table 5: Downstream jitter is quite low, however upstream jitter is significant. (SamKnows)

6.2 Last-Mile Latency

We obtain the last-mile latency by running `traceroute` to a wide-area destination and extracting the first IP address along the path that is not a NAT address. Note that we are measuring the latency to the first network-layer hop, which may not in fact be the DSLAM or the CMTS, since some ISPs have layer-two DSLAMs that are not visible in traceroute. This should not be problematic, since the latency between hops inside an ISP is typically much smaller than the last-mile latency.

How does access technology affect last-mile latency? Table 4 shows the average last-mile latency experienced by users in the ISPs included in our study. Last-mile latency is generally quite high, varying from about 10 ms to nearly 40 ms (ranging from $40 - -80\%$ of the end-to-end path latency). Variance is also high. One might expect that variance would be lower for DSL, since it is not a shared medium like cable. Surprisingly, the opposite is true: AT&T and Verizon have high variance compared to the mean. Qwest also has high variance, though it is a smaller fraction of the mean. To understand this variance, we divide different users in each ISP according to their baseline latency, as shown in Figure 12 Most users of cable ISPs are in the 0–10 ms interval. On the other hand, a significant proportion of DSL users have baseline last-mile latencies more than 20 ms, with some users seeing last-mile latencies as high as 50 to 60 ms. Based on discussions with network operators, we believe DSL companies may be enabling an interleaved local loop for these users.

Table 4 shows loss rates for users across ISPs. The average loss is small, but variance is high for all ISPs, suggesting bursty loss. Jitter has similar characteristics, as shown in Table 5; while the average jitter is low, the variation is high, especially on the upstream, also suggesting burstiness.

How does interleaving affect last-mile latency? ISPs enable interleaving for three main reasons: (1) the user is far from the DSLAM; (2) the user has a poor quality link to the DSLAM; or (3) the user subscribes to "triple play" services. An interleaved last-mile data path increases robustness to line noise at the cost of higher latency. The cost varies between two to four times the baseline latency.

Takeaway: Cable providers in general have lower last-mile latency and jitter. Baseline latencies for DSL users may vary significantly based on physical factors such as distance to the DSLAM or line quality.

6.3 Latency Under Load

We turn our attention to a problem that has gathered much interest recently because of its performance implications: modem buffering under load conditions [16]. We confirm that excessive buffering is a widespread problem afflicting most ISPs (and the equipment they provide). We profile different modems to study how the problem affects each of them. We also see the possible effect of ISP policies such as active queue and buffer management on latency and loss. Finally we explore exploiting shaping mechanisms such as PowerBoost might help mitigate the problem.

Problem: Oversized buffers. Buffers on DSL and cable modems are too large. Buffers do perform an important role: they absorb bursty traffic and enable smooth outflow at the configured rate [24]. Buffering only affects latency during periods when the access link is loaded, but during such periods, packets can see substantial delays as they queue up in the buffer. The capacity of the uplink also affects the latency introduced by buffering. Given a fixed buffer size, queuing delay will be lower for access links with higher capacities because the draining rate for such buffers is higher. We study the effect of buffering on access links by measuring latency when the access link is saturated, under the assumption that the last-mile is the bottleneck. We also present a simple model for modem buffering and use emulation to verify its accuracy.

How widespread are oversized buffers? Figure 13 shows the average ratios of latency under load to baseline latency for each user across different ISPs for the SamKnows data. The histogram shows the latencies when the uplink and the downlink are saturated separately. This figure confirms that oversized buffers affect users across all ISPs, though in differing intensity. The factor of increase when the uplink is saturated is much higher than when the downlink

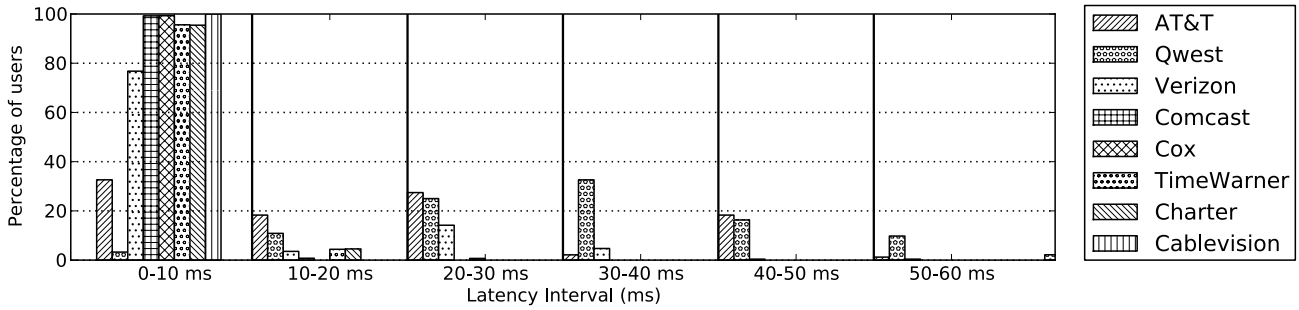


Figure 12: The baseline last mile latency for each user is computed as the 10^{th} percentile of the last mile latency. Most users see latencies less than 10 ms, but there are a significant number of users with the last mile latency greater than 10 ms. (SamKnows)

is saturated. One plausible explanation is that the downlink usually has more capacity than the uplink, so buffering on the ISP side is lower. The home network (at least 10 Mbits/s) is also probably better provisioned than the downlink, so there is minimal buffering in the modem for downstream traffic. The high variability in the latency under load can be partly explained by the variety in service plans; for instance, AT&T offers plans ranging from 768 Kbits/s to 6 Mbits/s for DSL and up to 18 Mbits/s for UVerse and from 128 Kbits/s to more than 1 Mbit/s for upstream. In contrast, Comcast offers fewer service plans, which makes it easier to design a device that works well for all service plans.

How does modem buffering affect latency under load? To study the effects of modem buffers on latency under load, we conduct tests on AT&T and Comcast modems using BISMark. We ran tests on the best AT&T DSL (6 Mbits/s down; 512 Kbits/s up) and Comcast (12.5 Mbits/s down; 2 Mbits/s up) plans. We perform the following experiment: we start ICMP `ping` (at the rate of 10 pkts/s for Comcast and 2 pkts/s for AT&T as some modems were blocking higher rates) to the last mile hop. After 30 seconds, we flood the uplink (at 1 Mbits/s for AT&T and at 10 Mbits/s for Comcast using `iperf` UDP). After 60 seconds, we stop `iperf`, but let `ping` continue for another 30 seconds. The `ping` measurements 30 seconds on either side of the `iperf` test establishes baseline latency. The Motorola and the 2Wire modems were brand new, while the Westell modem is about 5 years old, and was in place at the home where we conducted the experiment. We also saw the same Westell modem in two other homes in the BISMark deployment.

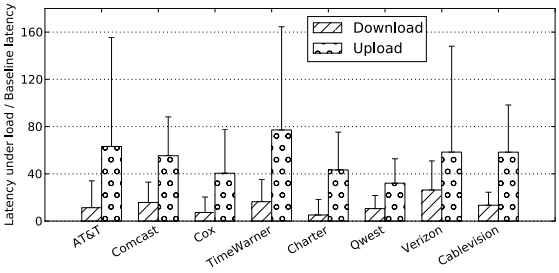
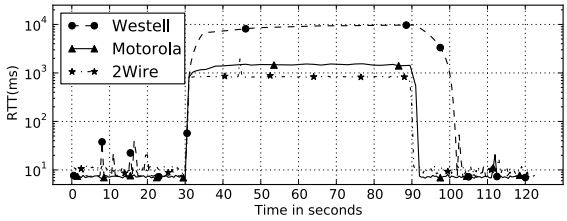


Figure 13: Latency under load: the factor by which baseline latency goes up when the upstream or the downstream is busy. The high ratios translate to significant real latencies, often in the order of seconds. (SamKnows)

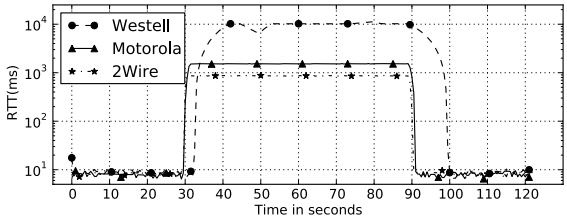
Figure 14a shows the latency under load for the three modems. In all cases, the latency increases dramatically at the start of the flooding and plateaus when the buffer is saturated. The delay experienced by packets at this stage indicates the size of the buffer, since we know the uplink draining rate. Surprisingly, we see more than an order of magnitude of difference between modems. The 2Wire modem has the lowest worst case latency, of 800 ms. Motorola's is about 1600 ms, while the Westell has a worst case latency of more than 10 seconds. Because modems are usually the same across service plans, we expect that this problem may be even worse for users with slower plans.

To model the effects of modem buffering, we emulated this setup in Emulab [13] with a 2 end-host, 1-router graph. We configured a token bucket filter using `tc`. We compute the buffer size to be: 512 Kbits/s$\times$max(latency of modem), which yields a size of 640 Kbytes for Westell, 100 Kbytes for Motorola, and 55 Kbytes for 2Wire. This simple setup almost perfectly captures the latency profile that the actual modems exhibit. Figure 14b shows the emulated latencies. Interestingly, we observed little difference in throughput for the three buffer sizes. We also emulated other buffer sizes. For a 512 Kbits/s uplink, we observed that the modem buffers exceeding 20 KBytes do little for throughput, but cause a linear increase in latency under load. Thus, the buffer sizes in all three modems are too large for the uplink.

How does PowerBoost traffic shaping affect latency under load? To understand latency under load for cable users, we study the Comcast users from BISMark. All of the modems we study have buffers that induce less than one second of delay, but these users see surprising latency under load profiles due to traffic shaping. Figures 15a and 15b show the latency under load for two Comcast users. The other two Comcast users (with the Scientific Atlanta and the Motorola modems) had latency profiles similar to the user with the Thomson modem, so we do not show them. The difference in the two latency profiles is interesting; the D-LINK user sees a jump in latency when the flooding begins and about 8 seconds later, *another increase* in latency. The Thomson user sees an initial increase in latency when flooding starts but then a *decrease* in latency after about 20 seconds. The first effect is consistent with buffering and PowerBoost. Packets see lower latencies during PowerBoost because, for a fixed buffer, the latency is inversely proportional to the draining rate. The increase in latency due to PowerBoost (from 200 ms to 700 ms) is proportional to the decrease in the draining rate (from 7 Mbits/s to 2 Mbits/s, as shown in Figure 10b). The decrease in latency for the Thomson user cannot be explained in the same way. Figure 15 shows the average loss rates alongside the latencies for the two users; interestingly for the user with the Thomson modem, the loss rate is low for about 20 seconds after the link is saturated, but there is a sharp hike in loss corresponding



(a) *Empirical measurements* of modem buffering. Different modems have different buffer sizes, leading to wide disparities in observed latencies when the upstream link is busy. (BISMark)



(b) *Emulated modems* with token bucket filters. We see similar latency progression. Emulated buffer sizes have minimal effect on throughput.

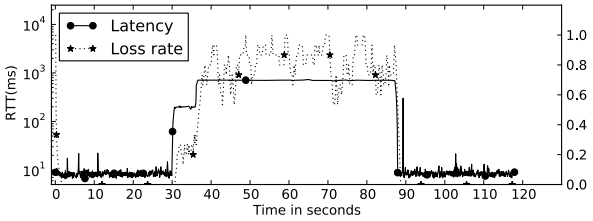
Figure 14: Buffering in AT&T modems. There is little benefit to the buffering seen in most modems.

to the drop in latency. This behavior may correspond to dynamic buffer sizing, as discussed in Section 5.3.

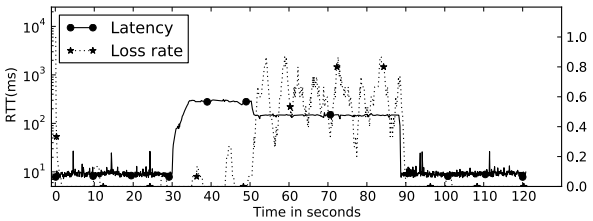
Can data transfer be modified to improve latency under load? We explore whether a user can modify their data transfer behavior so that large "bulk" flows and delay-sensitive flows can co-exist without interfering with one another. We compare the impact of a 50 MByte download on a G.711 VoIP call in three different conditions: (1) not applying any traffic control, (2) intermittent traffic at capacity on 10.8 seconds ON and 5.3 seconds OFF cycle, and (3) shaping using the WonderShaper [36] approach. Figure 16 shows the result of this experiment. In (1), the transfer takes 25.3 seconds; however, just after the PowerBoost period, the VoIP call starts suffering high latency and loss until the end of the transfer. In (2), traffic is sent in pulses, and the download takes 26.9 seconds. In (3), traffic is sent at just under the long term rate and the download takes 32.2 seconds. Both (2) and (3) do not increase latency significantly, this is because they do not deplete the tokens at any time, and therefore cause no queuing. In approach (2), the ON/OFF periods can be configured depending on the token bucket parameters,[3] and the size of the file to be transferred. Both approaches achieve similar long-term rates but yield significant latency benefit. The drawback is that any approach that exploits this behavior would need to know the shaping parameters.

Takeaway: Modem buffers are too large. Even the smallest buffers we see induce nearly one-second latency under load for AT&T and 300 ms for Comcast. Buffering is detrimental to both interactivity and throughput. Modifying data transfer behavior using short bursts or tools like WonderShaper might help mitigate the problem in the short term.

[3]If ρ_r is the rate we want to reserve for real-time applications, and ρ_t the token rate, the condition to be satisfied is: $(\rho_b + \rho_r - \rho_t) \times \tau_{on} \leq \tau_{off} \times (\rho_t - \rho_r)$, where ρ_b is the sending rate during the pulse, and τ_{on} and τ_{off} are the ON and the OFF times, respectively.



(a) Comcast user with D-LINK modem.



(b) Comcast user with RCA Thomson modem.

Figure 15: Possible effect of active buffer management: Loss rates increase when the latency drops. (BISMark)

7. LESSONS LEARNED

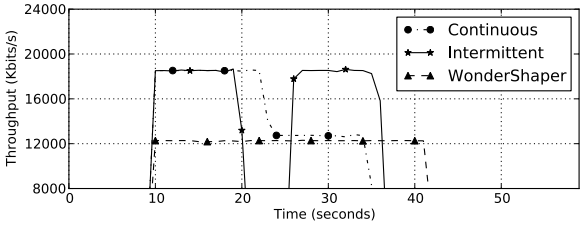
We conclude with some high-level lessons and suggestions for future research directions. One significant takeaway for users, policymakers, ISPs, and researchers is that *continual measurements, directly from home network gateways are crucial for understanding the details of home access network performance*. Existing "speed test" downloads and end-to-end latency measurements do not often reflect access network performance over an extended period of time, and they neglect various confounding factors on the host and within the home. Our ability to execute measurements directly, both from a small set of gateways where we can control the network conditions and measurements (BISMark) and a larger, more representative set of gateways across the United States (SamKnows), yields several lessons:

Lesson 1 (One Measurement Does Not Fit All) *Different ISPs use different policies and traffic shaping behaviors that make it difficult to compare measurements across ISPs.*

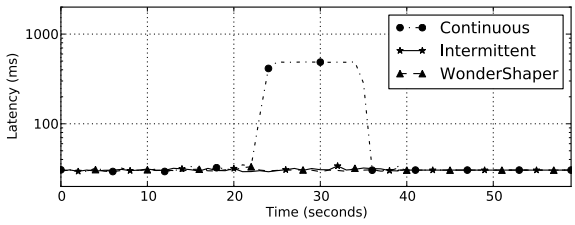
There is no single number that characterizes performance, or even throughput. Certain ISP practices such as PowerBoost can distort benchmarking measurements; ISPs might even design their networks so that widely used performance tests yield good performance. Developing a benchmarking suite for ISP performance that users can understand (*e.g.*, in terms of the applications they use) is critical; the measurements we develop in this paper may be a good starting point for that. Along these lines, more work is needed to understand the performance of specific applications, such as how video streaming performance compares across ISPs. The NetFlix study on ISP streaming performance [30] is a good start, but more such performance benchmarks are needed.

Lesson 2 (One ISP Does Not Fit All) *There is no "best" ISP for all users. Different users may prefer different ISPs depending on their usage profiles and how those ISPs perform along performance dimensions that matter to them.*

Different ISPs may be "better" along different performance dimen-



(a) Throughput.



(b) Latency.

Figure 16: It is possible to maintain low latency by modifying data transfer behavior. (BISMark)

sions, and the service plan that a user buys is only part of the picture. For example, we saw that, above a certain throughput, latency is the dominant factor in determining Web page loading time. Similarly, a gamer might be interested in low latency or jitter, while an avid file swapper may be more interested in high throughput. An imminent technical and usability challenge is to summarize access network performance data so that users can make informed choices about the service plans that are most appropriate for them (akin to a "performance nutrition label" [2]). Our recent work proposes some first steps in this direction [34].

Lesson 3 (Home Network Equipment Matters) *A user's home network infrastructure can significantly affect performance.*

Modems can introduce latency variations that are orders of magnitude more than the variations introduced by the ISP. Other effects inside the home that we have not yet studied, such as the wireless network, may also ultimately affect the user's experience. More research is needed to understand the characteristics of traffic inside the home and how it affects performance.

Acknowledgments

We thank the participants in the SamKnows and BISMark studies, and to Walter Johnston at the FCC for help and access to the data from the SamKnows study. We are grateful to Dave Täht for his continuing support of the BISMark platform. We also thank Sam Burnett and Hyojoon Kim for constructive comments on paper drafts. This project is supported by the the National Science Foundation through awards CNS-1059350, CNS-0643974, a generous Google Focus Grant, the European Community's Seventh Framework Programme (FP7/2007-2013) no. 258378 (FIGARO), and the ANR project C'MON.

REFERENCES

[1] Grenouille. http://www.grenouille.com/.
[2] Does broadband need its own government nutrition label? http://arstechnica.com/tech-policy/news/2009/10/does-broadband-needs-its-own-government-nutrition-label.ars, Oct. 2010. Ars Technica.
[3] C. Bastian, T. Klieber, J. Livingood, J. Mills, and R. Woundy. *Comcast's protocol-agnostisc congestion management system*. Internet Engineering Task Force, Dec. 2010. RFC 6057.
[4] G. Bernardi and M. K. Marina. Bsense: a system for enabling automated broadband census: short paper. In *Proc. of the 4th ACM Workshop on Networked Systems for Developing Regions (NSDR '10), June 2010.*, 2010.
[5] K. Bode. FCC: One Million Speedtests and Counting. http://www.dslreports.com/shownews/FCC-One-Million-Speedtests-And-Counting-109440, July 2010.
[6] A. Botta, A. Dainotti and A. Pescapé. Multi-protocol and multi-platform traffic generation and measurement. IEEE INFOCOM, Demo session, May 2007.
[7] R. Carlson. Network Diagnostic Tool. http://e2epi.internet2.edu/ndt/.
[8] K. Cho, K. Fukuda, H. Esaki, and A. Kato. The impact and implications of the growth in residential user-to-user traffic. In *ACM SIGCOMM 2006*, 2006.
[9] Comcast FAQ. http://customer.comcast.com/Pages/FAQViewer.aspx?Guid=024f23d4-c316-4a58-89f6-f5f3f5dbdcf6, Oct. 2007.
[10] R. Compton, C.L. Woundy and J. Leddy. Method and packet-level device for traffic regulation in a data network. U.S. Patent 7,289,447 B2, Oct. 2007.
[11] D. Croce, T. En-Najjary, G. Urvoy-Keller, and E. Biersack. Capacity Estimation of ADSL links. In *CoNEXT*, 2008.
[12] M. Dischinger, A. Haeberlen, K. P. Gummadi, and S. Saroiu. Characterizing residential broadband networks. In *Proc. ACM SIGCOMM Internet Measurement Conference*, San Diego, CA, USA, Oct. 2007.
[13] Emulab. http://www.emulab.net/, 2006.
[14] M. M. et al. Network Path and Application Diagnosis. http://www.psc.edu/networking/projects/pathdiag/.
[15] National Broadband Plan. http://www.broadband.gov/.
[16] J. Gettys. Bufferbloat. http://www.bufferbloat.net/.
[17] Glasnost: Bringing Transparency to the Internet. http://broadband.mpi-sws.mpg.de/transparency.
[18] D. Han, A. Agarwala, D. G. Andersen, M. Kaminsky, K. Papagiannaki, and S. Seshan. Mark-and-sweep: Getting the inside scoop on neighborhood networks. In *Proc. Internet Measurement Conference*, Vouliagmeni, Greece, Oct. 2008.
[19] Internet World Stats. http://www.internetworldstats.com/dsl.htm.
[20] Asymmetric Digital Subscriber Line Transceivers. ITU-T G.992.1, 1999.
[21] Asymmetric Digital Subscriber Line (ADSL) Transceivers - Extended Bandwidth ADSL2 (ADSL2Plus). ITU-T G.992.5, 2003.
[22] Data-over-cable service interface specifications: Radio-frequency interface specification. ITU-T J.112, 2004.
[23] C. R. S. Jr. and G. F. Riley. Neti@home: A distributed approach to collecting end-to-end network performance measurements. In *the Passive and Active Measurement Conference (PAM)*, 2004.
[24] C. Kreibich, N. Weaver, B. Nechaev, and V. Paxson. Netalyzr: Illuminating the edge network. In *Proc. Internet Measurement Conference*, Melbourne, Australia, Nov. 2010.
[25] K. Lakshminarayanan and V. N. Padmanabhan. Some findings on the network performance of broadband hosts. In *Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement*, IMC '03, pages 45–50, New York, NY, USA, 2003. ACM.
[26] G. Maier, A. Feldmann, V. Paxson, and M. Allman. On dominant characteristics of residential broadband internet traffic. In *ACM Internet Measurement Conference*, 2009.
[27] Measurement Lab. http://measurementlab.net.
[28] A. Morton and B. Claise. *Packet Delay Variation Applicability Statement*. Internet Engineering Task Force, Mar. 2009. RFC 5481.
[29] Netalyzr. http://netalyzr.icsi.berkeley.edu/.
[30] NetFlix Performance on Top ISP Networks. http://techblog.netflix.com/2011/01/netflix-performance-on-top-isp-networks.html, Jan. 2011.
[31] NOX Box. http://noxrepo.org/manual/noxbox.html.
[32] ShaperProbe. http://www.cc.gatech.edu/~partha/diffprobe/shaperprobe.html.
[33] M. Siekkinen, D. Collange, G. Urvoy-Keller, and E. Biersack. Performance limitations of ADSL users: A case study. In *the Passive and Active Measurement Conference (PAM)*, 2007.
[34] S. Sundaresan, N. Feamster, R. Teixeira, A. Tang, K. Edwards, R. Grinter, M. Chetty, and W. de Donato. Helping users shop for isps with internet nutrition labels. In *ACM SIGCOMM Workshop on Home Networks*, 2011.
[35] D. Vorhaus. A New Way to Measure Broadband in America. http://blog.broadband.gov/?entryId=359987, Apr. 2010.
[36] WonderShaper. http://lartc.org/wondershaper/, 2002.

Real-time probing of available bandwidth in home networks

Archi Delphinanto[$,*], Ton Koonen[$], Frank den Hartog[*]

[$]Electrical Engineering Department
Eindhoven University of Technology
Eindhoven, The Netherlands

[*]TNO
Delft, The Netherlands

Abstract— **Prioritization of flows in a home network based on traffic classification is still no guarantee that enough bandwidth will be available between a content server and a client. Besides, such QoS technologies need to be supported by every device in the end-to-end path to be effective, which is relatively expensive for the owners of home networks. In any small-scale IP network, best-effort or QoS-enabled, at home or anywhere else, it is therefore preferable to diagnose the network in real time before admitting a new flow. In this article we analyze existing probing techniques, and demonstrate a new method to probe the available bandwidth between a server and a client in a heterogeneous IP-based home network. The tool works with existing end-user devices, is non-intrusive, has a short measurement time, does not require pre-knowledge of the link layer network topology, and is accurate enough to make decisions about the admission of high-throughput high-quality streams such as for IPTV services.**

I. QoS IN HOME NETWORKS

Home networks are becoming ever more heterogeneous. This means that a single network consists of many different physical- and link-layer technologies and topologies, interconnecting many different devices with each other and the Internet, enabling many different services (see e.g. [1] and references therein). The Internet Protocol (IP) suite is the main enabler for the required interoperability. Correspondingly, a growing amount of consumer electronics devices contain an IP stack. Unfortunately IP has only limited support for Quality of Service (QoS), necessary to support many different services concurrently in a single shared home network. It mainly concerns reducing or avoiding jitter that occurs when multiple traffic streams contend for bandwidth. Furthermore noticeable packet loss will happen when User Datagram Protocol (UDP) based services such as high-quality telephony and HDTV need to be supported. This is especially an issue for broadband service providers (see e.g. [2] and [3]).

Many additional QoS solutions for IP networks are available, but most of them operate on the principle of traffic classification, where each data packet is placed into a limited number of traffic classes, and each router on the network is configured to differentiate the traffic based on its class. These solutions have not gained large popularity in the home networking market place, because they need to be supported by every device in the end-to-end (e2e) path to be effective. This makes them relatively expensive for consumers with many non-depreciated devices: to enjoy QoS they have to buy new devices. Besides, current solutions are different for different layer-2 technologies. Intermediate translators would then be needed to guarantee e2e QoS in a heterogeneous path. Though implementations for this exist (see e.g. [4]), they are deemed to be too expensive for mass-scale application today. Finally, prioritization of flows is still no guarantee that enough bandwidth (here synonymously used for application-level data throughput) will be available between a server and a client.

Before admitting a new flow to the home network, or rather any small-scale IP network (best-effort or QoS-enabled), we therefore propose to diagnose the network in real time. The information can be used for admission control, have the content service pragmatically adapt its properties to the actual condition of the network, or report intermittent issues to the user and/or the service provider's remote service management server. A crucial part of a diagnosing tool is the real-time assessment of e2e available bandwidth between the relevant client and server in the network. Though many e2e speed test applications exist for the Internet [5], none of them fulfills all of the requirements for use in today's home networks. Amongst these requirements are the following. The tool must:

1. be easy to implement. It should work with as many as possible existing devices in the home without firmware upgrades. Preferably the tool should only require a simple software module added on the server side of the e2e path of a flow. For most use cases we can therefore assume the diagnosing application to be a service running on the home gateway, serving various clients in the home network, which only need to have a regular IP stack.
2. be non-intrusive. It should not disrupt other traffic in the home noticeably.
3. have a short measurement time, i.e. it should have a low convergence time from an end-user perspective, and it should be fast enough to react to major changes in the home network traffic pattern. We assume this to be in the order of a few seconds; fluctuations within this time frame may be dealt with by application-layer buffering, for instance.

4. not require pre-knowledge of the link-layer network topology. Home networks may contain link-layer technologies that are not standardized or widely known.
5. be accurate enough to make informed decisions about the admission of delay- and jitter-critical applications. In the case of IPTV and IP telephony that means an accuracy of ~1 Mbit/s and ~50 kbit/s respectively.

In this article we propose a new tool, "AvaiLabLe Bandwidth ESTimator (Allbest)", which fulfills all requirements above to the extent that it is a software upgrade of only the probe server, it injects less than ~ 1 Mbit/s of probe traffic, produces results with an accuracy better than ~ 1 Mbit/s within less than 10 s, and can be applied on any Ethernet-Wifi combined topology. This is the first diagnosing application that successfully applies probe round-trip-time (RTT) measurement to wireless LANs and does not assume any home network topology *a priori*. The tool is based on a new probing method that we developed, and which is described in the following section. In the second half of the article we detail our test bed, followed by our test results. We finish with conclusion and a few words on our current and future experiments.

II. PROBING HOME NETWORKS

A. Bottleneck capacity and available bandwidth

We follow [6] for defining capacity of a hop as the bit rate, measured at the IP layer, at which the hop can transfer Maximum Transmission Unit (MTU)-sized IP packets. Therefore, the capacity of an e2e path is the maximum IP-layer rate that the path can transfer from source to sink. In our work, we assume the MTU size to be 1500 Byte, of Ethernet v2 (RFC 1191). As a path may consist of several links, the minimum link capacity in the path determines the path capacity. This link is called the narrow link. In contrast, the tight link is the link in the path with the maximum capacity utilization. This is the link with the least available bandwidth due to crossing traffic, i.e. other traffic in the path considered for admission of a new stream. In many cases, the tight link is in the narrow link, and the link is then referred to as the bottleneck.

For measuring bottleneck bandwidths in Internet paths, two types of tools can be distinguished: packet-pair dispersion tools (also called Probe Gap Model, or PGM) and self-loading techniques. The latter probe the network with trains of packets [7] at an increasing rate, and thus rely on flooding the network. They therefore do not fulfill requirement 2. PGM techniques were first explored in [8], and send only a few packets, at the rate C of the bottleneck capacity or somewhat slower. This allows crossing traffic to get in between the probe packets and disperse them, i.e. increase the difference in arrival time.

An issue with PGM techniques is that C needs to be known *a priori*. From requirement 4 follows that the tool must be able to estimate the available bandwidth A without such pre-knowledge of the path. This means that C needs to be determined first, and the estimation of A becomes a two-step process. In [9] we proposed and validated a new method for determining C in heterogeneous home networks based on round-trip-time measurements of probe packets, and fulfilling all requirements listed in the previous section. However, we also learned that probing a wireless LAN with rate C may yield the correct average dispersion rate at the receiver when measuring in one direction, but will not if it needs to be derived from round-trip times. This is easiest understood by looking at the details of our capacity estimation method first.

B. Capacity estimation

Our capacity estimation method is based on the packet-pair dispersion technique, which is usually implemented by sending two packets back-to-back on the network, thus minimizing the chance that crossing traffic will disperse the packets. It is then the bottleneck that will delay the second packet with respect to the first. C can subsequently be calculated simply from the minimum dispersion D and the packet size L as $C = L/D$. One then should minimize the chance that crossing traffic increases or decreases the bottleneck dispersion of the packets further down the path. To do so, we perform a series of n packet-pair probes, assume the crossing traffic stochastic, and then calculate D from the minimum RTT of the first packet (RTT_1) of a probe pair and the minimum RTT of the second packet (RTT_2) of a probe pair. C is then given by

$$C = L / (\min_{i=1\ldots n}[RTT_2(i)] - \min_{i=1\ldots n}[RTT_1(i)]) \quad (1)$$

RTTs can be measured without adaptation of the client side by using MTU-sized Internet Control Message Protocol (ICMP) Ping probe packets, or by sending MTU-sized UDP packets to a non-activated port. The client then automatically generates reply packets, respectively ICMP Echo packets or ICMP Error packets (i.e. code 3 or "Destination port unreachable"). ICMP Error packets are much smaller than ICMP Echo packets and therefore experience hardly any delay on the way back to the probing sender/receiver, assuming that the return one-way capacity between the client and the server $C_{reverse}$ is not much smaller than the sought-after one-way capacity $C_{forward}$ between server and client. The final result is then a good measure for $C_{forward}$. For symmetric media we may also use ICMP Ping probing packets, and assume that the delay and dispersion is the same for both directions of travel. Equation (1) then yields $C/2$ rather than C.

Equation (1) allows us to avoid unwanted contention of probe packets in the wireless medium. Existing packet-pair dispersion techniques will not work in wireless media in round trip, because the reply packet of the first probe packet contends with the second probe packet on the air interface (see Fig. 1). Irrespective of which packet wins, the reply packet of the second probe packet will eventually arrive at the probing sender/receiver too late. PK2 is acting as crossing traffic to the reply packets; the method is basically self-disturbing. As a result, C will be underestimated.

To avoid this contention, we need to prevent the first reply packet from being put on the network. We achieved this (see Fig. 2) by sending a single packet with size 2xMTU, instead of two packets back-to-back. On the network, this packet will automatically be fragmented (and behave like two individual packets back-to-back), and only after defragmentation a single reply will be sent back by the client. This will provide us the correct RTT_2., i.e. the RTT_2 which is only delayed by bottleneck dispersion and not by additional contention. Because we are not directly measuring D, but separate RTTs (see (1)), we can find the correct RTT_1 by sending a different series of single probe packets, well separated from each other and with size MTU.

Contention of probe and reply packets is the main reason why conventional packet-pair probing is not suitable for determining A. We can neither probe back-to-back nor with rate C without creating extra delay caused by contention. Unfortunately we cannot solve the latter by using fragmentation, because that can only mimic back-to-back probing.

C. *Available bandwidth estimation*

A deep analysis of the various delays that constitute the RTTs observed during capacity estimation allowed us to make a good estimation of A also. In Fig. 3 a typical histogram is shown of RTT_1 that we measured in an IEEE 802.11b network with 1.5 Mbit/s crossing traffic. Besides a clear minimum value, the RTT undergoes two random effects: the random back-off mechanism of IEEE 802.11 (mostly at short additional delays) and the delay caused by queuing due to crossing traffic.

We assume that, for UDP probing, most of the random delay is experienced in the forward direction. This is justified by the fact that the reply packet is very small, and we assume that the queuing mechanism of the system is fair. The reply packet is therefore hindered relatively little by the crossing traffic. We further assume that any systemic delay in the network (for instance processing delay) is either negligible or canceled when subtracting RTT_1 from RTT_2 [10], and that the delay caused by random effects is mainly happening in the bottleneck. A is then given by

$$A = L / \left(\frac{L}{C} + \overline{d_r} \right) \qquad (2)$$

with L/C the delay in the bottleneck without crossing traffic following from (1), and $\overline{d_r}$ the average delay caused by random effects in the bottleneck. The latter can be derived from RTT_1 as

$$\overline{d_r} = \underset{i=1\ldots n}{\text{avg}} [RTT_1(i)] - \underset{i=1\ldots n}{\min} [RTT_1(i)]) \qquad (3)$$

III. ALLBEST TEST BED

The set-up of our test bed is schematically drawn in Fig. 4. The Allbest server runs on the “prober/receiver” computer, and probes the “mirror” via any heterogeneous topology of our interest. The results presented in this article were obtained by configuring a WLAN IEEE 802.11b or 802.11g with a Linksys WRT54GL v. 1.1 access point as a bottleneck link. We switched off the automatic rate adaptation and Clear to Send (CTS) protection mode, and run both networks on their maximum physical rates of 11 Mbit/s and 54 Mbit/s respectively. The measurements were carried out in a Faraday cage to avoid uncontrolled interference.

We benchmarked Allbest against the well-known testing tool Iperf, and against Wbest [11]. Wbest is the only other real-time probing tool we know which is applicable to wireless networks. It requires the wireless hop to be in the last link, because it needs to be sure that the probing packets arrive at the bottleneck with rate C. For the estimation of C it uses standard PGM and packet-pair dispersion. Both Wbest and Iperf need to be installed on both the prober/receiver (which for Wbest and Iperf just acts as a prober) and the mirror (which for Wbest and Iperf acts as a receiver). The prober/receiver and mirror are laptop computers with a 2.0 GHz processor. Allbest and Iperf run on Windows XP service pack 3, and Wbest runs on Linux UBUNTU 10.04. To maximize the performance of the software, other processes running in the computers’ background memory were switched off whenever possible.

Allbest basically consists of a home-built configurable UDP packet generator and a home-built configurable ICMP Ping packet generator, combined with Wireshark to measure high-precision RTTs. Any RTT > 2 x min[RTT(i)] is discarded, and we have verified that most of those long RTTs are caused by uncontrollable processing delay in the laptops due to other tasks of the operating system. A measurement takes 90 probe pairs and is repeated 6 times.

With Iperf we measured at which UDP injection rate which packet loss occurs with 1472 Byte payload per packet. The result is fitted linearly and the point where the fitted line crosses the transmission rate axes is interpreted as being the available bandwidth. Each Iperf measurement is set for 10 seconds with 1 second interval. The UDP packet loss is averaged over 8-10 similar measurements, leading to 1% standard deviation. Also Wbest was configured to use 1472 Byte UDP payload. Each measurement of 90 packet pairs was repeated 30 times. Like Allbest, Wbest filters and discards unreliable results.

Random UDP crossing traffic is generated with the Distributed Internet Traffic Generator (D-ITG), available from the Universita degli Studi di Napoli "Federico II". The UDP packets have uniformly distributed packet sizes (40-1472 Byte), and are sent at Poisson-distributed exponential time intervals. We distinguished crossing traffic and contending traffic, and follow [11] for their definitions.

IV. AVAILABLE BANDWIDTH IN ETHERNET-WIFI NETWORKS

We have obtained the available bandwidth for three different topologies with Iperf, Wbest as well as Allbest (in its UDP probing variety):

1. Prober/receiver→100BASE-TX→IEEE 802.11b→mirror
2. Prober/receiver→100BASE-TX→IEEE 802.11g→mirror
3. Prober/receiver→IEEE802.11g→100BASE-TX→mirror

For every topology we generated three different amounts of crossing traffic (X), at about 0%, 25% and 50% of the capacity, and 25% of contending traffic.

For topology 1, the results are summarized in Fig. 5a. For this topology, with IEEE 802.11b, all tools yield similar results at first sight and within the error margins. Allbest estimates the capacity C of 802.11b on 7.6±0.2 Mbit/s, which is equal to the theoretical value [9]. For all tools, the available bandwidth A is lower than C for $X = 0$. This is caused by the random back-off mechanism of WLAN. If the random back-off algorithm were to be active for all packets, we would expect $A = 6.4$ Mbit/s [9]. Since all tools estimate available bandwidths somewhat higher than that, we suspect that there are still many packets that do not undergo random back-off.

On the whole, Allbest seems to find larger available bandwidths than Iperf, and Wbest finds even larger ones. Even though Iperf is a well-known benchmarking tool, it is probably underestimating A in our experiments. Because the crossing traffic is stochastic, some packet loss will already be recorded at relatively low Iperf injection rates. For both Allbest and Wbest, the values for A at $X > 0$ are also closer than Iperf to the expected value of A (obtained by simply subtracting the crossing traffic rate X from the available bandwidths A at $X = 0$ Mbit/s). The error margins of the Allbest results are remarkably lower than the ones for Wbest, though we tried to have the results based on the same number of probes. We do not have an explanation for this yet.

For Allbest, the value for A at $X = 3$ Mbit/s was calculated by discarding any RTT > 3 x min[RTT(i)], rather than using the default cut-off time of 2 x min[RTT(i)], as stated in the previous section. We found that with the latter, too many packets had been discarded that were clearly delayed by crossing traffic, and A was grossly overestimated (5.1±0.2 Mbit/s). Many PGM techniques use a default cut-off time of 2 x min [RTT(i)]. This follows from their assumption of fair queuing congestion management in the router. This means that bottleneck delays can never be larger than $2L/C$, even if the utilization by crossing traffic is larger than 50% (which then just results in larger packet loss). Crossing traffic of 3 Mbit/s with randomly distributed time intervals will utilize the bottleneck more than 50% for at least part of the time. Fortunately, the actual congestion management mechanism of the router (most probably store and forward) still allowed us to capture relevant packets with larger RTTs and compute a realistic value for A.

For topology 2, the results are summarized in Fig. 5b. For this topology, with IEEE 802.11g, Allbest is showing a clear supremacy. Allbest estimates the capacity C of 802.11g on 38±2 Mbit/s, which is equal to the theoretical value [9]. The value of A at $X = 0$ Mbit/s is then expected to be 26 Mbit/s if the random back-off algorithm is active for all packets [9]. Iperf and Allbest estimate somewhat higher again, but Wbest significantly underestimates A, also for larger X. The inventors of Wbest warn for underestimation when the probe packets arrive at the bottleneck at a rate larger than C [11]. Surprisingly, Wbest's capacity estimation for topology 2 is quite good, namely 38 Mbit/s. The fact that Wbest arrived at plausible answers for A with topology 1 can be explained by it grossly overestimating the C of topology 1 (8.8 Mbit/s). The results for Allbest are very close to the ones for Iperf, and have the lowest error margins of all. But like with Fig. 5a, it is not sure whether Iperf yields the correct values. More than Iperf, Allbest yields values for A at $X > 0$ close to what one obtains by subtracting X from $A(X = 0)$.

For topology 3, the results are much the same as for topology 2 (see Fig. 5c). This shows that Allbest can function in either order of physical- and link-layer technologies. Unfortunately we could not get any UDP results from Iperf, because it cannot inject faster than about 10 Mbit/s when directly connected to an 802.11g network.

V. CONCLUSIONS AND FUTURE WORK

We achieved a breakthrough in available bandwidth probing of heterogeneous home networks by understanding and then solving the contention issues that PGMs traditionally had with wireless links in the e2e path. This allowed us to design a new probing method based on round-trip-time measurements, with low intrusiveness and short convergence time, and without the need for knowing the home network topology *a priori*. Our tool, Allbest, is accurate enough to make informed decisions about the admission of IPTV streams and the like, and gives the service provider not more information than strictly needed. We have built a prototype and a test bed, and our performance measurements indicate that Allbest works well and outperforms Iperf and Wbest for various topologies based on 100BASE-TX, IEEE 802.11b, and 802.11g, for up to 50% crossing traffic.

Our work is opening up a whole field of research related to diagnostics of heterogeneous home networks. Many different configurations and network parameters need to be investigated. New network technologies supporting IP are currently entering the home, such as HomePlug, MoCa, IEEE 1901, IEEE 802.11n, and G.hn. Some of them (e.g. HomePlug) are exhibiting very different physical- and link-layer properties (such as fast rate adaptation) from the networks studied in this article. Our method needs to be improved to include these novel techniques. Also different queuing mechanisms than fair queuing should be studied. To increase the accuracy of our method up to a level that it can be used for Voice-over-IP services, network tomography techniques may be applied.

Another matter is how the obtained values should lead to intelligent decisions on the service level, which probably needs some form of cross-layer optimization. One of the questions following from this is how frequent a measurement should be repeated. It

will for sure depend on the dynamics of the traffic in the home network. It can be safely assumed that crossing traffic in homes cannot be modeled with the stochastic properties of Internet traffic. We are currently performing a series of experiments in Dutch homes in order to understand the in-home traffic dynamics.

Finally, the applicability of our method to other consumer networks should be studied. In-car networks, personal area networks, hotel networks, etc., exhibit similar properties and management issues to home networks. At the IEEE CCNC 2011 conference, operators also showed interest in using Allbest for mobile access networks.

ACKNOWLEDGMENT

This work was partly done in the collaborative Integrated Project FIGARO, which is supported by the European Commission under the 7th Framework Program, Grant Agreement Number 258378.

REFERENCES

[1] F.T.H. den Hartog *et al*, "First experiences with Personal Networks as an enabling platform for service providers", In Proceedings of the 4th Annual International Conference on Mobile and Ubiquitous Systems: Networking & Services (MobiQuitous '07), Philadelphia (USA), August 2007, pp. 1-8.

[2] Home Gateway Initiative, "HGI Guideline Document: QoS white paper", HGI-GD013-R2, 2009, www.homegateway.org.

[3] Chris Develder *et al*, "Delivering scalable video with QoS to the home", Telecommunication Systems, Springer, Online First™, 9 June 2010.

[4] Dan Jiang and Mo Li; "Quality of Service in the home network", In Proceedings of the 2nd International Conference on Future Generation Communication and Networking (FGCN '08), Hainan Island (PR China), December 2008, pp. 473-476.

[5] E. Goldoni and M. Schivi, "End-to-end available bandwidth estimation tools, an experimental comparison", Traffic Monitoring and Analysis, Lecture Notes in Computer Science, Springer, vol. 6003, 2010, pp. 171-182.

[6] R. Prasad, C. Dovrolis, M. Murray, and K. Claffy, "Bandwidth estimation: metrics, measurement techniques and tools", IEEE Network, vol. 17 (6), 2003, pp. 27—35.

[7] B. Melander, M. Bjorkman, and P. Gunninberg, "A new end - to - end probing and analysis method for estimating bandwidth bottlenecks", In Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM '00), San Francisco (USA), November 2000, vol. 1, pp. 415-420.

[8] V. Jacobson, "Congestion avoidance and control",. In Proceedings of the ACM Symposium on Communication Architectures and Protocols (SIGCOMM '88), Stanford (USA), August 1988, pp. 314-329.

[9] A. Delphinanto, T. Koonen, S. Zhang, and F. den Hartog, "Path Capacity Estimation in Heterogeneous, Best-effort, Small-scale IP Networks", In Proceedings of the 35th IEEE Conference on Local Computer Networks (LCN 2010), Denver, CO (USA), October 2010.

[10] W. Kampichler, and K.M. Goeschka, "On measuring Quality of Service limitations in Local Area Networks", In Proceedings of the 2003 IEEE International Conference on Communications (ICC '03), Anchorage (USA), May 2003, vol. 1., pp. 291—295.

[11] M. Li, M. Claypool, and R. Kinicki, "WBest: A bandwidth estimation tool for IEEE 802.11 wireless networks", In Proceedings of the 33rd IEEE Conference on Local Computer Networks (LCN 2008), Montreal (Canada), October 2008, pp. 374—381.

Archi Delphinanto received his B.Sc. in engineering physics from Bandung Institute of Technology, Indonesia, in 1998, and his PD.Eng. in ICT from Eindhoven University of Technology, the Netherlands, in 2003. He currently works as scientist at TNO, the Netherlands, and is also pursuing his PhD degree in electrical engineering department, Eindhoven University of Technology. His research interests focus on service discovery and access in heterogeneous networks, including interoperability, quality of service, and service monitoring. He is a member of IEEE.

Ton Koonen Ton Koonen is a full professor at Eindhoven University of Technology (TU/e) since 2001, and chairman of the group Electro-Optical Communication Systems. Before joining TU/e, he worked for more than 20 years in applied research in industry, a.o. at Bell Laboratories in Lucent Technologies, and as a part-time professor at Twente University. His current research interests are in optical fiber access and inbuilding networks, including radio-over-fibre techniques. Professor Koonen is a Bell Labs Fellow, and a Fellow of IEEE.

Frank den Hartog is Senior Scientist Future Internet at TNO, and deputy chair of the Technical Working Group of the worldwide Home Gateway Initiative (HGI). Between 1998 and 2003 he was Applied Scientist at KPN Research, where he pioneered the home networking research area. He now acquires and manages large collaborative research projects on every aspect of heterogeneous consumer networking. He authored over 100 conference articles, journal papers, patents and contributions to standardization, and is a member of IEEE.

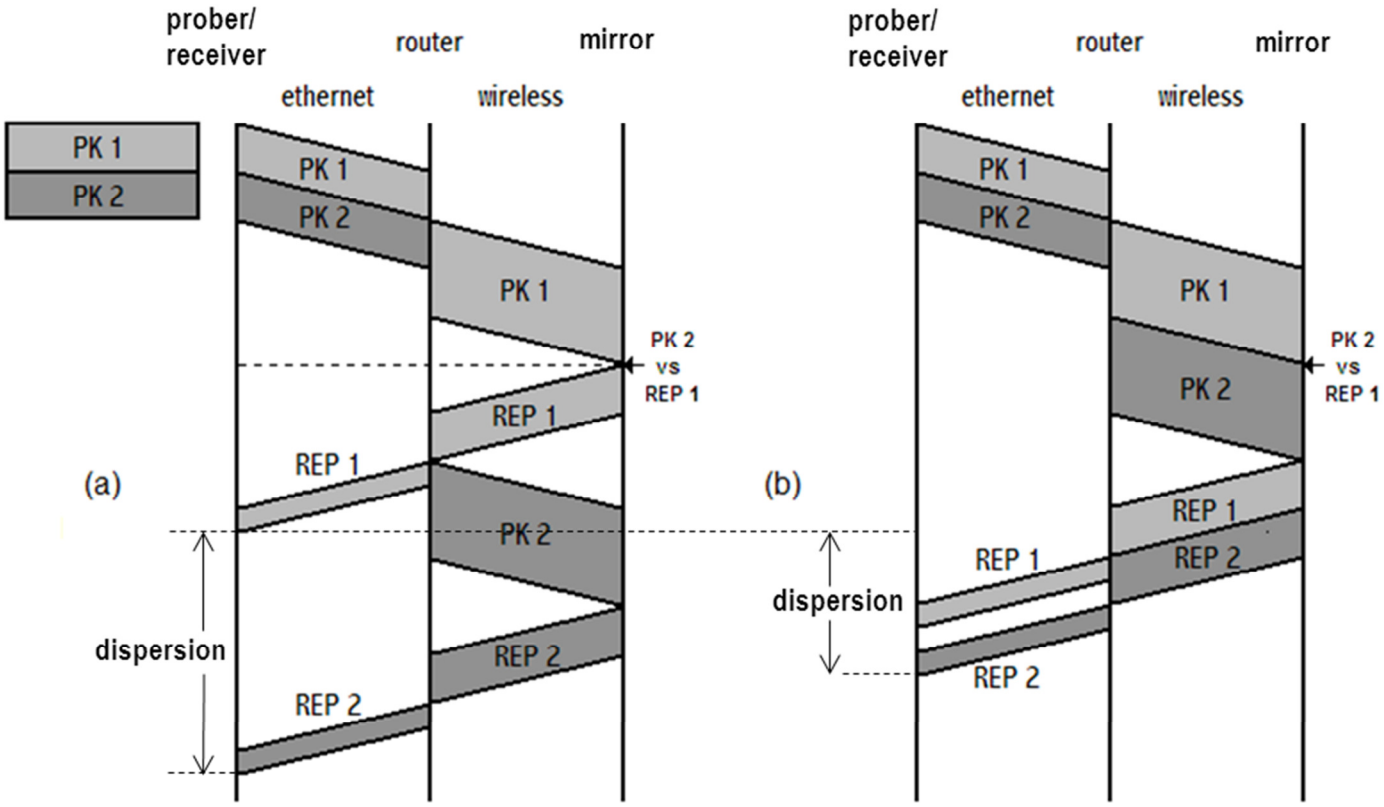


Figure 1. Conventional back-to-back packet-pair probing in heterogeneous home networks. PK1 and PK2 are the probe packets sent back-to-back on the path. REP1 and REP2 are the respective reply packets. a) PK2 has to wait until REP1 is off the wireless medium. b) REP2 has to wait until REP1 is off the wireless medium.

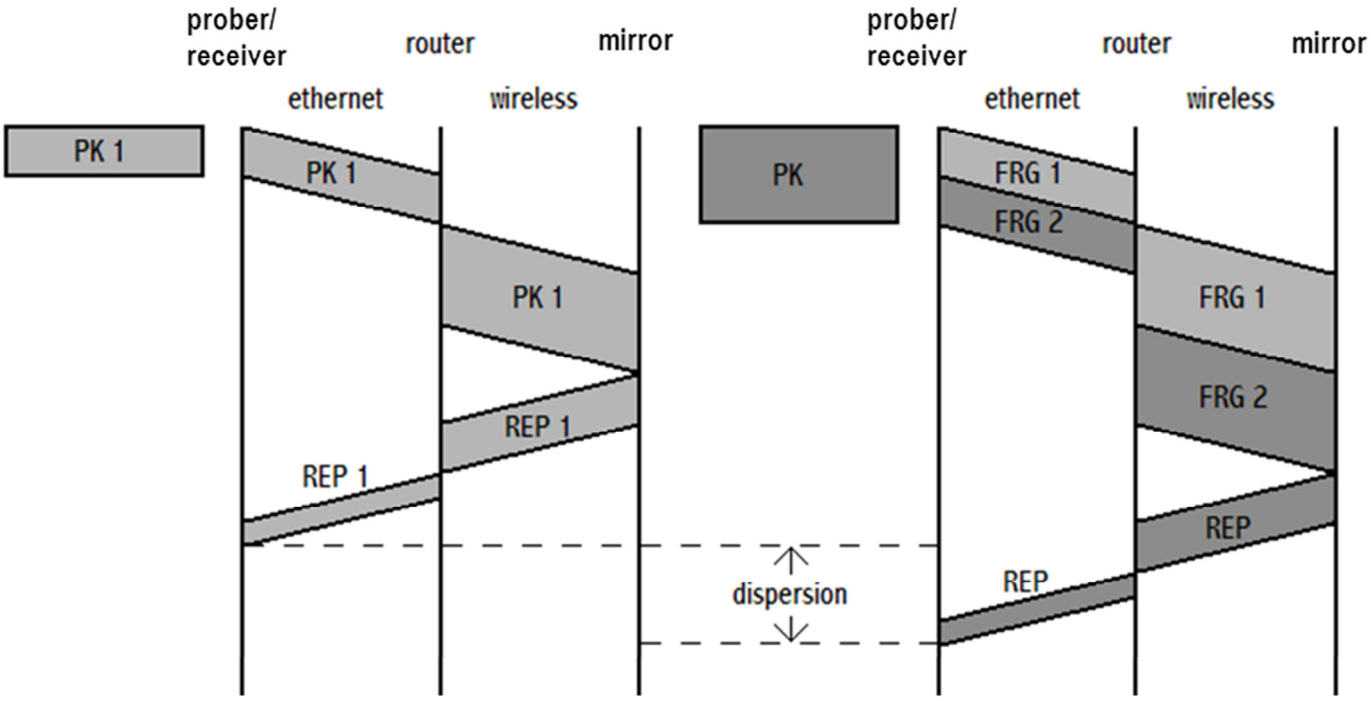


Figure 2. Allbest's method of active probing in heterogeneous home networks. Probe packets PK1 and PK are sent far apart from each other. PK has size 2xMTU and is automatically fragmented on the network. The fragments FRG are dispersed. A reply REP is sent only after defragmentation, thus no contention has occurred.

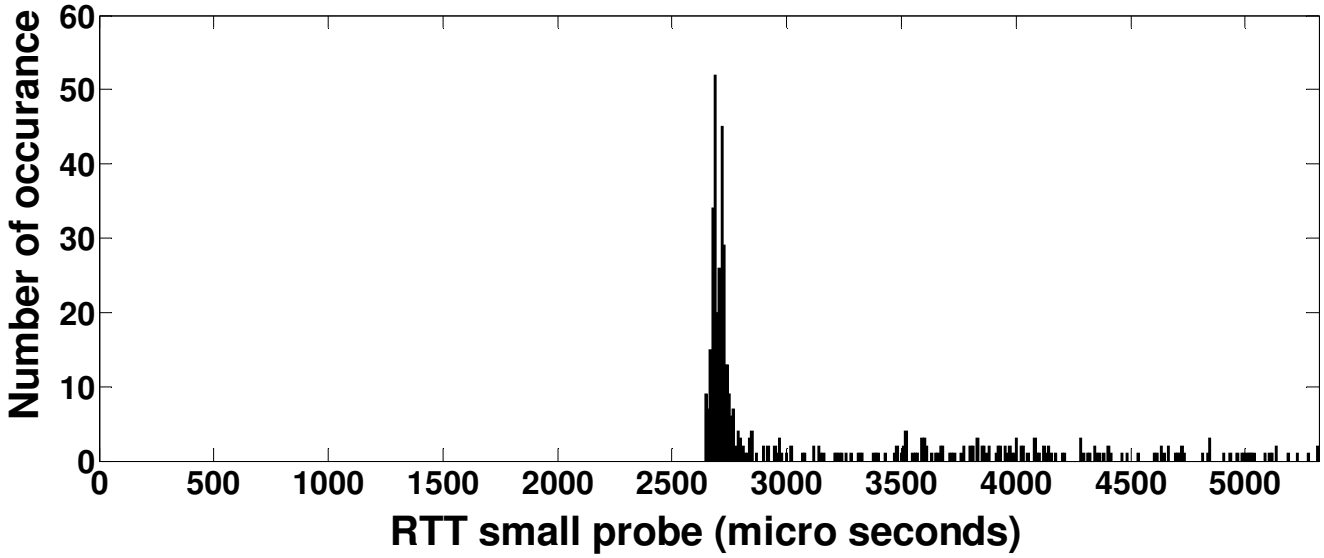


Figure 3. Histogram of 1000 RTTs of MTU-sized probe packets in an IEEE 802.11b network with 1.5 Mbit/s crossing traffic. Bin size = 20 µs.

