Grant Agreement No.: 258414

SCAMPI

Service platform for social Aware Mobile and Pervasive computing

Instrument: Collaborative Project
Thematic Priority: THEME [ICT-2009.1.6] Future Internet experimental facility and experimentally-driven research

D3.1: Enabling Services in OppNets: main directions

Due date of deliverable: 30/09/2011
Actual submission date: 30/09/2011

Start date of project: October 1st 2010
Duration: 36 months
Project Manager: Professor Jörg Ott
Revision: v.1.0

Abstract:

This document illustrates the main directions pursued by the SCAMPI Consortium toward the goal of enabling services in Opportunistic Networks. The document describes strategies to enable opportunistic communication and to provide services over a challenged network. Further key aspects are described, such as the exploitation of the social structure of the network, the exchange of content, and recommendation as well as privacy-aware schemes.

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	Authors
v.0.1	**20.09.2011**	**First version of deliverable**	**Eleonora Borgia (CNR), Enrico Gregori (CNR), Franck Legendre (ETHZ), Nidhi Hegde (Technicolor), Mikko Pitkänen (AALTO), Daniele Puccinelli (SUPSI)**
v.1.0	**30.09.2011**	**Final version of deliverable**	**Daniele Puccinelli (SUPSI), Mikko Pitkänen (AALTO), Monique Calisti (Martel), Nidhi Hegde (Technicolor)**

Table of Contents

1 INTRODUCTION

The SCAMPI vision is to allow end users to go beyond the set of locally available resources and to pool resources available in the environment. To this end, SCAMPI's opportunistic computing paradigm combines elements of pervasive computing/networking with opportunistic networking.

In SCAMPI vision, the environment is packed with resources that can be exploited, but exploiting them requires coping with an inherently dynamic and highly unstable network. SCAMPI users cooperate to enable shared services that leverage the collective wealth of hardware and software resources. To enable such cooperation, SCAMPI service platform provides services over opportunistic networks (OppNets) that do not require the presence of an infrastructure. In SCAMPI, mobile devices act as both network providers and network users. By aiming for infrastructure-less operation, SCAMPI can cater to an extremely broad user base under all sorts of conditions, including emergencies and disaster scenarios.

WP3 has a central role on enabling shared services, and its key objectives are to make shared services available and secure. The positioning of the WP3 activities with regards to overall SCAMPI architecture is explained in Figure 3 of D1.1: System Architecture Specification, where WP3 activities can be associated to *Services* block. In the overall architecture, the role of WP3 is to provide key supporting services for the service platform and the applications to build their operation on top of. The WP3 works in close interaction with WP2, which provides key parameters and characteristics of the operational environment to be used as input by the algorithms of WP3. Based upon this input and results from the practical experimentation conducted WP4, this work package develops decision making engines that can feed to operation of SCAMPI service composition function as well as to simplified interfaces that can be exposed for application developers to be used. Finally, the developed algorithms serve as an input to continuously evolving system architecture design and development that is carried out in WP1 and that produces actual implementations to be further tested and validated in WP4. Several of the mechanisms discussed in Section 3 are already integrated to service platform and are ready for further experimental evaluation.

The natural approach of making shared services available is to define mechanisms to enable basic service discovery and notification. The network considered within the SCAMPI project, however, is highly dynamic, topologically unstable, and seriously resource constrained. Applicable service discovery and notification schemes are therefore anything but basic, as they must be sufficiently robust to tame the underlying challenges posed by the network. The bottom line remains that services must be provided over a challenged network.

In the SCAMPI scenario, it is certainly not sufficient to make shared services available, unless they are also made secure. In a completely distributed networking environment that potentially lacks any kind of centralized authority, trust is a scarce resource and privacy is a major concern.

The goal of the present document is to illustrate the main avenues of research that have been pursued in the course of Year 1 by the SCAMPI Consortium toward the ambitious goal of enabling services in OppNets. The remainder of the document is organized as follows. We begin by illustrating and discussing strategies for enabling opportunistic communication in Section 2. We delve into the challenges of delivering services over a challenged network in Section 3. The detection and exploitation of the underlying social structure is the subject of Section 4. Finally, Section 5 focuses on content exchange, and Section 6 deals with recommendation systems and privacy preservation. The deliverable is complemented by a set of four appendices, each of which contains a key publication produced by the SCAMPI Consortium.

2 ENABLING OPPORTUNISTIC COMMUNICATION

To ensure the large success of SCAMPI solutions and services, we should make sure off-the-shelf smartphones support opportunistic communications.

Opportunistic networking offers many appealing application perspectives from local social-networking applications to supporting communications in remote areas or in disaster and emergency situations. Yet, despite the increasing penetration of smartphones, opportunistic networking is not feasible with most popular mobile devices. The concept of opportunistic networking and smartphone applications leveraging these concepts are since years confined to research labs and small-scale testbeds. There are several reasons for this: Firstly, there is still no support for seamless and high throughput short to mid-range Ad-Hoc communications in stock phones. WiFi Ad-Hoc is still not supported on most popular phones (unless they are rooted or jailbroken) and will probably not be in the near future, if ever. Secondly, Bluetooth suffers from its limited bandwidth and the cumbersome peering/service detection procedure, not to mention implementation incompatibilities. Thirdly, even if some platforms such as Windows Mobile or Symbian support WiFi Ad-Hoc, these suffer from the lack of a popular application store and low or decreasing market share. Eventually, WiFi Ad-Hoc is very power inefficient, draining the battery in only a few hours. One might claim that WiFi Direct will improve this issue, though nothing guards us from mobile OS restrictions (see iPhone WiFi tethering mode) or similar Bluetooth-like cumbersome pairing procedures.

We believe that applications leveraging opportunistic principles currently face a chicken and egg problem: *As long as opportunistic applications are not popular, smartphone manufactures will not implement the APIs required to setup WiFi Ad-Hoc connections. But as long as application developers do not have APIs to exploit the advantages of opportunistic networks, how can they create such popular applications?*

The SCAMPI consortium has adopted the strategy of designing and implementing a SCAMPI flagship application with the inclusion of a new partner, Futurice, a smartphone application development company. This application should promote SCAMPI and the principles of opportunistic networking. However, this does not guarantee us that high throughput opportunistic communications will be supported by mobile operating systems.

In order to put all the chances on our side, we proposed the following complimentary approach: supporting opportunistic communications over WiFi based on the current hardware and API available on stock devices. We hence propose WiFi-Opp[1], a realistic opportunistic setup relying on (i) open stationary APs and (ii) spontaneous mobile APs (i.e., smartphones in AP or tethering mode), a feature used to share Internet access, which we use to enable opportunistic communications. We compare WiFi-Opp to WiFi Ad-Hoc by replaying real-world contact traces and evaluate their performance in terms of capacity for content dissemination as well as energy consumption.

To summarize, the main contributions are:

- We propose WiFi-Opp, a simple and energy efficient way of enabling opportunistic networks using current smartphones' technology.

[1] WiFi-Opp will probably be renamed to WLAN-OPP because of trademarks issues with the registered trademark "WiFi" (or Wi-Fi). We are about to initiate discussions with the WiFi Alliance to have their consent over the use of their trademark.

- We evaluate the performance and energy efficiency or WiFi-Opp for content dissemination. We show that our approach is up to 10 times more energy efficient than WiFi Ad-Hoc for comparable dissemination performance.

- We present a proof of concept demonstrating WiFi-Opp's feasibility.

- We discuss the implications of becoming a spontaneous mobile AP, the current device heterogeneity in the market and the benefits of WiFi-Opp over WiFi Direct.

We are currently evolving our proof-of-concept to a fully fledge library and application on Android and plan to port it to both the iPhone and Windows Mobile. We are also about to launch a website in order to gather the open source community to help us developing and extending WiFi-Opp.

3 SERVICES OVER A CHALLENGED NETWORK

The ultimate goal is to make it possible for SCAMPI users to achieve their intended goals in spite of the underlying challenges posed by the network.

The network is indeed doubly challenged: its topology is inherently unstable and highly dynamic, and the resources of the individual nodes are greatly constrained. With these premises in mind, resource stability is not to be taken for granted, and techniques to cope with its time-varying behavior must be devised. Section 3.1 illustrates the progress in this direction.

The fundamental building block of service discovery lies at the heart of WP3, and Section 3.2 describes the main directions that have been and are being followed to achieve its definition.

3.1 Coping with Varying Levels of Resource Stability

Resource stability represents SCAMPI's notion of generalized connectivity. In general, remote resources are available with variable stability levels that fluctuate over time and space and across different devices. An analogy can be drawn to wireless connectivity, which also varies over time and comes in various flavors and degrees of robustness.

We can think of the network topology, which can be characterized in terms of its time-varying physical connectivity, and a concept of soft connectivity, which quantifies how well nodes are able to detect one another. Similarly, we can think of a resource topology, which can be characterized in terms of its time-varying resource availability, and a concept of resource stability, which quantifies how well nodes are able to access resources.

To tackle the understanding of time-varying resource stability, we begin by analyzing the often elusive interplay between resource stability and energy consumption.

Nodes that provide stable resources may be energy-challenged. In this case, their energy budget can only decrease, because such nodes are guaranteed to be popular, and we have a poor gets poorer scenario. Note that such nodes are poor in energy terms, but rich in resource terms. We can also think of this scenario from the point of view of resources (as opposed to energy), in which case we find that the resource-rich nodes get overexploited. Intuitively, the resource-rich pays taxes that are unfairly high.

To begin to understand the implications of the interplay between resource stability and energy consumption, we focuses on the most basic as well as elusive of all SCAMPI resources, wireless connectivity, and we investigated the tradeoff between energy consumption and wireless connectivity.

Our publication [1] presents an extensive set of experimental results on two remote-access wireless sensor network testbeds that illustrate the interplay between network connectivity and energy consumption. Using collection on top of a duty-cycled link layer, we observe that the energy consumption of unicast traffic is heavily affected by insufficient connectivity conditions. Specifically, the presence of connectivity outliers that require a large number of retransmissions to overcome transitional links has a significant impact on the energy consumption of the whole radio neighborhood of the outliers. At the other end of the connectivity spectrum, high connectivity may have a significant impact on the energy consumption of broadcast traffic, because it is conducive to contention and overhearing. To enable fair comparisons between experimental runs, we augment our results with quantitative data regarding the network topology during each run.

3.2 Service Discovery

Discovery mechanisms have a critical role in enabling opportunistic communications. We can divide discovery into two distinct parts: 1) discovery of nodes that are within communication range, and 2) discovery of the services provided by those nodes. The latter can be further divided into discovering the application level services (e.g., this node provides video streams) and SCAMPI middleware level services (e.g., this node provides bundle forwarding service) provided by the node.

Discovery mechanisms can be implemented at various layers of the network stack and may require different characteristics from the underlying network. For example, Bluetooth specifies its own mechanisms for scanning for surrounding nodes (which results in the discovery of hardware addresses) and for directly querying other nodes for services they provide. On the other hand, mechanisms such as mDNS/DNS-SD (see 3.2.1) can be used on any IP multicast network.

Beyond the discovery of nodes and services within the immediate network (i.e., nodes to which direct connection can be opened) the SCAMPI middleware should be able to discover services offered by nodes that can only be reached via store-carry-forward messaging. This can be achieved by a dedicated discovery mechanism built on top of such a network (e.g., DTN bundle layer discovery) or by transitively aggregating direct discovery information. The following reports several discovery mechanisms that we have tested and evaluated as a part of existing software implementation that works as a standalone Java application in laptops and desktops as well as a software library integrated to an application that runs on an actual Android phone.

3.2.1 mDNS/DNS-SD

The combination of multicast DNS (mDNS) and DNS Service Discovery (DNS-SD) is familiarly known as Apple's Bonjour service discovery. Multicast DNS defines mechanisms for performing DNS-like operations over a multicast network, while DNS-SD defines mechanisms for using DNS messages for service discovery.

This approach to service discovery is designed for largely fixed networks where services are assumed to be relatively long-lived, for example, for finding printers or file servers. These assumptions are codified in the protocol design in the form of various timers for mDNS mechanisms and pre-defined lifetimes for DNS-SD records.

3.2.2 DTN IP Neighbor Discovery

DTN IP Neighbor Discovery (draft-irtf-dtnrg-ipnd-01) defines a mechanism for discovering DTN Bundle Agents and services offered by them over IP-multicast networks. It is essentially a more lightweight (and limited) version of mDNS/DNS-SD optimized for networks where contacts are assumed to be infrequent and short-lived.

Multiple implementations exist for the protocol.

3.2.3 Bluetooth Discovery

Bluetooth specification defines a Service Discovery Protocol (SDP), which can be used by Bluetooth devices to discover what services are offered by other devices. The mechanism is based on searching for and browsing specific Universally Unique Identifiers (UUID) which map to different services.

Support for SDP is implemented in the Bluetooth hardware and should be accessible to any application on the device through Bluetooth APIs if such exist on the platform.

3.2.4 Multi-hop Service Discovery

There are currently no standardized protocols for service discovery in multi-hop, store-carry-forward networks. Such a protocol is being designed within SCAMPI, and is currently part of the ongoing design and development process carried out in WP1.

3.2.5 SCAMPI Discovery Solution

The approach taken in the SCAMPI middleware allows arbitrary discovery mechanisms to be used. This is done by defining a framework that makes minimal assumptions about the capabilities of the underlying networks. For example, node discovery only assumes that the discovery mechanism is able to report a unique name for discovered nodes that can be used later to open connection to the nodes (e.g., ability report hardware addresses). Such a discovery mechanism in its simplest form could be a list of pre-configured IP addresses, it could be a fully passive mechanism that discovers nodes by listening to the link layer messaging, or an IP-layer multicast/broadcast based query-response discovery mechanism.

The framework includes three parts: 1) peer discovery for finding nodes within direct communication range, 2) peer service discovery for discovering services offered by nodes within direct communication range, and 3) multi-hop service discovery for finding services offered by nodes that are not within direct communication range, but which can be reached through store-carry-forward messaging. Each part can be implemented autonomously (e.g., generic one-to-one query mechanism for service discovery) or as a combination (e.g., Bonjour discovery that discovers peers and the services offered by those peers).

At first, existing protocols will be used for implementing service discovery, but ultimately a new solution can be designed that is specifically tailored and optimized for service and node discovery in urban ad-hoc networks.

4 LEVERAGING SOCIAL STRUCTURE

A key strategy to cope with and tame the unprecedented networking challenges that arise within SCAMPI is to leverage the underlying social structure of the network. Section 4.1 provides details on this promising approach, while Section 4.2 illustrates our work in progress on community-based spreading.

4.1 Detecting Social Structure

Approaches aimed at discovering dynamic communities of people are fundamental in the context of opportunistic scenarios. For example, social information may be exploited to design smart strategies to choose the most suitable next hop in forwarding schemes or to efficiently disseminate content. People are social individuals that have social ties and form communities to better answer to their needs. Those who belong to the same community meet with high probability and regularly. On the contrary, people from different communities meet less frequently. Therefore, understanding the human structure and the rules which regulate social interactions and aggregations can be a great advantage in the reference SCAMPI scenario.

The objective of this activity is therefore to investigate on-line solutions able to identify communities within a mobile social network. Specifically, we are interested in carefully examining existing algorithms and in proposing new community detection algorithms able to capture the evolution of social communities in dynamic scenarios. With respect to the overall SCAMPI concept, community detection algorithms will serve as enabling services for the operation of the SCAMPI platform. On-line algorithms describing how users can be grouped in social communities can be used, for example, to predict which service components will be available to a tagged user, their level of stability, the amount of replication, etc. All these information can be used to plan how to opportunistically compose the services required by the applications.

The complete description of the activity is available in [2]. Part of this work has also been published in [3]. In the following we summarize the main findings. Specifically, we describe the main lines of reasoning behind the three reference community detection algorithms, while we go deep into our proposed solution.

4.1.1 SIMPLE, k-CLIQUE and MODULARITY

The study started from the analysis of three reference community detection algorithms, i.e., SIMPLE, k-CLIQUE, MODULARITY. The idea at the basis is that each node determines the composition of its community based on the contacts he has with other nodes. This is achieved by storing some information which are then exchanged by nodes during the meetings. Upon receiving such information, each node makes its decisions independently for including or not the encountered node into its social community. Specifically, a node's social community, here referred to as *Local Community*, includes all those encountered nodes with which that node has spent a sufficient amount of time together (long contact duration) or has met regularly (large number of contact). The *threshold criteria* provides a rule to evaluate the cumulative contact duration, i.e, the sum of the duration of all the different contacts for a pair of nodes, while the *admission criteria* provides a way to measure how often a node is encountered by the community. Finally, in case of high similarity among different communities, the *merging criteria* gives a method to partially or completely merge communities. Among the above criteria, the former is the only one which is common to all the three algorithms. On the contrary, the other two (i.e., admission and merging) are specific for each algorithm. In addition, the three algorithms differ for the complexity and the richness of features they offer. Specifically, as the name suggests, SIMPLE is the simplest solution as it has a low memory usage and low

computation cost, while MODULARITY represents the opposite extreme with its huge memory requirement and computation cost. k-CLIQUE represents an intermediate solution since it needs the same memory usage of MODULARITY but has a lower computation complexity.

In this section we present a subset of the performance results of the above community detection algorithms. A more extended set of results is available in [2]. We mainly focused at testing their ability at identifying the social network community when the underlying moves in different patterns of mobility. For this purpose, we considered a mobility model (HCMM) which well reproduces the natural patterns of users within a social network. For example, HCMM is able to correctly reproduce the statistical figures associated to the user patterns, such as inter-contact times and durations of contact. The performance evaluation is mainly based on similarity feature, which is measured by the Jaccard index, $\sigma_{Jaccard}$, as explained by the following expression:

$$\sigma_{Jaccard} = \frac{\left| G_i^{(d)} \cap G_i^{(c)} \right|}{\left| G_i^{(d)} \cup G_i^{(c)} \right|} \qquad (1)$$

Basically, $\sigma_{Jaccard}$ evaluates the similarity of the communities detected by the community detection algorithm with respect to those detected by the centralized version of the same algorithm which has the complete knowledge of the network.

Figure 1 shows a comparison of the similarity index among the three algorithms. What is important to highlight here is that, with a correct choice of the algorithm parameters, there exists a good match between the communities detected by distributed algorithms and the corresponding ones found by the centralized version. For example, by reducing the threshold value, the probability that the cumulative contact duration for pairs of nodes satisfies the *threshold criteria* increases. In addition, a fine tuning of this parameter should consider also the dynamic of communities, i.e., how much time (on average) people spend together. A second important result to highlight is that SIMPLE has equivalent if not better performance than the other two approaches, although it has fewer amount of available information which are exchanged by nodes. The performance gain reaches up to 25%-30%. This result is true for all the scenarios we have investigated.

Summarizing, from this performance analysis, we have identified SIMPLE to be the more appropriate solution for detecting communities of people as it has a low resource consumption, it is computationally lightweight (i.e., $O(n)$ vs $O(n^2)$) and it performs better than the other two approaches.

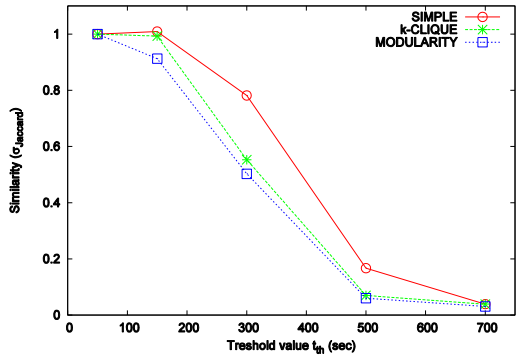


Figure 1. Similarity comparison as function of threshold value.

4.1.2 Adaptive Detection SIMPLE (AD-SIMPLE)

Although the three community-detection algorithms are able to identify the social community in most scenarios, but we have highlighted a major common limitation. When the scenario becomes more complex and dynamic (e.g., people belong to more than one communities simultaneously and spend much time in them), all three algorithms fail: nodes no longer have a consistent vision of the true composition of the community, mainly because they keep memory of all the nodes encountered so far.

To overcome the above issue, we proposed the Adaptive Detection SIMPLE (AD-SIMPLE), a novel and computationally lightweight social community detection algorithm which works efficiently also in dynamic scenarios. Basically, AD-SIMPLE maintains the basic features of SIMPLE, such as the same amount of information exchanged by nodes or the same three criteria for including nodes in communities. However, it adds some policies to identify and remove those nodes seen in the past but that are not seen anymore with the same frequency.

A first policy is related to maintaining always updated the Familiar Set of a node. Note that a Familiar Set of a node v_0 is defined to be the set composed by all the encountered nodes for which the threshold criteria holds true, hence constituting a subset of the Local Community. Specifically, a running average of the percentage of the total contact duration (i.e., $EstimatedΔT$) is maintained for each node within the Familiar Set and whenever this percentage falls below a predefined threshold (i.e., $FSout_{th}$), the corresponding node is removed from it. To this aim, time is divided into slots (i.e., T) and an esteem of percentage of the contact duration for each slot time is computed for all the nodes within the Familiar Set. Then, the running average value is defined as follows:

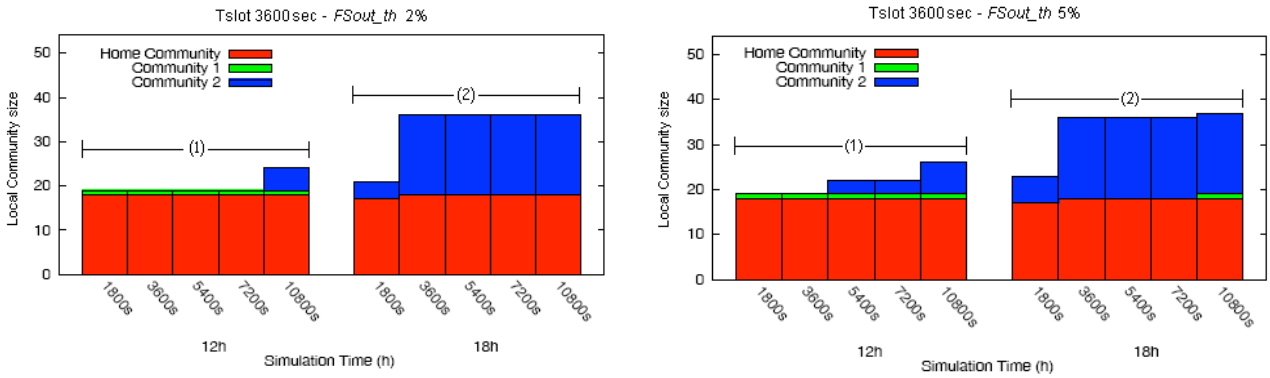
$$EstimatedΔT = \alpha \cdot EstimatedΔT + (1- \alpha) \cdot Sample\ ΔT \qquad (2)$$

The second policy is connected with keeping updated the Local Community of a node. Specifically, a timer (i.e, $LCout_{timer}$) is associated to each node within the Local Community and whenever it expires, the corresponding node is removed from it. In addition, every time two nodes meet directly (e.g., v_0 encounters v_i) or indirectly (i.e., v_i is stored in the Local Community of a third node v_j which is met by v_0), the corresponding timer is refreshed. The main reason behind a refresh of the timer is to maintain in the Local Community all those nodes which are met regularly.

As first analysis, we evaluated the sensitiveness of AD-SIMPLE on the parameters used by the above policies. In the following we focus on the discussion about parameters related to Local Community pruning policy. Figure 2 provides relevant examples of different behavior that can be obtained if such

parameters are not correctly tuned. Specifically, we refer to a dynamic scenario where nodes may be connected to two communities simultaneously, i.e., home and foreign, and move between them for an entire reconfiguration interval. At each reconfiguration, such nodes select randomly one foreign community to start visiting. Each bar represents a snapshot of the Local Community composition for a traveler at the end of two reconfiguration intervals when varying the $FSout_{th}$ and $LCout_{timer}$. Note that the number in brackets above the bars indicates the foreign community visited by the traveler during that period of time.

By fixing T (3600sec in this example), $LCout_{timer}$ (on the *x*-axis) must be of the same order. In fact, if $LCout_{timer}$ is set too high (e.g., 10800 sec), nodes of a community previously visited may still stored in the Local Community. This is apparent for example in Figure 2a at 12h, where some nodes of community 2 are present in the Local Community or in Figure 2b at 18h, where nodes of community 1 are still present. Furthermore, $LCout_{timer}$ must not set less than T, as this could lead to a too rapid emptying of the Local Community (i.e., timer expires too rapidly). For example, in Figure 2a at 18h, if, only a small percentage of nodes of community 2 are present in the Local Community in case of $LCout_{timer}$ equal to 1800 sec. This means that the system has not a sufficient time to detect all the nodes which belong to the communities. Finally, note that $LCout_{timer}$ is in part influenced by the value of $FSout_{th}$ as the Familiar Set represents a subset of the Local Community. To be precise, $LCout_{timer}$ is also influenced by the value of α. From this analysis, we points out that it is important to properly tune $LCout_{timer}$ in order to maintain a consistent view of the social community at each point in time and a reasonable value is by setting $LCout_{timer}$ = T.



(a) (b)

Figure 2. Representative cases of the composition of the Local Community for AD-SIMPLE as a function of $FSout_{th}$ and $LCout_{timer}$.

Figure 3 shows an example of the obtained results of a comparison between SIMPLE and AD-SIMPLE in terms of composition of the Local Community. Further results of such analysis are available in [2]. The limitation of SIMPLE is apparent: after each reconfiguration interval, the traveler adds continuously new nodes to its community, while it maintains memory of all the past encountered nodes. As a consequence, the community size reaches the maximum value after some time and maintains the maximum size for the rest of the simulation. On the contrary, AD-SIMPLE is able to keep always updated the community view of the traveler. Thanks to the two added policies, the traveler maintains only nodes belonging to the two communities which are currently visiting, while it removes those nodes which are not part of its community anymore.

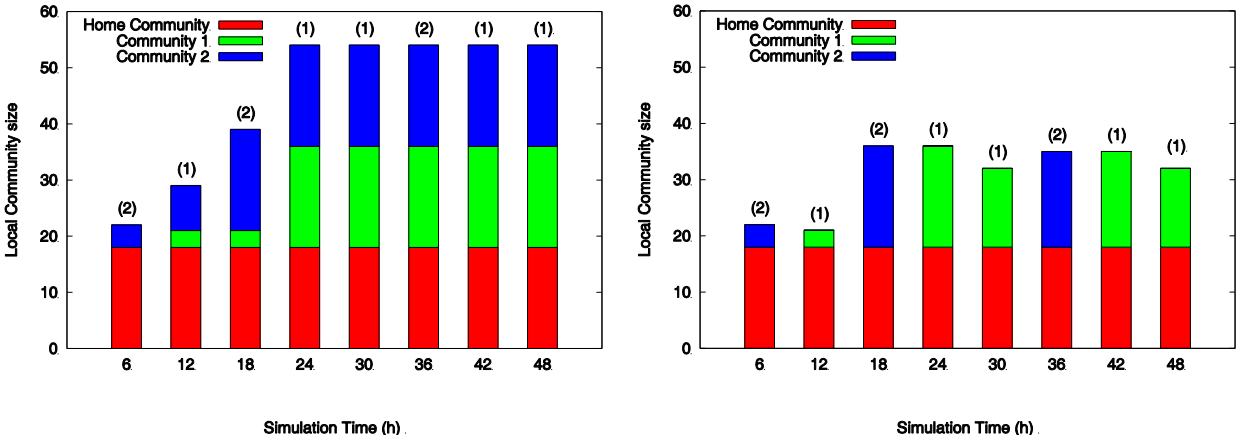


Figure 3. Local Community composition for the traveler for SIMPLE (a) and AD-SIMPLE (b).

Finally, results confirm that AD-SIMPLE exceeds SIMPLE from point of view of the similarity index. Specifically, for SIMPLE, the similarity values fall in the range [0.477-0.66] with a peak of 0.923 at 18h. On the contrary, for AD-SIMPLE, the similarity values are in the range [0.58-1] and they are always higher in each point of reconfiguration interval.

Summarizing, by an extensive set of simulations, we have demonstrated that AD-SIMPLE accurately detects communities and social changes in dynamics scenarios. At the same time, it maintains computation and storage requirements low resulting in a very attractive community detection algorithm for Opportunistic Networks.

4.2 Community-Based Spreading

The publish/subscribe paradigm is particularly suited for content dissemination over opportunistic networks as it automates content exchange whenever peers come into wireless contact. However, the contact patterns driven by human mobility allows for epidemic spreading of content, which a malicious node could exploit to disseminate spam.

In this work in progress, we investigate how social communities can help securing content distribution in opportunistic networks by preventing the dissemination of spam. We propose a community-based spreading (CBS) mechanism where social community members exchange white- and blacklists to promote or block the spread of content, respectively. CBS is scalable as whitelisting is generated implicitly whenever a content is consumed by a user (e.g., watched) and no black-listing follows. Besides, the spread of blacklists is constrained to communities thus, blocking spam at the source and avoiding to flood the entire network. We prove that with CBS legitimate content spreads with a O(1) delay i.e., as fast as basic epidemic spreading, without requiring any explicit user intervention. We model and evaluate our approach with stochastic reaction models and confirm results by replaying real-world mobility traces. We show that (i) a centralized anti-spam authority can never reach the performance of CBS and (ii) that the best spamming strategy is to send only one spam (as opposed to flooding) thus reducing its overhead.

CBS is not a recommendation system in a strict sense as ratings resemble more an objective assessment (i.e., legitimate or not) while usual recommendation systems relate to subjective opinions (i.e. like or not). As a matter of fact, CBS is complementary to the latter. It solves the cold start

problem of author-based recommendations systems or how to deal with new authors. CBS is currently under submission.

5 CONTENT EXCHANGE SCHEMES

In Deliverable D2.1 (Section 7.5), we described a model for content exchange among peers in a mobile opportunistic network. In that work, we studied a static-cache policy in a universal-swarm scenario. We now turn to incentive mechanisms that encourage users to store content that is not directly useful to them, but might be of interest of mobile peers they meet through contacts. Our analysis in Deliverable D2.1 has identified the optimal stationary points that minimize the average sojourn time. However, we have not described a method for leading the system to such points. In this section, we present BARON to bridge this gap. BARON dictates how peers should exchange content items so that the system converges to optimal points. We also demonstrate BARON's performance using numerical simulations.

We first review relevant aspects of the model here, then describe our content exchange mechanism, BARON.

5.1.1 Model

We consider a universal swarm in which content items belonging to a set $\mathbb{K}$, where $|\mathbb{K}| = K$, are shared among transient peers. Each peer arrives with a request $i \in \mathbb{K}$ and a cache of items $f \in \mathbb{K}$, where $C = |f|$ is the capacity of the cache. We denote by $\mathbb{F}=\{f \subset \mathbb{K}:|f|=C\}$ the set of all possible contents of a peer's cache.

A peer swarm consists of all peers interested in retrieving the same item $i \in \mathbb{K}$. We partition the peers in the system into classes according to both (a) the item they request and (b) the content in their cache. That is, each pair $(i,f) \in \mathbb{C} = \mathbb{K} \times \mathbb{F}$ defines a distinct peer class.

Peers requesting item $i \in \mathbb{K}$ and storing $f \in \mathbb{F}$ arrive according to a Poisson process with rate $\lambda_{i,f}$, and we assume that arrivals across different classes are independent. By definition, $\lambda_{i,f}=0$ if $i \in f$. We denote by $\lambda=\sum_{(i,f)\in C}\lambda_{i,f}$ the aggregate arrival rate of peers in the system.

Contact Process. Opportunities to exchange items among peers occur when two peers come into contact. A contact indicates that two mobile peers are within each other's transmission range. Formally, if $N(t)$ is the total number of peers in the system at time t, then a given peer a present in the system contacts other peers according to a non-homogeneous Poisson process with rate $\mu(N(t))1-\beta$, $\beta \in [0,2]$. The peer with which peer a comes into contact is selected uniformly at random from the $N(t)$ peers currently present in the system. Moreover, the above contact processes are independent across peers. The parameter β is used to capture different communication scenarios that may arise in a mobile or online network. We classify these below into contact-constrained, constant- bandwidth, and interference-constrained scenarios.

Contact-constrained communication. When $0 \leq \beta < 1$, the contact rate of a peer grows proportionally to the total peer population. This would be the case in a sparse, opportunistic or DTN-like wireless mobile network, where peers are within each other's transmission range very infrequently. In such cases, the bottleneck in data exchanges is determined by how often peers meet. Adding more peers in such an environment can increase the opportunities for contacts between peers. This is reflected in the increase of a peer's contact rate as the population size grows.

Constant-bandwidth communication. When $\beta = 1$ the contact rate of a peer does not depend on the population size. This reflects constant-bandwidth scenarios, where the system population has no effect on the bandwidth capabilities of a peer, and is thus a natural model of an online peer-to-peer network.

Interference-constrained communication. When $\beta \in (1,2]$, the contact rate of a peer decreases as the total peer population grows. This captures a dense wireless network in which peers share a wireless medium to communicate. As the number of peers increases, the wireless interference can become severe, degrading the network throughput. This is reflected in our model by a decrease in successful contact events and, thus, in a peer's contact rate.

If $\beta > 2$, the aggregate contact rate over all peers in the system decreases as the total peer population grows. Assuming constant arrival rates, such a system will be trivially unstable; as such, we do not consider this case.

For simplicity of notation, we allow self-contacts. Contacts are not symmetric; when Alice contacts Bob, Bob does not contact Alice, and vice versa. This, however, is not restrictive: symmetric contacts can be easily represented by appropriately defining symmetric interactions between two peers.

Under the above assumptions, when the system state is N, the aggregate rate with which users from class A contact users from class A' is

$$\mu_{A,A'}(N) = \mu N_A N_{A'} / N^{\beta}, \quad A, A' \in \mathbf{C}$$

We call $\mu_{A,A'}$ the inter-contact rate between A and A'.

5.1.2 BARON: Guiding cache replacement via valuations

BARON is a centralized scheme. In particular, it requires estimating the demand and supply of each item $i \in \mathbb{K}$, captured by the population of peers requesting and storing i, respectively. In practice, individual peers may maintain estimates of these quantities, e.g., either by gossiping or sampling. However, we leave the study of decentralized schemes for estimating the demand and supply for future work. As a result, we focus on scenarios in which these quantities are readily monitored through at a centralized tracker.

Valuations. BARON keeps track of whether an item is currently over-replicated or under-replicated in following way. In particular, for each content item i, BARON maintains a real-valued variable v_i. We will call this variable the valuation of item i.

Our choice of valuation is inspired by results presented in Deliverable D2.1, Section7.5, which state that there is an optimal point where the supply of an item is C times the demand. Motivated by this, the valuations are given by:

Eq. 1 $$v_i(t) = C n_{i,\cdot}(t) - n_{\cdot,i}(t), \quad i \in \mathbf{K},$$

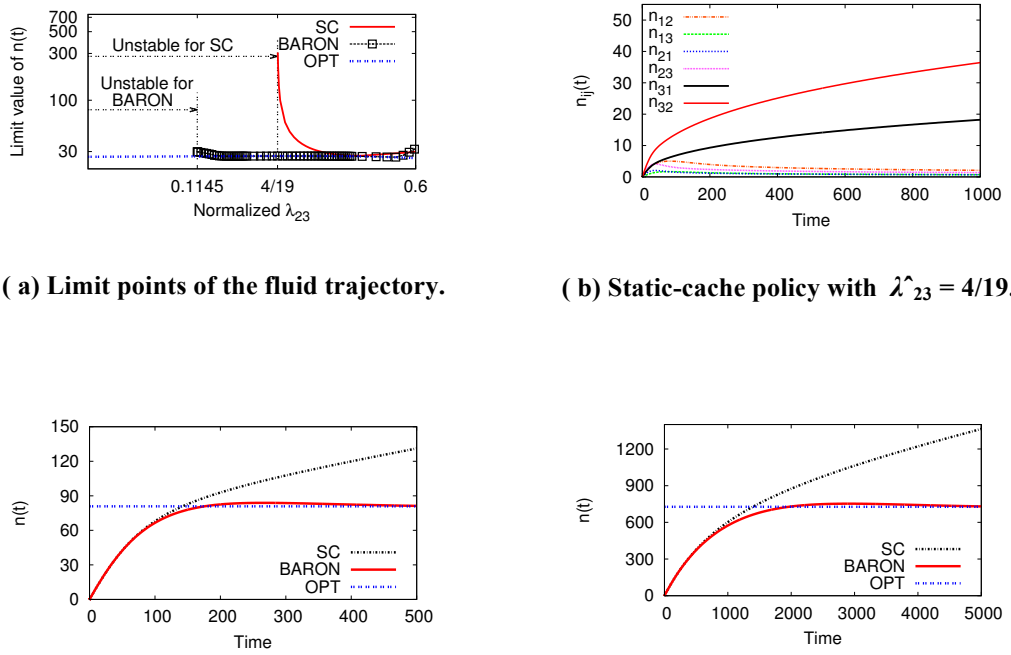
where $n_{i,\cdot}$ and $n_{\cdot,i}$ are the demand and the supply of item I, respectively. A positive valuation $v_i > 0$ indicates that item i is currently under-replicated. Similarly, a negative valuation $v_i < 0$ indicates that item i is currently over-replicated. One appealing property of Eq. 1 is that it requires prior knowledge only of the cache capacity C; in particular, it does not require knowledge of the arrival rates $\lambda_{i,f}$ of each peer class. Nevertheless, this valuation requires tracking of the supply and demand for each item.

Content exchange guided by valuations. BARON is a centralized design that relies on a central controller to maintain the valuations. In addition, this central controller lists the valuations on a public board, and makes them available to all peers. The content exchanges between peers are guided by these valuations following a negative-positive rule. More specifically, during a contact event between a peer A with cache f and a peer B with f', each peer checks if it has any over-replicated items. If so, it

further checks whether the other peer has any under-replicated items that it has not already stored in its cache. If such a pair of items exists, a replacement takes place. In particular, the first peer A replaces the item with the minimal negative valuation in its cache, i.e., peer A removes item i such that $i = \text{argmin}\{v_x | x \in f,\ v_x < 0\}$. Then, among the under-replicated items in the peer B's cache f' yet not in peer A's cache f, peer A replicates the item with the maximal positive valuation, i.e. peer A selects item j such that $j=\text{argmax}\{v_y | y \in f' \backslash f,\ v_y > 0\}$. After retrieving item j from peer B, peer A replaces i with j. Hence its cache f changes to $(f \setminus \{i\}) \cup \{j\}$. A similar procedure follows for peer B.

Evaluation. We evaluate BARON's fluid trajectories using numerical simulations in MATLAB. Our main observation is that, by guiding the content exchanges through valuations, BARON converges to the optimal stationary points, which minimize the average sojourn time. We compare BARON to the static-cache policy by the examining system stability and optimality when using each design.

We simulate the following scenario. Assume there are three items {1,2,3} in the system, and peer's cache size is one. Hence we have six peer classes, where each class of peers requesting item i and caching item j ($\neq i$) has a normalized arrival rate of $\lambda\hat{}_{i,j}$. Peers requesting one item form one swarm, leading to three swarms in total. We set the contact process parameters as $\beta = 0$ and $\mu = 0.002$. We assume initially no peer is in the system. We begin with comparing the system stability under BARON and the static-cache policy. In particular, we aim to understand under which conditions of arrival rates, the system stabilizes when using each design. So we leave $\lambda\hat{}_{23}$ as a free variable, and fix the relative ratios of the other five classes as 1/5, 1/15 , 2/15 , 1/5, 2/2, respectively of $(1 - \lambda\hat{}_{23})$. To identify the system stability for a given $\lambda\hat{}_{23}$ value, we examine the system's fluid trajectory over a significantly long time ($t \approx 105$).



(a) Limit points of the fluid trajectory.

(b) Static-cache policy with $\hat{\lambda}_{23}$ = 4/19.

(c) Contact-constrained communication (β = 0.5).

(d) Constant-bandwidth communication (β = 1).

Figure 4: Comparing BARON and static-cache policy (a and b) and BARON under various β (c and d).

Figure 4(a) shows the rescaled peer population when the system can stabilize as we vary $\hat{\lambda}_{23}$. We see that when using the static-cache policy, the system stabilizes only when $\hat{\lambda}_{23}$ is above 4/19. In contrast, when using BARON, the system has a much larger stability region. More specifically, the system is able to stabilize when $\hat{\lambda}_{23}$ is larger than 0.1145. This demonstrates BARON's effectiveness of guiding content exchanges. Further, BARON is able to guide the system to the optimal stationary state if the system stabilizes, sooner than the static-cache policy. This demonstrates that BARON achieves optimality by the use of valuations. Figure 4 (b) shows the population of each peer class along the time when $\hat{\lambda}_{23}$ = 4/19, demonstrating that only one swarm becomes unstable under the static-cache policy.

To comprehensively understand BARON's performance, we extend to cases with other β values. In particular, we examine two cases with β = 0.5 and β = 1 respectively. As shown in Section 5.1.1 a larger β indicates a smaller contact rate. The case when β = 1 is the constant-bandwidth communication scenario where a peer's contact rate is constant regardless of the peer population. We do not simulate the case where $\beta > 1$, because the optimality result does not hold for such β. Figure 4 (c) and (d) show the evolution of the total number of peers in time. The main observation is that, while the system does not stabilize when using the static-cache policy, the system under BARON converges to the optimal in both cases. This demonstrates the effectiveness of the valuations under various communication settings. Even though the aggregate contact rate decreases as β increases, BARON is still able to adapt the item supply according to the demand, guiding the system towards the optimal.

Furthermore, as the contact rate becomes smaller when β increases, the system with BARON takes longer time to stabilize to the optimal. This is because the item replacement and replication only occur

during contact events. A smaller contact rate slows down the adjustment of the item distribution, leading to a slower convergence.

Further results and detailed derivations of the theoretical results can be found in [4].

6 RECOMMENDATION SCHEMES AND PRIVACY PRESERVATION

Section 6.1 focuses on distributed recommendation, while Section 6.2 illustrates opportunistic privacy preservation.

6.1 Distributed Recommendation

Delivering content to mobile users using a peer-to-peer system is a promising alternative to centralized infrastructure for two reasons. First, it can leverage high bandwidth available locally between the nodes. Second, it presents a unique opportunity to exploit social relationships between users - as manifested through proximity - to discover and promote more relevant content. Unfortunately, these two advantages at first seem contradictory: recommender systems can leverage locality of interest and user profile, but they appear to require either centralized databases and computation, or significant communication exchanges. This has prevented their use in a constrained opportunistic mobile environment.

Here, we prove that these two advantages can be combined. We propose a new distributed rating prediction and recommender system that takes advantage of any peer-to-peer exchange opportunities. This algorithm incorporates users feedback implicitly, and does not require for users to provide interests a priori, or for content to be classified beforehand. We rigorously establish the property of convergence of this algorithm in terms of minimum prediction error and optimal recommendation. Although the algorithm may be used many general ways, we give a specific demonstration using an offline benchmark (i.e. the Netflix training data set) and show that it is competitive with current centralized solutions. We demonstrate its fast convergence in practice by conducting an online experiment with a prototype recommender system on Facebook.

Our proposed design has the following properties:

- It requires minimal feedback from the users. After users view a piece of content, they inform the system whether they liked, disliked or were indifferent to the content they saw. Users viewing content do not need to a priori declare their interests.

- It is fully distributed. All operations are executed in the absence of a central authority, and rely only on information maintained locally at each user.

- It can operate within an opportunistic mobile environment. In particular, all our algorithms work regardless of the type of content available and the frequency with which it arrives at a given user.

- It is adaptive. If a user's interests or mobility change through time, the algorithms employed by our system will adapt to these changes.

6.1.1 System Model

We consider a set $\mathcal{U}$ of mobile users generating and sharing content in an opportunistic manner. A subset $\mathcal{N} \subseteq \mathcal{U}$ of all the mobile users, whom we call *producers*, generate a stream of items that they exchange opportunistically with other users. For example, producers could maintain a blog, a news-feed or a twitter-feed on their mobile devices. Occasionally, a producer may generate new blog entries or new tweets that are added to her locally maintained feed and subsequently shared with other users encountered. We denote by $\mathcal{M} \subseteq \mathcal{U}$ the set of users who receive the content generated and distributed

by producers; we call users in $\mathcal{M}$ *consumers*. Note that $\mathcal{M}$ and $\mathcal{N}$ may intersect, as a user may both produce content and consume content generated by producers.

Our model aims at describing networks with arbitrary connectivity constraints. Hence, we consider a very general process of contacts between producers and consumers. We assume that time is divided into timeslots. Timeslots are indexed by k and they all have duration T. Within timeslot k, we denote the time of the first encounter between a producer $a \in \mathcal{N}$ and a consumer $b \in \mathcal{M}$ by $t_{a\to b}(k) \in [0,T] \cup \{\infty\}$, and by convention $t_{a\to b}(k) = \infty$ denotes that no such encounter takes place. Finally, we denote by $\lambda_{a,b} = \mathrm{P}\left(t_{a\to b} < \infty\right)$ the probability that such an encounter takes place.

We assume independence w.r.t. time; *i.e.*, opportunistic contacts between different timeslots occur independently, even for the same producer-consumer pair. However, we do *not* assume that contacts are independent between different pairs. This is essential because groups of mobile users tend to have correlated behaviors: two consumers Alice and Bob are in a group and hence if Charlie is a producer that meets Alice, he is much more likely to meet Bob in the same timeslot. In addition, contacts created by mobility exhibit locality: if we assume that Alice is both a producer and a consumer, the fact that Charlie meets Alice and that Alice meets Bob in the same timeslot makes it likely that Charlie meets Bob. This latter correlation exists even if Alice and Bob follow independent trajectories.

Producers generate content items that they deliver to consumers whenever they meet. Without loss of generality, we assume that every producer generates a content item at the beginning of each timeslot: the case where a producer skips a timeslot with some probability can be represented by a lack of encounters between this producer and a consumer. After the duration of a timeslot, items expire and are not shared anymore. This corresponds, for example, to blog posts or news items becoming stale.

Although producers may generate diverse content and although consumers' interests may vary, in many applications there exists a small number of characteristics or features that influence how consumers perceive the content. As such, content items should not be treated in isolation, but instead be grouped by similarity with respect to these features into classes or *categories*, that form a partition of the "content universe" of all possible items. These categories are relevant to how producers generate content: a producer may exhibit a bias towards generating content of a certain category. However, content categories may also arise due to shared features that are harder to characterize with a simple label. For example, blog posts may form a category that spans several topics (*e.g.*, news *and* technology) and be additionally defined by their credibility, whether they are humorous or not, *etc.*

We assume the existence of an (undisclosed) set $\mathcal{F}$ whose elements we refer to as categories, such that every content item exchanged generated by our producer belongs to a single category. Moreover, whenever a producer, $a \in \mathcal{N}$, creates an item, we assume that the category the new item belongs to is drawn independently, and is equal to $f \in \mathcal{F}$ with probability $p_{a,f} \geq 0$, where $\sum_{f\in\mathcal{F}} p_{a,f} = 1$. By "independently", we formally mean that the category the item belongs to is independent of (a) the categories of other items generated by producer a in the past and (b) the contact process between producers and consumers.

Although we assume that a set $\mathcal{F}$ of categories exists, we do *not* assume that this set is known, nor *a fortiori* that the mapping of items to categories or the bias exhibited by producers and consumers are known. In particular, producers never have to disclose the category to which the items they generate belong. This paper proves that we can exploit the existence of this structure *even if we do not know the category set or even its size*. The fact that the category remains unknown differentiates this work from

a content-based approach that aims at identifying the set $\mathcal{F}$, which deemed to be more costly and also much more sensitive to users' privacy. Although we assume that content items belong only to one category, this is not restrictive, as our definition of a category is quite abstract: content items belonging to multiple categories ("music" and "sports") can be grouped together to form a new category (the category in the intersection of music and sports).

In general, although the assumption that a set $\mathcal{F}$ exists is hard to validate *per se*, it is commonly used in distributed rating prediction. The empirical results that we obtain justify that this assumption is effective.

We denote by O the set of possible feedback provided by the consumer. We denote an element of this set by o and refer to it as a rating. In general, ratings are application-dependent. For example, consumers may indicate through an appropriate interface whether they liked, disliked or were neutral towards the content, so that $O=\{+,-,\varnothing\}$. Consumers may also indicate their interest on a scale from 1 (lowest interest) to 5 (highest interest) i.e., O={1,2,3,4,5}.

We assume that the rating a consumer gives depends on the category of the content in the following manner. Whenever a consumer, $b \in \mathcal{M}$, views a content belonging to category $f \in \mathcal{F}$, b provides a rating $o \in O$ which is drawn independently with probability $q^o_{b,f} \geq 0$, where $\sum_{o\in O} q^o_{b,f} = 1$. By "independently", we formally mean that, conditioned on the item category, the rating given by the user is independent of any ratings the consumer has given in the past. We note again here that the consumer never discloses the category to which the content item she views belongs.

Rating distribution. As a producer a comes in contact with a consumer b, an item produced by a is shown to b who, in turn, provides feedback. Let us denote by $\tilde{\pi}^o_{a,b}$ the probability that the rating provided in that case is $o \in O$. Note that $\sum_{o\in O} \tilde{\pi}^o_{a,b} = 1$. This probability may be computed as follows. First, determine the category of the item that was produced by a: this follows a probability characterized by $(\tilde{p}_{a,f})_{f\in\mathcal{F}}$. Second, by the assumption of independently drawn content, the category to which a content generated by a is independent of the contact process. Hence, conditioned on the event that a encounters b, the probability that the content category is f is still $\tilde{p}_{a,f}$. Third, conditioned on the category f, deduce the rating given by b according to the probability $\left(\tilde{q}^o_{b,f}\right)_{o\in O}$. In other words, we have

$$\tilde{\pi}_{a,b} = \sum_{f\in F} \tilde{p}_{a,f}\tilde{q}^o_{b,f} = \left\langle \tilde{p}_a, \tilde{q}^o_b \right\rangle,\ o \in O.$$

If, for any $o \in O$, we denote by Π^o the matrix $\left[\tilde{\pi}^o_{a,b}\right]_{a\in\mathcal{N},b\in\mathcal{M}}$, the equality above may be written in matrix form as $\Pi^o = P\left(Q^o\right)^T$, where $P = \left[\tilde{p}_{a,f}\right]_{a\in\mathcal{N},f\in\mathcal{F}}$ and $Q^o = \left[\tilde{q}^o_{b,f}\right]_{b\in\mathcal{M},f\in\mathcal{F}}$. Note that this implies that the matrices $\left(\Pi^o\right)_{o\in O}$ all admit a $|\mathcal{F}|$-rank decomposition and, as a result, that their ranks are no higher than $|\tilde{\mathcal{F}}|$. Thus, if the number of undisclosed categories is small, the matrices Π^o will be low rank. We stress that this is a simple consequence of the independence between the contact process and the nature of the content items, as captured by their undisclosed categories. This property will play an important role in predicting ratings, as we discuss next.

The goal of the rating prediction problem is to correctly estimate, for any producer/consumer pair, the probability that the consumer will react to content generated by the producer by providing rating $o \in O$. Such estimations are important because, as they can guide recommendation services for optimal content delivery.

The rating prediction problem amounts to correctly estimating the probability matrices Π^o, for all $o \in O$, in a distributed fashion. Our goal is then to devise an algorithm for finding profiles p_i, q_j, such that the prediction $\pi^o_{i,j}$ is as close to $\tilde{\pi}^o_{i,j}$ as possible. More formally, we wish to solve the following optimization problem:

$$\text{Minimize} \quad E = \sum_{i \in \mathcal{N},\, j \in \mathcal{M}} \lambda_{i,j} \sum_{o \in O} \left| \tilde{\pi}_{i,j} - \pi^o_{i,j} \right|^2$$

$$\text{subject to} \quad p_i \in D_1, i \in \mathcal{N}, \text{and}$$

$$q_j \in D_2 \; j \in \mathcal{M},$$

where D_1 and D_2 are the sets of profiles that satisfy the conditions for the probabilities p and q. We call the objective function E above the *error* of our estimate. It corresponds to the error of a rating selected uniformly at random among ratings within a timeslot. Its minimization is equivalent to minimizing a weighted root mean (RMSE), with weights equal to the delivery rates $\lambda_{i,j}$. In particular, when $\lambda_{i,j} = 0$ (*i.e.*, when i never delivers content to j), then E does not account for the distance between $\pi^o_{i,j}$ and $\tilde{\pi}^o_{i,j}$. This is not a bug but a useful feature: we never have to predict how j would react to content from i unless it receives such content.

6.1.2 A Distributed Algorithm

Our distributed algorithm for solving the above optimization problem is specified in Figure 5. It is fully distributed, and ensures that a consumer discloses the rating of a content item only to the producer that generated it. Moreover, producers share their profiles only with consumers that subscribe to their feeds, and vice-versa. A detailed analysis on the convergence properties of this algorithm is provided in the appendix and in [5].

Producer i at timeslot k:
  i generates new content item
  $\gamma \leftarrow \gamma(k)$
  for every pair i, j s.t. $a_{i,j}(k) = 1$:
    i sends its item and p_i to j.
    i receives $r_{i,j}$ and q_j from j.
    $p_i \leftarrow p_i + \gamma(k) \sum_{o \in \mathcal{O}} (\mathbb{1}_{r_{i,j}=o} - \langle p_i, q_j^o \rangle) q_j^o$
  $p_i \leftarrow \Pi_{D_1}(p_i)$

Consumer j at timeslot k:
  $\gamma \leftarrow \gamma(k)$
  for every pair i, j s.t. $a_{i,j}(k) = 1$:
    j receives item and p_i from i.
    j rates item with $r_{i,j} \in \mathcal{O}$.
    j sends $r_{i,j}$ and q_j to i.
    for every $o \in \mathcal{O}$:
      $q_j^o \leftarrow q_j^o + \gamma(\mathbb{1}_{r_{i,j}=o} - \langle p_i, q_j \rangle) p_i$.
  $q_j \leftarrow \Pi_{D_2}(q_j)$.

Figure 5. Distributed learning algorithm

6.1.3 Results

We test our rating prediction algorithm on the Netflix dataset. The wide use of the dataset as a benchmark allows us to implicitly compare the performance of our algorithm to the one achieved by state-of-the-art algorithms.

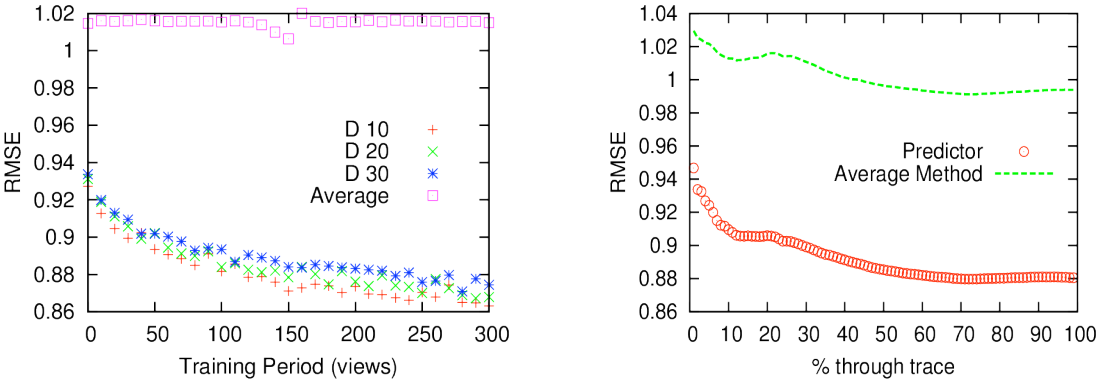
In 2006, Netflix announced a competition for recommendation systems, and released a dataset on which competitors could train their algorithms. The Netflix dataset consists of pairs of anonymized movies and anonymized users. Each trace entry includes a timestamp, the user ID, and the user's rating (on an integer scale of 1 to 5). The dataset includes both publicly available training data, for which ratings were provided, and a testing dataset, for which ratings were not disclosed. If for every movie in the test set, we simply always predict its average rated value from the training set, This approach would yield a root mean square error of 1.0540 on the test set. The winning team of the Netflix Prize challenge generated predictions with RMSE of 0.8572, a 10% improvement of the RMSE of Cinematch (0.9525), the algorithm designed by Netflix engineers.

We apply our algorithm as follows. Each movie is given a production profile $p_m \in [0,1]^d$. Similarly, each user is given a consumption profile $q_u \in [0,1]^{5\times d}$, corresponding to ratings with one, two, three, four, or five stars respectively (*i.e.*, O={1,2,3,4,5}). The Netflix dataset is arranged chronologically and as users rate movies, p for the movie and q for the user are updated as shown in the algorithm, Figure 5.

RMSE of rating prediction. Prior to each rating event, we predict the rating that a user, u, will give to a movie, m. We make this prediction by reporting the expected rating based on our estimation of the rating probabilities, *i.e.*, $\text{PredictedRating}(u,m) = \sum_{o \in O} o \cdot \langle q_u^o, p_m \rangle$.

We can then compare our prediction to the user's rating for this movie as recorded in the dataset, to calculate an RMSE. Our algorithm starts with a randomly selected p for each movie and a random q

for each user and adapts them as users rate movies. To account for a training period, we discard some of the early predictions before computing the RMSE. More precisely, we compute an RMSE for predictions with training period more than *k* by including predictions for which either the movie or the user profile has been adapted at least *k* times. In order to evaluate our effectiveness, we compare our RMSE against the Cinematch algorithm, the "average" algorithm applied to the training dataset (100,480,507 ratings that 480,189 users gave to 17,770 movies). We know the RMSE of this "average" algorithm for both training (1.015) and test (1.05) datasets, and so we can compare to these numbers.



(a) RMSE vs training time (b) RMSE evolution

Figure 6. RMSE as a function of training period and over time

Figure 6(a) shows the improvement in RMSE as we vary the training period for different numbers of dimensions *d* of the vectors. Regardless of the length of the training period, the RMSE of the "average" method remains at 1.015. Again, though we do not know how the original Netflix algorithm (Cinematch) functioned and how it performs on the training dataset, we expect this centralized solution to achieve roughly 10% improvement (9.18 RMSE over the training dataset). Our algorithm outperforms this value with a training period of only 10 iterations; a 15% improvement can be reached with training period of 300 views. Even when the training period is set to zero (*i.e.*, we include all predictions in the RMSE), it improves on the "average" method by 8% for all values of *d*, only slightly worse than Cinematch. Thus, our technique performs at least as well as some of the leading systems, even though our algorithm functions in a completely distributed manner.

Figure 6(b) shows how the RMSE evolves throughout the course of the Netflix trace. Our technique makes 50% of its improvements in RMSE during the first 10% of the trace, indicating that it performs well even in cold start situations. The algorithm consistently tracks the performance of the "average" method while demonstrating a consistent improvement of 8%-12%. More results on varying parameters of the algorithm as well as an implementation into a Facebook application are presented in the appendix and in [5].

We have proposed and validated the first content rating prediction technique that exploits the collective behavior of users without users or content producers providing any category information. The technique can operate in a fully distributed and asynchronous manner, such as in a mobile environment. We also empirically establish that in spite of its generality it performs as well as some of the best centralized techniques.

Our distributed algorithm can easily be used in combination with other techniques as we have shown that, no matter its starting point, the prediction error always decreases. As such it can also be thought

as a building block, ensuring scalability or helping to mitigate privacy issue. Its main advantage is that it applies to any situation of restricted connectivity, such as mobile networks leveraging intermittent peer-to-peer connections. Overall, our results greatly expand the scope of recommender systems beyond their current use: either by directly implementing it within one of the opportunistic mobile architectures that were recently, or inside any mobile architecture as a scalable prediction technique that allows exploiting locally relevant content.

6.2 Opportunistic privacy to defeat graph-based attacks

With the advance of mobile devices, a variety of new networking scenarios and applications are emerging with opportunistic networking. These networks are formed by mobile users collaborating and taking advantage of any wireless contact opportunity while on the move. In such open networks, one of the major issues is to maintain privacy for every user. This is even more true since users do not only consume but also provide services and thus may leak usage patterns or other privacy relevant data.

In this context, users get more and more concerned about their personal data being exposed and abused. In order to promote these technologies, one of the key challenges is to protect the privacy of users. This is even truer since users are mobile and operating on the move in open, possibly adversary, public environments (e.g., work, public transports, public places).

In this work in progress, we investigate ways to anonymize the contact graph i.e., the graph that can be observed by collecting all contacts (either through passive listening or exploiting routing information). While state-of-the-art anonymization scheme have mostly addressed mobility tracking, we are focusing on social network anonymization. Our goal is to increase the resilience graph-based attacks targeting more important nodes. Such attacks, depending on the type of graph, can severely disrupt the network. Figure 7 shows for both the exponential graph and the scale-free graph, the one-hop edges (green) that can be reached through the five highest node degree edges (red). 70% of a scale-free network can be reached through these five edges while only 40% in the exponential graph case. In other words, Scale-free networks (typically BA) are substantially more robust to the random deletion of vertices than ER graph, but, substantially less robust to deletion that specifically targets hubs with highest degrees. This is typically the kind of attack we want to avoid.

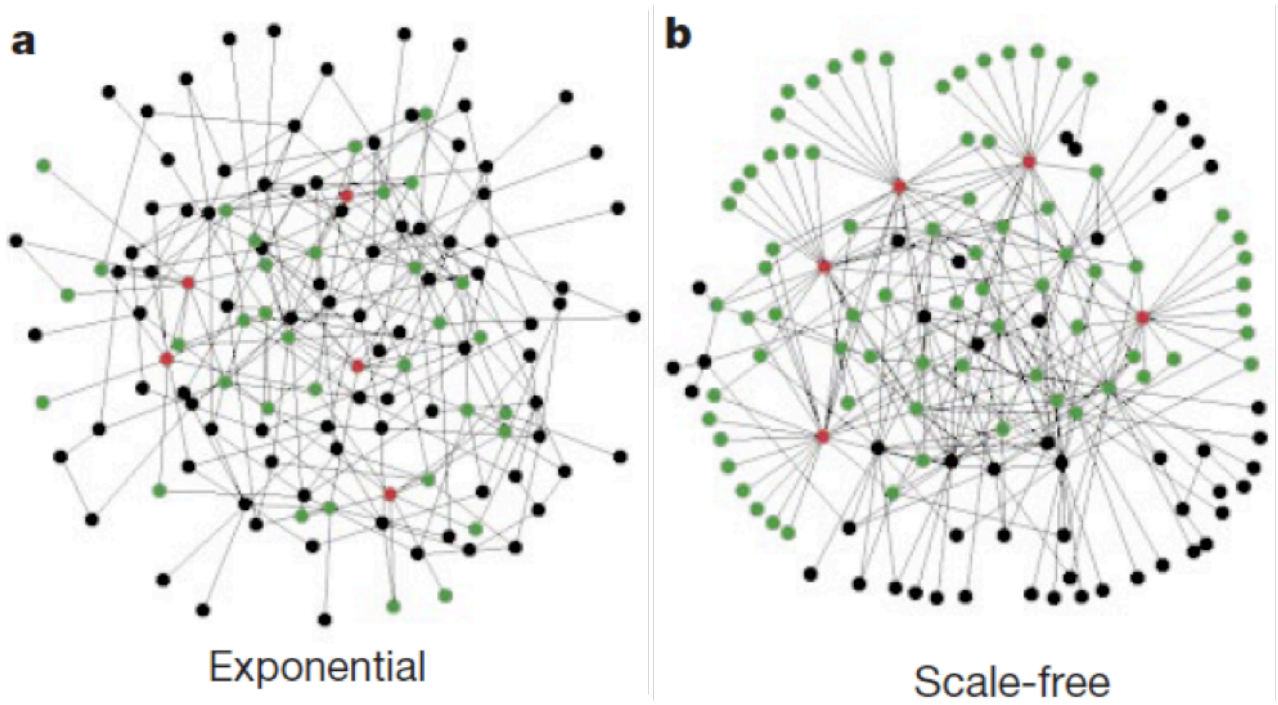


Figure 7 - Exponential vs. Scale-free networks: red edges represent the five edges with the highest node degree and the green edges represent their direct neighbors.

The current questions we are investigating are:

- What kinds of graphs have the highest entropy?
- How can we transform the original social contact graph to a more anonymized graph?
- How can sleep periods and the swapping and/or the random change of identifiers help in anonymizing the social graph?
- How does these schemes degrade the performance of content dissemination?

For now, we already have preliminary results on evaluating the impact of state-of-the-art privacy preserving techniques (i.e., techniques preventing mobility tracking) on the contact pattern and the resulting social graph using complex network metrics. The methodology is depicted in Figure 8. We can present these results during the Y1 review.

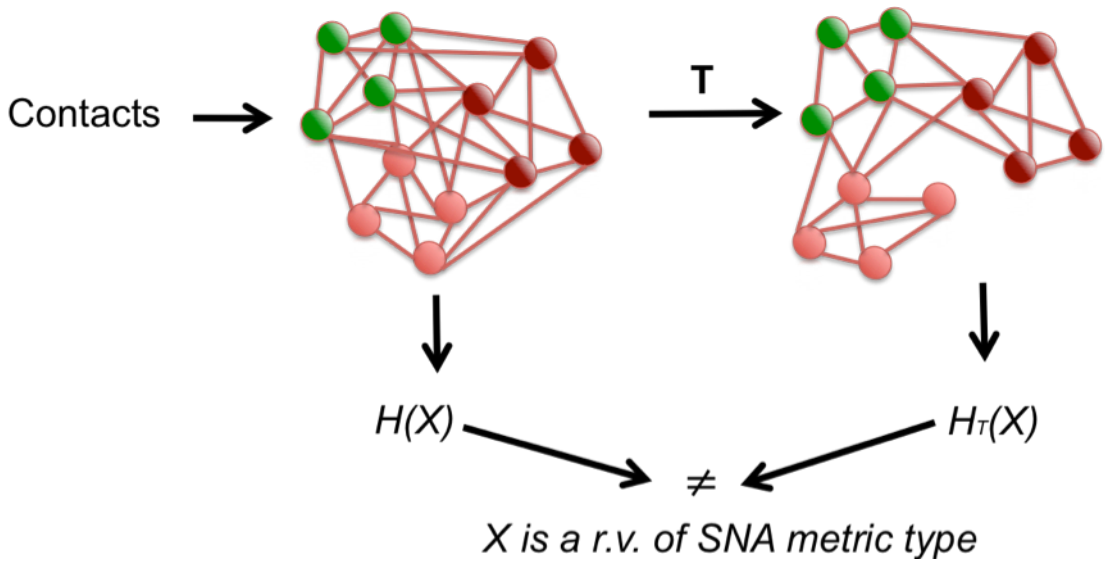


Figure 8 - Methodology: comparing the entropy of the original social graph with its anonymized (after transformation T) counterpart.

7 CONCLUSIONS

Enabling end users to go beyond the set of locally available resources and to pool resources available in the environment is the key goal of the SCAMPI Project. WP3 plays a fundamental role in achieving this vision, because its goal is to ensure the available of shared services that are secure and trustable. Moreover, results of the WP3 serve as central input to design of algorithms and structures of the SCAMPI service platform that are investigated and implemented in WP1.

SCAMPI's opportunistic computing paradigm is a fundamental component of the overall vision. Because it combines elements of pervasive computing/networking with opportunistic networking, we have presented strategies to enable opportunistic communication.

The environment is packed with resources that can be exploited, but exploiting them requires coping with an inherently dynamic and highly unstable network. Therefore, we have began to study how variable levels of resource stability can be successfully coped with, and we have addressed the basic notion of service discovery.

Moreover, we have sought ways to leverage and exploit the underlying social structure of the SCAMPI network. We believe this set of issues to be of the utmost importance for the remainder of our research efforts within the SCAMPI project.

An equally vital component of the overall picture is represented by content exchange schemes, which we have analyzed rigorously and thoroughly. Recommendation schemes and privacy preservation techniques complete the mosaic.

REFERENCES

[1] D. Puccinelli and S. Giordano, **On the Interplay of Low-Power Wireless Connectivity and Energy Consumption**, 2nd International Workshop on Networks of Cooperating Objects (CONET 2011), CPS Week, Chicago, IL, USA

[2] E. Borgia, M. Conti and A. Passarella, **Detecting users' communities in mobile social networks**, IIT-CNR Technical Report 19/2011, Available at http://www.iit.cnr.it/node/1096

[3] E. Borgia, M. Conti, A. Passarella, **Autonomic detection of dynamic social communities in Opportunistic Networks**, IFIP MedHocNet 2011, Favignana Island, Italy, 12-15 June 2011,

[4] Xia Zhou, Stratis Ioannidis and Laurent Massoulié, **On the Stability and Optimality of Universal Swarms**, ACM SIGMETRICS 2011, San Jose, CA, USA.

[5] Sibren Isaacman, Stratis Ioannidis, Augustin Chaintreau and Margaret Martonosi. **Distributed Rating Prediction in User Generated Content Streams.** RecSys 2011, Chicago, USA.

[6] S. Trifunovic, B. Distl, D. Schatzmann, F. Legendre, **WiFi-Opp: Ad-Hoc-less Opportunistic Networking,** CHANTS'11, Las Vegas, NV, USA

APPENDIX A

This appendix contains the reprint of the following papers:

S. Trifunovic, B. Distl, D. Schatzmann, F. Legendre, **WiFi-Opp: Ad-Hoc-less Opportunistic Networking,** CHANTS'11, Las Vegas, NV, USA

D. Puccinelli and S. Giordano, **On the Interplay of Low-Power Wireless Connectivity and Energy Consumption**, 2nd International Workshop on Networks of Cooperating Objects (CONET 2011), CPS Week, Chicago, IL, USA

E. Borgia, M. Conti and A. Passarella, **Detecting users' communities in mobile social networks**, IIT-CNR Technical Report 19/2011, Available at http://www.iit.cnr.it/node/1096

Xia Zhou, Stratis Ioannidis and Laurent Massoulié, **On the Stability and Optimality of Universal Swarms**, ACM SIGMETRICS 2011, San Jose, CA, USA

Sibren Isaacman, Stratis Ioannidis, Augustin Chaintreau and Margaret Martonosi. **Distributed Rating Prediction in User Generated Content Streams,** Technical report, Technicolor PAC-2011-05-01

WiFi-Opp: Ad-Hoc-less Opportunistic Networking

Sacha Trifunovic, Bernhard Distl, Dominik Schatzmann, Franck Legendre
Communication Systems Group
ETH Zurich, Switzerland
lastname@tik.ee.ethz.ch

ABSTRACT

Opportunistic networking offers many appealing application perspectives from local social-networking applications to supporting communications in remote areas or in disaster and emergency situations. Yet, despite the increasing penetration of smartphones, opportunistic networking is not feasible with most popular mobile devices. There is still no support for WiFi Ad-Hoc and protocols such as Bluetooth have severe limitations (short range, pairing). We believe that WiFi Ad-Hoc communication will not be supported by most popular mobile OSes (i.e., iOS and Android) and that WiFi Direct will not bring the desired features.

Instead, we propose WiFi-Opp, a realistic opportunistic setup relying on (i) open stationary APs and (ii) spontaneous mobile APs (i.e., smartphones in AP or tethering mode), a feature used to share Internet access, which we use to enable opportunistic communications. We compare WiFi-Opp to WiFi Ad-Hoc by replaying real-world contact traces and evaluate their performance in terms of capacity for content dissemination as well as energy consumption. While achieving comparable throughput, WiFi-Opp is up to 10 times more energy efficient than its Ad-Hoc counterpart. Eventually, a proof of concept demonstrates the feasibility of WiFi-Opp, which opens new perspectives for opportunistic networking.

Categories and Subject Descriptors

C.2.1 [**Network Architecture and Design**]: Wireless Communications

General Terms

Design, Performance

1. INTRODUCTION

The increasing penetration of smartphones with extended communication capabilities (3G/LTE, WiFi, Bluetooth) opens new networking possibilities. Opportunistic networking is one promising dimension which would bring networks back to the users and may instill the excitement of the Internet debut again.

Opportunistic networks bring many benefits to wireless infrastructure-based networks (e.g., 3G). They can increase capacity [1], efficiently use the available radio spectrum [2] and offload the infrastructure [3]. Moreover, they are a valuable asset to cope with partial or complete communication infrastructure outages caused by natural disasters or government censorship. Communication would be uphold in those situations at the cost of some delay.

There is objective evidence that opportunistic networking can bring many advantages. Nevertheless, no opportunistic application has found its way into an application store yet. The concept of opportunistic networking and smartphone applications leveraging these concepts are since years confined to research labs and small-scale testbeds [4, 5]. There are several reasons for this: Firstly, there is still no support for seamless and high throughput short to mid-range Ad-Hoc communications in stock phones. WiFi Ad-Hoc is still not supported on most popular phones[1] and will probably not be in the near future, if ever [6]. Secondly, Bluetooth suffers from its limited bandwidth and the cumbersome peering/service detection procedure, not to mention implementation incompatibilities. Thirdly, even if some platforms such as Windows Mobile or Symbian support WiFi Ad-Hoc, these suffer from the lack of a popular application store and low or decreasing market share. Eventually, WiFi Ad-Hoc is very power inefficient, draining the battery in only a few hours. One might claim that WiFi Direct [7] will improve this issue, though nothing guards us from mobile OS restrictions (see iPhone WiFi tethering mode) or similar Bluetooth-like cumbersome pairing procedures [8].

We believe that applications leveraging opportunistic principles currently face a chicken and egg problem: *As long as opportunistic applications are not popular, smartphone manufactures will not implement the APIs required to setup WiFi Ad-Hoc connections. But as long as application developers do not have APIs to exploit the advantages of opportunistic networks, how can they create such popular applications?* Instead of relying on wishful thinking [6], hoping for the coming support of WiFi Ad-Hoc or WiFi Direct, we reconsider the problem from a more Cartesian approach. By looking at what is available now (and not what we wish to be feasible in the future), we ask ourselves how we can overcome this chicken and egg problem using the current smartphone generation.

In this paper, we propose WiFi-Opp, an alternative way of enabling opportunistic networking based on current smartphones' communication features and APIs. WiFi-Opp leverages the mobile WiFi AP feature (also known as tethering) as well as stationary open APs to support opportunistic communications between classical WiFi client.

To summarize, the main contributions of this paper are:

- We propose WiFi-Opp, a simple and energy efficient way of enabling opportunistic networks using current smartphones' technology (Section 2).

[1]Unless they are rooted or jailbreaked.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
CHANTS'11, September 23, 2011, Las Vegas, Nevada, USA.
Copyright 2011 ACM 978-1-4503-0870-0/11/09 ...$10.00.