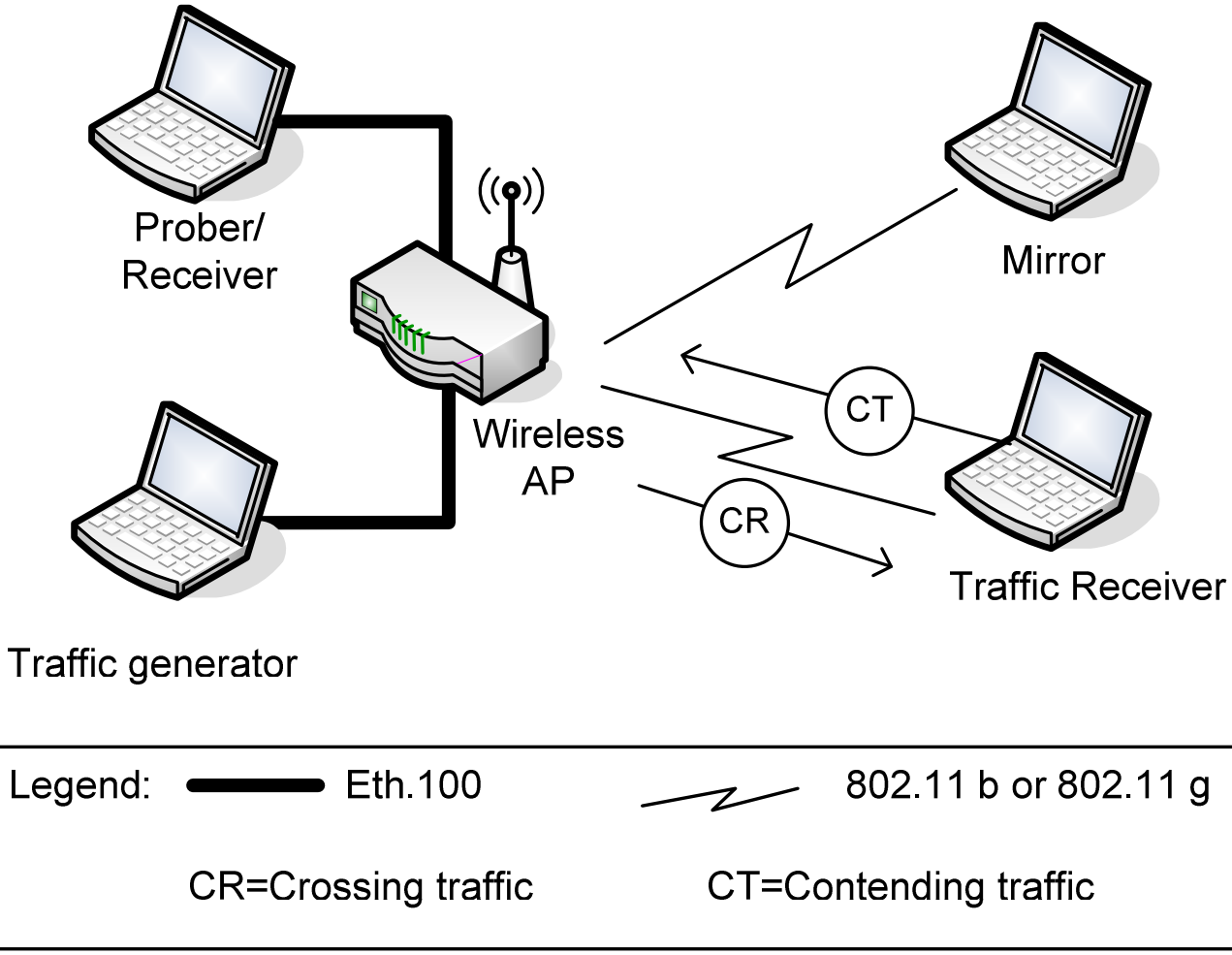


Figure 4. Schematic view of our heterogeneous wired/wireless LAN probing test bed. Allbest runs on the “prober/receiver”. The “traffic generator” generates crossing or contending traffic.

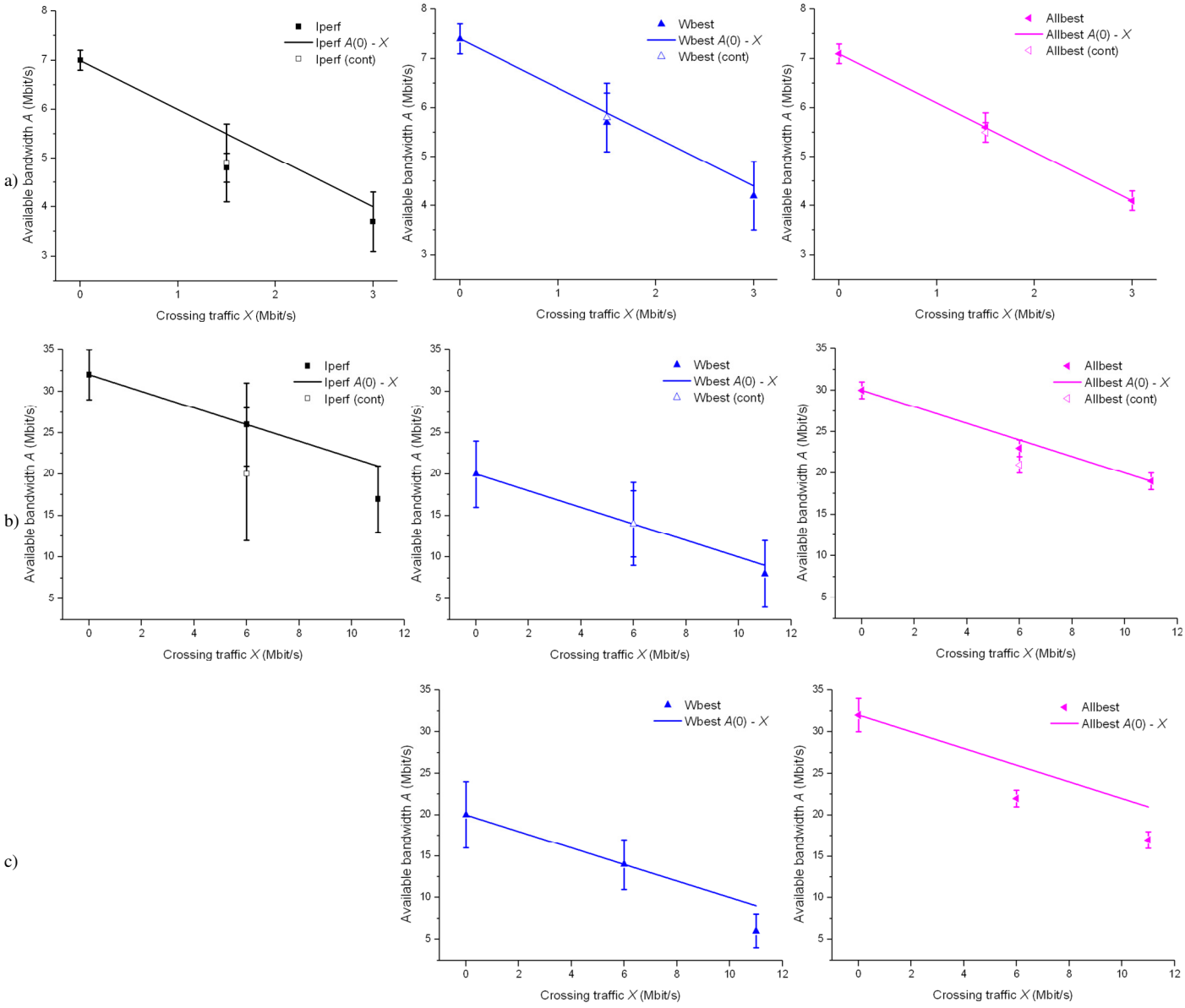


Figure 5. Available bandwidth measured with different tools and different amounts of crossing and contending (cont) traffic. a) topology 1; b) topology 2; c) topology 3

Performance Analysis of Home Network Topology Discovery Protocols

Erik German Diaz Castellanos, Archi Delphinanto, Frank den Hartog
TNO
Delft, The Netherlands
Frank.denHartog@tno.nl

Abstract—**Service providers are demanding the development of novel diagnostic tools with which they can remotely troubleshoot the home network. One of the tools should be able to gather information about the topology of the home network. In this paper we propose a set of key performance indicators for home network topology discovery architectures, and how they should be measured. We apply them to the Link-Layer Topology Discovery (LLTD) protocol and the Link-Layer Discovery Protocol (LLDP), and show that these protocols do not fulfill all the requirements as formulated by the service providers.**

Keywords: home network, topology discovery, LLTD, LLDP.

I. INTRODUCTION

The increasing popularity of Internet-Protocol (IP) based services has accelerated the evolution of consumer's home networks. The home network has become a complex environment providing connectivity to several devices with different capabilities. Any given home may now have one or more PCs, laptops, smartphones, tablets, Internet radio devices, set-top-boxes, network-attached storage devices, digital photo frames, game consoles, Internet phones, and printer servers, interconnected via a combination of Ethernet switches, Wifi access points, power line bridges, and cordless telephony base stations. These devices are often networked with each other in various ways, and connected to the managed operator network and the Internet via one or more Home Gateways (HGs). And in the near future it is expected that this heterogeneity of devices and networks in the home will prevail with smart meters, energy management devices and e-health devices using network technologies such as Zigbee, Z-wave, and Bluetooth.

For service providers, this is proving to be an increasing service management nightmare. Home networks are owned, installed, configured, and controlled by often ill-educated end users, but need to support the providers' services end-to-end. The network path between e.g. the HG and the relevant end device is beyond the control of the service provider, but more and more causes the services to perform under par, leading to end users calling the service providers' help desks. Although service providers possess tools to manage their own core and access networks, they lack the means to gather information related to home network characteristics. For them, the home network is largely a black box.

Service providers are therefore demanding the development of novel diagnostic tools with which they can remotely troubleshoot the home network. The worldwide Home Gateway Initiative is currently drafting a home network diagnostics requirements document [1] spelling out the service providers' needs. One of the required tools should be able to gather information about the topology of the home network. Topology is defined as a list of active devices and their capabilities, a list of connections between devices, and on a per-connection base the link technology that is used. In the remainder of this article, any device in the home that forwards, switches or routes traffic is called a Home Network Infrastructure Device (HNID). They are distinguished from the end devices on which the services are consumed, here also called STAtions (STAs), and the HG.

Current topology discovery mechanisms have been mainly developed for large-scale homogeneous Ethernet networks, such as business networks. For small-scale heterogeneous networks only two protocols are available nowadays. The most well-known is Microsoft's proprietary Link Layer Topology Discovery (LLTD) protocol, which is included in the modern Windows operating systems. An alternative protocol is provided by the open standard IEEE 802.1AB "Link Layer Discovery Protocol" (LLDP). There is no literature available on how well these protocols perform, let alone how they compare. Worse, it has not been established how the performance of these protocols should be measured.

In this paper we propose a set of key performance indicators for home network topology discovery architectures, and how they should be measured. We then describe our test bed and show our results for LLTD and LLDP. The results are then benchmarked against the service providers' requirements.

II. TOPOLOGY DISCOVERY PROTOCOLS

A. AFT and STP

The first step in topology discovery, namely device discovery, is well researched and many architectures exist, also for use in home networks [2]. A popular example is Universal Plug and Play (UPnP). The second step, connection discovery, is slightly more complicated. The most straightforward approach may seem to read out the Address Forwarding Tables (AFTs) stored in the switches in the network, and using the Spanning Tree Protocol (STP) for further topology discovery.

This work was partly done in the collaborative Integrated Project FIGARO, which is supported by the European Commission under the 7th Framework Program, Grant Agreement Number 258378.

These mechanisms have been well researched and described in [3,4] and subsequent papers. The advantage of this approach is that it uses standard technologies that are already supported by every switch in the market. There are two disadvantages: 1) it only works with Ethernet networks and not for other networks such as Wifi and Power Line Communication (PLC), and 2) for reading out the switches remotely, a management protocol needs to be used such as the Simple Network Management Protocol (SNMP). However, how SNMP should be used for this has not been standardized, and many switches do not support SNMP. Furthermore, the inference of the topology from the obtained data needs significant data processing.

B. LLDP

IEEE 802.1AB (LLDP) is a layer-2 protocol specifically designed for topology discovery in local area networks. It relies on LLDP agents being supported by the end devices and the HNIDs. The LLDP agents are the protocol end points. They encapsulate and transmit (or decapsulate after reception) LLDP Data Units (LLDPDUs) in a Medium Access Control (MAC) frame. The MAC frame is broadcasted on the broadcast domain that the transmitter is in. The LLDPDU consists of Type-Length-Value (TLV) fields containing information such as chassis ID, port ID, port description, system name, system description, system capabilities, and management address. With the received information, an LLDP agent updates a local Management Information Base (MIB), which can then be remotely accessed using a protocol such as SNMP.

A typical implementation has LLDP agents in transmit-only mode running on end devices. HNIDs then must support a full LLDP agent, a MIB, and an SNMP server (for which the HNID needs to be IP addressable). An SNMP client to read out the MIBs may run in the service provider's network, but to reduce overhead and complexity it should preferably run on the HG. Because of the rich information contained in de LLDPDUs, inferring the topology is easier than in the case of using AFTs and STP, and can also be done for heterogeneous networks, identifying the link types. Another advantage of LLDP stems from the fact that LLDPDUs are broadcast regularly, and MIBs are updated accordingly. The obtained topology map can therefore be assumed to be a good representation of the actual topology of the home network. The main disadvantage of LLDP is the heaviness of the requirements it puts on HNIDs. The HNIDs play a central role in the topology discovery to succeed, but many HNIDs currently in the market do not support SNMP or LLDP.

C. LLTD

LLTD [5] is a layer-2 protocol developed by Microsoft as part of the Windows Rally set of technologies. Its operation is based on a central entity, known as the "mapper", performing a series of tests on demand of the user. The mapper typically runs on PCs. But service providers are interested to have it implemented on the HG too. The other protocol end point is called a "responder". Its task is to respond to received LLTD test queries with the device and link information requested. Like LLDP, LLTD information is encapsulated in MAC frames. In contrast to LLDP, they are not broadcast autonomously, but as part of a test session initiated by the user. It is not publicly known how the mapper infers the topology from the received responses, but we assume that the algorithms used are close to the ones described in [6].

In LLTD, node and link discovery is a two-step process, first involving the mapper performing a Quick Discovery (QD) and then a sequence of Topology Discovery Tests (TDTs). With QD the responders in the home network are discovered via an exchange of broadcast "discover" frames and "hello" responses. The discovered responders are then interrogated by the TDT in a unicast fashion, using various sequences of "emit", "query" and "reset" frames, on which is responded with "probe", "train", "ack", "queryresp" or "flat" frames.

The main advantage of LLTD is that the requirements on the HNIDs are low. They should preferably run the lean responder stack. But even if they do not support LLTD, there is a significant chance that they will be detected indirectly by the mapper's intelligent discovery algorithm. It also means that the additional use of a remote management protocol is not needed. The mapper produces the topology map directly from the information it received.

D. HTIP

The International Telecommunication Union (ITU) G.phnt working group is currently working on a new protocol known as Home-network Topology Identifying Protocol (HTIP) [7]. Instead of using LLDP between the HNID and the end devices, HTIP uses a modified version of LLDP between the HNID and the HG. Between the HNIDs and the end devices it just uses the AFT information. The end devices themselves are discovered with UPnP. Now, the only requirement on the HNID is to run the modified LLDP agent. How the HG infers the topology from the obtained information has not been described yet. Because of the preliminary status of this new standard, we have not yet studied it any further.

III. PERFORMANCE INDICATORS

There is consensus among HGI members that a home network topology discovery diagnostics tool for service providers should fulfill the following requirements:

1. The accuracy must be close to 100%, i.e. the obtained map must contain a negligible amount of mistakes.
2. The time between requesting a topology map and obtaining it must be less than 2 seconds.
3. The overhead traffic that the topology discovery procedure creates and the memory resources it confiscates must not disturb other services in the home.

Inspired by these requirements we defined the following performance indicators for home network topology discovery.

The topology discovery problem can be divided into two problems: 1) discovery and classification of HNIDs and end devices within a home network according to their behavior and supported link layer technologies, and 2) the creation of a graph representing how these devices are interconnected. Let the HNIDs for example be an Ethernet switch (i.e. an Ethernet-Ethernet bridge, here abbreviated as SW), a Wifi access point (i.e. an Ethernet-Wifi bridge, here abbreviated as AP) or a HomePlug node (i.e. an Ethernet-PowerLine bridge, here abbreviated as HP). If we then classify all end devices as STAs,

that leaves us with four possible devices to discover. Receiver Operating Characteristics (ROC) is a method used to analyze classification systems. Our classifiers try to relate an unknown device to one of the four possible types according to its behavior or advertised information. The result of the match could be positive (P) or negative (N). After comparing the actual type of the device and the identified type of device, we have the following possible outcomes:
True Positive (TP): An obtained positive match is correct.
False Positive (FP): An obtained positive match is incorrect.
True Negative (TN): An obtained negative match is correct.
False Negative (FN): An obtained negative match is incorrect.
The classification accuracy Acc_{class} can then be expressed as

$$Acc_{class} = (\#TP+\#TN)/(\#P+\#N). \quad (1)$$

The quality of a classifier can also be represented as a coordinate in a ROC graph, which shows the true positives rates (*TPR*) on the Y-axis and the false positive rates (*FPR*) on the X-axis, with

$$FPR = \#FP/\#N \quad \text{and} \quad TPR = \#TP/\#P. \quad (2)$$

A classifier *A* is equally good or better than *B* if its position in the ROC graph is closer to (0,1).

A network can be modeled by using undirected graphs. Networked devices and connections are represented by nodes and links. A graph has a mathematical representation known as adjacency matrix. The matrix has size *M* x *M* (*M* is the number of nodes). Its elements are 1 where a link exists between nodes and 0 otherwise. We can compare the real topology with the generated map by comparing their adjacency matrices, and find the number of positions where the values in both matrices are 1 (#TP) or 0 (#TN). The graph accuracy Acc_{graph} then equals:

$$Acc_{graph} = (\#TP+\#TN)/M^2. \quad (3)$$

LLTD broadcasts a Reset frame when the LLTD discovery process is initiated and also when the LLTD discovery process ends. The difference between the timestamps of these frames (measured with e.g. Wireshark) gives us the discovery time for LLTD. The Network Management System (NMS), containing the SNMP client for topology discovery with LLDP, sends SNMP queries to all active HNIDs. To estimate the LLDP discovery time, we subtract the timestamps from the first and last SNMP queries.

The traffic generated by the topology discovery processes consists of sequences of probing and advertisement message exchanges during the discovery time. It does not contain large media streams, flooding experiments, etc. We therefore measure this overhead traffic as a rate averaged over a relatively long period of time. For the latter we do not choose the discovery time, because a long process injecting a lot of traffic would, in reality, be more disturbing than a short process injecting little traffic. We therefore decided to use a fixed duration of 60 s, and assume that any discovery process started at t = 0 s would be over by then.

The total memory resources required by each protocol are given by the memory space needed by daemons, the memory space required to perform the tests, and the memory space required to store the topology data. The first two are measured with, respectively, the *Mem Usage* and the *Mem Delta* parameter of Windows Task Manager. The memory space needed to store the topology data is calculated from the protocol specs. The end result for memory usage is the sum of *Mem Usage*, *Mem Delta*, and the storage space.

IV. TEST BED IMPLEMENTATION

The HG in our test bed must include the protocols of interest. HG manufacturers do not yet offer an appropriate HG supporting LLTD and LLDP. We therefore constructed an HG by taking a CISCO SF-300-08 Ethernet Switch for small business with LLDP agent, MIB and SNMP server (3 in Fig. 1), and connected it with a Linksys WRT54GL gateway containing a router and a DHCP server (1 in Fig. 1). Also connected is a Dell Latitude 2100 Netbook (2 in Fig. 1) running an LLTD mapper in its Microsoft Windows Vista OS, the Solarwinds Engineer's NMS which includes an SNMP client and a topology inference algorithm, and Wireshark.

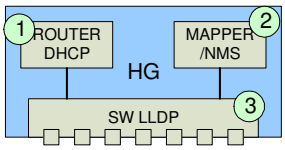


Figure 1. The test bed HG containing 3 different physical components.

Our end devices are mimicked by Acer Aspire ONE Netbooks running Windows XP on which we installed Microsoft's LLTD responder. There are many open source LLDP clients available. We chose the haneWIN LLDP agent because it runs on Windows and supports both transmission and reception mode.

Our choice of HNIDs is strongly depending on what the market has to offer. This makes that our measurements will give a good indication of what service providers may observe in real life today. The disadvantage is that not all theoretically possible configurations can be tested. The HNIDs we used are the CISCO SF-300-08 Ethernet Switch, the Hewlett Packard HP V-M200 802.11n Wireless Access Point, and the Sitecom LN – 513 HomePlug adapters. Table 1 shows for all devices used in our test bed their support for LLTD and LLDP.

TABLE I. LLDP AND LLTD SUPPORT OF DEVICES IN TEST BED

Device	Type	LLDP agent		LLTD	
		Tx mode	Rx mode	Responder	Mapper
HG	HG	No	yes	no	yes
Station	End-device	yes	no	yes	no
Access Point	HNID	Yes	no	no	no
Switch	HNID	yes	yes	no	no
Home Plug	HNID	no	no	no	no

The basic test bed configurations we used for testing LLTD and LLDP are shown in Fig. 2. They are based on research that TNO recently did in a selection of households in The Netherlands. The result was a snapshot of home network

topologies used by the "early majority" of technology adopters. We found that the vast majority of these home networks had a topology close to one of these basic configurations, or a simple linear combination of some of them. The number of stations connected was varied from 1 to 3 for every configuration.

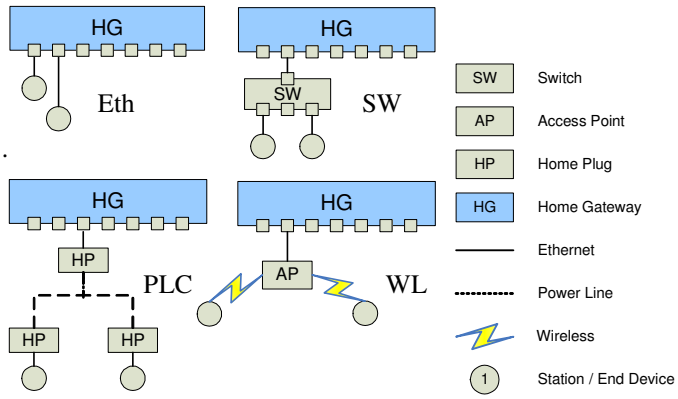


Figure 2. Basic test bed configurations

V. RESULTS

An example of a topology map generated by LLTD is shown in Fig. 3, for the PLC configuration with 2 stations. The upper three devices and their interconnection correctly represent the HG (and its disconnect from the Internet). Also the two stations and their links to the home network are correct. But the HomePlug power line nodes are incorrectly shown as an Ethernet hub and two switches interconnected with Ethernet. Maps generated by LLDP and the SNMP NMS look similar. Similarly we generated maps for every configuration, for 1-3 end devices, and for LLDP and LLTD, i.e. 24 in total. Every measurement is repeated 5 times to check reproducibility and precision. The latter is found to be ~5% for all results.

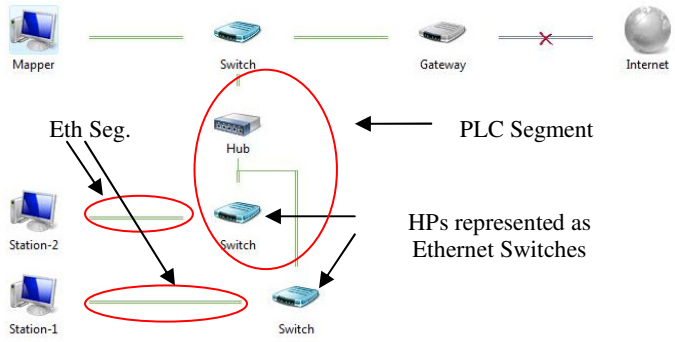


Figure 3. Example of a generated topology map (LLTD, config. PLC)

For every map we found the false and true positives and negatives and the adjacency matrix, and calculated the accuracy indicators according to (1) and (3). The results are shown in Table 2. The ROC graph is shown in Fig. 4. Note that in Fig. 4, LLDP gives the perfect score. From the results we conclude that LLDP has a better classification accuracy than LLTD, but a worse graph accuracy. This can be explained by the relatively superb advertisement mechanism of LLDP devices, as well as the limited support of LLDP by commercial HNIDs combined with the clever algorithms performed by the LLTD mapper. In none of the cases is the accuracy near 100%, and therefore req. 1 is not fulfilled.

TABLE II. CLASSIFICATION AND GRAPH ACCURACY FOR LLTD AND LLDP

	LLTD	LLDP
# TP	3	3
# TN	20	15
# FP	1	0
# FN	4	2
# P	4	3
# N	24	17
Acc_{class}	82%	90%

		Acc_{graph}	
Config.	**#STAs**	**LLTD**	**LLDP**
Eth	1	100%	100%
	2	100%	100%
	3	100%	100%
SW	1	100%	100%
	2	100%	100%
	3	100%	100%
PLC	1	63%	50%
	2	72%	56%
	3	83%	59%
WL	1	100%	78%
	2	100%	75%
	3	100%	76%

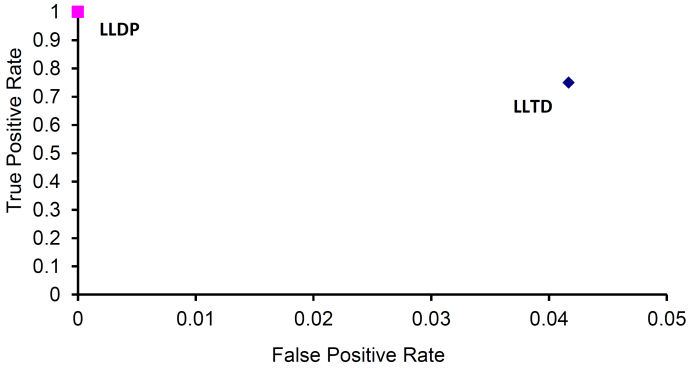


Figure 4. ROC graph for LLTD and LLDP

Fig. 5 shows the results for the discovery time. The results for LLTD are different for the different configurations, but are always in the range of 4-10 s. For configurations Eth, SW, and WL the topology discovery time increases linearly, but very little (by 1% to 5%) with every new station added. For PLC, the increment of topology discovery time is more pronounced compared to other configurations (10% to 30%). This can be explained by the algorithm deciding on doing many more tests than for the other configurations, probably because it understands that this configuration is relatively unusual, and it needs extra tests to indirectly infer the existence of the HomePlug nodes. For LLDP, the curves for the configurations Eth, SW and PLC virtually overlap. The discovery times are also much longer than for LLTD, namely in the range of 16-55 s. It turns out that most of the time is needed for reading out the MIBs. For WL, the discovery time seems to be independent of the number of active stations. For LLTD we found out that for the WL configuration, the QD test is already enough to discover the topology. The AP only supports the Tx mode of LLDP, and therefore the end devices are not discovered at all with LLDP. LLTD nor LLDP fulfills req. 2.

Fig. 6 shows the average injected traffic rate for LLDP and LLTD. The explanation of the results follows largely the same logic as for the discovery time. For any configuration, the absolute size of the injected traffic rate is very low and easily fulfills req. 3.

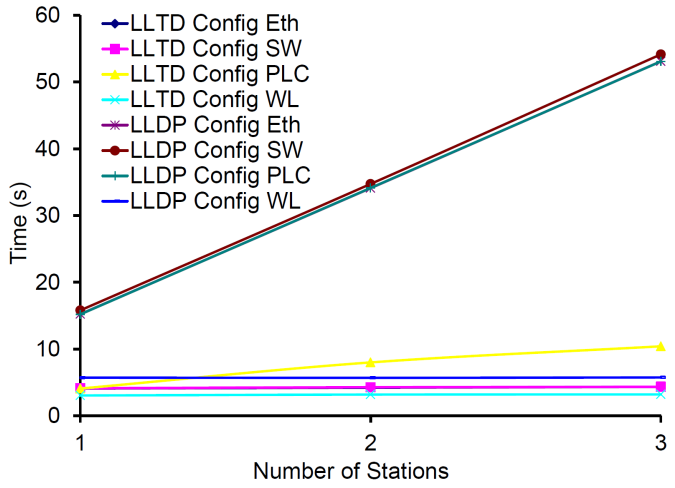


Figure 5. Discovery time for LLTD and LLDP

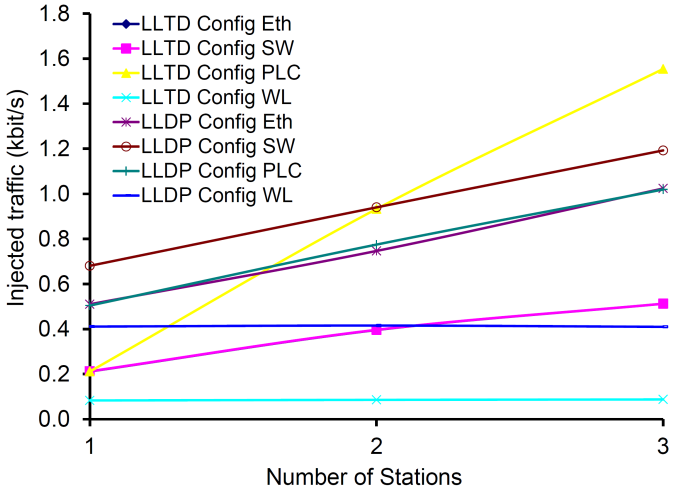


Figure 6. Average injected traffic rate

In terms of memory usage, we observed that the LLTD mapper requires a minimum of 44 Kbytes and a maximum of 120 Kbytes to operate. The relevant dll is called lltdsvc.dll and its size is 188 Kbytes. In total, LLTD thus requires 232-308 Kbytes of memory. If we use all mandatory and optional TLV types, and assuming a maximum data size for each field, for each neighbor device, an LLDP HNID MIB will require 1446 Bytes to store its information. The size of the LLDP daemon used in the test bed is approximately 500 Kbytes. In the worst scenario (3 stations), an LLDP HNID thus needs approximately 505 Kbytes of total memory. This is nearly double the amount of LLTD, but in both cases this use of memory should not disturb normal service operation on the HG too much (req. 3).

VI. CONCLUSIONS AND FUTURE WORK

To our knowledge, this is the first time that a framework is established to evaluate and compare the operation and performance of topology discovery protocols for heterogeneous consumer networks. We defined four performance indicators, namely discovery time, average injected traffic rate, accuracy and memory use. Of those four, accuracy is the least straightforward to measure, and to do so we used graph theory and receiver-operating-characteristics techniques. We decided to define accuracy with two separate numbers, namely the correctness of the discovered device types and the correctness of the discovered link types.

The assessment of LLTD and LLDP showed that, although LLTD performs slightly better than LLDP (mainly in terms of discovery time and graph accuracy), none of the protocols fulfills the service providers' requirements satisfactorily. LLDP relies on the presence of manageable HNIDs. Having to address individual HNIDs with a remote management protocol to read out the internal data bases makes an architecture based on LLDP relatively slow and therefore less suitable for use by service providers. Besides, manageable HNIDs are not yet common within home networks. The main weakness of LLTD is the accuracy of the discovery results for home networks containing collision domains linking more than two HNIDs.

In the future we aim to design, test, and standardize a new topology discovery architecture which addresses all service providers' requirements successfully, and also includes non-IP network domains such as Zigbee. We also intend to test ITU-T's HTIP when the standard is completed and implemented, and compare it with the other protocols.

Topology discovery is a monitoring application and in itself does not provide solutions for the detected problems. Methods or systems must be developed to intelligently exploit this information according to the service providers' needs. Examples are the design of an expert system to recognize topologies that could affect QoS, and a logging system to store historic topology information.

Although the set of configuration we used for assessing the protocols' performance is based on realistic home network topologies and is fairly complete, it should be analyzed how much the result differ for less common configurations.

REFERENCES

[1] HGI, "Requirements for home networks service diagnostics", HGI-RWD016-R3, www.homegateway.org, unpublished

[2] A. Delphinanto, B.A.G. Hillen, I. Passchier, B.H.A. van Schoonhoven, and F.T.H. den Hartog, "Remote discovery and management of end-user devices in heterogeneous private networks", Proc. of the 6th Annual IEEE Consumer Communications and Networking Conference (CCNC 2009), Las Vegas NV, United States.

[3] B. Lowekamp, D. O'Hallaron, and T. Gross, "Topology discovery for large ethernet networks", Proc. of the ACM SIGCOMM'01 conference, San Diego CA, United States.

[4] H. Peng, P. Heng, L. Xiangdong, and Z. Qiusheng, "Physical topology discovery based on Spanning Tree Protocol", Proc. of the International Conferenc on Computer Application and System Modeling 2010, Taiyuan, PR China.

[5] Microsoft, "Link Layer Topology Discovery protocol specification", http://msdn.microsoft.com/en-us/windows/hardware/gg463024, September 2010.

[6] R. Black, A. Donnelly, and C. Fournet, "Ethernet Topology Discovery without Network Assistance", Proc. of the 12th IEEE International Conference on Network Protocols (ICNP 2004), Berlin, Germany.

[7] Y. Mihara, T. Yamazaki, A. Takehiro, "Designing HTIP: Home Network Topology Identifying Protocol", Proc. of 2011 IEEE International Conference on Communications (ICC), Kyoto, Japan.

Bandwidth Monitoring in Multi-rate 802.11 WLANs with Elastic Traffic Awareness

Claudio Rossi, Claudio Casetti, Carla-Fabiana Chiasserini
Politecnico di Torino, Torino, Italy
Email: *lastname*@tlc.polito.it

***Abstract*—We present a lightweight algorithm for the estimation of the node achievable throughput and available bandwidth in IEEE 802.11 wireless networks. In particular, we consider a multirate WLAN with access point (AP), where there may be both elastic and inelastic traffic flows. Through our algorithm and leveraging previous theoretical results, the AP can autonomously estimate: (i) the available bandwidth that a new station wishing to associate with the AP can use, (ii) the impact on the system performance of admitting the new station, (iii) the bandwidth still available (if any) for inelastic traffic. The above quantities can be effectively used for admission control in WLANs and load balancing among APs with overlapping coverages. Indeed, simulation results obtained using Omnet++ show that the estimates yielded by our algorithm accurately reflect the system throughput behavior when there are both elastic and inelastic traffic flows, in the uplink and downlink directions.**

I. Introduction

The current trend of providing wireless users with ubiquitous connectivity to the Internet has determined a wide deployment of access points (APs) using the IEEE 802.11 technology. Often, especially in highly populated areas, one location is under the coverage of more than one AP, even though a wireless station (WS) that happens to be in that location is associated to a single AP. The selection of the AP is usually dictated by users' preferences or subscription plans, and, as such, it disregards congestion levels or channel quality issues. However, if we abstract from the current scenario and address a broader picture carrying concerns such as energy-efficiency and electromagnetic pollution, a solution that minimizes the overlapping coverages of APs without degrading the performance expected by the WSs is desirable. In order to achieve such a goal, APs with overlapping coverages should identify and optimally split the WSs among themselves, and, possibly, turn themselves off if a subset of APs can adequately support the current load requested by WSs.

To realize the above vision, discounting the obvious constraints currently imposed by contractual obligations, it is imperative that the APs should gauge the impact of one WS associating to a specific AP. Indeed, it is well known that users within the coverage of the same 802.11 AP share the available bandwidth by using the Distributed Coordination Function (DCF) at the MAC layer, which grants users an equal long-term channel access probability.

In this paper, we aim at evaluating the impact of the admission of a new WS (or, equivalently, of a new traffic flow) in a multirate WLAN scenario. Unlike previous work (see Sec. II), we present an algorithm that an AP can run to sensibly predict two important quantities. First, the available bandwidth, which is measured through a parameter, called b-metric, that we introduce. Second, the throughput decrease that, given the current load of the AP, the association of an additional WS will cause to the currently associated WSs.

The algorithm is based on online measurements by the AP integrated with previous theoretical results, and does not require explicit signaling between AP and WSs. Also, it accounts for the presence of both *elastic* and *inelastic* traffic, namely TCP and UDP traffic flows, in view of estimating the throughput achievable by the WSs as well as their losses. As a matter of fact, the user experience is severely affected by losses while running real-time audio/video applications supported by UDP, such as VoIP. On the contrary, losses on applications supported by TCP, such as HTTP or FTP, result in a throughput degradation which could be annoying for the user but not impairing. Hence, we need to take into account the nature of the existing traffic flows and their different impact on the user satisfaction level.

We point out that our algorithm can be seen as a stepping stone toward the definition of a more comprehensive framework, where traffic admission control and dynamic load balancing among APs with overlapping coverages can be achieved.

II. Related work and motivation

Here, we briefly recall the most popular approaches to load estimation in WLANs, and explain why they cannot be applied to a multirate Basic Service Set (BSS) in presence of different types of traffic.

Load estimation techniques can be classified as passive or active. The latter ones require to inject probing packets into the network and estimate the traffic load based on the delay experienced by such packets. Probing packets, however, yield additional overhead, and could have a negative impact on data flows, especially in case of real-time traffic [1].

We will therefore focus on passive techniques, which aim at estimating the traffic load by observing some meaningful metrics. As an example, in [2] the load is related to the number of WSs associated with the AP, which is a valid approach when all WSs have the same behavior. Under more general conditions, other metrics have been proposed, based either on the channel idle (or, equivalently, busy) time [3]–[5] or on the aggregated BSS throughput [6]. However, when

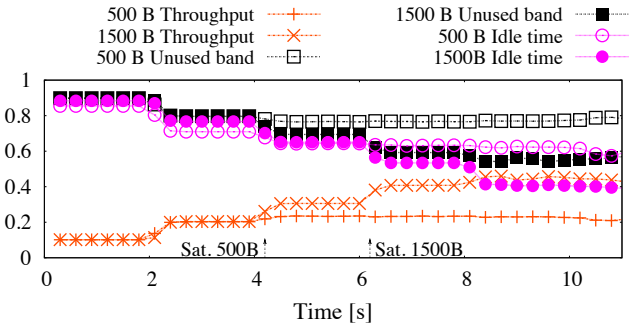


Fig. 1. Time evolution of the normalized aggregate throughput, unused bandwidth and idle time, when 6 WSs become active at different time instants and transmit packets with payload of 500 B and 1500 B.

the BSS nodes always have data to transmit, the idle time introduced by interframe spaces and backoff periods, as well as the value of saturation throughput, depends on the payload size and on the data rate of the transmitted packets. It follows that such metrics may indicate saturation in the presence of available bandwidth, or, conversely, availability of bandwidth when saturation has been already reached. This latter effect can be observed in Fig. 1, where the time evolutions of the normalized idle time and available bandwidth (computed as one minus the normalized aggregate throughput) are compared to the behavior of the throughput itself. The results refer to an IEEE 802.11g BSS with one AP and 6 WSs; all nodes initially transmit at 54 Mbps and then vary their data rate through the AARF scheme [7] . Every two seconds between 0 s and 10 s, a WS starts generating a CBR traffic flow at 5.5 Mbps. Two different values of payload are considered, namely, 500 and 1500 bytes. We can clearly see that when the throughput reaches saturation, both the idle time and the available bandwidth metrics stabilize to positive values, which are different and vary according to the considered scenario.

In order to address these shortcomings, we propose an algorithm for bandwidth monitoring and a new metric that account for varying payload size and data rate, and for the presence of both elastic and inelastic traffic. We would like to mention that an early definition of such a metric, based on inelastic traffic only, was sketched in our short paper [8].

III. System scenario and preliminaries

We consider an IEEE 802.11 BSS with AP. The technology we refer to can be any among a, b, and g; also, the solution is described for the standard 802.11 MAC without RTS/CTS, but it can be extended to the DCF with handshake as well as to the 802.11e EDCA.

The WSs within the BSS can be sources or destinations of elastic or inelastic traffic flows, i.e, flows that use either TCP[1] or UDP at the transport layer. At the MAC layer, the AP and the WSs may transmit frames with different payload size and their data rate may vary according to the experienced channel propagation conditions.

[1]Recall that each TCP flow implies the transmission of data packets in one direction and of acknowledgment messages in the opposite direction.

We assume that the AP can access the "protocol type" field in the IP packets, and collects statistics on the transmissions observed over the medium, such as the average payload size and data rate used for packet transmission within the BSS and the throughput experienced by the nodes.

More specifically, the AP carries out such measurements over time intervals, named *cycles*. A cycle is defined as the minimum between a time T_{max} and the period needed to let (1) each active WS successfully send at least one data frame carrying inelastic traffic, and (2) the AP successfully transmits at least one data frame carrying inelastic traffic to every WS for which it has data to send. The AP considers a WS to be active in the generic cycle j if it successfully receives from the WS at least one data frame within a time T_{max} since the current cycle starting time. Likewise, the AP is active in cycle j if it has sent at least one frame within the cycle. In the following, we denote by $C(j)$ the duration of cycle j and by $\mathcal{N}(j)$ the set of nodes (WSs and AP) that were active in the cycle ($|\mathcal{N}(j)| = N(j)$).

The AP keeps track of the number of active WSs and, for each cycle j, computes the duration $C(j)$. Then, at each cycle j and for each active WS k, the AP computes the uplink throughput for elastic and inelastic traffic of k, denoted by $\eta_k(j)$ and $\nu_k(j)$, respectively, as the ratio of the amount of data successfully transmitted by the WS in cycle j to the cycle duration. Similarly, considering the amount of data successfully received by the WSs in cycle j, the AP computes its own throughput for elastic and inelastic traffic, denoted by $\eta_{AP}(j)$ and $\nu_{AP}(j)$, respectively. Then, for each of these quantities, the AP computes a running average, $\overline{\eta}_k(j)$ and $\overline{\nu}_k(j)$ with $k \in \mathcal{N}(j)$, using the well-known exponential smoothing filtering. E.g., for the elastic traffic throughput of node k, we have:

$$\overline{\eta}_k(0) = \eta_k(0) \quad ; \quad \overline{\eta}_k(j) = \alpha\eta_k(j) + (1-\alpha)\overline{\eta}_k(j-1)$$

where $0 < \alpha < 1$.

Likewise, for each frame successfully transmitted by a WS or by the AP itself, the AP observes the payload size for elastic/inelastic traffic and the used data rates, and it computes the corresponding running averages: $\overline{P}_{k,e}(j)$, $\overline{P}_{k,i}(j)$, and $\overline{R}_k(j)$ ($k \in \mathcal{N}(j)$). For the data rate, the AP stores only one value because automatic rate adaptation algorithms do not distinguish between elastic and inelastic flows. Furthermore, the AP computes the running average of the data rate, $\overline{R}(j)$, and of the payload size, $\overline{P}(j)$, over all data frames, carrying either elastic or inelastic traffic, that it successfully sends or receives.

Next, we introduce a fundamental quantity for our bandwidth monitoring algorithm. Let us consider cycle j. At the end of the cycle, the AP computes the (aggregate) saturation throughput $S(j)$, as defined in [11], which extends the original Bianchi's model [10] in presence of errors due to channel propagation conditions:

$$S(j) = \frac{N(j)\tau(j)[1-\tau(j)]^{N(j)-1}\overline{P}(j)(1-\overline{p}_e(j))}{E[T(j)]}. \quad (1)$$

In (1), $\tau(j)$ is the probability that a node (either a WS or the AP) accesses the medium at a generic time slot in cycle j, $\overline{p}_e(j)$ is the filtered average packet error rate, and $E[T(j)]$ is the average duration of a time interval in which an event occurs (namely, an empty slot, a successful transmission, a transmission failed due to channel errors, or a collision). The expressions of $\tau(j)$ and $E[T(j)]$ can be derived following [11] and are reported in [9] for completeness, while $\overline{p}_e(j)$ can be estimated by the AP based on the modulations used for the transmissions in the j-th cycle, their associated signal-to-noise ratio, and assuming independent bit errors on the channel.

Using (1), the AP computes the average per-node throughput under saturation conditions, as $S_n(j) = S(j)/N(j)$. Note that $S_n(j)$ represents the saturation throughput for a node with average behavior, i.e., a node using a payload size $\overline{P}(j)$ and a data rate $\overline{R}(j)$.

IV. Bandwidth monitoring algorithm

As mentioned earlier, our aim is threefold: (i) estimating the throughput that either elastic or inelastic traffic flows newly originated within a BSS can achieve, (ii) gauging the impact that the newly generated traffic will have on the performance of inelastic traffic flows already present within the BSS, and (iii) monitoring the available bandwidth for inelastic traffic (if there is any left). We focus on the effects of the new traffic flows on inelastic traffic only, since this type of traffic has more stringent quality of service requirements.

Without loss of generality, in the following we describe our estimation algorithm in the case where a new WS, which may be source or destination of elastic or inelastic traffic, wishes to join the BSS. The extension to the case where a WS already associated to the BSS wishes to start a new traffic flow and notifies the AP about it (as foreseen in 802.11e BSSs) is straightforward.

Let j identify the last cycle. We consider that for an incoming WS, x, roaming from another BSS, the AP can acquire through signaling exchange between the APs the same statistics that it has been collecting for the active WSs, plus the downlink throughput that x would like to receive, hereinafter denoted by $\overline{\nu}_{AP}^{(x)}+\overline{\eta}_{AP}^{(x)}$. If, instead, such information is unavailable, in order to monitor the bandwidth availability the AP makes the following assumptions: (i) both uplink and downlink data frames from/to x have a payload size equal to the average value $\overline{P}(j)$; (ii) x transmits at the average data rate $\overline{R}(j)$; (iii) x's desired throughput is equal to $\overline{\nu}_x(j) = S(j)$ and $\overline{\eta}_x(j) = 0$ in uplink, and $\overline{\nu}_{AP}^{(x)} = S(j)$, $\overline{\eta}_{AP}^{(x)} = 0$ in downlink. Then, the AP updates its desired throughput as $\overline{\nu}_{AP}(j) = \overline{\nu}_{AP}(j) + \overline{\nu}_{AP}^{(x)}$ and $\overline{\eta}_{AP}(j) = \overline{\eta}_{AP}(j) + \overline{\eta}_{AP}^{(x)}$; also, it updates the set $\mathcal{N}(j)$ by adding x.

Considering that the 802.11 access scheme provides per-packet fairness, it is clear that any node $k \in \mathcal{N}(j)$, such that $\overline{\eta}_k(j) + \overline{\nu}_k(j) \leq S_n(j)$, will be able to transmit all its uplink traffic, both elastic and inelastic, while the others will reach $S_n(j)$ and then will share the remaining bandwidth, if any. Thus, in order to evaluate the throughput that x would achieve and its impact on the performance of the other nodes, we have to estimate the throughput that each active node can obtain with respect to the value it has experienced. To do so, we adopt the procedure reported below in Algorithm 1.

According to the proposed algorithm, we first compute the remaining bandwidth $B(j)$ as the difference between the available bandwidth, set equal to the saturation throughput $S(j)$, and the sum of the shares of the nodes (line 3). Each node share is computed as the minimum between $S_n(j)$ and its total (elastic and inelastic) throughput, as measured by the AP in cycle j. Then, lines 4–5 in Algorithm 1 report the amount of inelastic and elastic node throughput that can be accommodated within the $S_n(j)$ share.

We identify the set of nodes $\mathcal{N}_o$ whose total (elastic and inelastic) throughput exceeds $S_n(j)$ (line 7). Considering one of these nodes at a time, we assume that it will get a fraction of the remaining bandwidth so as to transmit one additional packet of average size. While doing this, the node will give priority to inelastic traffic. This occurs while (i) $B(j) > 0$ and (ii) there is at least one node for which the throughput experienced in cycle j has not been reached yet (lines 8–22). As $S_n(j)$ has been computed considering the average node behavior, we weigh the bandwidth consumed by the generic node k to transmit a packet by $\frac{\overline{R}(j)}{\overline{R}_k(j)}$, thus accounting for the actual data rate used by the node (lines 11 and 15). Also, we consider the worst case in which nodes with the lowest data rate seize the channel first. Indeed, the lower the data rate, the larger the consumed bandwidth (line 7).

At the end of this procedure, we obtain the estimated throughput that the nodes, including the AP and the incoming WS, can achieve for elastic and inelastic traffic ($\hat{\nu}_k(j)$ and $\hat{\eta}_k(j)$, $k \in \mathcal{N}(j)$), as well as the bandwidth ($B(j)$) still available (if any) for inelastic traffic.

Note that our procedure is able to reflect the impact of leaving WSs. Indeed, they will not be considered active in the following cycles, and consequently their traffic profile will not be accounted both in Algorithm 1 as well as in the computation of $S(J)$.

A. Computation of the b-metric

We now gauge the impact on the entire BSS resulting from the association of the new WS x.

In order to do so, on the one hand, we evaluate the degradation of inelastic traffic performance perceived by the nodes already associated with the AP. We neglect elastic traffic losses, as they can be recovered by TCP. We compute the estimated total throughput loss as:

$$\begin{aligned} L(j) &= \max\left\{\left[\overline{\nu}_{AP}(j) - \overline{\nu}_{AP}^{(x)}\right] - \hat{\nu}_{AP}(j), 0\right\} \\ &\quad + \sum_{k \in \mathcal{N}(j)\setminus\{x,AP\}} [\overline{\nu}_k(j) - \hat{\nu}_k(j)] \end{aligned} \quad (2)$$

where $[\overline{\nu}_{AP}(j) - \overline{\nu}_{AP}^{(x)}]$ is the throughput of the AP measured during cycle j, i.e., without considering the downlink traffic request of x.

Algorithm 1 Bandwidth monitoring

Input: $\mathcal{N}(j)$, $S(j)$, $S_n(j)$, $\overline{R}(j)$, $\overline{\eta}_k(j)$, $\overline{\nu}_k(j)$, $\overline{P}_{k,e}(j)$, $\overline{P}_{k,i}(j)$, $\overline{R}_k(j)$
Output: $B(j)$, $\hat{\nu}_k(j)$, $\hat{\eta}_k(j)$

$B(j) \leftarrow S(j)$
for $k \in \mathcal{N}(j)$ **do**
  $B(j) \leftarrow B(j) - \min\{\overline{\nu}_k(j) + \overline{\eta}_k(j), S_n(j)\}$
  $\hat{\nu}_k(j) \leftarrow \min\{\overline{\nu}_k(j), S_n(j)\}$
  $\hat{\eta}_k(j) \leftarrow \min\{\overline{\eta}_k(j), S_n(j) - \hat{\nu}_k(j)\}$
end for
$\mathcal{N}_o \leftarrow \text{Sort}\left(k \in \mathcal{N}(j) \,|\, \overline{\nu}_k(j) + \overline{\eta}_k(j) > S_n(j), \overline{R}_k(j)\right)$
while $B(j) > 0$ and $\mathcal{N}_o \neq \emptyset$ **do**
  for any $k \in \mathcal{N}_o$ and $B(j) > 0$ **do**
    if $\hat{\nu}_k(j) < \overline{\nu}_k(j)$ **then**
      $\delta \leftarrow \min\left\{\frac{\overline{P}_{k,i}(j)\overline{R}(j)}{C(j)\overline{R}_k(j)}, B(j)\right\}$
      $\hat{\nu}_k(j) \leftarrow \hat{\nu}_k(j) + \delta$
      $B(j) \leftarrow B(j) - \delta$
    else if $\hat{\eta}_k(j) < \overline{\eta}_k(j)$ **then**
      $\delta \leftarrow \min\left\{\frac{\overline{P}_{k,e}(j)\overline{R}(j)}{C(j)\overline{R}_k(j)}, B(j)\right\}$
      $\hat{\eta}_k(j) \leftarrow \hat{\eta}_k(j) + \delta$
      $B(j) \leftarrow B(j) - \delta$
    else
      $\mathcal{N}_o \leftarrow \mathcal{N}_o \setminus k$
    end if
  end for
end while

On the other hand, the gain $G(j)$ brought by WS x to the aggregate BSS throughput is given by

$$G(j) = \hat{\nu}_x(j) + \max\left\{\hat{\nu}_{AP}(j) - \left[\overline{\nu}_{AP}(j) - \overline{\nu}_{AP}^{(x)}\right], 0\right\} \quad (3)$$

where the first term on the right hand side is the (uplink) estimated throughput of x, while the second term is the estimated increase in the throughput of the AP due to the traffic it will deliver to x.

Then, to evaluate if the admission of WS x is beneficial in terms of aggregate throughput to the BSS, we define our b-metric as:

$$\text{b}(j) = G(j) - L(j) + B(j) \quad (4)$$

where $B(j) > 0$ only if $L(j) = 0$.

Note that such a metric accounts for the beneficial contribution to the aggregate throughput due to x, the possible loss experienced by the other nodes, as well as the bandwidth still available after the association of x. It follows that the b-metric clearly indicates whether the association of x will increase the total BSS throughput ($\text{b}(j) > 0$) or not ($\text{b}(j) \leq 0$); it can therefore be used as a parameter for admission control as well as for load balancing among neighboring APs. Finally, we highlight that, in view of designing an admission control scheme, the satisfaction of the throughput requirements of x, if known, could be taken into account. To this end, we could compute $\hat{\nu}_x(j)/\overline{\nu}_x(j)$ and $\hat{\nu}_{AP}(j)/\overline{\nu}_{AP}(j)$ for the uplink and downlink traffic, respectively, and verify that such quantities are above a given threshold.

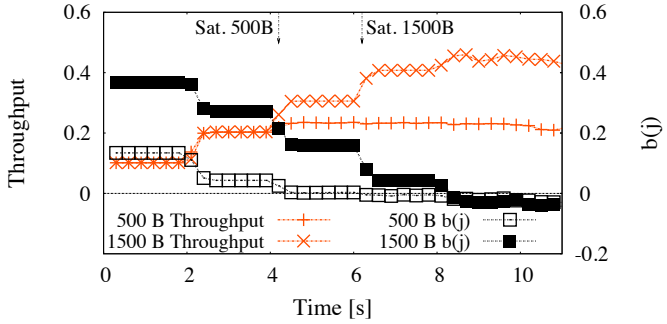


Fig. 2. Time evolution of the aggregate throughput and of the b-metric, when 6 WSs transmit packets with payload of 500 B and 1500 B.

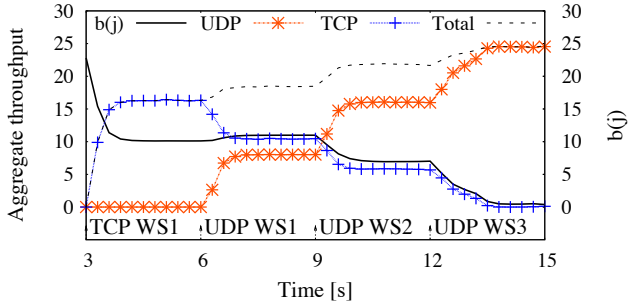
V. Performance Evaluation

We implemented our algorithm as well as the automatic data rate adaptation scheme AARF [7] in the Omnet++ v4.1 simulator. Our scenario consists of an IEEE 802.11g BSS, including one AP and a varying number of WSs. All nodes can initially transmit at 54 Mbps. The channel representation is a refinement of the ITU indoor channel model, obtained using the experimental measurements presented in [13]. As for the algorithm parameters, we set $\alpha = 0.4$ and $T_{max} = 0.1$ s. All results are averaged over 5 different simulation instances and uses TCP SACK [12].