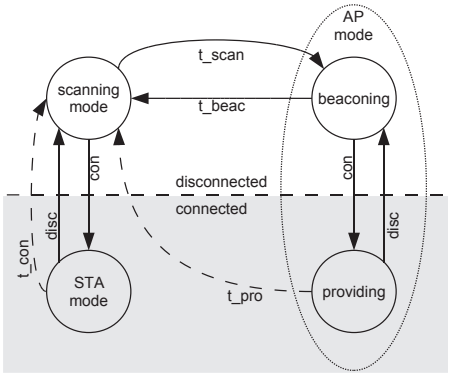


Figure 1: State transition diagram of WiFi-Opp.

- We evaluate the performance and energy efficiency or WiFi-Opp for content dissemination. We show that our approach is up to 10 times more energy efficient than WiFi Ad-Hoc for comparable dissemination performance (Section 3).
- We present a proof of concept demonstrating WiFi-Opp's feasibility (Section 4).
- We discuss the implications of becoming a spontaneous mobile AP, the current device heterogeneity in the market and the benefits of WiFi-Opp over WiFi Direct (Section 5).

2. WIFI-OPP: AD-HOC-LESS OPPORTUNISTIC NETWORKING

Wifi Ad-Hoc is absent from most popular smartphones (Android, iPhone) which requires us to find an alternative technology to enable opportunistic communications between co-located nodes. Devices should hence use the well established 802.11 infrastructure mode and leverage the stationary access points (APs) within range by associating to them and using them as relay to exchange data. But what if an AP prevents client-to-client communications [9], if it is protected by a key (WEP, WPA), or if none is within range in the first place? To overcome these limitations, we propose that some mobile devices themselves switch spontaneously to mobile APs. This can be achieved by putting the mobile device in the so called tethering or mobile AP mode. While this mode is initially meant to share one's 3G Internet access to clients, it can be used for client-to-client communications and even mobile AP-to-client communications thus enabling opportunistic communications. This is the core concept of WiFi-Opp.

Yet, one main drawback of 802.11 is that co-located APs cannot discover each other as well as co-located clients or stations (STA) connected to different APs (either stationary or spontaneous) will not be able to communicate with each other hence resulting in isolated clusters centered around APs. If we aim at content dissemination, we need to enforce some topological dynamics for content to spread. This can come for free thanks to the mobility of stations, which might roam between different APs hence carrying content further (this also applies to mobile APs). However, the scenarios we are targeting are mostly static (e.g., public transports, campus, bars, concert). Besides, most traces have highlighted that nodes are stationary for a large portion of time interrupted by mobility events. We hence introduce randomized switching between the spontaneous AP and STA modes. This will allow breaking clusters resulting in dynamic topological reconfigurations. In addition, this switching strategy will even the energy consumption between nodes, for a STA does not consume as much as an AP which relays traffic for others.

In WiFi-Opp, nodes can operate in 3 different modes as shown in Fig. 1: scanning, station (STA), and access-point (AP). For all scenarios we use the following parameter definitions:

- t_scan is the time a mobile device scans for APs.
- t_beac the time it advertises its presence in AP mode by sending SSID beacons.
- t_con the time a STA stays connected with a specific AP.
- t_pro the time a mobile device is providing AP functionality for connected STAs.

In the rest of this section, we present five different WiFi-Opp based communication scenarios, all using the aforementioned principles. The first one, called *Static*, is the simplest base scenario. We extend it to enforce topological reconfigurations with the *Flexible STA* and *Flexible AP* scenarios. The *Manual* case considers the manual switching from STA to AP by users. Those scenarios still ignore the presence of stationary open APs. Therefore, in a last step, the *Fixed APs* scenario considers the usage of existent infrastructure APs.

2.1 WiFi-Opp – Static

In the *Static* approach (see Fig. 1), each device periodically scans the spectrum[2] for $t\_scan \sim U[t\_scan_{min}, t\_scan_{max}]$ seconds to discover any APs in range. If an AP is found it will connect to it, otherwise, a node becomes an AP itself after its scanning time has expired. This strategy allows the randomized election of an AP for co-located nodes (within range). Devices becoming AP will stay up for $t\_beac \sim U[t\_beac_{min}, t\_beac_{max}]$ seconds awaiting for STA associations. If the AP has clients (STA) it will stay up as long as at least one STA is associated. Otherwise, after t_beac is over, it will fall back to scanning mode again for t_scan seconds. This way, an isolated node (without any neighbor) will not uselessly stay in AP mode (and hence save energy). A STA that gets disconnected from an AP due to a contact loss or any other reason, will fall back to scanning mode. In this case nevertheless, the node will not scan for t_scan seconds (which might be very long) but immediately try to find a new AP or becomes one within a few seconds (fast reconnect). The fast reconnect feature allows for a quicker response to topological changes and is also used in the next two scenarios.

2.2 WiFi-Opp – Flexible STA

The *Static* approach might lead to multiple independent clusters that cannot communicate although they are physically co-located. The reason is that stations associated to one AP cannot communicate to stations associated to another. Although mobility of the nodes might aid in this situation, for a temporally static scenario (e.g., people sitting in offices), we extend the *Static* approach and introduce the *Flexible STA* scenario, allowing for stations to disconnect from an AP after the time $t\_con \sim U[t\_con_{min}, t\_con_{max}]$ has elapsed (see Fig. 1) and scan for other APs or become one itself (fast reconnect explained above).

2.3 WiFi-Opp – Flexible AP

Although the *Flexible STA* approach might improve connectivity by dynamically reconfiguring the clusters, the AP cannot fall back to scanning mode as long as it has associated stations. This might lead to uncontrolled battery drainage. Hence the *Flexible AP* scenario provides APs the possibility to disconnect stations and become scanning again. In this case these stations will then do a fast reconnect (described above) in order not to loose valuable contacts.

In this scenario, APs switch back to scanning mode after t_pro seconds (see Fig. 1), despite associated stations. However, we make t_pro depend on the AP's importance i.e., the number of connected

[2]We assume that scans are triggered every 5s.

	H06	MIT	ETH
# Nodes	78	96	280
Time Period	93 hours	14.9 weeks	14.6 weeks
Type	Bluetooth	Bluetooth	AP Association
# Contacts Total	128'979	75'432	99'024
# Contacts / Node	1654	786	354

Table 1: Properties of contact traces.

stations ($\#_{STA}$) as follows: $t\_pro \sim U[\#_{STA} \cdot t\_con_{min}, \#_{STA} \cdot t\_con_{max}]$, where $[t\_con_{min}, t\_con_{max}]$ is the range of connection time per station (also used for the *Flexible STA* scenario).

2.4 WiFi-Opp – Manual

Some of the current smartphones do not yet provide the possibility to automatically switch the AP mode on/off[3]. In this case, this procedure has to be performed manually by the user. We hence consider the case where a certain percentage of users perform this action in order to exploit the opportunistic contacts while on the move. In such a case, spontaneous APs might be fewer but available for longer time periods (e.g., 30 minutes commuting time). In a more extreme settings such as disaster scenarios or situations where the network infrastructure is teared down, the incentive for manual configuration will probably be quite high.

2.5 WiFi-Opp – Fixed APs

In major urban environments, there exist many spots with open APs (e.g., stores, public WiFi hotspots). Although these open APs usually redirect users willing to access the Internet to a web portal to authenticate, they can still be leveraged for opportunistic communications. Two users connected to the same AP can still communicate without necessarily being authenticated. We consider this scenario assuming a certain density of stationary open APs.

3. PERFORMANCE EVALUATION

We compare the performance of WiFi Ad-Hoc and WiFi-Opp by replaying real-world contact traces and simulating content spreading. We assess the impact of our main parameters on the dissemination performance as well as the energy efficiency of each approach.

3.1 Mobility Traces

We evaluate the WiFi-Opp algorithm by replaying three real-world contact traces presented below. These traces differ in terms of size, duration, and contact frequency. Their main characteristics are summarized in Table 1.

Haggle Infocom 2006 (H06): During Infocom 2006 contact traces of 78 conference attendees wearing Bluetooth devices were collected for the Haggle project [10]. The granularity of the direct contacts collected on 4 continuous days is given by the 2 minutes scanning interval of the devices.

MIT Reality Mining (MIT): The Reality Mining project [11] collected Bluetooth contacts with a 5 minute scanning interval of 96 students and staff members on the MIT campus during several month. We extract the 15 weeks of the trace with the highest contact density.

ETH Campus (ETH): On the ETH campus access point (AP) associations where logged during 15 weeks. We only consider 280 users that have an AP association for at least 5 days a week. The trace is preprocessed in order to remove short disconnections due to interference and the well known ping-pong effect where devices jump back and forth between different APs. Two nodes are considered in contact while associated to the same AP.

[3]This is the case with iOS while it is feasible with Android (see Section 4).

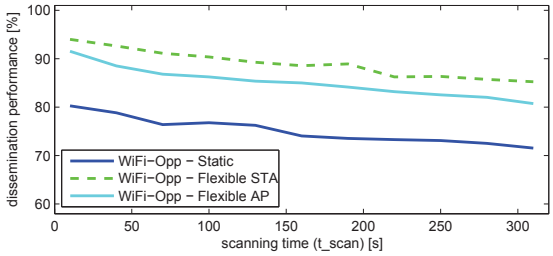


Figure 2: Dissemination performance depending on scanning time (ETH trace). 100% corresponds to the WiFi Ad-Hoc performance.

3.2 WiFi Ad-Hoc vs. WiFi-Opp: Dissemination Performance

We compare the amount of data any node can disseminate to any other node using either WiFi Ad-Hoc or WiFi-Opp. We use a simple epidemic spreading model, thus information is flooded to all nodes without any sophisticated distribution scheme. Nodes getting content from the source can then relay it to others. In each simulation run, we consider a different node as the source of content and average the results over all runs. We assume infinite buffer space so that the performance outcome only depends on the parameters of our approach.

Regarding the link capacity model, we constantly track the number of k nodes that are in contact in the traces by computing the cliques composing these nodes. In a clique of size k there are $l = k(k-1)$ directed links. In the WiFi Ad-Hoc case, assuming nodes composing a clique are in contact for duration t, we calculate the individual link capacity as $d = \frac{t}{l} \cdot throughput$. For the WiFi-Opp approach, there are also l directed links. However, only the $2(k-1)$ AP-to-STA links have capacity d. The STA-to-STA links have only half the capacity ($\frac{d}{2}$), since the traffic has to be relayed through the AP. Note that in both cases, we did not consider any interferences nor possible overheads.

In the sequel, the dissemination performance of the WiFi-Opp variants is always represented as a percentage of the dissemination performance based on WiFi Ad-Hoc communication that we use as a baseline. Although WiFi-Opp takes probabilistic decisions, the variance from one simulation run to the next is negligible since we are replaying traces. Two exceptions though are the *Manual* and *Fixed APs* approaches where we choose a random subset of participating nodes hence generating more variance. In these cases, we average over 10 simulation rounds. If not specified otherwise, the times t_scan_{min} and t_beac_{min} are set to 10 seconds and t_con_{min} is set to 120 seconds. The respective maximum times are always 3 times the min times, e.g $t\_scan_{max} = 3 \cdot t\_scan_{min}$. Note that the connection time t_con is only relevant for the *Flexible STA* and *Flexible AP* variants.

We analyze the impact of the three main WiFi-Opp parameters in isolation, namely the scanning time t_scan, the beaconing time t_beac, and the connection time t_con (and implicitly the AP providing time t_pro). The following results are all shown for the ETH trace. Nevertheless, simulations based on the Haggle or the MIT trace show similar results.

3.2.1 The impact of the scanning time

Intuitively, if two scanning nodes meet, the average time until one of them switches into AP mode, thus making a connection possible, is longer if the scanning time t_scan is longer. This can impact the performance, since more connection opportunities are missed. This impact is visible in Fig. 2. Depending on the scanning time (x-axis) the plot shows the average dissemination performance

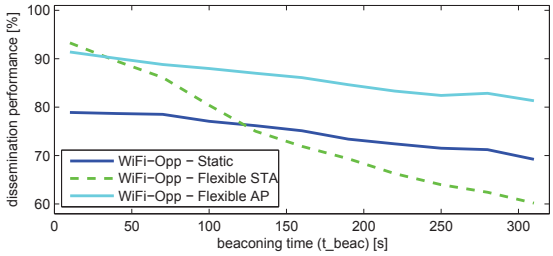


Figure 3: Dissemination performance depending on beaconing time (ETH trace). 100% corresponds to the WiFi Ad-Hoc performance.

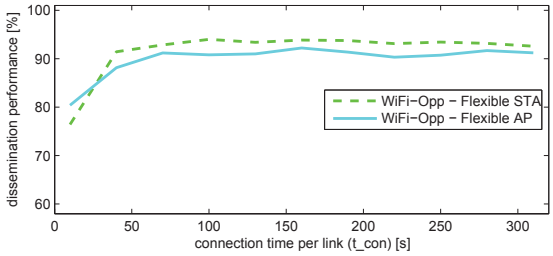


Figure 4: Dissemination performance depending on associated time (ETH trace). 100% corresponds to the WiFi Ad-Hoc performance.

(y-axis). The *Flexible STA* variant (green dashed line) generally performs best with 85-95% of the classical WiFi Ad-Hoc performance. The *Flexible AP* variant (light blue solid line) performs slightly worse since it disconnects stations simultaneously causing a short connection break. The *Static* variant (dark blue solid line) has significantly lower performance than the *Flexible STA* and *Flexible AP* variants due to a lack of dynamics in the network. For all the variants, the performance only decreases around 10% when increasing the scanning time from 10-30 seconds to 5-15 minutes.

Conclusion: The WiFi-Opp performance is almost independent of the scanning time, thus permitting long scanning times of 5-15 minutes (and saving energy as we will see later) with only a slight performance decrease compared to WiFi Ad-Hoc. Besides, enforcing topological reconfigurations with WiFi-Opp Flexible STA/AP improve by at least 10% the performance compared to the Static variant, which relies only on mobility to generate dynamics.

3.2.2 The impact of the beaconing time

The beaconing time t_beac defines how long a node advertises its presence in AP mode. Fig. 3 shows that increasing the beaconing time (x-axis) actually lowers the performance (y-axis) for all WiFi-Opp variants, namely the *Static* (dark blue solid line), the *Flexible STA* (green dashed line), and the *Flexible AP* variant (light blue solid line). The reason for this performance loss is that two beaconing nodes (in AP mode) can become co-located due to mobility i.e., two nodes switch to AP mode and then come into proximity. They thus cannot discover each other and the connection opportunities are missed until one node falls back to scanning mode. The *Flexible STA* variant is more affected since disconnected stations will beacon for a long time, even if there are no scanning nodes around (the AP they where connected to is also still beaconing as seen in Fig. 1).

Conclusion: Increasing the beaconing time in AP mode has a negative impact on the performance since mobility might lead to several co-located APs which cannot communicate between them.

3.2.3 The impact of flexible connection times

Flexible connection times between STAs and APs, t_con and $t\_pro = \#_{STA} \cdot t\_con$, allow an increase of topological dynamics. Nevertheless, if the connections are very short, the performance decreases (see Fig. 4). The *Flexible STA* (green dashed line) and the *Flexible AP* variant (light blue solid line) have the best dissemination performance (y-axis) if the connection time (x-axis) is above 40 seconds in the ETH trace. For longer connection times the performance stays constant.

Conclusion: The performance of the Flexible STA and Flexible AP variants of WiFi-Opp is largely independent of the connection times (if $\geq$ 40s), reaching 90% of the WiFi Ad-Hoc performance.

Operation	Battery consumption
802.11 scanning	4.80% per 24h
UDP beaconing (station)	28.66% per 24h
802.11 SSID beaconing (AP/Ad-Hoc)	124.55% per 24h

Table 2: Energy consumption for different operations expressed as the percentage of a Nexus One battery they consume in 24h.

3.3 WiFi Ad-Hoc vs. WiFi-Opp: Energy Consumption

In order to perform an energy consumption analysis we measured the required energy for different WiFi-Opp operations on real phones. We used 3 Nexus Ones running Android 2.3.4. As a baseline, we verified how much energy they consume with WiFi turned off. We measured then the energy to scan for WiFi APs (scanning mode), to maintain tethering mode (AP mode), and to send UDP beacons in 2 second intervals (STA mode). The UDP beacons are necessary for station associated to the same AP to discover each other and are not to be confused with the SSID beacons an AP sends. The AP also relays all these UDP beacons. WiFi Ad-Hoc consumes the same amount of energy as the AP mode, assuming no devices are connected to it. Connected devices distribute the SSID beaconing process among them.

The energy measurements can be seen in Table 2. Energy consumption of an operation will always be represented as a percentage of a Nexus One battery it consumes in 24h. This means that if an operation consumes 100% energy it would drain a Nexus One in 24h (assuming the phone does nothing else). More than 100% means that the battery is drained in less than a day, thus the smaller the values, the longer is the phone's lifetime.

In Table 2 we can see that most energy is required for sending SSID beacons. WiFi-Opp should thus reduce the sending of useless SSID beacons, i.e. being an unused AP (time t_beac), as much as possible. We thus analyze the impact of the scanning and the beaconing time on the energy consumption next.

3.3.1 The impact of the scanning time

The energy consumption is mainly impacted by the scanning time t_scan and the beaconing time t_beac. Since scanning is the cheapest operation in terms of energy, the longer we scan and do not beacon, the less energy we consume. This is confirmed by Fig. 5 for the MIT trace. By increasing the scanning period (x-axis) the energy consumption (y-axis) of WiFi-Opp (blue solid line) drops considerably. For long scanning periods (5-15 minutes), energy consumption drops to 12% of what WiFi Ad-Hoc (red dashed line) consumes (36% for H06 and 10% for ETH). Recall from Fig. 2 that performance is still good for long scanning periods.

Conclusion: WiFi-Opp can consume as low as 10% of its WiFi Ad-Hoc counterpart while still achieving almost 90% of the WiFi Ad-Hoc dissemination performance (see Fig. 2).

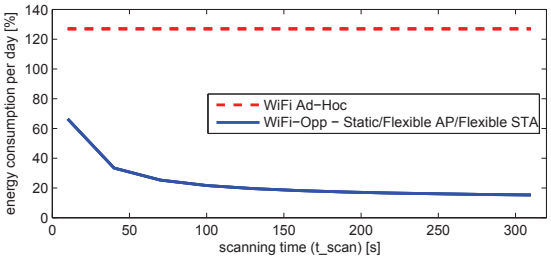


Figure 5: Energy consumption depending on scanning time (MIT trace).

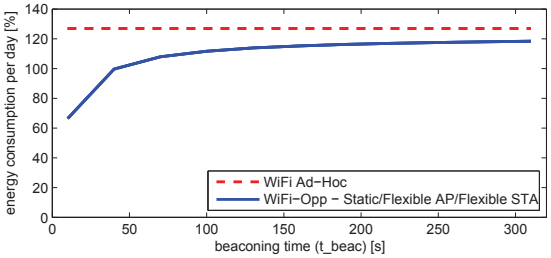


Figure 6: Energy consumption depending on beaconing time (MIT trace).

3.3.2 *The impact of the beaconing time*

While increasing the scanning time reduces the energy consumption, increasing the beaconing time t_beac does the opposite. This can be seen in Fig. 6 for the MIT trace. By increasing the beaconing time (x-axis) the energy consumption (y-axis) of WiFi-Opp (blue solid line) increase drastically, close to the consumption of a classical WiFi Ad-Hoc network (red dashed line). As we saw in Fig. 3 there is actually no performance gain from increasing the beaconing period.

Conclusion: The beaconing time should be kept to a minimum. Increasing it has a negative impact, both on the energy consumption (Fig. 6) and dissemination performance (Fig. 3).

3.4 Manual Scenario

In the *Manual* scenario, we assume that a certain percentage of users participate in the creation of the network by manually setting their device into AP mode. They do so twice a day, (e.g. in the morning and evening while commuting from/to work), for a time between 30 minutes to 1.5 hours. The rest of the time all the nodes are in scanning mode. The resulting dissemination performance for the Haggle trace is shown in Fig. 7. As expected with an increasing percentage of participating users (x-axis) the dissemination performance (y-axis) increases. The performance between 2% and 17% might seem low in relative terms but looking at the actual amount of data that can be transferred, this shows a different picture. In the Haggle trace, 17% of the capacity means that a node can disseminate around 1400 seconds worth of data to each other node on average during the 3 days of the trace. Assuming a transfer rate of $500kB \cdot s^{-1}$ this corresponds to nearly $700MB$ of data in 3 days.

Conclusion: In a manual scenario the capacity naturally shrinks drastically compared to the automatic scenarios but in terms of amount of data that can be transferred it is still significant.

3.5 Fixed APs Scenario

Open AP infrastructure has a lot of potential in aiding opportunistic communication. We simulated its effect by choosing a fraction of random nodes in the trace to be fixed infrastructure nodes. For the Haggle trace, Fig. 8 shows the performance (y-axis) that

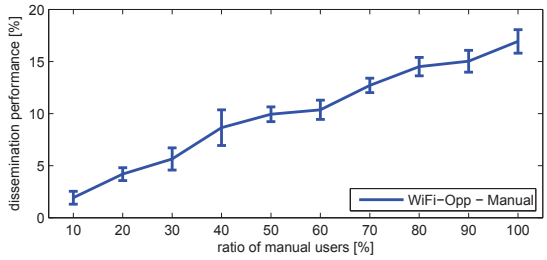


Figure 7: Dissemination performance depending on manual users (H06 trace). 100% corresponds to the WiFi Ad-Hoc performance.

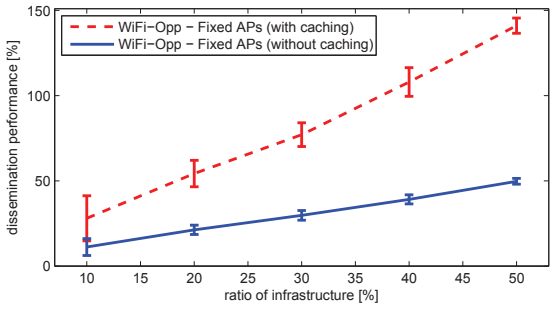


Figure 8: Dissemination performance depending on available APs (H06 trace). 100% corresponds to the WiFi Ad-Hoc performance.

can be gained by just exploiting open APs for different amounts of infrastructure nodes (represented on the x-axis as a percentage of total nodes). By just relaying data the infrastructure can provide considerable performance (blue solid line). This will be the general case. Nevertheless, if some APs, e.g. belonging to a campus network, want to foster opportunistic communication they can more than double the performance by caching data (red dashed line). By caching data the dissemination performance of the network can even surpass the one of classical WiFi Ad-Hoc networks. This can be explained by the higher bandwidth of AP-to-STA communications (twice the STA-to-STA bandwidth) and the increased availability of content at one-hop i.e., AP with cache instead of multi-hop dissemination.

Conclusion: Open infrastructure APs hold a lot of potential for opportunistic communications and even more if adjuncted with caches.

4. PROOF OF CONCEPT

We implemented a WiFi-Opp proof of concept application on Android. The implementation was tested on stock (unrooted) Google Nexus One phones running Android 2.3.4. The application switches to tethering mode (AP), disabling 3G connection while in it, and then goes back to scanning once the beaconing time is over. To automatically switch to the tethering mode, we used the Java Reflection API since there is no Android API call for this purpose. Disabling the 3G connection to protect the Internet access can be done by renaming the APN (Access Point Name) in the 3G configuration. For the rest of the functionalities, normal API calls were used. To continuously send UDP broadcast packets (beacons) while being associated to an AP, even while the screen is off and the device is running on battery, we needed to enable the WiFi Lock feature of Android. In order to omit the UDP beacons from the AP, which serve the sole purpose of communicating the current IP address, a station can find the IP address of the AP in the DHCP information. To the best of our knowledge, WiFi-Opp features are currently fully supported by Android and with some limitations (only for the AP mode) on Symbian, Windows Mobile and the iPhone.

5. DISCUSSION AND RELATED WORK

In the following, we discuss (i) the issue of establishing opportunistic communications and having access to the Internet simultaneously, (ii) the current device heterogeneity in the market, (iii) the relation of WiFi-Opp with WiFi Direct, and (iv) how WiFi-Opp could be further enhanced to be more energy efficient.

Internet access: The tethering (AP) mode is explicitly designed to share a smartphone's 3G Internet access with other devices. Nevertheless, users running WiFi-Opp in AP mode (tethering) might not be willing to share their 3G access with others. Currently we just disable 3G while using WiFi-Opp. The only way we found to protect 3G access from station access is by setting up a L2TP VPN connection which has to be launched manually. Further, users might not have a 3G subscription and exclusively use WiFi in order to connect to the Internet. WiFi-Opp might prevent these users from adopting opportunistic networking since they will lose their ability to access the Internet while WiFi-Opp is running. This could be solved by virtualizing the WiFi cards but this is not yet supported by mobile OSes. The *Fixed AP* approach, assuming the AP in question provides Internet access, is currently the ideal solution. For the remaining automatic WiFi-Opp scenarios the phone would need to switch between the opportunistic and the infrastructure realm. A simple tradeoff would be to enable opportunistic networking while the phone is in standby (screen off) and connect to the WiFi Internet AP once the phone wakes up (screen on) and the user needs it. This will be integrated into WiFi-Opp.

Device Heterogeneity: Currently we only have a proof of concept for the Android OS (version 2.2 and higher). Nevertheless, it is currently the most popular OS with the highest and still increasing market share. A WiFi-Opp application should also work for operator branded Android phones which only allow enabling the tethering mode with an additional subscription. Since only the AP functionality but not the Internet sharing feature of the tethering mode is used, operators have no reason to block a WiFi-Opp application from the market. Other OSes like the iOS only allow for a manual change into the tethering mode. Nevertheless, all devices with WiFi capabilities are able to participate as a station. We believe with the current mix of available devices and taking into account the fixed open APs, opportunistic networking is feasible.

WiFi Direct: The recently released WiFi Direct standard [7] enables two or more peers to discover each other and communicate over WiFi. Nevertheless, it was not designed with opportunistic networking in mind, but aims at connecting groups of WiFi enabled devices. Pairing such as entering a PIN is compulsory in the WiFi Direct standard and thus group formation can take up to 2 minutes and requires user interaction. Since we want to enable communication in a dynamic environment this is not too limiting. Security should be addressed at a higher layer. Nevertheless, security-less WiFi Direct might be used to enable opportunistic networking but since the AP mode is much simpler and works just as well, why not use WiFi-Opp. Furthermore, the WiFi Direct standard was just released and it is yet unclear how it will be supported in mobile OSes.

Context-aware WiFi-Opp: Although WiFi-Opp is energy efficient, there is still a lot of room for improvement by making it context-aware. The most promising extension is to use the embedded low-power sensors (e.g., accelerometer) to infer the motion context and adapt the WiFi-Opp strategy [12]. One could also go one step further and adapt the discovery strategy depending on the battery level. This is done by Google Latitude in order to provide an energy efficient position tracking service [13]. Such additional measures could also be exploited by WiFi-Opp and would render the energy consumption argument against opportunistic networks not holding anymore.

6. CONCLUSIONS AND FUTURE WORK

In this paper, we have proposed WiFi-Opp, a viable and energy efficient alternative to WiFi Ad-Hoc to support opportunistic communications by exploiting the mobile AP mode of today's smartphones. A proof of concept of WiFi-Opp has been implemented on stock smartphones and does not require root privileges. Future work will investigate incentives for users to become APs and share their Internet access. We will also consider more realistic traffic patterns for our evaluation. Eventually, we will further enhance the energy efficiency of WiFi-Opp. We are currently implementing WiFi-Opp as a fully-fledged opportunistic framework for Android and the iPhone which can be used by application developers to design opportunistic (and hybrid) applications. Once implemented, it will be deployed in our testbed for an in situ performance evaluation and be released to the public.

Acknowledgments

We want to thank the anonymous reviewers for their advice in improving the paper and Cristian Tuduce for the ETH access point trace. We also thank the Google4EDU project for providing us with Nexus Ones. This work was partially funded by the European Commission under the SCAMPI (258414) and the OpenLab (287581) FIRE projects.

7. REFERENCES

[1] M. Grossglauser and D. N. C. Tse, "Mobility increases the capacity of ad hoc wireless networks," *IEEE/ACM ToN*, vol. 10, 2002.

[2] V. Vukadinovic and G. Karlsson, "Spectral efficiency of mobility-assisted podcasting in cellular networks," in *ACM MobiOpp*, 2010.

[3] B. Han, P. Hui, M. Marathe, G. Pei, A. Srinivasan, and A. Vullikanti, "Cellular traffic offloading through opportunistic communications: A case study," in *ACM Chants*, 2010.

[4] "PhotoShare Haggle Demo," http://code.google.com/p/haggle/wiki/PhotoShare.

[5] A.-K. Pietiläinen, E. Oliver, J. LeBrun, G. Varghese, and C. Diot, "Mobiclique: middleware for mobile social networking," in *WOSN*, 2009.

[6] "Android issue 82: Support ad hoc networking," http://code.google.com/p/android/issues/detail?id=82.

[7] WiFi Alliance, "Wi-Fi Peer-to-Peer (P2P) Technical 7 Specification," www.wi-fi.org/Wi-Fi_Direct.php.

[8] "WiFi Direct Demonstration at CES 2011," http://www.youtube.com/watch?v=_mv-XFZmwNA.

[9] Wikipedia, "Wireless lan security," http://en.wikipedia.org/w/index.php?title=Wireless_LAN_security&oldid=415749161.

[10] A. Chaintreau, P. Hui, J. Crowcroft, C. Diot, R. Gass, and J. Scott, "Impact of Human Mobility on the Design of Opportunistic Forwarding Algorithms," in *IEEE INFOCOM*, 2006.

[11] N. Eagle and A. Pentland, "Reality mining: sensing complex social systems," *Personal Ubiquitous Comput.*, vol. 10, 2006.

[12] K.-H. Kim, A. W. Min, D. Gupta, P. Mohapatra, and J. Singh, "Improving energy efficiency of wi-fi sensing on smartphones," in *IEEE INFOCOM*, 2011.

[13] "Google Latitude - Location Update Frequency," www.google.com/support/mobile/bin/answer.py?hl=en&answer=136647.

On the Interplay of Low-Power Wireless Connectivity and Energy Consumption

Daniele Puccinelli and Silvia Giordano

University of Applied Sciences of Southern Switzerland

Abstract. We present an extensive set of experimental results on two remote-access wireless sensor network testbeds that illustrate the interplay between network connectivity and energy consumption. Using collection on top of a duty-cycled link layer, we observe that the energy consumption of unicast traffic is heavily affected by insufficient connectivity conditions. Specifically, the presence of connectivity outliers that require a large number of retransmissions to overcome transitional links has a significant impact on the energy consumption of the whole radio neighborhood of the outliers. At the other end of the connectivity spectrum, high connectivity may have a significant impact on the energy consumption of broadcast traffic, because it is conducive to contention and overhearing. To enable fair comparisons between experimental runs, we augment our results with quantitative data regarding the network topology during each run.

1 Introduction

Low-power wireless networks, such as wireless sensor networks, operate at very low transmit power levels and typically employ radio hardware with a relatively low sensitivity. Therefore, low-power wireless networks are exposed to all the vagaries of wireless propagation [1]. In low-power wireless, the transmit power typically ranges between -20 dBm and a few dBms, dropping to about -50 dBm at the standard reference distance of 1m from the signal source. A loss in the order of 50 dB is sufficient to challenge the radio sensitivity, which typically lies in the [-110, -100] dBm range. The large-scale path loss alone would only cause a loss of this order of magnitude over a relatively long distance (tens to hundreds of meters, depending on the path loss exponent). Nonetheless, in indoor environments, shadowing and fading effects tend to dominate over the large-scale path loss [2]. Shadowing is caused by any obstacle that blocks the line-of-sight path, be it a wall, a floor, a ceiling, or a person standing or passing between wireless terminals (body shadowing); in the latter case, shadowing is time-varying. Multipath fading is caused by the superposition of reflected paths and also plays a major role in increasing the power loss with respect to what would be expected based on the large-scale path loss.

Link dynamics are the deterministic product of the layout of the deployment area, which in turn determines the amount of large-scale path loss and static multipath fading that affects the nodes. Indeed, multipath fading is a spatial

phenomenon that is solely determined by the layout of the deployment area [2]. In practice, however, the layout of the deployment area cannot be expected to remain stationary (furniture gets moved around, doors and windows are opened and closed, ...), and the motion of people and objects across the deployment area alters the existing fading patterns and produces time-varying induced fading [3]. Time-varying shadowing and fading effects, along with time-varying interference, are the key drivers of the time dynamics of wireless links in the case of stationary nodes. Variations of one or two orders of magnitude in the received power over wireless links over relatively short periods of time (of the order of seconds or less) are commonplace. Due to the transitional nature of wireless connectivity at received signal strength values in the [-95, -85] dBm range [4][1], such variations inevitably lead to unstable connectivity.

In this paper, connectivity refers to the ability of a given node to deliver data to the sink. If a node has poor connectivity, it means that there exist no stable paths between the node and the sink (and therefore, no stable links to its neighbors). If a node has good connectivity, the opposite is true: there exists at least one stable path between the node and the sink. A stable path is a collection of stable links, and an individual link is considered stable if it has a Required Number of Packets (RNP) [5] below 2 (on average over the course of a whole experiment, less than 2 transmissions are required to get a packet across the link). We use the concept of connectivity because it is more general than other closely related concepts such as node density and node degree. A node can have good connectivity even if the network is not dense, and even if it has a low node degree. Another reason is that it is easy to quantify connectivity in terms of link stability.

We consider a wireless sensor network data collection scenario with N nodes attempting to deliver their data to a sink over the necessary number of hops (directly if possible, over multiple hops if needed). Running a tree collection protocol over a low-power link layer, we perform an experimental study of the interplay of low-power wireless connectivity and energy consumption. Duty-cycling is essential to low-power wireless: without it, network lifetime will be measured in days as opposed to months or years [6]. At the same time, duty-cycling has two side effects:

- It makes it harder to communicate over lossy links, because it may prevent nodes from exploiting tidbits of fleeting connectivity. If nodes have poor connectivity as a baseline, sleeping may cause them to miss out on connectivity opportunities.
- It increases the cost of broadcast traffic with respect to unicast traffic, because broadcasts are significantly longer than unicasts and therefore use up more energy. When radios are duty cycled, one physical transmission cannot be expected to reach all neighboring nodes, since their radios are switched off most of the time. To send a broadcast transmission, the sender needs to make sure that all its neighbors are awake to receive the broadcast transmission, and must therefore engage in a longer transmission [6].

These are certainly minor matters compared to the enormous benefits of duty cycling, but their impact is unclear, and our goal is to quantify it. As it has been shown in [7], the radio duty cycle correlates well with the energy consumption; therefore, we estimate the energy consumption by measuring the duty cycle of the nodes through online software-based estimation [8]. After investigating the performance of a collection protocol in duty-cycled networks in Section 2, we employ the protocol as a tool to study the impact of poor connectivity on the energy footprint of unicast traffic in Section 3 and the impact of high connectivity on the energy footprint of broadcast traffic in Section 4.

2 Collection Performance in Duty-Cycled Networks

To illustrate how collection behaves on top of a duty-cycled link layer, we begin by showing the results of an extensive set of experiments carried out on MoteLab [9] and Twist [10] using the Arbutus collection routing protocol [11]. We run Arbutus on top of a Low-Power-Listening (LPL) [12] link layer called BoX-MAC [13]. With BoX-MAC, which is informed with the basic principles of B-MAC [12] and X-MAC [14], all nodes are duty-cycled; the minimum on time is dictated by the minimum time required by the radio to sample the medium, while the theoretical sleep time is defined by the programmer. In the absence of traffic, all nodes transition between a radio sleep state, in which they remain for the duration of the prescribed sleep time, and a radio on state, in which they remain for the minimum on time. Whenever a node wishes to perform a unicast transmission, it unicasts a train of copies of the outgoing packet (in principle, a long strobed preamble) over a time window set to slightly exceed the duration of the sleep time. The train of copies is cut short as soon as a link layer acknowledgment is received. In the case of broadcast traffic, the duration of the train of copies of the outgoing packet must necessarily exceed the duration of the sleep time to ensure that all nodes within radio range have a chance to receive the outgoing packet. For this reason, broadcasts take longer than unicasts and are inherently more energy-costly.

Arbutus [11] is a cost-based collection routing protocol that employs various mechanisms to boost reliability under tough network conditions. Most notably, it uses a hybrid form of link estimation [15] that merges Channel State Information (CSI) with data-driven feedback and sets up a cost field to make routing decisions that account for the vagaries of real-world low-power wireless propagation [1]. CSI is derived from control beacons that are periodically broadcast by each node, while data-driven feedback is obtained in the form of RNP.

We assume a standard collection application whereby each node injects data packets at a constant Inter-Packet Interval (IPI). Because our focus is on low data rate operation, we use an IPI of 5 minutes. For BoX-MAC, we choose a sleep time of 1 second. We choose this operating point to replicate one of the low data rate setups in [16].

Arbutus uses a fixed Inter-Beaconing Interval (IBI) with on-demand extra control traffic on top: the inter-beacon interval always stays the same, and the

extra traffic is injected only when needed. A fixed beaconing interval is, in general, not as efficient as an adaptive beaconing interval [17], but it is useful for our purposes because we can span the control overhead space by modifying the beaconing interval. Specifically, we focus on a high overhead point at IBI=IPI and a low overhead point at IBI=20 IPI (100 minutes), chosen (arbitrarily) to exceed the upper limit (one hour) of the Trickle adaptive beaconing scheme used by CTP. Most of our low overhead runs are from Twist because of its looser constraints on the usage time windows. In Twist we also span the transmit power space by using the highest transmit power setting of the CC2420 (0 dBm) and its lowest setting (-25 dBm). On MoteLab, all experiments are run at 0 dBm because of the relatively low density of the testbed (lower transmit power settings are conducive to network partitioning).