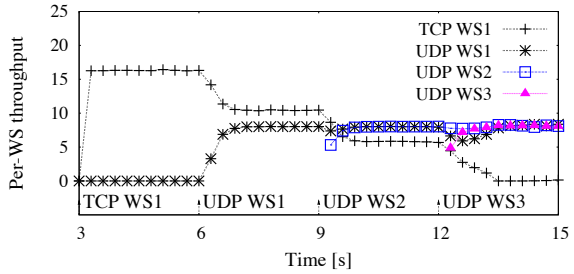
We start by considering the same settings used to highlight the inaccuracy of the idle time and of the throughput-based metrics in Fig. 1, i.e., 6 WSs transmitting packets with payload of 500 or 1500 bytes. The comparison between the plots in Figs. 2 and 1 highlights that, unlike the other metrics, our b-metric correctly reflects the behavior of the BSS aggregate throughput at the MAC layer and detects the saturation condition in all cases, approaching zero when the aggregate throughput reaches its maximum possible value.

Next, we evaluate the effectiveness of our algorithm in predicting the throughput that a WS starting one or more traffic flows can achieve, when both elastic (TCP) and inelastic (UDP) traffic are present. The latter is modeled as CBR traffic with an offered load of 8 Mbps. We fix the payload size to 1500 bytes and, for clarity of presentation, we limit our study to 3 WSs, assuming that the WS joining the BSS is not in roaming mode (thus, exact information on its traffic requirements is not available). Also, the depicted throughput is computed at the MAC layer and, for TCP traffic, it includes both data and TCP ACK packets.

We first focus on the following scenario: WS 1 starts a TCP connection at $t = 3$ s and, subsequently, a UDP flow at $t = 6$ s. The other two stations, WS 2 and WS 3, start a UDP flow at $t = 9$ s and $t = 12$ s, respectively. Fig. 3 shows the temporal evolution of the BSS aggregate throughput and of our b-metric,

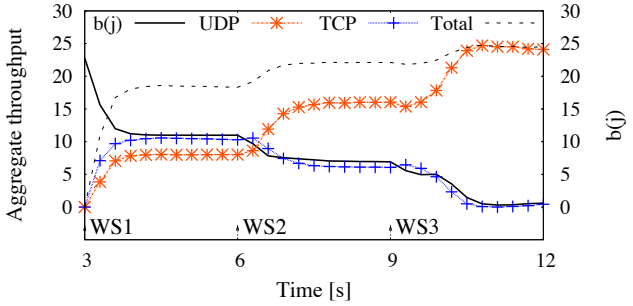


(a) Aggregate throughput and b-metric

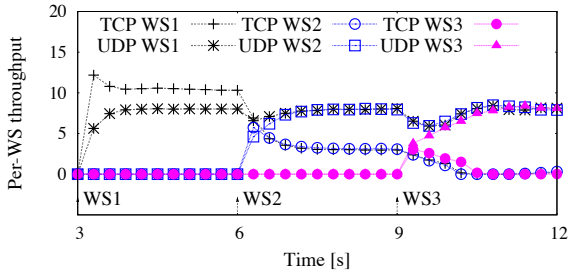


(b) Per-WS throughput

Fig. 3. WS 1 originates one TCP and one UDP flow, while WS 2 and WS 3 originate one UDP stream each. The flows become active at 3 s, 6 s, 9 s and 12 s, respectively.



(a) Aggregate throughput and b-metric



(b) Per-WS throughput

Fig. 4. Three WSs originate one TCP and one UDP flow each. The WSs become active at 3 s, 6 s and 9 s, respectively.

as well as the throughput achieved by each WS. In spite of the saturation condition caused by the TCP session started by WS 1 at $t = 3$ s, the b-metric correctly reflects that some bandwidth is available for the newly originated flow. As the UDP stream starts at 6 s, TCP adjusts its throughput and lets UDP take the desired bandwidth. Interestingly, we note that the b-metric is not significantly affected by this new condition. This is due to two reasons: (i) the UDP stream is originated by the same WS that started the TCP flow and (ii) the UDP demand is less than the estimated remaining bandwidth. The slight change that we observe in the b-metric results from the smaller number of TCP ACKs within the cycle, hence from a greater observed average payload size. Conversely, when the UDP flow of WS 2 becomes active at $t = 9$ s, the b-metric drops to 8 Mbps. The available bandwidth, though, is enough to accommodate the flow by WS 3, which starts at $t = 12$ s and brings the system to saturation, hence the b-metric drops to 0. Also, as expected, the TCP flow almost dies out after $t = 12$ s. It follows that, if the b-metric were used for admission control, the AP would admit both WS 2 and WS 3. By looking at Fig. 3(b), it is evident that this would be a good choice, as all WSs are able to meet their UDP demand.

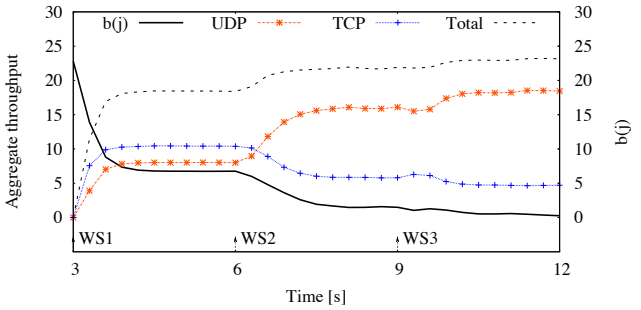
We then consider that all WSs originate one UDP and one TCP flow each, and that WS 1, WS 2 and WS 3 become active at $t = 3$, 6 and 9 s, respectively. Due to the competition between elastic and inelastic traffic within the same WS, we expect that all TCP flows will die out as the UDP streams accommodate their demand. Fig. 4 confirms such a guess showing that the time evolution of the aggregate TCP throughput matches that of the bandwidth available for inelastic traffic; again, the b-metric reflects such a behavior very closely.

At last, we consider the same settings but for the TCP flows direction: all WSs are now destinations of the TCP traffic. Fig. 5 shows that in this case the UDP throughput equals the value of offered traffic only for $t \in [3, 6]$ s, i.e., when only WS 1 and the AP are active. In this time interval, the b-metric correctly detects enough bandwidth to accommodate an 8 Mbps-traffic flow. Then, by looking at Fig. 5(b), we note that, after $t = 9$ s, both WS 1 and WS 2 suffer a loss with respect to their demand, due to the new UDP flow started by WS 3. Consistently, the b-metric in Fig. 5(a) indicates that no bandwidth was available for inelastic traffic. We point out that the throughput share of the AP, which is used for TCP traffic, erodes some of the resources available for the WSs, due to the per-packet fairness provided by the DCF.

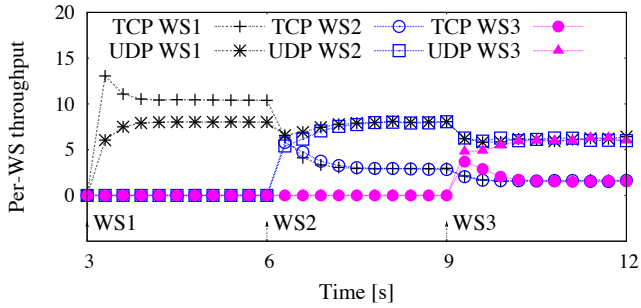
VI. Conclusion and Future Work

We designed an algorithm and a metric for bandwidth monitoring in a multirate IEEE 802.11 BSS with AP, in presence of elastic and inelastic traffic. The metric we propose can be autonomously computed by the AP, without requiring any cooperation from the wireless stations. Furthermore, the results we derived show that our solution can accurately estimate the bandwidth available for inelastic traffic, as well as the (both elastic and inelastic) throughput performance of all BSS nodes.

As future work, we will eventually evolve our metric in order to account for latency and jitter, which may affect the QoS of realtime flows, and we will evaluate the impact of short-lived TCP.



(a) Aggregate throughput and b-metric



(b) Per-WS throughput

Fig. 5. WS 1, WS 2 and WS 3 originate one UDP flow and are destinations of one TCP flow each. The WSs become active at 3 s, 6 s and 9 s, respectively.

The definition of our metric and the procedure to compute it are a stepping stone towards the design of a distributed load balancing algorithm for 802.11 networks that maximizes the total system throughput while minimizing the number of active APs. Future work will therefore address the design of such an algorithm and the study of its performance in presence of residential APs with overlapping coverages.

REFERENCES

[1] N. Cranley, M. Davis, "The effects of background traffic on the end-to-end delay for video streaming applications over IEEE 802.11b WLAN networks," *IEEE PIMRC*, pp. 1, Sept. 2006.

[2] N. Blefari Melazzi, D. Di Sorte, M. Femminella, G. Reali, "Toward an autonomic control of wireless access networks," *IEEE GLOBECOM*, pp. 954–959, Nov. 2005.

[3] A. P. Jardosh, K. N. Ramachandran, K. C. Almeroth, E. M. Belding-Royer, "Understanding congestion in IEEE 802.11b wireless networks," *ACM SIGCOMM*, pp. 25, Aug. 2005.

[4] G. Sawma, G. Ben-El-Kezadri, R. Aib, I. Pujolle, "Autonomic management for capacity improvement in wireless networks," *IEEE CCNC*, pp. 1, Feb. 2009.

[5] A. Jardosh, K. Papagiannaki, E. Belding, K. Almeroth, G. Iannaccone, B. Vinnakota, "Green WLANs: On-Demand WLAN infrastructure," *SpringerLink Mobile Networks and Applications*, vol. 14, no. 6, pp. 798–814, Dec. 2009.

[6] H. Velayos, V. Aleo, G. Karlsson, "Load balancing in overlapping wireless LAN cells," *IEEE ICC*, pp. 3833–3836, June 2004.

[7] M. Lacage, H. Manshaei, T. Turletti, "IEEE 802.11 rate adaptation: A practical approach," *ACM MSWIM'04*, pp. 126–134, Oct. 2004.

[8] C. Rossi, C. Casetti, C.-F. Chiasserini, G.Rondini, "A new metric for Admission Control in Multi-rate 802.11 WLANs," *IEEE WONS*, pp. 150–153, Jan. 2011.

[9] C. Rossi, "Computation of the BSS Saturation Throughput," *Tech. Rep.*, www.telematica.polito.it/~casetti/SaturationThroughput.pdf, 2011.

[10] G. Bianchi, "Performance analysis of the IEEE 802.11 Distributed Coordination Function," *IEEE JSAC*, vol. 18, no. 3, pp. 535–547, Mar. 2000.

[11] P. Chatzimisios, A. C. Boucouvalas, V. Vistas, "Performance analysis of the IEEE 802.11 DCF in presence of transmission errors," *IEEE ICC*, pp. 3854–3858, July 2004.

[12] TCP SACK, http://www.faqs.org/rfcs/rfc2018.html

[13] T. Chrysikos, G. Georgopoulos, S. Kotsopoulos, "Site-specific validation of ITU indoor path loss model at 2.4 GHz," *IEEE WoWMoM'09*, pp. 1–6, June 2009.

Fair WLAN Backhaul Aggregation

Domenico Giustiniano[†*], Eduard Goma[†], Alberto Lopez Toledo[†‖], Ian Dangerfield[‡¶], Julian Morillo[§¶], and Pablo Rodriguez[†]
[†]Telefonica Research, [‡]Hamilton Institute, [§]Universitat Politècnica de Catalunya
{domenic,goma,alopezt,pablorr}@tid.es, ian.dangerfield@nuim.ie, jmorillo@ac.upc.edu

ABSTRACT

Aggregating multiple 802.11 Access Point (AP) backhauls using a single-radio WLAN card has been considered as a way of bypassing the backhaul capacity limit. However, current AP aggregation solutions greedily maximize the individual station throughput without taking fairness into account. This can lead to grossly unfair throughput distributions, which can discourage user participation and severely limit commercial deployability.

Motivated by this problem, we present THEMIS, a single-radio station that performs multi-AP backhaul aggregation in a fair and distributed way, without requiring any change in the network. We implement THEMIS on commodity hardware, evaluate it extensively through controlled experimental tests, and validate it in a deployment spanning 3 floors of a multistory building. THEMIS is being used in a commercial trial by a major broadband provider to its customers.

Categories and Subject Descriptors

C.2.5 [**Computer Communication Networks**]: Local and Wide-Area Networks—*Access schemes*; C.2.1 [**Computer Communication Networks**]: Network Architecture and Design—*Wireless communication*

General Terms

Design, Experimentation, Performance.

1. INTRODUCTION

In urban environments, residential users can potentially see multiple 802.11 APs in range with high quality [1], usually connected to broadband links. As the speeds of 802.11 WLAN are typically an order of magnitude higher than those of standard broadband connections, one can use a single 802.11 wireless card to aggregate the bandwidth of multiple AP backhauls in range by virtualizing the card and cycling over the APs in a TDMA fashion. The result of such multi-AP aggregation scheme is that stations will connect to several APs in range and share their backhaul connections.

In that scenario, using an aggregation scheme like FatVAP [2], where stations greedily maximize their individual throughput, may lead to severe unfair situations. Fairness is important because it can impact individual users performance and reduce its applicability on a commercial setting. For example, a station that is unluckily located in an area with only one AP in range, can see its throughput significantly lowered by other stations sharing the same AP, even if those stations could get spare bandwidth from other APs. This is what we call *topology unfairness*. We argue that providing a fair distribution of throughput even in such heterogeneous situations is crucial to maintain a certain level of service across all users. Without some form of fairness, the perceived value of the system is severely reduced, and users will not participate.

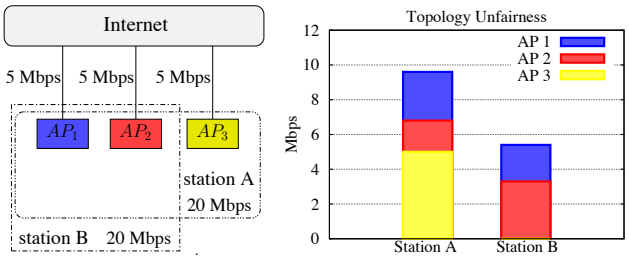
Other unfairness situations also exist. For instance, stations using applications with many TCP flows, such as P2P, can severely affect the performance of other stations running single-flow applications such as Web downloads. We call this situation *flow distribution unfairness*, and can result in some stations obtaining much less throughput than what they would obtain without sharing.

Another example of unfairness could appear in a scenario where customers with different subscription plans share their broadband links. For instance, fast broadband customers (that pay more than slow broadband customers), should obtain a greater share of the spare backhaul capacity. If this is not enforced, customers may be inclined to buy slower (and cheaper) broadband connections and free-ride on their neighbors' spare bandwidth. This is a typical "tragedy of the commons" example: people tend to over-exploit the shared resource by minimizing their contribution (their broadband contracted speed), ultimately cannibalizing the shared resource. Moreover, this eliminates the incentive of an ISP to deploy the sharing system, because it threats its business model. We call this *billing unfairness* (Section 2).

The above fairness scenarios can have a dramatic impact on the deployability of various multi-AP aggregation schemes including: a) community-based sharing schemes (e.g. FON [3], Wi-Sh [4]), b) Telco-managed sharing schemes where residential WiFi gateways are shared across all users

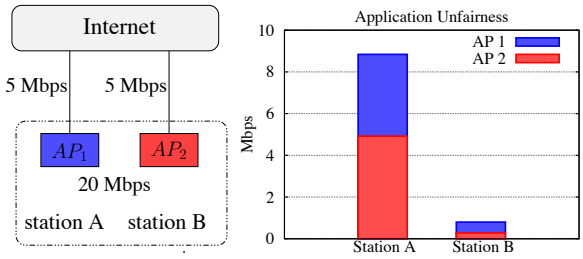
[¶]This work was carried out while the authors were at Telefonica Research Barcelona, Spain.
[*]Now at Disney Research: domenico@disneyresearch.com.
[‖]Supported by the Institució Catalana de Recerca i Estudis Avançats (ICREA).

MobiCom'10, September 20–24, 2010, Chicago, Illinois, USA.



(a) Deployment. (b) Unfair result.

Figure 1: Unfairness for users with different AP connectivity.



(a) Deployment. (b) Unfair result.

Figure 2: Unfairness for users with different number of flows.

that subscribe to the service, and c) commercial AP aggregation scenarios (e.g. airport hotspots). Moreover, existing aggregation schemes such as FatVAP [2] and VirtualWiFi [5] are not designed with fairness in mind, and hence cannot be directly applied to the above scenarios.

Motivated by these problems, we introduce THEMIS[1], a single-radio station that fairly aggregates the backhaul bandwidths of several APs. We extensively evaluate THEMIS in controlled scenarios, and show that it provides a fair distribution of the available backhaul bandwidth among users (Section 4). Finally, we validated THEMIS by emulating a typical urban neighborhood environment consisting of a setting of 10 commercial ADSLs with their correspondent 802.11g APs over 3 consecutive floors of a multistory building (Section 5).

2. FAIR WIRELESS BACKHAUL AGGREGATION

Let us consider the multi-AP backhaul aggregation system depicted in Fig. 3, where single-radio 802.11 stations simultaneously connect to one or more APs. In this scenario, the AP backhaul bandwidth of the APs is shared among the stations. Next we will give some illustrative examples which show the need for fairness and how greedy schemes, such as [2], fail.

Topology unfairness. Consider the experiment[2] depicted in Fig. 1(a), where stations A and B share 3 APs, each of them having a 5 Mbps backhaul. The wireless speed from each station to the three APs is 20 Mbps. However, because of its location, station B has only two APs in range, while station A can reliably connect to all the APs. A fair distribution of the aggregated AP backhaul would assign half of the backhaul capacity — 7.5 Mbps — to each station. Using a throughput maximization scheme as in [2], station B obtains 5 Mbps, almost half of the throughput of station A, which obtains more than 9 Mbps due to its better location (Fig. 1(b)).

Flow distribution unfairness. Consider now the experiment in Fig. 2(a), where stations A and B connect to two APs with 5 Mbps backhauls. The wireless speed between the stations and the APs is 20 Mbps. Station B starts one download per AP, each using a single TCP flow. Station A, on the other hand, starts one download per AP, but each using 10 TCP flows. The experiment is set up to guarantee that the flows are not limited by the end-to-end connection, i.e. the bottleneck is in the AP backhaul. In this scenario, a fair distribution of the AP backhaul would result in each station receiving 5 Mbps. However, if the stations aim to maximize their individual aggregate throughput without taking fairness into account as in [2], the result is a gross unfair distribution of the bandwidth, with station A receiving almost 9 Mbps, most of the aggregated bandwidth, while station B receives less than 1 Mbps (Fig. 2(b)). A similar scenario could be shown for the case of billing unfairness.

The above examples clearly illustrate the need to provide a fairness mechanism for the multi-AP backhaul aggregation scheme. However, it is important to agree on some notion of fairness, since each one could have different design implications and trade-offs. We discuss this in detail next.

2.1 What Kind of Fairness?

In order to address the unfairness situations described above, we start by describing our fairness requirements. First we would like to ensure that fairness is achieved at the level of the station's total received throughput, as opposed to individual flows or packet level fairness (**per-station fairness**). Second, we would like to ensure that users with better subscription plans (e.g. faster broadband links) obtain greater share of the aggregated AP backhaul bandwidth than users with cheaper subscription plans. Thus, in the examples above the throughput should be obtained proportionally to their priority (**weighted fairness**). Third, fairness should be enforced across all shared APs, and not just at the single AP level to ensure a fair global throughput allocation (**across-AP fairness**). Fourth, we want to provide a fairness scheme that is efficient in terms of network utilization and strikes a good balance between fairness and throughput (**efficient fairness**). And finally, we would like to provide a fairness scheme that is stable and has good convergence properties (**stable fairness**). Furthermore, in order to facilitate a wide adoption, we want to minimize the impact on the existing network infrastructure.

There are different reasons why the above requirements cannot be achieved using existing network technologies. For instance, in infrastructure mode, 802.11 does not provide per-station fairness because its downlink behavior is largely dominated by its FIFO packet-level scheduler [6]. TCP, on the other hand, only provides per-flow fairness among competing downlink flows, which is in fact the cause of the flow

[1]THEMIS is the Greek goddess of Justice, usually portrayed as an impassive blindfolded woman, holding scales outside a courthouse.

[2]All the tests and validations in this paper are performed experimentally on realistic scenarios. See Section 4 for details about the experimental setup.

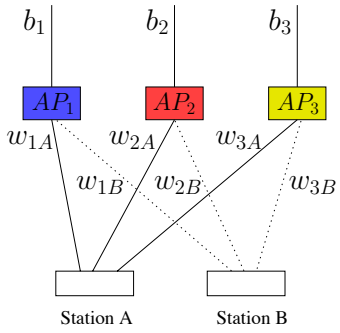


Figure 3: Multi-AP aggregation scenario.

distribution unfairness [7]. Even if one would manage to implement some fairness mechanism at the individual AP level (for example changing the FIFO behavior or introducing some clever time-based scheduler [8]), this would not result in across-AP fairness without the use of explicit signaling among the APs.

Our Choice of Fairness

In wireless systems, it is well known that fairness and throughput are often at odds [9]. For instance, imagine a scenario where two stations are sharing a wireless medium, and their wireless speeds are at a ratio of 10:1. A throughput optimal allocation would only allow the fast station to transmit, because every time slot devoted to the slow station would be wasted in low speed, losing the chance of a fast transmission. At the other extreme, a max-min fair allocation (e.g. one that maximizes the minimum of all station throughputs) would equalize the throughput transmitted by both stations. This allows the slow station to transmit most of the time, causing *performance anomaly* [9], that severely reduces the overall throughput.

Proportional fairness lies in the middle of the two extremes, providing a good compromise between fairness and efficiency (e.g. in [10]). It also achieves a good trade-off in terms of convergence and stability as shown in [11]. Finally, it allows for weighted fairness formulation. *Weighted proportional fairness* meets our efficient, stable and weighted requirements.

To comply with the other two requirements (per station and across-AP fairness), we cannot rely on existing formulations such as those in [2]. In fact [2] uses a knapsack scheduler that maximizes the individual station's throughput, and does not consider how the aggregate throughput is partitioned across stations. As a result, we need a new formulation that takes this problem into consideration. We describe it next.

2.2 Fairness Formulation

Recall the scenario depicted in Fig. 3. Let $\mathcal{S}$ be the set of stations and $\mathcal{A}$ the set of APs. Denote T_{ik} as the throughput sent from AP_i to station k. And let $y_k = \sum_{i\in\mathcal{A}} T_{ik}$ denote the total throughput received by station k. Let $U(\cdot)$ be a differentiable, strictly concave, increasing function which represents the utility at every station as a function of the received throughput. We model the fairness problem as[3]

$$\max \sum_{k\in\mathcal{S}} U(y_k) \tag{1}$$

$$\text{s. t.} \quad \sum_{k\in\mathcal{S}} T_{ik} \le b_i, \quad \forall i \in \mathcal{A}, \tag{2}$$

$$\sum_{i\in\mathcal{A}, w_{ik}>0} \frac{T_{ik}}{w_{ik}} \le 1, \forall k \in \mathcal{S}, \tag{3}$$

$$T_{ik} \ge 0, \quad \forall i \in \mathcal{A}, \forall k \in \mathcal{S}, \tag{4}$$

where w_{ik} is the wireless capacity[4] at which station k can receive from AP_i, that takes into account the interference from other clients connected to that AP, and b_i is the backhaul capacity of AP_i.

Eq. (2) is the *AP backhaul capacity constraint*, and ensures that the total traffic traversing the AP_i backhaul does not exceed the backhaul capacity b_i. Eq. (3) corresponds to the *station k wireless capacity constraint*, and guarantees that the total traffic received by station k does not exceed the total capacity of its wireless interface. Finally (4) forces the values T_{ik} to be positive.

Note that there exists an additional constraint not included in the formulation, corresponding to the *AP wireless capacity constraint*, namely $\sum_{k\in\mathcal{S}} \frac{T_{ik}}{w_{ik}} \le 1, \forall i \in \mathcal{A}$. This constraint ensures that the maximum capacity of the wireless interface at AP_i is not exceeded. However, we verified analytically that this constraint may be violated only in the extreme cases of clients severely limited by the wireless. We avoid this situation by preventing stations from connecting to *APs* if their signal-to-noise ratio (SNR) is very low. This makes sense, as a multi-AP aggregation scheme is only useful if the speed of WLAN is greater than the speed of the AP backhaul.

Finally, as described in Section 2.1, we choose a *weighted proportionally fair* utility function $U(y_k) = K_k \cdot \log y_k$, where K_k represents the relative priority of user k (for example, a value linearly dependent to the AP backhaul bandwidth owned by user k). If all the users have the same priority we use $K_k = 1$.

Decomposition and Interpretation

As described in [11], the solution of the above optimization problem can be obtained via the primal-dual formulation using a gradient descent algorithm. From there we derive the following optimal rate update rule

$$T_{ik} = \hat{T_{ik}} + \alpha \left(U'(y_k) - p_i - q_{ik} \right), \tag{5}$$

where $\hat{T_{ik}}$ is the bandwidth request in the previous step of the algorithm, $U'(y_k)$ is the derivative of the utility function evaluated at the current throughput received by the station y_k, and α is the step size of the rate update algorithm[5]. The quantities p_i and q_{ik} are the prices corresponding to

[3]For simplicity, and given that current residential traffic is heavily biased towards downloads, our formulation only considers downlink traffic. However an equivalent formulation can be designed for uplink traffic.

[4]Note that $w_{ik} = 0$ if station k does not connect to AP_i.

[5]When using proportional fairness, and in order to reduce oscillations as suggested by [12], we use $\alpha = \alpha' y_k$, with α' the new step size.

constraints (2) and (3) respectively, calculated as follows

$$p_i = \left[\hat{p_i} - \frac{\delta}{b_i}\left(\lambda b_i - \sum_{k\in\mathcal{S}} T_{ik}\right)\right]^+, \quad (6)$$

$$q_{ik} = \left[\hat{q_{ik}} - \frac{\gamma}{w_{ik}}\left(\mu - \sum_{i\in\mathcal{A}} \frac{T_{ik}}{w_{ik}}\right)\right]^+, \quad (7)$$

where $\hat{p_i}$, $\hat{q_{ik}}$ are the prices obtained in the previous step of the algorithm, and δ and γ are the step sizes of the price update algorithm. In order to improve the network utilization, and as suggested in [12], we normalize the price step size by the link capacities to favor good links. Finally $\lambda, \mu \leq 1$ are the congestion thresholds and $(x)^+ = \max(x, 0)$.

The price p_i in (6) represents the level of congestion on the backhaul of AP_i, and it is a linear function of its available bandwidth. Similarly, q_{ik} in (7) represents the level of congestion on the wireless link from station k to AP_i, and it is a function of the available card time at the station[6]. As congestion increases, the respective prices will increase and the throughput demand T_{ik} of station k through AP_i will decrease according to (5).

The values λ and μ are the *congestion thresholds*, i.e. respectively the level of utilization of the AP_i backhaul and the wireless radio-interface of station k that will trigger the algorithm congestion control. When that happens, the prices p_i and q_{ik} increase, prompting the throughput requests for their respective paths to decrease[7].

In order to distributedly solve the optimization problem in (1), each station has to periodically obtain the prices (6) and (7) for its links, and then update its rates following (5). However, implementing this algorithm locally at each station without sharing information with the APs and/or other stations has the following challenges:

- once the values T_{ik} in (5) are obtained at station k, those rates need to be enforced at AP_i (Section 3.1).
- in order to calculate the prices p_i in (6) and q_{ik} in (7), each station k needs to obtain the values of b_i and T_{ij} $j \neq k$, which are not directly available *at the station*. Moreover each station k needs to accurately know the wireless capacity w_{ik} of each AP_i (Section 3.2).
- a single-radio station has to manage the communication with multiple APs on independent radio frequencies. And it has to do it efficiently and using standard-compliant 802.11 (Section 3.3).

Addressing the above challenges in a real system requires careful design and implementation, which we describe next.

3. THEMIS

THEMIS is a single-radio wireless station based on the MadWiFi 0.9.4 driver [13] and the Click modular router 1.6.0 [14], that connects to multiple APs and aggregates their backhaul bandwidth. As shown in Fig. 4, THEMIS communicates separately to APs at different radio-frequencies using Time-Division Multiple Access (TDMA). Once connected to one AP, THEMIS transmits and receives traffic according to the 802.11 DCF protocol. The amount of time spent on AP_i is denoted *duty cycle* f_i. The constant time T that THEMIS takes to perform a standard TDMA cycle is called *wireless period*. THEMIS will use any spare duty cycle to do other operations such as AP scanning or saving energy.

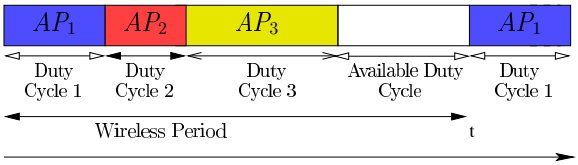


Figure 4: Time-division access to multiple APs.

3.1 Scheduler

Let us consider a THEMIS station running the optimization algorithm in (1), and calculating the request rate to AP_i to be T_{ik} in (5). In principle, in order to collect the bandwidth T_{ik} from AP_i, station k needs to connect to AP_i during a duty cycle $f_{ik} = T_{ik}/w_{ik}$, where w_{ik} is the wireless capacity from AP_i to station k. By reducing the time spent on AP_i, the duty cycle f_{ik} effectively acts as a gauge that limits the amount of bandwidth that can be received from the AP. As a consequence, TCP flows adjust their transmission rate to meet the request T_{ik}.

There are cases where station k does not receive the expected traffic T_{ik} during the duty cycle f_{ik}. There are various reasons for this discrepancy: wireless losses, congestion in the AP queue, CSMA contention delays in the wireless links, etc. We introduce a correction factor $\sigma_{ik} = T_{ik}/x_{ik}$ to account for the deviation between the expected received traffic T_{ik} and the *actual* traffic x_{ik} received by station k from AP_i during the selected duty cycle f_{ik}. As a result, THEMIS connects to AP_i for

$$f_{ik} = \sigma_{ik}\frac{T_{ik}}{w_{ik}} + c_i, \quad (8)$$

where σ_{ik} is the correction factor, and c_i is the overhead of switching from one AP to the next (see Section 3.3). Note that after applying the correction factor it may happen that the corrected duty cycles exceed the allowed time, violating the station k wireless capacity constraint, i.e., $\sum_{i\in\mathcal{A}} f_{ik} > 1$. In that case we distribute the *wireless period* proportionally among the links as described in the Appendix.

3.2 Estimators

The calculation of the duty cycle f_{ik} in (8) at station k for a given AP_i and the update of the prices p_i and q_{ik} in (6) and (7) require the following information:

- the utilization rate $\beta_i = \sum_{k\in\mathcal{S}} T_{ik}$ of the AP_i backhaul;
- the wireless capacity w_{ik}, that determines the maximum transmission rate of the wireless link; and
- the AP backhaul capacity b_i, that measures the maximum speed at which the AP_i backhaul can send traffic.

[6]The time that the card is not being used for transmitting or receiving.

[7]The values of the congestion thresholds represent a performance threshold: the closer to 1 the better if for the network utilization, but the worse is for the short-term fairness of the algorithm.

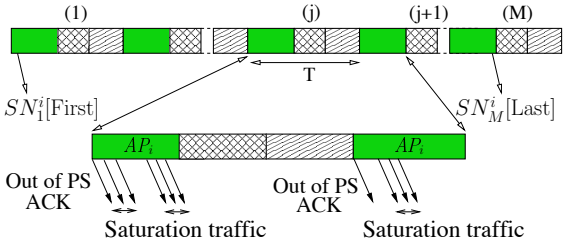


Figure 5: THEMIS estimators are based on local processing at MAC and PHY layer.

A straightforward way to obtain those values would be to introduce new signaling to exchange this information between the APs and the stations. However, that would introduce extra overhead, and would also require modifying (or replacing) the existing AP installed base. To avoid it, THEMIS estimates these values locally. Note that it is important to achieve high accuracy on the estimations, because wrong estimations would affect the performance of the scheduler. This is a hard problem because:

- *AP backhaul*: the AP backhaul is shared with other stations and any measure of the AP_i utilization rate β_i and capacity b_i must be done in the limited slice of time $f_{ik} \cdot T$ that station k dedicates to AP_i.

- *wireless link*: the wireless capacity of one AP has to be measured while the AP transmits in saturation. This is not guaranteed because the wireless link is usually not the bottleneck of the end-to-end communication.

We next describe how THEMIS estimates these values.

Utilization Rate of the AP Backhaul

The estimation of the utilization rate β_i of the AP backhaul relies on the fact that every frame sent by an 802.11 AP carries a MAC Sequence Number (SN) in the header. The SN is a module 4095 integer incremented by the AP each time a new frame is sent, and it is independent of the destination. THEMIS stations listen to the traffic sent by AP_i, and store its SNs. By counting the SNs, the THEMIS station knows the amount of packets traversing the AP_i backhaul[8]. Note that this way of counting is robust to packet loss and disconnection periods, as long as the stations do not miss more than 4095 successful frames (retransmitted frames do not increase the SN), which for an average 802.11 frame size would correspond to seconds, an order of magnitude larger than the THEMIS' TDMA period[9].

Formally, let us refer to Fig. 5. We denote SN_1^i[First] and SN_M^i[Last] the MAC sequence number of the first and last packet, respectively, sent by AP_i to any station, during a window of time $M \cdot T$, where M is an integer equal or greater than 1. Then, THEMIS derives the number of packets sent from AP_i in the time $M \cdot T$ as:

$$N^i = \big(SN_M^i[\text{Last}] - SN_1^i[\text{First}]\big) \bmod 4095.$$

Let us also denote $E[L_i]$ the average bit length per packet at IP layer over *all* the packets received by station k when it is connected to AP_i. We make the reasonable hypothesis that $E[L_i]$ does not change between the connection and disconnection time from AP_i. Finally, we calculate the AP_i backhaul utilization rate as

$$\beta_i = \frac{E[L_i] \cdot N^i}{M \cdot T}. \tag{9}$$

Wireless Capacity

THEMIS measures the wireless capacity by calculating the packet dispersion of frames directed to it when the AP is transmitting in saturation. In order to detect saturation periods, station k run-time senses the *wireless channel occupancy*, that is, the percentage of time that the channel is bus, between two consecutive received packets. These statistics are collected from specific 802.11 baseband registers, exposed by the NIC card. If the occupancy is above a certain threshold (80% in our implementation), we define the AP in saturation for that pair and store the packet length of the second packet and the dispersion between the packets. Then, referring to Fig. 5, w_{ik} is derived averaging over the window of measure $M \cdot T$ as

$$w_{ik} = \frac{\sum_{j=1}^{M} B_j}{\sum_{j=1}^{M} T_{j,SAT}^i}, \tag{10}$$

where B_j is the sum of the packet length in saturation sent from AP_i to station k and $T_{j,SAT}^i$ is the sum of the dispersions when station k receives in saturation mode during the j-th connection to AP_i. Note that w_{ik} takes into account the existing interference, and depends on the current PHY rate of APs and stations, the signal quality, and the performance anomaly [9] during the measurement period.

AP Backhaul Capacity

Several Internet services can be used to estimate the AP backhaul capacity b_i[10], some of them also provided by ISPs to their clients. Usually, a file coupled to a script is downloaded from a server. The script detects when the client has completed the download and determines b_i.

The server report may be hindered by the cross-traffic rate of the packets (eventually) being sent through the same AP_i backhaul to the other stations. THEMIS connects to a capacity server, but instead of relying on the server report, it calculates the peak reached by the utilization rate β_i during the connection time to the capacity server as

$$b_i = \max_{l=1,2,\ldots L} \overline{\beta_i[l]},$$

where L represents the number of measures during the test at the $1/(M \cdot T)$ rate, and $\overline{\beta_i[l]}$ denotes the smoothed average of $\beta_i[l]$ after the l-th calculation.

3.3 Multiple APs manager

In order to provide an *efficient* TDMA implementation in THEMIS, the wireless driver on top of the single radio interface is *virtualized*, i.e., it appears as independent Virtual

[8]Here we assume that most of the 802.11 data traffic traversed the AP backhaul, as it is often the case when using 802.11 in infrastructure mode.

[9]To increase the accuracy of the estimation, THEMIS operates in promiscuous mode, thus accounting for the information of the packets sent to other THEMIS stations. This information is never encrypted and can always be retrieved, even when the payload is encrypted.

[10]See for example `http://www.bandwidthplace.com` or `http://www.speedtest.net`.

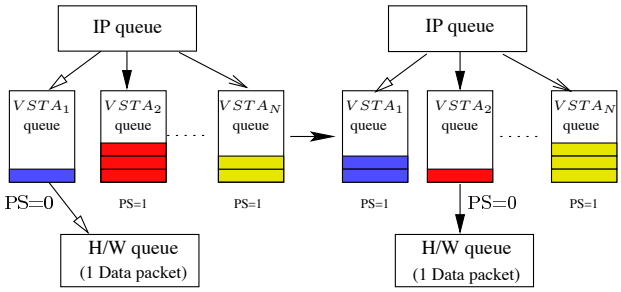


Figure 6: Queue Management.

STAtions ($VSTA_i$) associated to their respective APs. Each $VSTA_i$ is responsible for managing the data communication with AP_i and the related procedures such as association, authentication and scanning. To prevent losses during the TDMA operation, each THEMIS station k uses the 802.11 Power Save (PS) feature as follows (Fig. 6):

- During the active *duty cycle*, $VSTA_1$ exchanges traffic according to the 802.11 protocol, while the other *VSTAs* are dormant in PS mode. During the PS time, both the AP_1 and the station can only buffer packets [2, 5, 15].
- When the *duty cycle* expires, $VSTA_1$ sends a frame to inform AP_1 that it is going into PS mode. Once received the MAC ACK, $VSTA_1$ and AP_1 start to buffer the packets destined to each other.
- THEMIS assigns the control of the card to $VSTA_2$ and switches to the AP_2 radio-frequency.
- $VSTA_2$ sends a frame to AP_2 to indicate that it is ready to send/receive traffic, and awaits for the MAC ACK.
- The process continues until the station has cycled through all the *VSTAs*. The spare duty cycle can be used for other operations such as scanning or sleeping see Fig. 4). The station then restarts the TDMA cycle.

In order to minimize the switching cost c_i in (8), THEMIS achieves a fine-grained timing at MAC/PHY level, using the following techniques:

- THEMIS introduces a *MAC virtual queue* per AP. This allows to buffer packets in the MAC virtual queue, when THEMIS is selecting some other AP.
- THEMIS efficiently manages a hardware buffer (common to *all* the VSTAs) of one (1) data packet to quickly switch among MAC virtual queues. This is a challenging task, because short H/W queues cause inefficiencies that negatively affect throughput (as a comparison, the original MadWiFi driver sets the H/W queue size to 200)[11].
- In order to switch the PS state, THEMIS piggybacks the MAC PS bit on the header of the pending data on top of the MAC virtual queue. THEMIS reverts to the classical use of probes (as done in [2, 5]) in the rare event of not having data packets ready for transmission.

[11]Packets in the hardware queue must be sent before the end of the *duty cycle* assigned to the *VSTA*. This causes a delay respect to the expected end of the duty cycle imposed by the THEMIS scheduler. The efficient management of a hardware buffer of size one minimizes any extra-delay.

With the techniques described above, THEMIS incurs in a switching cost c_i of about 1.2-1.5 ms, most of which (around 800 μsec) is spent in hardware radio-channel commutation. This limited overhead, significantly less than [2,5], increases the stability of the system by reducing the jitter in the switching procedure. This enables a fine-grained selection of duty cycles assigned by the scheduler even if the station transmits in saturation mode, which is of particular importance for TCP traffic.

On top of the MAC implementation, THEMIS uses a *flow mapper* to assign new TCP flows from the upper layers to a specific *VSTA*. While we could use a more sophisticated flow mapper, we employed a proportional based mapper as in [2]: the amount of traffic r_{ik} assigned to AP_i maintains the proportions of the bandwidth obtainable from each AP and equal to $r_{ik}=\frac{f_{ik}w_{ik}}{\sum_j f_{jk}w_{jk}}$.

Finally, THEMIS implements a Reverse-NAT module that i) makes sure that the packets leave the station with the correct source IP address (i.e. the one corresponding to the outgoing *VSTA*, as assigned by the AP); and ii) presents a consistent (dummy) IP to the applications, providing IP transparency to higher layers.

4. VALIDATION

We evaluate THEMIS in an extensive set of tests. Our findings show that

- the estimators described in Section 3.2 are accurate, and stations do not need to request information from the network.
- THEMIS achieves a fair sharing of the aggregate network capacity among stations, while efficiently using the aggregated network capacity.

In our experiments, the APs are off-the-shelf Linksys, running Linux DD-WRTv24 firmware. The stations are Linux laptops, equipped with a single-radio Atheros-based wireless NIC. For every AP and station in the network, the wireless multimedia extensions (WME) and the RTS/CTS handshake are disabled. Any non-standard compliant 802.11 feature is also disabled, and H/W queues are set up with 802.11 best effort parameters.

4.1 Evaluation of THEMIS Estimators

We first verify the accuracy of the estimators used by THEMIS. We start studying the estimation of the backhaul utilization rate β_i in a test where 3 THEMIS stations download HTTP files using 3 Mbps lines. Stations are connected to the AP using a fixed connection time of 25 ms over a period of 100 ms. Stations are not synchronized, and they connect to the corresponding APs at independent times. Consequently, stations can only observe a fraction of the traffic load sent to other stations. Moreover, because of the wireless nature, they may not receive some packets sent to other stations, missing information such as the sequence number SN and packet length L_i needed by the estimator in (9).

In this configuration, we compare the estimations of the backhaul utilization rate over the time at each THEMIS station with the actual rate measured at the AP. The results in Fig. 7 show that all stations obtain a very accurate estimation.

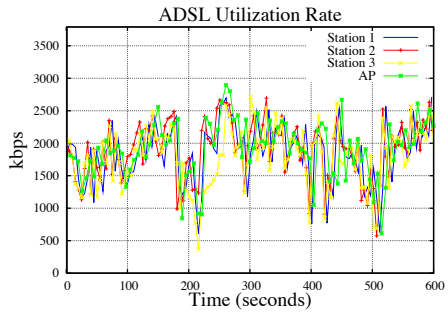


Figure 7: AP backhaul utilization estimation (β_i).

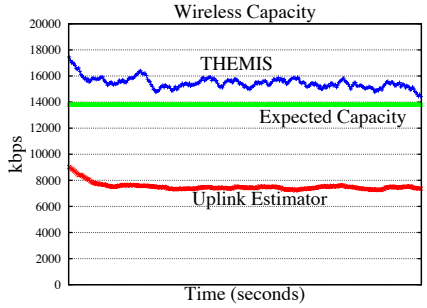


Figure 8: Wireless capacity estimation (w_{ik}).

We next evaluate the THEMIS' wireless capacity estimator described in Section 3.2. In the test, the THEMIS station connects to an AP with a duty cycle of 25 ms over a period of 100 ms, and performs several HTTP downloads from different Internet servers. Fig. 8 shows the estimation of w_{ik} in a period of 4 minutes. THEMIS estimator gives a good approximation (around 13.7 Mbps) of the speed reported with a downlink Iperf test from a server located in the same LAN of the AP.

Estimators of w_{ik} are also proposed in [2, 16]. However, these estimators are based on the time needed to transmit a packet from the 802.11 station, and so they better represent uplink speeds rather than downlink. This can result in severe errors in the estimation of the downlink wireless capacity. As an example, Fig. 8 shows the performance of the estimator in [2] for the same scenario, and we observe that it under-estimates the wireless capacity. In fact, a high downlink speed will cause a long air-time before transmitting a packet in uplink, that translates in a low (and erroneous) downlink wireless capacity estimation.

4.2 System Evaluation

We now evaluate the system implementation of THEMIS through different tests. For every scenario, we run five tests of 1800 secs and plot the average results obtained. We choose such a configuration to verify that results are stable in time and across different tests. To achieve independent tests, stations are configured so that the THEMIS estimators are reset after each test. For the transport layer, we use Linux standard TCP Reno with SACK and delayed ACK option enabled and we emulate the AP backhaul capacities using the *tc* Linux traffic shaper. Unless otherwise stated: i) we open a TCP flow per AP using *iperf*, ii) the AP backhaul capacity is known at each station k, while the ADSLs utilization rates $\{\beta_{ik}\}$, and the wireless capacities $\{w_{ik}\}$ are estimated at THEMIS station k as described in Section 3.2.

	station A	**station B**
802.11 Legacy	0.45 Mbps	6.24 Mbps
THEMIS	3.15 Mbps (f=0.19)	3.40 Mbps (f=0.15)

Table 1: Two stations connected to one AP.

THEMIS parameters

Selecting the appropriate wireless period represents a complex trade-off. On one side, switching among APs introduces overhead, so selecting long *wireless periods* reduces the overhead. However, long periods affect TCP performance because they artificially increase the end-to-end delay. On the other hand, short periods are desirable, as they reduce the disconnection time from the APs in PS mode, and prevent TCP from timing-out. In order to meet all the previous requirements, we select a *wireless period* T of 100 ms. The scheduler and estimators are updated every $20 \cdot T = 2$ seconds. We also impose that the time of connection to each AP is at least equal to the switching cost plus 2 ms (that gives a minimum *duty cycle* $f_i \geq 0.03$).

The values of α (5), δ (6) and γ (7) have been selected based on extensive simulations, with values that provide a good trade-off between convergence and stability. Similarly, we choose the congestion thresholds for the AP backhaul and the wireless capacity to be λ=0.95 and μ=0.95 respectively. A more detailed sensitivity analysis of the parameters falls outside the scope of this paper.

Two stations connected to one AP

We first consider the configuration where two stations are connected to the same AP (802.11 legacy operation). In the test, we consider that both stations receive traffic from the AP at a downlink wireless rate of about w_1=20-22 Mbps and are connected to an AP backhaul of b_1=7 Mbps[12]. We also consider that station A opens one TCP flow per AP while station B opens 10 TCP flows per AP.

The results are summarized in Table 1. With legacy 802.11, station B uses most of the backhaul capacity with an average received throughput of 6.24 Mbps while station A starves at 0.45 Mbps, at a throughput more than 13 times smaller than station A. On the other hand, each THEMIS station connects for a limited percentage of card time on each AP to collect the requested bandwidth T_{1k}. The result is that station B — that opens more flows — connects less time than A, i.e. 14% versus 19% of their time, and then for just a few ms of the entire wireless period. Indeed station B needs in average less time to achieve the bandwidth from the AP, because it is less affected by the TCP's sawtooth behavior of each flow. As a result, station A and B obtain similar throughput (3.15 Mbps vs 3.40 Mbps), with a network utilization of 6.55 Mbps instead of 6.69 Mbps, a consequence of the THEMIS congestion control.

[12]This is the AP backhaul capacity, and hence the actual speed available for TCP traffic may be lower. In fact, because of TCP's sawtooth behaviour, not all the *available bandwidth* at the bottleneck may be used at any time. The bandwidth utilization per path can increase establishing more than one TCP connection over each AP.

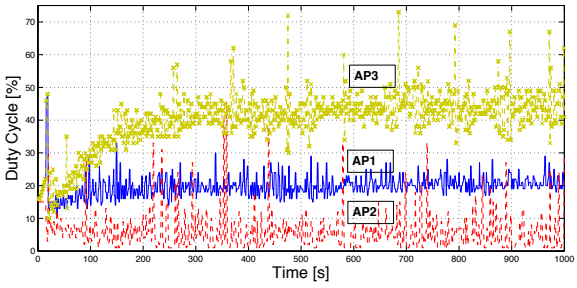
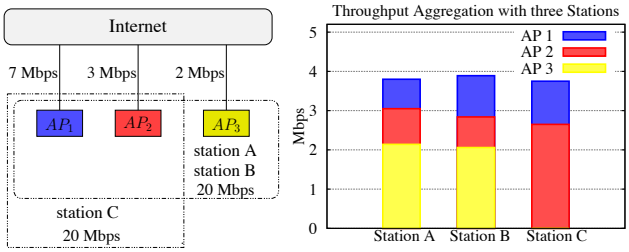


Figure 9: Duty Cycles evolution with one station.



(a) Deployment. (b) Results.

Figure 10: 3 THEMIS stations sharing 3 APs.

	AP_1	AP_2	
Capacity	b_1=5 Mbps	b_2=5 Mbps	b_2=2.73 Mbps
Duty Cycle	0.25	0.71	0.67
Throughput	4.74 Mbps	2.43 Mbps	2.21 Mbps

Table 2: Connection to two APs, one wireless bottleneck.

One station connected to Multiple APs

In these tests we evaluate the efficiency in terms of network utilization with one THEMIS station connected to two APs. Let us first consider the case where the throughput is not limited by the wireless card speed on any of the connections, i.e. the expected result is to completely utilize the available backhaul capacity of the three APs. Here, station A is associated to 3 APs, at a downlink wireless rate of about w_1=w_2=w_3=20 Mbps and is connected to AP backhauls of b_1=5 Mbps, b_2=1 Mbps and b_3=10 Mbps respectively, with a total bandwidth of 16 Mbps.

As we can see in Fig. 9, the duty cycles converge to stable range of values. THEMIS spends most of the time on the best network path (via AP_3) and less time on the worst network path (via AP_2). This results in a total aggregated throughput of 15.05 Mbps, that is with an average utilization of 94% of the network aggregated capacity, as we expect from the setting of λ=0.95.

We then consider a scenario where a THEMIS station connects to two APs, and is limited by the wireless speed on one link. In the test, a THEMIS station measures a downlink wireless capacity of w_1=20.74 Mbps on AP_1 and w_2=2.73 Mbps on AP_2 and is connected to AP backhauls of 5 Mbps each, bottlenecked by the wireless on path 2.

Results are summarized in Table 2. We consider two settings: first, the ideal case where the AP backhaul capacities are correctly estimated at 5 Mbps, and second, the most realistic scenario where the estimation of the AP backhaul capacity of the path limited by wireless (path 2) is bottlenecked by the wireless capacity b_2=w_2=2.73 Mbps.

In the first case, THEMIS spends f_1=0.25 on the path with higher wireless speed, obtaining a throughput of 4.74 Mbps. The rest of the time it is spent in the path limited by the wireless link (f_2=0.71), where it achieves a throughput of 2.43 Mbps (for an aggregated 7.17 Mbps). Note that a small time (f=1-0.25-0.71=0.04) is used by THEMIS to detect card time congestions as shown in (7).

In the second case, the throughput achieved on path 2 slightly reduces to 2.21 Mbps, with a sub-utilization of the path of 2.43-2.21=0.21 Mbps. In fact, a smaller (and wrong) AP backhaul capacity estimation causes a higher AP backhaul price p_2 on the link, that in turn causes the station to request less throughput on this connection according to (5). This translates in a smaller duty cycle f_2=0.67 rather than 0.74, that in turns reduces the bandwidth received on this path.

In both tests, THEMIS makes an efficient usage of the network: the overall throughput is higher than the one obtained being connected 100% of the time to AP_1 (at most 5 Mbps) or to AP_2 (at most w_2=2.73 Mbps).

Multiple Stations connected to Multiple APs

We evaluate the fairness and the network utilization efficiency, when different stations are connected to multiple APs. First, we analyze the case of 3 THEMIS stations, in the scenario in Fig. 10(a), with 3 APs with backhaul speeds of b_1=7 Mbps, b_2=3 Mbps and b_3=2 Mbps respectively, resulting in a total aggregated capacity of 12 Mbps. Given that none of the stations is limited by the wireless links, each station is expected to get an average aggregated speed close to 12/3=4 Mbps, even if the stations share a different number of APs.

This is demonstrated in Fig. 10(b): the 3 stations obtain a fair share of the aggregate AP backhaul speed, averaging 3.80 Mbps, 3.89 Mbps and 3.75 Mbps on station A, B and C, respectively, for a total aggregate throughput of 11.44 Mbps, again around the 95% of the overall available capacity.

Then we consider the scenario in Fig. 11(a), where station B shares two AP backhauls with station A at wired speeds of 5 and 1 Mbps, respectively. Station A can also connect to a third AP (AP_3) with a backhaul speed of 10 Mbps. Then, station B can obtain *at most* 6 Mbps and can never reach the 10 Mbps speed of AP_3 backhaul.

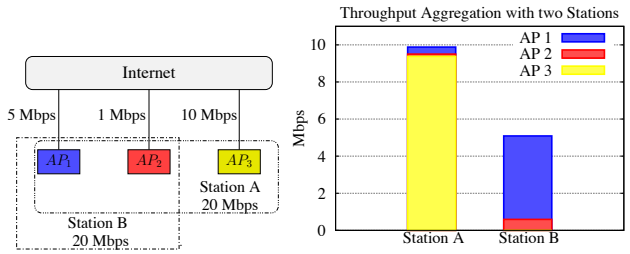
The results in Fig. 11(b) show a total aggregate TCP throughput of 9.88 Mbps on station A (with f_1=0.08, f_2=0.05 and f_3=0.47), and 5.09 Mbps on station B (f_1=0.28, f_2=0.09). Station A makes the fair decision, reducing the amount of time connected to the shared APs as much as possible.

Stations with an uneven number of TCP flows

Let us recall the flow distribution unfairness example shown in Fig. 1(a) (Section 2) where two stations are sharing two 5 Mbps backhaul APs and use an uneven number of TCP flows. Fig. 12 shows that THEMIS is able to guarantee a fair share of the aggregated backhaul capacity to each station.

Stations with different priorities

Consider the same scenario as before, where now Station A and Station B happen to be roaming and sharing two 5



(a) Deployment. (b) Results.

Figure 11: Two stations sharing partially overlapping sets of APs where station B cannot obtain the throughput obtained by station A.

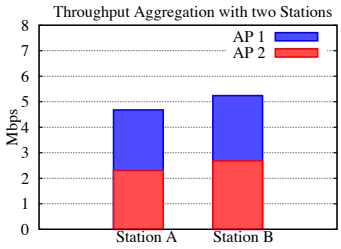


Figure 12: Fair share of the backhaul bandwidth using THEMIS.

Mbps AP backhauls. Let us consider that Station A belongs to a user that has higher priority than the user of Station B. For sake of illustration, we suppose that THEMIS applies weighted proportional fairness using $K_A = 4$ and $K_B = 1$. Therefore, it is expected that Station A obtains $K_A/(K_A + K_B) = 0.8$ of the total bandwidth while Station B obtains the remaining $K_B/(K_A+K_B) = 0.2$. The experiments show that THEMIS stations obtain a throughput of 7.64 Mbps for Station A and 2.0 Mbps for Station B.

5. THEMIS IN THE WILD

In order to test the scalability of THEMIS, we deploy a realistic testbed spanning three floors of a multistory building. The network consists of 10 commercial ADSLs with their corresponding WLAN APs and 10 THEMIS stations, i.e. the owners of each line. Nine of the ADSL lines have a nominal capacity of 3 Mbps and one has a nominal capacity of 1 Mbps. The APs are distributed every 80 square meters to emulate the average residential flat size (see Fig. 13) and are set to independent radio-frequencies in the 2.4 GHz ISM band[13].

In the bootstrap phase, the APs are selected based on a passive analysis of the SNRs of the 802.11 AP beacons with "THEMIS" essid. Stations scan for the APs in range and start authenticating and associating to the AP with highest SNR sequentially associating to APs with smaller SNR. THEMIS requires a minimum SNR of 10dB to guarantee a stable reception at 1 Mbps PHY basic rate. In each test, automatic rate selection is active in each THEMIS station, with independent instances of the Minstrel rate selection algorithm [17] over each wireless uplink.

[13]The channels optimization is out-of-the-scope of this paper.

Figure 13: Testbed deployment. APs and stations have been deployed over 3 floors, ground floor (on the left), mezzanine (in the middle), and first floor (on the right). Each circle represents an AP, while stations are placed nearby the APs, one station per AP. Only stations A, B and C, relevant for some experiments, are shown in the map. Obstacles, as walls and desks are presented between all the AP links.

5.1 Characterization

We measure the capacity of each link of the network (i.e. the ADSLs and the 10×10 wireless links). Our findings are that the 3 Mbps lines offer a constant maximum speed of 2.65 Mbps and the 1 Mbps line offers 0.89 Mbps. Regarding the wireless links, apart from the 10 "home" links where the station is located nearby the AP, the SNR measured per wireless link is consistently lower than 30 dB.

We then generate traffic from a server connected to the APs via an 802.3 LAN, activating one AP-station link at a time, with 5 minutes dedicated to each test. We calculate the average throughput and the standard deviation for each link. Then, we re-order the 10 links in descending order per-station, based on the average throughput.

Results are reported in Fig. 14. Each station can receive TCP traffic from at least 3 APs (and up to 5) at a speed higher than 10 Mbps. The results show the feasibility of aggregating the low-speed backhaul bandwidth of at least three APs.

5.2 The Effect of Location

To show the effect of location, we perform a test in which two stations (Station A and Station B as shown in Fig. 13) initially share the same set of APs and are located a few meters away from its "home" AP. For this test we use three APs connected to 3Mbps lines, hence, the total backhaul capacity that Station A and Station B share is $2.65 \times 3 = 7.95$ Mbps. As a result, a fair share of the total bandwidth would be $7.95/2 = 3.975$ Mbps per station. Both stations perform several HTTP downloads per AP during 2400 seconds. After 1200 seconds of test, Station B moves to a second location from which it can only be connected to two of the former APs. As we do not implement IP mobility in our testbed, all the connections of Station B are dropped and started again in the new location. As a consequence of the movement of Station B, the topology of the network changes and stations observe an uneven AP backhaul capacity.

We run the test using a throughput maximization algorithm as in FatVAP [2][14] (Fig. 15(a)), and using THEMIS

[14]We implemented the throughput maximization algorithm according to the description in [2]. To provide a fair compar-

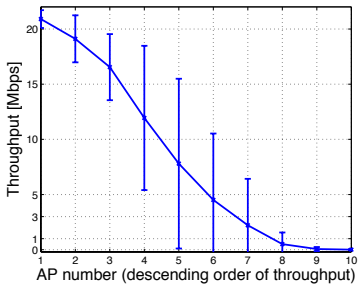


Figure 14: Wireless link quality assessment.

(Fig. 15(b)). Results show that, when the network topology is similar for both stations (they are both connected to 3 APs at similar speed), using throughput maximization results in a similar long-term performance for both stations, but with no guarantee of short-term fairness. Moreover, when the topology changes, Station B is clearly penalized by its new unlucky location obtaining 2.8 Mbps while Station A obtains 4.8 Mbps.

On the other hand, THEMIS guarantees a fair share of the backhaul capacity in both topologies, offering 3.5 Mbps to each station. Note that when Station B moves to the new position, the PHY rate is quickly reduced because of the lower signal strength, with THEMIS quickly converging to a fair assignment of the backhaul capacity. Also note that because the fairness mechanism relies on the congestion thresholds λ and μ (Section 2.2), the network utilization is slightly lower than the optimal.

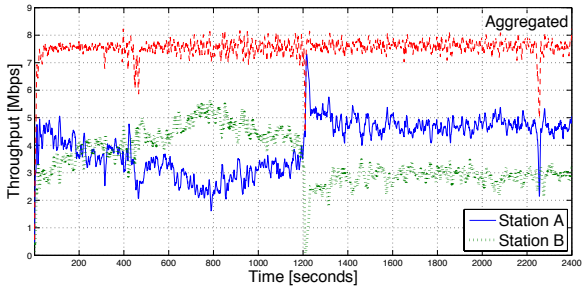
5.3 Integrated Operations

We have shown via different deployments that THEMIS is able to deal with the three types of unfairness that arise when aggregating AP backhaul bandwidth. However, in a real life scenario, these unfairness can take place at the same time. Thus, we perform a test that evaluates THEMIS in presence of a P2P station (Station A), an unluckily located station (Station B) and a low priority station (Station C). The location of the stations is shown in Fig. 13. For this test we use three APs, each with a 3 Mbps backhaul. The P2P and the low priority stations are connected to 3 APs while the unluckily located station is connected to 2 APs. Given that the low priority station owns a 1 Mbps ADSL while the others own a 3 Mbps ADSL line, the weights have been set to $K_A = K_B = 3$ and $K_C = 1$. In such experiment, a fair system should be able to allocate the bandwidth proportionally to the priority of the users.

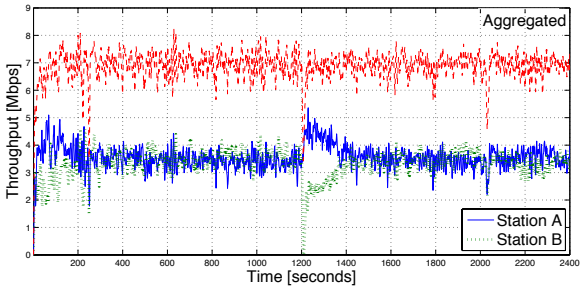
At the beginning of the test, Station A starts downloading P2P traffic from the three APs. After 1200 seconds, Station B starts several HTTP downloads from the two APs it is connected to. Finally after 1200 seconds more, Station C also starts HTTP traffic from the APs.

The result of using a throughput maximization algorithm is shown in Fig. 16(a). It is noticeable that Station A, due to the high number of TCP flows opened by P2P applications, obtains most of the backhaul capacity preventing Station B from obtaining its fair share of the bandwidth. Furthermore, when Station C starts its downloads, the absence of priority among users further reduces the throughput obtained by Station B, introducing billing unfairness. Finally, since Station B and C achieve a similar throughput despite that station B is unluckily located, the flow distribution unfairness dominates over the topology unfairness.

ison with THEMIS, we use the wireless capacity estimation of THEMIS and the APs manager described in Section 3.3.



(a) FatVAP



(b) THEMIS

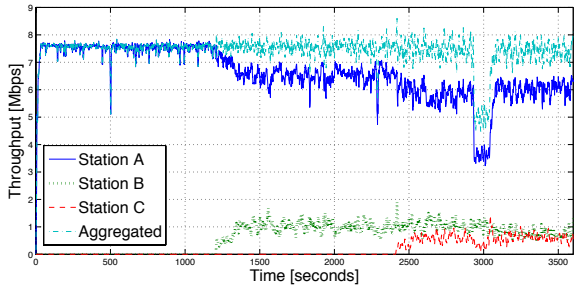
Figure 15: Assessment of the topology unfairness in the residential-like deployment.

The result of using THEMIS is shown in Fig. 16(b). When the unluckily located station B starts its downloads after 1200 seconds, the wireless capacity measured at station A over the shared APs is reduced because of the performance anomaly [9]. However, the system quickly adapts: the wireless links with lower wireless capacity receive a higher wireless price q_{ik} and hence smaller throughput demand T_{ik} and dedicated duty cycle f_{ik}. A smaller duty cycle for both station A and B means that the probability of being connected to the same AP at the same time, and consequently the occurrence of performance anomaly, is reduced. Concluding, THEMIS offers a fair share of the aggregated bandwidth to both stations, while providing a high usage of the backhaul bandwidth. Finally, when Station C starts its downloads, the priorities are preserved and Station A and B obtain a greater share of the backhaul capacity.

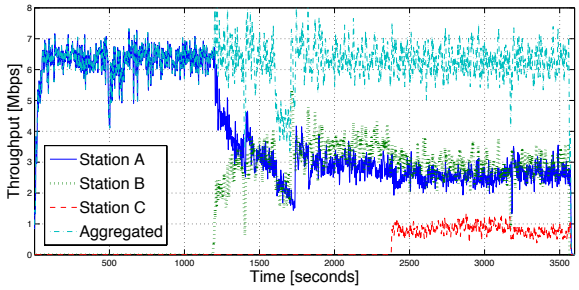
6. RELATED WORK

In recent years Wi-Fi communities have attracted the attention of both the research community and the wireless industry because of the uptake of WLAN in residential areas. In this direction [3, 4, 18] propose to allow members of the communities to share the backhaul bandwidth of their WLAN APs. Among those, Wi-Sh [4] discusses the fairness problems that can arise from sharing resources. However, it does not consider the use of multiple APs to aggregate their backhaul bandwidth.

Backhaul bandwidth aggregation has been explored in



(a) FatVAP



(b) THEMIS

Figure 16: Test with the effect of the three types of unfairness: Station A uses P2P traffic, Station B is unluckily located (starts after 1200s) and Station C is a low priority station (starts after 2400s).

[19, 20], where stations connect to their home APs via ethernet and to the remote APs using WLAN. However, they do not connect to multiple APs via WiFi, so the number of APs they can aggregate is limited by the number of physical interfaces (ethernet and WiFi) available at the stations.

The idea of connecting to multiple APs through a single radio was first shown in VirtualWiFi [5]. The authors rely on the WLAN standard power saving (PS) mode to switch among different Wi-Fi nodes in time division. Switching among Wi-Fi nodes is transparent to the applications, but at a high cost in time (30-600 ms). In fact, VirtualWiFi implements the code on top of the driver card with a MAC instance for connection.

Within the problem of single radio AP backhaul aggregation, the closest work is FatVAP [2]. The authors introduce a scheduler to select the percentage of time to spend on each AP to maximize the aggregate throughput at each station. However [2] has a limited focus because it does not resolve the unfairness across stations, and it only considers stations connected to (strictly) more than one AP. Furthermore, the local throughput maximization approach in [2] can not be extended in order to take into account priority-based per-station fairness. Compared to [2], THEMIS fairly aggregates the AP backhaul bandwidth among the different THEMIS stations, irrespectively of their location, link quality and number of APs they have in range. Moreover, THEMIS is able to adapt to different fairness objectives in order to accommodate the different scenarios discussed in Section 1, and it achieves this in a completely distributed manner. Finally, THEMIS implementation of the single-radio multi-AP TDMA access is improved compared to [2, 5], reducing the frequency switching overhead and increasing the accuracy when selecting the amount of time that the station connects to the different APs. This results in a more efficient operation and increased throughput.

Among other work, [21] introduces a support for a seamless hand-off between WLAN APs. In [15], standard solutions have been exploited to increase the aggregate throughput observed by a single station with respect to the design in [2, 5, 21]. However, these works do not consider the problem of fairness.

Link-alike [22] tackles the problem of minimizing the uplink total transfer time via multiple wireless links. However, the solution requires cooperation among the APs, with 802.11 APs transmitting and receiving at the same radio-frequency, and a custom TCP protocol over the wireless link.

Several tools have been designed to estimate the available bandwidth along a network path. However, these tools typically send active probes along a path and/or require a cooperative implementation at both the sender and receiver [23, 24].

There is little work on non-cooperative estimation of the ADSL available bandwidth. Most notably ABwProbe [25] and FAB-Probe [26] rely on the asymmetry of the ADSL downlink capacity to send TCP ACK packets of different sizes and receive small TCP RST packets from the TCP client. Since the TCP RST is at fixed length, they cannot estimate the available bandwidth from the client-side, as done by THEMIS.

The estimation of the wireless capacity has been studied with different levels of accuracy (see e.g. [2, 27]). A comparison with the implementation of THEMIS has been provided in Section 4.1. Our experimental evaluation has demonstrated that the robustness of THEMIS in realistic scenarios, under MAC contention, adaptive PHY rates and performance anomaly.

7. CONCLUSION AND FUTURE WORK

We have shown that fairness is a crucial factor for the success of multi-AP aggregation schemes. Without fairness, the perceived value of the system is severely reduced, eliminating the incentives of users to participate, and of providers to deploy it. This effectively renders such aggregation schemes unfeasible. In order to achieve fairness, existing multi-AP aggregation systems that maximize the throughput of single users cannot be extended. As a consequence a complete re-design of the system is required.

To address this problem we introduced THEMIS, a single-radio station implemented in commodity hardware that is fair in a multi-AP aggregation scenario. THEMIS operates locally at each station, using standard 802.11, without requiring any change in the network. This makes THEMIS ready to be deployed. In fact, THEMIS is being used by a major broadband provider in a commercial trial.

There are several interesting future lines for this work. On the technical side, we plan to extend THEMIS to include uplink traffic in the formulation, and investigate the impact and trade-offs that TDMA may have over the TCP performance. From an architectural point of view, we are currently exploring the use of THEMIS to design more power efficient access networks. Finally, it would be interesting to understand how THEMIS can be leveraged to perform efficient large-scale cellular data offloading, which appears to be a difficult challenge for the years to come.

8. REFERENCES

[1] D. Han, A. Agarwala, D. Andersen, M. Kaminsky, D. Papagiannaki, and S. Seshan, "Mark-and-Sweep: Getting the 'Inside' Scoop on Neighborhood Networks," in *Proc. of the ACM IMC Conf.*, (Vouliagmeni, Greece), pp. 99–104, October 2008.

[2] S. Kandula, K. Lin, T. Badirkhanli, and D. Katabi, "FatVAP: Aggregating AP Backhaul Capacity to Maximize Throughput," in *Proc. of the USENIX NSDI Conf.*, (San Francisco, CA), pp. 89–04, April 2008.

[3] "FON." `http://www.fon.com/`.

[4] X. Ai, V. Srinivasan, and C.-K. Tham, "Wi-Sh: A Simple, Robust Credit Based Wi-Fi Community Network," in *Proc. of the IEEE INFOCOM Conf.*, (Rio de Janeiro, Brazil), pp. 1638–1646, April 2009.

[5] R. Chandra and P. Bahl, "MultiNet: Connecting to Multiple IEEE 802.11 Networks Using a Single Wireless Card," in *Proc. of the IEEE INFOCOM Conf.*, (Hong Kong, China), pp. 882–893, March 2004.

[6] E. Park, D. Kim, H. Kim, and C. Choi, "A Cross-Layer Approach for Per-Station Fairness in TCP over WLANs," *IEEE Trans. Mobile Comput.*, vol. 7, no. 7, pp. 898–911, 2008.

[7] B. Briscoe, "Flow Rate Fairness: Dismantling a Religion," *ACM SIGCOMM Comp. Commun. Review*, vol. 37, no. 2, pp. 63–74, 2007.

[8] G. Tan and J. Guttag, "Time-based Fairness Improves Performance in Multi-rate WLANs," in *Proc. of the USENIX Annual Tech. Conf.*, (Boston, MA), pp. 23–24, June 2004.

[9] F. R. G. B.-S. M. Heusse and A. Duda, "Performance Anomaly of 802.11b," in *Proc. of the IEEE INFOCOM Conf.*, vol. 2, (San Francisco, CA), pp. 836–843, April 2003.

[10] H. J. Kushner and P. A. Whiting, "Convergence of Proportional-Fair Sharing Algorithms under General Conditions," *IEEE Trans. Wireless Commun.*, vol. 3, pp. 1250–1259, 2003.

[11] R. Srikant, *The Mathematics of Internet Congestion Control (Systems and Control: Foundations and Applications)*. Springer Verlag, 2004.

[12] M. P. W. Wang and S. H. Low, "Optimal Flow Control and Routing in Multi-path Networks," *Elsevier Perform. Eval.*, vol. 52, no. 2-3, pp. 119–132, 2003.

[13] "Madwifi project." `http://madwifi-project.org`.

[14] E. Kohler, R. Morris, B. Chen, J. Jannotti, and F. M. Kaashoek, "The Click Modular Router," *ACM Trans. Comput. Syst.*, vol. 18, pp. 263–297, August 2000.

[15] D. Giustiniano, E. Goma, A. Lopez Toledo, and P. Rodriguez, "WiSwitcher: An Efficient Client for Managing Multiple APs," in *Proc. of ACM PRESTO Wrkshp.*, (Barcelona, Spain), pp. 43–48, August 2009.

[16] D. Malone, I. Dangerfield, and D. J. Leith, "Verification of Common 802.11 MAC Model Assumptions," in *Proc. of the ACM PAM Conf.*, (Louvain-la-Neuve, Belgium), pp. 63–72, April 2007.

[17] "Minstrel rate control algorithm." `http://linuxwireless.org/en/developers/Documentation/mac80211/RateControl/minstrel`.

[18] "Whisher Wifi Sharing Community." `http://www.whisher.com`.

[19] E. Tan, L. Guo, S. Chen, and X. Zhang, "CUBS: Coordinated Upload Bandwidth Sharing in Residential Networks," in *Proc. of the IEEE ICNP Conf.*, (Plainsboro, NJ), pp. 193–202, October 2009.

[20] N. Thompson, G. He, and H. Luo, "Flow Scheduling for End-host Multihoming," in *Proc. of the IEEE INFOCOM Conf.*, (Barcelona, Spain), pp. 1 – 12, April 2006.

[21] A. Nicholson, S. Wolchok, and B. Noble, "Juggler: Virtual Networks for Fun and Profit," *IEEE Trans. Mobile Comput.*, vol. 9, no. 1, pp. 31–43, 2010.

[22] S. Jakubczak, D. G. Andersen, M. Kaminsky, D. Papagiannaki, and S. Seshan, "Link-alike: Using Wireless to Share Network Resources in a Neighborhood," *IEEE Trans. Mobile Comput.*, vol. 12, no. 4, pp. 1–14, 2008.

[23] D. K. J. Strauss and F. Kaashoek, "A Measurement Study of Available Bandwidth Estimation Tools," in *Proc. of the ACM IMC Conf.*, (Miami Beach, FL), pp. 39 – 44, October 2003.

[24] V. Ribeiro, R. Riedi, R. Baraniuk, J. Navratil, and L. Cot, "pathChirp: Efficient Available Bandwidth Estimation for Network Paths," in *Proc. of the ACM PAM Wrkshp.*, (San Diego, CA), April 2003.

[25] D. Croce, T. En-Najjary, G. Urvoy-Keller, and E. W. Biersack, "Non-cooperative Available Bandwidth Estimation Towards ADSL Links," in *Proc. of the IEEE Global Internet Symp.*, (Phoenix, AZ), April 2008.

[26] D. Croce, T. En-Najjary, G. Urvoy-Keller, and E. W. Biersack, "Fast Available Bandwidth Sampling for ADSL Links: Rethinking the Estimation for Larger-Scale Measurements," in *Proc. of the ACM PAM Conf.*, (Seoul, South Korea), pp. 67–76, April 2009.

[27] J. P. R. Draves and B. Zill, "Routing in Multi-radio, Multi-hop Wireless Mesh Networks," in *Proc. of the ACM MobiCom Conf.*, (New York, NY), pp. 114–128, October 2004.

APPENDIX

If after applying the correction factors in eq. (8), the resulting corrected duty cycles are such that $\sum_i f_{ik} > 1$, we apply the following algorithm to distribute the spare duty cycle:

1. we first reduce the duty cycles for those stations that overestimated it, i.e., we recalculate the adjusted duty cycles f'_{ik} as follows

$$f'_{ik} = \begin{cases} \sigma_{ik} f_{ik} & \text{if } \sigma_{ik} \leq 1 \\ f_{ik} & \text{otherwise} \end{cases}$$

2. Once adjusted, if the demanded duty cycles exceeds the capacity of the card, i.e., $\sum_i f'_{ik} > 1$, then we normalize them $f''_{ik} = f'_{ik} / \sum_i f'_{ik}$. If, on the other hand, there is spare time $f_{sp} = 1 - \sum_i f'_{ik}$, we distribute it among the links that need to increase their duty cycles ($\sigma_{ik} > 1$) proportionally to their needs as follows

$$f''_{ik} = \begin{cases} f'_{ik} + f_{sp} \frac{\sigma_{ik}}{\sum_i \sigma_{ik}} & \text{if } \sigma_{ik} > 1 \\ f'_{ik} & \text{otherwise} \end{cases}$$

3. Each station uses the resulting values f''_{ik}.

MIMO Wireless Networks with Directional Antennas in Indoor Environments

Tae Hyun Kim[†], Theodoros Salonidis[‡], and Henrik Lundgren[‡]
The University of Illinois at Urbana-Champaign[†] Technicolor Paris Research Lab[‡]

***Abstract*—We perform an experimental characterization of an indoor MIMO system with directional antennas (our prototype multi-sector antennas). The study reveals that, even without antenna directivity gain, the directionality of signals changes the MIMO channel structure and provides a way to improve MIMO throughput performance. It also shows that it is sufficient for the improvement to consider only a small subset out of all possible antenna direction combinations. Finally, our study shows that it is possible to achieve both throughput gains and interference reduction, thus increasing network spatial reuse.**

I. Introduction

Due to the increasing demands for large throughput wireless communications, Multiple Input Multiple Output (MIMO) has become one of the key technologies. MIMO combines multiple omni-directional antennas with signal processing techniques to extend the dimension of available wireless resources to time, frequency, and space [1]. Adopted in many standard protocols such as [2]) MIMO has already been widely deployed to transport streamed voice and high-definition video traffic applications.

A second wireless technology that has long been receiving interests is directional antennas. Directional antennas use narrow beams to focus RF energy toward desired receivers. This achieves throughput gains and reduces interference. Moreover, the directivity of antennas facilitates determining proper orientation in outdoor environments [3]. Recently, it has been shown that directional antennas in indoor environments provide a few strong paths between nodes even without the line-of-sight path. The consequent benefits are empirically demonstrated, attracting renewed interests [4], [5].

The benefits and losses of combining MIMO and directional antennas have not been fully studied to date. MIMO achieves capacity gains in rich scattering multi-path environments. Outdoor environments typically have a single strong line-of-sight path and a directional antenna would decrease the capacity of a MIMO link. In contrast, it is not clear how directional antennas for MIMO would perform in indoor environments. On one hand the narrow beams of directional antennas might decrease the degree of multi-path or signal scattering and decrease the capacity of MIMO links. On the other hand, directivity might change the structure of propagation paths and increase signal strength thus increasing the MIMO link capacity.

This paper takes an experimental approach to develop a principled understanding of the performance of MIMO wireless networks with directional antennas. For experiments, our 802.11n based MIMO testbed is equipped with multi-sector antennas without directivity gain. We use the term directivity gain to indicate the additional antenna gain of a directional antenna toward one direction, compared to an omni one. Also, the term sector activation pattern or simply sector pattern indicates a combination of activated sectors for all antennas involved in a transmission. We investigate the impact of different factors such as RSS, the number of activated sectors, Tx (transmit) or Rx (receive) sector activation, geographic location, and interference.

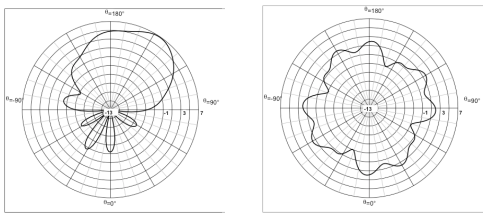
We make three interesting observations through the empirical study. First, even without directivity gain, multi-sector antennas can yield large throughput gains by changing the structure of MIMO channels. However, they may also produce large throughput loss if sector activation patterns are not properly selected. Interestingly, this gain and loss are all observed in a very small subset of activation patterns. Second, the sector patterns with the highest throughput largely depend on the environment and are not associated with the number of activated sectors, Tx or Rx sector activation or antenna orientation. Third, the interference level when multi-sector antennas are used is proportional to the number activated sectors. This allows us to have a coordination mechanism for more concurrent transmissions, improving the spatial reuse of a network.

Related Work: Aside from [3], [4], [5], the performance of MIMO with directional antennas was studied in both indoor and outdoor environments in [6]. However, the study was under fixed orientation of the antennas while this paper is about the sector activation, which is equivalent to the change of antenna orientation. Unlike their conclusion that the gain of using directional antennas is marginal, we will show that the control of antenna orientation can bring a huge throughput gain. In [7], only outdoor to indoor scenarios are considered with directional multiple antennas. In this case, less scattering is expected, compared to the indoor-to-indoor case that this paper concerns about.

II. Experimental Setup

A. Multi-sector Antennas

We use the Vivaldi multi-sector antenna developed by Technicolor Research. Each antenna has four antenna elements (sectors) printed on a PCB and covers the entire horizontal plane in the 5 GHz band. Any combination of sectors can be activated through a feeding network, which provides $(2^4 - 1) = 15$ activation patterns. One of them is four activated sectors, corresponding to an omni directional mode



(a) Single sector pattern. (b) Omni-mode pattern.

Fig. 1. Radiation pattern of multi-sector antenna without feeding loss.

TABLE I
ANTENNA GAINS (GAINS IN DBI AND LOSS IN DB)

	Omni	3 Sec	2 Adj	2 Opp	1 Sec
Directivity gain	2.4	3.5	4.6	5.7	6.9
Feeding loss	0	-1.25	-3	-3	-6
Overall gain	2.4	2.25	1.6	2.7	0.9

pattern. We simply call it omni-mode. Fig. 1(b) and 1(a) depict the radiation patterns with one and four activated sectors, respectively. Table I shows that the antenna directivity gain for each pattern depends on both number of activated sectors and, for the case of two activated sectors, on whether they are opposite (*2 Opp*) or adjacent (*2 Adj*). We see that the directivity gain is higher for sector patterns with less activated sectors. However, the antenna feeding network has been designed to introduce a feeding loss such that all sector patterns exhibit approximately equal peak gains.

B. Testbed

Our wireless testbed is deployed in a single floor in the Technicolor Lab, as shown in Fig. 2. This is a typical office environment consisting of cubicles, booths and offices separated by glass walls. Due to the availability of multi-sector antennas, we have only four nodes, but different topologies are emulated by Tx power control.

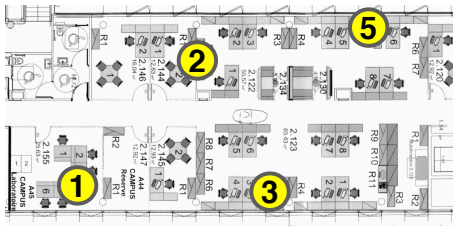


Fig. 2. The testbed deployment in Technicolor Paris Research Lab.

Each testbed node is a PC with Intel Pentium (M) 1.73 GHz processor and 512 MB RAM, running Ubuntu Linux distribution version 10.04. The PC hosts a commercial mini-PCI 802.11n Network Interface Card (NIC) with a Ralink RT2880 chipset. A NIC includes three antenna ports with two radio chains supporting both Spatial Division Multiplexing (SDM) and Space-Time Block Coding (STBC) MIMO communications. We disable one of three ports to have 2×2 MIMO. Ralink NIC is also capable of adjusting Tx (transmit) power level to one of 18, 17, 15, 12, 9 and 6 dBm.

During the experiments, the NICs are configured to operate in the 5.3 GHz band with 20MHz bandwidth with 400 ns guard-interval. During operation the NICs can select among basic 16 Modulation and Coding Sets (MCS), which correspond to 8 modulation rates under SDM MIMO mode and 8 under STBC MIMO mode. Each node has two Vivaldi multi-sector antennas connected to individual radio chains in a RT2880 NIC. The spacing between antennas is set to 7 cm according to our previous work with omni-antennas [8]. The feeding network of each Vivaldi antenna is controlled via a USBIO24 R Digital I/O Module interface, which allows us to control sector activations directly from the PC host using user-level Linux shell scripts.

III. THROUGHPUT ESTIMATION

In this section, we describe how we address multi-sector antenna MIMO measurement challenges and show that SNR can be used as a throughput predictor.

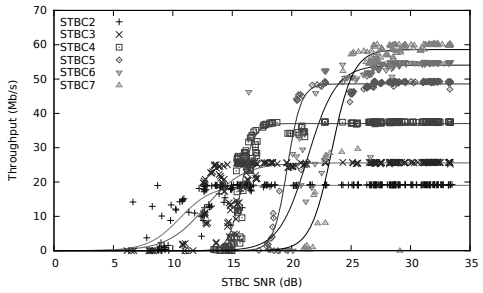
A. Measurement Challenges

We use UDP throughput as performance metric. However, direct measurement of the maximum throughput of a MIMO link with multi-sector antennas entails the following challenges.

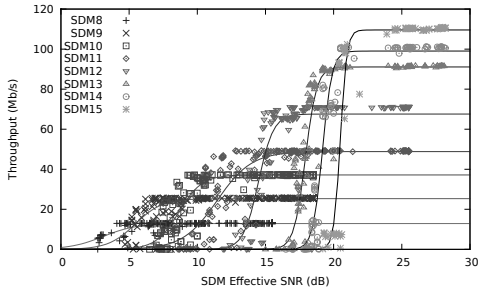
Large number of activation patterns: In a $M \times M$ MIMO system with K MCS data rates, where both Tx and Rx use multi-sector antennas of s sectors each, testing all combinations requires $K \times (2^s - 1)^{2M}$ throughput measurements. This corresponds to 810,000 throughput measurements in our system, where $M = 2$, $s = 4$ and $K = 16$. To address this, we restrict the number of activation patterns considered for each link. Specifically, sector activation is performed at either Tx or Rx, with the other end of the link in omni-mode. Moreover, the number of active sectors is kept the same for each antenna. We call the restricted set of activation patterns as a *pattern set*. This reduces the number of considered sector patterns from $(2^s - 1)^{2M}$ to $\sum_{x=1}^{s-1} \binom{s}{x}^M$. In our case, $\sum_{x=1}^{3} \binom{4}{x}^2 = 68$ which comprise 6 pattern sets.

Multiple MCS rates: We cope with multiple MCS rates, using UDP throughput vs. SNR mappings. In addition to Received Signal Strength (RSS) provided by most 802.11 wireless cards, the hardware of our Ralink R2880 chipset internally stores Signal-to-Noise Ratio (SNR) information for each received packet, but does not export it. Our workaround is to directly access the internal memory to obtain this information. If a packet is encoded in the SDM mode, a pair of SNR values (SNR per spatial stream) is available, and, if encoded in the STBC mode, a single SNR value is available. We call these types of SNRs as SDM SNR and STBC SNR, respectively.

Measurement under channel variation: Another challenge is that the measurement results for different activation patterns may be affected by time variation of wireless channels. To quantify this impact, we measure the fraction of SDM effective SNR samples which fall into $\pm\delta$ dB range of the long term



(a) STBC SNR to throughput.



(b) SDM effective SNR to throughput.

Fig. 3. Ralink RT2880 chipset-specific SNR to throughput relationships.

average. The effective SNR is a single representative SNR for a pair of SDM SNRs proposed in [9]. At night, only 10% of the samples deviate more than 2 dB from the long term average, even for time intervals up to 200 seconds. In daytime, around 15% of the samples deviate more than 2 dB for time intervals of 20 seconds, which is close to the 1 dB granularity of our 802.11n cards. To further minimize the outliers' effect, we perform all measurements at night (unless specified otherwise), measure throughputs of each activation pattern and omni-mode back-to-back, and take their difference, measuring relative change at each instance.

B. SNR-based Throughput Estimation

We set up a series of experiments to map our hardware-specific SNR information to UDP throughput. For 5 seconds, we measure the UDP throughput of each MCS rate with each of the restricted pattern sets, using the `iperf` and `tcpdump` tools. We found that 5 seconds measurement duration for each configuration is sufficient and that this duration results in the best mapping between SNR and throughput. The traffic load is set to be higher than each MCS rate. Since our workaround to extract SNR does not work when the NIC is busy to process incoming packets, as soon as the throughput measurement is over, a light traffic load (1 Mbps) is applied to collect STBC and SDM SNR information for 5 seconds for each. Then, the SNR values are averaged and mapped to the throughput. Note that for SDM, each SNR value pair is combined to a single effective SNR [9] for one-to-one mapping to the throughput. We emulate different topologies by power control to diversify the range of samples.

Fig. 3 shows each of the averaged throughput samples and corresponding SNR or (effective SNR) values. To generate the mappings, we fit a generalized Sigmoid function to each set of throughput samples with the same MCS rate.

In the rest of the paper, we estimate the throughput of a link using a certain sector pattern by measuring the STBC SNR and SDM effective SNR values, converting these two SNR values to throughputs using the mappings, and taking the maximum.

IV. Throughput Gains

Experimental setup. We use the same setup for SNR measurement as Section III. The SNR of omni-mode is also measured in a back-to-back fashion to obtain the throughput gain via the constructed mapping in the previous section. We repeat experiments five times for 10 links with all 6 pattern sets.

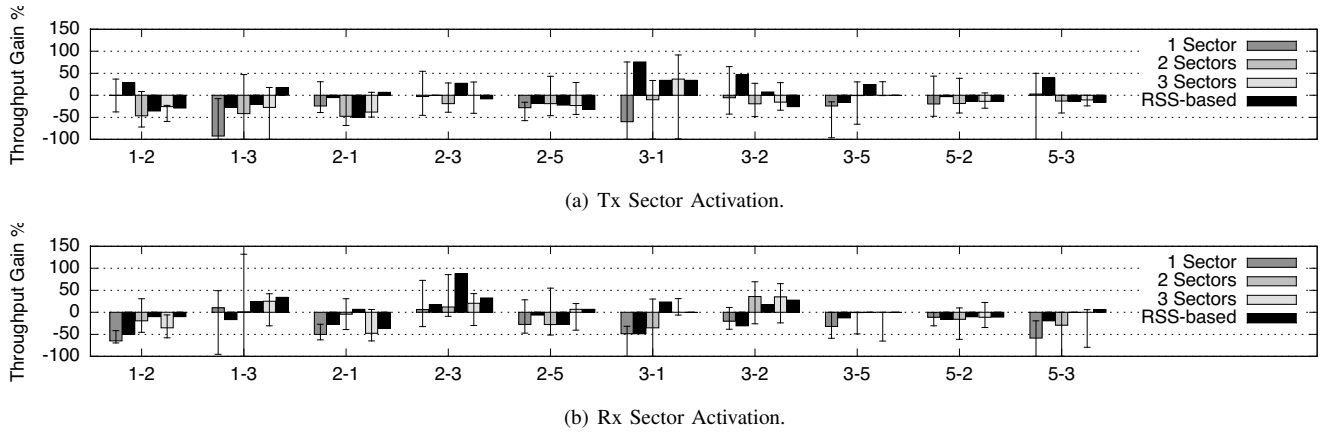
Throughput gains. Fig. 4 shows the throughput gains for all pattern sets that we consider. Each gray bar in Fig. 4 shows the median gain for one, two and three activated sectors per antenna; its error bars indicate maximum and minimum throughput gains. The black bars will be explained shortly.

Most links achieve a positive maximum throughput gain over omni mode (at most 130% on link 1-3 with 2 Rx sectors per antenna and 21% averaged across all links). This appears counter-intuitive because, as mentioned earlier, our multi-sector antennas do not provide antenna directivity gain over omni mode. Since multi-sector antennas transmit or receive at equal or less signal power, and some of diverse paths between Tx and Rx are suppressed, one might argue that it would not be possible to observe positive throughput gains.

The positive throughput gains are due to the clustered propagation of signals in the angular domain. Propagation measurements in indoor environments [10] have shown that the angles of departure (AoD) and arrival (AoA) form correlated signal clusters. Moreover, only 2 to 4 clusters mostly contribute to the received signals. The sector patterns that achieve positive throughput gains, are aligned in phase with these dominant signal clusters, thus avoiding negative gains. At the same time, they are also misaligned with other clusters that induce signal correlations at the antenna input. This misalignment reduces the received signal correlation, making multiple information streams in MIMO channel appear more de-correlated. *Therefore, sector activation without directivity gain can create throughput gains by structurally changing the MIMO channel.*

Despite the potential for a large positive maximum throughput gain, Fig. 4 also shows that most links achieve negative median throughput gain (-9.3% in average), and the minimum gain can reach as low as -100% (e.g., link 1-3 for 2 Rx sectors per antenna). Thus, less than half of the sector patterns provide positive gains and if a sector activation pattern is not carefully chosen, it may yield a large penalty on throughput performance.

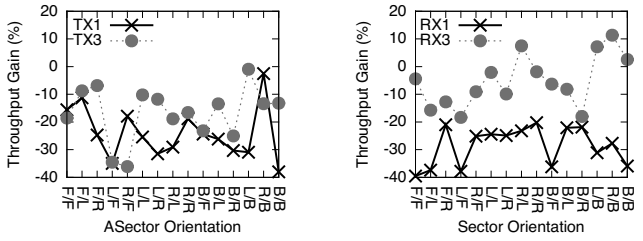
Pattern selection criteria. Can we determine patterns with maximum or positive throughput gain in a systematic way? The answer may depend on different criteria investigated below.



(a) Tx Sector Activation.

(b) Rx Sector Activation.

Fig. 4. Median throughput gain per link and the gain when a pattern with the highest RSS is chosen. Error bars on each median gain are the maximum and minimum gains, respectively.



(a) Tx activation

(b) Rx activation

Fig. 5. Throughput gain vs orientation of active sectors. TX1 and TX3 indicate 1 Tx and 3 Tx sector activation patterns, respectively (RX1 and RX3 correspond to RX activation). F, L, R, and B abbreviate Face, Left, Right, and Back, respectively. For example, F/F in TX1 (TX3) orientation means both antennas have their active Tx sectors which face the receiver.

1) Number of activated sectors per antenna: Fig. 4 shows that the maximum throughput gains do not depend on the number of activated sectors or whether Tx or Rx activations are used.

2) Geographical orientation of active sectors: Fig. 5 depicts the average throughput gains over all links, as a function of sector pattern orientations sorted by geographical direction of activated sectors toward the other end of the link. Unlike the case with a single multi-sector antenna per node [4], the geographical relationship is not correlated with throughput gains, again regardless of number of active sectors and Tx or Rx activation.

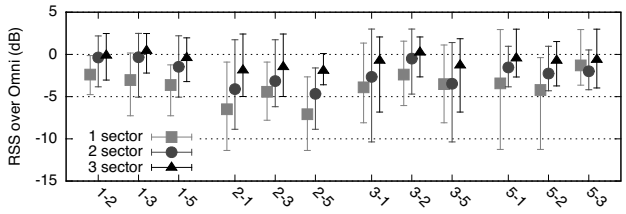
3) Reciprocity: For most node pairs (x,y) in Fig. 4, the performance of link x-y with Tx activation can be radically different to the performance of reciprocal link y-x with Rx activation. Thus, in general link reciprocity cannot be leveraged to reduce measurement overhead.

4) RSS: Each black bar next to a gray bar in Fig. 4 is the throughput gain when the pattern with the maximum RSS is chosen among the corresponding activation patterns. For most links, the throughput gains from RSS-based selection are not close to the maximum throughput gains. Instead, they are close to the median throughput gains, which are primarily negative. This holds for both Tx and Rx activation in Figs. 4(a) and 4(b), respectively. Thus, without antenna directivity gain, the RSS cannot predict throughput gain well.

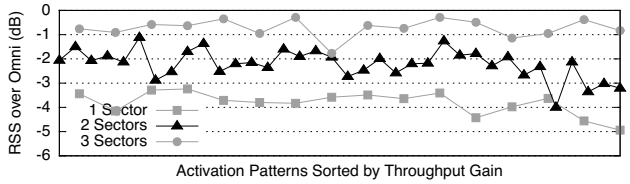
5) Location: One may guess that the trend of throughput gains over sector patterns in a fixed pattern set is similar to those of other links, irrespective of their locations. If so, a probability distribution for the gains could be found, which might lead us to a systematic way to check if it is possible to achieve a positive gain and to further find good candidate patterns to achieve it. Not shown in this paper due to the space limit, however, it is observed that individual links show radically different trends. Therefore, it is not straightforward to develop a systematic way to find a good pattern, which works across all links.

Reason for arbitrary throughput gains. Based on the above observations, we conclude that the throughput gains appear "arbitrary." Our interpretation is that sector activation changes the structure of MIMO channels in a way that depends highly on the surrounding environment, which is "arbitrary." In theory, the relationship between Tx and Rx antenna signals in a $M \times N$ MIMO channel can be simplified as $\mathbf{y} = \gamma \cdot \mathbf{H}\mathbf{x}$ where $\mathbf{x}$ is an $1 \times M$ input (Tx symbol) vector, γ is a scalar path loss-based channel gain, $\mathbf{H}$ is the normalized $M \times N$ MIMO channel matrix, and $\mathbf{y}$ is an $1 \times N$ output (Rx symbol) vector. Our previous observation that RSS is not a good indicator of the throughput gain means that the major contributor to throughput gain is the structure of MIMO channel matrix $\mathbf{H}$ instead of the scalar gain γ. Existing studies state that the structure of $\mathbf{H}$ highly depends on the surrounding environments of a link and the resultant paths [10]. In indoor environments, the surroundings for each link are noticeably different and lead to different throughput gain characteristics across activation patterns.

We conclude that it is possible to achieve positive throughput gains even with a limited subset of all patterns. By collecting SNRs of activation patterns from either Tx or Rx pattern set with a fixed number of activated sectors, multi-sector antennas provide 21% of throughput gains in average



(a) Average RSS_{diff} without antenna directivity gain: Average interference differences of Tx sector activations over omni mode at neighbors of each link.



(b) Average RSS_{diff} across all sector patterns: Average interference difference at neighbors. For each link, the patterns are first sorted by descending throughput gains and then the RSS_{diff} values with the same ranking are averaged.

Fig. 6. Interference properties without antenna directional gain.

for MIMO communications. Our measurements also show that it is challenging to nd a criterion to select a pattern with positive throughput gain from the limited pattern set because of the arbitrary change of MIMO channel structure in the indoor environment. Thus, selecting a good pattern might require to periodically collect SNRs of all patterns in a pattern set. Although one may argue that the measurement overhead cannot be ignored, we claim that it can be reduced as SNRs are collected at a lower frequency.

V. Interference properties

So far the focus of performance characterization has been on a link. This section provides some insights into network performance through the analysis on interference properties of MIMO multi-sector antennas.

Experimental setup. Using a similar experimental methodology, we measure SNR and RSS for each pattern in the Tx activation sets, immediately followed by omni-mode. We use the difference $RSS_{\text{diff}} = RSS_x - RSS_{\text{omni}}$ as interference metric. A negative value means that sector activation pattern x reduces interference compared to omni-mode and increases spatial reuse. All measurements are performed at night, and the results are the average of five iterations.

Interference without directivity gain. Fig. 6(a) depicts the average RSS_{diff} at the neighborhood of each link when sector activation is in use. For example, the 1 sector point of link 1-2 is the average of RSS_{diff} values from all neighbor links, which are link 1-3 and link 1-4. From Fig. 6(a), the interference reduction increases as the number of active sectors decreases. With 1 Tx sector per multi-sector antenna, the interference over omni-mode can be reduced up to 12 dB at maximum (link 2-1) and 8 dB on average (link 2-5).

Although sector activations reduce interference level, they may not necessarily increase throughput gain. Fig. 6(b) depicts the interference amount in descending order of throughput gains. We observe that, for each number of activated sectors, the amount of interference is not related to the throughput gains. Thus, by selecting a number of activated sectors, it is possible to maximize throughput gain subject to a constant interference level, which is minimum when 1 Tx Sector activation patterns are considered.

We conclude that the interference level without antenna directivity gain is proportional to the number of activated sectors and has little correlation with the amount of throughput gain. Therefore, one can exploit spatial reuse in addition to throughput gain to enhance network-wide performance. However, spatial reuse comes at the expense of coordination mechanisms of multiple concurrent transmissions.

VI. Conclusions

In this paper we presented performance characteristics of multi-sector antenna-equipped IEEE 802.11n MIMO wireless networks. Even with the absence of directivity gain, the use of sector activation can improve throughput over omni-mode, and leaves a room for further improvement through spatial reuse. Our empirical study confirms the benefits of multi-sector antennas for MIMO communications, calling for future work on the details of how it can be harvested in real network scenarios.

References

[1] A. J. Paulraj, D. A. Gore, R. U. Nabar, and H. Bolcskei, "An overview of MIMO communications- A key to gigabit wireless," *Proceedings of IEEE*, vol. 92, pp. 198–218, 2004.

[2] 3GPP Workgroup R1, *Requirements for further advancements for E-UTRA (LTE-Advanced)*, 3GPP TR 36.913, Std.

[3] T. Korakis, G. Jakllari, and L. Tassiulas, "A MAC protocol for full exploitation of directional antennas in ad-hoc wireless networks," in *Proc. ACM MobiHoc*, 2003.

[4] A. P. Subramanian, H. Lundgren, T. Salonidis, and D. Towsley, "Topology control protocol using sectorized antennas in dense 802.11 wireless networks," in *Proc. IEEE Internat. Conf. Network Protocol (ICNP)*, Sept. 2009.

[5] X. Liu, A. Sheth, M. Kaminsky, K. Papagiannaki, S. Seshan, and P. Steenkiste, "Pushing the envelope of indoor wireless spatial reuse using directional access points and clients," in *Proc. ACM MobiCom*, Sept. 2010.

[6] N. Razai-Ghods, M. Abdalla, and S. Salous, "Characterisation of MIMO propagation channels using directional antenna arrays," in *in Proc. IEEE Sarnoff Symp.*, Princeton, NJ, Mar. 2009.

[7] C. Hermosilla, R. Feick, R. A. Valenzuela, and L. Ahumada, "Improving MIMO capacity with directive antennas for outdoor-indoor scenario," *IEEE Trans. Wireless Comm.*, vol. 8, no. 8, pp. 2177 – 2181, May 2009.

[8] K. Pelechrinis, T. Salonidis, H. Lundgren, and N. Vaidya, "Analyzing 802.11n performance gains," in *Proc. ACM MobiCom (Poster Session)*, Sept. 2009.

[9] D. Halperin, W. Hu, A. Sheth, and D. Wetherall, "Predictable 802.11 packet delivery from wireless channel measurements," in *Proc. ACM SIGCOMM*, Aug. 2010.

[10] Q. H. Spencer, B. D. Jeffs, M. A. Jensen, and A. L. Swindlehurst, "Modeling the statistical time and angle of arrival characteristics of an indoor multipath channel," *IEEE Journal on Sel. Areas in Commun.*, vol. 18, no. 3, pp. 347–360, 2000.

Wireless Home Network Monitoring

Nan Deng, A-K Pietilainen, Henrik Lundgren

Technicolor Technical Report
Number: **CR-PRL-2012-09-001**
First Publication: September 2012

Abstract: Wireless technologies also known as WiFi have become more and more important in our home networks. But WiFi is not perfect, with the rapid increase of WiFi deployments, different problems have been appearing. Users may experience slow download speed, poor video streaming, voice communication delays because of the malfunctions of Wireless LANs. Due to a lack of capabilities of troubleshooting in both home network gateways and end-user devices, numerous calls to helpdesks make service providers obliged to spend money and time on manual services. Both of end-users and ISPs are unsatisfied about this kind of situation. Therefore, instead of manual services, we try to empower the home network gateways some capabilities to troubleshoot and end-users can help themselves to find problems. We investigate into several specific wireless problems such as low signal strength, congestion and hidden terminal, propose monitoring and analysis method which can help to do troubleshooting. We deploy controlled testbeds to validate our methodologies and apply them to a selected real home network.

1. INTRODUCTION

Nowadays, the wireless technologies also know as WiFi are becoming more and more important in our lives. According to an investigation and forecast by Cisco Visual Networking Index(Figure 1), in 2011, the WiFi traffic has accounted for about 35 percent of IP traffic. For the next five years, the WiFi traffic will continue to rapidly grow and it will exceed the traffic from wired devices by 2016. In 2016, wired devices will account for 39 percent of IP traffic, while WiFi and mobile devices will account for 61 percent of IP traffic"[13].

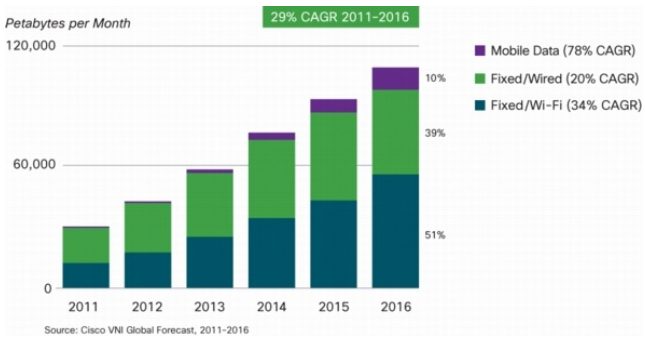


Figure 1: Wireless evolution[13]

However, the wireless technologies are not perfect. For reason of the nature of WiFi technologies and inappropriate deployments, more and more Wireless LANs problems have been appearing, such as congestion, hidden terminal and non-802.11 interferences. These problems cause customer frustrations, but due to the fact that few network troubleshooting and management tools are available for end-users, the resulting calls to helpdesk are numerous and expensive for ISPs. Our work is motivated by this lack of capabilities of troubleshooting. By embedding some simple monitoring capabilities on gateways, supported by smart, distributed algorithms, we can empower users to diagnose problems easily. When they cannot be diagnosed locally, the ISP helpdesk also has more contexts to reason about the problem being experienced. This saves costs for the ISPs, increases satisfaction and reduces frustration for end-users.

There is already some work which has been done in this area. Some of them have spent efforts to understand the natural and theoretical problems of the 802.11 protocols[30][21][8]. Jigsaw[10] and MOJO[28] are tools which have been developed to collect wireless metrics. WiFi Profiler[9] mainly focuses on the cooperative diagnosis in Wireless LANs. But we think the research in the home network gateway based troubleshooting is not enough, thus we mainly focus on some well-known problems on the home Wireless LANs and propose our ways to troubleshoot.

We firstly study the nature of some most common problems in the wireless home networks, such as low signal strength, congestion and hidden terminal. For each different problem, we investigate their causes, characteristics and possible ways to detect them. To validate our methodologies, several controlled experiments have been deployed in the lab. By collecting and analyzing data in these experiments, our detection methods have been gradually improved. Once our methodologies are validated, we then apply them to the real home networks in order to find true problems and also the constraints of our methods in real wireless home network environments.

This report is organized as follows : In Chapter 2, some 802.11 protocol natures will be introduced. In Chapter 3, we will talk about several specific wireless problems and the analysis about them. In Chapter 4, we will describe our methodology and procedures for troubleshooting these problems. The Chapter 5 is about the motivations and environmental descriptions about our controlled experiments and home experiments. In Chapter 6, we will present the main results and our analysis about them. Chapter 7 will introduce some similar works others have done. We will then conclude in the Chapter 8 and give a brief description about future work.

2. BACKGROUND

In this chapter, we will give a brief introduction about the main technology of WiFi which is 802.11 protocols. We will also shortly present the 802.11 channels, protocols, operating modes and MAC layer.

2.1 802.11 Protocols

Until today, the IEEE commission has launched five versions of 802.11 protocols. We summarize the different protocol versions in Table 1. Different protocol versions mainly differ in terms of supported frequency bands, spread spectrum technology and the supported maximum bit rates. The latest protocol is 802.11n, which deploys the MIMO technology, supports both 2.4 GHz and 5 GHz bands, and its maximum bit rate can be as much as 540 Mbps. But it is not greatly deployed yet, IEEE 802.11 a/b/g[7] are still the main protocols in the home network environments.

2.2 802.11 Channels

The IEEE 802.11 standard establishes several requirements for the RF transmission characteristics of an 802.11 radio. Included in these are the channelization scheme as well as the spectrum radiation of the signal (that is, how the RF energy spreads across the channel frequencies). As shown in the Figure 2 The 2.4-GHz band used by 802.11 b/g is broken down into 11 channels for the North American domain and 13 channels for the European domain. These channels have a center frequency separation of only 5 MHz and an overall channel bandwidth (or frequency occupation) of 22 MHz. This means that adjacent channels overlap, and can interfere with each other. There are only three non-overlapped channels, which are 1, 6 and 11. 802.11a uses the 5 GHz band, which, for much of the world, offers at least 23 non-overlapping channels rather than the 2.4 GHz frequency band. Better or worse performance with higher or lower frequencies (channels) may be realized, depending on the environment.

2.3 802.11 Operating Modes

The wireless cards which support 802.11 protocols have two working modes, ad-hoc and infrastructure mode.

In the infrastructure mode as the Figure 3 shows, a workstation in the network serves as a main station, other workstations must communicate with the whole Internet through this workstation, this kind of network is called BSS(Basic Service Set), and this main workstation is called AP(Access Point). This mode is widely used in the home networks, and the Access point is normally our gateway. In this report, all our discussions are based on this mode.

Protocol	Year	Band	Spread Spectrum Technology	Max Tx Bit rate
802.11	1997	2.4 GHz	FHSS or DSSS	2 Mbps
802.11a	1999	5 GHz	OFDM	54 Mbps
802.11b	1999	2.4 GHz	HR-DSSS	11 Mbps
802.11g	2003	2.4 GHz	OFDM	54 Mbps
802.11n	2004	2.4GHz or 5GHz	OFDM	540 Mbps

Table 1: 802.11 protocols

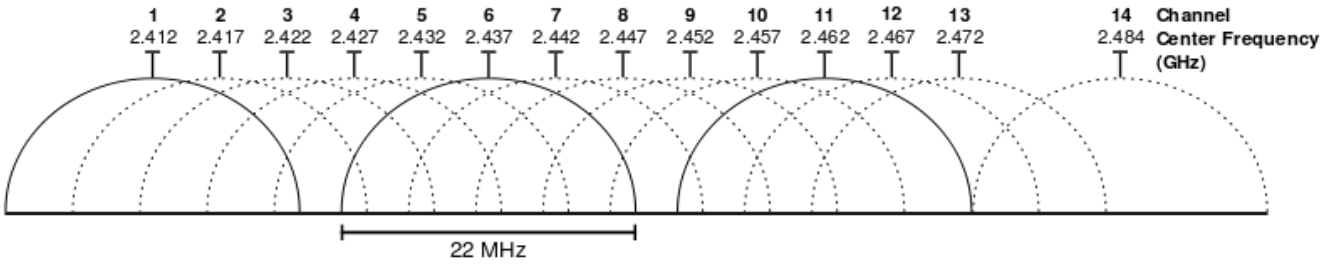


Figure 2: Channels in 2.4 GHz band[15]

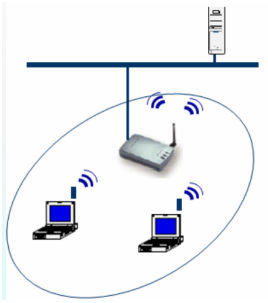


Figure 3: Infrastructure mode

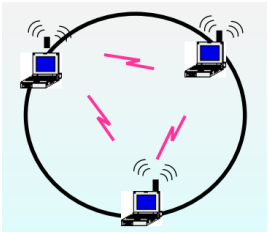


Figure 4: Ad-hoc mode

In the ad-hoc mode as the Figure 4 illustrates, no main workstation is needed any more, the peer-to-peer communication is realized directly between two workstations. This network is called IBSS(Independent Basic Service Set), it is widely used in the military and sensor networks.

2.4 802.11 MAC Layer

In this section we will shortly introduce some important features about 802.11 MAC layer, which will help to understand in the further discussions. The MAC protocol supplies the functionality required to provide a reliable delivery mechanism for user data over noisy, unreliable wireless media[7]. It also ensures the fairly control access to the shared wireless medium and protects the data that it delivers.

802.11 MAC layer has three access methods : DCF(Distributed coordination function) CSMA/CA(Carrier Sense Multiple Access with Collision Avoidance) which is the mandatory one, DCF RTS/CTS(Ready To Send/Clear To Send) and PCF(Point Coordination Function).

As illustrated in the Figure 5, the CSMA/CA algorithm says that if the channel is idle for more than DIFS(Distributed Inter-Frame Spacings), a sender is allowed to use the medium immediately. If several senders would like to use the medium, they will have to back off for a random interval chosen from 0 to Max_{BO}. The sender who firstly has counted down its backoff time to 0 will have access to the medium. Max_{BO} increases exponentially from 31 to N slot times whenever an retransmission happens. N differs in different protocols. When a sender have access to the medium, it has to wait DIFS before starting. Once the sender starts to transmit, a unicast packet is sent to all the other stations waiting for the medium. The unicast packets contain the amount of time the DATA frame needs the medium. When the transmission of DATA is finished, after waiting for SIFS, receiver will acknowledge by responding an ACK frame if the packet was received correctly. Automatic retransmission of DATA frames will start in case of transmission errors.