For each experiment, we capture the network connectivity by employing the methodology in [18], which consists in measuring the connectivity matrix of the network *in vivo*, *i.e.*, as it is being used by a protocol under test. Passive network measurements are taken based solely on the protocol's own traffic in order to estimate the expected path delivery from each node to the sink (based on optimal routing choices that maximize the overall delivery to the sink). We employ the principal metric from [18], the Expected Network Delivery (END), to distil the network topology conditions estimated during each run as the average expected path delivery. The END is computed as the average expected path delivery (over all nodes). In essence, the END quantifies the connectivity over the achievable paths to the sink focusing on the key links that keep the network connected and neglecting the redundant links that might as well not exist. The rationale behind the END is that not all links were created equal: a few are essential to keep the network connected, while most of them are redundant. Networks whose key links are stable result in high END topologies, while networks whose key links are volatile and unstable result in low END topologies. Note that the study in [18] only considers an always-on link layer, while in this paper we focus on a duty-cycled link layer. The END is a normalized value within [0, 1]; a network whose key links are challenged will have a low END value, while a network with good key link connectivity will have a high END value. The END makes it possible to tease out the impact of the topology from the impact of the protocol. If the END is low, then we know that the topology is hostile, and we cannot draw any conclusions regarding the quality of the protocol under test. If the END is high, then we know that the topology is benign; if the protocol does not perform as expected, there must be something wrong in the protocol itself.

Twist has a regular grid-like topology and a much higher node density than MoteLab. According to our measurements, the average number of neighbors in Twist is more than four times the number in MoteLab, and the poorest sink assignment in Twist typically has 30% more neighbors with respect to the most well-connected MoteLab sink. Due to Twist's dense connectivity, the sink placement in Twist does not affect the network topology to a significant extent, and the END is therefore generally insensitive to the sink placement: all the sink placements in the Twist experiments resulted in high-end END ranges. On the

other hand, because of its complex layout and sparser connectivity, MoteLab displays the same dichotomy between high-performing (high END) and poorly-performing (low END) topologies observed in the always-on experiments in [18].

The LPL-based nature of the BoX-MAC link layer requires node to send out trains of packets whose length must be matched to the sleep time as a baseline but is cut short upon receipt of an ACK, as in X-MAC [14]. Because on average individual transmissions take much longer than in the case of an always-on link-layer, low-power operation implies more contention for reduced channel access. Even for topologies with an END above 0.9, low power operation increases the link layer failure rate and takes a toll on reliability. The packet loss averaged over all our runs is 2%, which compares favorably with the 4.9% reported in Table 4 of [16] that corresponds to the same IPI/sleep operating point on MoteLab (CTP's packet loss is higher due to its finite number of retransmissions and the presence of connectivity outliers).

Testbed	END Range	Packet Loss		Goodput	Hops		Cost		Delay		Duty Cycle	
				[pkts/sec]					[s]		[%]	
		mean	σ	mean	mean	σ	mean	σ	mean	σ	mean	σ
MoteLab	[0, 0.7)	0.22	0.13	1.7e-3	3.0	0.7	4.7	1.0	37.1	18.8	6.1	2.8
MoteLab	[0.7, 1]	0.02	0.01	3.1e-3	2.8	0.5	3.7	0.6	4.6	3.9	4.3	1.1
Twist	[0.8, 0.9)	6.7e-3	3.6e-3	3.0e-3	2.3	0.8	2.6	0.9	3.4	3.1	2.5	0.5
Twist	[0.9, 1]	7.7e-3	6.0e-3	3.1e-3	1.8	0.6	2.1	0.7	1.2	2.6	6.5	2.9

Table 1. Performance of Arbutus with BoX-MAC (LPL) at IPI=300s on MoteLab and Twist averaged over different END intervals. Due to Twist's regular topology and dense connectivity, all the sink placements that we used within Twist result in high END topologies.

Table 1 reports a comprehensive breakdown of our results for specific END ranges. We note a large discrepancy between the duty cycles on MoteLab and Twist, also evident in Figure 1, which shows the END vs. the duty cycle. While the mean duty cycle in MoteLab is 4.3% in the high END regime and 6.1% in low END regime, the mean duty cycle in the Twist runs (all at high END regime) is as high as 5.8%. In MoteLab, a higher END means a lower duty cycle; the duty cycles in high-END Twist, however, are much closer to MoteLab's low-END duty cycles than they are to MoteLab's high-END duty cycles. The reason for this lies in the interplay between the topology and the overhearing effect, a major duty cycle driver at ultra-light offered load points (such as our operating point, IPI=300s).

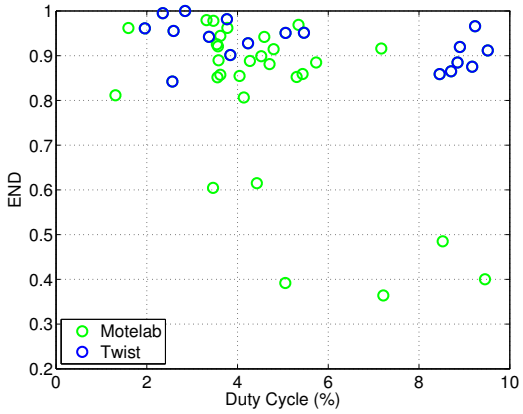


Fig. 1. END vs. duty-cycle in the MoteLab and Twist experiments with BoX-MAC. In MoteLab, a high END means a low duty-cycle, and vice-versa. In Twist, however, there is no correlation between the END and the duty-cycle.

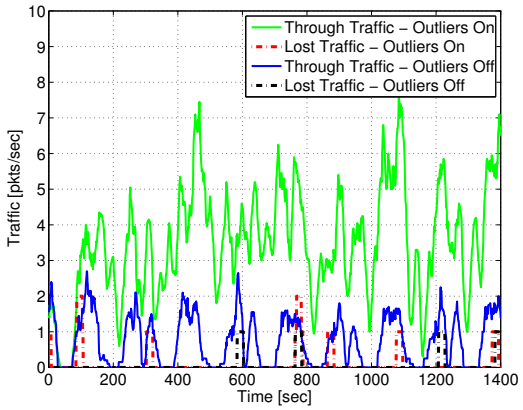


Fig. 2. A handful of connectivity outliers drastically increase the through traffic (and the packet loss) in MoteLab.

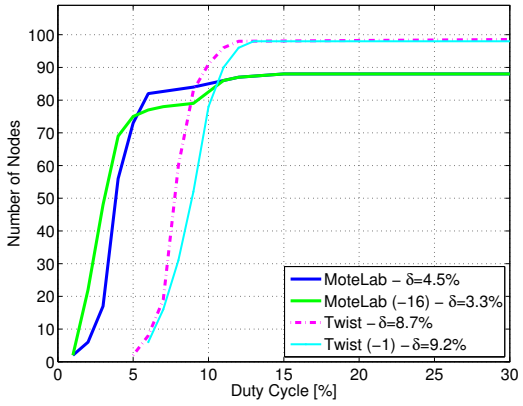


Fig. 3. The empirical CDF of the duty cycle for the two runs in Figure 2 confirms the impact of the outliers in MoteLab, and the empirical CDF for two Twist runs shows that even a single outlier can have a huge energy impact.

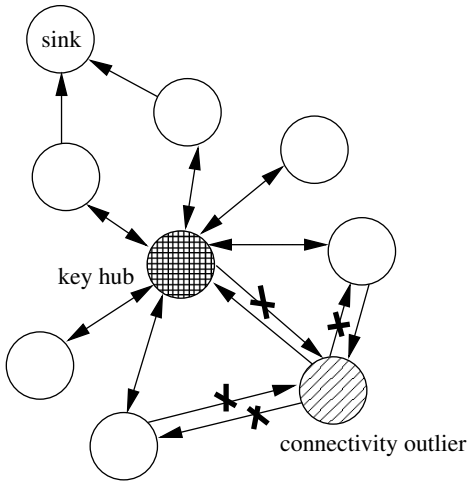


Fig. 4. A poorly connected leaf can operate the equivalent of a denial-of-sleep attack to the victim, which may be a well-connected hub. The crosses indicate the low-PRR links.

3 Poor Connectivity and Energy Consumption

Nodes with no stable links to the rest of the network represent connectivity outliers. Their poor connectivity has a significant energy footprint, particularly in the presence of unicast traffic. When dealing with unicast traffic, such nodes may require numerous retransmissions to communicate with any of their radio neighbors. To study the impact of connectivity outliers on the energy footprint of unicast traffic, we performed a set of runs on Motelab with a high END sink assignment (node 114). We singled out the connectivity outliers based on their average RNP: nodes that required, on average, at least 10 transmissions per successful delivery were labeled as outliers. Figure 2 shows the overall amount of traffic per time unit (packets/sec) that is sent over the air in the MoteLab network with and without the connectivity outliers. Figure 2 also displays the lost traffic, *i.e.*, the injected packets that are dropped due to a link layer failure. As shown in [11], Arbutus eliminates all causes of packet loss [19] other than link layer failures. The difference in the total traffic through the network caused by the 16 outliers is impressive: with the outliers there are 12.8 injected packets for every successful delivery, while without them there are only 3. Without the outliers the total network traffic decreases drastically, and the packet loss also decreases by almost 50% (from 1.73% to 0.92%): the outliers make link layer failures much more likely.

The energy impact of the outliers is also very large, as the duty-cycle distribution for the two runs shows (Figure 3). The outliers are responsible for an average increase of the duty-cycle of more than 30%. The average duty-cycle of the outliers is 6.6%, while the average duty-cycle of the rest of the nodes in the presence of the outliers is 4.17%, which is an extra 24% with respect to their average duty cycle in the absence of the outliers. The implication is that the outliers, in spite of their apparently harmless leaf status, drain a quarter of the energy reserve of their peers.

By inspecting our logs we found that what makes leaf outliers harmful is predominantly outbound loss: the outliers flood their upstream peers with duplicates. The poorly connected leaves keep their one-hop neighbors listening, thus creating the equivalent of a denial-of-sleep attack. This is particularly detrimental in the likely event that one of the neighbors is a well-connected hub or a bottleneck, as is the case in the example in Figure 4.

We also found a leaf outlier in Twist and turned it on and off to isolate out its impact. Figure 3 shows the empirical distribution of a Twist run with the outlier on and one without it (-1), and we find that a single outlier boosts the duty cycle by over 5%.

Our experimental evidence makes it clear that connectivity outliers can cause much harm. Nevertheless, because outliers with predominantly outbound loss typically have very high duty cycles, a node can easily self-diagnose its outlier status. Including a drastic measure such as a duty cycle-based auto-shutdown mechanism in a networking protocol for sensor networks may be an advantageous tradeoff: the outlier is sacrificed to extend the lifetime of its neighbors. Of course, this is only possible if the outlier is a leaf and no other node depends on it. It is

worth noting that have observed examples of low-END networks where poorly connected nodes serve as relays because, despite their insufficient connectivity, they still offer better connectivity than their peers. These extreme scenarios suffer from a general lack of physical connectivity, and in these cases the only viable option is to deploy additional nodes to improve the general connectivity of the network.

Testbed	Power	Overhead	Packet Loss	Hops	Cost	Delay		Duty Cycle
						μ[s]	med.[s]	[%]
MoteLab (high END)	high	low	5e-3	2.4	2.9	9.7	0.2	1.4
MoteLab (high END)	high	high	0.02	2.8	3.7	4.3	0.9	4.4
Twist	high	low	2.4e-3	1.5	1.6	0.14	0.03	2.9
Twist	high	high	5.6e-3	1.4	1.7	0.93	0.03	9.0
Twist	low	low	9.1e-3	2.7	3.0	3.5	0.3	2.4
Twist	low	high	0.017	2.7	3.2	3.1	0.3	3.9

Table 2. Performance breakdown in the dimensions of transmit power (high corresponds to 0 dBm and low corresponds to -25 dBm) and overhead (high corresponds to IBI=IPI and low corresponds to IBI=20 IPI). All the runs in this table are in the END range [0.8, 1].

4 High Connectivity and Energy Consumption

We have seen that connectivity outliers take a toll on the network's energy reserve by affecting the energy footprint of unicast traffic. At the other end of the connectivity spectrum, we find that dense connectivity and redundant links affect the energy footprint of broadcast traffic.

To study the effect of dense connectivity on the energy footprint of the broadcast traffic, we vary the transmit power: lowering the transmit power reduces the number of available links in the network. In Twist, our measurements indicate that, when the transmit power is reduced from 0 dBm to -25 dBm, the average number of stable neighbors drops by 75%. Denser or sparser does not necessarily mean higher END or lower END, because the END captures the availability of the key links: a sparse network whose key links are solid may very well have the same END as a very dense network. A lower transmit power typically means longer paths and higher costs (more transmissions per successful delivery), but the performance can be expected to be stable if the END remains high in spite of the transmit power reduction. On the other hand, the extra energy required by the longer paths may very well be compensated for by the fact that the nodes have fewer neighbors and are less exposed to overhearing, which has a direct

impact on the LPL duty cycle. For this reason, although the reduction of the transmit power does not yield proportional energy savings on the transmitter side [20], it may be conducive to significant energy savings on the receiver side. Given the IPI, the overhearing is mostly driven by the control overhead, also because the broadcast nature of control traffic makes it more energy-costly as a baseline [6].

Table 2 provides a breakdown of the parameter space that we explored in our experiments in the two dimensions of transmit power (high at 0 dBm and low at -25 dBm) and control overhead (high at IBI=IPI and low at IBI=20 IPI); the MoteLab results are only broken down in the dimension of control overhead. The Twist experiments were performed using two different transmit power settings (high at 0 dBm and low at -25 dBm) and beaconing rates (high at IBI=IPI and low at IBI=20 IPI). The MoteLab experiments were all performed using one power setting (high at 0 dBm) and only explore different beaconing rates (the same two rates as in the Twist runs). All MoteLab experiments are run at 0 dBm because of the relatively low density of the testbed (lower transmit power settings are conducive to network partitioning). We put the MoteLab results into the proper topological context by leveraging the methodology in [18]: in order to ensure a fair comparison between the MoteLab and the high-END Twist runs, we separate out the MoteLab runs based on the END in Table 2.

The results in Table 2 provide a clear indication of the interplay of overhearing, contention, and connectivity. A higher beaconing rate means higher overhead, more overhearing, and more contention, while a higher transmit power means more connectivity but also more overhearing and more contention.

In MoteLab, at high transmit power, the impact of the control overhead (beaconing rate) is significant: the duty cycle ranges from just 1.4% at low overhead to over three times as much at high overhead and almost five times as much at high overhead and low END. Aside from the duty cycle, the packet loss, the cost, and the delay also deteriorate at high overhead. In particular, the packet loss is four times higher at high overhead than it is at low overhead, and the reason for this is that the extra broadcast traffic competes with data traffic for channel access (a scarce resource in low-power operation) and makes link-layer failures more likely. In Twist, at high transmit power the impact of the control overhead is also considerable: the duty cycle ranges from just below 3% at low overhead to as much as 9% at high overhead. At high overhead, the packet loss doubles but the other performance dimensions do not suffer considerably (the mean delay does, but the median delay remains constant). At low transmit power, the performance is not sensitive to the overhead in the dimensions of packet loss, cost, and delay; the impact of the overhead on the duty cycle is relatively small compared to its impact at high transmit power. The joint effect of the transmit power and the overhead is the main driver of energy consumption through overhearing at high END. This result suggests that more connectivity is not necessarily better. A lower transmit power typically reduces the connectivity and increases the path length, and therefore the number of transmissions per delivery. However, it also reduces contention and overhearing, resulting in

a lower duty-cycle because overhearing dominates the energy consumption by the radio on a node, even with a low-power-listening link layer. Control overhead over stable routes is not only unnecessary, but also detrimental in terms of energy and reliability.

The side effects of Twist's denser connectivity, exacerbated by the control overhead, are the reason behind the high duty-cycles recorded at high END values in Figure 1. Indeed, Table 2 shows that the duty cycle is highest in the runs at high transmit power and high control overhead, while relatively normal (below 5%) in all other runs.

5 Related Work

In recent years, low-power wireless connectivity has been the object of extensive investigations. Transitional connectivity has been studied in [4], and its protocol-level implications have been investigated in [21] and [22]. The methodology from [18] that we employ to compute the END value that characterizes the network topology during our experiments is part of the body of work for the characterization of the topology of low-power wireless. The work in [18] is significantly different from other studies that also pursue a low-power *wireless lexicon* [23], such as [24] and [25], because it addresses the *in vivo* characterization of the network topology as the network is in use. Collection protocols have been the object of several investigations [21][16][11]. The behavior of collection on top of a duty-cycled link layer has been studied in [26], [16], [27], [28], and [29], and the energy footprint of collection is the specific focus of [7]. The energy footprint of broadcast in the presence of duty-cycling is illustrated in [6], and the relative energy footprints of broadcast and unicast traffic are discussed in [30].

6 Conclusion

We have examined the behavior of a collection protocol on top of a duty-cycled link layer, and we have investigated the energy footprint of poor connectivity as well as redundant connectivity. Poor connectivity primarily affects unicast traffic (data traffic). Not only do connectivity outliers drain their own resources, but they also drain the resources of their neighbors. The impact of poor connectivity may be mitigated, but only with drastic measures, such as auto-shutdown mechanisms for nodes that self-diagnose their connectivity outlier status.

Overly redundant connectivity also has a significant energy footprint, because it increases the cost of broadcast traffic (control traffic). The impact of excessive connectivity may be mitigated with transmission power control and the aggressive reduction of unnecessary control traffic.

Acknowledgements

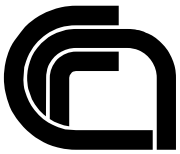
This work was partially supported by the European Commission under the SCAMPI Project (ICT grant agreement 258414). The authors wish to thank Omprakash Gnawali for his insightful comments and suggestions.

References

1. K. Srinivasan, P. Dutta, A. Tavakoli, and P. Levis. An Empirical Study of Low-Power Wireless. *ACM Transactions on Sensor Networks (To appear)*, 2010.
2. D. Puccinelli and M. Haenggi. Multipath Fading in Wireless Sensor Networks: Measurements and Interpretation. In *International Wireless Communications and Mobile Computing Conference (IWCMC'06)*, Vancouver, BC, Canada, July 2006.
3. D. Puccinelli and M. Haenggi. Spatial Diversity Benefits by Means of Induced Fading. In *Third IEEE International Conference on Sensor and Ad Hoc Communications and Networks (SECON'06)*, Reston, VA, USA, September 2006.
4. M. Zuniga and B. Krishnamachari. An Analysis of Unreliability and Asymmetry in Low-Power Wireless Links. *ACM Transactions on Sensor Networks*, 3(2):1–30, 2007.
5. A. Cerpa, J. Wong, L. Kuang, M. Potkonjak, and D. Estrin. Statistical Model of Lossy Links in Wireless Sensor Networks. In *6th ACM international Symposium on Mobile Ad Hoc Networking and Computing*, Urbana-Champaign, IL, USA, May 2005.
6. A. Dunkels, L. Mottola, N. Tsiftes, F. Österlind, J. Eriksson, and N. Finne. The Announcement Layer: Beacon Coordination for the Sensornet Stack. In *European Conference on Wireless Sensor Networks (EWSN'11)*, Bonn, Germany, February 2011.
7. M. Martins, R. Fonseca, T. Schmid, and P. Dutta. Poster Abstract: Network-Wide Energy Profiling of CTP. In *8th ACM Conference on Embedded Networked Sensor Systems (SenSys'10)*, Zurich, Switzerland, November 2010.
8. A. Dunkels, F. Osterlind, N. Tsiftes, and Z. He. Software-based on-line energy estimation for sensor nodes. In *Proceedings of the Fourth Workshop on Embedded Networked Sensors (Emnets IV)*, Cork, Ireland, June 2007.
9. G. Werner-Allen, P. Swieskowski, and M. Welsh. MoteLab: a Wireless Sensor Network Testbed. In *4th International Symposium on Information Processing in Sensor Networks (IPSN'05)*, Los Angeles, CA, April 2005.
10. V. Handziski, A. Koepke, A. Willig, and A. Wolisz. TWIST: A Scalable and Reconfigurable Testbed for Wireless Indoor Experiments with Sensor Networks. In *2nd International Workshop on Multi-hop Ad Hoc Networks: from Theory to Reality (REALMAN'06)*, Florence, Italy, 2006.
11. D. Puccinelli and M. Haenggi. Reliable Data Delivery in Large-Scale Low-Power Sensor Networks. *ACM Transactions on Sensor Networks*, Jul. 2010.
12. J. Polastre, J. Hill, and D. Culler. Versatile Low Power Media Access for Wireless Sensor Networks. In *2nd ACM Conference on Embedded Networked Sensor Systems (SenSys'04)*, Baltimore, MD, November 2004.
13. D. Moss and P. Levis. BoX-MACs: Exploiting Physical and Link Layer. Technical Report 08-00, Stanford University, 2008.

14. M. Buettner, G. Yee, E. Anderson, and R. Han. X-MAC: a short preamble MAC protocol for duty-cycled wireless sensor networks. In *4th ACM Conference on Embedded Networked Sensor Systems (SenSys'06)*, Boulder, CO, USA, November 2006.
15. D. Puccinelli and M. Haenggi. DUCHY: Double Cost Field Hybrid Link Estimation for Low-Power Wireless Sensor Networks. In *Fifth Workshop on Embedded Networked Sensors (HotEmNets'08)*, Charlottesville, VA, USA, 2008.
16. O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis. Collection Tree Protocol. In *7th ACM Conference on Embedded Networked Sensor Systems (SenSys'09)*, Berkeley, CA, November 2009.
17. P. Levis, N. Patel, D. Culler, and S. Shenker. Trickle: A Self-Regulating Algorithm for Code Propagation and Maintenance in Wireless Sensor Networks. In *First USENIX/ACM Symposium on Networked Systems Design and Implementation (NSDI'04)*, San Francisco, CA, March 2004.
18. D. Puccinelli, O. Gnawali, S. Yoon, S. Santini, U. Colesanti, and L. Guibas. The Impact of Network Topology on Collection Performance. In *8th European Conference on Wireless Sensor Networks (EWSN'11)*, Bonn, Germany, February 2011.
19. M. Wachs, J. Choi, J. Lee, K. Srinivasan, Z. Chen, M. Jain, and P. Levis. Visibility: A New Metric for Protocol Design. In *5th ACM Conference on Embedded Networked Sensor Systems (SenSys'07)*, Sydney, Australia, November 2007.
20. M. Haenggi. The Impact of Power Amplifier Characteristics on Routing in Random Wireless Networks. In *IEEE 2003 Global Communications Conference (GLOBECOM'03)*, San Francisco, CA, December 2003.
21. A. Woo, T. Tong, and D. Culler. Taming the Underlying Challenges of Reliable Multihop Routing in Sensor Networks. In *1st ACM Conference on Embedded Networked Sensor Systems (SenSys'03)*, Los Angeles, CA, November 2003.
22. O. Gnawali, M. Yarvis, J. Heidemann, and R. Govindan. Interaction of Retransmission, Blacklisting, and Routing Metrics for Reliability in Sensor Network Routing. In *First IEEE Conference on Sensor and Ad Hoc Communication and Networks (SECON'04)*, pages 34–43, Santa Clara, CA, October 2004.
23. K. Srinivasan. *Towards a Wireless Lexicon.* PhD thesis, Stanford University, September 2010.
24. K. Srinivasan, M. Kazandjieva, S. Agarwal, and P. Levis. The Beta-Factor: Improving Bimodal Wireless Networks. In *6th ACM Conference on Embedded Networked Sensor Systems (SenSys'07)*, Raleigh, NC, November 2008.
25. K. Srinivasan, M. Jain, J. Choi, T. Azim, and E. Kim. The Kappa Factor: Inferring Protocol Performance Using Inter-link Reception Correlation. Technical Report 09-02, Stanford University, 2009.
26. L. Filipponi, S. Santini, and A. Vitaletti. Data Collection in Wireless Sensor Networks for Noise Pollution Monitoring. In *International Conference on Distributed Computing in Sensor Systems (DCOSS'08)*, Santorini, Greece, June 2008.
27. L. Mottola, G. P. Picco, M. Ceriotti, S. Guna, and A. Murphy. Not All Wireless Sensor Networks Are Created Equal: A Comparative Study On Tunnels. *ACM Transactions on Sensor Networks*, 7, August 2010.
28. J. Vanhie-Van Gerwen, E. De Poorter, B. Latré, I. Moerman, and P. Demeester. Real-Life Performance of Protocol Combinations for Wireless Sensor Networks. In *IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC'10)*, Newport Beach, CA, USA, June 2010.
29. D. Puccinelli and S. Giordano. Duty-Cycle Distribution in Low-Power Collection. In *European Conference on Wireless Sensor Networks (EWSN'10)*, Coimbra, Portugal, February 2010.

30. G. Halkes and K. Langendoen. Practical considerations for wireless sensor network algorithms. *Wireless Sensor Network*, 2(6):441–446, June 2010.



Consiglio Nazionale delle Ricerche

Detecting users' communities in mobile social networks

E. Borgia, M. Conti, A. Passarella

IIT TR-19/2011

Technical report

settembre 2011

Istituto di Informatica e Telematica

Detecting users' communities in mobile social networks

Eleonora Borgia, Marco Conti, Andrea Passarella
Institute of Informatics and Telematics (IIT)- CNR
via G. Moruzzi,1 - 56124 Pisa, Italy
email: {*e.borgia, m.conti, a.passarella*}*@iit.cnr.it*

Abstract

In this paper we focus on approaches which aim at discovering communities of people in Opportunistic Networks. We first study the behaviour of three community detection distributed algorithms proposed in literature [1], in a scenario where people move according to a mobility model which well reproduces the nature of human contacts, namely HCMM [2]. By a simulation analysis, we show that these distributed approaches can satisfactory detect the communities formed by people only when they do not significantly change over time. Otherwise, as they maintain memory of all encountered nodes forever, these algorithms fail to capture dynamic evolutions of the social communities users are part of. To this aim we propose AD-SIMPLE, a new solution which captures the dynamic evolution of social communities. By an extensive simulation analysis, we demonstrate that it accurately detects communities and social changes while keeping computation and storage requirements low.

I. Introduction

Opportunistic Networks [3] are self-organising wireless networks formed by mobile devices carried by people. Due to people mobility, the network is disconnected most of the time and the existence of a complete path between pairs of senders/destinations cannot be assumed. However, communications among nodes are still possible since local interactions are exploited to deliver messages in a hop-by-hop fashion. This can be achieved by means of *i)* dissemination-based routing protocols, which essentially implement form of controlled flooding [4], or *ii)* context-based routing protocols, which aim at maximising the probability of message delivery searching for the most appropriate relay node. The latter category exploits information about the context which are used to construct the so-called *utility* of a node, i.e., the usefulness of a node to be the best next-hop for a message. To this end, several schemes have been proposed taking into account the history of encounters [5] [6], or the mobility [7] or a combination of attributes [8].

Among the various classes of context-based routing protocols, those exploiting information about social relationship between the users are considered very promising. Humans are social individuals that have social ties and form communities to better answer to their needs. People belonging to the same community meet with high probability and regularly. On the contrary, people from different communities meet less frequently. Therefore, understanding the human structure and the rules which regulate social interactions and aggregations can be a great advantage. An accurate knowledge of local community can be exploited to increase the performance of forwarding schemes for Opportunistic Networks, as shown for example

in HiBOp [8] and BubbleRAP [9]. As users belonging to the same community share common interests, social information can be efficiently used to design smart strategies to choose the most suitable next hop.

Complex network analysis has been recently proposed as a efficient way to detect the structure of a network, and indices such as betweenness, similarity and centrality are defined as a compact representation of the properties of the nodes. The approaches presented in [10] [11] [12] show to reliably extract the community structure in real world networks by deriving statistics of communities from traces. They rely on centralised schemes which require the complete knowledge of the entire structure of the graph. Conversely, the approach proposed in [13] infers the community structure, named *local modularity*, without relying on some centralised points by exploring each vertex at a time. By exploiting such results, in [1] authors proposed the distributed version of three centralised community detection algorithms (i.e., SIMPLE, k-CLIQUE and MODULARITY) and show their great potential and their accuracy in identifying social community wrt the centralised versions.

In this paper, we focus on approaches aimed at discovering dynamic communities of people in an opportunistic scenario by starting from the three distributed algorithms proposed in [1]. We can summarise our major contributions as follows:

i) We compare and contrast them in different scenarios in terms of the achieved *similarity* metric, i.e., how similar the detected communities are with respect to those detected by the centralised algorithms. We differentiate from [1] in the type of traces which are used for running the analysis. We use synthetic traces generated from a mobility model which well reproduces human movement patterns observed in real traces, namely HCMM [2]. This allows us also to create different scenarios wrt those in [1] and thus to make a deeper analysis. Obtained results show that SIMPLE performs better with respect to k-CLIQUE and MODULARITY.

ii) We also demonstrate that their accuracy in identifying communities decreases with the increase of the complexity of the scenario. They are not able to correctly represent the dynamic users' social behaviour in scenarios with nodes moving across different communities.

iii) We finally propose a new community detection algorithm, namely Adaptive Detection SIMPLE (AD-SIMPLE), which is able to capture the evolution of social communities in dynamic scenarios, while keeping computation and storage requirements low. To the best of our knowledge this is the first attempt proposed in literature that explicitly includes *adaptive* community detection mechanisms.

The paper is organised as follows. Section II states the problem to be faced, introducing the community detection algorithms defined in [1] and the mobility model assumptions. Section III presents the simulation analysis to compare the distributed algorithms proposed in [1], and discusses the obtained results. Section IV introduces the proposed Adaptive Detection SIMPLE algorithm, while in Section V we provide an extensive evaluation of the proposed solution by means of simulation. Specifically, first, we show the sensitiveness of AD-SIMPLE on specific parameters, then we present results of a comparison of AD-SIMPLE against that of the algorithms in [1] in case of dynamic social communities. Finally, Section VI concludes the paper.

TABLE I
NOTATION

$F_0(F_i)$:	Familiar Set of node $v_0(v_i)$
$C_0(C_i)$:	Local Community of node $v_0(v_i)$
$\tilde{F}_j$:	local approximation of the Familiar Set of node $v_j \in C_0$
$FSoLC_0$:	local approximation of the Familiar Sets of all vertices in C_0 ($FSoLC_0 = \{\tilde{F}_j \mid v_j \in C_0\}$)

II. Distributed Communication Detection algorithms

In this section we provide the background information necessary to understand the three distributed algorithms which are used in the simulation analysis to detect social communities. We first provide some basic definitions and then we report the main characteristics of each algorithm. Interested readers can refer to [1] for additional details.

A. Definitions

The following definitions are common to all the three algorithms:

- **Familiar Set (F):** a familiar set of a node v_0 is composed by all those encountered nodes for which the cumulative contact duration[1] exceeds a predefined threshold t_{th}.
- **Local Community (C):** a local community of a node v_0 is composed by all the nodes in its familiar set and the nodes for which criteria, specific to each algorithm, are valid.

Table I summarises the notation used hereafter.

The idea at the basis of all the three community detection algorithms is that each node determines the composition of its community based on the contacts it has with other nodes. This is achieved by storing specific information. Specifically, when two nodes meet each other, they first exchange local information and then take independent decisions about including or not the encountered node into the Familiar Set (i.e., *threshold criteria*) [2], or only into the Local Community (i.e., *admission criteria*). In addition, each node evaluates if its Local Community can be merged (partially or completely) with the Local Community of the encountered node (i.e., *merging criteria*).

The threshold criteria are common to all the the three algorithms.

Threshold Criteria: when two nodes meet, each node evaluates the cumulative contact duration t_{cum}. If $t_{cum} \geq t_{th}$, then the encountered node is added to the Familiar Set (and consequently to the Local Community).

On the contrary, the admission criteria and merging criteria are specific for each algorithm and they are carefully described in the next subsections.

[1]The cumulative contact duration for a pair of nodes (t_{cum}) is the sum of the duration of all the different contacts.

[2]When a node is added to the Familiar Set it is automatically added to the Local Community

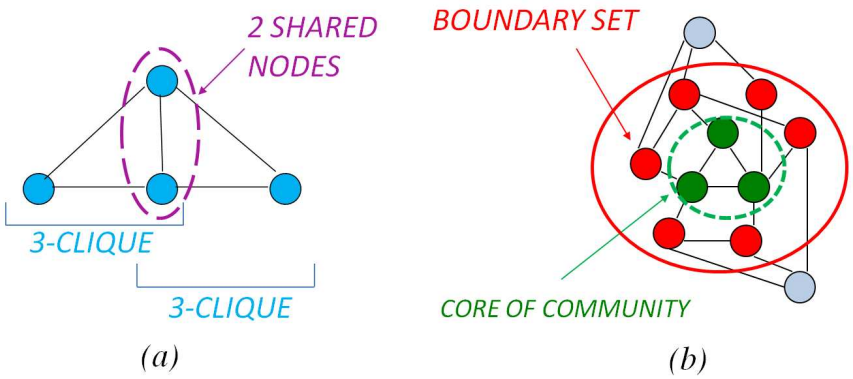


Fig. 1. (a) Example of k-CLIQUE with k=3 and (b) example of Boundary Set for the Modularity algorithm.

B. SIMPLE community detection algorithm

As suggested by the name, in SIMPLE nodes exchange very few information. Suppose that two nodes, e.g., v_0 and v_i, make contact. During the contact duration, each node sends to the other node its Familiar Set and its Local Community. At the end of the contact, each node evaluates the following two criteria to update the relative data structures. Focusing on v_0:

B1. **Admission Criteria:** in case of failure of the threshold criteria, if the number of nodes shared among C_0 and F_i is higher than λ times the number of nodes in F_i, then the encountered node v_i is added to the Local Community of v_0 (i.e., $if |C_0 \cap F_i| > \lambda \cdot |F_i| \Longrightarrow C_0 = C_0 \cup \{v_i\}$).

B2. **Merging Criteria:** in case v_i is added to the Local Community of v_0 as a consequence of the success of threshold or the admission criteria, if the number of nodes shared between C_0 and C_i is higher than γ times the number of nodes in the set union of C_0 and C_i, then the two Local Communities are merged (i.e., $if |C_0 \cap C_i| > \gamma \cdot |C_0 \cup C_i| \Longrightarrow C_0 = C_0 \cup C_i$).

C. k-CLIQUE community detection algorithm

k-CLIQUE is the distributed version of the centralised method originally proposed in [10]. Authors define a community as the union of all *k*-cliques which can be reached from each other through a series of adjacent *k*-cliques, where a *k*-clique is a complete subgraph of size *k* while two *k*-cliques are adjacent if they share $k-1$ nodes. An example of a 3-clique community is provided in Figure 1a.

In addition to the Familiar Set and Local Community, nodes exchange also their local approximation of the Familiar Sets of all the nodes within their Local Community ($FSoLC$)[3]. Focusing on node v_0, the following two criteria are evaluated at the end of the contact:

C1. **Admission Criteria:** in case of failure of the threshold criteria, if the Familiar Set of v_i contains at least $k-1$ nodes of the Local Community of v_0, then the encountered node v_i is added to the Local Community of v_0 (i.e., $if |C_0 \cap F_i| \geq k-1 \Longrightarrow C_0 = C_0 \cup \{v_i\}$).

C2. **Merging Criteria:** in case v_i is added to the Local Community of v_0 as a consequence of the success of threshold or the admission criteria, if the Familiar Set of each node inside the Local Community

[3]This is an approximation because it contains information about the Familiar Set of the encountered nodes which are collected during contacts and may be incomplete and not updated.

of v_i (e.g., v_j) contains at least $k-1$ nodes of the Local Community of v_0, then v_j is added to the Local Community of v_0 (i.e., $if|C_0 \cap \tilde{F}_j| \geq k-1 \Longrightarrow C_0 = C_0 \cup \{v_j\}$, where $v_j \in C_i$ and $\tilde{F}_j \in FSoLC_i$).

In addition, if the merging criteria are satisfied then the local approximation of the Familiar Sets of all nodes in C_0 needs to be updated, i.e., $FSoLC_0 = FSoLC_0 \cup \tilde{F}_j$.

D. MODULARITY community detection algorithm

MODULARITY is a variation of the method for discovering local community structures which has been presented in [13]. The original method is based on the concept of *Local Modularity (R)* and on the computation of the variation rate of the Local Modularity (ΔR) when adding a new vertex to an existing local community.

To define the Local Modularity the concept of *Boundary Set* should also be introduced. Specifically, the Boundary Set of a node (B) is defined to be the subset of vertices in a local community whose members have edges connecting to one or more vertices located outside the local community (see Figure 1b).

Taking into account the above definition, the Local Modularity measures the sharpness of local community boundary of each node and can be expressed by the quantity:

$$R = \frac{I}{|T|} \tag{1}$$

where I is the number of edges with no endpoints outside the Local Community and T is the set of edges with one endpoints in the Boundary Set. If the Boundary Set coincides with the Local Community, R is equal to 1 by definition.

When a new vertex v_i is added to an existing community C_0 of vertex v_0 with Boundary Set B_0, the variation of the Local Modularity can be computed by the following equation:

$$\Delta R_0 = \frac{x - R_0 \cdot y - z(1 - R_0)}{|T| - z + y} \tag{2}$$

where x is the number of edges in T that terminate in v_i, y is the number of edges that will be added to T due to the inclusion of v_i and z is the number of edges that will be removed from T due to the inclusion of v_i.