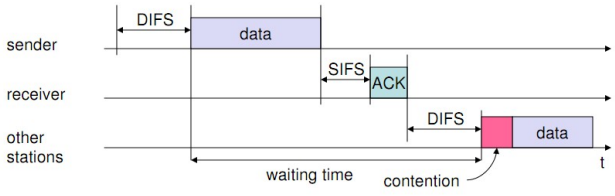


Figure 5: CSMA/CA access method[7]

In the RTS/CTS mode as Figure 6 shows, a sender can send RTS with reservation parameter after waiting for DIFS. If the RTS is transmitted successfully, after SIFS, the re-

ceiver acknowledge by CTS. The sender can then send DATA after SIFS, and acknowledged by an ACK. The other stations store the medium reservations distributed by RTS and CTS indicating the time that the transmission takes. This mechanism is designed to solve the well-known hidden terminal and exposed terminal problems. Further discussions about these will be in the Chapter 3.

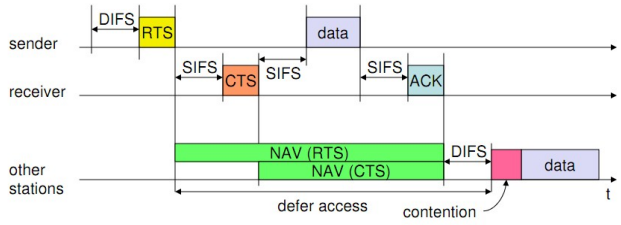


Figure 6: RTS/CTS access method[7]

The PCF mode is to use the access point as an arbitrage who manages the channel. It is very useful when the voice or video data are transmitted because the access point would have rights to control the priority. There is also no collision in this mode, but it is not considered in this paper.

In this report, most of the measurements are based on DCF CSMA/CA mode, some will be based on both DCF CSMA/CA and DCF RTS/CTS.

3. PROBLEM STATEMENT

As we presented in the Chapter1, user experiences and service qualities are greatly influenced by the wide spread use of WiFi technology. For the end-users, WiFi problems often manifest in poor video streaming qualities, slow download speed or voice communication delays. More technically, the consequences are reflected by the effects such as the drop of throughput, the increase of delay and packet loss. However, these are only the results of the problems but not the root causes of the problems. In this chapter, we will introduce some root causes of wireless problems, such as low signal strength, congestion, hidden terminal and low bit rate.

3.1 Low Signal Strength

In Wireless LANs, the low signal strength is one of the most important issue in Wireless LANs. However, too many common terms, such as "signal strength", "signal to noise ratio", "signal quality", have been deployed to define this metric and it is not always clear what are their real meanings. First of all, we will try to give the correct meanings for all the terms and refer to the really used and correct ones as our signal strength metric[6].

- **Signal Strength** : The 802.11 tools often provide us the signal strength with different units which are mW(milliwatts),dBm(" dB"-milliwatts),RSSI(Received Signal Strength Indicator). The term "signal strength" here means the received energy which has the very first unit of mW. The mW is converted to dBm by performing a base-10 logarithm and then multiply by 10(dBm=log(mW)*10). RSSI represents a numeric value from 0 to RSSI_Max, the RSSI_Max differs between NIC vendors because it is often intended to be used in its internal logic.
- **Signal Quality** : The term "Signal Quality" is often deployed by people to indicate the "goodness" of an 802.11 signal or sometimes just as a substitute for "Signal Strength". However, the true "Signal Quality" defined by 802.11 as "PN code correlation strength" and only for the DSSS PHY. The most likely definition of "Signal Quality" or "PN code correlation strength" is that it is some metric of the correlation between the correct symbol-stream and the actual symbol-stream received. If that definition is correct, then signal quality should be used as a metric of the amount of corruption in the environment between the access point and the client.
- **Signal to Noise Ratio** : The SNR generally refers to the power level of an incoming signal relative to some type of background noise. While the noise always referred by Wireless Engineers means the background environment interference, but for electrical engineers it turns out to be the internal chip set noise related to electrical effects. Since 802.11 cards do not typically report SNR and the definition of noise is always unclear, it is hard to be used in practice.

In this report, the signal strength is used as the metric, with the unit of "dBm". The signal strength is related to the distance and the transmission power of wireless cards. The signal strength in "dBm" fades in a linear manner. This means that if we are a particular distance from an access point and we measure the signal power, then if we move twice as far away, the signal strength will have decreased by a factor of two. The signal strength is also linear to the transmission power of AP in "dBm".

Due to a fact that higher frequency waves are smaller and they do not travel far, the different 802.11 protocols have different operating range because their frequencies differ. The max operating ranges of different protocols are shown in the Table2 [1].

When a wireless device has a low signal strength, the data frames with a high bit rate cannot be decoded any more. Thus a rate adaptation algorithm is applied to help the device to choose a lower bit rate in case of low signal strength. The lower bit rate could again lead to problems such as performance anomaly problem, which will be discussed later. If the signal strength is lower than a certain threshold, the packets will not be reliably received any more. This threshold is a fundamental specification of an 802.11 card which is called its receive sensitivity. If the actual RF energy present at that card were less than the receive sensitivity, then the card would no longer be able to differentiate between signal and noise. The NIC would not detect incoming packets at all, and the packet would be lost.

3.2 Congestion

As the sharp increase of WiFi deployments, the wireless portion of the network is a major performance bottleneck in heavily utilized wireless networks. Congestion is becoming a common and serious problem in these networks. With a single collision domain, the occurrence of a high density of nodes and a big amount of data which exceeds the available capacity results in congestion, thereby causing a significant performance bottleneck. Effects of congestion include drastic drops in network throughput, unacceptable packet delays and session disruptions[19]. There are mainly two root causes of congestion. One is the lack of non-overlapped chan-

1. http://www.far-far-away.com/~yousif/articles/wifi-sig.php

Protocol	Frequency	Speed	Operating Range Max(approx.)
802.11a	5.0-5.8 GHz	54 Mbps	25 m
802.11b	2.4 GHz	11 Mbps	45-90 m
802.11g	2.4 GHz	54 Mbps	30-60 m

Table 2: Ranges of protocols

nels, the other one is the nature of 802.11 CSMA/CD mode.

Firstly, as introduced in the Chapter2, there are mainly two spectrum bands used in the Wireless LANs. In the mostly used 2.4 GHz band, only 3 channels out of 14 are non-overlapped channels which are 1, 6 and 11. 22 MHz is needed to well separate the spectrum of channels while only 5 MHz is designed between channels. Thus most of the Wireless LANs are recommended or forced to operate in these 3 channels. The usages in the overlapped channels would be harmful to the Wireless LANs[29]. According to an investigation of WiFi networks collected by HomeNet Profiler[14] from 463 individual homes, DiCioccio et al.[14] found out that a typical WiFi neighborhood is rather dense. They show an average of 8 neighborhood APs per home, and 80% of the homes have more than 3 neighborhood APs. As shown in Figure7, the channel distributions are mainly centralized on the channel 1, 6, 11 while there are still some APs operating in overlapped channels which are highly not recommended. Thereby we could draw a conclusion that in average most of the home Wireless LANs may share the medium with others. In this kind of networks, the occurrence of congestion can be often when several devices are using the medium at the same time.

Country	Channel						Total
	1	2 – 5	6	7 – 10	11	12 – 14	
France	18%	6%	17%	9%	40%	6%	2547
United States	26%	11%	27%	7%	28%	0%	360
Canada	14%	14%	26%	15%	28%	0%	209
Brazil	20%	6%	37%	13%	20%	1%	145
Italy	37%	3%	26%	4%	27%	2%	96
Overall	20%	7%	20%	9%	35%	5%	3864

Figure 7: Crowded channel[14]

Another root cause is the nature of CSMA/CA access method of 802.11 protocols. As we presented in the Chapter 2, most of the access points deploy the DCF CSMA/CA mode, which leaves the stations themselves to compete for the channel. The algorithm of backing off tries to be fair to all the stations. When many clients try to access to the medium at the same time, every client is only given a small fraction of accessing the medium, hence their throughputs decrease.

The consequences of congestion are sometimes catastrophic, the medium utilization percentage, frame retransmission attempts and data dropped will increase while the global throughput suffers a big drop.

There is almost no commercial tools now in the market which can easily measure the network and tell end-users if their network is congested or not. Even if some academic efforts were made, it is still very difficult to popularize them.

3.3 Hidden Terminal And Exposed Terminal

Hidden terminal and exposed terminal[20] are also the well-known problems of 802.11 protocols. Wireless stations have carrier sensing ranges and not all the stations are within the carrier sensing range of each other, this natural asymmetry is the main cause of hidden terminal and exposed terminal problems.

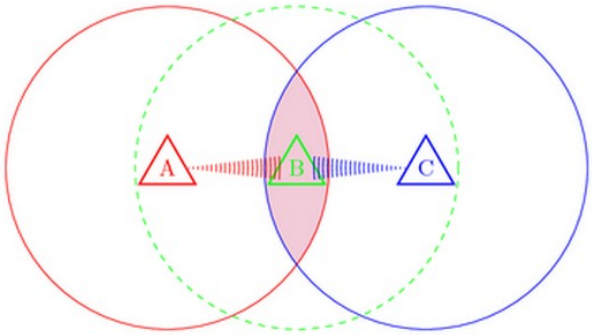


Figure 8: Hidden terminal

As illustrated in the Figure 8, node A and node C are both in the carrier sensing range of node B, but they are out of the carrier sensing range of each other. Suppose that both of node A and node C want to transmit to node B. By only sensing the medium, node A will not be able to hear any transmissions from node C, while node C is neither be able to know about the transmission of node A. When both A and C start transmitting, it leads to collisions at node B. This is the well known hidden terminal problem[20].

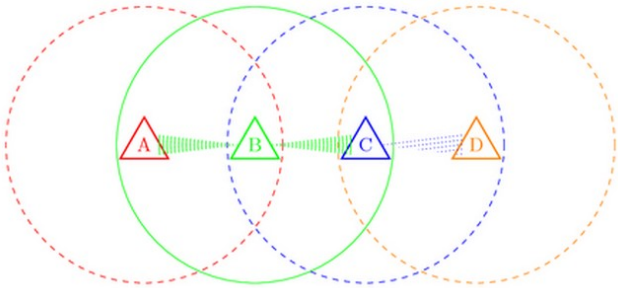


Figure 9: Exposed terminal

For another situation illustrated in the Figure 9, still because of the limit of carrier sensing ranges for wireless devices, we will experience the exposed terminal problem. The positions of node A, B and C are unchanged, while a node D is put in a place which is out of carrier sensing ranges of both A and B but within the carrier sensing range of C. Suppose that B is transmitting to A, node C will regard the medium as busy because B is in the carrier sensing range of C. If now C wants to transmit to D, it must wait until B is finished even if C is actually be able to transmit to D because it assumes the medium is busy and its transmission will lead to collision. Thus some transmissions from Node

C which could be performed are prohibited. Node D is the exposed terminal from C to node pair A/B[20].

The consequences of these two situations are quite obvious, both of them lead to the degradation of a node's throughput. In terms of hidden terminal in the Figure 8, unsuccessful transmissions result from collisions between a transmission originated by a node such as A which cannot hear the on going transmissions to its corresponding node B. The throughput of A is decreased, and the probability of such a collision is proportional to the total number of terminals hidden from A. In terms of exposed terminal in the Figure 9, unsuccessful transmissions result from nodes such as C being prevented from transmitting. The throughput of C is decreased and the channel utilization ratio is reduced.

3.4 Low Bit rate

For wireless cards, the algorithm of deciding which bit rate should be taken is called the rate adaptation algorithm[24]. As the 802.11 protocol considers the low bit rate frames containing less information than the high bit rate frames so that the probability of successful transmission could be higher[7], the rate adaptation algorithm is designed to assign the device a lower bit rate when repeated unsuccessfully frame retransmissions have been detected. The first case when this happens is that the wireless stations have very low signal strength so that transmissions with high bit rate fail repeatedly. Another case is when problems such as congestion or hidden terminal happen, a lot of retransmissions make the algorithm to choose lower bit rate to increase the rate of successful transmissions. It is commonly understood that only the device with low bit rate would experience worse wireless service quality. But according to our investigation, even a single device with low bit rate can degrade the performance of the whole network.

Heusse et al.[17] present us a performance anomaly problem of 802.11b, which is the same problem as ours. If there is at least one host with a lower rate in the Wireless LANs, the throughput of all hosts transmitting at the higher rate is degraded below the level of the lower rate. Such a behavior penalizes fast hosts and privileges the slow one. The reason for this is that the basic CSMA/CA channel access method guarantees an equal long term channel access probability to all hosts. When one host captures the channel for a long time because its bit rate is low, it penalizes other hosts that use the higher rate.

As illustrated in the figure 10, we consider the Client1 with 54 Mbps and the Client2 with 6Mbps having the same possibility of using the medium, as the CSMA/CA access method achieves. Firstly, if the Client1 with 54 Mbps is alone in the network and it is transmitting DATA packets of length L, then for each DATA packet, we consider the transmission time is t. Secondly, we put the Client2 with 6 Mbps alone in the network and transmit the same length of DATA packets, obviously the transmission time must be $9 * t$. At last, when the network have both of these two clients sending the same DATA packets, since their channel access probabilities are the same, the device with 6 Mbps will cost nine times the time than the device with 54 Mbps. So it is clear that the device with low bit rate occupy the channel most of the time, but the device with high bit rate can only send its frames after waiting for a long time which results in its huge drop in its throughput.

This problem is quite common now in the real home networks. In a typical wireless local area network, some hosts may be far away from their access point so that they have low signal strength. The rate adaptation algorithm helps wireless cards to choose lower bit rates to avoid repeated retransmissions.

4. METHODOLOGY

As we introduced in the Chapter 3, the wireless problems are always challenging issues. Even if some research work has been done, it is not sufficient to fully understand them and provide efficient solutions. We emphasize on some of the known problems and investigate the methods to troubleshoot them. In this chapter, we will introduce our methods of measurements and analysis for signal strength, congestion and hidden terminal.

4.1 Signal Measurements

As we introduced in the Chapter 3, low signal strength is one of the main problems in Wireless LANs. If a wireless network has a low signal strength, the transfer of information across the network could be slow. The signal strength itself is the best metric to detect these problems. As we discussed in the Chapter3, we take the "dBm" as unit of the metric signal strength.

4.1.1 Approach

The first way to get signal strength information is to use the tool such as *iw* to retrieve low level information. Based on the open-source wireless drivers, the *iw* tool reads the wireless-related values from PHY layer and MAC layer. The command "*iw dev wlan0 link*" gives us some basic information about the wireless link that we are measuring as the Figure 11 shows. In principle, the *iw* command reads the value from the */proc/net/wireless* file system which is designed to give some wireless specific statistics on each wireless interface in the system [2]. However, it is unclear how the wireless signal strength values have been processed before being stored in the */proc/net/wireless* entry. Thus we regard these results as inaccurate values. An alternative way is to perform passive sniffing by *Tcpdump*. Basically, *Tcpdump* can print out the description of the contents of packets and save the packet data on a network interface that match the boolean expression set by the command. Among the 802.11 frames captured in the *Tcpdump* traces, the BEACON frames are broadcast by APs periodically to indicate their existence and their signal strength. We retrieve the signal strength information from those BEACON frames.

```
technicolor@technicolor-Latitude-D630:~$ iw dev wlan0 link
Connected to 00:1d:7d:49:3d:e3 (on wlan0)
        SSID: demoroom
        freq: 5240
        RX: 54162 bytes (942 packets)
        TX: 26114 bytes (150 packets)
        signal: -48 dBm
        tx bitrate: 48.0 MBit/s
```

Figure 11: Iw signal strength results

4.2 Busy Time Measurements

2. http://www.hpl.hp.com/personal/Jean_Tourrilhes/Linux/Linux.Wireless.Extensions.html

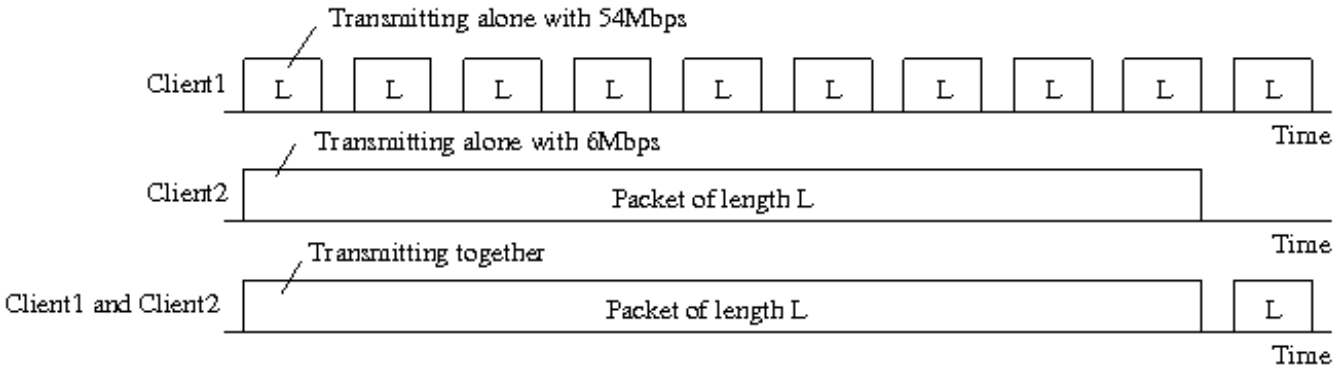


Figure 10: Different channel occupancy time by frames with different bit rates

As we said in the Chapter 3, the congestion is a significant problem in the real home networks due to the dense deployments and heavy usage. We use busy time to estimate the level of congestion in the network. The busy time metric indicates the fraction of time that the wireless medium is busy. In this section, we will introduce an existing way to get the busy time information along with its constraints, and our improved method of calculation and the procedure of our measurements.

4.2.1 Approach

In terms of measuring busy time, we found out that the *iw*[3] tool implemented for the Atheros cards provides us information about the busy time. The command "*iw dev wlan0 survey dump*" gives us the channel active time and busy time as the Figure 12 shows. We believe that the values are accurate because the command reads the values from */proc/net* file system which comes directly from the wireless driver. However, this tool is very limited in terms of wireless card types, it only supports a series of Atheros cards. Furthermore, it has also stability problems. During our experiments it sometimes turned out to be a looped value and even with a different loop range.

```
[:)][bee200] data%iw dev wlan0 survey dump
Survey data from wlan0
        frequency:                      5240 MHz
        noise:                          -81 dBm
        channel active time:            51109 ms
        channel busy time:              1254 ms
        channel receive time:           1236 ms
        channel transmit time:          32 ms
```

Figure 12: Iw busy time

Therefore, to improve the busy time measurements, Jardosh et al.[19] introduced a method of calculating the channel busy time based on passive monitoring. Their work can overcome the constraints of hardware and provide stable outcomes, thereby we developed our measurements based on their work.

First of all, we virtualise a wireless interface "mon0" from the interface "wlan0" which serves as AP by the command "*iw dev wlan0 interface add mon0 type monitor flags none*". The interface "mon0" is on "monitor" mode[4] which allows us to capture all the traffic that this wireless card can hear in its carrier sensing range. Secondly, we use *Tcpdump*[5] to perform passive sniffing on *mon0* and store the traces captured by *Tcpdump*. Since the wireless card in the gateway provides the wireless service for the connected clients, the traffic captured in this card can represent all the wireless traffic in this network and external traffic in the same channel. At last, we perform our analysis methods and calculations with these traces and get the busy time.

The busy time of the channel is calculated by adding the time of all data frames, management frames and control frames in that channel, and the total number of delay components like DIFS and SIFS. Even if during these delays there is no traffic in the medium, the channel is not considered as free so we must take them into account for a better accuracy.

We select one second of interval as granularity to calculate the busy time. In a second, the main possible frames captured could be DATA frames, ACK frames, BEACON frames and also RTS frames, CTS frames. Even RTS/CTS mode is not largely deployed, we still take them into account for more accuracy As illustrated before, for each kind of frame, the busy time consists of the time needed to transmit and also the corresponding IFS.

We define the delay component for each kind of frame as D, the CBT(Channel Busy Time) is computed as follows :

- DATA frames : $CBT_{DATA} = D_{DIFS} + D_{DATA}$
- RTS frames : $CBT_{RTS} = D_{RTS}$
- CTS frames : $CBT_{CTS} = D_{SIFS} + D_{CTS}$
- ACK frames : $CBT_{ACK} = D_{SIFS} + D_{ACK}$
- BEACON frames : $CBT_{BEACON} = D_{SIFS} + D_{BEACON}$

Considering that we have captured d DATA frames, r RTS frames, c CTS frames, a ACK frames, b BEACON frames, the total CBT in this second is computed as :

$$\begin{aligned} CBT_{TOTAL} = (d * CBT_{DATA}) + \\ (r * CBT_{RTS}) + (c * CBT_{CTS}) + \\ (a * CBT_{ACK}) + (b * CBT_{BEACON}) \end{aligned} \quad (1)$$

Then the percentage of channel utilization is calculated as :

3. http://wireless.kernel.org/en/users/Documentation/modes

4. http://wireless.kernel.org/en/users/Documentation/modes

5. http://www.tcpdump.org/tcpdump_man.html

$$U = \frac{CBT_{TOTAL}}{10^6} * 100\% \quad (2)$$

The three main 802.11 protocols a, b and g are very different in terms of transmission rates and encoding technologies. According to the standard ways to calculate delays in different protocols, the Table 3 below shows the values of delay components. For all the three protocols, their DIFS, SIFS delays and the delays of different frames are defined by the standard protocols[1][2][3]. Since the managements frames such as RTS, CTS, ACK and BEACON frames have fixed bit rates and lengths, we give here directly the calculated values. The delays of DATA frame will differ in terms of packet sizes and bit rates. In terms of the Back-Off time, we consider the network being highly utilized and the BO should be 0. In that case, at any time, a minimum of a single user is ready to send a packet which at least there is always a BO is counted down to 0.

Protocol	Delay components	Delay(μsec.)
802.11b	D_{DIFS}	50
802.11b	D_{SIFS}	10
802.11b	D_{RTS}	352
802.11b	D_{CTS}	304
802.11b	D_{ACK}	304
802.11b	D_{BEACON}	304
802.11b	D_{BO}	0
802.11b	D_{PLCP}	192
802.11b	$D_{DATA}(size)(rate)$	$D_{PLCP} + 8 * (\frac{34+size}{rate})$
Protocol	Delay components	Delay(μsec.)
802.11g	D_{DIFS}	34
802.11g	D_{SIFS}	16
802.11g	D_{RTS}	27.167
802.11g	D_{CTS}	25.58
802.11g	D_{ACK}	25.58
802.11g	D_{BEACON}	25.58
802.11g	D_{BO}	0
802.11g	$D_{DATA}(size)(rate)$	$20 + 8 * (\frac{34+size}{rate})$
Protocol	Delay components	Delay(μsec.)
802.11a	D_{DIFS}	28
802.11a	D_{SIFS}	10
802.11a	D_{RTS}	28.66
802.11a	D_{CTS}	26.667
802.11a	D_{ACK}	26.667
802.11a	D_{BEACON}	26.667
802.11a	D_{BO}	0
802.11a	$D_{DATA}(size)(rate)$	$22 + 8 * (\frac{size*8+22}{rate})$

Table 3: Delay components for 802.11

By our studies, the busy time calculated by this method has an error rate of three percent compared to the *iw*. Since we do not consider the Back-Off time and also the non-802.11 interferences(such as Bluetooth, Radio and Microwave which are operating on 2.4 GHz) which may also lead to the occupancy of channel, we regard our calculated busy time as a lower bound.

Our main contribution for improving the busy time measurements is to break down the busy time into finer details. Our method could help to map the busy time to external ones and internal ones, and even every single client in the internal network. This gives a more concrete view about how the channel is used by different networks and different clients.

To realize this, the passive measurement of associated clients must be performed along with the *Tcpdump* sniffing. This can be easily acquired by reading the wireless card values with command *"iw dev wlan0 station dump"*. The command could give us the list of connected clients, their MAC addresses, signal strength and transmission power. We record these values and also the current timestamps every second. Afterwards, we combine the *Tcpdump* traces and associated client lists to perform this operation.

In general, the procedure of the measurements(Figure 13) is as follows : First of all, we passively monitor the wireless network in the home network gateway, which consists of capturing *Tcpdump* traces and recording connected clients. Secondly, we map our traces to every connected client and external traffic. In the end, we parse our traces into the accurate percentage of channel utilization and output the global channel usage view. We will now detail each of these steps.

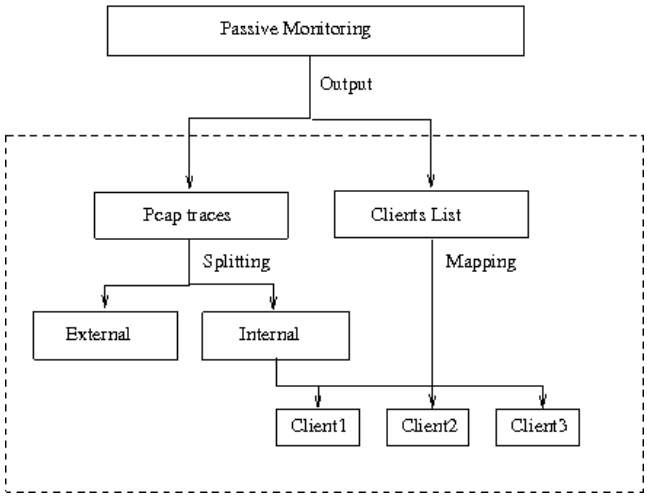


Figure 13: Mapping procedures

There are three steps of analysis. In the first step, we split the *Tcpdump* traces into two parts of internal and external traffic, by giving the MAC address of our access point. As stipulated by the 802.11 protocol, the general frame format of 802.11 frames has five address fields, which are sender address, destination address, BSSID address, transmitter address and receiver address. Each single frame has one or several these addresses depending on the type of the frame, to provide clear transmitting direction. Except the transmitter address which stands for the MAC address of the wired station that transmit the frame to the wireless medium, every frame which has a sender or destination or receiver or BSSID address corresponding to the given MAC address of AP can be considered as internal traffic because it means that this frame is either from or to AP, or traffic through this AP. In the second step, we combine our internal traces with the associated clients lists to map the traffic to each client in the network. The method is similar to the first one, when a match is found between client MAC address and one of the four address fields, the internal traffic could be splitted into each client's traffic. In the final step, we deploy the method presented before to calculate the accurate busy time for each splitted trace.

4.3 Hidden Terminal Detection

Recall that, hidden terminal is a big problem for home networks and there is still no way to detect it. In the ad-hoc networks, some academic efforts are made to identify the hidden terminals. Y.Li et al[32] introduces this method which takes an important feature of hidden terminals as a metric for detecting. We found out that this method is also applicable to the infrastructure networks. The nodes in the ad-hoc networks are equivalent to the devices in the infrastructure networks.

In the hidden terminal scenario of Figure14 and Figure15, node 0 and node 2 are hidden to each other. In the first case(Figure14), when node 1 is transmitting DATA to node 2, according to 802.11 protocols, an ACK must be sent from node 2 to node 1 following the DATA frame. Since node 2 is out of the carrier sensing range of node 0, so node 0 can capture the DATA frames sent by node 1 but not the ACK frames sent by node 2. In the second case(Figure15), when node 2 is sending DATA to node 1, node 0 can only capture ACK frames sent by node 1 but not DATA frames. The metric which can represent the probabilities of hidden terminals can be the fraction of incomplete DATA/ACK pairs or ACK/DATA pairs.

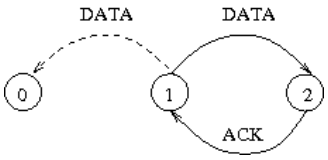


Figure 14: Hidden receiver

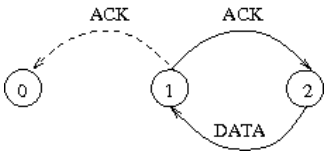


Figure 15: Hidden sender

4.3.1 Approach

In this measurement, as shown in Figure 16, we only do passive monitoring and collect the pcap traces. First of all, we filter the traces and get only DATA frames and ACK frames. We also exclude every single packet which has a sender address or destination address or receiver address the same as the device where we are capturing. Because the capturing device cannot be the hidden terminal of itself. In our case, since we are capturing in the gateway, the clients in the same network cannot be hidden terminals for the AP. We mainly focus on the other neighborhood networks.

After filtering, the frames we still have are basically the frames from other networks in the same channel which help us to identify the devices hidden from us. The analysis part is all about if the data frames have corresponding ACK frames, and if the ACK frames have corresponding DATA frames.

For the DATA frames, the possible hidden terminals are the destinations of those DATA frames, which are also hidden receivers. If a hidden terminal scenario appears, we will not capture an ACK frame immediately after this DATA

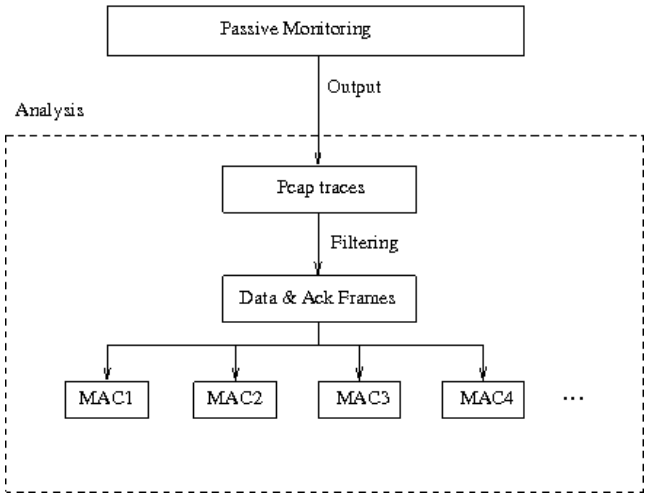


Figure 16: Measurement procedures

frame if the DATA frame is not retransmitted. The whole verification process is illustrated in the Figure 17. If we point at a current DATA frame, we must verify if its next frame is ACK frame or not. If not, then the next frame must also be a DATA frame since our filtering only leaves us ACK and DATA frames. We investigate at the sequence numbers of these two frames, if they are the same, then it means the current frame is retransmitted, so it is normal that it does not have acknowledgment. In this case, we categorize it as "Not Hidden" to simplify the analysis. If the retransmission condition is not satisfied, then this DATA frame would be recorded as "No ACK", hence we call this the hidden receiver case, the possible hidden terminal device is the destination address of this DATA frame. In another branch, if the next frame is ACK frame, we must first look at their frame numbers since our filtering may eliminate the other frames between them. The successful pair of DATA and ACK must have consecutive frame numbers. The frames who do not satisfy this condition are also considered as "No ACK". Then at last, the sender of the current DATA frame must be the same with the receiver of the next ACK frame. As long as all these conditions are fulfilled, a successful DATA/ACK would be identified.

For the ACK frames, the basic process of identifying is very similar to the data frames except that there is no retransmission verifications. As the Figure 18 shows, an ACK frame which satisfies the three conditions of having a preceding DATA frame, consecutive frame numbers and the match of addresses would be regarded as a good pair of ACK/DATA. The frames marked with "No DATA" would have their receiver addresses as suspected hidden senders.

Generally speaking, after the process of analyzing the traces, we get a list of suspected hidden receiver and hidden sender. For each of them, we will have the statistics of successful and unsuccessful pairs. By calculating the fractions, we can infer for certain devices their possibilities of being hidden terminals.

5. EXPERIMENTS SETUP

In this section, we will describe our two main experiments, one is the controlled experiments in the lab, another one

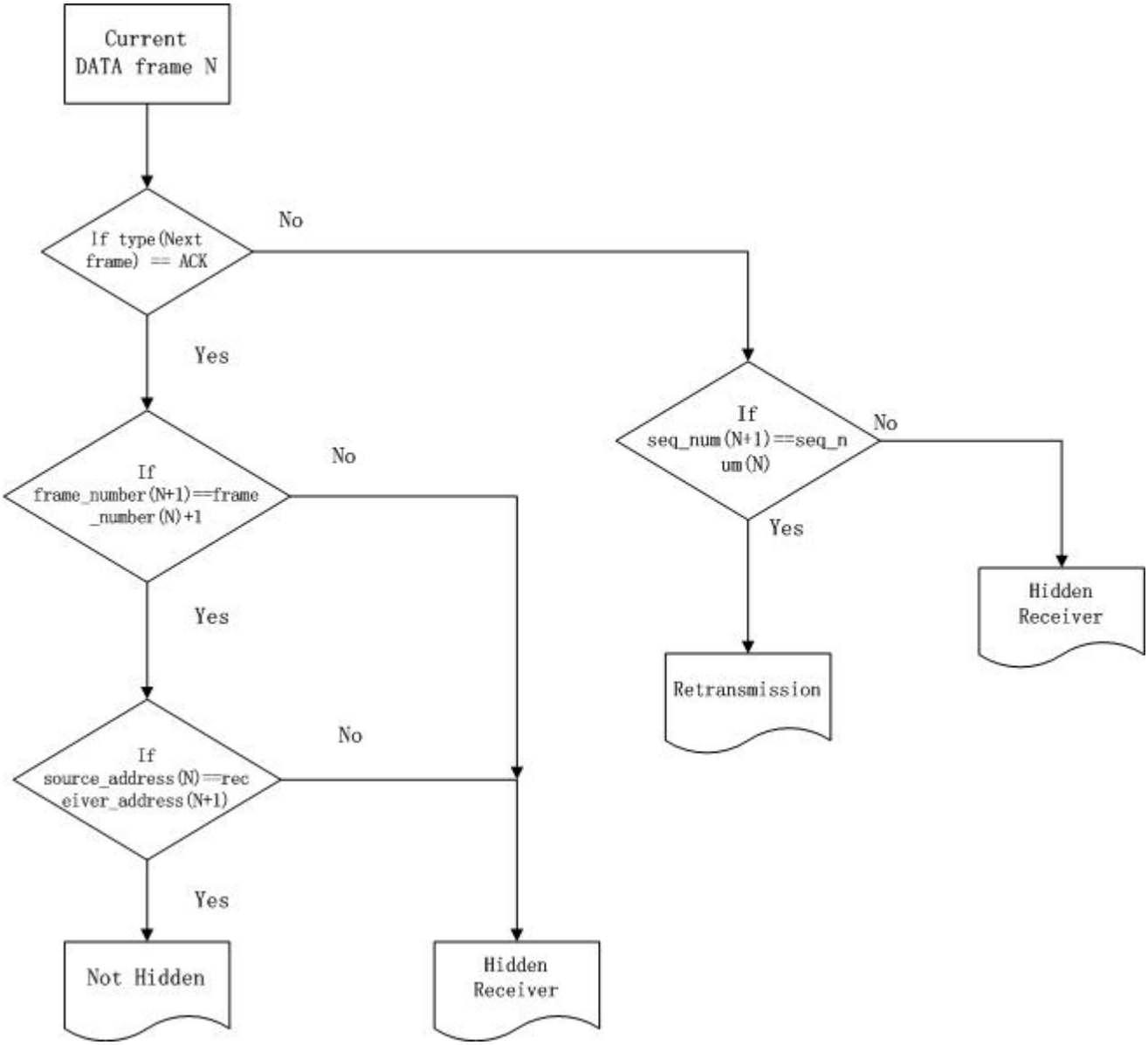


Figure 17: Hidden receiver detection

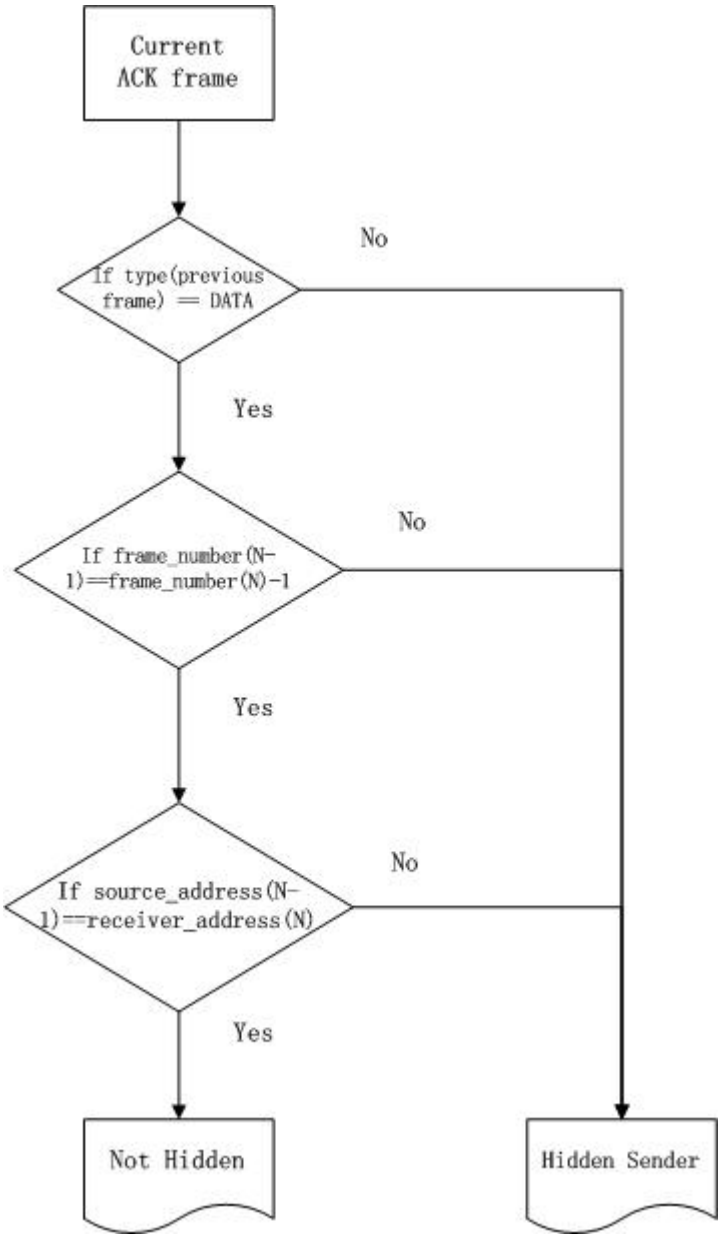


Figure 18: Hidden sender detection

is the experiments in real home networks, including their settings, parameters of hardware, measurements performed and environment mappings.

5.1 Experiment Principles

The results in this report are mainly obtained by two experiments, one is the controlled experiments in the laboratory of Technicolor, the other one is the passive experiments in a real home network. The motivation for doing controlled experiments in the laboratory is to verify some basic theoretical guesses and also the correctness of our methodology for analysis. The created controlled environment can aid us to strictly follow some main principles when we do wireless measurements. These principles are reflected as follows :

- Clean Channel : the controlled tests are in a clean channel, which means almost no other neighborhood access points are in the same channel, most of the traffic in this channel can only be generated by our experiments.
- Independent Usage of Devices : during the experiments time, we have to make sure that the devices only run our process which make the results scientifically independent from any other processes. In our case, night time and weekends are chosen to perform those controlled experiments.
- Back to Back : the different test scenarios must be performed back to back to ensure the most similar network environments.

After doing controlled experiments, it is essential to apply our measurements and analysis methods to real home environment traces. Since collecting data at home is more or less related to the user privacy, we only do the measurements at a well selected home, and also only passive monitoring for not impacting the owner's normal network usage.

5.2 Controlled Experiments

5.2.1 Environment

The controlled experiments are realized in a big enclosed room. The room is inside the Technicolor building which is situated besides the peripheral roads of Paris. The room is in the second floor, east wing of the building, surrounded by open offices and enclosed offices. The 2.4 GHz band is quite occupied, we could detect a dozen public or private WiFi signals. The four main 5GHz channels (36, 40, 44, 48) are used by the cooperation networks. Because of the limits of our cards, the measurements have to be taken in the channel 48. But to avoid the cooperation network traffic, the measurements are always performed in the night time or weekends. The analysis of our traces also indicates very little traffic coming from outside of our own network.

5.2.2 Setup

In the controlled experiments in the laboratory, four main devices are used. Here we give a table 4 of the main parameters of four machines.

The measurements are all based on these four devices. First of all, a main home network is built with the Shuttle XS35 as AP and Dell Latitude D630(the second item in the table 4) as a client. We measure the metrics in both sides. To fully understand the status of the experiments, we run both network measurements and local machine status measurements. The explicit tests are shown in the Table 5. Then another neighborhood network is set up close to the main network and also in the same channel 48 by the other two devices (Dell Latitude D630 as AP and Acer AOA110 as client) to interfere with the main network.

5.2.3 Scenarios

In the controlled experiments, we tried five different scenarios with different configurations. The duration for each one is two hours. The intention was to verify our ideas about problems and also to verify the correctness of our analysis methods.

Baseline.

In this baseline test, we only setup our initial configurations without generating any traffic. As the Figure 19 illustrates, there are only an access point and a client in the network, without communications between them yet. The test helps to verify the cleanness of the channel and the correctness of all our settings. As long as the environment is as clean as we think, the metrics would not give us any surprised results. With this straight baseline, we can be sure about the results of the next experiments come from our different settings or active measurements.

Baseline with iperf.

In this test(Figure 20), we tried to emulate a simple user case in the real home network. With the same setting as the first scenario, a client in the main network is exchanging data frames with the AP. The scenario is realized by sending bidirectional iperf UDP packets which also helps us to measure the throughput of the wireless. Every two minutes, the client will first perform a 15 seconds' UDP download stream and then immediately after that a 15 seconds' UDP upload. The motivation for performing this experiment is to get the best performance of wireless in an experimental pure environment. The results from this baseline will be base for comparing results from the next tests.

Congestion.

In this test(Figure 21), the scenario is to have two networks competing for the medium at the same time and generate as much traffic as they can. This is very alike to the same case in the highly congested network environment. The motivation of this test is to investigate the behaviors of the two networks and the influence of congestion.

We set up another network which operates at the same channel as the main one. There are also an AP and a client inside that network. All these four devices are close enough and they are in the carrier sensing ranges of each other. The downloads and uploads of an Ubuntu image file were set between the AP2 and C2 to represent real traffic of neighborhood network. In the first hour of the test, every twenty minutes a download was performed, and every twenty minutes an upload was performed in the next one hour.

Hidden Terminal.

In this test(Figure 22), we would like to artificially create the hidden terminal scenario to try our detection method. We changed the positions of the two clients. The C1 is out of the carrier sensing range of C2, the purpose was to create the scenario of hidden terminal. It is obvious in the graph that the C2 is a hidden terminal for C1. Thus we would like to see if the measurement results captured in C1 correspond to our expectations about the incomplete DATA/ACK pairs.

Device Type	Role in the Experiments	Kernel Version	CPU	Memory	Wireless Card Type
Shuttle XS35	Main Access Point	Linux 3.2.0-26	Intel Atom(TM) D525 1.80GHz	2GB	Atheros AR242x/AR542x 802.11abg
Dell Latitude D630	Main Client	Linux 3.0.0-12	Intel Core(TM)2 Duo CPU T8300 2.40GHz	2GB	Atheros AR242x/AR542x 802.11abg
Dell Latitude D630	Neighborhood Access Point	Linux 3.0.0-12	Intel Core(TM)2 Duo CPU T8300 2.40GHz	2GB	Atheros AR242x/AR542x 802.11abg
Acer AOA110	Neighborhood Client	Linux 3.0.0-12	Intel Atom(TM) CPU N270 1.60GHz	1GB	Atheros AR242x/AR542x 802.11abg

Table 4: Devices used in controlled experiments

Metric	Test Tool	Measurement Side	Frequency	Other Parameters
Associated Clients	iw dev wlan0 station dump	AP	1 sec	None
CPU Usage	dstat	AP	1 sec	None
Channel Busy time	iw dev wlan0 survey dump	AP	1 sec	None
Throughput	iperf	AP & Client	2 min	Duration of 30s, 15s for each direction, bandwidth 40M, UDP data streams
Network Traffic	Tcpdump	AP & Client	Always	Capture size of 200 Bytes
RTT	Ping	AP	1 sec	From AP to all connected clients
Signal Strength	iwconfig	Client	1 sec	None

Table 5: Measurements in controlled experiments

Figure 19: Baseline

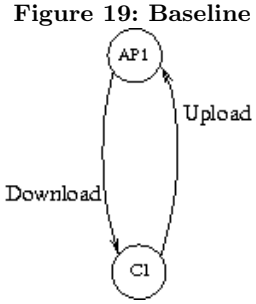


Figure 20: Baseline with iperf

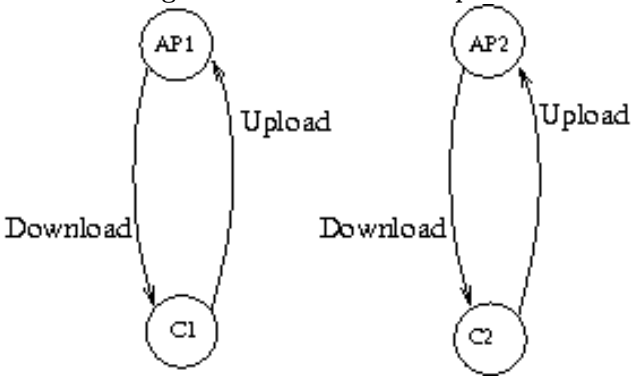


Figure 21: Congestion

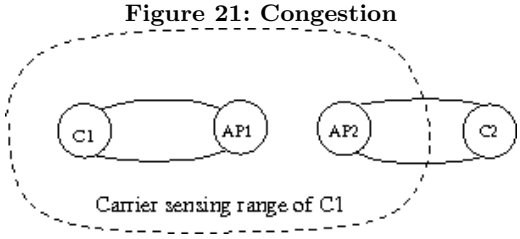


Figure 22: Hidden terminal

Low Bit rate.

In this test(Figure 21), we reuse the scenario of the congestion experiments. In the normal case, the bit rate of wireless cards is chosen by the rate adaptation algorithm according to the signal strength. In the previous tests, since C2 and AP2 are close enough, C2 always has a bit rate of 54 Mbps or 48 Mbps (the best bit rates for 802.11 a). In this test, the bit rate of C2 is forced to be 6 Mbps which is the lowest bit rate of 802.11a. The motivation of this test is to investigate the impact of a low bit rate client in the network.

5.3 Real Home Experiments

5.3.1 Environment

The real home experiments are deployed at a selected home which situates in the south of Paris, in a residence for young workers. The rooms are all small studios, differs from $12m^2$ to $18m^2$. Our selected room is in the end of the second floor. A simplified view of the front face of the building is given in the Figure 23. The cubic marked with the window is the home for our experiments. The wireless conditions at that home are quite crowded. From our investigation, we can detect 28 to 42 neighborhood APs at this home. It is obvious that the rooms in the center of the map may suffer even more from this high density. The home is quite small, so the devices are in a distance of less than 2 meters to the AP.

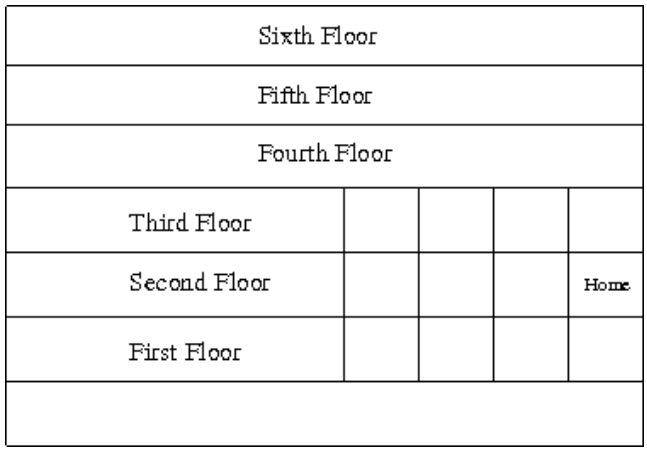


Figure 23: Front map of selected real home

5.3.2 Setup

For the real home environment, the same type of device(Shuttle XS35) as the controlled experiment is used as AP. The two main clients in this network are a cellphone and an iPad 2. The main parameters of the devices in this Wireless LANs are given in the Table 6

In the real home experiments, in order not to impact the normal usage of the owner's network, only passive experiments are setup in the home experiments, and only in the AP side. The test plans are listed in the Table 7

The owner's normal activities with the two wireless devices(cellphone and iPad) related to the Wireless LANs usages are watching videos, listing to Internet-based music radios, download magazines, books and applications. Sometimes the owner uses his iPad to download movies but he

Device Type	Role in the Experiments	OS Version	CPU	Memory	Wireless Card Type
Shuttle XS35	Access Point	Linux 3.0.0-12	Intel Atom(TM) D525 1.80GHz	2GB	Atheros AR242x/AR542x 802.11abg
Samsung GT-I9100	Client	Android 2.3.6	ARMv7	Unknown	Unknown
IPad 2	Client	IOS 5.1	A5 1GHz	Unknown	Unknown

Table 6: Devices used in real home experiments

never keeps downloading big HD movies or running P2P streams all the time. So we could regard the owner as a average network user, which would be similar to a large number of users.

6. RESULTS & ANALYSIS

In this section we will present the results from our experiments and also the analysis. Generally, our results correspond well to our expectations about the wireless network problems and show interesting application possibilities based on our results. We will also introduce a demo "Measurements as a service", realized by staff of Technicolor Research Laboratory. The demo was presented in the Technicolor Research&Innovation open days, and appealed to many network service providers and also gateway manufacturers. Our work was included as an important part of the demo, which used simple web interface to show WiFi status of home network and existing problems.

6.1 Controlled Experiments

6.1.1 Congestion

In the congestion experiments, two networks in the same channel competes for the channel. Since both of them are sending the maximum data they can, a congestion would cause a sharp decrease in both of their throughputs. The results of this experiment confirm our expectations. As the Figure 24 shows, compared to the baseline with iperf in the figure above, the arrival of the traffic of another network degrades the main network, and at the moment when both of the networks are trying to get the maximum utilization, the channel seems to be fair. Both of the results decrease to half. Moreover, in the zoomed Figure 25, the moments when this happens are exactly when both of the traffic are in the channel. The results of iperf(Figure 26) also confirm our ideas, the throughputs of both uplink and downlink of the main network decrease in the same moment when the congestion happen.

6.1.2 Hidden Terminal

In the hidden terminal experiments, the main purpose is to verify the correctness of our detection method of finding unpaired DATA/ACK frames and the corresponding hidden sender and receiver based on that. By setting the client in the main network as a hidden terminal for the client in the other network, we compare the traces captured from this client to look at the fraction of unpaired DATA/ACK frames.

In the Figure 27, the test scenario corresponds to the congestion scenario as Figure 21, the neighborhood client C2 is downloading from AP2, the DATA frames which have a destination address of C2 are also captured in the C1. It is well illustrated that for these DATA frames, the fraction of having corresponding ACK frames is more than 99%. The second column corresponds to the uploading of C2, most of the ACK frames captured by the C1 find their corresponding DATA frames. In this situation, we can conclude that C2 is not a hidden terminal for C1.

In the Figure 28, the first column corresponds to the hidden receiver in Figure 14, the same as the previous one, C2 as node 2 is downloading, for the DATA frames captured by C1 as node 0, only 45% of their ACK frames have been observed. On the other hand, as the second column in Figure 28 shows, almost no ACK frames have their corresponding DATA frames captured in the hidden sender scenario15. We can identify C2 as both a hidden receiver and a hidden sender. This result corresponds well to our detection method.

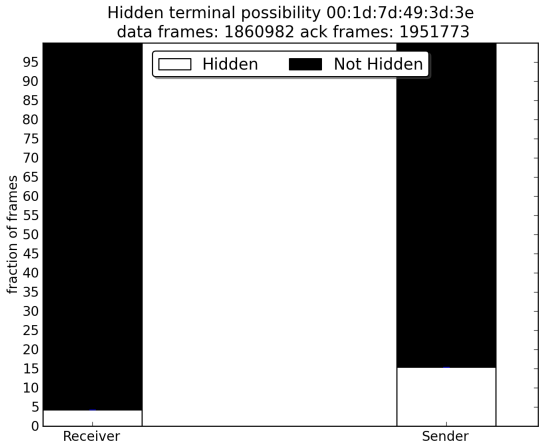


Figure 27: Probability of being hidden terminal

6.1.3 Low Bit rate

In the Chapter 3, we discussed about the harm of low bit rate devices to the whole network. The scenario of this experiment is shown in Figure 21, where the bit rate of C2 was forced to be set at 6Mbps, the lowest bit rate in the 802.11a protocol. As illustrated in the Figure 29, when the client C2 performs downloading where the DATA streams are given bit rates of AP2, so the results are similar to the the results of congestion(Figure 24). However, as C2 is uploading, with the given bit rate 6Mbps, the C2 starts to occupy completely the channel. The external utilization goes above 80%, while the main network suffers a great drop in utilization and also in throughput(Figure 30).

Metric	Test Tool	Measurement Side	Frequency	Other Parameters
Associated Clients	iw dev wlan0 station dump	AP	1 sec	None
Network Traffic	Tcpdump	AP	Always	capture size 150 Bytes
Surrounding APs	iwlist wlan0 scanning	AP	1 min	None

Table 7: Measurements in real home experiments

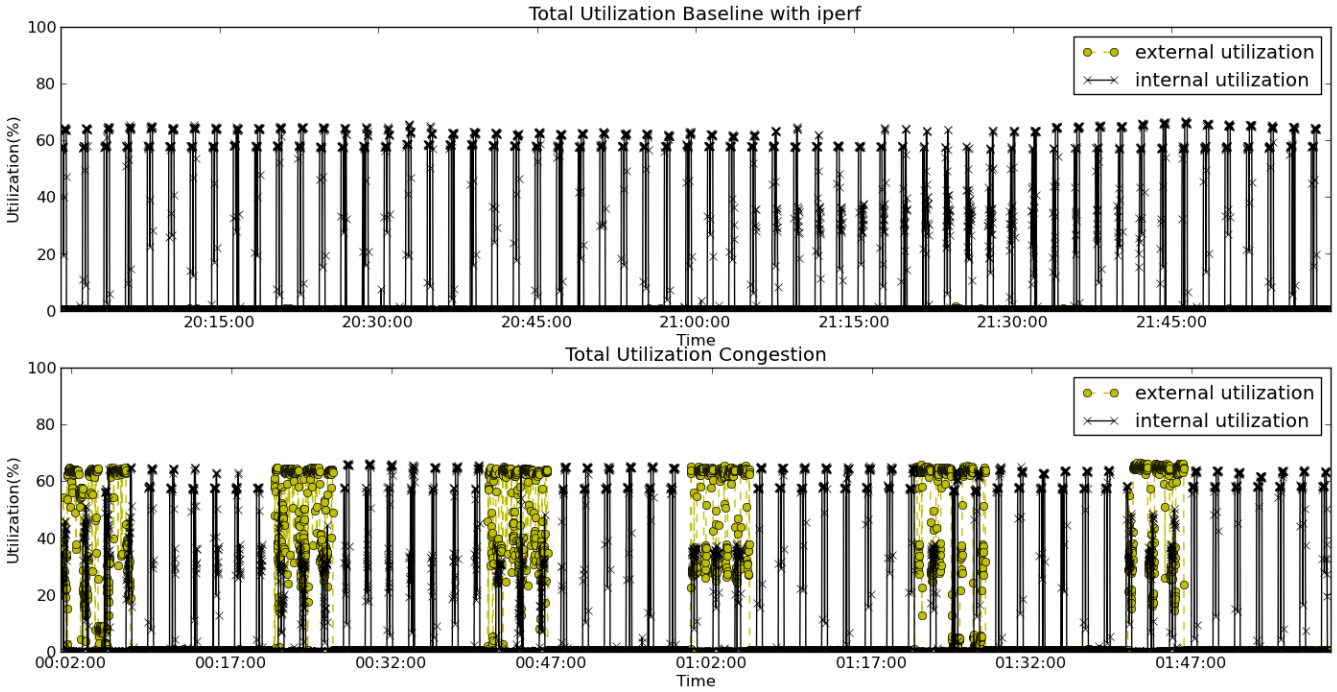


Figure 24: Channel utilization of baseline with iperf and congestion

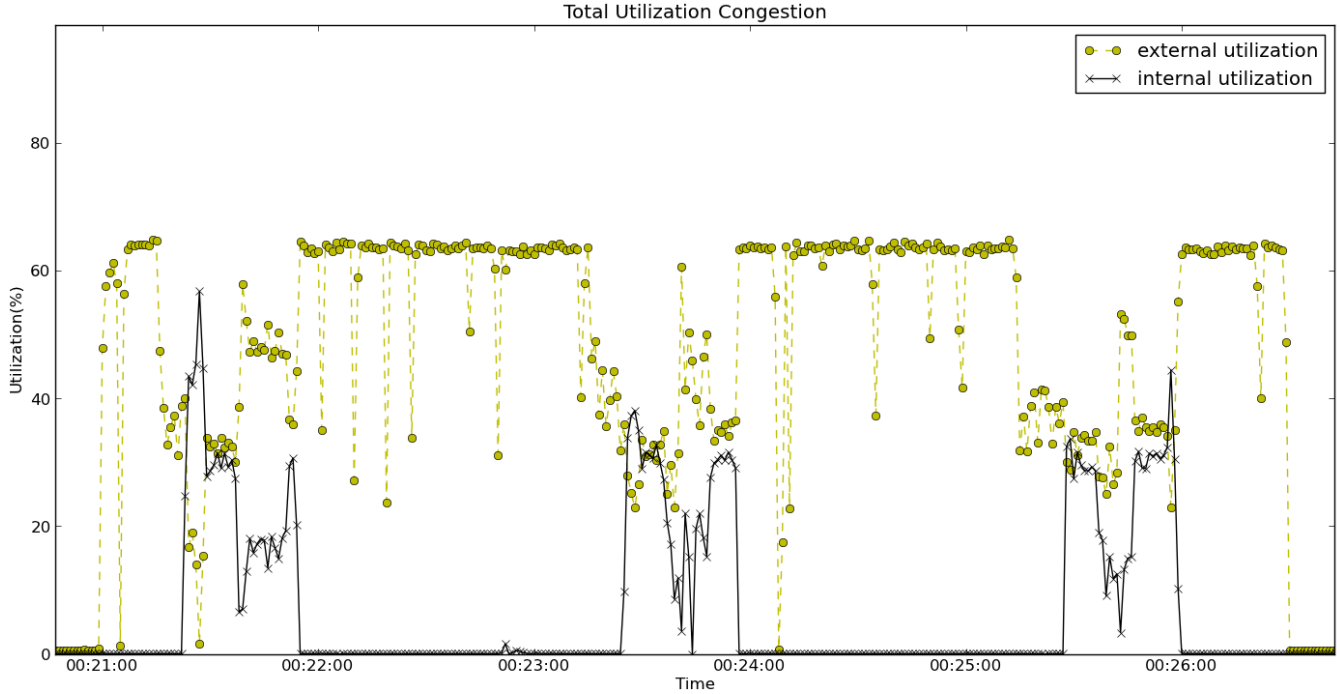


Figure 25: Channel competition

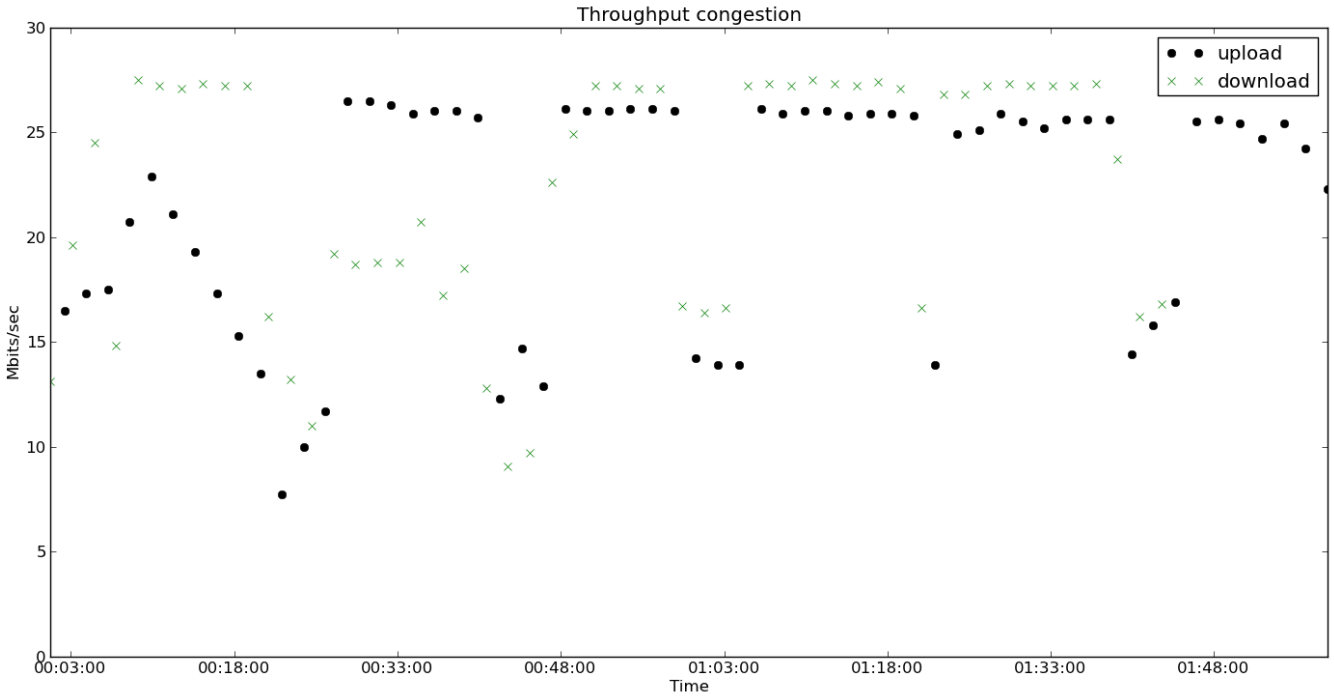


Figure 26: Throughput of congestion experiment

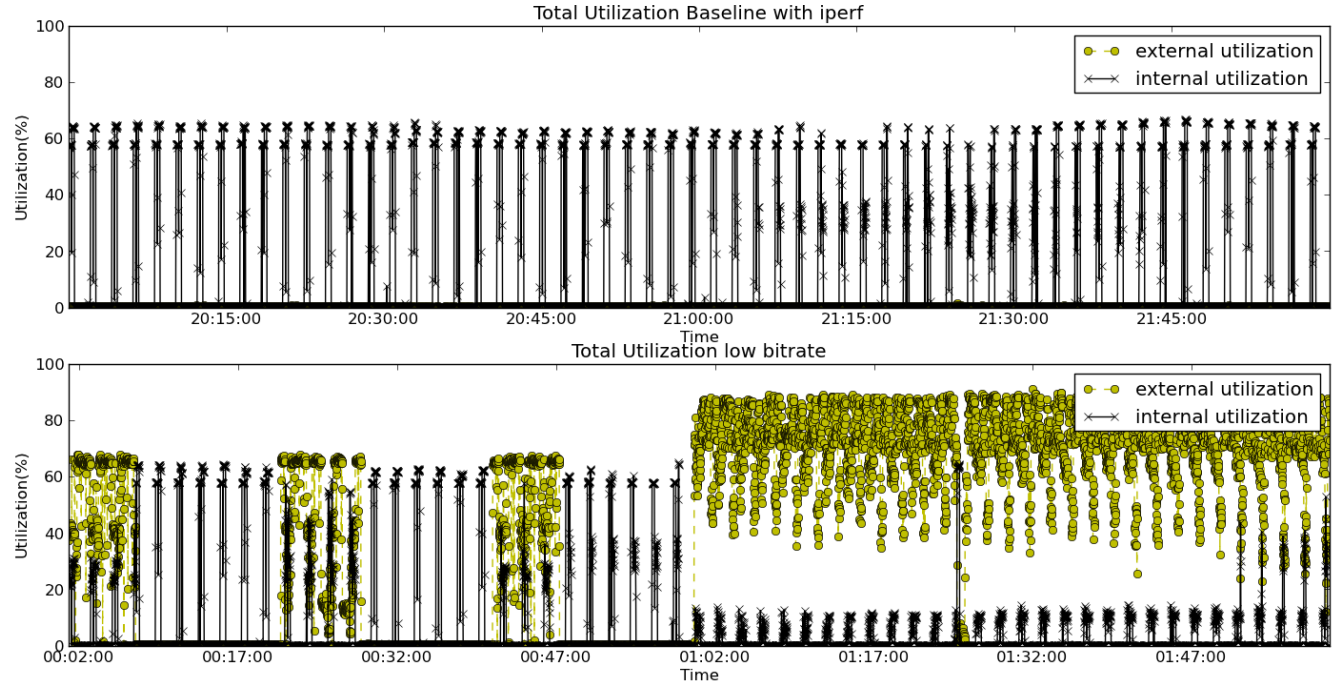


Figure 29: Channel utilization of low bit rate experiment

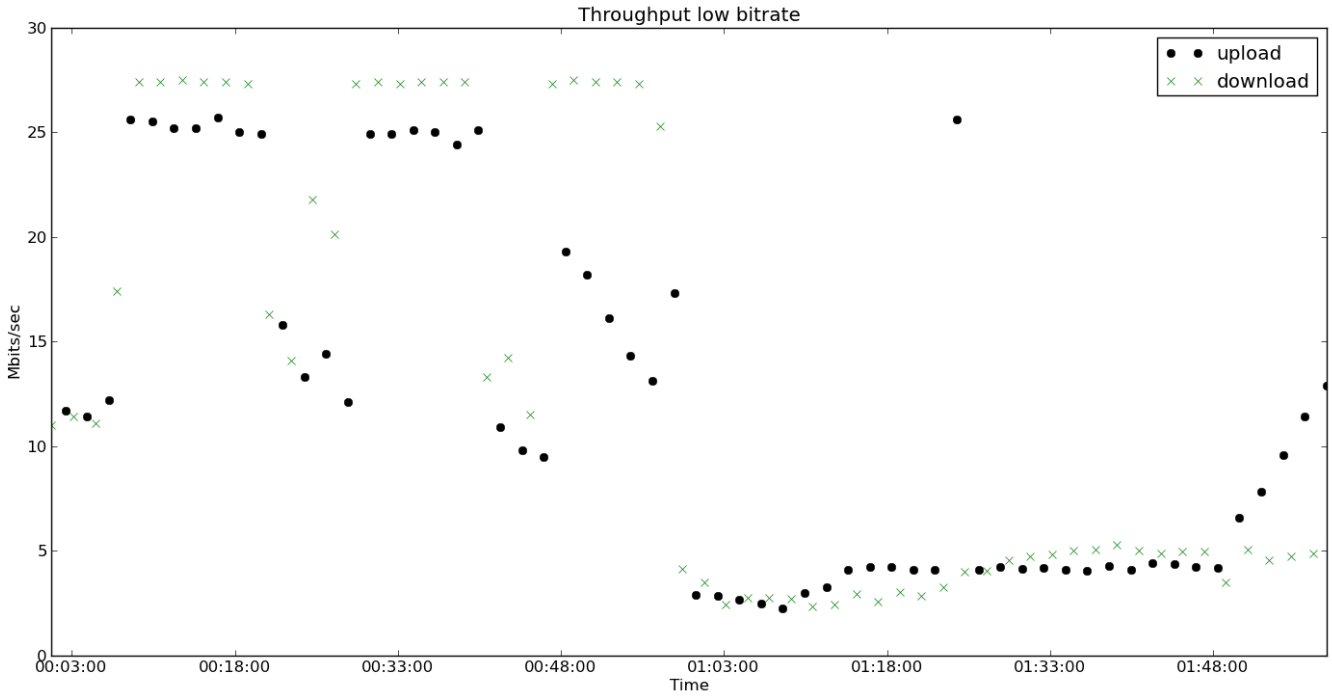


Figure 30: Iperf results of low bit rate experiment

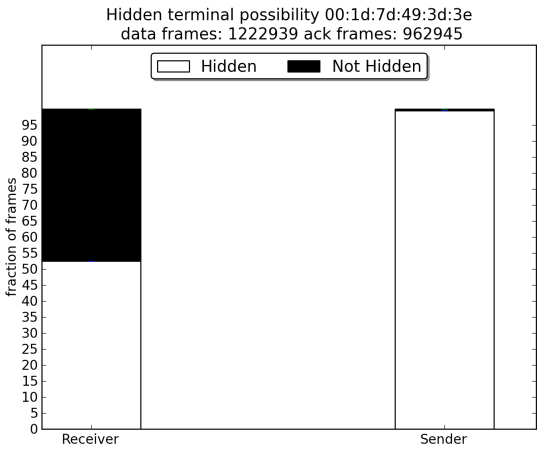


Figure 28: Probability of being hidden terminal

The results are not difficult to understand, as the Figure 10 shows, for the same size of packet, the device using a bit rate of 6Mbps would cost 9 times than the device operating in a bit rate of 54Mbps. The algorithm tried to be fair about the times for clients to access the channel, while it results in the unfair distribution of time.

6.2 Real Home Experiments

In this section we show some initial results from measurements deployed in a selected home. The goal is to understand what are the most common problems in real home Wireless LANs. But limited to the lack of devices and user privacy problems, we can only realize this single test which may not be representative. Therefore, the results are giving us the first ideas of how the real home networks are.

6.2.1 Congestion

In the Figure 31, we give a precise channel usage of our home network during about six days. Some observations are very clear in this graph.

- In the graph above, we can see that the channel is not extremely highly used, there are only a few moments when the utilization goes above 60 which means the general utilization of our network is healthy.
- The competition between our network and external networks exist and sometimes the impact for our traffic can be huge when the external utilization is very high.
- In the graph below, the channel seems to be fair for both of my two clients as the averages of the max usage are the same.
- It shows clearly when the client ”cellphone Galaxy SII” has or has no traffic, that corresponded the time when I was at home and I was not at home. We think it is very useful to study the user behaviors from the networking

traffic's point of view.
– In the zoomed area of the graph, the "cellphone Galaxy SII" seems to have a traffic peak per minute. We believe it comes from the nature of mobile applications which refresh messages every minute.

6.2.2 Hidden Terminal

As we introduced before, our measurements show us a big number of surrounding APs in our home network, thus there is a high probability of having quite a lot of devices being hidden terminals. Our results confirm this hypothesis. 122 devices are observed as possible hidden terminals. Since for most of the devices we have captured too little traffic and the results can be just packet loss because of the weak signals, the 10 most significant suspected results are shown here. Figure 32 shows us the hidden receiver case, we can see that terminals 1,4,5,7,8,9 seem to be total hidden as they do not receive any ACK frames after DATA frames, while the other terminals are still under suspected because the low fraction could indicate partially hidden terminal or natural frame losses. Figure 33 illustrates the hidden sender case for the same 10 devices. Obviously, by combining the results from both of the two cases. terminals 1,4,5,7,8,9 are confirmed to be hidden terminals.

The terminals 1,3,4,7 are partly hidden because the device may be moved or varying in terms of signal strength. But the others seem to be totally hidden from us.

Moreover, a detailed view about when the hidden terminal situation happens is more useful for further troubleshooting. Figure 34 and 35 gives us the uncompleted DATA/ACK or ACK/DATA pair numbers with timeline for the five most significant devices. For the hidden receiver case in Figure 34, the suspected devices 2 and 5 seem to be the client devices, because they follow the work time pattern. When they are generating the most traffic as their peak points in the graph, the hidden terminal problems can be the most severe at that time. The same as the hidden receiver case, the hidden sender graph(Figure 35) confirms again the same time point when hidden terminal severity can be the highest.

7. RELATED WORK

A number of studies have been done in the home networking. The most relevant to our work is the previous work on Wireless LANs diagnosis and trouble shooting. In terms of measurement tools, many efforts have been spent on the extraction of low level information. Some commercial tools are developed to troubleshoot the wireless networks by retrieving data from some hardware sensors [6] [7]. MOJO[28] is a unified framework to diagnose physical layer faults that are commonly observed in existing 802.11 based wireless networks. Adya et al.[4] uses wireless clients and enhanced APs to do the sensing and feed information into a back-end server for analysis.

But all these previous solutions focused on the network operators' point of view, which is more suitable to the IT department of big enterprises and campus networks. But our work concentrates on the end-users, the idea is to give both ISPs and end-users the simple understanding of their networks. The users can help themselves to find the problems, or ISPs could quickly identify the problems when they could know more about the status of their users' networks. The massive deployment of WiFi technologies and the nature of 802.11 protocols lead to users' frustrations. Poole et al.[27], Chetty et al.[12] and Grinter et al.[16] all describe home users' current frustrations with the network management tools available to them.

Some systems start to emerge the troubleshooting capabilities of home network gateway to perform monitoring[23, 26]. Chetty et al.[11] introduces the Home Watch System which takes a enhanced home router to perform monitoring, the results are used to build a visualization system that allows users to know about their bandwidth usage. Yang et el.[31] presents a similar tool which uses a modified router to collect data and provide capabilities for users to manage their network functions. For enterprise networks, tools like NetMedic[22] have demonstrated the power of network instrumentation to support diagnosis.

In terms of the concrete problems, many efforts have been spent on the accurate calculation of wireless throughput[5, 18]. But few of them take advantage of their methods and investigate into the wireless congestion problem. Jardosh et al.[19] and Lakshminarayanan et al.[25] study the congestion problem with different ways to retrieve the busy time. The former one performs accurate calculation by computing and accumulating the delay components in the network. The latter one takes the busy time value from low level information. The difference between our work and their is that we extend the single channel busy time to a concrete view of network traffic, including both internal and external, even each client in our networks. Also the low level information is used only to be referenced and to remedy errors because of its limitations. The deployment of our method would be a great tool for studying wireless problems and user behaviors.

For hidden terminals problem, the 802.11 protocols suggest us to operate in RTS/CTS mode. But the great overhead sometimes affects our performance more than the hidden terminal itself. Thus the detection of hidden terminals is essential. Most of the related work is realized for Ad-Hoc networks[32, 20]. In fact, we found out that these methods also apply for infrastructure mode. The methods which only perform passive monitoring and do not create any overhead itself help us to identify the number and also MAC addresses of hidden devices. We believe once we can know about the severity of the hidden terminal problem, the RTS/CTS solution could be deployed once the influence of hidden terminals exceeds the influence of its own.

8. CONCLUSION AND FUTURE WORK

The analysis of the Wireless LANs is crucial for the robust operations of such networks. To this end, this report has presented the analysis of several well-known problems like Low Signal Strength, Congestion, Hidden Terminal and etc. We have investigated at the nature of these problems and proposed the way to measure and detect. Then to study the correctness of our analysis, the controlled experiments in the laboratory and in the real homes have been performed. We have deployed many monitoring and measurement tools, both active and passive ones, to collect the data from network traffic and perform the analysis. The experiments made in the laboratory confirm our expectations of the problems, and prove the correctness and efficiency of our detection and analysis methods. Observations made from our real home network suggest that hidden terminal could be the most

6. http://www.arubanetworks.com
7. http://www.airtightnetworks.com/

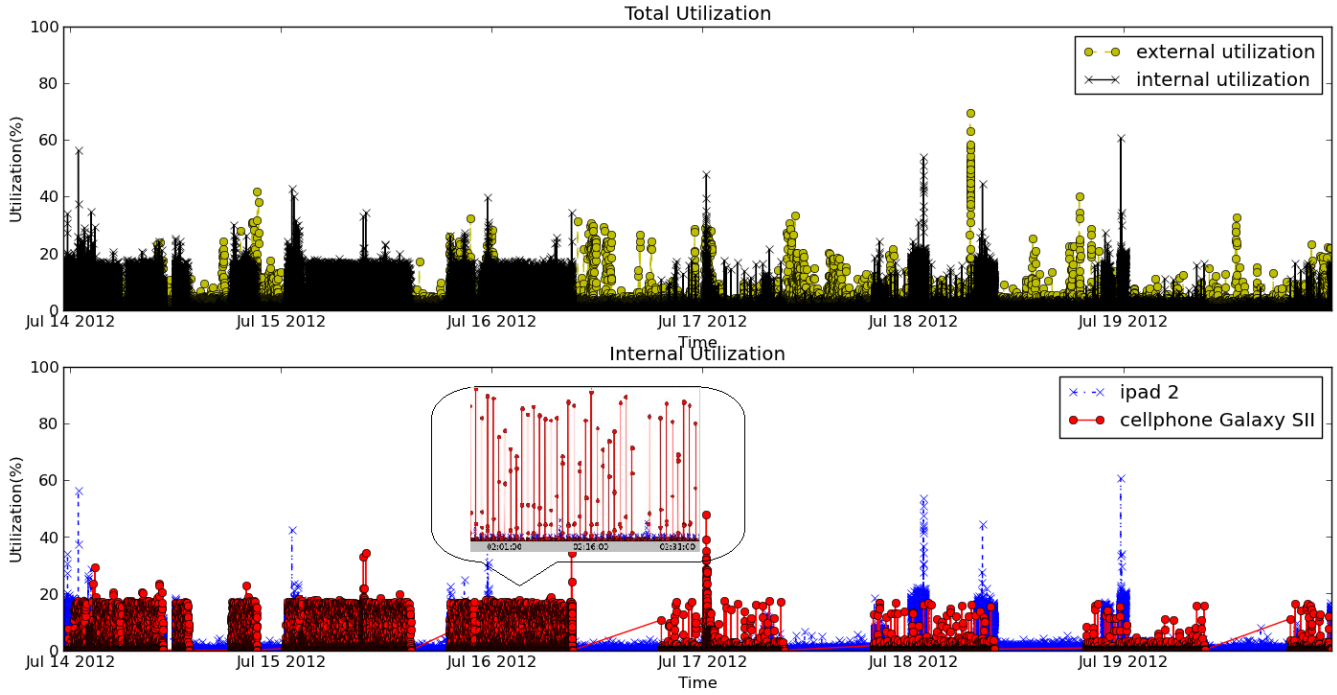


Figure 31: Channel usage

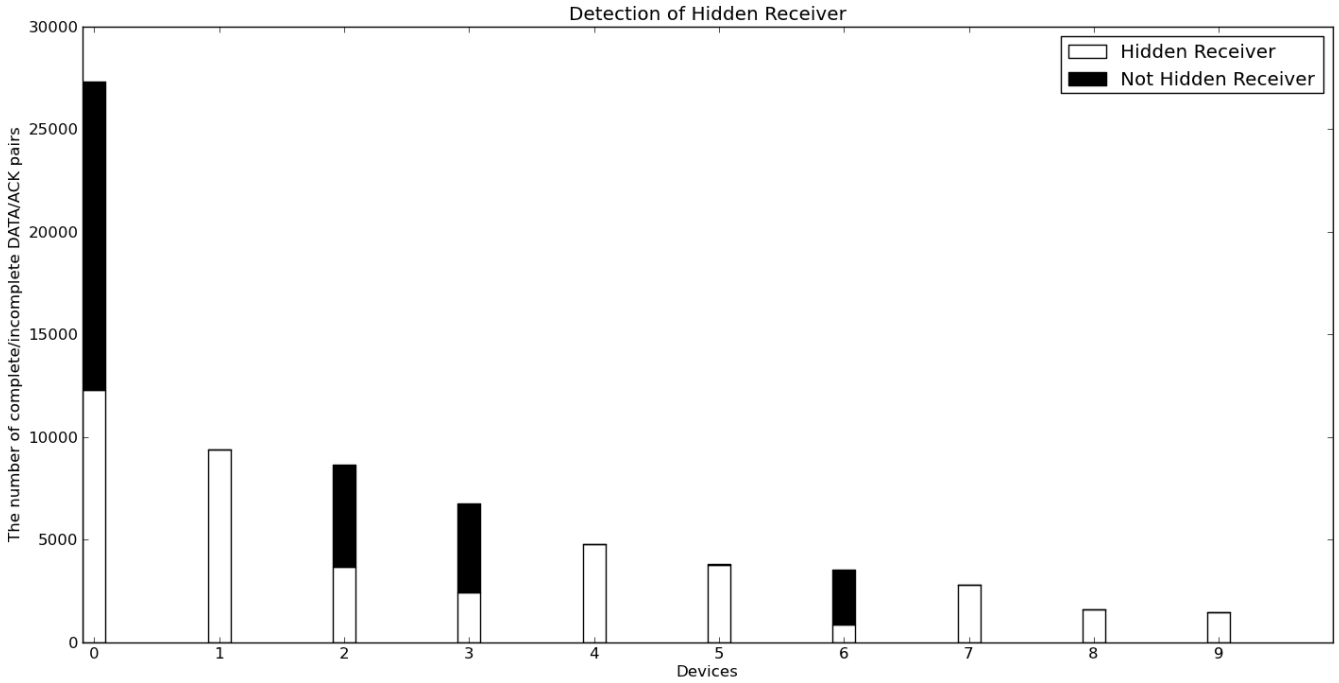


Figure 32: Hidden receivers

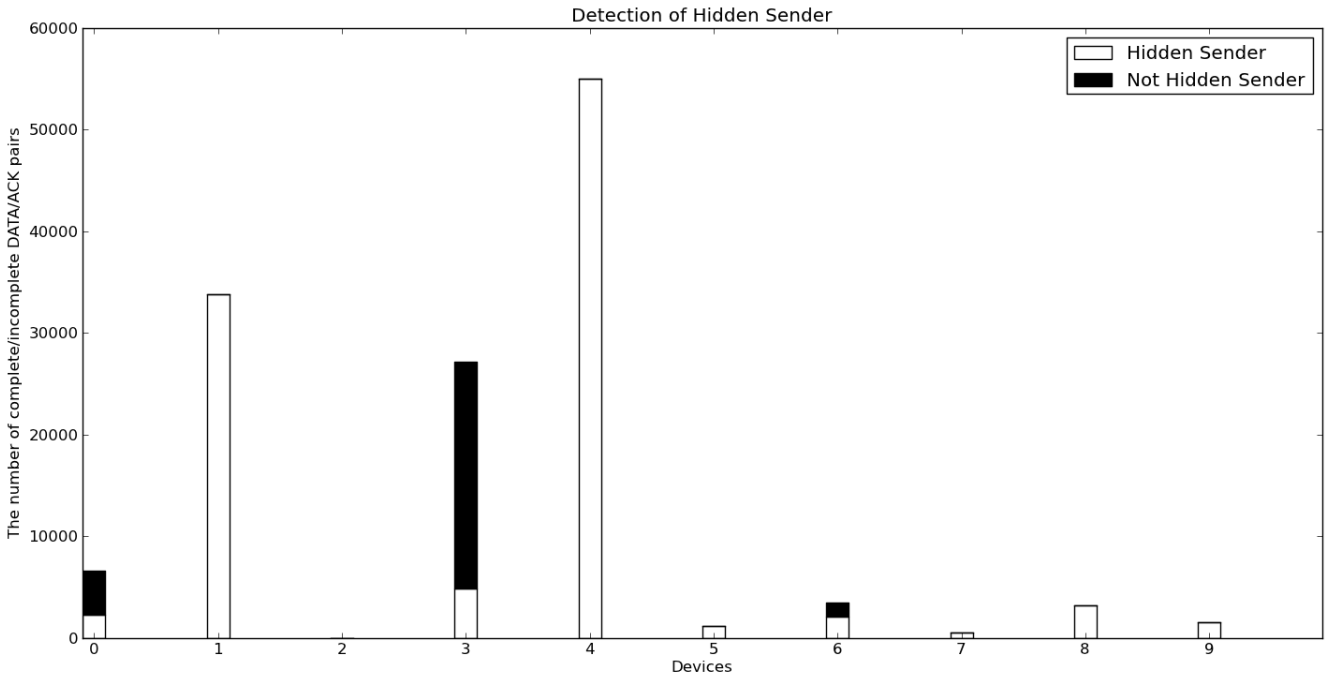


Figure 33: Hidden senders

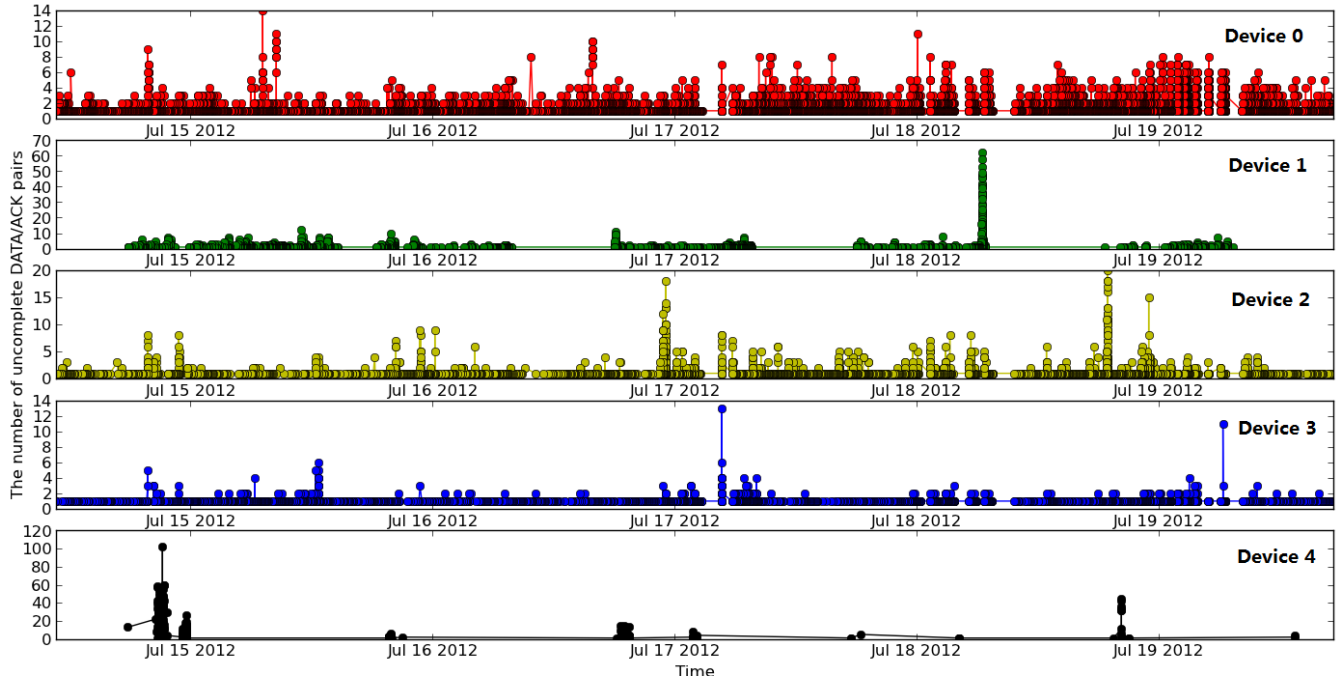


Figure 34: Hidden receivers with timeline

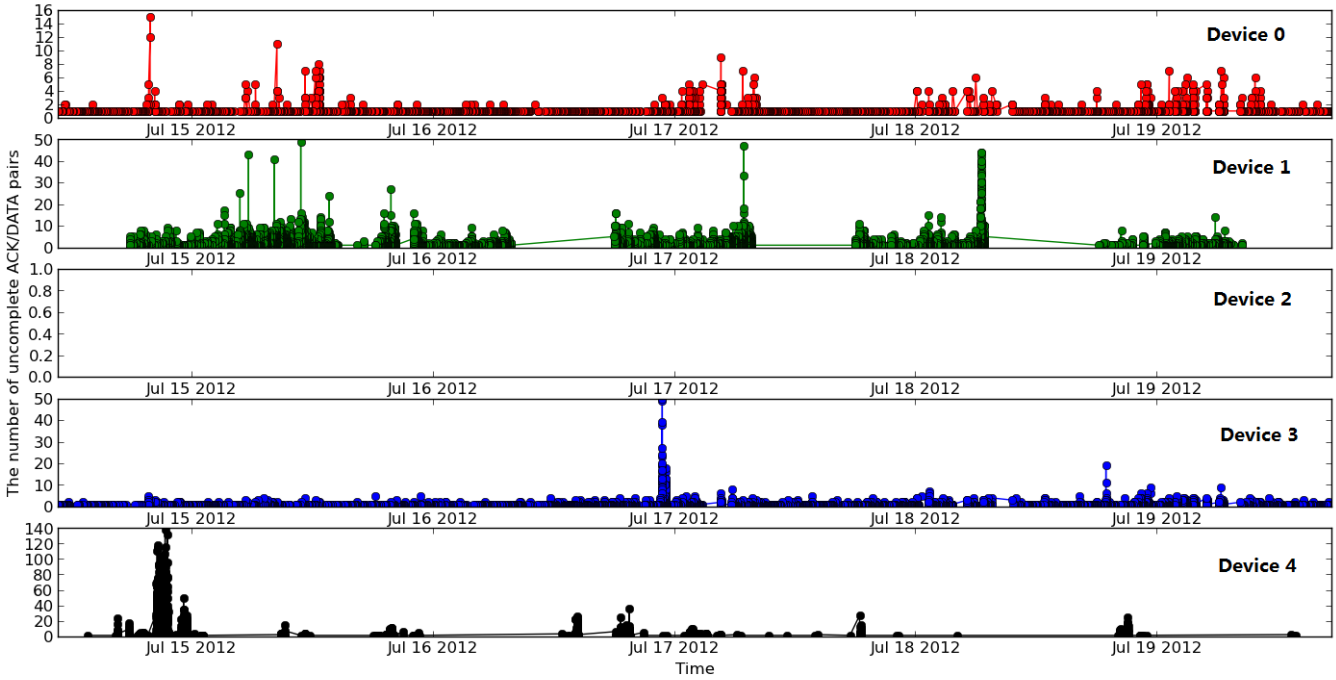


Figure 35: Hidden senders with timeline

significant problems in the real home networks. It may decrease significantly the throughput and performance of the network.

However, the research in this area is still not sufficient. A lot of future work still remains to be done. First of all, our measurements are not sufficient, tests in more real home networks must be deployed to improve our monitoring and analysis methods. Secondly, the solutions for the problems are still not very obvious. One possible work in the future is to create the cooperation between several networks to optimize the channel usage, and avoid hidden terminals. Thirdly, the demo could show us the current status of our network but many user behaviors could be studied by studying longer time data which needs more capacities of storage in the gateway. Fourthly, for automatically troubleshoot, self-learning algorithms must be implemented to record the normal status of the network. Once some metrics in the network stand out, the troubleshooting algorithm will be called to find the problem. At last, a good idea for identifying the problem is to apply the algorithms like K-values to the multi-dimensional metric vectors to find an independent space for each kind of problem and map the standing-out values to the corresponding space.

9. REFERENCES

[1] Ieee std 802.11a-1999.

[2] Ieee std 802.11b-1999.

[3] Ieee std 802.11g(tm)-2003.

[4] Atul Adya, Paramvir Bahl, Ranveer Chandra, and Lili Qiu. Architecture and techniques for diagnosing faults in ieee 802.11 infrastructure networks. In *MobiCom*, 2004.

[5] Alex V. Barbosa, Marcos F. Caetano, and Jacir L. Bordim. The theoretical maximum throughput calculation for the ieee802.11g standard. *International Journal of Computer Science and Network Security*, 11 :4, 2011.

[6] Joshua Bardwell. You believe you understand what you think i said¡the truth about 802.11 signal and noise metrics. Technical report, connect802, 2004.

[7] Pablo Brenner. *A Technical Tutorial on the IEEE 802.11 Protocol*. BREEZECOM Wireless Communication.

[8] Kenneth L. Calvert, W. Keith Edwards, Nick Feamster, Rebecca E. Grinter, Ye Deng, and Xuzi Zhou. Instrumenting home networks. In *SIGCOMM*, 2010.

[9] Ranveer Chandra, Venkata N. Padmanabhan, and Ming Zhang. Wifi profiler : Cooperative diagnosis in wireless lans. In *MobiSys*, 2006.

[10] Yu-Chung Cheng, John Bellardo, Peter Benko, Alex C. Snoeren, Geoffrey M. Voelker, and Stefan Savage. Jigsaw : solving the puzzle of enterprise 802.11 analysis. In *SIGCOMM*, 2006.

[11] Marshini Chetty, Richard Banks, Richard Harper, Tim Regan, Abigail Sellen, Christos Gkantsidis, Thomas Karagiannis, and Peter Key, editors. *Who is Hogging The Bandwidth ? The Consequences Of Revealing The Invisible In The Home*. Association for Computing Machinery, Inc., 2010.

[12] Marshini Chetty, Ja-Young Sung, and Rebecca E. Grinter, editors. *How Smart Homes Learn : The Evolution of the Networked Home and Household*. Proceedings of the 9th international conference on Ubiquitous computing, 2007.

[13] Cisco. The zettabyte era, 2012.

[14] Lucas DiCioccio, Renata Teixeira, and Catherine Rosenberg. Characterizing home networks with

homenet profiler. Technical report, Technicolor and UPMC Sorbonne Universities and CNRS and University of Waterloo, 2011.

[15] Michael Gauthier. *Wireless Networking in the Developing World*. 2007.

[16] Rebecca E. Grinter and W. Keith Edwards. The work to make a home network work. In *ECSCW*, 2008.

[17] Martin Heusse, Franck Rousseau, Gilles Berger-Sabbatel, and Andrzej Duda. Performance anomaly of 802.11b. In *INFOCOM*, 2003.

[18] Mihail Sichitiu Jangeun Jun, Pushkin Peddabachagari, editor. *Theoretical Maximum Throughput of IEEE802.11 and its Applications*. Network Computing and Applications, 2003.

[19] Amit P. Jardosh, Krishna N. Ramachandran, Kevin C.Almeroth, and Elizabeth M. Belding-Royer. Understanding congestion in ieee 802.11b wireless networks. In *IMC*, 2005.

[20] Aruna Jayasuriya, Sylvie Perreau, Arek Dadej, and Steven Gordon. Hidden vs. exposed terminal problem in ad hoc networks. In *ATNAC*, 2004.

[21] Bobby Jose, Hujun Yin, Praveen Mehrotra, and Ed Casas. Mac layer issues and challenges of using smart antennas with 802.11. *Vehicular Technology Conference*, 2003.

[22] Srikanth Kandula, Ratul Mahajan, Patrick Verkaik (UCSD), Sharad Agarwal, Jitendra Padhye, and Paramvir Bahl. Detailed diagnosis in enterprise networks. In *SIGCOMM*, 2009.

[23] Partha Kanuparthy, Constantine Dovrolis, Konstantina Papagiannaki, Srinivasan Seshan, and Peter Steenkiste. Can user-level probing detect and diagnose common home-wlan pathologies ? In *SIGCOMM*, 2012.

[24] Mathieu Lacage, Mohammad Hossein Manshaei, and Thierry Turletti. Ieee 802.11 rate adaptation : a practical approach. In *MSWiM*, 2004.

[25] Kaushik Lakshminarayanan, Srinivasan Seshan, and Peter Steenkiste. Understanding 802.11 performance in heterogeneous environments. In *HomeNets*, 2011.

[26] Konstantina Papagiannaki, Mark Yarvis, and W. Steven Conner. Experimental characterization of home wireless networks and design implications. In *INFOCOM*, 2006.

[27] Erika Shehan Poole, Marshini Chetty, Rebecca E. Grinter, and W. Keith Edwards. More than meets the eye : Transforming the user experience of home network management. In *Proceedings of the 7th ACM conference on Designing interactive systems*, 2008.

[28] Anmol Sheth, Christian Doerr, Dirk Grunwald, Richard Han, and Douglas Sicker. Mojo : A distributed physical layer anomaly detection system for 802.11 wlans. In *Proceedings of the 4th international conference on Mobile systems, applications and services*, 2006.

[29] Eduard Garcia Villegas, Elena Lopez-Aguilera, Rafael Vidal, and Josep Paradells. Effect of adjacent-channel interference in ieee 802.11 wlans. In *CrownCom*, 2007.

[30] Shugong Xu and Tarek Saadawi. Revealing the problems with 802.11 medium access control protocol in multi-hop wireless ad hoc networks. *Computer Networks*, 2001.

[31] Jeonghwa Yang. *EDEN : AN INTERACTIVE HOME NETWORK MANAGEMENT SYSTEM*. PhD thesis, College of Computing, Georgia Institute of Technolog, 2009.

[32] Frank Y.Li, Arild Kristensen, and Paal Engelstad. Hidden terminal detection in 802.11-based wireless ad hoc networks. *IST*, 2006.

Characterizing end-host application performance across multiple networking environments

Diana Joumblatt*, Oana Goga*, Renata Teixeira*, Jaideep Chandrashekar† and Nina Taft†
*CNRS and UPMC Sorbonne Universites
†Technicolor Research

***Abstract*—Users today connect to the Internet everywhere - from home, work, airports, friend's homes, and more. This paper characterizes how the performance of networked applications varies across networking environments. Using data from a few dozen end-hosts, we compare the distributions of RTTs and download rates across pairs of environments. We illustrate that for most users the performance difference is statistically significant. We contrast the influence of the application mix and environmental factors on these performance differences.**

I. Introduction

Users connect to the Internet via their laptops or notebooks (which we generically refer to as 'end-host') in a number of different contexts or networking environments, such as at home, work, coffee shops, or airports. The network performance of one single end-host can potentially vary across different networking environments. The goal of this paper is twofold. First, we quantify and characterize the multiple networking environments that users employ. Second we seek to understand if the performance of the end-host varies significantly in different environments. These steps are critical to subsequent tasks such as application performance diagnosis and network performance management.

We carry out our study using data, from dozens on end-hosts, that was collected via the HostView end-host monitoring tool [1]. HostView logs network packet traces as well as application and location information. Given that users ran HostView on their end-hosts for weeks or months, HostView was able to witness use of multiple environments for individuals; hence this unusual dataset with an end-host perspective is well suited for our goals.

Using this data from an admittedly small set of end-hosts, enables us to explore the following questions. *First*, how many environments does a single user employ and what are the different characteristics of these environments? (Sec. III) Environment factors such as source ISP, network interface, country, and others, define different networking environments. Overall, we observe a fair amount of diversity in the number and types of environments individuals use (e.g., 75% of users connect to multiple environments), as well as in the application mix across different environments. *Second*, does network performance vary across environments? (Sec. IV). We compare the performance in pairs of environments using two metrics, the distributions of round-trip times (RTTs) and download rates, in each environment. We use the Hellinger distance [2] to identify statistically significant differences. We also find that the application mix has a stronger influencer on data rates than environment factors, whereas the reverse is true for round trip time behaviors.

II. End-host data

The data used in this paper was collected directly on end-hosts using the HostView tool [1], [3]. We briefly describe the data collected by HostView, how we define a network environment, and the metrics of network performance that we extract from this dataset. For a longer description of HostView, please refer to our previous work [1].

A. HostView tool and data

HostView runs on MacOS and Linux and logs network traffic, application context, and information about the network the end-host is connected to. Then, it uploads the traces to a central repository every four hours.

Network traces: Packet traces are collected with the libpcap library. HostView collects the first 100 bytes of every packet (the first 96 bytes are usually header); for DNS packets, it stores the entire packet so as to enable recreating the hostname to IP address mappings offline. It also parses HTTP header to extract the HTTP content type (common content types are text, image, or video).

Application context: We complement packet traces with the application responsible for each flow. We define a *network flow* as a five-tuple of source and destination IP, source and destination port, and protocol; a *connection* refers to two network flows in opposite directions. By *application*, we mean any entity that is communicating on the Internet. In some cases, the application is interchangeable with the process executable: e.g., `Skype`. We collect process executable information with the `gt` tool [4]. In other cases, however, applications are delivered as web services. If a user spends time interacting with `facebook.com`, this is not captured by simply using the name of the browser executable (`Firefox`). To deal with this subtlety, we resort to the following rule: if the process executable is not a web browser, the application is simply the same as the process executable (e.g.,`iTunes, Skype, Mail.app`); otherwise, the application is the top-level domain name of the destination (e.g., `facebook.com, google.com, yahoo.com`). This definition allows for a better accounting of a user's online activity, but it will also consider third-party sites as an application (for instance, `akamai.net` is one of the top applications in our data).

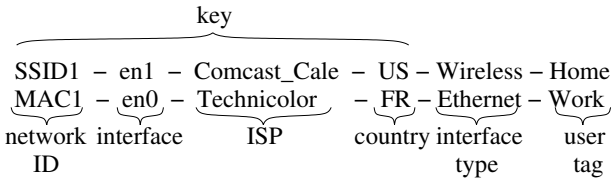


Figure 1. Example of environments

Although these third-party services are not an application directly initiated by users, they do generate traffic and can influence the network performance in a given environment.

Location and machine context: HostView generates a new trace file either when it detects changes in the network interface or the IP address, or if four hours have elapsed. Every trace file is annotated with a label describing the environment it was collected in: we record the specific network interface, a hash of the wireless network SSID and of the BSSID (when the interface is wireless), or the MAC of the first network device (when the interface is wired). We also record the country, city, and ISP to which the user is connected. We obtain this information at the collection server by mapping the public IP of the host (or the public IP of the router in case the host is behind a NAT) using the MaxMind GeoIP commercial database from March 2011. In addition to these automatically-generated environment descriptors, whenever HostView observes a new SSID, it asks users to label this SSID with one of the following tags: home, work, airport, hotel, conference meeting, friend's home, public place, coffee shop or other. We call this label the *user tag*.

The data used in this paper was collected between November 2010 and February 2011 from 40 users, who ran HostView for at least two weeks. We have 22 users from Europe, 12 from the United States, 2 from Asia, 2 from Australia, 1 from Africa and 1 from Brazil. Due to the nature of our deployment (where we recruit users to install the tool on their systems), there is a lot of variation on how long each user ran the the tool (from two weeks to three months).

We recruited users mainly through advertisements in computer science mailing lists and conferences. Even though our user population is mainly of computer scientists and admittedly small, we see a great deal of diversity in application usage and network environments as shown in Sec. III. Moreover, a minimum of two weeks of data from each end-host ensures that we have a large number of network-performance samples taken at different environments.We believe that the results discussed in this paper have useful lessons in understanding network performance as seen from end-hosts "in the wild".

B. Definition of environment

We describe an environment with six features as illustrated in Fig. 1. The first tag captures the *network identifier*, which encodes the MAC (*i.e.* the MAC of the first network device) if the trace was collected when the wired interface was active and the hashed SSID when on wireless; the second tag encodes the name of *network interface* used; the third tag encodes the name of the *local ISP* (or the ISP the user connects to); the forth tag reflects the *local country*; the fifth tag captures the *type of the interface* (wireless or wired); and the last tag is the *user tag*.

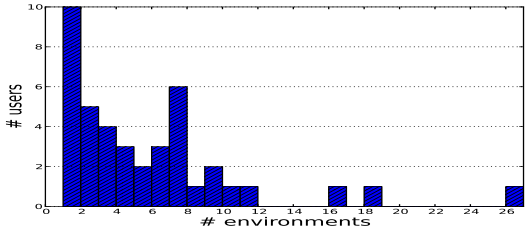


Figure 2. Histogram of the number of environments per user

We use a four-tuple composed of the network ID, the network interface, the local ISP, and the local country to identify an environment. In most cases, a pair with the network ID and the network interface is a good identifier for an environment, but there are some exceptions. First, an SSID can be identical across different locations (*e.g.* coffee shops or hotel chains). Although the BSSID could disambiguate the environments in this case, we do not include it to define an environment because it would also artificially split some unique environments (for instance, when an enterprise or university deploys multiple access points to implement a single network). Second, we observe one user who always use the same device to access the Internet from many different places (*i.e.* we observe a single MAC address for traces uploaded from different ISPs and countries). Adding the local ISP and country ensures that environments in our dataset are uniquely identified.

III. Environments and applications

The first characteristic we looked into was how many environments an individual uses. Fig. 2 shows the histogram of the number of distinct environments per HostView user. We see, that among our users, 25% only use a single environment, while 50% use 5 or more. We even observe a number of individuals with very many environments, 12% use 10 or more and one individual actually used 26 environments. Since our goal is to contrast network performance across environments per end-host, we exclude those users, only connecting via one environment, from the remaining analysis.

Next we examined how much time users spend in each of their environments, and the proportion of traffic generated per environment. Our analysis shows that the fraction of time spent per environment varies significantly from one user to another (plots not shown for conciseness), but nevertheless we observed some general trends. First, users have a small set of dominant environments: 24 users spend 80% of their time in less than three environments. Often a user's "most-used" environment accounts for more than 50% of their time. Second, there is a strong correlation between the time spent per environment and the number of bytes sent and received in that environment (Pearson's coefficient is above 0.9 for 90% of the users).

We now investigate where the diversity in environments come from; in other words, is this behavior due to users

Table I
EXAMPLE APPLICATIONS

Multiple environments	Single environment
facebook.com	rhythmbox
twitter.com	iTunes
akamai.net	hulu.com
google.com	netflix.com
Skype	VLC
Mail	speedtest.net
wordpress.com	uTorrent
ssh	openvpn

employing multiple interfaces or protocols, or due to high mobility such as lot of travel, etc. This can be useful for understanding diversity in network performance; for example, knowing that a user always uses the same network interface, but use numerous source ISPs (or vice versa) might help explain performance differences a user experiences across environments. We found that most of our users connect from only one country, yet up to 75% of the users employee between 2 and 4 ISPs regularly. We do have a few users that connected to 3 or more countries. Most individuals connected to the Internet with ethernet and wireless, but a handful also use ppp, bluetooth, and phone-tethered. Overall, although for roughly 10% of our users, their environment diversity comes from their travel, the main source of diversity for most users is their use of multiple ISPs, and the pairing of a given ISP with either ethernet or wireless.

Another key component influencing the performance in a given environment is the mix of applications used within that environment. Thus we next examine the set of applications used across environments. We use the term *single-environment app* to refer to applications that are only used in one environment, and similarly we use *multi-environment app* to refer to applications used in multiple environments. Table I lists examples of single-environment and multi-environment applications. The applications included are those that are popular across many users, or else frequently used by some individuals. Many of the popular applications (such as facebook.com and google.com) appear in multiple environments, as expected. There are intuitive hypotheses as to why some of the single-environment applications only occur in one environment. For example, video-on-demand and TV applications occur in environments users tag as 'home'; users typically only need openvpn when they are accessing their work network from outside; and speedtest.net is an application users run mainly when they are experiencing problems (which may only occur regularly in one of their environments). Overall, we found that the majority of users (26 out of 30) employ at least 50% of their applications in a single environment. In addition to the reasons cited above (why some applications make sense in a single environment), we note that a number of applications are only used once (such as a web service).

IV. PERFORMANCE ACROSS ENVIRONMENTS

It is interesting to ask whether the same application, or the same set of applications, appearing in two different environments experiences the same or different performance in each environment. We contrast the performance of a given end-host in two environments by examining both the RTTs and download data rates in each environment. More specifically, we compare the distribution of RTTs in one environment with their distribution in the second environment, and then do the same for download rates. These two metrics capture important aspects of application performance, including interactive applications (i.e. that need low delay) and bandwidth hungry applications (i.e. that need high data rates). We do this for many pairs of environments using the following methodology.

A. Methodology

We extract RTTs and data rates from network traces using the `tcptrace` tool. An RTT sample is the time elapsed between the data packet and its corresponding acknowledgment (only for TCP connections). `tcptrace` does not compute RTTs for retransmitted packets or for delayed or reordered acknowledgments. In our analysis, *RTT* refers to the average value of RTT samples of a TCP connection over a second. *Download data rate* is computed as the total number of unique bytes received by the end-host in one second (i.e., data bytes received excluding retransmitted bytes). We modify `tcptrace` to generate the data rate per connection in one second bins instead of an average goodput every ten packets or an instantaneous goodput. This yields two time series per network flow, one for RTTs and the other for download rates.

We want to compare the distribution of RTTs in two environments i and j. Let $f_i^{RTT-all}(x)$ denote the empirical probability distribution that an RTT will take value x in environment i. The superscript *RTT-all* indicates that the set of RTTs considered are those coming from all applications. Hence our task is to compare the two distributions $f_i^{RTT-all}(x)$ and $f_j^{RTT-all}(x)$. Similarly we also seek to compare download rates in 2 environments (i.e. $f_i^{down-all}(x)$ and $f_j^{down-all}(x)$).

To ensure sufficient statistics, we only compare distributions for a given metric if both environments have at least 5000 samples points of the given metric. Hence, this section uses 21 rather than all 30 users, because some users had insufficient data from some environments. If one user has E environments, then we can perform $E(E-1)/2$ pairwise comparisons for that user. In total, the number of pairs of environments is 164 for download rates and 112 pairs for RTTs. The reason that we have different numbers of pairs of environments for each metric is because in a single environment we can have an unequal number of samples for download rates and RTTs, depending on what the user is doing.

The statistics literature offers a number of metrics that can be used to compare two distributions, such as the Kolmogorov–Smirnov (KS), Kullback–Leibler (KL) divergence, and the Hellinger distance. These are typically used as part of a hypothesis test to determine whether or not two distributions are similar. We decided not to use a KS test because it returns the maximum vertical distance between two distributions; this is not suitable for our data since we can have gaps in some ranges of the performance metric for an environment. (This

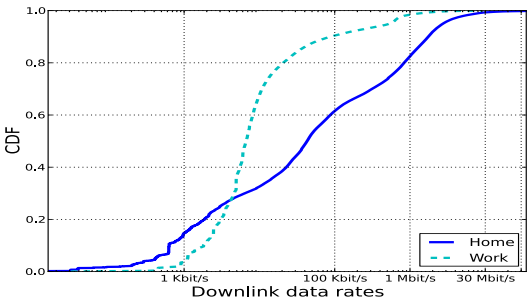


Figure 3. Download rates for two environments: $HD_{ij}^{down} = 0.38$.

occurs since some environments are more used than others and because the mix of applications can be environment dependent.) Moreover the KS test assumes the underlying distributions are continuous. We also decided not to use the KL distance since it is asymmetric (i.e., the results between i and j are different than j and i) and because it is unbounded thus making it harder to interpret. Instead we elected to use the a discrete version of the *Hellinger distance (HD)* [2] that captures the area between two distributions and the difference in shape of two probability mass functions. The HD is computed from two empirical densities $p(x)$ and $q(x)$ as follows:

$$HD(p,q) = \sqrt{1 - \sum_{x \in X} \sqrt{p(x)q(x)}}. \quad (1)$$

When we apply this to RTT distributions in environments i and j, we have $p(x) = f_i^{RTT-all}(x)$ and $q(x) = f_j^{RTT-all}(x)$. Let $HD_{ij}^{RTT-all}$ denote the Hellinger distance between $f_i^{RTT-all}(x)$ and $f_j^{RTT-all}(x)$. The HD is symmetric and bounded in [0,1] where $HD = 0$ means the distributions are the same and $HD = 1$ is the maximum divergence between two distributions.