The variation to the Modularity algorithm proposed by [1] works as follows. When two nodes meet they exchange the following information: *i)* F, *ii)* C and *iii)* $FSoLC$. Once received, each node uses them to update its local structures. Specifically, v_0 first updates its local approximation of $\tilde{F}_i$ by merging it with F_i directly received by v_i. Then, it updates each local approximation of all the Familiar Sets in $FSoLC_0$ with the corresponding version in $FSoLC_i$. The same is for v_i as well. Afterwards each node evaluates the following criteria. Referring to node v_0:

D1. **Admission Criteria:** in case of failure of the threshold criteria, if the difference between the Local Modularity measured before and after including v_i into the Local Community of v_0 exceeds 0, then the encountered node v_i is added to the Local Community of v_0 (i.e., $if \Delta R_0 \geq 0 \Longrightarrow C_0 = C_0 \cup \{v_i\}$).

D2. **Merging Criteria:** in case v_i is added to the Local Community of v_0 as a consequence of the success of threshold or the admission criteria, the algorithm considers adding to C_0 the nodes inside the set K:

$K = \{v_k \mid \exists j s.t. v_j \in C_0 \cap C_i \wedge v_k \in \tilde{F}_j \wedge v_k \in C_i \setminus C_0\}$.

Specifically, the set K consists of the subset of all the nodes of C_i that are adjacent to those nodes shared between C_0 and C_i.

For each node v_k in K, if the Familiar Set of v_k is a subset of the Local Community of v_0, then v_k is added to the Local Community of v_0 (i.e., $if \tilde{F}_k \subseteq C_0 \Longrightarrow C_0 = C_0 \cup \{v_k\}$).

In addition, the variation of Local Modularity (ΔR_0) is computed for all the remaining nodes in K. Nodes with $\Delta R_0 > 0$ are directly added to the Local Community of v_0. For those nodes with $\Delta R_0 \leq 0$, the value of ΔR_0 is re-computed and re-checked after each inclusion. This procedure may be repeated several times and it ends when *i)* K is empty, or *ii)* after having added v_k to C_0, $\Delta R_0 \leq 0$ for all the remaining nodes in K.

Furthermore, after each inclusion, $\tilde{F}_k$ is merged with the corresponding version in $FSoLC_0$.

Note that the thresholds t_{th}, λ, γ and k used in the three algorithms are design parameters, thus they need to be chosen appropriately in order to calibrate the system. For this reason they are objects of investigation of the paper (see Section III-C).

As come out from the above description, the three community detection algorithms have different memory and computational requirements. On one hand, the SIMPLE algorithm makes use of low memory usage and low power computation. On the other hand, the MODULARITY algorithm is the most complex, both in terms of memory usage as it has to maintain a copy of Familiar Set of all the nodes in the Local Community and in terms of computation as it has to compute ΔR at each iteration for all the nodes belonging to set K. The k-CLIQUE algorithm represents an intermediate solution since it needs the same memory usage of MODULARITY but has lower computation complexity.

III. Performance Analysis

In this section we evaluate the communities detected by the distributed algorithms against the centralised algorithms, where a centralised algorithm has an a-priori knowledge of the composition of each community. We developed the described community detection algorithms as part of the OMNeT++ discrete-event network simulator[4]. The default scenario is composed of 54 nodes divided into 2 communities. Nodes move with an average speed of 1.5 m/s (representing a walking person) in a square of 1000mx1000m. Nodes move according to the HCMM mobility model [2] which is briefly described in the following subsection. The default parameters are set as in Table II. Despite being a particular and simple scenario, it is sufficient to highlight the properties of the community detection algorithms that we want to check.

A. Mobility Model: HCMM

HCMM [2] is a mobility model which well reproduces the statistical figures of real human movement patterns. Each node is initially associated with a specific community (its *home community*), and has

[4]http://www.omnetpp.org/

TABLE II
DEFAULT PARAMETERS

Parameter	Value	Parameter	Value
Area	1000mx1000m	N	54
Transmission range	20m	Community number	2
Average speed	1-1.86m/s	Traveller number	1

social ties with all the other members of its home community. Certain nodes (*travellers*) have also social links with communities other than the home (*foreign community*). For each social tie, the specific foreign community and the specific node inside the foreign community are selected according to a uniform distribution. The mobility pattern of a node is driven by its social links, i.e., a node located into its home community moves towards a given community (i.e., home and/or foreign) with a probability proportional to the number of ties with nodes of that community. In addition, when the node reaches a community which is not its home, it remains in the foreign community for the next movement with a given probability (p_e) and goes back home with probability $1 - p_e$. Hence, HCMM models a realistic scenario in which users are generally attracted to those people within the same home community, but they are also attracted to foreign people with whom they spend some time before coming back home.

HCMM permits to periodically re-select the foreign communities for the traveller.

B. Performance metrics

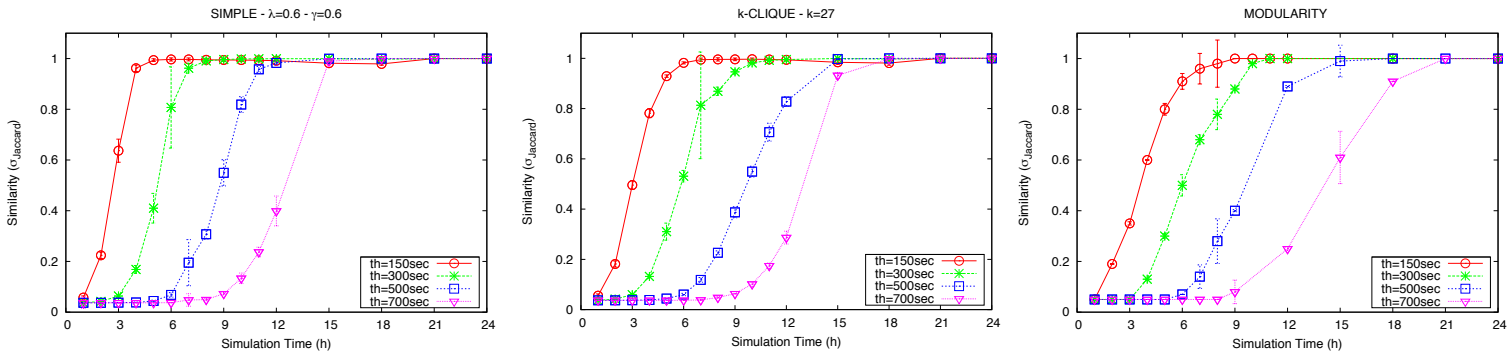
The objective of the simulation analysis is to measure how the distributed approaches proposed in [1] are able to identify communities formed by people. To this aim, we use the classic Jaccard index [14] as the metric to evaluate the similarity among the communities detected by the distributed algorithms and those detected by the centralised algorithms. The Jaccard index is defined by the following quantity:

$$\sigma_{Jaccard} = \frac{\left| G_i^{(c)} \bigcap G_i^{(d)} \right|}{\left| G_i^{(c)} \bigcup G_i^{(d)} \right|} \tag{3}$$

where G_i is the set of nodes which belongs to community *i* and $|G_i|$ is the cardinality of the G_i, while (c) and (d) indicate that the set is detected by the centralised algorithm and by the distributed algorithm, respectively. In the following we give an estimation of the local community similarity by averaging the Jaccard index for all the nodes. Specifically, each simulation is replicated 5 times and the results are averaged over all the replicas with 95% confidence intervals.

C. Results

The analysis presented in this section refers to the default scenario in Table II which is not subjected to any reconfiguration. Figure 2 shows the similarity metric as a function of simulation time and threshold value, respectively, for the three algorithms. Each curve represents, for a fixed threshold value, the corresponding $\sigma_{Jaccard}$ obtained at different points in time in the simulation. As depicted by the figure, the similarity value increases with the increase of simulation time for all the three algorithms. This is



(a) (b) (c)

Fig. 2. Impact of the Familiar Set threshold as function of simulation time for SIMPLE (a), k-CLIQUE (b) and MODULARITY (c).

quite obvious since increasing the simulation time corresponds to gather more precise information for the distributed algorithms.

The similarity also increases when the threshold value becomes lower. By reducing t_{th} the probability that the cumulative contact duration for pairs of nodes satisfies the corresponding criterion increases. The threshold should be also set taking into account the nature and the dynamic of the groups. If nodes have strong social interactions, the corresponding cumulative contact duration will be higher. On the other hand, if nodes have weak social interactions it is high probable that their cumulative contact duration remains low. A suitable tuning of the threshold value may guarantee better performance in terms of similarity. For example, we find out that, for a 6h simulation run, the average cumulative contact times are about 550 sec. In this case, appropriate values are 150 sec for k-CLIQUE and MODULARITY or even 300 sec for SIMPLE. By setting a threshold of 150 sec, all the three distributed algorithms can reach at least 95% of the performance of the corresponding centralised algorithm. In addition, SIMPLE is able to have good performance (around 80%) with a higher threshold (300 sec).

Figure 3 shows the similarity metric for SIMPLE when varying the admission and merging thresholds. The trend of curves is similar to figure 2(a). The similarity increases with the simulation duration as the system has more time to correctly detect the communities. However, a variation of λ and γ has less impact in the overall performance. There is a slight difference between curves, especially for values of λ and γ greater than 0.6. Note that similar trends are obtained for different values of t_{th}, but here they are omitted due to space reasons.

Concerning the k parameter of k-CLIQUE, note that it represents the largest community which we would like to detect. As a consequence, in this scenario it should be set to 27. Changing the scenario requires setting it to a different value.

A comparison among the three algorithms in terms of similarity is illustrated in Figure 4. It refers to a simulation running for six hours with the choice of parameters that optimises the behaviour of the three algorithms. We can see that k-CLIQUE and MODULARITY show almost similar performance while SIMPLE has better performance. This is more apparent when the threshold is set to 300 sec. In this case SIMPLE reaches 80% similarity while both k-CLIQUE and MODULARITY do not exceed 55%. However, the difference between the curves decreases for higher threshold values, where all the algorithms

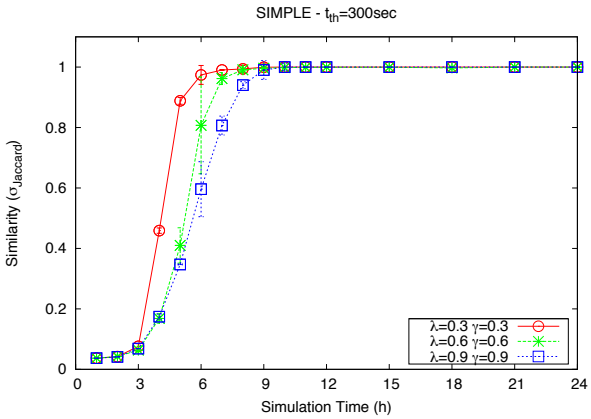


Fig. 3. Impact of λ and γ as function of simulation time for SIMPLE

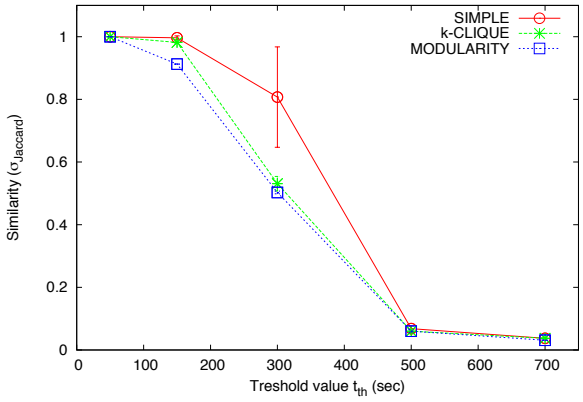


Fig. 4. Similarity as function of threshold value for SIMPLE(λ=0.6, γ=0.6), k-CLIQUE and MODULARITY

result in a very low performance. Note that the same line of reasoning can be applied for all the other investigated scenarios.

Summarising, from this analysis, SIMPLE results the more appropriate community detection solution as it is computationally lightweight and it generally performs better than the other two approaches.

IV. Towards an Adaptive Simple algorithm

Although the three presented algorithms are generally able to detect social communities, they suffer from a common limitation: during its whole life, each node (the traveller in particular) has not always a consistent view of its real familiar set and/or local community since it maintains memory of all encountered nodes.

For example, let's consider a scenario where a user A spends its time between two social communities: the work community during the day and the friend community after the working day. Let's also suppose that B is a colleague of A. Since A and B meet every day, it is correct to include B into the local community of A (and viceversa) for the time they meet at work. However, the contact with B becomes less relevant when A has finished to work and meets his friends for playing football, for example. This is even more apparent if B changes job or moves to another city. In above situations, no actions are made by the three algorithms to reflect these changes in the social community structures. On the contrary, B (A) will always remain in the A's community (B's community) even though they will not meet again.

From the above discussion, it turns out that the presented community detection algorithms need to be improved to well represent the dynamic users'social behaviours. This can be achieved by implementing some policies which take into account:

i) mechanisms for aging contacts among users;
ii) rules for deleting nodes from communities.

To this aim, in the following we present a novel social community detection algorithm, named **Adaptive Detection SIMPLE** (**AD-SIMPLE**), which is able to dynamically change social structures.

AD-SIMPLE takes inspiration from the original distributed SIMPLE. We focus on it since SIMPLE represents a good compromise among complexity and performance.

The main idea behind this algorithm is to maintain the basic features of the original SIMPLE for what concerns the inclusion of nodes while to improve its behaviour with additional policies in order to identify and remove from communities the nodes which have not be seen for a long period of time. Therefore, during a contact time, nodes exchange the Familiar Set and the Local Community and update their local view of social communities by applying the same criteria of SIMPLE (see section II-B). The additional policies to remove nodes from communities are carefully described in the following subsections.

A. Familiar Set pruning policy

The idea is to keep a running average of the percentage of the contact duration (over the total simulation time) for each node in the Familiar Set and to decide whether to remove or not a node if this percentage falls below a given threshold. To this aim, time is divided in slots (i.e., T) and the node computes the percentage of the contact duration in each slot for all the nodes in its Familiar Set as a $Sample\Delta T$. At the end of the slot, the node computes an $Estimated\Delta T$ as a weighted average value between the previous estimate and this new sample, as defined by equation 4:

$$Estimated\Delta T = \alpha \cdot Estimated\Delta T + (1 - \alpha) \cdot Sample\Delta T \tag{4}$$

where α is a parameter to smooth the $Estimated\Delta T$. Note that a small α tracks more quickly changes in the ΔT, while a large α guarantees a more stable value but it is not able to quickly adapt to real changes.

Finally, a node v_i with $Estimated\Delta T_i$ is deleted from the Familiar Set if the following equation is true:

$$Estimated\Delta T_i < FSout_{th} \tag{5}$$

where $FSout_{th}$ is the minimum threshold to be kept in the community.

B. Local Community pruning policy

The idea is to keep a timer for each node in the Local Community (i.e., $LCout_{timer}$). The timer is set when a node is inserted into the Local Community (e.g., v_i is inserted in C_0) and is refreshed when one of the following conditions occurs:

i) v_0 encounters directly v_i;

ii) v_0 encounters another node (e.g., v_j) and v_i is stored in the Local Community of v_j.

A node is removed from the Local Community when the corresponding timer expires.

If this happens and the node is in the Familiar Set, it is also removed from it. This guarantees to maintain consistency among the two sets during the pruning process.

In practise, *i)* and *ii)* are analogous to the criteria used for including nodes into Local Community. Note also that criterion *ii)* guarantees that a node is deleted only if no node of the community has met him for a long time. In fact, if there exists at least one node in the community with which it has still some social interactions, the node must still be kept in the Local Community of all the community members.

TABLE III
PARAMETERS USED FOR THE EVALUATION

Parameter	**Value**	**Parameter**	**Value**
Simulation time	172800 sec	Traveller number	1
Reconfiguration time	21600 sec	t_{th}	150 sec
N	54	γ	0.6
Community number	3	λ	0.6

V. RESULTS

In this section we evaluate the performance of the proposed protocol first in isolation and then by a comparison with the SIMPLE algorithm. We show that AD-SIMPLE results in higher values of the similarity metric because it identifies and correctly adapts to social changes.

A. Sensitiveness of AD-SIMPLE on α, $FSout_{th}$ and $LCout_{timer}$

As first analysis, we evaluated the sensitiveness of AD-SIMPLE on its parameters.

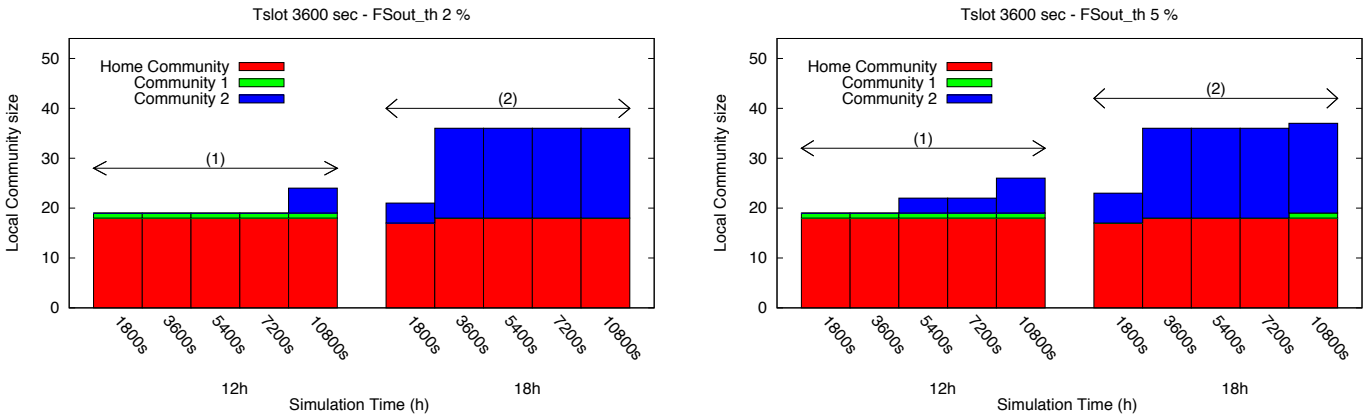
In scenarios such as those described in the previous section, we find out that the behaviour of AD-SIMPLE when varying t_{th}, λ and γ is similar to the original SIMPLE algorithm. This is quite intuitive since AD-SIMPLE makes use of the same criteria for what concern the inclusion of nodes in social communities. Specifically, we observe high variability by changing t_{th} (i.e., the similarity increase when the threshold becomes lower), while we find almost the same behaviour when varying λ and γ.

Concerning to parameters introduced for the Familiar Set policy, we observe that, for a fixed value of time slot T, $FSout_{th}$ should be chosen taking into account also the dynamic of social interactions. In our case study, where nodes have short social interactions (i.e., the cumulative contact duration is low), we note that $FSout_{th}$ should not exceed 5%, otherwise the Familiar Sets would remain empty for most of the time. Regarding the α parameter, we note that high values (i.e, more than 0.6) imply that quick changes at the social level are not reproduced so quickly at the Familiar Set level. Hence, it is important to maintain $\alpha <= 0.3$.

To better investigate how a proper tuning of the $LCout_{timer}$ parameter affects the performance of the LC pruning policy, we refer to a dynamic scenario where nodes may be connected to two communities simultaneously, i.e., home and foreign, and move between them for an entire reconfiguration interval. At each reconfiguration, such nodes select randomly one foreign community to start visiting. The simulation parameters are specified in Table III.

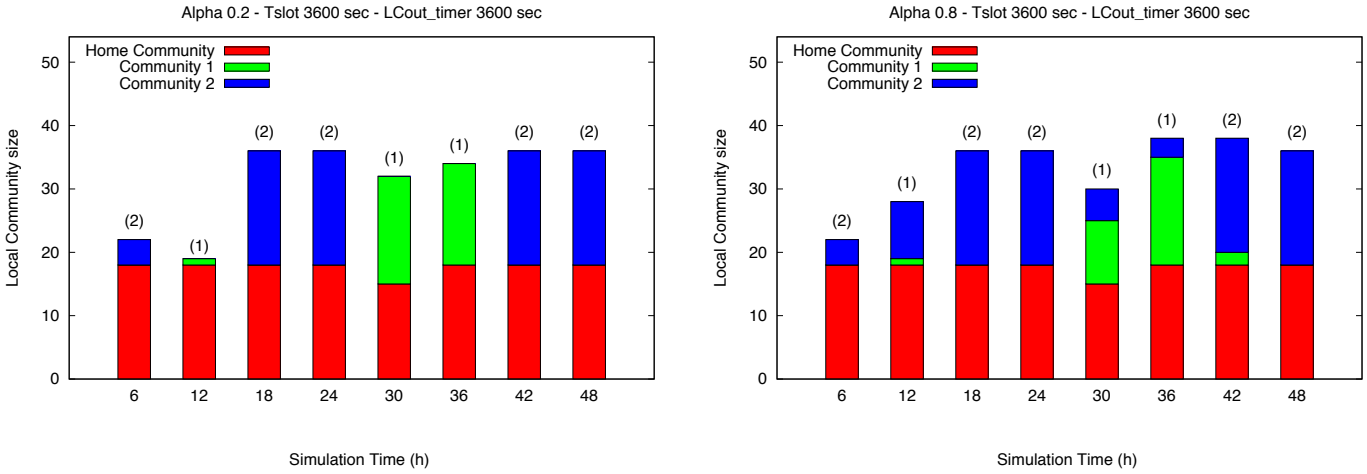
Figure 5 provides relevant examples of different behavior that can be obtained if such parameter is not correctly tuned. Each bar represents a snapshot of the Local Community composition for a traveller at the end of two reconfiguration intervals when varying $LCout_{timer}$ and the $FSout_{th}$. Note that the number in brackets above the bars indicates the foreign community visited by the traveller during that period of time.

By fixing T (3600sec in this example), $LCout_{timer}$ (on the x-axis) must be of the same order. In fact, if $LCout_{timer}$ is set too high (e.g., 10800 sec), nodes of a community previously visited may still stored in the Local Community. This is apparent for example in Figure 5(a) at 12h, where some nodes of community



(a) (b)

Fig. 5. Representative cases of the composition of the Local Community for AD-SIMPLE as a function of $LCout_{timer}$ and $FSout_{th}$.



(a) (b)

Fig. 6. Representative cases of the composition of the Local Community for AD-SIMPLE for different values of α.

2 are present in the Local Community or in Figure 5(b) at 18h, where nodes of community 1 are still present. Furthermore, $LCout_{timer}$ must not set less than T, as this could lead to a too rapid emptying of the Local Community (i.e., timer expires too rapidly). For example, in Figure 5(a) at 18h, only a small percentage of nodes of community 2 are present in the Local Community in case of $LCout_{timer}$ equal to 1800 sec. This means that the system has not a sufficient time to detect all the nodes which belong to the communities. From this analysis, we points out that it is important to properly tune $LCout_{timer}$ in order to maintain a consistent view of the social community at each point in time and a reasonable value is obtained by setting $LCout_{timer}$ = T.

Finally, note that $LCout_{timer}$ is influenced in part by the value of $FSout_{th}$ and in part by the value of α. For example, some nodes of community 2 still remain in LC when $FSout_{th}$ is set to 5% (see,

TABLE IV
AD-SIMPLE PARAMETERS

Parameter	Value	Parameter	Value
α	0.2	$FSout_{th}$	2
T	3600 sec	$LCout_{timer}$	3600 sec

for example, results of 3rd and 4th bars at 12h in Figure 5(b)). As far as the α parameter, if it set too high, not only the Familiar Set is not able to report changes, but Local Community too: a variation of α impacts also on the social community. This is highlighted by Figure 6 that shows results for two different value of α (i.e., 0.2 and 0.8). As it is apparent, it is important to maintain α low and specifically $<= 0.3$, which is also in line with that found earlier for the Familiar Set policy, otherwise this might cause wrong decisions when deleting nodes from the Local Community.

B. SIMPLE vs AD-SIMPLE: a comparison analysis

As far as the comparison with SIMPLE, we consider the same scenario as before (see Table III), which is composed of 54 nodes divided into 3 social communities. The simulation lasts for 48 hours with reconfigurations every 6 hours. This means that, after each reconfiguration, the traveller selects randomly one of the two foreign communities to start visiting. The rest of simulation parameters for AD-SIMPLE are summarised in Table IV.

In this scenario we aim at verifying the ability of AD-SIMPLE to correctly detect social communities, reflecting also any dynamic changes. To this aim, we mainly investigate the number of nodes discovered by the traveller during each reconfiguration interval.

Figure 7 and 8 show the simulation results of two selected simulation runs (SR), breaking down the Local Community composition for the traveller. The two simulation runs differ for the exact mobility actions taken by nodes, as well as for the sequence of foreign communities visited by the traveller during the simulation.

The limitation of SIMPLE is clearly illustrated by the figures. In both runs, the traveller's Local Community increases with the simulation time. After each reconfiguration interval, the traveller adds continuously new nodes to its community, while maintains memory of all the past encountered nodes. As a consequence, the community size reaches 54 after some time (i.e., 24h for both scenarios) and maintains the maximum size for the rest of the simulation. It is worth pointing out a particular behaviour of SIMPLE between 6h and 12h. Referring for example to SR 1, the number of nodes of community 2 increases even though the traveller is in contact with community 1. This is due to a transient phase which follows the beginning of each reconfiguration and it is strictly connected with the traveller location. As explained in Section III-A, if the traveller is visiting nodes of a foreign community, it goes back home with probability $1 - p_e$ while it remains in the foreign community for the next movement with probability (p_e). Thus, if the latter event occurs it will continue meeting nodes of the previous foreign community. This is the reason behind the highlighted increase at 12h.

As highlighted by Figures 7(b) and 8(b), AD-SIMPLE is able to keep always updated the community view of the traveller. Thanks to the implemented pruning policies, the traveller maintains in its Local

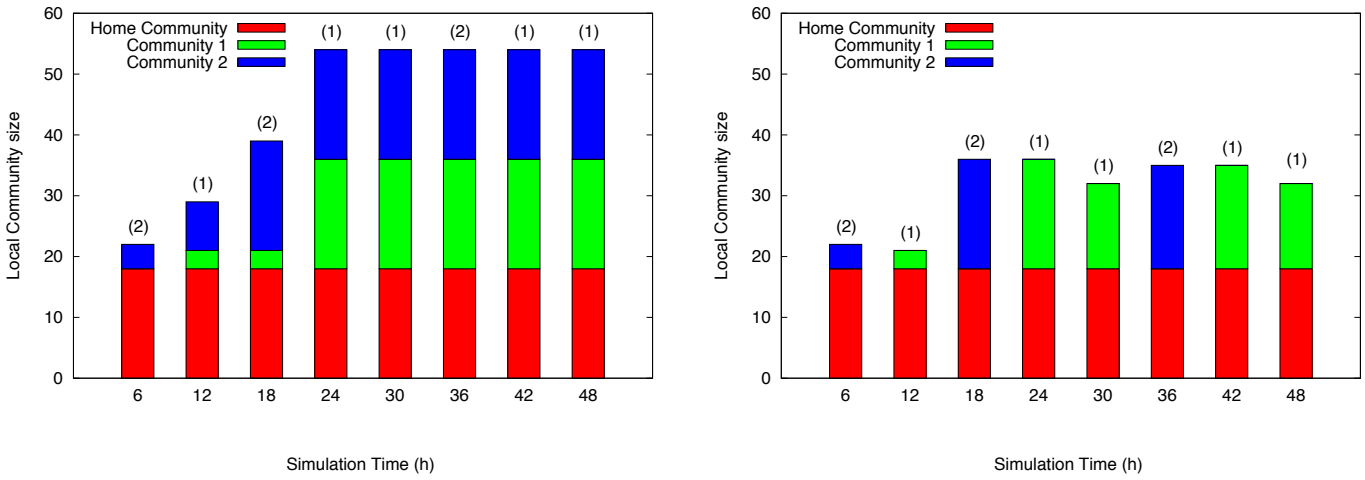


Fig. 7. SR 1: Local Community composition of the traveller for SIMPLE (a) and AD-SIMPLE (b).

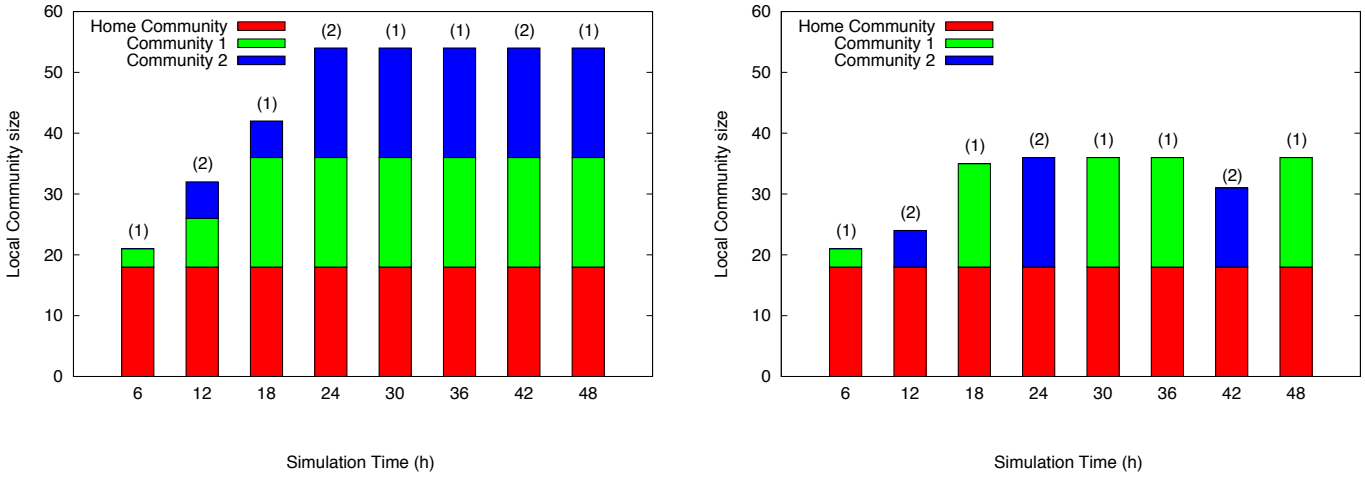


Fig. 8. SR 2: Local Community composition of the traveller for SIMPLE (a) and AD-SIMPLE (b).

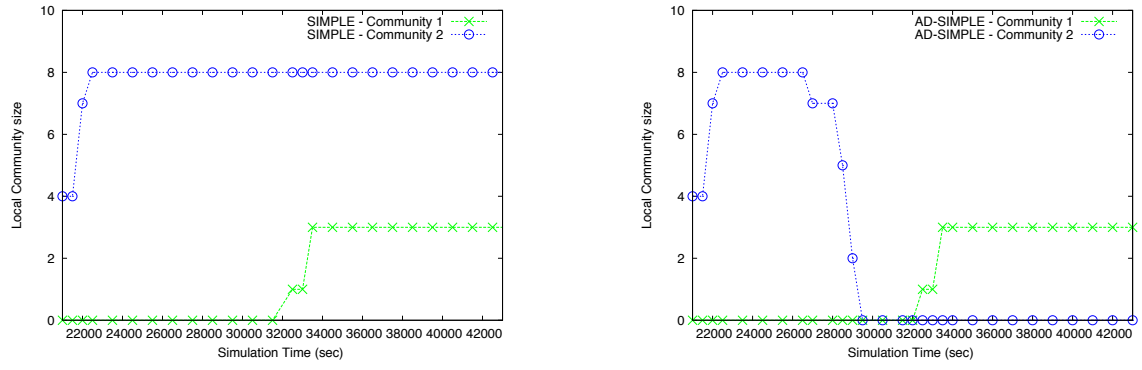
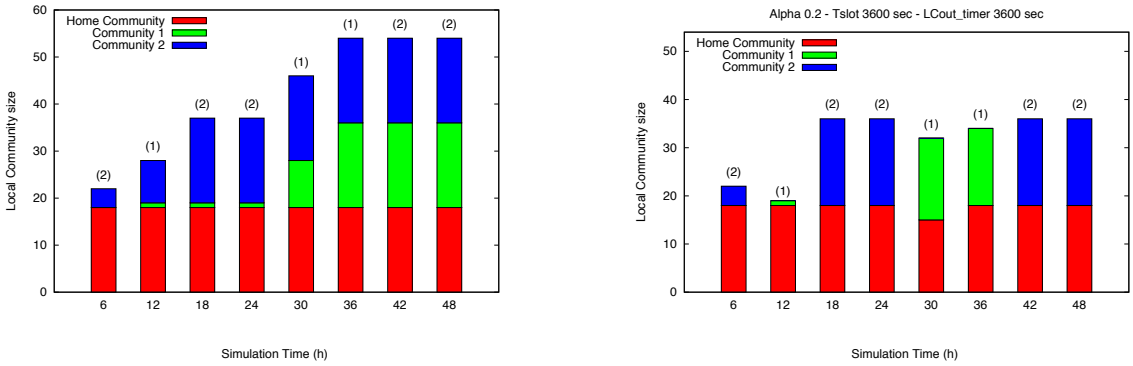
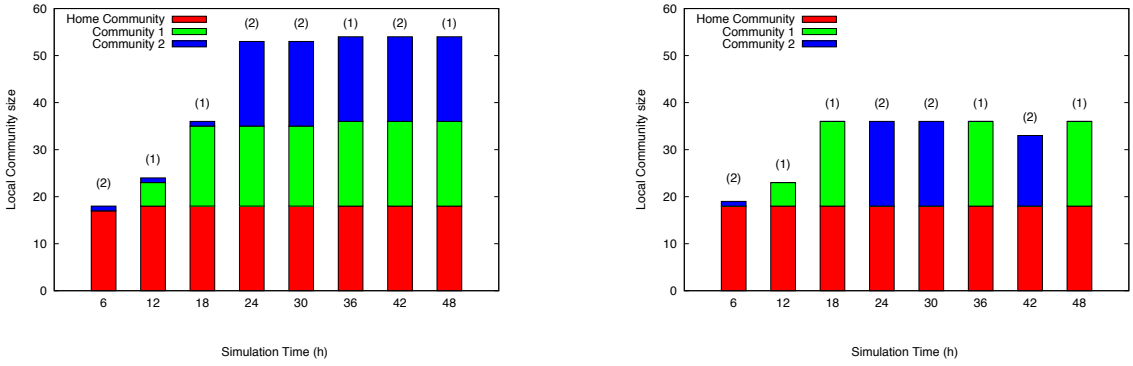


Fig. 9. SR1: Local Community evolution of the traveller for SIMPLE (a) and AD-SIMPLE (b).



(a) (b)

Fig. 10. SR3: Local Community evolution of the traveller for SIMPLE (a) and AD-SIMPLE (b).



(a) (b)

Fig. 11. SR4: Local Community evolution of the traveller for SIMPLE (a) and AD-SIMPLE (b).

Community only nodes belonging to its home community and to the foreign community that it is visiting, while it removes the nodes that are not part of its community anymore. These entries either belong to the previous visited community (e.g., passing from community 1 to 2, it removes all the entries related to community 1 and viceversa) or are nodes of the currently visited community that it has not met for sufficient time. For example, this happens in SR 1 at 30h, where the Local Community size decreases even though the foreign community is not changed after the reconfiguration.

Note also the high increase of the Local Community from 12h to 18h. In this period, the traveller visits one of the community already visited. Thus, it is high probable to add more nodes as the correspondent cumulative durations are close to the threshold t_{th}. Figure 9 shows the evolution of the traveller's Local Community during the reconfiguration interval [6h,12h] (note that nodes belonging to the home community are not included in the graph) for both community detection algorithms. Focusing on the SIMPLE behaviour (see Figure 9 (a)), at the beginning (21600sec), the Local Community is composed of only 4 nodes of community 2. Between 21600sec-24000sec (transient phase), the curve of Community 2 increases since the traveller has still some social integrations with community 2. When these interactions become sporadic and eventually completely disappear, the curve remains constant. At the same time, since interactions with community 1 become predominant, the curve related to community 1 increases too.

Figure 9 shows the evolution of the traveller's Local Community during the reconfiguration interval

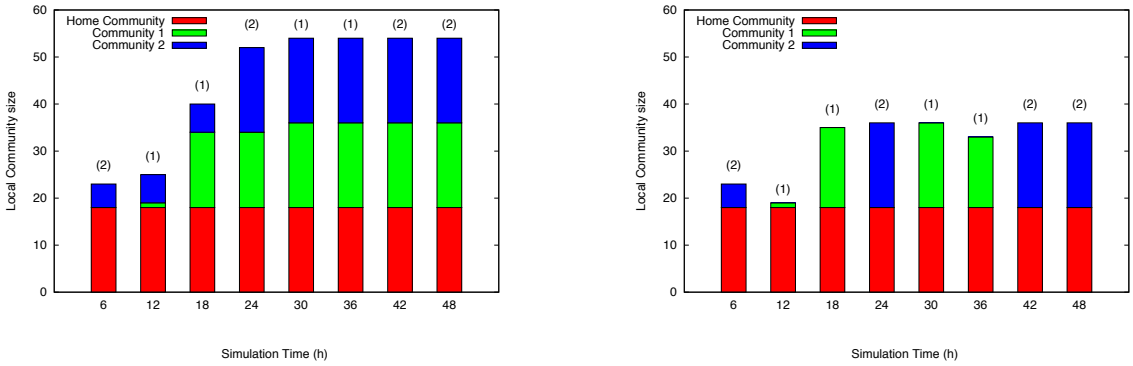


Fig. 12. SR5: Local Community evolution of the traveller for SIMPLE (a) and AD-SIMPLE (b).