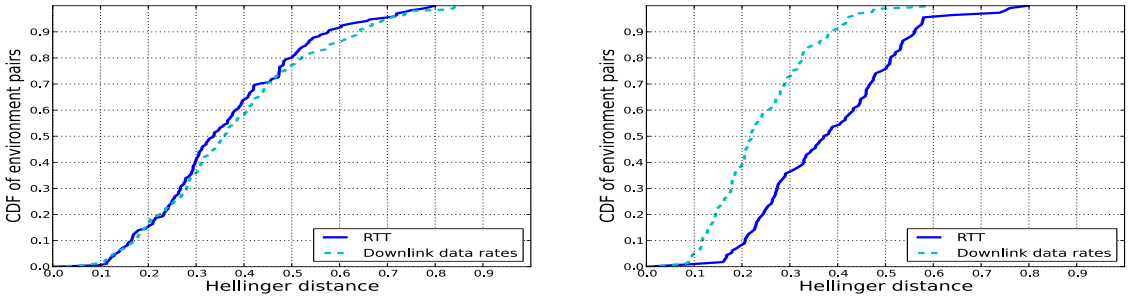
It is always challenging to select a threshold for rejecting the null hypothesis that two distributions are similar. We followed a classical bootstrapping procedure [5] to identify a threshold that would correspond to a P-value of 0.05 and 0.1 (typical P-values for rejecting the null hypothesis). Our bootstrapping procedure revealed that comparisons across data partitions coming from the same distribution have HD values less than 0.05. However we found that using this procedure we almost always rejected the hypothesis and thus this isn't useful for our task at hand. (As is well known, statistics is mainly an art form.) We seek to understand when the performance across environments is different and our subsequent work is based on these cases. In order not to exaggerate, we are conservative in our reporting if we underestimate the number of environments that are deemed different. Hence we slightly increase the threshold value used (0.1) to decide if two environments are deemed different. Thus if $HD_{ij} > 0.1$ we consider f_i and f_j different. We performed visual inspection of hundreds of pairs of histograms and found that with a threshold of 0.3, the two distributions were clearly vastly different. In these cases the distributions are "significantly" different because either the mass of one distribution is largely shifted or the shape of the distribution is completely different. We provide a single illustrative example in Fig. 3. We thus identify three ranges to quantitatively describe the difference between two histograms f_i and f_j (we omit the superscript when the context is clear). If $HD_{ij} \leq 0.1$ we consider f_i and f_j similar; when $0.1 < HD_{i,j} \leq 0.3$, then f_i and f_j are different, and if $HD_{i,j} > 0.3$, then f_i and f_j are significantly different. Although 0.3 is a heuristic, we consider it safe because it is conservative based upon our bootstrapping experiments.

B. Results

We computed $HD_{ij}^{RTT-all}$ and $HD_{ij}^{down-all}$ for all pairs i, j for all users. We plot the cumulative distribution of all these values (with one curve per performance metric) in Fig. 4(a). We see that approximately 60% of all environment pairs have a Hellinger distance greater than 0.3 for both RTTs and data rates. This result means that in most cases the distribution of delays and data rates that a host experiences differs significantly across environments. These large differences happen for the vast majority of users: 17 out of 21 users have at least one pair of environments with $HD_{ij}^{down-all} > 0.3$; this number is 20 out of 21 for RTTs.

The RTT and download data in Fig. 4(a) comes from *all* the applications in a given environment, however we observed in Sec. 3 that the mix of applications across environments often differs as there are a number of single-environment applications. The different application mix would explain at least some of the performance differences across environments. In order to understand whether or not performance differences are dominated by environment factors (i.e., the ISP, network interface, country, etc.) rather than the application mix, we extract the set of *common applications* for all environment pairs. We can then contrast the performance of a pair of environments using performance data (RTTs and download rates) generated only by these common applications. The HDs for RTTs in a pair of environments that includes only the RTTs generated by the common applications are denoted by $HD_{ij}^{RTT-com}$. (Similarly we compute $HD_{ij}^{down-com}$.)

These modified HD scores are shown in Fig. 4(b). We see that the difference between environments is less pronounced for download data rates when considering only common applications. For example we saw that 64% of environment pairs differed significantly (Fig. 4(a)) when considering all applications, whereas we only 27% of environment pairs exhibit significant difference when considering only common applications. Since the only difference between the experiment in Fig. 4(a) and that in Fig. 4(b) is the inclusion/exclusion (respectively) of single-environment apps, these graphs suggest that the application mix (including the single-environment apps) has a stronger influence on the data rates than the environmental factors. However the results are different for RTT behavior. In comparing these two experiments, we find that 64% of environment pairs differ significantly when all applications are considered, and similarly 63% of environment pairs differ significantly when considering common applica-



(a) All apps (b) Common apps

Figure 4. Distribution of Hellinger distance scores between pairs of network environments.

tions. Thus the application mix is less influential in explaining the significant difference in RTT behaviors across pairs of environments. It thus appears that the environmental factors have a stronger influence on delays than the application mix.

V. Related Work

We focus our discussion on studies based on passive measurements of network traffic and structure it according to the measurement vantage point.

In-network measurements. Zhang et al. [6] developed a tool T-RAT to breakdown the factors (*e.g.* congestion, receiver/sender window, bandwidth, short transfers) that limit the data rates achieved by individual TCP connections. A more recent analysis of network traces collected in an ISP network found that TCP data rates are often limited by the application itself, and not the network [7]. Our analysis of data collected on end-hosts confirms that applications often limit achieved data rates, but we also identify a considerable number of instances when the environment limits data rates.

End-host measurements. Before HostView [1], there have been few efforts to collect data on end-hosts [8]–[10]. A characterization study of enterprise traces [11] analyzed the lifetime of environments (where environment is defined as inside and outside the enterprise) and some network behavior (*e.g* number of TCP/UDP connections). This study does not analyze the performance metrics we study here and how they vary across environments. Our initial analysis of HostView data studied seven performance metrics only on few instances when users report that performance is poor [12]. Here, we focus on two performance metrics, but we perform a longitudinal study of how these metrics vary across environments and applications.

VI. Conclusion

In this paper we look at the characteristics and performance of the numerous environments users employ to connect to the Internet. Factors such as the network interface, source ISP, country and user tags differentiate particular environments. We found that users connect to the Internet via many environments (with many people using 4 to 10, and some even higher). We then examined how groups of applications perform, in terms of delay and data rates, in pairs of environments. We observed that the end-host as a whole (including all applications) typically experiences statistically significant performance differences in two environments employed a single user. Based on our initial experiments, it appears that the application mix has a stronger influencer on data rates than environmental factors, whereas the reverse is true for round trip time behaviors.

Acknowledgements. This work was supported by the European Community's Seventh Framework Programme (FP7/2007-2013) grant no. 258378 (FIGARO) and the Agence National de la Recherche grant C'MON.

References

[1] D. Joumblatt, R. Teixeira, J. Chandrashekar, and N. Taft, "Hostview: Annotating end-host performance measurements with user feedback," in *Hotmetrics*, 2010.

[2] G. L. Yang and L. M. L. Cam, *Asymptotics in Statistics: Some Basic Concepts*. Berlin: Springer, 2000.

[3] D. Joumblatt. http://cmon.lip6.fr/EMD.

[4] F. Gringoli, L. Salgarelli, M. Dusi, N. Cascarano, F. Risso, and K. Claffy, "Gt: picking up the truth from the ground for internet traffic," in *ACM SIGCOMM CCR*, 2009.

[5] M. M. B. Tariq, A. Dhamdhere, C. Dovrolis, and M. Ammar, "Poisson versus periodic path probing (or, does pasta matter," in *In Proc. of Internet Measurements Conference*, 2005.

[6] Y. Zhang, L. Breslau, V. Paxson, and S. Shenker, "On the characteristics and origins of internet flow rates," in *In ACM SIGCOMM*, pp. 309–322, 2002.

[7] M. Siekkinen, G. Urvoy-keller, E. W. Biersack, and D. Collange, "A root cause analysis toolkit for tcp," *Computer Networks and Isdn Systems*, vol. 52, pp. 1846–1858, 2008.

[8] C. R. Simpson, Jr, D. Reddy, and G. F. Riley, "Empirical models of end-user network behavior from neti@home data analysis," *Simulation*, vol. 84, pp. 557–571, October 2008.

[9] E. Cooke, R. Mortier, A. Donnelly, P. Barham, and R. Isaacs, "Reclaiming Network-wide Visibility Using Ubiquitous Endsystem Monitors," in *Usenix Technical Conference*, 2006.

[10] S. Guha, J. Chandrashekar, N. Taft, and D. Papagiannaki, "How Healthy are Todayís Enterprise Networks?," in *Proc. f the Internet Measurement Conference*, October 2008.

[11] F. Giroire, J. Chandrashekar, G. Iannaccone, K. Papagiannaki, E. Schooler, and N. Taft, "The cubicle vs. the coffee shop: Behavioral modes in enterprise end-users," *Passive and Active Measurement (PAM) Conference*, 2008.

[12] D. Joumblatt, R. Teixeira, J. Chandrashekar, and N. Taft, "Performance of Networked Applications: The Challenges in Capturing the User's Perception," in *ACM SIGCOMM Workshop on Measurements Up the Stack (W-MUST)*, August 2011.

Performance of Networked Applications: The Challenges in Capturing the User's Perception

Diana Joumblatt, Renata Teixeira
CNRS and UPMC Sorbonne Universites
{diana.joumblatt,renata.teixeira}@lip6.fr

Jaideep Chandrashekar, Nina Taft
Technicolor
{jaideep.chandrashekar,nina.taft}@technicolor.com

ABSTRACT

There is much interest recently in doing automated performance diagnosis on user laptops or desktops. One interesting aspect of performance diagnosis that has received little attention is the user perspective on performance. To conduct research on both end-host performance diagnosis and user perception of network and application performance, we designed an end-host data collection tool, called HostView. HostView not only collects network, application and machine level data, but also gathers feedback directly from users. User feedback is obtained via two mechanisms, a system-triggered questionnaire and a user-triggered feedback form, that for example asks users to rate the performance of their network and applications. In this paper, we describe our experience with the first deployment of HostView. Using data from 40 users, we illustrate the diversity of our users, articulate the challenges in this line of research, and report on initial findings in correlating user data to system-level data.

Categories and Subject Descriptors

C.2.3 [Network Operations]: Network monitoring

General Terms

Human Factors, Measurement, Performance

Keywords

End-host measurement, Network performance diagnosis, User experience

1. INTRODUCTION

Slow web downloads, choppy Skype calls and YouTube videos, or broken access to your email server are just few examples of how Internet disruptions can frustrate the user experience. A number of research projects aim to help users with tools that automatically diagnose performance problems. Some of this research has focused on enterprise networks [1,13] and ISP networks supporting popular applications such as VoIP [18] or IPTV [16]. Other efforts diagnose performance and configuration problems from an end-host upon the user request [15].

The ultimate goal of such tools is to improve the user experience, yet such efforts rarely engage the user in the process in order to understand their perception of how the network is performing. Studies of user quality of experience have been limited to a few niche applications, for instance VoIP [9], online-gaming [2], and video playback [3,19]. We believe it is important to understand the user's perspective in assessing performance degradation. Clearly, a user's perspective of what constitutes a problem is a subjective matter. Such a perspective could potentially be used to decide whether or not to launch a diagnosis process. For example, if a user's network experiences a problem, but the application masks it and the result doesn't affect the user, it may not necessary to launch a diagnosis process.

The critical first step towards automated performance diagnosis and repair of end-hosts is to gather data directly from laptops or desktops and annotate that data with the user's perspective. Although some end-host datasets exist [6,8,17], none of them is annotated with the user perceived quality of the performance of the network and applications. The only exception is HomeMaestro [14], which instrumented the gateway in a small set of homes to capture all packets and asked home users to keep logs of how they perceive the network performance. Because it collects traces for a home and not per end-host and the logs are not automated, they cannot have fine-grained correlation of each user's perception of network performance.

Towards this goal we designed and built HostView[1] [11], an end-host tracing tool that runs on Mac OS and Linux PCs. HostView collects network traffic, system performance information, and the application associated with network traffic. Importantly, it also prompts the user for feedback on network performance. HostView incorporates two mechanisms for getting user feedback: a system-triggered questionnaire based upon the Experience Sampling Methodology [5], and a user-triggered mechanism called an "I'm annoyed!" button. The ESM mechanism prompts the user no more than three times per day to respond to a questionnaire about their network experience in the five minutes preceding the questionnaire. The "I'm annoyed!" button is always displayed at the edge of the screen and we ask users to click on this button when they are not happy with their network performance. Clicking the button brings up the same feedback questionnaire as in the ESM prompts. The particular questions in the questionnaire can be seen in [12]. For more details on the design of HostView, please refer to [11].

In this paper, we report on preliminary findings from our first deployment with HostView. First, we find that there is a great deal

[1] http://cmon.lip6.fr/EMD

W-MUST'11, August 19, 2011, Toronto, Ontario, Canada.

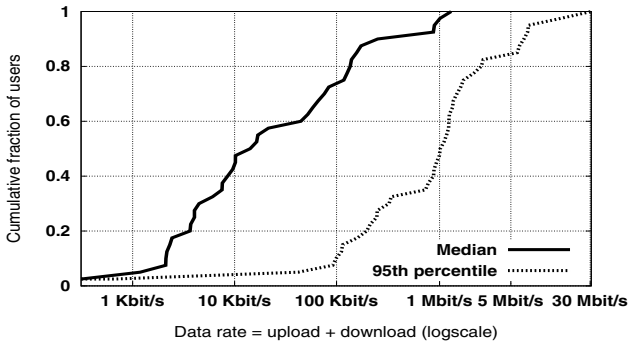


Figure 1: Median and 95^{th} percentile of data rates.

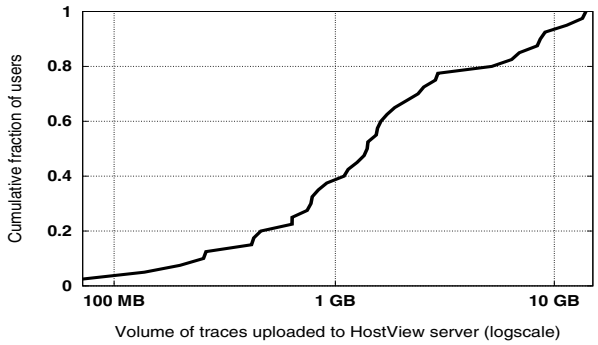


Figure 2: Trace size for the first two weeks of collection.

of diversity in the installed user base – in network speeds, application usage, and the level of network use (§ 2). While expected, the extent of this diversity took us by surprise and necessitated several modifications to Hostview after it was released (to support particular users). This experience leads us to believe that monitoring tools must be designed so as to adapt to particular individuals. Second, collecting feedback from users, particularly about poor network performance, is fundamentally challenging because it is hard to collect sufficient statistics about rare events. By allowing two different mechanisms to register their feedback, our tool is able to collect a reasonable number of reports (§ 3). We were able to collect 576 total reports from users in a 2 week period, 95 of which were deemed instances of poor network performance. Finally, we find that a small set (seven, to be exact) of relatively simple performance metrics are able to uncover problems in a vast majority of cases where user's report performance problems. In 82 out of the 95 poor performance instance, at least one of the metrics also exhibits "anomalous" behavior (§ 4).

2. USER CHARACTERISTICS

Our first deployment of HostView was between November 2010 and February 2011. The data in this paper comes from 40 users (26 on Mac OS and 14 Linux users). Although many users ran HostView for over a month, this paper analyzes data from the first two weeks of each of these users for the purposes of consistency. We gathered these users both via publicity and via incentives. We publicized HostView by distributing leaflets during the ACM Internet Measurement Conference 2010 and by sending emails to a number of CS mailing lists in November 2010. We offered two types of incentives for users to install HostView. First, HostView reports "network usage" summaries, where users can see their throughput as well the fractional bandwidth of active applications. Second, we offered 50 USD Amazon gift cards to a set of 40 users randomly picked from the first 100 who run HostView for an entire month.

Given that we mainly recruited users from the networking community and that HostView runs only for Mac OS and Linux, we expect that most of our users are computer experts. Despite this bias and the small number of users, our users exhibit a great deal of diversity in many ways. First, our users come from 14 different countries in Europe, North and South America, Asia-Pacific, and Middle-East. Second, the total number of different applications employed by these users over two weeks was 385. Table 1 presents the top-ten applications in our traces in terms of the number of users and the number of bytes transferred. Application names are identified in HostView from the system process names using the *gt* toolkit [7], so traffic to web-based applications such as Facebook or YouTube will appear under the browser name. We see that a number of our popular applications, such as web browsers, e-mail, and Skype, are similar to popularity rankings by the broader community captured in the the wakoopa ranking[2]. It is interesting to contrast the most popular applications with the applications that transfer the most bytes. The web browsers and Skype appear in both lists. Given that Bittorent transfers files, videos and other (often large) content, it isn't surprising that it tops the list in terms of number of bytes. The fact that SSH appears in the top-ten reflects the working habits of computer scientists.

Ranking by users	Ranking by bytes
1. Firefox	1. uTorrent
2. Skype	2. Safari
3. Safari	3. SSH
4. Mail	4. Firefox
5. Google Chrome	5. Google Chrome
6. Dropbox	9. smbd
7. iTunes	6. Skype
8. Thunderbird	7. Mail
9. Adium	8. VLC media player
10. VirtualBox	10. Chromium

Table 1: Top-ten applications.

Third, we observe behavioral diversity, even with a bias towards computer scientists. For example, network usage varies considerably across the set of studied users. We compute data rates per user as the number of bits downloaded plus the bits uploaded every second. A user's data rate reflects both the capacity of her access network and the amount of data exchanged by applications. If a user is connected to a 100 Mbits/s Ethernet but only reading text email, the data rate will be low. Inversely, if a user is downloading movies over a 512 Kbits/s DSL line, the download can only go as fast as the DSL link. Fig. 1 presents the cumulative distribution of the median and 95^{th} percentile of data rates per user (computed over the two week period) in log scale. This figure shows that both statistics vary by orders of magnitude across users. There is also a significant difference between the median and the 95^{th} percentile data rates, which indicates that individual users generate very different data rates over time.

This difference in network usage also reflects on the volume of trace data collected for each user. Fig. 2 shows the cumulative distribution of the HostView trace size (in bytes) collected during the first two weeks for each of the 40 users (x-axis is in log scale). This figure shows that trace sizes vary from tens of megabytes to over

[2]Wakoopa is a social networking site that tracks the applications used by its members: http://social.wakoopa.com/software

ten gigabytes. The trace size depends not only on the data rates, but also on other factors such as the amount of time a user spends online, the applications used, and a person's use of HostView's *pause* feature. The offered pause function allows users to temporarily halt data collection. Our analysis of HostView logs shows that 26 users never paused the data collection, other users paused it only few times (at most 4 times during the two week period), except for one user who paused it 13 times. The users with the smallest traces never used the pause button; they simply spent little time online. In summary, we observe user diversity over orders of magnitude in terms of data rates and volume of network data generated; and we observe some outlier user behavior in terms of their use of the pause feature.

3. CHALLENGE: USER DIVERSITY

Although the diversity in user behavior is attractive for gathering a broad set of data, and for subsequent analysis, this same diversity leads to challenges in this line of research, as discussed in this section.

3.1 Monitoring tool design

There are often many ways to carry out the software design of a particular component in the monitoring tool itself. The "best" design may in fact depend upon the configuration and usage patterns of a user's computer. We illustrate this point with one example and discuss the implications for monitoring tool design.

A core component of an end-host monitoring tool is the ability to upload the collected data from an individual machine to a back-end server for later processing. This should be done in a way that does not impact the PC performance nor annoy the user. One approach is to upload the most recent trace all at once, per each data transfer attempt (we try every 4 hours). This way the task completes quickly and removes potentially large trace files off a user's machine in one step. We tried this approach initially and found that although it worked for the majority of users, there were a few who reported problems because uploading traces was consuming too much bandwidth and impacting machine performance. We learned that these users in fact suffered (with some regularity) from very low bandwidth rates, illustrating that one cannot make assumptions even about minimal upload quality.

We redesigned this mechanism following a second approach that also attempts to upload every four hours, but that introduces two features to avoid consuming too much bandwidth. First, it caps the upload bandwidth at 100 Kbits/s. Second, it sends traces split in 2 Mbytes chunks. After sending each chunk, if the chunk is larger than 150 Kbytes and the achieved throughput is less than 10 Kbits/s, then HostView aborts uploading the trace and tries again after four hours. This check ensures that if a user is at a place with bad network connectivity, HostView will not exacerbate the problem. Four of our users are running the new upload algorithm and no uploading problems have since been reported.

We conclude that the diversity in trace sizes and network connectivity of an individual's machine implies that a "one-size-fits-all" approach isn't necessarily the best design paradigm for end-host tracing tools. Moreover it seems a shame to have to design a tool for the worst case scenario, i.e. the few users with problems. We have come to believe that developing monitoring software whose configurations are *adaptive* is the right design paradigm for the future development of end-host tracing tools. For example, a tool could be adaptive in that it first observes the user's behavior or the machine performance and then selects among a few options for uploading the trace data.

ESM			I'm annoyed!
Completed	Deferred	Not Answered	
541	311	448	35

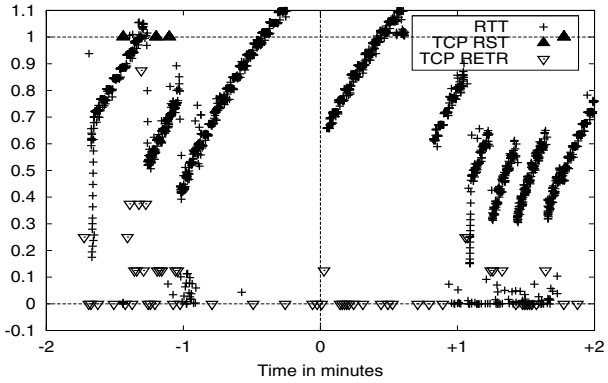
Table 2: User feedback (1335 questionnaires)

3.2 Sampling users for feedback

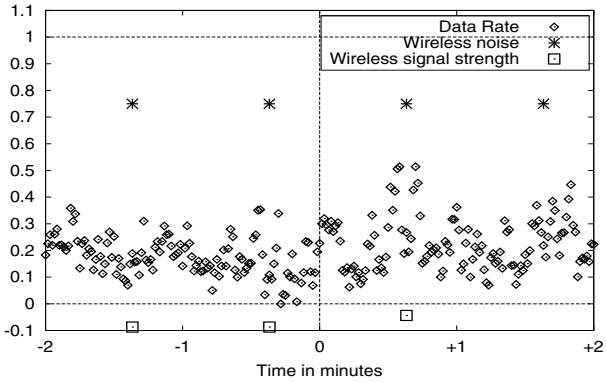
The study of network disruptions as perceived by end-users presents fundamental challenges in getting sufficient data for the task at hand. The challenge is in figuring out *at what time* and *under what conditions* to sample a user. By "sample a user" we mean the act of popping up a questionnaire and getting the user to complete it. We identify five issues in gathering both machine and user data during the very moment of a performance problem. First, poor performance events are by definition supposed to not be the norm, and thus depending upon the user's machine and network, such events could be far and few between. Second, in order to avoid user annoyance, the questionnaire should pop up a limited number of times per day. We selected to use 3 based upon the results of our 400-person survey [10]. Third, the questionnaires need to be spaced somewhat far apart - again to minimize user annoyance. This is complicated by the fact that predicting user uptime is hard [11] which makes the task of deciding when to next pop up a questionnaire difficult. On days when the user doesn't use her PC much, fewer than 3 samples could be collected. Fourth, promising the user that they will be asked to fill out the questionnaire less than a small finite number (e.g. 3 times a day), means that once they are all filled out, another questionnaire cannot be issued even if a poor performance epoch occurs. Fifth, there is a large diversity of user willingness [10] and we suspect that there will also be a wide range in the consistency within users fill-in the questionnaires.

All of this means that popping up questionnaires randomly, when limited to 3 a day and multiple hours in between questionnaires, isn't going to generate many surveys at poor performance epochs. To improve the chances we designed HostView (described in [11]) to pop up the questionnaires using a scheme that weighs bad performance moments more heavily when deciding when to pop-up the questionnaire. Remember, that our "I'm annoyed!" button also increases the chances of getting feedback since the user can elect to engage it at any time.

How did our users respond to the two mechanisms for gathering data? Table 2 presents a summary of the responses we gathered. The numbers in the first 3 columns, under the ESM heading, refer to system-triggered questionnaires. We received 541 completed surveys that came from 35 users, indicating that most of our users (88%) used the ESM mechanism regularly. This also implies that 5 of our users never responded to the system-triggered questionnaire. Unfortunately we found that 448 questionnaires were never answered; this count refers to questionnaires that popped up and either were closed by the user or timed out after 6 minutes because the user ignored it. The 311 *Deferred* questionnaires refers to the number of times a user hit the "Ask Me Later" button when the survey popped up. Of the deferred questionnaires, we found that roughly 23% were completed the following time the questionnaire popped up. We found that 80% of the deferred questionnaires were answered within a 12 hour period. This illustrates that the deferral option is a good feature to have in ESM system. Overall we found that the majority of our users did use the ESM mechanism and complete the questionnaires, however this same set of users also left plenty of questionnaires empty and made regular use of the deferral option. We obtained 28 reports from 8 users that were generated from the "I'm annoyed!" button. Only 11 of our users used both the ESM and "I'm annoyed!" button mechanisms. This



(a) 'I am annoyed : Speed rate: Slow; Application: iTerm; Problems: "Can't connect to some sites or services"



(b) I am annoyed: Speed rate: Super slow; Problems: "Slow download or upload"

Figure 3: Correlation of poor performance reports with system and network performance indicators

indicates that many users prefer one mechanism over the other and thus having both definitely improves the chances of obtaining the target user feedback data.

In the following, we focus on the poor performance reports out of the 569 completed questionnaires (541 from ESM and 28 from the "I'm annoyed!" button) . We say a completed questionnaire constitutes a **poor performance report** if the user rated their connection quality as either a 1 or a 2, out of a 5-point scale (with 5 being the fastest) or if the user explicitly named an application they believe to be performing poorly or provided a description of the problem. Among the 541 ESM questionnaires, 67 of them were poor performance reports, and these were generated by 23 users. So in total, over our 2 week period, we received 95 poor performance reports from 24 users.

All of this points to the fact that in order to get sufficient data, a large number of users are needed, or the measurement experiment needs to be run for a very long time. Simply scaling up the number of users is at odds with well known practices in the HCI community. There, most surveys are done with a small set of users (say 20) [4] and the users agree to respond to the questionnaires. This well accepted methodology has the advantage that the users participating will use the tool properly, thus enabling a qualitative study. However the disadvantage of this approach is that the number of users that participate is very limited (i.e. rarely in the hundreds). In the Internet measurement community, we are used to data gathering activities that assembles thousands or millions of data points from an ongoing stream of events. Further exploration is needed in approaches that target a large user base, even if they are only partially trained or aren't guaranteed to be diligent in responding to feedback questionnaires. (Although we only present data from 40 users here, our next deployment will target a much larger user base.)

4. USER REPORTS ON PERFORMANCE

We now compare user reports of poor performance with system and network performance metrics that are easily measured automatically. We start with a simple analysis in which we consider seven basic metrics, and seek to understand if the poor performance epochs (as flagged by the user) co-occur with any of our automated metrics experiencing atypical, or "anomalous" behavior.

The specific performance indicators, seven in all, that we extracted from the traces are as follows: *Round-trip times (RTTs)*, computed as the elapsed time between sending a TCP packet and receiving the corresponding acknowledgment; *Data rate*, the aggregate bandwidth (in both directions) measured in bits/second;

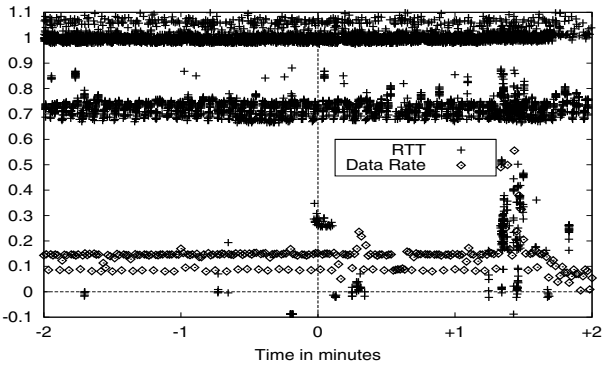


Figure 4: ESM: Everything Good

TCP retransmissions, which counts the number of segments retransmitted (per second); *TCP resets*, measured as the number of packets containing a RST flag (TCP reset); *CPU load*, the CPU utilization measured at 30 second intervals. Finally, when the end-host had an active and operational wireless interface, we also logged *wireless signal strength* and *wireless noise*, as reported by the `airport` or `iwconfig` utilities (on Mac OS and Linux, respectively).

For each of these seven performance indicators, we define an anomaly as follows: we build a distribution of the values using two weeks of data and identify the 5^{th} and 95^{th} percentile values of the distribution. We consider an instance of a metric that falls inside this range (5^{th} to 95^{th} percentile values) to be normal. We consider the following cases to be anomalous: i) if the *RTT*, *CPU load*, *TCP retransmissions* or *wireless noise* exceed the 95^{th} percentile value; ii) if the *data rate* or *wireless signal strength* fall below the 5^{th} percentile value; or iii) any instance of a *any* TCP reset.

We now look at some detailed examples that illustrate the sequence of events in time that occur just before and after a user completes a poor performance report. Figures 3(a), 3(b) and 4 all follow the same format; the time the user submitted the report is marked 0 on the x-axis (indicated by the vertical line) and we assume the poor performance epoch occurred close to this time. The brief time series on the x-axis indicates what occurred for 2 minutes before and after the report was filed. Because the performance metrics all have different ranges, we rescaled them so as to be able to view them (y-axis) on the same plot. The normal range is mapped to values between $[0, 1]$ with the 5^{th} (95^{th}) percentile value mapped to 0 (1), respectively. Values in the range $[min, 5^{th}\text{percentile}]$ (and that in $[95^{th}percentile, max]$)

Problem (number of reports)	Data rate	Wireless signal strength	Wireless noise	CPU load	RTT	TCP reset	TCP retransmissions
Slow browser (27)	14	2	0	10	18	23	14
Slow download or upload (24)	12	2	1	7	17	22	14
Poor voice or audio quality (14)	7	0	0	2	8	9	2
Poor video quality (11)	5	2	0	2	7	9	6
Can't connect to some sites or services (10)	3	1	1	3	8	8	6
Total (95 reports)	38	8	2	24	56	74	38

Table 3: Breakdown of anomalies across bad performance problems picked by users

are mapped to $[-0.1, 0]$ (and $[1, 1.1]$ respectively). Thus anomalies are those falling below 0 or above 1 (depending upon the metric). TCP RSTs are treated differently, and we simply mark the time that the RST was observed.

In Figure 3(a) the user reported a bad performance epoch by using the "I'm annoyed!" mechanism and provided the additional context: *"Speed rate: Slow"; "Application: iTerm"; "Problems: Can't connect to some sites or services"*. Here we see that the *RTT*s appear to rise and fall, and occasionally cross the $y = 1$ line (i.e., exceed the 95^{th} percentile of the distribution), and we posit that the performance problem experienced by the user was due to the very large round trip times observed. This variation (pattern) in RTTs is typical behavior when the user is behind a large buffer. The stream of packets exchanged with the end-host gradually fills up the buffer introducing additional latency. When the buffer fills up, it starts dropping packets causing TCP connections to reduce their rate; this buffer bloat issue is becoming a major problem for ISPs (see `http://www.bufferbloat.net/`). To keep the plot readable, we only indicate the performance metrics relevant to this instance being anomalous.

Figure 3(b) is another instance where the user complained about bad performance. Here, the metrics plotted are *data rate*, *wireless signal strength*, and *wireless noise*. We see that the *wireless signal strength* is consistently below the 5^{th} percentile value (below the $y = 0$ line) around the time the user complained. In this case, the user annoyance with their performance may have been due to poor wireless quality that adversely affected the available bandwidth.

In the above two cases we found anomalous events co-occurring with the user's poor performance report and thus could postulate that when network performance was at the fringe of its normal behavior, this particular user unhappy with it. Two other scenarios are possible in terms of the consistency between system metrics and user reporting. If the performance metrics are anomalous, but the user files a report indicating there is no problem, then this is an indication that the user can tolerate such performance levels. On the other hand, if a user files a report indicating poor performance and none of our metrics exhibit anomalous behavior, then it means that more metrics are needed to understand what annoyed the user. (Clearly, our seven metrics considered here are merely a starting point, and we plan to examine others, especially application level metrics, shortly.)

Figure 4 shows an example of a completed ESM performance report. The user said there was no problem when completing the report, yet it is very clear from the figure that the RTT values consistently exceed the 95^{th} percentile values, and this is also reflected in the low data rates. We looked into this case in more detail and found that at the time the report was completed, the user had 3 processes running: a large file transfer to a foreign country, mail and HostView. HostView was engaged in a data upload to a central server but this activity was capped at 100 Kbps and the RTTs to the server were between 0.7 and 0.8 in Figure 4; the mail activity generated minimal traffic, however the RTTs for the file transfer traffic were above 95^{th} percentile. In this mix of applications, it is possible that the user was happily reading emails and not bothered if the file transfer occurred slowly.

We performed this type of analysis over all of the 95 poor performance reports we received. Overall, we found that in 82 out of these 95 instances, at least one of our seven performance metrics exhibited anomalous behavior. The anomalies that we could associate with each of these 82 poor performance reports are summarized in Table 3. The first column indicates the type of problem the user selected (among a list of choices that included 'Other' and 'None'). The number next to the problem indicates the number of reports received of that problem type; for example, we received 27 reports about slow browsers. The count in the data rate column indicates that in 14 of those 27 reports (row 1), the data rate metric crossed its threshold line for our simple definition of anomalous. Note that the values in a single row can exceed the total number of reports because sometimes in a single report, multiple metrics can be anomalous. In order to explain the 13 reports for which we found no anomalous event across our performance indicators, we need to examine additional performance metrics.

We see in the table that all of our metrics, except wireless noise, were implicated in some problem. We are not claiming that the poor performance epoch a user experienced was caused by the anomaly we observe; in fact our metrics often themselves indicate some other underlying performance problem. We are merely claiming that there can often be simple indicators that track user annoyance, and the fact that our seven metrics are helpful in explaining 82 of the 95 poor performance reports we received is promising; one may not need to look at hundreds of features to understand the majority of incidents. Pinpointing the exact root causes for poor network performance problems is a challenging task and much more work needs to be done. Our work here is an initial foray toward the eventual goal of detecting performance degradations on the fly and performing proactive mitigation where possible.

5. CONCLUSION

We learned some interesting lessons in our first deployment of HostView. First, we believe that designing for diversity is an important paradigm in the design of end-host tracing tools. Designing for diversity implies that both the monitoring tool and the user-feedback mechanism should be adaptive. Second, due to the multiple challenges in getting user feedback data at poor performance epochs, we believe that the sampling mechanism should be weighted to prompt users when anomalous events occur. Third, we found that despite all of the challenges in collecting feedback for performance diagnosis, the two sampling methods we used together generated a fair amount of feedback. Fourth, we found that most users reporting poor performance episodes did so consistently and that reports often correlate with some performance metrics exhibiting outlier behaviors. This is promising for research in the area of automated diagnosis at end-hosts.

Acknowledgements

This project is supported by the European Community's Seventh Framework Programme (FP7/2007-2013) no. 258378 (FIGARO) and the ANR project C'MON.

6. REFERENCES

[1] P. Bahl, R. Chandra, A. Greenberg, S. Kandula, D. Maltz, and M. Zhang. Towards Highly Reliable Enterprise Network Services via Inference of Multi-level Dependencies. In *ACM SIGCOMM*, 2007.

[2] K.-T. Chen, P. Huang, and C.-L. Lei. How sensitive are online gamers to network quality? *Commun. ACM*, 49(11):34–38, 2006.

[3] K.-T. Chen, C. C. Tu, and W.-C. Xiao. Oneclick: A framework for measuring network quality of experience. In *Proceedings of IEEE INFOCOM 2009*, 2009.

[4] S. Consolvo, J. Jung, D. Avrahami, P. Powledge, B. Greenstein, and G. Maganis. The Wi-Fi Privacy Ticker: Improving Awareness and Control of Personal Information Exposure on Wi-Fi. In *Proc. of Ubicomp*, October 2010.

[5] S. Consolvo and M. Walker. Using the Experience Sampling Method to Evaluate Ubicomp Applications. *IEEE Pervasive Computing Magazine*, 2(2), 2003.

[6] E. Cooke, R. Mortier, A. Donnelly, P. Barham, and R. Isaacs. Reclaiming Network-wide Visibility Using Ubiquitous Endsystem Monitors. In *Usenix Technical Conference*, 2006.

[7] F. Gringoli, L. Salgarelli, M. Dusi, N. Cascarano, F. Risso, and K. Claffy. Gt: picking up the truth from the ground for internet traffic. In *ACM SIGCOMM CCR*, 2009.

[8] S. Guha, J. Chandrashekar, N. Taft, and D. Papagiannaki. How Healthy are TodayŠs Enterprise Networks? In *Proc. f the Internet Measurement Conference*, October 2008.

[9] W. Jiang and H. Schulzrinne. Modeling of packet loss and delay and their effect on real-time multimedia service quality. In *Proc. NOSSDAV*, 2000.

[10] D. Joumblatt, R. Teixeira, J. Chandrashekar, and N. Taft. Perspectives on Tracing End-Hosts: A Survey Summary. In *SIGCOMM CCR*, April 2009.

[11] D. Joumblatt, R. Teixeira, J. Chandrashekar, and N. Taft. HostView: Annotating End-Host Performance Measurements with User Feedback. In *HotMetrics, ACM SIGMETRICS Workshop*, 2010.

[12] D. Joumblatt, R. Teixeira, J. Chandrashekar, and N. Taft. HostView User Manual. `http://cmon.lip6.fr/EMD/Download_files/user-manual_Linux.pdf`, 2010. Pages 4,6.

[13] S. Kandula, R. Mahajan, P. Verkaik, S. Agarwal, J. Padhye, and P. Bahl. Detailed diagnosis in enterprise networks. In *ACM SIGCOMM*, 2009.

[14] T. Karagiannis, C. Gkantsidis, P. Key, E. Athanasopoulos, and E. Raftopoulos. Homemaestro: Distributed monitoring and diagnosis of performance anomalies in home networks. Technical Report MSR-TR-2008-161, Microsoft Research, 2008.

[15] C. Kreibich, N. Weaver, B. Nechaev, and V. Paxson. Netalyzr: illuminating the edge network. In *IMC*, 2010.

[16] A. Mahimkar, Z. Ge, A. Shaikh, J. Wang, J. Yates, Y. Zhang, and Q. Zhao. Towards Automated Performance Diagnosis in a Large IPTV Network. In *ACM SIGCOMM*, 2009.

[17] C. R. Simpson, Jr. and G. F. Riley. NETI@home: A distributed approach to collecting end-to-end network performance measurements. In *PAM2004*, April 2004.

[18] V. Singh, H. Schulzrinne, and K. Miao. DYSWIS: An architecture for automated diagnosis of networks. In *IEEE Network Operations and Management Symposium*, 2008.

[19] S. Tao, J. Apostolopoulos, and R. Guérin. Real-time monitoring of video quality in IP networks. *IEEE/ACM Trans. Netw.*, 16(5):1052–1065, October 2008.

Predicting User Dissatisfaction with Internet Application Performance at End-Hosts

Diana Joumblatt*, Jaideep Chandrashekar†, Branislav Kveton†, Nina Taft†and Renata Teixeira*
*CNRS and UPMC Sorbonne Universites
†Technicolor Research

***Abstract*—If a user's end system could predict when the user will be dissatisfied with the performance of networked applications, then the system could launch automated tools to improve the users's experience. Example tools include root cause diagnosis to assist the user in fixing the problem, or resource managers (e.g., bandwidth or video playout buffers) to tune the allocation of network resources to better serve the user. Unfortunately, predicting user dissatisfaction with application performance is not as simple as identifying outliers in typical network metrics such as high round-trip times or loss rates. Understanding user perception requires direct feedback from end users. This paper develops predictors of user dissatisfaction with Internet application performance. We train these predictors using network performance data collected passively from the machines of 19 users, who also input their feedback on network performance a few times per day. The main challenges of modeling user dissatisfaction with network performance comes from the scarcity of user feedback and the fact that poor performance episodes are rare. We develop a methodology to build training sets in face of these challenges. Then, we show that predictors based on non-linear support vector machine achieve higher true positive rates than predictors based on linear models. Our predictors consistently achieve true positive rates above 0.9. Finally we quantify the benefits of building per-application predictors over building general predictors that try to anticipate user dissatisfaction across multiple applications.**

I. Introduction

Users today expect high quality performance of computers that run demanding applications such as VoIP, video streaming, and always-on applications (cross-device or cloud synching). Moreover, users have little tolerance for slow URL loading or emails that aren't perceived as instantly delivered. Service providers must be able to measure and ensure users a high quality of experience (QoE) or they risk customer dissatisfaction and potential customer loss. If service providers, or end-host tools, could anticipate when a user might be dissatisfied with the performance of an application, they could proactively troubleshoot performance problems thereby improving the user's experience.

Previous research has developed tools and methodologies that can do fault diagnosis [1]–[4], test a user's network connectivity [5], [6], and arbitrate network resources to improve the end-user's experience [7], [8]. However many of these tools are launched either after a network problem has been detected or upon the user's request. Launching after a problem at the network layer is detected could be unnecessary if the user doesn't perceive it as a problem. Clearly removing the need to engage the user will improve their experience. However determining when to launch diagnosis or troubleshooting tools *automatically* is challenging because it is hard to reason about how a particular set of networking conditions affect the end user's perception of performance. There are two sources of difficulty here; one is that users perception of performance is inherently subjective, and the second is that today's monitoring tools can produce a vast amount of fine-grained measurements of many network and system metrics. This data then needs to be quantized, summarized and relevant features extracted in order to learn useful mappings from this data to a user's perception. In this paper, we focus on the goal of relating user perception and networking metrics and describe a method that can predict *episodes* when users are likely to be dissatisfied with application performance.

Previous research at tracking and improving the end-user experience has followed two different directions. In the first, network performance degradations are determined by using heuristics over raw network performance metrics such as delay, available bandwidth, losses, reordering, and such [9], [10]. The obvious shortcoming here is that the actual user response and perception is not taken into account, and the heuristics may not really track these well. To illustrate with an example, most modern video streaming applications incorporate enough of a buffer to tide over transient network outages. Tracking the performance metrics would reflect the transient outages, even as the end-user might be completely oblivious to them. In the second category, the research explicitly engages the end user but is focused on a few niche applications such as VoIP [11], [12], online gaming [13], video streaming [14], [15], and IPTV [16], [17]. While these are able to exploit application level semantics to build richer models, the results cannot be generalized to other applications. Building such application specific models is time consuming and requires a deep knowledge of the application (this information is often not available publicly). Despite recent interest in explicitly leveraging user feedback to achieve more general mappings of QoS to QoE [18]–[20], our community still doesn't have techniques that remain application agnostic and can predict user satisfaction given typical network-level metrics.

In this paper, we analyze passive measurements of system and network performance, which are annotated with a limited set of samples of end-user perception of network performance. Our user labeled performance data was collected from a deployment of 19 users who agreed to run our monitoring software on their laptops for periods varying from 2 weeks to

6 months. The basic problem we face is to design a binary classifier (satisfied or not) based on a key set of features that are culled from detailed low-level measurements. Having labeled data (user feedback) allows us to consider supervised learning techniques. However there are numerous challenges in building a training set for such techniques, that are likely to arise in many QoS to QoE mapping problems. First, users don't want to spend much time on giving feedback and thus provide at most a few samples per day (e.g. less than 5). Clearly the volume of user input is many orders of magnitude less than the data volume produced by monitoring tools that collect performance data on sub-second time scales. Second, because poor performance episodes are rare, we typically end up with far more samples labeled as satisfied than dissatisfied. Third, many monitoring tools miss some data. Our first contribution is a methodology for dealing with these issues in order to enable supervised learning techniques to be applied to this type of problem.

Our second contribution is to provide a solution for determining how to aggregate fine-grained machine data into a few meaningful metrics that are correlated with user perception. We identify a small set of features that are useful for prediction. One of the key ideas in our solution is to compute our selected features over two different windows of time, and use those simultaneously as input to our predictor. We propose LDA and SVM based predictors, and show that non-linear SVM outperforms other simpler linear predictors.

Our third contribution focuses on the issue of whether one should build one predictor for each application, or one predictor for multiple applications. The advantage of the former case is a better predictor, but the disadvantage is that it requires building many predictors. We select specific applications as use cases and quantify the performance of both specific and general predictors.

The remainder of this paper is structured as follows. Sec. II summarizes the data we collected using HostView. In Sec III we give an overview of our problem formulation and the challenges we face. Sec. IV presents our methodology for dealing with these issues. Sec. V shows that a naive predictor based on simple statistical heuristics is inaccurate. We then parametrize and evaluate our predictors in Section VI. Related work is summarized in Sec. VII and we conclude in Sec. VIII.

II. HostView measurements

The data used in this paper was collected using a tool we built, called HostView, that runs directly on enduser laptops. We conducted a user study with 19 users who kindly agreed to install and run our tool. Our users, mostly computer scientists, ran it for variable amounts of time ranging from 2 weeks to 6 months (starting in November of 2011). Seven of our users ran it on a Linux platform, while the other 12 used it on a MAC. For a more detailed description of the data collection methodology, please refer to our earlier work [21].

Our study is based upon data from 19 users, a typical number for user studies. We acknowledge that this is a small set of users. However building models for these users raises a number of issues that are likely to come up in a large population, and thus we believe that the problems we face in our task here are typical and require general solutions. We also point out that because monitoring tools such as HostView get so close to the user, it is hard to do large scale studies as many users perceive the tool as a privacy threat.

A. Performance metrics

To capture the *actual* performance that applications experience, HostView collects packet traces with libpcap; the traces are then processed offline using `tcptrace` [22] to extract detailed TCP connection metrics (RTT, jitter, resets and retransmissions) and data rates for TCP and UDP connections. Concurrently, we periodically poll for CPU load and WiFi signal strength from the host OS. Table I lists all the performance metrics extracted along with the manner in which the metrics are obtained.

We note some limitations in the metrics we track in terms of their comprehensiveness. First, tcptrace provides performance indicators for TCP, but only data rates for UDP. In our traces 93% of the connections are TCP based. Although some applications use both TCP and UDP, we will see that it is possible to build good predictors even without much UDP data. Second, also related to tcptrace, extracting RTT samples from the traces requires matching acknowledgements for data packets, which can only be done for upload traffic. When the data transfer is sparse or mostly download (as in the case of web downloads, or video streaming), sometimes there are an insufficient number of samples to estimate the RTT. However, even in such cases, RTT estimates are available during the connection setup and teardown phases. The same arguments hold for jitter estimates, which are derived from the RTT samples. Finally, with respect to the system and OS, we only monitor two high level metrics – CPU load and Wifi signal strength. There are potentially many other factors that could impact how a user perceives an application (excessive memory swapping, malfunctioning speakers, bad screen resolution). We believe it is unlikely that these problems persist over time (users tend to notice such things and fix them). Moreover, if some of the hard to measure variables correlate with the metrics we do measure, then they are often not needed for training a predictor, as they are likely redundant. In spite of these limitations, our results will show that in most cases, the relatively small set of metrics in Table I is capable of capturing the user's perception of performance.

B. Application context

HostView also records the application context for each connection, i.e., it identifies the process executable associated with the connection, and the service being used (for web traffic, this is the top level domain). The process executable is obtained using the `gt` tool [23], and the top level domain is extracted by parsing DNS replies in the traces and associating them with the destination addresses.

	Metrics	Method	Freq.	Comments
Network	RTT	pcap+tcptrace	continuous	average per second
	Jitter	pcap+tcptrace	continuous	Difference between two consecutive RTTs of a connection, average per second
	Resets	pcap+tcptrace	continuous	Only resets that follow a SYN segment
	Retransmissions	pcap+tcptrace	continuous	-
	Per-connection data rates (up and down)	pcap+tcptrace	continuous	TCP and UDP
	Host data rates (up and down)	pcap+tcptrace	continuous	All TCP and UDP connections
Machine	CPU	top	30 secs	-
	SNR	airport (Mac OS) / iwconfig (Linux)	60 secs	$SNR_{dB} = RSSI_{dB} - Noise_{dB}$

Table I
PERFORMANCE METRICS

C. Recording User (Dis)satisfaction

HostView collects user feedback with two complementary mechanisms: a system-triggered questionnaire based on the Experience Sampling Methodology [24], and the "I'm annoyed!" button (which is engaged by clicking an icon that is always present, but unobtrusive, in a corner of the screen). The triggered questionnaire is designed to popup no more than thee times a day in order to avoid user annoyance. The questionnaire triggers at different performance levels so we can calibrate user feedback. However, poor performance episodes tend to be rare and we cannot hope to catch a lot of them by sampling merely 2 or 3 times/day. Thus the "I am annoyed" button allows the user to engage at any time - when they believe performance to be substandard and affecting them. In both mechanisms, the user is presented with the same questionnaire: (i) rate your internet speed from 1 (slow) - 5 (very fast), (ii) identify any applications (from a list of those running) that they are unhappy with, (iii) indicate the problem (from a set of choices), and (iv) express any other additional information via a freeform text box.

Based on the answers to these questions, we categorize each user feedback sample into one of two classes: the user is considered *dissatisfied* if she notes an application as problematic, or if she explicitly selects a problem among the pre-defined list of problems, or if she rates the Internet speed below three. Otherwise, the user is considered *satisfied* at the moment the feedback is given. All of the measurements from the lower layers that are collected in a small time window (discussed later) before the user supplies a completed questionnaire are labeled as satisfied or dissatisfied according to the user's input.

In the data we obtained from our user study, we received 1278 surveys indicating a satisfied user, and 422 indicating a dissatisfied user. About 55% of the times when users supplied feedback, they did not explicitly indicate any particular application being used. Table II enumerates all the feedback reports obtained from users, along the the specific "applications" that were identified as problematic by the end-users; the frequencies are also reported. Note here that some of the user labels point to *online services* (e.g. `youtube`, `tf1.fr`), while others refer to application executables on the host machine.

Applications	Dissatisfied	Satisfied	Users
Firefox	51	384	8
Mail	23	332	4
Google Chrome	22	104	3
Safari	21	46	7
Skype	19	260	5
SSH	12	227	7
Adium	8	385	3
YouTube	7	18	3
Totem	6	10	1
tf1.fr	5	0	1
GTalk, iCal, iTerm, git	8	23	4
Thunderbird, Dropbox	2	317	2

Table II
SUMMARY OF USER LABELS PER APPLICATIONS

III. OVERVIEW

A. Prediction problem formulation and evaluation metrics

Our end goal is to build a predictor that can anticipate whether or not a user is satisfied with the performance of their machine, as a function of the state of the machine at the time. We take "state" to be captured by the metrics collected (see Table I). Before realizing such a predictor, there are two challenges to address. First, we need to "summarize" the raw data and convert it to a form that can be used by a learning algorithm, typically a vector of features. If we suppose that a user feedback sample was obtained at time t, we denote $\mathbb{X}_t$ as the feature vector that is generated by the raw data logged in time interval $[t-\delta, t]$. Second, we need to find the prediction function f which has the following property.

$$f(\mathbb{X}_t) = \begin{cases} 1 & \text{if user is } \mathbf{dissatisfied} \text{ at } t \\ 0 & \text{if user is } \mathbf{satisfied} \text{ at } t \end{cases}$$

Since we have data samples that are associated with labels, i.e., the user's (dis)satisfaction, and considering that the predictor function f returns a binary value, we look at *supervised* machine learning methods to build binary classifiers.

For every data sample in our dataset, the predictor returns a binary value, and there are four possibilities to consider and these are enumerated below, along with the common notations for each.

		Output of $f(\mathbb{X}_t)$	
		1	0
User label at time t	Satisfied	False Positive	True Negative
	Dissatisfied	True Positive	False Negative

Let TP denote the total number of true positives in the test set, TN the total number of true negatives, FP the total

number of false positives, and FN the total number of false negatives. The performance of the predictors on the data is characterized by True and False Positive Rates which are defined as follows.

$$TPR = \frac{TP}{(TP + FN)}, FPR = \frac{FP}{(FP + TN)}$$

A low TPR indicates that the predictor will miss poor performance episodes and be ineffective, whereas a high false positive rate indicates that the system will often, and incorrectly, trigger troubleshooting mechanisms on the host and is thus inefficient and leads to very high overhead. Thus, we are looking for predictors with high TPR and low FPR.

B. Challenges

User feedback is scarce. HostView, as with any tool that requires user interaction, has to be very conscientious about prompting the user for feedback. If it is not, the tool is likely to pose an annoyance and will be soon uninstalled. Moreover, users will often choose to ignore the prompt (for feedback) when they are focused on a task. In our user study, users only answered 40% of the HostView ESM questionnaires presented to them. Thus the first source of data scarcity arises from the limitation of the mechanism itself. Dealing with this necessitates running the tool for a very long period of time (months, years maybe), which is hard to accomplish.

Each user feedback sample is limited to few applications. A second source of scarcity in user data is the following. Even when users do fill-in the feedback form to file a complaint, they typically name just one or two applications explicitly. Since there may be other applications running simultaneously, we cannot learn anything about the user's dissatisfaction with those other applications. On the other hand, when a user completes the questionnaire indicating they are satisfied (checking "Everything is good"), then we do assume that there is no misbehaving application. Yet, it is possible that an application running in the background may be performing poorly, but the user either doesn't care or is unaware of it. The fact that user feedback is per application implies that to get enough labeled samples *per application*, we must monitor each user for an even longer time period or find alternative ways to train predictors with a smaller number of labeled samples.

User feedback is unbalanced. In the vast majority of reports we received, users say that they are satisfied with the network performance. This is natural since poor performance episodes are rare events (and we cannot sample these aggressively). The HostView dataset has three times as many satisfied samples as those marked dissatisfied. This imbalance represents a challenge when training predictors, which tend to bias towards predicting the majority class.

Performance metrics may be missing. Section II discusses the limitations of our passive measurement approach that causes the values of some of the performance metrics to be missing. There are also other reasons that lead to missing measurements: transmission errors between the end-host and the HostView collection server; missing CPU data because of delayed invocation of the *top* command; and missing RTT and jitter when no data is being sent or received (in these instances, there are only resets and retransmissions). We must either find suitable values to fill in the missing values or prediction methods that work efficiently even when some of the input values are missing.

A predictor that works for one application may not work for another. Our goal is to train predictors that will accurately predict user dissatisfaction with network performance *at any given time*. Since a user's experience depends on the primary application being used, a natural question arises. Should we build predictors separately for each application, or can we build more general predictors that work across multiple applications? Building per-application predictors is likely to lead to more accurate predictors. However doing so requires running multiple predictors in parallel and this might be impractical. General predictors suffer the reverse tradeoff. It is an open question as to how accurate a single predictor for multiple applications can be. If the same set of conditions always annoys a given user, then for some users a general model might be sufficient. We explore both types in this paper.

Sec. IV presents our methods to build training sets in face of the first four challenges, whereas Sec. VI addresses the last challenges.

IV. How to build training data

In this section we describe how we process the raw traces to obtain post processed data that can be input to a learning algorithm.

A. Extracting Feature Vectors

Data that is used to train a classifier needs to be processed and put in the right form. This processing commonly involves four stages: (i) feature selection, (ii) quantization, (iii) featurization, and (iv) labeling.

Feature Selection: This is a critical element of applying machine learning methods and can take two forms: blindly select as many features as possible and rely on regularization (or automatic feature selection) methods to extract those of interest, or else use available domain knowledge and select features that are relevant to the problem. In our work, we follow the second approach and select a number of features, based on our previous knowledge, that are very likely to affect the network performance of applications. Approaches such as the first are more applicable in areas where there is very little known about the underlying problem (which is not the case for us). The particular features that we use are enumerated in Table I.

Quantization: The first step is to select a time window (or bins) of interest around the time of the user feedback. Small windows will localize the problem, but not track problems that the user does not immediately perceive. Larger windows could "smooth" out any transient that affect the user. We use δ to denote the bin size, and consider sizes of 1, 2 and 5 minutes. At the same time, it might be a good idea to incorporate history

since the particular problem that is frustrating the user at time t might be the accumulation (or end result) of performance anomalies that manifested in the past. To this end, we also define a longer window of time (going back into the past) denoted Δ. Thus, for each user feedback sample at time t, we also process and incorporate the data seen in $[t-\Delta, t]$ (as discussed next). The features computed over a longer history $[t-\Delta, t]$ are intended to capture performance disruptions that are persistent. Our intuition is that by incorporating state from both δ and Δ, it might be easier to find a portion of the feature space where user dissatisfaction is apparent.