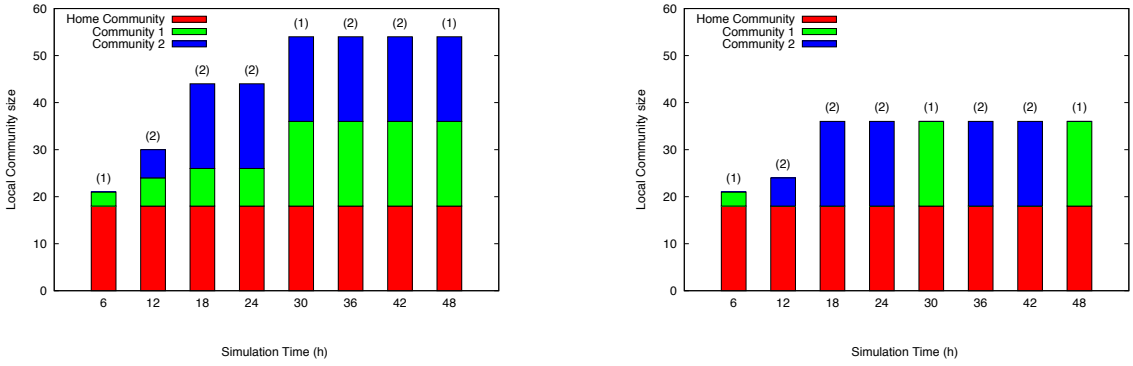


Fig. 13. SR6: Local Community evolution of the traveller for SIMPLE (a) and AD-SIMPLE (b).

[6h,12h] (note that nodes belonging to the home community are not included in the graph) for both community detection algorithms. Focusing on the SIMPLE behaviour (see Figure 9 (a)), at the beginning (21600sec), the Local Community is composed of only 4 nodes of community 2. Between 21600sec-24000sec (transient phase), the curve of Community 2 increases since the traveller has still some social integrations with community 2. When these interactions become sporadic and eventually completely disappear, the curve remains constant. At the same time, since interactions with community 1 become predominant, the curve related to community 1 increases too.

Figure 9(b) shows the same simulation period for AD-SIMPLE. The behaviour of the curve of Community 2 is similar to the previous graph until 26500sec, then the curve decreases reaching 0 at 30000sec due to the expiration of the timeouts associated with Local Community entries. On the contrary, from 32000sec the curve of Community 1 increases since contacts with community 1 become frequent. At the end of the interval, the traveller's Local Community is composed only of nodes belonging to community 1 (and home, of course).

Figures from 10 to 13 provide results for additional simulation runs, here referred to as SR 3 - SR 6. It is worth noting that they show the same behaviour of SR 1 and SR 2. For what concern SIMPLE, when the scenario becomes more complex and dynamic (e.g., people belong to more than one communities simultaneously and spend much time in them), nodes no longer have a consistent vision of the true

TABLE V
SIMILARITY FOR SIMPLE AND AD-SIMPLE IN SR 1

Sim_time	SIMPLE	AD-SIMPLE	Sim_time	SIMPLE	AD-SIMPLE
6h	0.611	0.611	30h	0.66	0.88
12h	0.477	0.58	36h	0.66	0.972
18h	0.923	1	42h	0.66	0.972
24h	0.66	1	48h	0.66	0.88

composition of the community, mainly because they keep memory of all the nodes encountered so far. On the contrary, AD-SIMPLE overcomes such limitation and the traveller maintains only nodes belonging to the two communities which are currently visiting, while it removes those nodes which are not part of its community anymore.

We conclude this section with Table V which summarises the similarity results for SR 1. AD-SIMPLE reaches higher performance while SIMPLE results in very low performance due to its inefficiency of adapting to social changes. Similarity results of SR 2-6 are aligned with those of SR 1 and are omitted mainly for sake of space.

To summarise, AD-SIMPLE results a good solution as it performs better that SIMPLE when detecting social communities and keeps low the computational burden. In addition, its ability to capture the evolution of social interactions adapting to the changes makes it very attractive.

VI. CONCLUSIONS

In this paper we have evaluated the performance of three distributed community detection algorithms for Opportunistic Networks. We have compared them by a simulation analysis in scenarios where nodes move according to a mobility model which realistically reproduces the human mobility patterns. Results from this analysis have shown a good correspondence degree between the communities detected by the three distributed algorithms and the corresponding centralised algorithms. We have also highlighted a common limitation: nodes moving in different social communities may not have a consistent view of their local community since they maintain memory of all encountered nodes. To this aim, we have also investigated a new computationally light-weight solution, namely AD-SIMPLE, which overcomes the above problem. We have shown by simulation that AD-SIMPLE is able to adapt to the dynamic evolutions of social communities maintaining a consistent view of each user.

ACKNOWLEDGEMENT

This work was funded by the European Commission under the SCAMPI (258414) FIRE project.

REFERENCES

[1] P. Hui, E. Yoneki, S.-Y. Chan, and J. Crowcroft, "Distributed community detection in delay tolerant networks," in *Proceedings of MobiArch'07*, 2007.

[2] C. Boldrini and A. Passarella, "Hcmm: Modelling spatial and temporal properties of human mobility driven by users," *Computer Communication*, vol. 33, no. 9, pp. 1056 – 1074, 2010.

[3] L. Pelusi, A. Passarella, and M. Conti, "Opportunistic networking: data forwarding in disconnected mobile ad hoc networks," *IEEE Commnication Magazine*, vol. 44, no. 11, pp. 134 – 141, 2006.

[4] T. Spyropoulos, K. Psounis, and C. S. Raghavendra, "Efficient routing in intermittently connected mobile networks: The multiple-copy case," *IEEE/ACM Transactions on Networking*, vol. 16, no. 1, pp. 77–90, 2008.

[5] X. Chen and A. Murphy, "Enabling disconnected transitive communication in mobile ad hoc networks," in *Proc. of the Workshop on Principles of Mobile and Computing, in conj. with PODC'01*, Newport, RI, August 2001.

[6] B. Lidgren, A. Doria, and O. Shelén, "Probabilistic routing in intermittently connected networks," *Mobile Computing and Communications Review Networks*, vol. 7 (3), July 2003.

[7] J. Leguay, T. Friedman, and V. Conan, "Dtn routing in a mobility pattern space," in *Proc. of the ACM SIGCOMM 2005 Workshop on delay tolerant networks*, Philadelphia, PA, August 2005.

[8] C. Boldrini, M. Conti, and A. Passarella, "Exploiting users' social relations to forward data in opportunistic networks: The HiBOp solution," *Pervasive and Mobile Computing*, vol. 4, no. 5, pp. 633–657, 2008.

[9] P. Hui, J. Crowcroft, and E. Yoneki, "BUBBLE Rap: Social-based forwarding in delay tolerant networks," *IEEE Transactions on Mobile Computing (to appear)*, 2011.

[10] G. Palla, I. Dereny, I. Farkas, and T. Vicsek, "Uncovering the overlapping community structure of complex networks in nature and society," *Nature*, vol. 435, no. 9, pp. 814 – 818, 2005.

[11] L. Danon, J. Duch, A. Diaz-Guilera, and A. Arenas, "Comparing community structures identifications," 2005.

[12] M. Newman, "Detecting community structures in networks," *The European Physical Journal B - Condensed Matter and Complex Systems*, vol. 38, no. 2, pp. 321–330, 2004.

[13] A. Clauset, "Finding local community structures in networks," *Physical Review E*, vol. 72, no. 2, pp. 026 132+, 2005.

[14] P. Jaccard, *Bulletin de la Societe Vaudoise des Sciences Naturelles*, vol. 37, no. 547, 1901.

Distributed Rating Prediction in User Generated Content Streams

Sibren Isaacman, Stratis Ioannidis,
Augustin Chaintreau, and Margaret Martonosi

Technicolor Technical Report
Number: **PAC-2011-05-01**
First publication date: Sept. 2011

Abstract: Recommender systems predict user preferences based on a range of available information. For systems in which users generate streams of content (*e.g.*, blogs or periodically-updated newsfeeds), users may rate the produced content that they read, and be given accurate predictions about future content they are most likely to prefer. We design a distributed mechanism for predicting user ratings that avoids the disclosure of information to a centralized authority or an untrusted third party: users disclose the rating they give to certain content only to the user that produced this content.

We demonstrate how rating prediction in this context can be formulated as a matrix factorization problem. Using this intuition, we propose a distributed gradient descent algorithm for its solution that abides with the above restriction on how information is exchanged between users. We formally analyse the convergence properties of this algorithm, showing that it reduces a weighted root mean square error of the accuracy of predictions. Although our algorithm may be used many different ways, we evaluate it on the Neflix data set and prediction problem as a benchmark. In addition to the improved privacy properties that stem from its distributed nature, our algorithm is competitive with current centralized solutions. Finally, we demonstrate the algorithm's fast convergence in practice by conducting an online experiment with a prototype user-generated content exchange system implemented as a Facebook application.

1. INTRODUCTION

A considerable portion of web activity today can be attributed to direct interactions between online users. Blogs, social networks such as Facebook, micro-blogging applications such as Twitter, and video sharing sites such as YouTube have provided online platforms through which users author and share original content, establishing an online following that often overlaps with their real-life social circles.

The sheer volume of content generated daily in the blogosphere and on social networks makes identifying relevant and interesting content a challenging task. At present, providers of these services have deployed rating mechanisms though which a user can give feedback on content generated by other users. Facebook and Twitter have expanded their rating mechanisms to the blogosphere, competing with other traditional aggregators such as Digg, Reddit and StumbleUpon that offer similar rating interfaces. Access to such ratings allow such companies to improve their recommendations but also to profile users; such profiles are a resource that companies monetize, *e.g.*, through advertising.

On the other hand, the increased monetization of private data has been met by a sharp rise in privacy concerns within advocacy groups like the Electronic Frontier Foundation and regulatory bodies like the US Congress [2]. Privacy in general, and online privacy in particular, is recognized as a fundamental human right by laws such as the European Directive on Protection of Personal Data and the Electronic Communications Privacy Act in the U.S. Nevertheless, there are many reasons why online users readily disclose private information to the above companies. Behavioral economists have identified bounded rationality, immediate gratification, underestimating risk [1] and the paradox of control [5] as some of them. Nevertheless, the importance of respecting the privacy of online users has been recognized not only by the authorities and non-profit organizations but by companies as well. For example, Google and Microsoft have struggled to find compromises that respect the privacy of their users without significantly undermining their profits [29, 31].

The challenge thus arising from this state of affairs is enabling users to view and access relevant, interesting, user-generated content without releasing their private information to untrusted third parties. In this paper, we propose a solution to this problem by designing a distributed rating prediction mechanism that allows users to restrict information sharing only among trusted parties.

More specifically, we consider a general system in which users generate and share streams of content items. For example, a content producer in such a system may maintain a blog, a Facebook wall or a Twitter feed. Updates generated by users (*i.e.*, new blog posts, wall entries or tweets) are shared with select subscribers, who may respond to received content by rating it. Note that, at present, such systems are centralized. However, this need not be the case in the system we consider here: users may generate and share both their content as well as their ratings with each other in a peer-to-peer fashion.

Our goal is to design a fully distributed mechanism that learns and predicts the ratings of content consumers—*i.e.*, the subscribers. More specifically, we would like to devise a collaborative filtering scheme operating under the constraint that information is shared only between (a) a content producer and (b) its subscribers. For example, a user rating certain content should share this rating *only* with the user that generated the content.

Again, the main reason motivating our focus on distributed rating prediction is privacy: we wish to avoid the disclosure of ratings or, in fact, any user information (such as profiles), to a central authority or an untrusted third party. Another reason is scalability, as a distributed approach can scale as the number of content producers and consumers increases. Furthermore, since ratings are exchanged among content producer/consumer pairs, our algorithm naturally exploits the social relationships between them: if two friends subscribe to each other's feed, training our predictor based on content exchanges between them can exploit latent similarities between their interests.

Our contributions can be summarized as follows:

- We propose a mathematical model of a system for distributed sharing of user-generated content streams. Our model captures a variety of different applications, and incorporates correlations both on how producers deliver content and how consumers rate it.

- We illustrate that estimating the *probability distribution* of content ratings can be naturally expressed as a Matrix Factorization (MF) problem. This is in contrast to standard MF formulations that focus on estimating ratings directly, rather than their distribution. To the best of our knowledge, our work is the first to apply a MF technique in the context of rating prediction in user-generated content streams.

- Using the above intuition, we propose a decentralized rating prediction algorithm in which information is exchanged only across content producer/consumer pairs. Producers and consumers maintain their own individual profiles; a producer shares its profile *only* with consumers to which it delivers content, and consumers share a rating they give to an item, as well as their profile, *only* with the producer that generated it.

- In spite of the above restriction on how information is exchanged among users, our distributed prediction algorithm optimizes a global performance objective. In particular, we formally characterize the algorithm's convergence properties under our model, showing that it reduces a weighted mean square error of its rating distribution estimates.

- We validate our algorithm empirically. First, we use the Netflix data set as a benchmark to compare the performance of our distributed approach to offline centralized algorithms. Second, we developed a Facebook application that reproduces the main features of a peer-to-peer content exchange environment. Using a month-long experiment with 43 users we show that our algorithm predicts ratings accurately with limited user feedback.

The remainder of this paper is organized as follows. In Section 2 we briefly describe the relationship of our work to previous rating prediction mechanisms. In Section 3 we introduce our mathematical model, and in Section 4 we present our distributed prediction algorithm as well as our main results on its convergence. Our numerical evaluation over the Netflix data set and our case study using a FaceBook application are in Sections 6 and 7, respectively. Finally, we conclude in Section 8.

2. RELATED WORK

A traditional approach to distributed rating prediction in peer-to-peer networks involves computing a similarity metric (*e.g.*, Pearson correlation) among neighboring peers; predicted ratings are obtained as a function of the ratings in a peer's neighborhood, taking into account its similarity to its neighbors [9, 19, 22, 25, 30]. Neighbor selection is part of system design, as predictions improve when neighbors have high similarity scores. We depart considerably from these works by focusing on dynamic user generated content (UGC), rather than static content, as well as by providing formal performance guarantees on the prediction error.

Matrix factorization is popular among collaborative filtering methods due to its ability to scale over large datasets [4, 16, 27]. A simple method for obtaining a low-rank factorization of a rating matrix is gradient descent on the prediction RMSE, potentially with added regularization terms (see, *e.g.*, [27], and the references therein). Assuming that ratings follow the Gaussian distribution, minimizing the RMSE is equivalent to maximizing the likelihood of observed entries [26]. In this work, we do not rely on a Gaussian assumption; instead, we directly apply MF to the *probability distribution* of ratings in dynamic UGC streams. Moreover, our distributed algorithm behaves as gradient descent over a *weighted* RMSE metric, whose weights correspond to content delivery rates across different consumers.

Recent work on MF has demonstrated that if entries removed from an r-rank matrix are selected uniformly at random, the matrix can be reconstructed accurately with high probability. Candès and Recht [7] show that, under certain conditions, if at least $\Omega(n^{6/5} r \log n)$ entries of an $n \times n$-sized matrix are known, the entire matrix can be retreived w.h.p. by solving a semi-definite problem (a nuclear norm minimization); Candès and Tao [6] reduce this bound to $\Omega(nr \log n)$. Keshavan *et al.* [13] propose an alterative approach that reconstructs the matrix with the same bound on known entries as [6]: missing entries are replaced by zeros and a low-rank approximation through singular value decomposition (SVD) is attained; the error on the known entries is then minimized through gradient descent. In fact, simply getting a low-rank approximation through SVD on the matrix padded with zeros is known to reconstruct the original matrix w.h.p., albeit under looser lower bounds [3]. These methods have been extended to provide guarantees of matrix recostruction w.h.p. in the presence of noise [14, 32].

These approaches yield stronger results than [27] as, due to the lack of convexity, gradient descent does not always converge to the original matrix, or even a minimizer of the prediction error. Nevertheless, gradient descent methods can reduce the prediction error even when entries are not removed uniformly at random—which is true for the system we study. Most importantly, the algorithms in [3, 6, 7, 13, 14, 32] are centralized and thus cannot be directly applied to the distributed setting considered in this work.

Although the above traditional MF methods do not focus on distributed implementations, there exist distributed algorithms for principal component analysis (PCA) of the adjacency matrix of a graph that restrict message exchanges only across adjacent nodes. These algorithms relate to our work as PCA can be used to construct an optimal low-rank approximation of the adjacency matrix—though, technically speaking, a MF need not be ortho-normalized, so PCA is harder than factorization. Kempe and McSherry [12] propose such an algorithm based on the power method, and Korada *et al.* [15] extend it to PCA over the expectation of the adjacency matrix of a random graph. Tomozei and Massoulié [28] consider a similar setting as [15] relying however on Oja's method [23]. Unfortunately, such approaches do not apply to our setting as we *do not* factorize the random adjacency matrix restricting communication between nodes/users (matrix $A(k)$ in our model): we factorize the distribution of ratings, which are decoupled from the communication process. Moreover, in contrast to our algorithm, the above methods are not fully distributed, as orthonormalizing principal components requires broadcasting or gossiping normalization factors across all nodes.

Privacy in collaborative filtering was previously addressed using homomorphic encryption [8], randomized perturbation [24] and concordance measure [18] techniques. Our approach, in contrast, relies on trust between participants: we restrict information exchanges only between trusted content producer-consumer pairs. Within this context, secure multiparty computation is not required to predict ratings without disclosing user ratings publicly.

3. SYSTEM MODEL

In this section, we present the mathematical model that we use in our analysis.

3.1 Content Sharing

We consider a set $\mathcal{U}$ of users generating and sharing content in a peer-to-peer manner. A subset $\mathcal{N} \subseteq \mathcal{U}$ of all users, whom we call *producers*, generate a stream of items. For example, producers could maintain a blog, a news-feed or a twitter-feed. Time is divided into timeslots, and in each timeslot every producer generates a new content item (*i.e.*, a blog entry or a tweet) that is added to her locally-maintained feed; this is subsequently shared with other users in a set $\mathcal{M} \subseteq \mathcal{U}$, which we call *consumers*. Note that $\mathcal{M}$ and $\mathcal{N}$ may intersect, as a user may both produce and consume content.

Producers share items only with consumers that belong to their social circle and/or subscribe to their feed. Even so, items may fail to reach all such consumers, either because the producer shares them selectively or because the consumers fail to receive them (*e.g.*, because they do not observe the feed continuously).

We model this as follows. Let $a_{i,j}(k) \in \{0, 1\}$ be a binary random variable indicating whether the item generated by $i \in \mathcal{N}$ at timeslot k is delivered to $j \in \mathcal{M}$, and let $A(k) = [a_{i,j}(k)]_{i \in \mathcal{N}, j \in \mathcal{M}}$ be the corresponding $|\mathcal{N}| \times |\mathcal{M}|$ matrix. We make the following assumption:

ASSUMPTION 1. *$\{A(k)\}_{k \in \mathbb{N}}$ is an i.i.d. sequence.*

I.e., deliveries are independent and identically distributed across time. Let $\lambda_{i,j} = \mathbb{E}[a_{i,j}]$ be the probability that i delivers an item to j. If j does not subscribe to i's feed, then $\lambda_{i,j} = 0$. Different consumers may receive items from a feed with different probabilities; our model thus allows heterogeneity in how content is targeted by producers and how often consumers fail to observe it. Note that Assumption 1 *does not imply* that deliveries between different user pairs are independent. *E.g.*, $A(k)$ may be such that an item delivered to Alice is always also delivered to Bob.

3.2 Content Ratings

Whenever a producer i delivers content to a consumer j, the consumer provides some feedback to i in the form of a *rating*. We denote by $\mathcal{O}$ the set of possible ratings provided by consumers. In general, ratings are application-dependent. For example, consumers may indicate through an appropriate interface whether they liked, disliked or were neutral towards the content, so that $\mathcal{O} = \{+, -, \varnothing\}$. Con-

sumers may also indicate their interest on a scale from 1 (lowest interest) to 5 (highest interest), *i.e.*, $\mathcal{O} = \{1, 2, 3, 4, 5\}$.

Let $\mathcal{I}_{i,j} = \{k \in \mathbb{N} : a_{i,j}(k) = 1\}$ be the set of timeslots at which i delivers content to j. W.l.o.g., for all $k \in \mathcal{I}_{i,j}$, j provides a rating to i within the duration of the timeslot k. Depending on the application, lack of feedback can be modelled either as a failed delivery ($a_{i,j}(k) = 0$) or an additional rating (element in $\mathcal{O}$). For $k \in \mathcal{I}_{i,j}$, let $r_{i,j}(k) \in \mathcal{O}$ be the the rating given by j to i's content. We assume that:

ASSUMPTION 2. $\{r_{i,j}(k)\}_{k \in \mathcal{I}_{i,j}}$ *is an i.i.d. sequence.*

Note that ratings *need not* be independent across users. For example, Alice and Charlie may always give the same rating to an item from Bob.

3.3 Rating Distributions

Denote by $\tilde{\pi}^o_{i,j}$, $o \in \mathcal{O}$, the probability that $r_{i,j}(k) = o$ for $k \in \mathcal{I}_{i,j}$. For every rating $o \in \mathcal{O}$, let

$$\tilde{\Pi}^o = [\tilde{\pi}^o_{i,j}]_{i \in \mathcal{N}, j \in \mathcal{M}}. \quad (1)$$

This is a $|\mathcal{N}| \times |\mathcal{M}|$ matrix; each element corresponds to a producer/consumer pair i, j, and contains the probability that j gives rating o to an item from i. Our goal is to correctly estimate the probability matrices $\tilde{\Pi}^o$, for all $o \in \mathcal{O}$. *I.e.*, for any producer/consumer pair and any rating, we wish to find the probability that the consumer will react to content generated by the producer by providing this rating. Most importantly, we wish to do so in a distributed fashion, by restricting information exchanges only directly between producers and consumers.

Note that every time a consumer rates a content item the rating is in effect a sample from the distribution defined by the matrices $\tilde{\Pi}^o$, $o \in \mathcal{O}$. However, due to the heterogeneity of the process $A(k)$, samples are not obtained at the same rate. In fact, if $\lambda_{i,j} = 0$, the distribution of ratings of the (i, j) producer/consumer pair is *never* sampled; this relates the estimation of $\tilde{\Pi}^o$ to matrix completion, as certain entries of $\tilde{\Pi}^o$ are missing and cannot be directly observed.

Contrary to traditional matrix completion, missing entries of $\tilde{\Pi}^o$ are not selected uniformly at random—their absence is determined by, *e.g.*, the feeds to which a consumer subscribes. However, just as in traditional matrix completion, it is very natural to make the following assumption:

ASSUMPTION 3. *The probability matrices* $\tilde{\Pi}^o$ *are low-rank.*

This assumption implies that MF techniques can be applied to estimate these matrices; indeed our algorithm, presented in Section 4, exploits this relationship. Note that the low-rank property holds for the *rating distribution*, rather than the ratings themselves—as each producer generates an infinite number of items, this distinction is necessary.

Assumption 3 can be interpreted as a consequence of a generative latent factor model and the total probability theorem. As such, it is indeed very natural in the context of our system. We illustrate this below.

3.4 A Low-Rank Latent Factor Model

In this section we give an example of how the low-rank property of the matrices $\tilde{\Pi}^o$ may manifest by making additional assumptions on how users generate and rate content. This is only for the sake of illustration and to motivate our approach: our main results (Theorems 1 to 3) rely only on the assumptions we have made so far (in particular, Assumptions 1 and 2).

Suppose that content items generated by producers are grouped by similarity with respect to some features, thus forming a partition of the "content universe" into categories. For example, categories may pertain to topics (*e.g.*, news, music, sports, *etc.*). Formally, assume the existence of a set $\tilde{\mathcal{F}}$, whose elements we refer to as categories, such that every content item generated by a producer belongs to a single category. When $i \in \mathcal{N}$ generates new item, this item belongs to category $f \in \tilde{\mathcal{F}}$ with probability

$$\tilde{p}_{i,f} \geq 0, \quad \text{where } \textstyle\sum_{f \in \tilde{\mathcal{F}}} \tilde{p}_{i,f} = 1, \quad (2)$$

independently of any categories of items the producer has generated in the past. Moreover, when $j \in \mathcal{M}$ views an item in category $f \in \tilde{\mathcal{F}}$, j provides rating $o \in \mathcal{O}$ with probability

$$\tilde{q}^o_{j,f} \geq 0, \quad \text{where } \textstyle\sum_{o \in \mathcal{O}} \tilde{q}^o_{j,f} = 1, \quad (3)$$

independently of any ratings the user has given in the past.

Then, from the total probability theorem, the probability that j gives rating o when viewing content from i is:

$$\tilde{\pi}^o_{i,j} = \textstyle\sum_{f \in \tilde{\mathcal{F}}} \tilde{p}_{i,f} \tilde{q}^o_{j,f} = \langle \tilde{p}_i, \tilde{q}^o_j \rangle, \quad o \in \mathcal{O}, \quad (4)$$

or, in matrix form,

$$\tilde{\Pi}^o = \tilde{P} \cdot (\tilde{Q}^o)^T, \quad o \in \mathcal{O}, \quad (5)$$

where $\tilde{P} = [\tilde{p}_{i,f}]_{i \in \mathcal{N}, f \in \mathcal{F}}$ and $\tilde{Q}^o = [\tilde{q}^o_{j,f}]_{j \in \mathcal{M}, f \in \mathcal{F}}$. *I.e.*, the matrices $\tilde{\Pi}^o$, $o \in \mathcal{O}$, admit a $|\tilde{\mathcal{F}}|$-rank decomposition and, as such, their ranks are at most $|\tilde{\mathcal{F}}|$. Thus, *if the number of categories is small, the matrices* $\tilde{\Pi}^o$ *are low-rank.* Moreover, the l.h.s. matrix $\tilde{P}$ is the same in all $|\mathcal{O}|$ decompositions. The low-rank property is thus a consequence of the existence of content categories and the total probability theorem.

4. DISTRIBUTED RATING PREDICTION

The rating prediction problem in the context of this work is to correctly estimate the probability matrices $\tilde{\Pi}^o$, $o \in \mathcal{O}$, in a distributed fashion. Below, we first formulate this problem as a MF problem and then present our distributed gradient-descent mechanism for its solution.

4.1 Problem Formulation as Matrix Factorization

To estimate $\tilde{\Pi}^o$ through MF, we construct low-dimensional *profiles* of producers and consumers. The inner product of two such profiles yields an estimate of the rating probabilities $\tilde{\pi}^o_{i,j}$. More specifically, let $d \in \mathbb{N}$ be a small integer such that $d \ll \min(|\mathcal{N}|, |\mathcal{M}|)$ and denote by $\mathcal{F}$ the set $\{1, 2, \ldots, d\}$. For now, we make no assumptions on how d relates to the rank of $\tilde{\Pi}^o$ (*i.e.*, the number of "categories" $|\tilde{\mathcal{F}}|$ in the example given by (5)).

Each producer $i \in \mathcal{N}$ maintains a vector $p_i \in [0, 1]^d$ that satisfies (2), which we call the *production profile* of i. Similarly, each consumer $j \in \mathcal{M}$ maintains $|\mathcal{O}|$ vectors $q^o_j \in \mathbb{R}^d_+$, one for every rating $o \in \mathcal{O}$, that satisfy (3). These $|\mathcal{O}|$ vectors constitute the *consumption profile* of j:

$$q_j = (q^{o_1}_j, q^{o_2}_j, \ldots, q^{o_{|\mathcal{O}|}}_j).$$

Given the above profiles, our estimate of the probability $\tilde{\pi}^o_{i,j}$ (the probability that when j views content generated by i it gives rating o)—is computed as follows:

$$\pi^o_{i,j} = \textstyle\sum_{f \in \mathcal{F}} p_{i,f} q^o_{j,f} = \langle p_i, q^o_j \rangle, \quad o \in \mathcal{O}. \quad (6)$$

Note the similarity between Equations (6) and (4). The constraints (2) and (3) immediately imply that the inner

products in (6) are non-negative and $\sum_{o\in\mathcal{O}} \pi^o_{i,j} = 1$, *i.e.*, the latter indeed constitute a probability distribution.

Our goal is then to devise an algorithm for finding profiles p_i, q_j such that the prediction $\pi^o_{i,j}$ is as close to $\tilde{\pi}^o_{i,j}$ as possible. More formally, we wish to solve the following optimization problem:

RATING PREDICTION

$$\text{Minimize} \quad E = \sum_{i\in\mathcal{N}, j\in\mathcal{M}} \lambda_{i,j} \sum_{o\in\mathcal{O}} |\tilde{\pi}^o_{i,j} - \pi^o_{i,j}|^2, \quad (7a)$$

$$\text{subject to:} \quad p_i \in D_1, \quad i \in \mathcal{N}, \text{ and} \quad (7b)$$

$$q_j \in D_2, \quad j \in \mathcal{M}, \quad (7c)$$

where D_1 and D_2 are the sets of profiles that satisfy (2) and (3), respectively. We call the objective function E in (7a) the *error* of our estimate. It corresponds to the error of a rating selected uniformly at random among ratings within a timeslot. Its minimization is equivalent to minimizing a weighted root mean square error (RMSE), with weights equal to the delivery rates $\lambda_{i,j}$. In particular, when $\lambda_{i,j} = 0$ (*i.e.*, when i never delivers content to j), then E does not account for the distance between $\pi^o_{i,j}$ and $\tilde{\pi}^o_{i,j}$. This is not a bug but a useful feature: we never have to predict how j would react to content from i unless it receives such content.

Assumption 3 implies that there exists a small dimension, namely $|\tilde{\mathcal{F}}|$, such that if $d \geq |\tilde{\mathcal{F}}|$ the minimum error will be zero. When $d < |\tilde{\mathcal{F}}|$, there may not exist profiles yielding $E = 0$. In other words, *underestimating* the number of categories may preclude achieving a zero error; this is possible as the number of "categories" is often unknown.

4.2 A Distributed Learning Algorithm

Our distributed algorithm for solving RATING PREDICTION is specified in Figure 1. It is fully distributed, and ensures that a consumer discloses the rating of a content item only to the producer that generated it. Moreover, producers share their profiles only with consumers that subscribe to their feeds, and vice-versa.

In more detail, whenever producer $i \in \mathcal{N}$ delivers content to consumer $j \in \mathcal{M}$, the following interactions take place. First, in addition to the content item, i sends to j its profile p_i. Second, the consumer views and rates the content item with a rating $r_{i,j} \in \mathcal{O}$. Third, the consumer reports to the producer (a) the rating $r_{i,j}$ for this content, as well as (b) its consumption profile q_j. We assume that consumer j reports its rating and consumption profile to i instantaneously, upon receipt of the item. Nevertheless, our results can be directly extended to the case where these are reported with an arbitrary delay within the current timeslot (see Appendix A).

Upon exchanging the above information, i and j update their profiles as follows:

$$p_i \leftarrow p_i + \gamma \sum_{o\in\mathcal{O}} (\mathbb{1}_{r_{i,j}=o} - \langle p_i, q^o_j\rangle) q^o_j \quad (8a)$$

$$q^o_j \leftarrow q^o_j + \gamma(\mathbb{1}_{r_{i,j}=o} - \langle p_i, q^o_j\rangle) p_i, \quad o \in \mathcal{O} \quad (8b)$$

where $\gamma = \gamma(k)$ is the learning rate. We assume that γ satisfies the following properties:

$$\gamma(k) \geq 0, \quad \sum_{k=1}^{\infty} \gamma(k) = \infty, \text{ and } \sum_{k=1}^{\infty} (\gamma(k))^2 < \infty, \quad (9)$$

which hold, *e.g.*, for $\gamma(k) = 1/k$.

Finally, at the end of the timeslot, after (8a) and (8b) have been applied at every encounter the producers and consumers further update their profiles by forcing them to satisfy (2) and (3):

$$p_i \leftarrow \Pi_{D_1}(p_i), \quad i \in \mathcal{N} \quad (10a)$$

$$q_j \leftarrow \Pi_{D_2}(q_j), \quad j \in \mathcal{M} \quad (10b)$$

where $\Pi_{D_1} : \mathbb{R}^{|\mathcal{F}|} \to D_1$, $\Pi_{D_2} : \mathbb{R}^{|\mathcal{O}|\times|\mathcal{F}|} \to D_2$ are the orthogonal projections to D_1 and D_2, respectively. Due to the simple geometry of D_1, D_2, there are known algorithms that compute these projections efficiently [20, 21]. In particular, Π_{D_1} can be computed in $O(|\mathcal{F}|)$ steps, while Π_{D_2} can be computed in $O(|\mathcal{O}||\mathcal{F}|)$ steps.

Producer i at timeslot k:
  i generates new content item
  $\gamma \leftarrow \gamma(k)$
  for every pair i, j s.t. $a_{i,j}(k) = 1$:
    i sends its item and p_i to j.
    i receives $r_{i,j}$ and q_j from j.
    $p_i \leftarrow p_i + \gamma(k) \sum_{o\in\mathcal{O}} (\mathbb{1}_{r_{i,j}=o} - \langle p_i, q^o_j\rangle) q^o_j$
  $p_i \leftarrow \Pi_{D_1}(p_i)$

Consumer j at timeslot k:
  $\gamma \leftarrow \gamma(k)$
  for every pair i, j s.t. $a_{i,j}(k) = 1$:
    j receives item and p_i from i.
    j rates item with $r_{i,j} \in \mathcal{O}$.
    j sends $r_{i,j}$ and q_j to i.
    for every $o \in \mathcal{O}$:
      $q^o_j \leftarrow q^o_j + \gamma(\mathbb{1}_{r_{i,j}=o} - \langle p_i, q_j\rangle) p_i$.
  $q_j \leftarrow \Pi_{D_2}(q_j)$.

Figure 1: Distributed learning algorithm.

4.3 Convergence Properties

In this section, we state our main result: the dynamics of our algorithm *always* lead to a decrease in E. This important property ensures that updating the production and consumption profiles improves our predictions, even if we underestimate the number of dimensions (*i.e.*, when $d < |\tilde{\mathcal{F}}|$). We also characterize the limit points of this system and prove that it is locally stable around an optimal solution. In the rest of this section, we state formally the above results. All proofs are included in the Appendix.

Our argument is as follows. First, we show that the trajectories of the stochastic system we study asymptotically converge to the solution of a system of ordinary differential equations. Second, we prove that the error E always decreases; this, in turn, implies local stability properties. We note that our results rely on Assumptions 1 and 2 but not on Assumption 3: in particular, the weighted error E always decreases *even if the matrices* $\tilde{\Pi}^o$, $o \in \mathcal{O}$, *are not low-rank.* Nevertheless, as discused in Section 4.1, when Assumption 3 fails to hold, profiles under which E is zero may not exist.

Convergence to the solution of a system of ODEs.

Consider the partial gradients

$$\nabla_{p_i} E = \left[\partial E/\partial p_{i,f}\right]_{f\in\mathcal{F}}, \quad \nabla_{q_j} E = \left[\partial E/\partial q^o_{j,f}\right]_{f\in\mathcal{F}, o\in\mathcal{O}},$$

and the following system of ODEs:

$$\frac{dp_i}{dt} = -\nabla_{p_i} E + z_{D_1}, \qquad i \in \mathcal{N}, \quad (11a)$$

$$\frac{dq_j}{dt} = -\nabla_{q_j} E + z_{D_2}, \qquad j \in \mathcal{M}, \quad (11b)$$

where z_{D_1}, z_{D_2}, defined formally in Appendix A, are the minimum forces required to keep the evolution of the system within the constraint set $D_1 \times D_2$. In other words, the ODE

(11) can be viewed as a *projected gradient descent* for the RATING PREDICTION problem.

Let $p_i(k)$, $q_j(k)$, be the production and consumption profiles of $i \in \mathcal{N}$, $j \in \mathcal{M}$, respectively, at timeslot $k \in \mathbb{N}$. We extend these to continuous-time functions $p_i : \mathbb{R}_+ \to D_1$, $q_j : \mathbb{R}_+ \to D_2$ as follows. Let $t_k = \sum_{i=1}^{k} \gamma(k)$ and define for all i and j the continuous-time interpolated processes:

$$p_i(s) = p_i(k), \quad q_j(s) = q_j(k), \text{ for } s \in [t_k; t_{k+1}[.$$

Theorem 1 below establishes that after sufficiently long time, the continuous-time interpolated trajectories of our system can be arbitrarily well approximated by solutions to (11). Moreover, any limit points of our system must also be limit points of the ODE (11). Its proof can be found in Appendix A.

THEOREM 1. *Consider the interpolated processes*

$$[p_i(t), q_j(t)]_{i\in\mathcal{N}, j\in\mathcal{M}}, \quad t \in \mathbb{R}_+.$$

Then, for any $T > 0$ there exist solutions of (11)

$$[p_i^*(t), q_j^*(t)]_{i\in\mathcal{N}, j\in\mathcal{M}}, \quad t \in \mathbb{R}_+,$$

such that, as $k \to \infty$, the quantity

$$\sup_{t_k \le \tau \le t_k + T} \Big(\sum_i \|p_i(\tau) - p_i^*(\tau)\|_\infty + \sum_j \|q_j(\tau) - q_j^*(\tau)\|_\infty \Big)$$

converges to 0, in probability. In addition, $[p_i, q_j]_{i\in\mathcal{N}, j\in\mathcal{M}}$ converge, in probability, to the limit set of ODE (11).

Error reduction, local stability.