Featurization: Next, having selected the right time granularity, we need to summarize the raw data into features that represent each bin. Theoretically, we could just feed the raw measurements into the training algorithms (as long as they are regularized), but this significantly introduces complexity and also introduces noise into the process. We consider the metrics listed in Tab. I and process as follows: for RTT, jitter, CPU load and SNR, we compute four descriptors of the distribution (computed with data observed only inside the window), which are the mean, median, std. dev., and 95%-ile. For resets and retransmissions, we simply report the total seen in the bin. For metrics related to data rates, we compute both outgoing and incoming and add it to the vector. Thus, in total, for each bin $[t-\delta, t]$, we generate vector $\mathbb{X}_t$ containing 20+20 features (corresponding to the small window and the history duration).

Another important step in featurizing the data deals with the granularity which is tracked (should we consider applications individually, or aggregate them). One possibility is to compute the RTT, jitter, throughput statistics using all connections from all applications. Alternatively we could separate the traffic for each application and generate these features at the granularity of application. Since it is difficult to know which method is best ahead of time, we do both and extract two different sets: in the first, which we call, machine-level featurization, we aggregate all the traffic and do not break out that of any applications (here, each vector contains 20 features). In the second, *application level featurization*, we partition the traffic by application and then extract features. Here, the RTT and jitter measurements reflect only the performance of the particular application. Note that we introduce two new features in this case, corresponding to *application* throughput (in each direction). Thus, the feature vectors we generate with application level featurization have 44 features.

Labeling Feature Vectors: The final step is to assign a label to each feature vector $\mathbb{X}_t$ by the particular label that was obtained from the end-user when the feedback was collected. Thus, each vector $\mathbb{X}_t$ is associated with a label that is either *satisfied* or *dissatisfied*.

B. Dealing with missing values

In some of the bins, the raw data is missing and we thus cannot compute the summary statistics to populate our feature vector. This is a problem because most supervised learning algorithms require the feature vectors which are input to be fully populated. Table III shows the percentage of missing values for two sample bin sizes and 3 specific applications. We see that RTT and jitter (which depends on RTTs) have the highest incidence of missing values, particularly in the case of Mail. However, the incidence drops drastically (45% with 2 minute bins to less than 2.5% when the window is 30 minutes).

App	SNR		CPU		RTT		Jitter	
	2	30	2	30	2	30	2	30
Skype	0.1	0.58	0	0	5.19	2.18	5.33	2.33
Mail	3.98	3.14	0.05	0	45.18	2.34	45.25	2.34
SSH	1.93	3.57	0.08	0.02	7.93	1.8	8.03	1.92

Table III
PERCENTAGE OF MISSING VALUES FOR VARIOUS BIN SIZES IN MINUTES

The missing values are caused by two factors. First, our raw data consists of measurements collected at very different timescales; ranging from packets collected at microsecond scales to CPU load measurements carried out every second and these are all synchronized together in time to generate the feature vectors. Occasionally values fall on one side or the other of a window boundary leading to missing values. Second, and more common, we see missing RTT measurements and this is an artifact of how TCP RTTs are estimated (as explained in Section II). With larger windows, the probability of seeing data packets increases and so the missing values decrease. We use a heuristic to fill in the missing values: in each bin where there are missing values, we simply extrapolate the values from the *median* RTT values seen in preceding bins. Since every connection has at least a single RTT measurement (from the SYN-ACK exchange at the beginning), we are able to populate the RTT and jitter values in every missing bin.

Finally, we often see missing SNR measurements when the user switches from a wireless to a wired interface. In order to deal with this case and ensure that this feature is populated, we assume a *perfect* wireless connection when using a wired interface and set the SNR value to 80 dB.

C. Dealing with unbalanced data

Recall that our data is severely unbalanced because we have an order of magnitude more "satisfied" reports than "dissatisfied" reports. If not dealt with, the resulting predictors will be biased and more likely to predict the windows when users are satisified (which corresponds to a low TPR). In order to deal with this, we employ a trick that is commonly used in the literature for situations like this [25]. In the training phase, we randomly repeat data samples for the dissatisfied reports until the two classes are more or less even. This rebalancing is only done in the training phase when building the detector. When testing, the duplicated instances are removed and only the original data is used.

D. Per-application case studies

To build per-application predictors, we select the five applications with the most user feedback in Tab. II (excluding web browsers): Mail, Skype, SSH, YouTube, and Adium. We exclude web browsers because they are used to carry out very

Case	TPR	FPR
Machine	0.82	0.69
Skype	0.65	0.30
Mail	0.5	0.26
SSH	0.5	0.3

Table IV
EVALUATION OF NAIVE PREDICTOR

different tasks, and interact with a diverse set of services. Each of these interactions could look very different from the other; this counteracts the notion of a baseline performance profile. Hence, we only consider the user reports for browsers when the user has explicitly indicated an online service as problematic (for example, in the case of YouTube). In these cases, we collect and associate the browsers traffic to the service in question and include it in our data; however, we use the online service as the name of the application.

V. NAIVE PREDICTOR

Here we provide some motivation for using fairly sophisticated machine learning techniques to build the user (dis)satisfaction models we explore in this paper. We construct a naive, strawman predictor which is defined as follows: the predictor set $f(\mathbb{X}_t) = 1$ if the value of any feature in $\mathbb{X}_t$ is an outlier. The latter is defined, for almost all the features, based on the 95%-ile value of the feature distribution (as computed in the entire measurement period for each user). Exceptions are SNR (outlier is below the 5%-ile value), and TCP resets (outlier if at least one reset was present).

Table IV illustrates the performance of the naive predictor. In the table, we show results for a predictor based on machine-level featurization (marked "Machine" in the table), and those based on application-level featurization for a few sample applications (marked with the application name). We see that in the machine case, the TPR of the naive predictor is above 0.8 (which is not terrible); however the FPR is unacceptably high. One could argue that using a different threshold (perhaps the 98%-ile) would decrease the FPR; however, this would also cause the TPR to drop significantly. We also see poor performance for the applications; both TPR and FPR are at unacceptable levels. In all likelihood, the naive predictor performs poorly because it cannot account for the dependencies and correlations between different features, which may be quite complicated. This directly motivates the use of more principled machine learning techniques that can consider features simultaneously.

VI. MULTI-FEATURE PREDICTOR

In this section, we compare the performance of three fully supervised machine learning techniques: Linear Discriminant Analysis (LDA), Linear Support Vector Machines (l-svm) and Non-Linear Support Vector Machines (nl-svm). We find that, among the three, nl-svm produces optimal results with our data set. We construct three *families of predictors*, one based on machine-level features, and two based on application-level features and compare these while varying the parameters (δ, Δ). In the best case, $(\delta = 5, \Delta = 60)$, we find that the nl-svm based predictor that was built with application-level features outperforms the other two families. Finally, with the best performing predictor in hand, we extrapolate its performance on unlabeled data.

A. Overview of Techniques

We briefly describe some of the background for the three machine learning (classification) techniques that we employ to build our predictor. At a high level, all three are instances of algorithms that build two-class classifiers, and differ in what they assume about the data distributions, and their ability to deal with varying kinds of data representation. In this paper, we focus on supervised learning approaches and leave semi-supervised learning as future work. The latter tend to be computationally expensive [26] For instance, the complexity of graph-based methods, which inherit nice theoretical properties, is $\Omega(n^2)$, where n is the number of data points. These methods are impractical in our setting because n is too large (from the most prolific user, we obtain about 30k data samples). Another criticism is that of such methods is that unlabeled data does not always improve performance [27].

Linear Discriminant Analysis [28], a classical method for classification, is applicable if we can assume that there are exactly two classes of data (e.g., satisfied and dissatisfied users) extracted from normal populations. It finds a hyperplane whose normal maximizes the ratio of the interclass variance to the intraclass variance. The hyperplane is given by $f(\mathbf{x}) = \langle w, \mathbf{x} \rangle + w_0$. Then, $d(\mathbf{x}_i) = \frac{f(\mathbf{x}_i)}{||w||_2}$ is the distance of a sample $\mathbf{x}_i$ to the separating hyperplane. Intuitively, the confidence of the discriminator increases with greater distances. In the actual prediction, we also use a threshold parameter, τ, that can exclude points that are close to the separator in order to get greater confidence in the results.

Support Vector Machines are a method to learn max-margin classifiers of data [29]. Unlike LDA, it does not rely on strong assumptions about the underlying distribution in the data. It also attempts to find a "best" separating boundary between the two classes; the difference here is that hyperplane maximizes the distance from the closest data points, called support vectors, for both classes. The SVM solution is:

$$\min_{f \in \mathcal{H}_K} \frac{1}{T} \sum_{t=1}^{T} V(f, \mathbf{x}_t, y_t) + \gamma \left\| f \right\|_K^2 ,$$

where V captures loss from misclassification and $\|\cdot\|_K$ penalizes the complexity of f (roughly, number of features). The tradeoff between model complexity and the loss is controlled by the parameter γ.

The SVM predictor has the form $f(\mathbf{x}) = \sum_i k(\mathbf{x}, \mathbf{x}_i)\alpha_i y_i$ where $\mathbf{x}_i$ is the i-th data sample with label y_i, and α_i is the coefficient learned by the SVM. If the data point "supports" the discovered separator, α_i is non-zero. The *confidence* of the prediction, $\mathrm{sgn}(f(\mathbf{x}))$, is given by $1/(1 + exp(-f(\mathbf{x})/c)$, i.e., the sigmoid of $f(\mathbf{x})$ multiplied by some scalar c, and this is monotonic in $f(\mathbf{x})$ (and can be thresholded).

If the input data is not linearly separable, svm applies the kernel trick by mapping the data into a higher dimensional

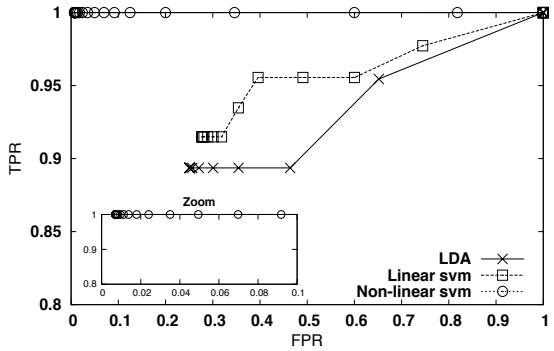


Figure 1. ROC CURVE FOR THE GENERAL PREDICTOR (δ=5, Δ=60)

space where the classifier is a hyperplane. One of the kernel families that is commonly used in non-linear SVMs is the *radial basis kernel*, given by $k(\mathbf{x}_i, \mathbf{x}_j) = \exp\left[-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{2\sigma^2}\right]$, where σ is the kernel width which is chosen in advance (we use the recommended setting $\sigma = 1/|\text{features}|$)

Again, when building the predictor, we also incorporate a threshold τ to exclude data samples that are close to the hyperplane.

B. Comparison of predictors

We construct three families of predictors using the techniques introduced previously. *Machine* predictors are constructed by training on machine-level features across all machines, and the testing is carried out for similar data. *General* predictors are built by training on application-level features, but with data that is the union of all the applications (and all the hosts). The predictor is tested on data from particular applications. Finally, *application specific* predictors are built by training on data that is specific to that application (from all the hosts), and performance is tested on data of that particular application. Note that the first family of predictor works with vectors of 40 features; the other two use 44 features. Note from Sec. II that traffic from browsers is excluded except in select cases.

We experimented with various settings for δ, Δ, τ and selected the optimal settings using leave-one-out cross validation. In figure 1, we show a representative RoC curve where we hold the window parameters fixed ($\delta = 5, \Delta = 60$) while sweeping through the threshold parameter τ. The best performing predictor is found at the top-left of the graph (high TPR and low FPR).

Unsurprisingly the plot shows that the nl-svm based predictor is consistently above and to the left of the curves for l-svm and LDA, indicating that overall nl-SVM outperforms the other two methods (the take-away was identical when we generated curves varying δ and Δ). The l-svm predictor achieves a respectable TPR of roughly 91%, however it does so with a FPR approaching 30%. This RoC illustrates that by using a non-linear SVM instead of a linear one, we can significantly lower the false positive rate. If we zoom into the top left of the plot (show inset figure) and look at the nl-svm in the best performing region we see that as τ increases (-3.5 to -0.4), TPR remains constant but FPR increases gradually from 0.003 to 0.1; a possible interpretation the metrics observed during bad performance episodes are different (for the most part) from "normal", and the nl-svm can uncover the structure of this set, but there are a few samples where users were not dissatisfied that also stand out from the norm. This again underscores the perception "gap" between what user's experience and what is observed in the network metrics. We point out that since the linear models perform decently and the non-linear model performs excellently, this means that our methods for handling missing data (such as interpolation) are effective.

C. Optimal Predictor Parameters

In order to determine the ideal value for δ, we compare three different settings $\delta = 1, 2, 5$ using the nl-svm predictor (the most powerful one). The results are reported in Table V and we find that a window of 5 minutes returns the best performance.

Subsequently, with the ideal window size, we then look at the effect of varying history lengths (i.e, value of Δ) upon performance. We select δ=5 and compare performance for the three predictor families varying Δ=15, 30 or 60 minutes. Table VI reports these results and it is very clear incorporating history improves both TPR and FPR across the board. We also answer the "how much" question. In Table VI $\Delta = 60$ brings the greatest benefit. This is intuitive considering that when $\delta = 5$ minutes, $\Delta = 60$ implies adding 11 additional little bins. In previous experiments, we also considered $\Delta = 0$ (no history is used) and this performed quite poorly. Interestingly, the *machine* predictor does not improve even as history is taken into account. This seems to imply that distinguishing applications has a much higher impact on performance than accounting for history.

D. Application predictor vs. general predictor

To further develop the closing statement made in the last paragraph, here we compare the general performance of the three different predictors built with application-level features. Here we are trying to understand if the predictors that see more samples of application behavior, albeit from many different heterogenous applications have an edge over those that only see homogenous data (from a single application). We build *application specific* predictors for the 5 applications indicated in Table VII, and compare them with the other two *general* predictors. In the first of these, denoted "general", the training data mixes features vectors from *all* the applications; in the second, denoted "general, no Mail" (the training data incorporates data from all other apps save for Mail). In both cases, the testing is done against the application-level feature vectors corresponding to Mail. To read the table, consider that the training data for the predictor is as denoted in the "predictor" column, and the testing data is as denoted in the "tested app" column.

We see in the table that for 3 applications, namely Mail, SSH and Adium, the per-application predictor outperforms a general predictor. In these cases, the improvement of a

Predictor	1 minute		2 minutes		5 minutes	
	TPR	FPR	TPR	FPR	TPR	FPR
Mail	0.94	0.09	0.84	0.08	0.95	0.06
Skype	0.86	0.16	0.93	0.11	0.93	0.14
SSH	0.8	0.06	1	0.12	1	0.01
YouTube	1	0	1	0.15	1	0.06
Adium	1	0.02	1	0.02	1	0.01
Machine	0.49	0.12	0.55	0.12	0.57	0.14

Table V
COMPARISON OF BIN SIZES, δ, FOR NON-LINEAR SVM WITH NO HISTORY

Predictor	15 minutes		30 minutes		60 minutes	
	TPR	FPR	TPR	FPR	TPR	FPR
Mail	0.89	0.04	0.95	0.03	1	0.03
Skype	1	0.09	1	0.06	1	0.06
SSH	1	0.01	1	0.01	1	0.01
YouTube	1	0	1	0	1	0
Adium	1	0	1	0	1	0
Machine	0.60	0.13	0.60	0.11	0.62	0.11

Table VI
AFFECT OF VARYING Δ, FOR NON-LINEAR SVM WITH HISTORY, AND FIXED δ=5

Tested App	Predictor	TPR	FPR
Mail	**Mail**	1	0.03
	general	0.84	0.06
	general, no Mail	0.05	0.03
Skype	**Skype**	1	0.06
	general	1	0.03
	general, no Skype	0.07	0.01
SSH	**SSH**	1	0.01
	general	0.75	0.27
	general, no SSH	0	0.05
YouTube	**YouTube**	1	0
	general	1	0.06
	general, no YouTube	0	0
Adium	**Adium**	1	0
	general	0.5	0.005
	general, no Adium	0	0.1

Table VII
APPLICATION PREDICTORS VS. GENERAL PREDICTOR (NON-LINEAR SVM, δ= 5, Δ=60)

per-application predictor over a general one is significant in terms of TPR. Interestingly, for YouTube and Skype, the per-application and general predictors perform quite similarly. They both achieve full true positive detection (TPR=1) and only differ slightly in the FPR. In general, we would advocate for building per application predictors, especially frequently used ones, if there is enough data. Although the general ones do very well in some cases, the per-application predictors are more consistently good across multiple applications.

VII. RELATED WORK

The general area of network performance monitoring and QoS has been extremely prolific. Similarly, the area of user quality and QoE is fairly mature (for example, ITU-T's standards for voice quality measurement in the telephone system are over a decade old [30]). QoS studies focus only on raw performance metrics, whereas QoE studies mostly map user experience to application metrics (for instance, channel zapping time in IPTV or web page load times).

In this paper, we aim to bridge the gap between these two fields and map QoS metrics to QoE. A number of prior studies have built models of QoS to QoE by using application-level knowledge and metrics [11]–[13], [15]–[17]. These models, however, won't apply for other applications. We instead take an application-agnostic approach to better capture the user online experience.

Most related to our work are a number of efforts to collect and characterize network and system performance data annotated with user feedback from laptops and desktops [18]–[21], [31] as well as smartphones [32]. HostView [21], which provided the dataset used in this paper, is one example of these measurement efforts. These efforts have led to interesting preliminary insights on the relationship between QoS and QoE and some of the causes of user dissatisfaction, but not to general models or predictors of QoE based on QoS parameters as we do in this paper.

The only exception is OneClick [33], which uses a Poisson regression model to correlate network metrics of one application with the rate that users click on a button to indicate dissatisfaction with this specific application. Although this general framework can be applied to different applications, the model has to be trained for one application at a time. OneClick's evaluation was based on asking users to input feedback while watching or listening pre-recorded traces with different levels of loss rates, not with traces collected when users are interacting with their usual network environment and experiencing real performance degradations. In contrast, our solution can automatically build predictors for any application users interact with and flag user dissatisfaction under real network conditions (based on a large set of features).

VIII. DISCUSSION AND FUTURE WORK

In this section, we discuss some open issues that must be addressed for our predictor to be integrated into an online diagnosis system.

Implementation of online prediction and diagnosis. This paper relied on the traffic traces processed offline to extract connection metrics and generate the feature vector $\mathbb{X}_t$ per application. In an actual system implementation, we must generate $\mathbb{X}_t$ online. HostView already collects packet traces and machine metrics online, one would need to add a module that processes packets as they come to extract the relevant features. Alternatively, one could use a kernel extension such as `WEB100` [34] to directly query TCP state information. Besides tracking $\mathbb{X}_t$, an online system will also have to predict whether the user is dissatisfied at every time bin and launch a diagnostic tool when a time bin is labeled as dissatisfied. The design and implementation of a system that performs all these tasks in real-time without overloading the user's machine in terms of CPU or storage represents a challenge that we plan to address in our future work.

Another aspect related to running such a prediction tool in real time is to understand the "annoyance" factor, or firing rate. This is important in our case because a prediction of *dissatisfaction* will cause the sytem to invoke a more heavy

duty diagnosis tool that might interrupt the user (or disconnect the system). In order to understand this, we ran both l-svm and nl-svm using the discovered optimal parameters against all the samples of unlabeled data, i.e., features extracted when data was collected which do not have a user feedback label associated with it. We found that the nl-svm raises fewer flags (unsurprisingly) than the l-svm, with the rate of predicting bad performance (or user dissatisfaction) staying below 2%. This gives us a greater deal of confidence that the end-user will not be adversely affected by our predictor when running in an on-line fashion.

Configuration of online predictor. The online predictor has to be configured with the models we obtain during our training phase. In the ideal scenario, we would train the predictor once and apply the same model to all users. For that, we need to conduct further research to better understand the generality of our predictors. Our results in Table VII show that a predictor learned for one application doesn't apply to another. Hence, in order to accurately predict user dissatisfaction with one application we need labeled samples of this particular application to train predictors. In our future work, we plan to explore other related questions. Does a predictor learned from one user apply to another? How often do we need to re-train predictors? If predictors are specific for a given user, then we can imagine that the system would have an initial calibration phase, during which users would provide explicit feedback. Once the system gathers enough data, it could then perform the training phase and configure the online predictor. This calibration phase could be re-run periodically to ensure high prediction accuracy.

Characteristics of samples labeled with user dissatisfaction. Finally, we would also like to study in our future work the characteristics of poor performance episodes to understand which features (or combinations of features) are the most important in predicting user dissatisfaction. We could also investigate whether this set of dominant features varies or remains the same across different applications. Intuitively, it should vary as different applications have different requirements of bandwidth, delay, or jitter.

REFERENCES

[1] P. Bahl, R. Chandra, A. Greenberg, S. Kandula, D. Maltz, and M. Zhang, "Towards Highly Reliable Enterprise Network Services via Inference of Multi-level Dependencies," in *Proc. ACM SIGCOMM*, 2007.

[2] S. Kandula, R. Mahajan, P. Verkaik, S. Agarwal, J. Padhye, and P. Bahl, "Detailed diagnosis in enterprise networks," in *Proc. ACM SIGCOMM*, 2009.

[3] A. Dhamdhere, R. Teixeira, C. Drovolis, and C. Diot, "NetDiagnoser: Troubleshooting Network Unreachabilities Using End-to-end Probes and Routing Data," in *Proc. CoNEXT*, 2007.

[4] R. Kompella, J. Yates, A. Greenberg, and A. Snoeren, "Detection and Localization of Network Blackholes," in *Proc. IEEE INFOCOM*, 2007.

[5] C. Kreibich, N. Weaver, B. Nechaev, and V. Paxson, "Netalyzr: illuminating the edge network," in *Proc. Internet Measurement Conference*, 2010.

[6] R. Mortier, B. Bedwell, K. Glover, T. Lodge, T. Rodden, C. Rotsos, A. W. Moore, A. Koliousis, and J. Sventek, "Supporting novel home network management interfaces with openflow and nox," *ACM SIGCOMM Computer Communication Review*, 2011.

[7] T. S. An Chan, Henrik Lundgren, "Video-aware rate adaptation for mimo wlans," in *Proc. International Conference on Network Protocols*, 2011.

[8] C. Gkantsidis, T. Karagiannis, P. Key, B. Radunovic, E. Raftopoulos, and D. Manjunath, "Traffic management and resource allocation in small wired/wireless networks," in *Proc. CoNEXT*, 2009.

[9] M. Zhang, C. Zhang, V. Pai, L. Peterson, and R. Wang, "PlanetSeer: Internet Path Failure Monitoring and Characterization in Wide-area Services," in *Proc. USENIX OSDI*, 2004.

[10] J. Sommers, P. Barford, N. Duffield, and A. Ron, "Accurate and Efficient SLA Compliance Monitoring," in *Proc. ACM SIGCOMM*, 2007.

[11] W. Jiang and H. Schulzrinne, "Modeling of packet loss and delay and their effect on real-time multimedia service quality," in *NOSSDAV*, 2000.

[12] K. Chen, C. Huang, P. Huang, and C.-L. Lei, "Quantifying skype user satisfaction," in *Proc. ACM SIGCOMM*, 2006.

[13] K.-T. Chen, P. Huang, and C.-L. Lei, "How sensitive are online gamers to network quality?," *Commun. ACM*, vol. 49, no. 11, pp. 34–38, 2006.

[14] S. Tao, J. Apostolopoulos, and R. Guérin, "Real-time monitoring of video quality in IP networks," in *Proc. ACM NOSSDAV Network and Operating System Support for Digital Audio and Video*, 2008.

[15] R. K. Mok, E. W. Chan, X. Luo, and R. K. Chang, "Inferring the QoE of HTTP video streaming from user-viewing activities," in *Proc. SIGCOMM Workshop on Measurements Up the Stack*, 2011.

[16] A. Mahimkar, Z. Ge, A. Shaikh, J. Wang, J. Yates, Y. Zhang, and Q. Zhao, "Towards Automated Performance Diagnosis in a Large IPTV Network," in *Proc. ACM SIGCOMM*, 2009.

[17] H. H. Song, Z. Ge, A. Mahimkar, J. Wang, J. Yates, Y. Zhang, A. Basso, and M. Chen, "Q-score: proactive service quality assessment in a large iptv system," in *Proc. Internet Measurement Conference*, 2011.

[18] J. S. Miller, A. Mondal, R. Potharaju, P. A. Dinda, and A. Kuzmanovic, "Understanding end-user perception of network problems," in *Proc. SIGCOMM Workshop on Measurements Up the Stack*, 2011.

[19] R. Schatz and S. Egger, "Vienna surfing: assessing mobile broadband quality in the field," in *Proc. SIGCOMM Workshop on Measurements Up the Stack*, 2011.

[20] D. Joumblatt, R. Teixeira, J. Chandrashekar, and N. Taft, "Performance of Networked Applications: The Challenges in Capturing the User's Perception," in *Proc. SIGCOMM Workshop on Measurements Up the Stack*, 2011.

[21] D. Joumblatt, R. Teixeira, J. Chandrashekar, and N. Taft, "HostView: Annotating End-Host Performance Measurements with User Feedback," *ACM SIGMETRICS Performance Evaluation Review*, vol. 38, no. 3, pp. 43–48, 2011.

[22] "TcpTrace." http://www.tcptrace.org.

[23] F. Gringoli, L. Salgarelli, M. Dusi, N. Cascarano, F. Risso, and K. Claffy, "Gt: picking up the truth from the ground for internet traffic," in *ACM SIGCOMM Computer Communication Review*, 2009.

[24] S. Consolvo and M. Walker, "Using the Experience Sampling Method to Evaluate Ubicomp Applications," *IEEE Pervasive Computing Magazine*, vol. 2, no. 2, 2003.

[25] R. Duda, P. Hart, and D. Stork, *Pattern Classification*. Wiley, 2000.

[26] X. Zhu, "Semi-supervised learning literature survey," Tech. Rep. 1530, University of Wisconsin-Madison, 2008.

[27] S. Ben-David, T. Lu, and D. Pal, "Does unlabeled data provably help? worst-case analysis of the sample complexity of semi-supervised learning," in *Proceedings of the 21st Annual Conference on Learning Theory*, pp. 33–44, 2008.

[28] R. Duda and P. Hart, *Pattern Classification and Scene Analysis*. Wiley, 1973.

[29] V. Vapnik, *The Nature of Statistical Learning Theory*. New York, NY: Springer-Verlag, 1995.

[30] "Itu-t recommendation p.862. perceptual evaluation of speech quality (pesq): An objective method for end-to-end speech quality assessment of narrow-band telephone networks and speech codecs," tech. rep., 2001.

[31] T. Karagiannis, C. Gkantsidis, P. Key, E. Athanasopoulos, and E. Raftopoulos, "Homemaestro: Distributed monitoring and diagnosis of performance anomalies in home networks," Tech. Rep. MSR-TR-2008-161, Microsoft Research, 2008.

[32] K. W, S. Ickin, J. Hong, L. Janowski, M. F., and A. Dey, "Studying the experience of mobile applications used in different contexts of daily life," in *Proc. SIGCOMM Workshop on Measurements Up the Stack*, 2011.

[33] K.-T. Chen, C. C. Tu, and W.-C. Xiao, "Oneclick: A framework for measuring network quality of experience," in *Proc. IEEE INFOCOM*, 2009.

[34] "WEB100." http://www.web100.org.

An end-host view on local traffic at home and work

Ahlem Reggani[1], Fabian Schneider[2], and Renata Teixeira[1]

[1] UPMC Sorbonne Universités and CNRS, LIP6, Paris, France
[2] NEC Laboratories Europe, Heidelberg, Germany (work done at UPMC[1])

Abstract. This paper compares local and wide-area traffic from end-hosts connected to different home and work networks. We base our analysis on network and application traces collected from 47 end-hosts for at least one week. We compare traffic patterns in terms of number of connections, bytes, duration, and applications. Not surprisingly, wide-area traffic dominates local traffic for most users. Local connections are often shorter and smaller than Internet connections. Moreover, we find that name services (DNS) and network file systems are the most common local applications, whereas web surfing and P2P, which are the most popular applications in the wide-area, are not significant locally.

1 Introduction

The past couple of decades has seen many studies that characterize Internet traffic [1, 6, 7, 12]. These studies are based on packet traces collected in ISP networks, at border routers of university campuses or enterprise networks. As such, most prior studies focus on wide-area traffic. Little is known about the traffic that stays inside a network, which we call *local traffic*. The main exception is the study of traffic from one enterprise [8, 9], which shows that local traffic is different from wide-area traffic with a significant amount of name service, network file system, and backup traffic. As the authors point out their study is "an example of what modern enterprise traffic looks like" [9]. It is crucial to reappraise such analysis in other enterprises and more important in other types of edge networks. For instance, the spread of broadband Internet has caused an increase in the number of households that have a home network. Yet, there has only been limited analysis of local traffic volumes in three home networks [5], but no in depth characterization of in-home traffic patterns. The challenge of studying local traffic across multiple edge networks is to obtain measurements from *inside* multiple networks.

This paper characterizes local network traffic of multiple networks from the perspective of an end-host that connects inside an edge network. This approach is in contrast with previous work [5,9], which instruments routers in the local network. Although instrumenting routers could capture all traffic traversing the local network, it is hard to have access to routers at more than a few networks. By monitoring traffic directly at end-hosts, we can sample a larger number of networks, but we can only see the traffic from one of the hosts in the network. For smaller networks (such as home networks) a single host's traffic captures a significant fraction of those networks total traffic, whereas for larger networks (as enterprises) this fraction is less significant.

We rely on data collected at end-hosts using the HostView monitoring tool [4]. HostView records packet header traces and information about applications and user

environment. The data we study was collected from 47 users who ran HostView for more than a week each. Given that users move between different networks, this dataset contains end-host traffic from a total of 185 different networks spread over 18 different countries. Section 2 gives an overview of the HostView data. The analysis of local and wide-area traffic from HostView data is challenging, because HostView has no information of which traffic flows are local. Worse, HostView scrapes the end-host IP address from the traces to protect user's privacy, which makes the identification of local traffic more challenging. Therefore, we develop a heuristic to separate local from wide-area traffic. Section 3 describes this heuristic together with our method to categorize environments and applications in the HostView data.

Our analysis (presented in Section 4) asks some high-level questions, for instance: How does the volume of an end-host's local traffic compare to wide-area traffic? Do local and wide-area applications differ? How does traffic vary between home and work? The results show that for most users wide-area traffic dominates local traffic, but that some users have over 80 % of local traffic. Local connections are mostly shorter and smaller than wide-area connections, but sometimes they transfer a larger amount of traffic than large wide-area connections. We find that typical local applications are DNS, ssh, and network file systems (confirming previous findings [9]). Moreover, common applications at work include backup, printing, and web. Yet, these applications are rarely used at home.

2 Summary of HostView Data

In this paper, we use three of the datasets collected by the HostView tool [4]: network packet traces, application labels, and the end-host's network environment. HostView logs all this data directly at the end-host into a trace file, which is periodically uploaded to a server. A new trace is created every four hours or when a change in the network interface or the IP address is detected.

Network traces and application context HostView logs the first 100 bytes of each packet sent and received by the end-host with libpcap. For DNS packets, it records the whole packet to enable offline hostname to IP address mappings. In this paper, we use the connection summaries generated by previous work [3]. Each connection summary record describes both directions of a TCP or UDP connection and includes (among other fields): The source and destination IP addresses (replacing the host IP address with "0.0.0.0" to comply with French privacy laws), the source and destination port numbers, and the network protocol; The number of bytes, the number of packets, and the duration of the connection; And the name of the process executable that generated the connection.

Network environment HostView labels each trace file with information describing the network environment the end-host is connected to, including the network interface, a hash of the wireless network SSID and of the BSSID of the access point for wireless networks or a hash of the MAC address of the gateway for wired networks. It also records the ISP, the city, and the country for each trace using the MaxMind GeoIP commercial database from March 2011. When the end-host connects to a new wireless

network, HostView asks the user to specify the network type from a pre-defined list: Home, Work, Airport, Hotel, Conference meeting, Friend's home, Public place, Coffee shop or Other (with the possibility to specify). This user tag is used to classify the network the user connects to according to an environment type. Unfortunately, this tag is not available for wired connections and users sometimes skip the questionnaire. Originally, only 40 % of HostView traces had a user tag, but after applying some heuristics (which exploit the fact that users connect to the same network with both wireless and wired, for instance) previous work was able to label 78 % of the traces [3]. Still, the data includes at least one unlabeled trace per user. The next section describes our method to label most of the remaining traces with an environment type.

Dataset characteristics and biases HostView was announced in networking conferences and researcher mailing lists. Volunteer users downloaded HostView (which is available only for Mac OS and Linux) and ran it during different time intervals between November 2010 and August 2011. In this paper, we use traces from 47 users who ran HostView for at least one week; 32 of these users ran HostView for more than a month.

Because of the way HostView was advertised and its limited operating-system support, the user population is biased towards networking researchers. We acknowledge that networking researchers probably use different applications than the average user and may also work from home. It is still interesting to study examples of the differences between local and wide-area traffic. We do observe a diverse set of applications among different users and our users do use some popular applications like YouTube, Facebook and BitTorrent. Furthermore, this bias influences the types of networks we study. Importantly, "work" is often a university. Overall, we study end-hosts connected to 185 unique networks spread over 18 different countries (Italy: 25, France: 22, Germany: 21, Rest of Europe: 31, Asia: 19, US: 63, Australia: 3, and Brazil: 1); 34 distinct home networks and 38 distinct work environments (29 are universities and 9 enterprises).

Another bias comes from using data collected for a limited time period on only one single end-host in the network. It is well known that traffic characteristics can vary considerably between different networks and over time [10]. HostView can only see a small fraction of the network's traffic and there are some types of traffic that it can never observe. For example, some homes may have a media server that serves content to the TV; this type of traffic traverses the home network, but it is not originated or consumed by an end-host. Despite these shortcomings, we believe that this end-host perspective on local versus wide-area traffic offers the unique opportunity to sample traffic in a relative large number of networks. Whenever appropriate, we also contrast our findings with previous work.

3 Methodology

In this paper, we compare local and wide-area traffic in networks of different types. In addition, we are interested in the traffic application mix. We follow three steps to label HostView traces before our analysis: *(i)* Differentiation of local and wide-area traffic, *(ii)* Extension of the incomplete network type labeling, and *(iii)* Categorization of connection records into application groups.

Table 1. Examples of process names and network services to category mappings. This list is not complete and only intended to give an idea.

Category	Process name (Examples)	Application protocols
Backup	retroclient	amanda
Chat	Skype, iChat, Adium, Pidgin	ircd, SIP, msnp, snpp, xmpp
DistantControl	ssh, sshd, VNC, screen sharing	ssh(22), webmin
Email	Mail, Outlook, Thunderbird	IMAP(S), POP3(S), (S)SMTP
Personal	Media players, games, productivity	rtsp
FileTransfer	ftp, dropbox, svn, git, SW updates	ftp, rsync, svn, cvspserver
Management	traceroute, iperf, nmap, ntpd, uPNP	BOOTP, MySQL, VPN, SNMP, whois
Miscellaneous	perl, python, VirtualBox, openvpn	—
NameService	dns, nmblookup, named, nmbd, nscd	domain(53), mdns, netbios-ns
NetworkFS	smbclient, smbd, AppleFileServer	AFP, AFS, LDAP, netbios, nfs
P2P	amule, uTorrent, transmission	amule, Kazaa, BitTorrent
Printing	cupsd, lpd, HP, Lexmark	ipp, printer
Web	Firefox, Chrome, Safari, Opera, httpd, plugin-container, WebKitPluginHost	HTTP(S)

Local vs. wide-area HostView does not collect the host IP address, so we cannot identify the local subnet based on the host IP prefix. We develop a number of heuristics to classify traffic as local or wide-area. We define *local* traffic as all the traffic exchanged between an end-user machine and a private IP address, i.e., `192.168/16`, `172.16/12`, `10/8`. We expect this classification to correctly match most local traffic at homes, as those typically connect through a NAT gateway sharing one public IP on the outside. To avoid misclassification when the ISP employs carrier-grade NAT, we develop a second heuristic that analyzes the remote IP addresses of all traffic flows classified as local. When we observe that the remote IP addresses fall in more than five different subnets, we compute the number of connections and bytes for each remote /24 to identify whether there is a "preferred subnet", i.e., a remote subnet that carries most of the traffic (>99.9 %). If there is a preferred subnet, then we leave all traffic classified as local. Otherwise, we flag the network for manual inspection. The HostView data had a total of five home networks which contacted more than five different remote subnets, four of these had a preferred subnet. We manually inspected the remaining home network and found that a large fraction of P2P traffic going to IPs in `10.*` networks. In fact, this user's home ISP is known deploy carrier-grade NAT, so we label this `10.*` traffic as wide-area and we leave the `192.*` traffic as local. For work networks, we might misclassify local traffic as wide-area when hosts connected to the local network have public IP addresses. We address this issue with a third heuristic that labels all traffic to a destination IP address that has the exact same organization name as that of the source network as local. Finally, we classify all broadcast traffic as local. We label all the remaining traffic as *wide-area*.

Extension of network environment labels As discussed in Section 2, some of the HostView traces have no network type tag (e. g., Home or Work). We manually inspect the ISP, the network interface, and the geo-location of each unlabeled trace and assign a label. For example, we label a trace annotated with *ISP: "University of California";*

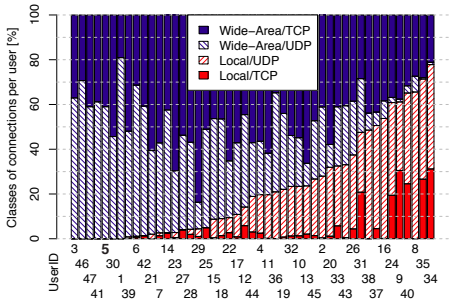


Fig. 1. Local vs. wide-area connections per user (Total number of connections per user varies between 2.5 K and 3 M.).

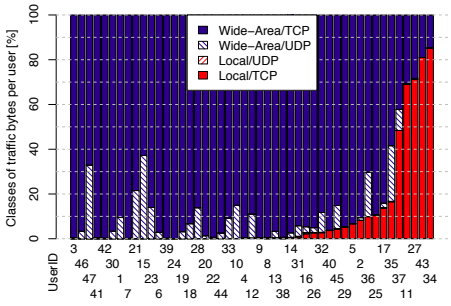


Fig. 2. Bytes transferred on local vs. wide-area connections per user (Total amount of traffic per user varies between 800 MB and 770 GB).

City: "Santa Cruz, California"; Country: "United States" as *Work*. Another example containing *ISP: "Free"; City: "Paris"; Country: "France"* is labeled *Home*. This manual classification reduced the fraction of unlabeled traces to 2 %. Some traces have no information that indicates the type of network.

Application Categorization For our analysis of popular applications we rely on a two-staged categorization process. First, we assign one of eleven application categories or "unclassified" to each connection based on the process executable name. Second, we label any connection that remains unclassified based on the application protocol as derived from the port number using the IANA mapping. We assign categories to those process names and application protocols that account for the most connections and the most volume. Table 1 lists the eleven categories and gives example process names and application protocols for each of them.

4 Results

This section first compares local and wide-area traffic in general. Then, it studies the split of local and wide-area traffic at home and at work.

Local vs. Internet: Connection and Bytes Figures 1 and 2 show the fraction of local (two bottom bars) and wide-area (two top bars) traffic for each user (UserIDs are the same across figures for comparison). For each user, we separate UDP (shaded bars) from TCP (solid bars) traffic. We consider the composition of traffic by number of connections (Figure 1) and bytes (Figure 2).

Take the example of the rightmost user in Figure 1, UserID *34*, 77 % (46 % UDP and 31 % TCP) of this user's connections are local. The remaining traffic is directed to the Internet (0 % UDP and 23 % TCP). In general, we observe that Internet traffic dominates both in number of connections and bytes, although this dominance is much more pronounced for bytes. In total, we classify 780 GB as local and 3 TB as wide-area traffic. Furthermore, we see that UDP dominates local connections for almost 80 % of the users. The absence of shaded bars in Figure 2 clearly shows that almost all bytes are transferred in TCP connections (>89 %).

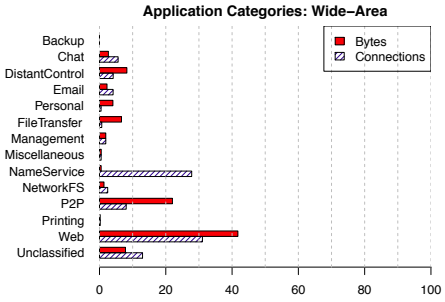


Fig. 3. Application mix for wide-area traffic.

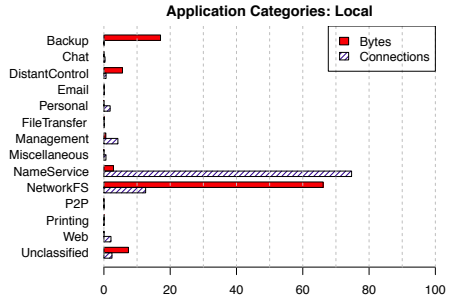


Fig. 4. Application mix for local traffic.

The four rightmost users in Figure 2 transfer more bytes locally than in the wide-area. As we discuss in the next section, most of this traffic corresponds to network file system, so these users could be playing music or watching videos from a local network storage. In Figure 2, more than half of the users exchange almost all traffic with hosts in the wide-area (corroborating previous findings [5]). In the rare cases these users do exchange traffic with hosts in the local network, they mainly perform file transfers.

Local vs. Internet: Application Mix We now study how local and wide-area applications differ. Figures 3 and 4 show the application mix in terms of connections (shaded bars) and data bytes (solid bars). These figures use the application categorization method described in Section 3, which leaves no more than 12 % of connections and 7 % of bytes *unclassified*.

Figure 3 shows the application mix for wide-area traffic. We see that the proportion of bytes per application class agrees with results from previous studies [6,7]. Web traffic and P2P are the top applications. In addition, we see some file transfers and distant control traffic (ssh and VNC). When we classify in terms of number of connections, the mix changes and name services take the second place behind Web. Chat and Email are also more prevalent in terms of connections than bytes.

Figure 4 shows that name services (e. g., DNS) dominates local traffic in terms of connections, whereas backup and network file systems (e. g., AFP and SMB) in terms of bytes. A previous study of enterprise traffic [9] also found that network file system and name service dominate local traffic, but their study found considerably more local email and web traffic than what we find. A significant part of our data is of home traffic, which may explain this difference. We now split the traffic into home and work.

Traffic at Home and Work Our analysis so far has mixed traffic from multiple network environments, including home, work, airports, coffee shops, or hotels. Based on our extended environment labels (see Section 3) we investigate the differences not only between local and wide-area traffic, but also across different types of network environments. Figure 5 shows the distribution of traffic and users over the different environments. Note that a single user can visit multiple environments. After applying our heuristics the 'Other' category, which includes instances when users labeled the environment as other and when our heuristic could not label the environment, only accounts for 12 % of the bytes and 18 % of the connections. We see that users (light shaded bars)

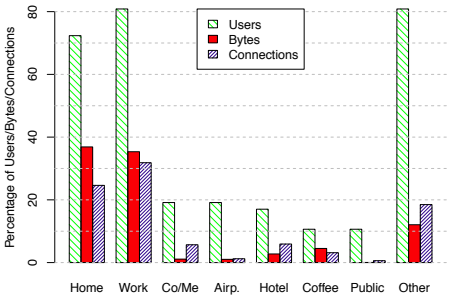


Fig. 5. Percentage of Users, volume, and connections by environment.

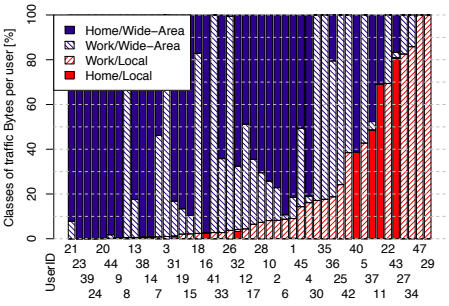


Fig. 6. Bytes transferred at home vs. work and traffic target per user.

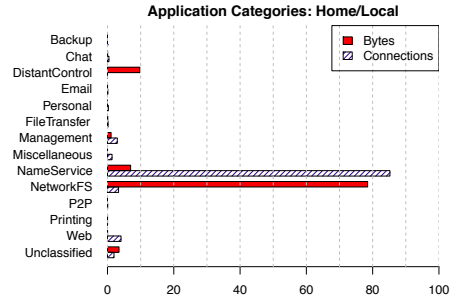


Fig. 7. Application Mix for Home/Local traffic.

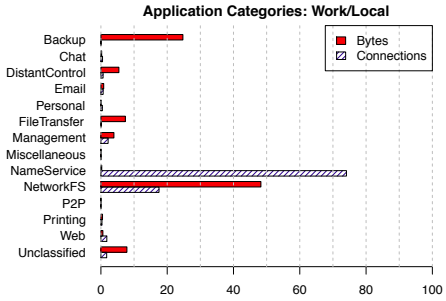


Fig. 8. Application Mix for Work/Local traffic.

are primarily at home or work, thus we select these two environments for further study. These environments include 56 % of the connections (heavy shaded bars) and 72 % of the bytes (solid bars). Moreover, our analysis of local traffic in different environments (not shown) shows that the fraction of local traffic in all environments but home and work is marginal ($<$1.25 %).

Figure 6 shows the number of bytes sent and received per user for all four combinations: home/wide-area, work/wide-area, work/local, and home/local. As expected, we see a similar split between local (bottom) and wide-area (top) traffic The differences between Figure 6 and Figure 2 happen because here we only include traffic from home and work. The majority of users has more local traffic at work. Only four users have a significant fraction of local traffic at home.

Application Mix at Home and Work Now that we established a basic understanding of how traffic differs between home and work as well as local and wide-area, we investigate the application mix in each of these cases. The analysis of wide-area traffic at work (omitted for conciseness) shows almost no P2P traffic, but a considerable fraction of file transfers and distant control traffic. These results are consistent with previous findings by Pang et al. [9].

We study the application mix of local traffic at home in Figure 7 and at work in Figure 8. Local traffic at work includes file transfers and backup traffic, which are not present in home traffic. Different from Pang et al. [9], we see little local email or web traffic at work. Indeed, it turns out that email traffic of most HostView users is wide-

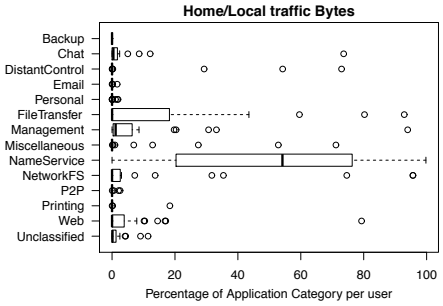


Fig. 9. Boxplot of application mix per user for Home/Local traffic.

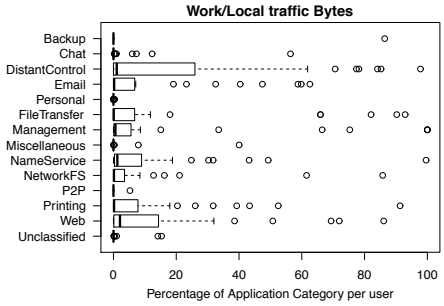


Fig. 10. Boxplot of application mix per user for Work/Local traffic.

area. A possible explanation is that they are typically mobile and hence rely less on local infrastructure.

Another difference is the lack of backup traffic at home, which may reflect users' preference to backup directly at external disks when at home, instead of over the network. The backup traffic at work is mainly from a single user, who is responsible for almost all the bytes of backup traffic in Figure 8. We do also observe some file transfer traffic locally at work. Most of that is transmit (file transfer client for Mac OS) and FTP, but some is Dropbox (a cloud storage/synchronization service). Given it is a cloud service (cloud = wide-area) we did not expect to find Dropbox locally. It turns out that Dropbox is using a direct connection for synchronization across devices in the same LAN. Dropbox constitutes half of the file transfers in our local home traces.

As single users can have a distorting impact on the overall traffic composition, we now calculate the application mix per user. Figures 9 and 10 show boxplots[3] of the application mix per user in terms of bytes. Each row shows the distribution of the individual contribution of the corresponding application category across all users. We find that although network file system traffic dominates local traffic, most users have less than 10 % of traffic in this category both at home and at work. Reversely, although name service represents a small percent of the total number of bytes in Figure 7, the median across all users is over 50 %. We find similar effects for file transfers at home. At work, contrary to Figure 8, we do see web, email, and printing usage.

Connection size and duration We end our analysis with a study of the characteristics of local and wide-area connections both at home and work. We show the complimentary cumulative distribution of the number of bytes per connection in Figure 11 and connection durations in Figure 12. For example, the 'work/local' point at x = 10kB in Figure 11 indicates that only 1 % (y-axis) of all the connections are larger that 10kB.

In terms of bytes, we observe in general larger (further to the right) connections for wide-area traffic. Local connections are typically small, but the largest local connections exceed the size and duration of wide-area connections. This observation confirms one previous study showing that home traffic sometimes have short spikes [5]. Although

[3] The box (line inside the box) shows the quartiles (median); whiskers show nearest values not beyond a standard span from the quartiles; points beyond (outliers) are drawn individually.

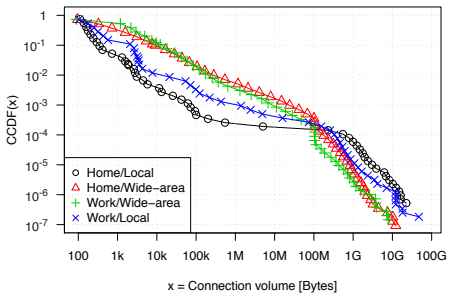


Fig. 11. CCDF of connection volumes.

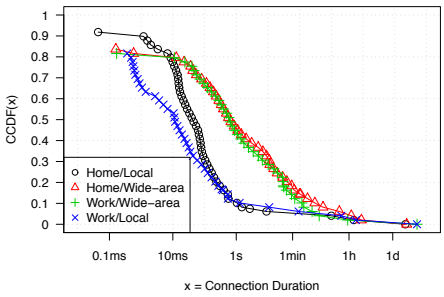


Fig. 12. CCDF of conn. durations (log-linear)

the connection durations in Figure 12 are limited by the 4 hour trace file cutoff, most connections are shorter than this limit. We also see the local connections (circles and crosses) are up to two orders of magnitude shorter than wide-area connections.

5 Related work

Wide-area traffic measured from *inside* the network has been analyzed from different angles over the past decades [1,6,7,12]. These measurements, however, cannot capture local traffic in networks at the edge. Our study analyzes local traffic and how it compares with wide-area traffic with data collected directly at end-hosts using HostView [4]. Other studies have collected and analyzed similar end-host data in the past [2,11]. In particular, Giroire et al. [2] has compared network traffic from end-hosts across three network environments (inside the company, VPN to company, and outside the company). Different from ours, their study has not characterized local network traffic in depth and although it measured laptops of a larger number of users than HostView measured, they are all employees of a single enterprise.

Most similar to our work are the studies of one enterprise network [8,9] and of three home networks [5]. These prior studies instrument the local network to collect packet traces and can hence observe most local and wide-area traffic. Our study measures one (or at most a couple) of end-host in each network and hence cannot have such a complete view of each of the studied networks, but it can sample a larger number of networks. The home network study focuses mainly on network performance, not on traffic characterization. Their few traffic characterization results show that wide-area traffic dominates local traffic in the three homes, but that there are some, rare spikes of local traffic. The analysis in the enterprise study [9] is most similar to ours and we contrasted their findings with ours throughout this paper. Given that Internet traffic can vary significantly among sites and over time [10], our study contributes to show the diversity of traffic patterns in different network environments.

6 Summary

This paper presented a comparison of local traffic in different network environments from the perspective of end-hosts. The advantage of using end-hosts as vantage points is that we study traffic collected from over one hundred different edge networks. Our

results showed that there is a large diversity in importance of local traffic relative to wide-area traffic, but that in general wide-area traffic dominates. In some networks (like airports and coffee-shops), we rarely see any local traffic, the only local traffic is DNS. At home and work, we do observe a non-negligible fraction of local traffic. Most local traffic is composed by short connections, but sometimes local connections transfer an extremely large number of bytes. Besides DNS, the most typical local applications are network file system and backup, but the composition of local traffic depends on the user and the network. The drawback of measuring local traffic from end-hosts is that we can only see a small fraction of each network's traffic. In the future, we plan to collect data directly from home gateways to measure all traffic from a single home over a longer period of time. In fact, home users are already deploying home gateways modified to perform measurements. We are working with the developers of Bismark (http://projectbismark.net/) to collect passive traffic measurements as well.

Acknowledgments

We thank D. Joumblatt and O. Goga for their help with the HostView data. This work was supported by the European Community's Seventh Framework Programme (FP7/2007-2013) no. 258378 (FIGARO) and carried out at LINCS (www.lincs.fr).

References

1. CÁCERES, R., DANZIG, P. B., JAMIN, S., AND MITZEL, D. J. Characteristics of wide-area TCP/IP conversations. In *Proc. ACM SIGCOMM* (1991), pp. 101–112.
2. GIROIRE, F., CHANDRASHEKAR, J., IANNACCONE, G., PAPAGIANNAKI, K., SCHOOLER, E. M., AND TAFT, N. The cubicle vs. the coffee shop: behavioral modes in enterprise end-users. In *Proc. PAM* (2008), pp. 202–211.
3. JOUMBLATT, D., GOGA, O., TEIXEIRA, R., CHANDRASHEKAR, J., AND TAFT, N. Characterizing end-host application performance across multiple networking environments. In *Proc. INFOCOM (Mini-Conference)* (2012).
4. JOUMBLATT, D., TEIXEIRA, R., CHANDRASHEKAR, J., AND TAFT, N. Hostview: annotating end-host performance measurements with user feedback. *SIGMETRICS Perform. Eval. Rev. 38* (January 2011), 43–48.
5. KARAGIANNIS, T., CHRISTOS, G., AND KEY, P. Homemaestro: Distributed monitoring and diagnosis of performance anomalies in home networks, Oct. 2008. Tech. Rep. MSR.
6. LABOVITZ, C., IEKEL-JOHNSON, S., MCPHERSON, D., OBERHEIDE, J., AND JAHANIAN, F. Internet inter-domain traffic. In *Proc. ACM SIGCOMM* (2010), pp. 75–86.
7. MAIER, G., FELDMANN, A., PAXSON, V., AND ALLMAN, M. On dominant characteristics of residential broadband internet traffic. In *Proc. ACM IMC* (2009), pp. 90–102.
8. NECHAEV, B., ALLMAN, M., PAXSON, V., AND GURTOV, A. A preliminary analysis of tcp performance in an enterprise network. In *Proc. INM/. WREN'10* (2010), pp. 7–7.
9. PANG, R., ALLMAN, M., BENNETT, M., LEE, J., PAXSON, V., AND TIERNEY, B. A first look at modern enterprise traffic. In *Proc. ACM IMC* (2005).
10. PAXSON, V. Empirically-derived analytic models of wide- area tcp connections. *IEEE/ACM Transactions on Networking 2* (August 1994).
11. SAIKAT, G., CHANDRASHEKAR, J., TAFT, N., AND PAPAGIANNAKI, K. How healthy are today's enterprise networks? In *Proc. IMC* (2008), pp. 145–150.
12. THOMPSON, K., MILLER, G., AND WILDER, R. Wide-area internet traffic patterns and characteristics. *Network, IEEE 11*, 6 (Nov/Dec 1997), 10 –23.