Armed with the above description of system dynamics, the following theorem establishes that under any solution of (11) the error E decreases with time.

THEOREM 2. *If, $p_i(t)$, $q_j(t)$, $t \in \mathbb{R}_+$, evolve according to* (11)*, then $\frac{dE}{dt} \le 0$.*

Hence, the evolution of (11) pushes the system in the right direction, reducing the error function E. The above result is true *irrespective* of whether the user profiles have the same dimension as the rank of $\tilde{\Pi}^o$. Even if the ranks of the probability matrices are underestimated, the dynamics of (11) still push towards a decrease. The proof can be found in Appendix B.

Moreover, Theorem 2 has an implication regarding the stability of (11) around the minimizers of the error E. The following corollary follows immediately from Theorem 2 and Lyapunov's stability theorem.

COROLLARY 1. *Let $x^* = [p_i^*, q_j^*]_{i\in\mathcal{N}, j\in\mathcal{M}}$ be a solution of the* RATING PREDICTION *problem* (7)*. Then,* (11) *is locally stable in the sense of Lyapunov around x^*.*

In other words, if the system starts close to a global minimum x^*, it is guaranteed to never drift away from it. Wherever E is locally convex, we can make a stronger statement:

THEOREM 3. *Let $x^* = [p_i^*, q_j^*]_{i\in\mathcal{N}, j\in\mathcal{M}}$ local minimum of E and assume that E is convex in a neighborhood around x^*. Then, there exists a δ such that if* (11) *is within a δ-neighborhood of x^*, it will converge to a point x s.t. $E(x) = E(x^*)$.*

Hence, if the ODE starts from profiles close enough to x^*, it is guaranteed the converge to profiles at an error at least as good as the error of x^*.

5. SYSTEM EXTENSIONS

5.1 Improving Predictions using Item Profiles

Recall that content items are sent to consumers along with the production profile of their producer. When an item is propagated among several consumers, rather than modifying these profiles, it may be preferable to adapt them *as the item is propagated from one consumer to the next.*

In particular, the system can be extended so that content items generated by a producer a are associated with a profile vector $t \in [0, 1]^d$, that is initialized to $t = p_a$ when the item is generated. This profile is delivered along with the item to every consumer that it passes through. However, instead of remaining static, the item profile is adapted through (8a) and (10): after the k-th viewing of the item, say at consumer b, the profile can be adapted as follows

$$t(k+1) = \Pi_{D_1}\big(t + \gamma \sum_{o\in\mathcal{O}} (\mathbb{1}_{r^o_{a,b}} - \langle t, q_b \rangle) q_b^o\big) \tag{12}$$

To gain some intuition as to why this would improve predictions, recall that each generated item belongs to a certain category. At the time of its generation, our prediction of which category an item belongs to is given by $\tilde{p}_a$, namely, the inherent probability distribution its producer. However, as an item travels through the system, the ratings of users can be used to enforce our belief on the category the item belongs to. The update rule (12) aims at exploiting the additional knowledge gained by consumer reactions.

A formal analysis of joint dynamics of a system in which publisher, consumer *and* item profiles are adapted is quite intricate and beyond the scope of the present paper. Nevertheless, we verified the performance of such a combined evolution through our experimental study in Section 7, where item profiles are introduced into the system's design.

5.2 Using Predictions for Recommendations

In this section we show that when the predicted rating distribution is correctly estimated we can generate optimal recommendations.

More specifically, we design two recommendation algorithms, each aiming to attain a different objective. The firs maximizes the fraction of content items that receive a given rating, subject to a constraint on the number of items shown to the consumer. The second maximizes the number of content items shown to the user, given a lower bound on the fraction of content items that receive a given rating. Below, we discuss these two objectives in more detail, and then present the two algorithms that indeed meet these objectives.

5.2.1 Recommendation Objectives

Assume that a sequence of content items, labeled as $k = 1, 2, 3, \ldots$ arrive at a given consumer. When a new item arrives, a recommendation algorithm decides whether to display (*i.e.*, recommend) this item to the consumer.

Let $N(k)$ be the number of items that the algorithm has recommended to the consumer up to and including the k-th item. For $o \in \mathcal{O}$, $N^o(k)$ is the number of items that were rated o. Obviously, $N^o(k) \le N(k)$, as the user reacts only to items recommended by our algorithm. Let

$$r = \lim_{k\to\infty} \frac{N(k)}{k} \qquad r^o = \lim_{k\to\infty} \frac{N^o(k)}{N(k)}$$

be the fraction of items recommended to the user and the fraction of these items that receive rating o, respectively.

We will refer to r and r^o as the *recommendation rate* and the *o-rate*, respectively.

Constrained Bitrate.

The objective of our first recommendation algorithm is to maximize r^o subject to the constraint that $r = \rho$, for some $\rho \in [0, 1]$. We call this the *constrained-bitrate* objective.

$$\text{Maximize:} \quad r^o, \tag{13a}$$

$$\text{subject to:} \quad r = \rho, \tag{13b}$$

for some $\rho \in (0, 1)$. Intuitively, the constrained-bitrate objective is most suitable when the application requires a steady, controlled rate of items of being presented to the consumer. This is natural, *e.g.*, in a network where bandwidth or the users attention span is limited: given that a device (or the consumer herself) cannot process more than a fraction r of all incoming items, (13) aims to maximize the fraction of displayed items receiving rating o.

Maximum Bitrate.

The constrained-bitrate objective is suitable when the user desires to see a specific, steady volume of content items. However, the above objective does not give guarantees on how good r^o can be: it will be the maximum possible, but that will largely depend on the quality of the content arriving at the user. In a system where bandwidth or attention span is not an issue, it makes sense giving guarantees in terms of r^o, rather than r:

$$\text{Maximize:} \quad r, \tag{14a}$$

$$\text{subject to:} \quad r^o \geq \rho, \tag{14b}$$

where, again, $\rho \in (0, 1)$. This objective guarantees that the o-rate of the content shown to the consumer will be at least ρ. It is more suitable than (13) when the user prefers guarantees on the quality—rather than the volume—of the content recommended to her; subject to these guarantees, the consumer wishes to see as much content as possible.

5.2.2 Optimal Recommendation Algorithms

In this section we present two algorithms for recommending items in that achieve objectives (13) and (14), respectively. These recommendation algorithms have access to a prediction algorithm like the one presented in Section 4: for every item k, the recommendation algorithm can obtain $\pi^o(k)$, $o \in \mathcal{O}$, the probability that the consumer gives rating o to this item.

The recommendation algorithm that solves the optimization problem (13) is the following. At each point in time, the algorithm maintains a threshold τ. When a new item arrives, the recommendation algorithm obtains from the prediction algorithm of Section 4 the quantity π^o, *i.e.*, the probability that the item receives a rating $o \in \mathcal{O}$. The item is then recommended to the user if and only if $\pi^o \geq \tau$.

The threshold τ is updated when the k-th item arrives in the following manner: it decreases whenever the quantity $\pi^o(k)$ is below τ and increases whenever $\pi^o(k)$ is above τ. In particular, after the viewing of the k-th item,

$$\tau(k+1) = \tau(k) + \gamma(k)\left(\mathbb{1}_{\pi^o(k) \geq \tau(k)} - \rho\right) \tag{15}$$

where $\gamma(k)$ is a decreasing gain factor satisfying (9).

The recommendation algorithm that solves (14) is similar. A threshold τ is again used to determine whether an item with a given π^o will be recommended to the user. The difference between the two algorithms lies in how this threshold is updated: here, the threshold increases when the item is shown and receives rating o, and decreases under other ratings. More specifically, given that an item is shown to the consumer, denote by $r \in \mathcal{O}$ the rating the consumer provided. Then the threshold τ is updated as follows

$$\tau(k+1) = \tau(k) + \gamma(k)\left(\mathbb{1}_{\pi^o(k) \geq \tau(k) \wedge r(k) = +} - \rho \mathbb{1}_{\pi^o(k) \geq \tau(k)}\right) \tag{16}$$

where $\gamma(k)$ again a decreasing gain factor satisfying (9). Intuitively, the threshold increases when the item is shown and receives rating o, and decreases under other ratings.

5.2.3 Optimality of Recommendations

We establish in this section the optimality of the recommendation algorithms introduced in Section 5.2.2. We assume here that the prediction algorithm has already converged, and gives perfect predictions of the rating distribution, *i.e.*, $\pi^o = \tilde{\pi}^o$; an evaluation of the two recommendaiton algorithms under imperfect predictions can be found in Section 6.

The algorithm will be optimal among the class of recommendation algorithms that decide whether to display the k-th item based on $\pi^o(k)$. Formally, these algorithms can be defined through a function $x(\pi^o)$, which is 1 if the algorithm shows the item given that the approval probability is π^o, and 0 otherwise. We will say that a recommendation algorithm is a *threshold algorithm* if $x(\pi^o) = \mathbb{1}_{\pi^o \geq \tau}$ for some threshold $\tau \in [0, 1]$.

The following theorem states that the algorithm meeting the constrained bitrate objective is a threshold algorithm; moreover, the recommendation algorithm (15) eventually converges to this threshold.

THEOREM 4. *Assume that the sequence $\pi^o(k)$, $k = 1, \ldots$ is i.i.d. Further assume that the c.d.f. $\mathbf{P}(\pi^o < z)$ is Lipschitz continuous and strictly monotone. Then, the optimal solution of the constrained-bitrate optimization problem is a threshold algorithm $\mathbb{1}_{\pi^o \geq \tau^*}$ and* (15) *converges to τ^* w.p.1.*

The proof of this Theorem can be found in Appendix D.

A similar result holds for the iterative algorithm (16). In this case, the problem (14) may not always be feasible: for example, if the items submitted never receive the rating o, the o-ratio cannot be bounded from below. Nevertheless, provided that the problem is feasible, there exists an optimal solution that is of a threshold algorithm.

THEOREM 5. *Assume that the sequence $\pi^o(i)$, $i = 1, \ldots$ is i.i.d.. Further assume that the c.d.f $\mathbf{P}(\pi^o < z)$ is Lipschitz continuous and strictly monotone. Then, if* (14) *is feasible, it admits an optimal solution that is a threshold algorithm $\mathbb{1}_{\pi^o \geq \tau^*}$ Moreover, if τ^* is this policy's threshold and $\gamma(k) = \frac{1}{k}$ then and* (16) *converges to τ^* w.p.1.*

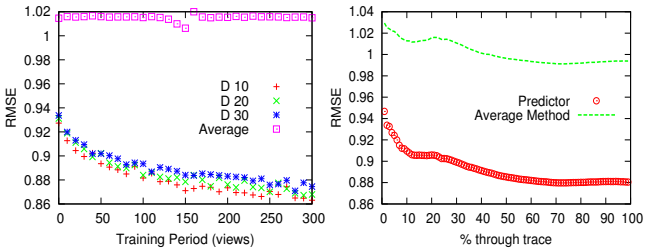
This theorem is proved in Appendix E

6. COMPARISON ON NETFLIX DATA SET

In this section, we test our rating prediction algorithm on the Netflix dataset. The wide use of the dataset as a benchmark allows us to implicitly compare the performance of our algorithm to the one achieved by state-of-the-art algorithms.

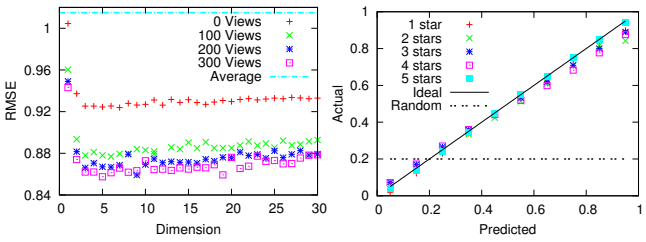
6.1 Netflix Data Set

In 2006, Netflix announced a competition for recommendation systems, and released a dataset on which competitors could train their algorithms. The Netflix dataset consists of pairs of anonymized movies and anonymized users. Each trace entry includes a timestamp, the user ID, and the user's rating (on an integer scale of 1 to 5).



(a) RMSE v. training time (b) RMSE Evolution

Figure 2: (a) RMSE as a function of the training period. (b) RMSE evolution through the course of the Netflix trace. Most of the drop in RMSE occurs in the first 10% of the trace.



(a) RMSE v. dimension (b) Rating Distribution

Figure 3: (a) Effect of dimensionality on the RMSE, which rapidly decreases until the time is insufficient to train all dimensions. (b) Comparison between predicted and empirical rating distributions.

The dataset includes both publicly-available training data, for which ratings were provided, and a testing dataset, for which ratings were not disclosed. If for every movie in the test set, we simply always predict its average rated value from the training set, this approach would yield a root mean square error of 1.0540 on the test set. The winning team of the Netflix Prize challenge generated predictions with RMSE of 0.8572 [4], a 10% improvement of the RMSE of Cinematch (0.9525), the algorithm designed by Netflix engineers.

We apply our algorithm as follows. Each movie is given a production profile $p_m \in [0,1]^d$. Similarly, each user is given an consumption profile $q_u \in [0,1]^{5\times d}$, corresponding to ratings with one, two, three, four, or five stars respectively (*i.e.*, $\mathcal{O} = \{1,2,3,4,5\}$). The Netflix dataset is arranged chronologically and as users rate movies, p for the movie and q for the user are updated according to (8a) and (8b) with each rating and are subsequently projected to D_1 and D_2 according to (10).

6.2 Evaluating our Predictions on Netflix

We evaluate the performance of our predictions of user behavior in two ways, as discussed below.

RMSE of rating prediction.

Prior to each rating event, we predict the rating that a user, u, will give to a movie, m. We make this prediction by reporting the expected rating based on our estimation of the rating probabilities, *i.e.*,

$$PredictedRating(u,m) = \sum_{o\in\mathcal{O}} o \cdot \langle q_u^o, p_m \rangle.$$

We can then compare our prediction to the user's rating for this movie as recorded in the dataset, to calculate an RMSE.

Our algorithm starts with a randomly selected p for each movie and a random q for each user and adapts them as users rate movies. To account for a training period, we discard some of the early predictions before computing the RMSE. More precisely, we compute an RMSE for predictions with training period more than k by including predictions for which either the movie or the user profile has been adapted at least k times.

Although we do not know how the Cinematch algorithm (Netflix's baseline) performs on the *training* dataset, we know that on the test set it performs roughly 10% better than the naïve algorithm that guesses the average rating for each movie. Therefore, to evaluate our effectiveness, we compare our RMSE against this "average" algorithm applied to the training dataset (100,480,507 ratings that 480,189 users gave to 17,770 movies). We know the RMSE of this "average" algorithm for both training (1.015) and test (1.05) datasets, and so we can compare to these numbers.

Figure 2(a) shows the improvement in RMSE as we vary the training period for different numbers of dimensions d of the vectors. Regardless of the length of the training period, the RMSE of the "average" method remains at 1.015. Again, though we do not know how the original Netflix algorithm (Cinematch) functioned and how it performs on the training dataset, we expect this centralized solution to achieve roughly 10% improvement (9.18 RMSE over the training dataset). Our algorithm ourperforms this value with a training period of only 10 iterations; a 15% improvement can be reached with training period of 300 views. Even when the training period is set to zero (*i.e.*, we include all predictions in the RMSE), it improves on the "average" method by 8% for all values of d, only slightly worse than Cinematch. Thus, our technique performs at least as well as some of the leading systems, even though our algorithm functions in a completely distributed manner.

Figure 2(b) shows how the RMSE evolves throughout the course of the netflix trace. Our technique makes 50% of it's improvements in RMSE during the first 10% of the trace, indicating that it performs well even in cold start situations. The algorithm consistantly tracks the performance of the "average" method while demonstrating a consistant improvement of 8%-12%. Figure 3(a) shows the effect of modifying the dimensionality of the prediction vectors on the RMSE for different training periods. As the dimensionality increases, we see a large drop; the RMSE quickly reaches a plateau between $d = 3$ and $d = 10$, and then increases slowly. We see the same trend for all training periods, with longer training periods tending to be flatter.

Thus, choosing the dimensionality is a tradeoff. On the one hand, a minimum number is needed to capture the diversity of the system, while on the other hand, too many dimensions makes the algorithm slow to learn and requires a longer training period. In practice, the best number of dimensions can be small (typically, 5 or 10), which also means that very little meta-data (5 to 10 floating-point numbers) must be transferred with each piece of content.

Predicted vs. empirical rating distributions.

We have seen that our algorithm performs well in terms of rating prediction RMSE. However, it can also be used to estimate the rating probability distribution. We now evaluate whether our estimated distribution is observed in practice across all users and movies: *e.g.*, for all events where we predict that rating o is obtained with probability 0.3, do we observe that this rating is seen 30% of the time? This is, in

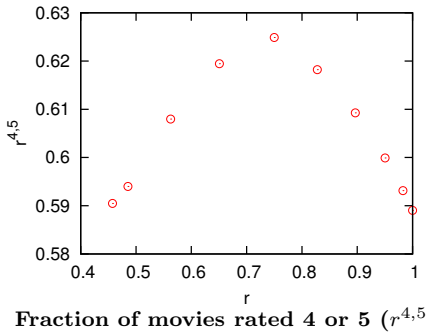


Figure 4: Fraction of movies rated 4 or 5 ($r^{4,5}$) against total fraction of movies viewed (r).

effect, a measure of the precision of the rating predictor.

We measure the goodness of fit of our distribution to the empirical distribution in Figure 3(b). To compute it, we first bin our predicted probability into 10 bins (0 to 0.1, 0.1 to 0.2, etc.). We then compare the bin value to the actual rate of occurrence in the bin, for different ratings. Note that for all ratings, the square correlation coefficient R^2 is above 0.98 indicating a very good match. The slope of the best fit line is thus nearly 1. Another metric is the distance of our result to the line $y = x$ which would represent an ideal predictor. Random guessing, for instance, has a mean distance of 0.415. Our rating predictor brings a large improvement, with distance errors as small as 6×10^{-4}.

This accuracy implies that the system can do more than provide a single estimated rating for a specific movie and user, it can actually characterize its behavior on a finer grain which may help provide additional confidence. As an example, it is interesting to note that users are more predictable when they express strong opinions (5 starts, 1 stars), slightly less predictable when they are neutral, and even less predictable when they indicate 4 or 2 stars.

6.3 Evaluating Recommendations

Another application of our prediction technique is to apply a threshold as in Section 5.2 to limit the bytes transferred to the user while maximizing the fraction of content they enjoy (here defined as all movies they rated 4 or 5).

Figure 4 plots the ratio of movies viewed and the fraction of these that are enjoyed for different threshold values. A low threshold value corresponds to almost all movies being shown (on the right on the Figure). Using our prediction probability and a larger threshold allows to both reduce the volume of movies shown, until about 75% while achieving a maximum fraction of 63% content enjoyed. However, applying larger threshold does not improve this fraction since learning begins to slow.

This indicates that using the rating prediction is effective to select the most pertinent content to display to the user, providing both an improvement in bandwidth as well as an improved user experience. On the other hand, it is important to make sure that users get exposed to sufficient content for the algorithm to train.

7. CASE STUDY: WEBDOSE

The evaluation of our rating prediction algorithm on the Netflix data set demonstrates that it can, indeed, be used to make accurate predictions. However, it does not illustrate its behavior in a distributed, heterogeneous environment.

Therefore, we implemented our system as a Facebook application ("WebDose"), which allows Facebook users to view, rate, and share content in the form of web pages. We used Facebook as a distribution platform to ensure that the application would be immediately available to a wide range of users without requiring extensive development overhead.

7.1 Application Description

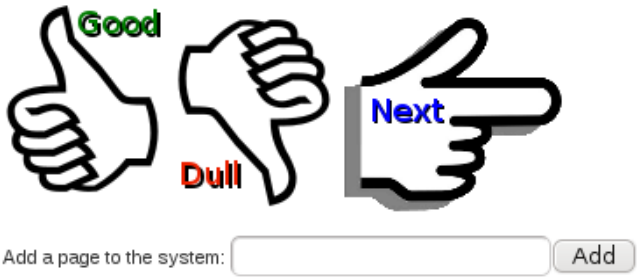


Figure 5: Function buttons in the WebDose application. These buttons are displayed along with the web page to be rated by the user.

User interface.

When users log into the system, WebDose presents them with a screenshot of a web page and three rating icons: "thumbs up," "thumbs down," or "next" (see Fig. 5). We interpret each of these ratings as positive, negative or neutral, respectively (*i.e.*, $\mathcal{O} = \{+, -, \varnothing\}$). By design, users are not shown a new page until they have clicked a button and thus rated the page they are currently viewing. Additionally, users have the option to "Add" (*i.e.*, produce) content to the system by typing a url into a text box at the bottom of their screen. WebDose, thus, follows the same producer/consumer model used throughout this paper.

Emulating Content Sharing.

We emulate content sharing through discrete "contact events" between users. When a user logs into the application (or refreshes the page), she experiences a "contact" with one user that has also logged in to WebDose within the previous three hours. If no such user exists, no contact occurs. A user may experience many contacts during a browsing session as other users log in and trigger their own "contact events."

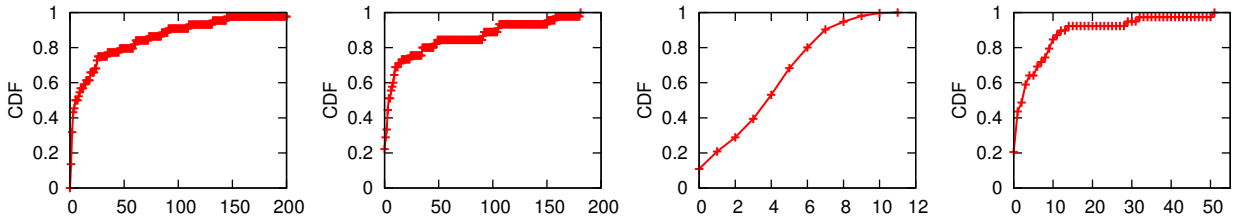
To account for social relationships, when multiple people have logged in the system in the past 3 hours, we bias contact events between users that are Facebook friends. If multiple possible swaps exist, we choose to swap with a random friend half the time and the other half of the time we choose randomly among all users (including friends).

Content is "produced" when a user adds a web page via the provided interface. Each user is limited to viewing webpages in a "cache" whose contents change dynamically. When two users "meet", each item in each user's cache is randomly reassigned to the other user with probability 0.5. Web pages thus perform a "random walk" through WebDose users, and there is no centralized control dictating how they move.

Behind-the-scenes prediction.

The first time a user u logs into the system, she is assigned a random production profile p_u and a random consumption profile q_u. The dimension of these profiles is set to $d = 10$; this will be examined in Section 7.3.

We also incorporate item profiles in WebDose. When an



(a) Contacts per user (b) Ratings given per user (c) Views per page (d) Pages published per user

Figure 6: CDFs of various facets of user behavior throughout the experiment. In (a) the most active 50% experience a wide range of contacts. In (b) we see that 20% of users rate more than 50 pages each, indicating high use. In (c) we observe that no page is viewed more than 11 times but half the pages are viewed more than 4 times. Finally, (d) indicates that the majority of content is published by a small number of users.

item is propagated among several consumers, rather than modifying these profiles, it is preferable to adapt them *as the item is propagated from one consumer to the next*. The system is extended so that content items generated by a producer a are associated with a profile vector $t \in [0,1]^d$, that is initialized to $t = p_a$ when the item is generated. This profile is delivered along with the item to every consumer that it passes through. However, instead of remaining static, the item profile is adapted through (8a) and (10). A formal analysis of joint dynamics of a system in which publisher, consumer *and* item profiles are adapted is quite intricate and beyond the scope of the present paper. Nevertheless, we verify the performance of such a combined evolution through the WebDose experiment.

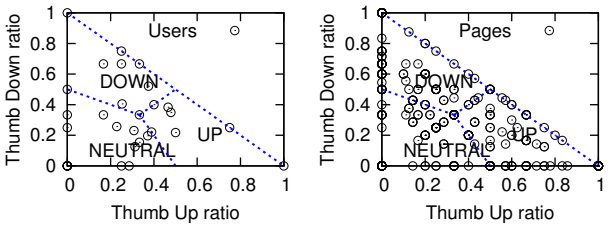
As web pages are rated by other users of the system, t_w and q_u are updated according to Eq. (8a) and (8b) respectively. Both the predicted as well as the actual rating are logged. Web pages in the system carry an identifier of the web page's producer, *i.e.*, the user that added it to WebDose. When a user rates a page, the system stores the producer and rating locally. Subsequently, when the two users meet, the web page consumer informs the web page producer of the rating given to its content. The producer then uses this information to update its profile p_u according to Algorithm (8a). All updates are followed by an immediate projection to D_1, D_2, as in (10).

Recommendation.

WebDose uses our prediction algorithm to select pages to show to a user. Each time a page is shown, WebDose first calculates π^+ for each as-yet-unrated web page in the user's cache. The page with the highest π^+ is then shown to the user. Once that page has been rated, π^+ is recalculated for each unrated page and the new highest page is shown. We chose this form of recommendation to ensure that a user is always able to view all of the content in their cache, and that WebDose would not be hindered from training user's profiles through pages not viewed.

7.2 WEBDOSE Experiment

During a 33 day period, 43 users spanning 12 time zones registered for WebDose and viewed or added 326 web pages. Although these small numbers make learning user preferences difficult, we will show in Section 7.3 that the system was able to adapt well. Users were free to log in as frequently as they wished and could rate as many pages as they wanted. Once a page was rated, it was never shown to that user again. As an incentive to add quality content to the system, we provided each user with statistics of the ratings their content received, as well as their ranking in



(a) User Ratios (b) Page Ratios

Figure 7: Regions are marked to indicate dominant user behavior. In (a) users are marked according to their ratio of thumb ups to thumb downs. Users generally fall into the thumbs down or neutral areas. However, (b) shows that this is not a property of the pages themselves as pages fill the space uniformly.

"thumbs-up" and "thumbs-down" received.

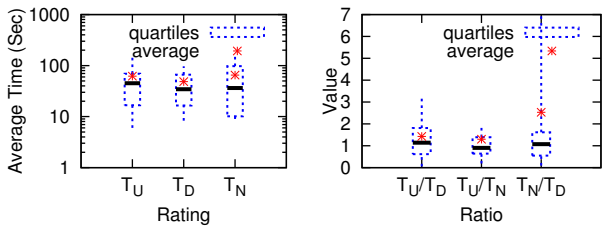
User activity.

Here we examine some of the trends in the way that users interact with this system. Because users are given few instructions beyond "thumb it up if you like it, down if you don't, and remember to add pages," we feel that the broad trends from this deployment would hold in any application built with a similar model.

Figure 6(a) provides the CDF of the number of "contact-events" that users experienced. Half of the users had more than 10 contacts and 20% had over 50. Figure 6(b) shows the CDF of the ratings given by each user. The most active user rated over 180 pages; 25% of users rated more than 20 pages. This is consistent with a real mobile environment: most users look for web pages occasionally, while an active few regularly check for pages.

For the system to function as intended the pages must be viewed frequently enough that the item profile t is trained. Figure 6(c) shows the frequency with which pages are viewed. The high number of pages in the system (over 300) compared to the number of users means that each page is viewed by only a limited number of users. Over the course of our experiment, no page was viewed more than 11 times. However, more than half the pages were viewed more than 4 times, which proves to be sufficient.

Finally, we examine the process by which web pages are added to the system. Figure 6(d) examines the number of pages added to the system by each user. The top 20% of users each produce more than 10 pages with the highest producer contributing around 50 web pages (16% of all web



(a) Average (b) Ratio

Figure 8: In these boxplots, the ends of the whiskers denote the 5^{th} and 95^{th} percentiles of users while the box denotes the 25^{th} and 75^{th} percentiles. The dark bar is the median and the star is the mean. In (a) we look at the raw times taken by users to rate a page. (b) examines the ratio of times taken by a single user. Thumbing down a page is faster that of thumbs up and neutral, both of which take roughly the same amount of time.

pages in the system). It is not necessarily true that the highest producers are also the highest consumers. The relatively low number of high producers indicate that it is crucial that item profiles are introduced, otherwise all pages from the same producer will have the same profile, which would slow learning. Item profiles allow the system to learn and distinguish different pages from the same producer, accelerating convergence by training individual web pages while the production profiles train.

User rating behavior.

Here, we focus on how users rated the pages in WebDose. Define the *"Thumb up (down)" ratio* to be the fraction of all pages that a user rates positively (negatively). Figure 7 examines these ratios by users and by pages.

Figure 7(a) shows that the majority of users favor neutral ratings, followed closely by those that mostly rate pages thumbs down. With most people thus being generally ambivalent or negative on most content, it is important that the system makes accurate predictions.

Figure 7(b) shows the rating ratios given to individual pages. Despite the generally negative user behavior, pages themselves fill the interest space uniformly. The wide range of topics serves to train the system as well as demonstrate the need for accurate recommendation.

We next examine the speed with which users provide ratings to the system. Intuitively, a user will spend more time looking at a web page that she finds enjoyable and quickly reject pages that do not interest her. Figure 7.2 shows comparisons of the speed with which users make decision about various ratings.

We define T_U,T_D, and T_N as the time taken to rate a page "up," "down," or "neutral," respectively. In Figure 8(a) we see the distribution of times taken to rate a page. "Thumb up" indeed takes the longest to select, followed by neutral. By examining the ratio of these quantities, as in Figure 8(b), we see that 70% of users thumb a page up slower then they thumb a page down (*i.e.* $\frac{T_U}{T_D} > 1$). Thus, one may be able to judge the relative like or dislike of a page simply by observing how long the user takes to make a decision. Interestingly, it may in principle be possible to remove ratings completely, relying only on time to judge a users interest, or to nuance the ratings with timing information.

7.3 Predictive Power

	a priori		convergence	
	R^2	slope	R^2	slope
Thumb Up	0.43	0.29	0.98	0.95
Thumb Down	0.80	0.63	0.98	1.20
Neutral	0.96	1.07	0.95	1.29

Table 1: R^2 values and slopes for thumb rate predictions both *a priori* and after convergence.

WebDose *Prediction accuracy.*

To assess the correctness of predictions in WebDose, we repeated the evaluation of our predictors with respect to the two metrics we considered over the Netflix data set, the RMSE and the goodness of fit of the predicted distribution.

Figure 9(a) displays the RMSE of our predictions as a function of the dimension d of our profiles. The dimension 10 was used in the actual experiment and other values were obtained by running our algorithm on the collected trace.

To compute the RMSE, we associate the values -1,0,+1 to $-, \varnothing, +$ respectively. Before the rating of a web page w by a user u, we again generate our estimate of the rating as

$$PredictedRating(u, v) = \sum_{o \in \{-1,0,+1\}} o \cdot \langle t_w, q_u^o \rangle,$$

where t_w, q_u the item and consumption profiles, respectively. As in Netflix, we again observe the error quickly decreases up to a dimension of 10, after which it remains constant. We note that the RMSE for $d = 10$ is 0.819, 42% less than the RMSE obtained by predictions obtained by the running average ratio of each rating.

The goodness of fit of our distribution to the empirical distribution is illustrated in Figure 9(b) (using the same metrics from Sec. 6.2). The values for R^2 and the slope of the best fit lines are provided in Table 1.

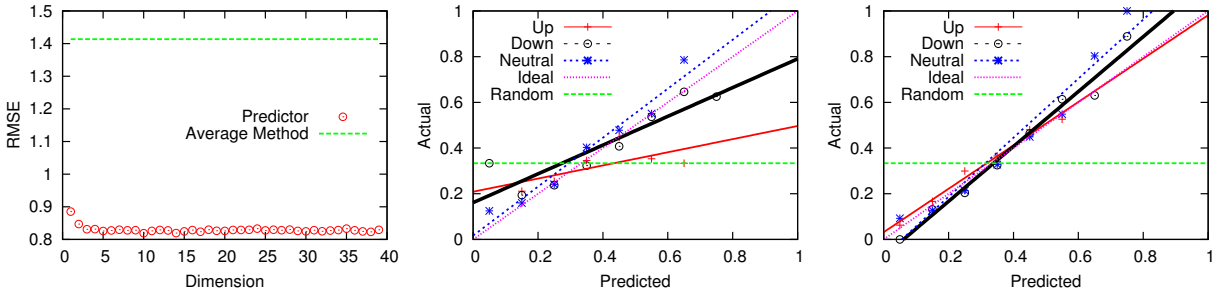
The outcomes "Thumb down" and "neutral" are both predicted accurately, with slopes above 0.63 and exhibiting good fits (R^2 above 0.8). The "thumbs-up" predictions are considerably worse, with a slope of 0.29. This is not surprising, given that most ratings in the system were for "thumbs-down" and "neutral": the experiment does not provide enough data to train quickly enough for "thumbs-up" ratings.

Since WebDose has far more complicated dynamics than the simple model we analysed, it is interesting to investigate the convergence of our prediction scheme. For this reason, we repeated the goodness of fit test using instead the content and consumption profiles *at the end* of the experiment. The result is shown in Figure 9(c), with R^2 vales and slopes again found in Table 1. It is quite clear that the content and consumption profiles have adapted to fit the empirical distribution of ratings very well.

8. CONCLUSIONS

User generated content is a primary factor to the success of the social web as it connects users through content sharing. It presents unique challenges for collaborative filtering due to a large volume and the sensitivity of sharing information beyond one's immediate circle of acquaintances. This paper proves that information exchange can be deployed only among trusted pairs while providing strong guarantees on the rating prediction error, thanks to a fully distributed and asynchronous learning algorithm.

Our results expand the scope of recommender systems to operate on top of any social content sharing platform, which unveils important research questions. The case where trust is not reciprocal (such as the follower relationship within Twitter) remains an interesting open case. Our algorithm



(a) RMSE (b) *a priori* prediction (c) convergence

Figure 9: Predictions of the thumb rate of pages. (a) shows the effect of the dimension chosen at the outset of the experiment. As the dimension increases, the error decreases. Our selection of 10 occurs below the knee of the curve. In (b) and (c) pages are binned according to estimated percentage of "thumbing" them in a given direction. (b) shows *a priori* prediction and moderate accuracy, with slopes from 0.29 to 1.07. (c) shows convergence of π at the end of the experiment and has slopes from 0.95 to 1.3. In all figures, the green line gives a comparison to other predictors.

could leverage secure multiparty computation, to provide at a smaller cost the privacy guarantee offered by centralized schemes. Finally, our model can be used to analyze how social proximity, captured through "rate of delivery" for producer-consumer pairs, impacts the efficiency of learning. Both questions are interesting open problems.

9. ACKNOWLEDGEMENTS

This work was partially funded by the European Commission under the FIRE SCAMPI project (FP7-IST-258414) and by the French National Research Agency (ANR) under the PROSE project (VERSO program).

10. REFERENCES

[1] Acquisti, A., and Grossklags, J. What can behavioral economics teach us about privacy? In *Digital Privacy: Theory, Technologies and Practices* (2007), pp. 363–377.

[2] Angwin, J. US seeks web privacy 'bill of rights'. *Wall Street Journal* (Dec. 17th 2010).

[3] Azar, Y., Fiat, A., Karlin, A., McSherry, F., and Saia, J. Spectral analysis of data. In *STOC* (2001).

[4] Bell, R. M., and Koren, Y. Scalable collaborative filtering with jointly derived neighborhood interpolation weights. In *ICDM* (2007).

[5] Brandimarte, L., Acquisti, A., and Loewenstein, G. Privacy concerns and information disclosure: An illusion of control hypothesis. In *CIST* (2010).

[6] Candès, E., and Tao, T. The power of convex relaxation: Near-optimal matrix completion. *IEEE Trans. Inform. Theory 56*, 5 (2009), 2053–2080.

[7] Candès, E. J., and Recht, B. Exact matrix completion via convex optimization. *Found. of Comput. Math. 9* (2008), 717–772.

[8] Canny, J. Collaborative filtering with privacy via factor analysis. In *SIGIR* (2002).

[9] Castagnos, S., and Boyer, A. Personalized Communities in a Distributed Recommender System. In *ECIR* (2007), pp. 343–355.

[10] Haraux, A. How to differentiate the projection on a convex set in Hilbert space. Some applications to variational inequalities. *J. Math. Soc. Japan 29* (1977), 615–631.

[11] Ioannidis, S., and Massoulié, L. Surfing the blogosphere: Optimal personalized strategies for searching the web. In *Proc. IEEE Infocom* (2010).

[12] Kempe, D., and McSherry, F. A decentralized algorithm for spectral analysis. *Journal of Computer and System Sciences 74*, 1 (feb 2008).

[13] Keshavan, R., Montanari, A., and Oh, S. Matrix completion from a few entries. *Trans. Inform. Theory* (2010).

[14] Keshavan, R., Montanari, A., and Oh, S. Matrix completion from noisy entries. *JMLR* (2010).

[15] Korada, S. B., Montanari, A., and Oh, S. Gossip PCA. *SIGMETRICS* (2011).

[16] Koren, Y. Collaborative filtering with temporal dynamics. In *KDD* (2009).

[17] Kushner, H. J., and Yin, G. *Stochastic Approximation and Recursive Algorithms and Applications*, 2nd ed. Springer, 2003.

[18] Lathia, N., Hailes, S., and Capra, L. Private distributed collaborative filtering using estimated concordance measures. In *RecSys* (Oct. 2007).

[19] Liu, K., Bhaduri, K., Das, K., Nguyen, P., and Kargupta, H. Client-side web mining for community formation in peer-to-peer environments. *WEBKDD* (2006).

[20] Maculan, N., Santiago, C., Macambira, E., and Jardim, M. An $O(n)$ algorithm for projecting a vector on the intersection of a hyperplane and a box in $\mathbb{R}^n$. *J. of Opt. Th. and App. 117*, 3 (2003), 553–574.

[21] Michelot, C. A finite algorithm for fiding the projection of a point onto the canonical simplex of $\mathbb{R}^n$. *J. of Opt. Th. and App. 50*, 1 (1986), 195–200.

[22] Miller, B., Konstan, J., and Riedl, J. PocketLens. *ACM Transactions on Information Systems 22* (2004), 437–476.

[23] Oja, E. On stochastic approximation of the eigenvectors and eigenvalues of the expectation of a random matrix. *J. of Math. Anal. and App.*, 106 (1985), 69—84.

[24] Polat, H., and Du, W. Privacy-preserving collaborative filtering using randomized perturbation techniques. In *IEEE ICDM* (2003), pp. 625–628.

[25] Ruffo, G., and Schifanella, R. A peer-to-peer recommender system based on spontaneous affinities. *ACM Transactions on Internet Technology 9* (2009).

[26] Salakhutdinov, R., and Mnih, A. Probabilistic matrix factorization. *Advances in Neural Information Processing Systems 20* (2008), 1257–1264.

[27] Takács, G., Pilászy, I., Németh, B., and Tikk, D. Scalable collaborative filtering approaches for large recommender systems. *JMLR 10* (2009), 623–656.

[28] Tomozei, D.-C., and Massoulié, L. Distributed user profiling via spectral methods. In *SIGMETRICS* (2010).

[29] Vascellaro, J. E. Google agonizes on privacy as ad world vaults ahead. *Wall Street Journal* (Aug. 10th 2010).

[30] Wang, J., Pouwelse, J., Lagendijk, R., and Reinders, M. Distributed collaborative filtering for peer-to-peer file sharing systems. In *SAC* (2006).

[31] Wingfield, N. Microsoft quashed effort to boost online privacy. *Wall Street Journal* (Aug. 2nd 2010).

[32] Zhou, Z., Wright, J., Li, X., Candès, E. J., and Ma, Y. Stable principal component pursuit. In *ISIT* (2010).

APPENDIX

A. PROOF OF THEOREM 1

Our proof follows the standard method outlined in Kushner and Yin [17]. Assume that the duration if each timeslot is T. We denote the time at which the transmission from i to j by

$$t_{i\rightsquigarrow j}(k) \in [0,T) \cup \{\infty\},$$

By convention, $t_{i\rightsquigarrow j}(k) = \infty$ denotes that no such transmission takes place within the k-th slot. Put differently, $\{t_{i\rightsquigarrow j}(k) < \infty\} = \{a_{i,j}(k) = 1\}$.

We first precisely define the "minimum forces" z_{D_1} and z_{D_2}. Let $D \subseteq \mathbb{R}^d$ be a closed convex subset of $\mathbb{R}^d$, and consider two vectors $x \in \mathbb{R}^d$, $p \in D$. Following Kushner and Yin [17], we define $z_D(p,x) \in \mathbb{R}^d$ as follows.

$$z_D(p,x) = \lim_{\delta\to 0^+} \frac{\Pi_D(p+\delta x) - (p+\delta x)}{\delta},$$

where $\Pi_D : \mathbb{R}^d \to D$ is the orthogonal projection to D. For D closed and convex, the above limit is guaranteed to exist (see, *e.g.*, [10]). The quantity $z_D(p,x)$ is called the minimum additional force to maintain p within D because, if "force" x is applied to p, *i.e.*, $\frac{dp}{dt} = x$ then, for small $\delta > 0$

$$p + \delta x + \delta z_D(p,x) \simeq \Pi_D(p+\delta x).$$

Using the above notation, ODE (11) can be written as

$$\frac{dp_i}{dt} = -\nabla_{p_i} E + z_{D_1}(p_i, -\nabla_{p_i} E), \qquad a \in \mathcal{N},$$
$$\frac{dq_j}{dt} = -\nabla_{q_j} E + z_{D_2}(q_j, -\nabla_{q_j} E). \qquad b \in \mathcal{M},$$

Note that, within a timeslot, as users encounter each other, their profiles change through (8a) and (8b). As a result, when a producer i encounters a consumer j, their profiles may differ from the ones they had in the beginning of the timeslot. The following lemma however states that, if we assume that during encounters profile had their initial values, we introduce only a very small error compared to how the actual process behaves:

Lemma 1. *Let $p_i(\tau)$, $q_j(\tau)$, $\tau \in [0,T)$, the values of the profiles of users $i \in \mathcal{N}$ and $j \in \mathcal{M}$ at some time τ since the beginning of the timeslot. Denote by $E(\tau) = \{(a_i \rightsquigarrow b_i) : t_{i_i\rightsquigarrow b_i} \le \tau\}$ the set of all encounter events that have taken place until time τ. Then, for any $\tau \in [O,T)$,*

$$p_i(\tau) = p_i(0) + \gamma \sum_{j:i\rightsquigarrow j\in E(\tau),o} (\mathbb{1}_{r_{i,j}=o} - \langle p_i(0), q_j^o(0)\rangle) q_j^o(0) + \beta_i, \tag{17a}$$

$$q_j^o(\tau) = q_j^o(0) + \gamma \sum_{i:i\rightsquigarrow j\in E(\tau)} (\mathbb{1}_{r_{i,j}=o} - \langle p_i(0), q_j^o(0)\rangle) p_i(0) + \beta_j, \tag{17b}$$

where β_i, β_j are random variables s.t. $\|\beta_i\|_2, \|\beta_j\|_2 \le K\gamma^2$, where K depends only on the system dimensions $|\mathcal{N}|$, $|\mathcal{M}|$, $|\mathcal{O}|$ and $|\mathcal{F}|$.

Proof. We will prove the lemma by induction on the encounters between users. W.l.o.g., we assume that encounters occur at distinct times: encounters that occur simultaneously can be ordered arbitrarily if they do not involved the same user, or by the order within which (8a) and (8b) take place, otherwise.

For the induction basis, observe that (17) holds at $\tau = 0$, before any encounters have taken place. Suppose that it also holds at τ^-, right before an encounter $i' \rightsquigarrow j'$ takes place. Then, it still holds time τ^+, right after the encounter took place, for all $i \ne i'$ and all $j \ne j'$. Observe that, because $p_i \in D_1$ and $q_j \in D_2$, for all $i \in \mathcal{N}$ and all $j \in \mathcal{M}$

$$|\mathbb{1}_{r_{i,j}=o} - \langle p_i, q_j^o\rangle| \le 1, \ \|q_j^o\|_2 \le |\mathcal{F}|, \ \|p_i\|_2 \le 1$$

As a result, using the induction hypothesis we can show that

$$|\langle p_{i'}(\tau^-), q_{j'}^{o'}(\tau^-)\rangle - \langle p_{i'}(0), q_{j'}^{o'}(0)\rangle| = O(\gamma)$$

where the constants in $O(\gamma)$ depend only on the system dimensions. By (8a) we have that

$$\begin{aligned} p_{i'}(\tau^+) &= p_{i'}(\tau^-) + \sum_{o'} (\mathbb{1}_{r_{i',b'}=o'} - \langle p_{i'}(\tau^-), q_{j'}^{o'}(\tau^-)\rangle) q_{j'}^{o'}(\tau^-) \\ &= p_i(0) + \gamma \sum_{j:i\rightsquigarrow j\in E(\tau),o} (\mathbb{1}_{r_{i,j}=o} - \langle p_i(0), q_j^o(0)\rangle) q_j^o(0)) + \beta_{i'} + \\ &\quad + \gamma \sum_{o'} \big(\mathbb{1}_{r_{i',b'}=o'} - \langle p_{i'}(0), q_{j'}^{o'}(0)\rangle + O(\gamma)\big)\big((q_{j'}^{o'}(0) + O(\gamma)\big) \end{aligned}$$

where the constants in the $O(\gamma)$ terms only depend on the system dimensions. As, by the induction hypothesis, $\beta_{i'} = O(\gamma^2)$, the induction step follows; a similar argument can be made for the consumption profile q_j^o. □

Note that the above proof by induction can be easily extended to the case where consumers report their ratings and consumption profiles at an arbitrary time after $t_{i\rightsquigarrow j}$.

Lemma 17 implies that the evolution of profiles from one timeslot to the next can be written as:

$$p_i(k+1) = \Pi_{D_1}\big(p_i(k) + \gamma(k) Y_i(k) + \beta_i(k)\big),$$
$$q_j(k+1) = \Pi_{D_2}\big(q_j(k) + \gamma(k) Y_j(k) + \beta_j(k)\big)$$

where

$$Y_i = \sum_j \mathbb{1}_{a_{i,j}(k)=1} \sum_{o\in\mathcal{O}} (\mathbb{1}_{r_{i,j}=o}(k) - \langle p_i(k), q_j^o(k)\rangle) \cdot q_j^o(k)$$

and $Y_j = (Y_j^{o_1}, Y_j^{o_2}, \ldots, Y_j^{o_{|\mathcal{O}|}})$ such that

$$Y_j^o = \sum_i \mathbb{1}_{a_{i,j}(k)=1} (\mathbb{1}_{r_{i,j}=o}(k) - \langle p_i(k), q_j^o(k)\rangle) \cdot p_j(k).$$

while $\beta_i(k), \beta_j(k)$ are random variables s.t. $\|\beta_i(k)\|_2 = O(\gamma^2(k)), \|\beta_j\|_2 = O(\gamma_k^2)$. By the independence of propagation from the content categories, we have that

$$\mathbb{E}[\mathbb{1}_{i\rightsquigarrow j}(k)\cdot\mathbb{1}_{r_{i,j}=o}(k)] = \lambda_{i,j}\cdot\tilde{\pi}^o_{i,j}(k).$$

As a result $\mathbb{E}[Y_i] = \nabla_{p_i}E$ and $\mathbb{E}[Y_j] = \nabla_{q_j}E$. Moreover, by the fact that $p_i \in D_1$ and $q_j \in D_2$, $\mathbb{E}[\|Y_i(k)\|_2^2] < \infty$ and $\mathbb{E}[\|Y_j\|_2^2] < \infty$ uniformly on k. Thus, the assumptions (A2.1)-(A2.5) of Theorem 2.3 in Chapter 5 of Kushner and Yin [17] are satisfied, and the theorem follows. □

B. PROOF OF THEOREM 2

To prove Theorem 2, we will make use of the following lemma (Lemma 5 in [11]) concerning the orthogonal projection on the positive simplex.

LEMMA 2. *Let $D = \{x \in \mathbb{R}_+^d : \sum_{i=1}^d x_i = 1\}$ and, for $x \in D$, denote by $I_0(x) = \{i : x_i = 0\}$ the set of zero-valued coordinates of x. Then, there exists a set $B \subset I_0(x)$ such that $z_D(x,y) = z$ where*

$$z_i = \begin{cases} -y_i, & i \in B \\ -\frac{1}{|\bar{B}|}\sum_{j\in\bar{B}} y_j, & i \in \bar{B}. \end{cases}$$

Moreover, for every $i \in B$, $y_i - \frac{1}{|B|}\sum_{j\in\bar{B}} y_j < 0$.

We have that

$$\begin{aligned}\frac{dE}{dt} =& \sum_i \langle \nabla_{p_i}E, \frac{dp_i}{dt}\rangle + \sum_j \langle \nabla_{q_j}E, \frac{dq_j}{dt}\rangle \\ \overset{(11)}{=}& \sum_i \left[-\langle \nabla_{p_i}E, \nabla_{p_i}E\rangle + \langle \nabla_{p_i}E, z_{D_1}(p_i, -\nabla_{p_i}E)\rangle\right] \\ & \sum_j \left[-\langle \nabla_{q_j}E, \nabla_{q_j}E\rangle + \langle \nabla_{q_j}E, z_{D_2}(q_j, -\nabla_{q_j}E)\rangle\right]\end{aligned}$$

From Lemma 2, for every $i \in \mathcal{N}$ there exists a $B_i \subseteq \mathcal{F}$ such that $\langle \nabla_{p_i}E, z_{D_1}(p_i, \nabla_{p_i}E)\rangle$ equals

$$\sum_{f\in B_i} (\partial E/\partial p_{i,f})^2 + \frac{1}{|\bar{B}_i|}\Big(\sum_{f\in\bar{B}_i} \partial E/\partial p_{i,f}\Big)^2.$$

Similarly, for every $j \in \mathcal{M}$ and every $f \in \mathcal{F}$ there exists a $B_{j,f} \subset \mathcal{O}$ such that $\langle \nabla_{q_j}E, z_{D_2}(q_j, -\nabla_{q_j}E)\rangle$ equals

$$\sum_f \left[\sum_{o\in B_{j,f}} \left(\partial E/\partial q^o_{i,f}\right)^2 + \frac{1}{|\bar{B}_{j,f}|}\Big(\sum_{o\in\bar{B}_{j,f}} \partial E/\partial q^o_{i,f}\Big)^2\right].$$

Using the above, dE/dt becomes:

$$\begin{aligned}\frac{dE}{dt} =& -\sum_i\sum_{f\in\bar{B}_i}\Big(\frac{\partial E}{\partial p_{i,f}} - \frac{1}{|\bar{B}_i|}\sum_{f'\in\bar{B}_i}\frac{\partial E}{\partial p_{i,f'}}\Big)^2 \qquad (18)\\ & -\sum_{j,f}\sum_{o\in\bar{B}_{j,f}}\Big(\frac{\partial E}{\partial q^o_{i,f}} - \frac{1}{|\bar{B}_{j,f}|}\sum_{o'\in\bar{B}_{j,f}}\frac{\partial E}{\partial q^o_{i,f}}\Big)^2 \le 0. \quad \square\end{aligned}$$

C. PROOF OF THEOREM 3

We begin by stating the Karush-Kuhn-Tucker (KKT) conditions for OUTPUT PREDICTION. The Lagrangian of (7) is

$$\begin{aligned}L =& E + \sum_i \mu_i(\sum_f p_{i,f} - 1) + \sum_{j,f}\nu_{j,f}(\sum_o q^o_{j,f} - 1) \\ & -\sum_{i,f}\xi_{i,f}p_{i,f} - \sum_{j,f,o}\zeta^o_{j,f}q^o_{j,f}\end{aligned}$$

where $\mu_i, \nu_{j,f}, \xi_{i,f}, \zeta^o_{j,f}$ are the Lagrangian multipliers of the constraints (7b) and (7c). Thus, the KKT conditions can be written as

$$\frac{\partial E}{\partial p_{i,f}} + \mu_i - \xi_{i,f} = 0,\ \xi_{i,f} \ge 0,\ \xi_{i,f}p_{i,f} = 0,\ \forall a,f \qquad (19a)$$

$$\frac{\partial E}{\partial q^o_{i,f}} + \nu_{j,f} - \zeta^o_{j,f} = 0,\ \zeta^o_{j,f} \ge 0,\ \zeta^o_{j,f}q^o_{j,f} = 0,\ \forall b,f,o \qquad (19b)$$

$$p_i \in D_1,\ q_j \in D_2, \quad \forall a,b \qquad (19c)$$

The following lemma holds

LEMMA 3. *Let $[p_i^*, q_j^*]_{i\in\mathcal{N}, j\in\mathcal{M}}$ be profiles for which $dE/dt = 0$. Then, there exist $\mu_i, \nu_{j,f}, \xi_{i,f}, \zeta^o_{j,f}$ for which the KKT conditions are satisfied.*

PROOF. From (18) and Lemma 2, if $dE/dt = 0$ at then for every $i \in \mathcal{N}$ there exists a $B_i \subset \mathcal{F}$ such that, for $\mu_i = -\frac{1}{|B_i|}\sum_{f'\in\bar{B}_i}\frac{\partial E(x^*)}{\partial p_{i,f'}}$

$$\begin{aligned}& p^*_{i,f} = 0,\ E(x^*)/\partial p_{i,f} + \mu_i > 0, && \forall f \in B_i, \\ & \partial E(x^*)/\partial p_{i,f} + \mu_i, && \forall f \in \bar{B}_i.\end{aligned}$$

Similarly, for every $j \in \mathcal{M}$ and $f \in \mathcal{F}$ there exists a $B_{j,f} \subset \mathcal{O}$ such that for $\nu_{j,f} = -\frac{1}{|B_{j,f}|}\sum_{o'\in\bar{B}_{jf}}\frac{\partial E(x^*)}{\partial q^o_{j,f'}}$

$$\begin{aligned}& (q^o_{j,f})^* = 0,\ E(x^*)/\partial q^o_{j,f} + \nu_{j,f} > 0, && \forall o \in B_{j,f}, \\ & \partial E(x^*)/\partial q^o_{j,f} + \nu_{j,f} = 0 && \forall o \in \bar{B}_{j,f}.\end{aligned}$$

It is easy to verify that p_i^*, q_j^*, along with the above values μ_i, $\nu_{j,f}$ and the values

$$\xi_{i,f} = \max(0, \frac{\partial E(x^*)}{\partial p_{i,f}} + \mu_i),\ \zeta^o_{i,f} = \max(0, \frac{\partial E(x^*)}{\partial q^o_{j,f}} + \nu_{j,f})$$

satisfy the KKT conditions (19). □

Suppose now that x^* is a local minimum and that E is locally convex at a δ-neighborhood around this minimum. Then, by Theorem 2, there exists a $\delta' \le \delta$ such that if the ODE (11) starts from a δ'-neighborhood of x^* it will remain in this neighborood. Consider now the problem (7) with the additional constraint $\|x - x^*\|_2 < \delta'$. From Lemma (3), that any limit point of (11) will satisfy the KKT constraints of this problem with a 0 lagrange multiplier for the added condition. Hence, any limit point of (11) must attain $E(x^*)$, as E is convex in this δ neighborhood. □

D. PROOF OF THEOREM 4

We first show that the optimal recommendation policy is a threshold policy. Let ν be the probability measure of π_+, that is, for every Borel set A, $\mathbf{P}(\pi_+ \in A) = \int_A d\nu(s)$. Since the sequence $\pi_+(i)$, $i = 1, \ldots$ is i.i.d., the constrained-bitrate problem is equivalent to maximizing $\frac{1}{\rho}\int_0^1 sx(s)d\nu(s)$ subject to

$$\int_0^1 x(s)d\nu(s) = \rho \qquad (20)$$

Consider two probability measures ν_0 and ν_1 that are absolutly continuous w.r.t. ν. Then, by the Radon-Nikodym theorem, there exist functions ℓ_0, ℓ_1 such that $\nu_0(A) = \int_A \ell_0(s)d\nu(s)$ and $\nu_1(A) = \int_A \ell_1(s)d\nu(s)$. The Neyman-Pearson lemma states the function x that maximizes $\mathbb{E}_1[1 - x(\pi_+)]$, subj. to $\mathbb{E}_0[x(\pi_+)] = \alpha$, has the following form $x(s) = \mathbb{1}_{\ell_1(s) > k\ell_0(s)}$, for some $k > 0$. Take $\ell_0(s) = 1$ and $\ell_1(s) = s/c$, where $c = \int_0^1 sd\nu(s)$. Then the Neyman-Pearson lemma

gives us that the function x that maximizes $\frac{1}{c}\int_0^1 sx(s)d\nu(s)$, subj. to $\int_0^1 x(s)d\nu(s)=\rho$, is given by $x=\mathbb{1}_{s\geq kc}$, for some k. Since ρ, c are positive constants, the above optimization problem has the same optimal solution as the constrained-bitrate objective, so a solution to the latter is indeed a threshold policy. The threshold $\tau^*=kc$ computed by the constraint (20), which implies that τ^* is the solution of the equation $\int_{\tau^*}^1 d\nu(s)=\rho$. Note that, by our hypothesis $\mathbf{P}(\pi_+\geq\tau)$ is continuous and strictly monotone in $[0,1]$, and the above equation has a unique solution for $\alpha\in[0,1)$.

We now prove that (15) converges to the aforementioned τ^*. Note that $\mathbb{E}[\mathbb{1}^2_{\pi_+\geq\tau}]\leq 1$ and let $F(\tau)=\mathbf{P}(\pi_+\geq\tau)$, which is Lipschitz continuous. Hence, by Theorem 2.3 in Chapter 5 of [17], if $\gamma(k)=\frac{1}{k}$ the limit points (15) are a subset of the limit points of the ODE $\dot{\tau}=F(\tau)-\rho$. Consider the candidate Lyapunov function $L(\tau)=(\tau-\tau^*)^2$. Then $\frac{dL}{dt}=2(\tau-\tau^*)(F(\tau)-\rho)\leq 0$ by the definition of τ^* and the monotonicity of F; moreover, by the strict monotonicity of F, the above is an equality iff $\tau=\tau^*$, hence $\tau\to\tau^*$. □

E. PROOF OF THEOREM 5

We first show that there exists a threshold strategy that is optimal. Let again ν be the probability measure of π_+. Then (14) is equivalent to maximizing $\int_0^1 x(s)d\nu(s)$ subject to

$$\int_0^1 sx(s)d\nu(s)-\rho\int_0^1 x(s)d\nu(s)\geq 0 \tag{21}$$

Suppose that the problem is feasible, and let $x^*(s)$ be an optimal recommendation strategy. Let $opt=\int_0^1 x^*(s)d\nu(s)$. By Theorem 5 there exists a threshold strategy x s.t.

$$\int_0^1 x(s)d\nu(s)=opt$$

and

$$\int_0^1 sx(s)d\nu(s)\geq\int_0^1 sx^*(s)d\nu(s)\geq\rho\cdot opt=\rho\int_0^1 x(s)d\nu(s).$$

Hence, x is also an optimal strategy. The threshold can again be computed through (21) as the solution of the following equation$G(\tau)=\int_\tau^1(s-\rho)d\nu(s)=0$. Observe (*e.g.*, by differentiating), that under out assumptions on the c.d.f. of π_+, $G(\tau)$ is Lipschitz, strictly increasing for $\tau\leq\rho$ and decreasing for $\tau>\rho$. Given that it is zero for $\tau=1$, we deduce that $\tau_*\in[0,\rho)$ and that $G(\tau)<0$ for $\tau<\tau^*$ and $G(\tau)>0$ for $\tau\geq\tau^*$. Moreover, we also deduce that the necessary and sufficient condition for the existence of τ^* and, hence, the feasibility of (14), is that $G(0)\leq 0$. The same arguments as in the proof of Theorem 4 can be used to show that the limit points of (16) are, with probability one, a subset of the limit points of the ODE $\dot{\tau}=G(\tau)$. Using our above observations on G, it is easy to show that $(\tau-\tau^*)^2$ is a Lyapunov function of the above ODE, and that τ thus $\tau\to\tau^*$. □

On the Stability and Optimality of Universal Swarms

Xia Zhou
Dept. of Computer Science
UC Santa Barbara, CA, USA
xiazhou@cs.ucsb.edu

Stratis Ioannidis
Technicolor
Palo Alto, CA, USA
stratis.ioannidis@technicolor.com

Laurent Massoulié
Technicolor
Paris, France
laurent.massoulie@technicolor.com

ABSTRACT

Recent work on BitTorrent swarms has demonstrated that a bandwidth bottleneck at the seed can lead to the under-utilization of the aggregate swarm capacity. Bandwidth underutilization also occurs naturally in mobile peer-to-peer swarms, as a mobile peer may not always be within the range of peers storing the content it desires. We argue in this paper that, in both cases, idle bandwidth can be exploited to allow content sharing across multiple swarms, thereby forming a *universal swarm* system. We propose a model for universal swarms that applies to a variety of peer-to-peer environments, both mobile and online. Through a fluid limit analysis, we demonstrate that universal swarms have significantly improved stability properties compared to individually autonomous swarms. In addition, by studying a swarm's stationary behavior, we identify content replication ratios across different swarms that minimize the average sojourn time in the system. We then propose a content exchange scheme between peers that leads to these optimal replication ratios, and study its convergence numerically.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design–*distributed networks, store and forward networks*; C.4 [**Performance of Systems**]: Performance Attributes

General Terms

Theory, Algorithms

Keywords

Universal swarms, content distribution, peer-to-peer networks

1. INTRODUCTION

Peer-to-peer systems have been tremendously successful in enabling sharing of large files in a massive scale. This success has motivated several approaches of modeling BitTorrent swarms [6, 11, 12, 13]. Such models have illuminated important aspects of swarm behavior, including determining conditions for *swarm stability* and minimizing the system's *average sojourn time*. The study of swarm stability amounts to identifying conditions under which the swarm population remains finite as time progresses, while the sojourn time captures the time required until peers retrieve the file they request and leave the swarm.

Our aim in this work is to provide answers for similar questions in the context of *universal swarms* [15]. Rather than considering swarms as individual autonomous systems, we study scenarios in which peers from different swarms are permitted to exchange file pieces (chunks) with each other. Such inter-swarm exchanges make sense when bottlenecks in a single swarm lead to bandwidth under-utilization.

One application in which such bandwidth bottlenecks naturally arise is the peer-to-peer distribution of content over mobile opportunistic networks. Mobile peers wishing to retrieve a file can do so by downloading chunks from other peers they encounter opportunistically. These mobile content distribution systems have received considerable attention recently [1, 2, 4, 7, 8, 9, 14], as they alleviate the load on the wireless infrastructure by harnessing the bandwidth available during local interactions among mobile peers.

Bottlenecks in such peer-to-peer systems are a result of the opportunistic nature of the communication between peers: two peers meeting may not necessarily belong to the same swarm and may not be interested in the same content. Nevertheless, during encounters with peers from other swarms, a peer may use its idle bandwidth to obtain pieces of files in other swarms. Such exchanges can aid the propagation of under-replicated pieces that are otherwise hard to locate. If designed properly, such inter-swarm exchanges have the potential to improve the overall performance in terms of sojourn times and system stability.

Bottlenecks can also lead to bandwidth underutilization in online swarms. An example can be found in the recent work of Hajek and Zhu [6]. The authors considered the stability of a single BitTorrent swarm comprising a single seed and a steady stream of arriving peers (leechers). The peers share pieces they have retrieved while they are in the system but immediately depart once they download all pieces of a file. Hajek and Zhu observed that if the arrival rate of peers exceeds the upload capacity of the seed, the system becomes unstable in a very specific way: almost all peers arriving in the swarm very quickly obtain every missing piece *except for one*. The seed is unable to serve these peers with the

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
SIGMETRICS'11, June 7–11, 2011, San Jose, California, USA.
Copyright 2011 ACM 978-1-4503-0262-3/11/06 ...$10.00.

missing piece fast enough and, as a result, the size of this set of peers—termed the "one-club" [6]—grows to infinity.

When this so-called "missing piece syndrome" occurs, peers waiting for the missing piece are effectively idle, and their available upload bandwidth is essentially under-utilized. In this work we argue that, provided that peers have excess storage, this idle bandwidth capacity can be exploited in the presence of other swarms to store and to exchange pieces of other files. Such inter-swarm exchanges have the potential of improving the overall stability of the universal swarm system, as the peers in the "one-club" may be able to retrieve their missing piece from collaborating peers in other swarms. Most importantly, such transactions can be restricted to take place only when the intra-swarm bottleneck has rendered the peers idle, so inter-swarm exchanges do not hinder the delivery of the file in any way.

Our contributions can be summarized as follows:

- We propose a novel mathematical model for inter-swarm data exchange. Our model is simple but versatile enough to capture several different peer-to-peer file-sharing environments, both mobile and online.
- Using the above model, we analyze the stability of a universal swarm in which peers can retrieve items they miss from other swarms, but otherwise keep their caches static.
- Studying the stationary points of the data exchange process, we characterize the optimal replication ratios of pieces across swarms that minimize the system's average sojourn times.
- We propose BARON, a scheme for guiding data exchanges to yield optimal replication ratios, and study its convergence to these ratios numerically.

To the best of our knowledge, our work is the first systematic study of file sharing in a universal swarm system. Our results suggest that universal swarms can indeed achieve considerable performance improvements over independent autonomous swarm systems.

In particular, we establish the following surprising result: in a universal swarm where inter-swarm piece exchanges take place, *only one* swarm can become unstable. This is an interesting finding, especially when viewed in the context of the work of Hajek and Zhu [6]. An intuitive explanation of this phenomenon is this: a swarm growing to infinity attains an ever-growing capacity, which can be used to serve the missing pieces of other swarms. This service suppresses the growth of other swarms and, as a result, no two "one-clubs" can exist simultaneously.

Furthermore, our proposed scheme for guiding content exchanges can be used to enlarge the stability region for a universal swarm. Our design raises interesting open questions, such as the construction of schemes that work, *e.g.*, in fully distributed or non-cooperative environments. Though our model is simple, and our analysis is a first attempt at analyzing universal swarm behavior, we believe that these results are very promising. They indicate that universal swarms have very appealing stability properties, and certainly merit further investigation.

The remainder of this paper is structured as follows. We begin with an overview of related work in Section 2 and introduce our mathematical model for universal swarms in Section 3. We present our main results on convergence, stability, and optimality in Section 4, and provide their proofs in Section 5. We further propose BARON, a scheme to guide the system to the optimal stationary state, and evaluate it numerically in Section 6. We conclude by presenting future directions in Section 7.

2. RELATED WORK

Qiu and Srikant [13] were the first to introduce a fluid model for BitTorrent. Using an ordinary differential equation (ODE) to capture peer dynamics, they study sojourn times at the fixed points of this ODE, as well as the impact of incentive schemes. Our work is most similar to Massoulié and Vojnovic [12] who, contrary to [13], study directly the dynamics of the stochastic system determined by piece exchanges between peers. As in the present work, no seed exists: peers arrive already storing several pieces of a file, and exchange pieces by contacting uniformly at random other peers in the swarm. The authors identify conditions for system stability and determine the sojourn time at equilibrium. Massoulié and Twigg [11] study similar issues in the context of P2P streaming, which differs by requiring that pieces are retrieved in a certain order. Our work generalizes [12] by allowing piece exchanges across swarms and, as [11, 12], studies a fluid limit of the resulting system.

Recent work by Hajek and Zhu [6] identifies the "missing piece syndrome" described in the introduction. Their model differs from [12] in assuming that a single, non-transient seed exists while all other peers arrive with no pieces. The bandwidth bottleneck due to the missing piece syndrome partially motivates our study of universal swarms. We will further elaborate on the relationship of our work to [6] in our concluding remarks.

In the context of mobile peer-to-peer systems, BARON, our scheme for guiding content exchanges, is related to a series of recent papers on optimizing mobile content delivery. In general, the goal of these works is to ensure fast delivery of content to mobile users through opportunistic exchanges while using as few bandwidth and storage resources as possible. Schemes studied involve selecting which content to transmit during contacts [1, 2, 7], which information to cache in local memory [9, 14], or where to inject new content [8]. Our work differs both in considering an open system, where mobile users depart once obtaining the content they want, as well as in capturing several different (*e.g.*, contact or interference constrained) communication scenarios.

3. SYSTEM MODEL

3.1 Overview

The system that we model is a universal swarm, consisting of several peers wishing to retrieve different content items. Peers share content they store with other peers while they are in the system; once a peer retrieves the content item that it is interested in, it exits the system.

Our model describes both mobile and online peer-to-peer swarms. In both cases, we assume that downloads take place as in [12]: each peer is idle for an exponentially distributed time and then contacts a peer selected uniformly at random from the peers present in the system. During such contacts, peers may choose to exchange content items they store and all transfers are instantaneous.

In the wireless mobile case, the above contact process aims to model mobility. That is, two mobile peers come into contact whenever they are within each other's transmission range. In an online peer-to-peer network, the contact process captures *random sampling*. In particular, peers sample the system population uniformly at random to find the items they want. No "universal tracker" exists, and peers do not know which peers in other swarms may be storing the items they request, hence the need for random sampling.

We make the following assumptions. First, every peer entering the system is only interested in downloading *a single content item*; once retrieving this single item, the peer immediately exits the system. Second, whenever a peer contacts another peer that stores its requested item, it is able to retrieve the *entire item* immediately. Third, as in [12], peers arrive with non-empty caches, and begin to share immediately when they enter the system.

The above assumptions are obviously simplifications of real-life peer-to-peer system behavior. On one hand, if our items correspond to the granularity of files, a peer would not be able to download an entire file within one downloading session with another peer. If, on the other hand, items correspond to the granularity of chunks, peers would need to retrieve several items before exiting the system. Nevertheless, in spite of these simplifications, our analysis provides interesting insights in universal swarm behavior, especially in light of the "missing piece syndrome" observed by Hajek and Zhu [6]. We will revisit this issue in Section 7.

3.2 Peer Swarms and Classes

We consider a *universal swarm* in which content items belonging to a set $\mathbb{K}$, where $|\mathbb{K}| = K$, are shared among transient peers. Each peer arrives with a request $i \in \mathbb{K}$ and a cache of items $f \subset \mathbb{K}$, where $C = |f|$ is the capacity of the cache. We denote by $\mathbb{F} = \{f \subset \mathbb{K} : |f| = C\}$ the set of all possible contents of a peer's cache.

A peer *swarm* consists of all peers interested in retrieving the same item $i \in \mathbb{K}$. We partition the peers in the system into *classes* according to both (a) the item they request and (b) the content in their cache. That is, each pair $(i, f) \in \mathbb{C} = \mathbb{K} \times \mathbb{F}$ defines a distinct peer class.

We denote by $N_{i,f}(t)$ the number of peers requesting i and storing f at time t. We use the notation

$$\mathbf{N}(t) = [N_{i,f}(t)]_{(i,f)\in\mathbb{C}}$$

for the vector representing the system state, *i.e.*, the number of peers in each class. We also denote by

$$N(t) = \sum_{(i,f)\in\mathbb{C}} N_{i,f}(t) = \mathbf{1}^T \cdot \mathbf{N}(t)$$

the total number of peers in the system at time t.

3.3 Peer Arrival Process

Peers requesting item $i \in \mathbb{K}$ and storing $f \in \mathbb{F}$ arrive according to a Poisson process with rate $\lambda_{i,f}$, and that arrivals across different classes are independent. By definition, $\lambda_{i,f} = 0$ if $i \in f$. We denote by $\lambda = \sum_{(i,f)\in\mathbb{C}} \lambda_{i,f}$ the aggregate arrival rate of peers in the system. We also define

$$\lambda_{i,\cdot} = \sum_{f\in\mathbb{F}} \lambda_{i,f}, \quad \lambda_{\cdot,i} = \sum_{\substack{j\in\mathbb{K} \\ f: i\in f}} \lambda_{j,f}, \quad i \in \mathbb{K} \tag{1}$$

as the aggregate arrival rates of peers requesting and caching item i, respectively.

For some of our results, we require that λ tends to infinity; when doing so, we assume that the arrival rate corresponding to each class increases proportionally to λ, *i.e.*, the normalized arrival rate

$$\hat{\lambda}_{i,f} = \lambda_{i,f}/\lambda \tag{2}$$

is constant w.r.t. λ.

3.4 Contact Process

Opportunities to exchange items among peers occur when two peers come into contact. As mentioned in Section 3.1, contacts model different processes in a mobile network and an online peer-to-peer network. In the mobile case, a contact indicates that two mobile peers are within each other's transmission range. In the online case, contacts capture random peer sampling in the universal swarm.

Formally, if $N(t)$ is the total number of peers in the system at time t, then a given peer a present in the system contacts other peers according to a non-homogeneous Poisson process with rate

$$\mu \cdot (N(t))^{1-\beta}, \quad \beta \in [0, 2].$$

The peer with which peer a comes into contact is selected uniformly at random from the $N(t)$ peers currently present in the system. Moreover, the above contact processes are independent across peers.

The parameter β is used to capture different communication scenarios that may arise in a mobile or online network. We classify these below into *contact-constrained*, *constant-bandwidth*, and *interference-constrained* scenarios.

Contact-constrained communication. When $0 \leq \beta < 1$, the contact rate of a peer is growing proportionally to the total peer population. This would be the case in a sparse, opportunistic or DTN-like wireless mobile network, where peers are within each other's transmission range very infrequently. In such cases, the bottleneck in data exchanges is determined by how often peers meet. Adding more peers in such an environment can increase the opportunities for contacts between peers. This is reflected in the increase of a peer's contact rate as the population size grows.

Constant-bandwidth communication. When $\beta = 1$ the contact rate of a peer does not depend on the population size. This reflects constant-bandwidth scenarios, where the system population has no effect on the bandwidth capabilities of a peer, and is thus a natural model of an online peer-to-peer network.

Interference-constrained communication. When $\beta \in (1, 2]$, the contact rate of a peer decreases as the total peer population grows. This captures a dense wireless network in which peers share a wireless medium to communicate. As the number of peers increases, the wireless interference can become severe, degrading the network throughput. This is reflected in our model by a decrease in successful contact events and, thus, in a peer's contact rate.

If $\beta > 2$, the aggregate contact rate over *all* peers in the system decreases as the total peer population grows. Assuming constant arrival rates, such a system will be trivially unstable; as such, we do not consider this case.

For simplicity of notation, we allow self-contacts. Contacts are not symmetric; when Alice contacts Bob, Bob does

not contact Alice, and vice versa. This, however, is not restrictive: symmetric contacts can be easily represented by appropriately defining symmetric interactions between two peers (*c.f.* the conversion probabilities appearing below).

Under the above assumptions, when the system state is $\mathbf{N}$, the aggregate rate with which users from class A contact users from class A' is

$$\mu_{A,A'}(\mathbf{N}) = \mu N_A N_{A'}/N^{\beta}, \quad A,A' \in \mathbb{C}. \tag{3}$$

We call $\mu_{A,A'}$ as the *inter-contact* rate between A and A'.

3.5 Content Exchanges During Contacts

When a peer in class $A \in \mathbb{C} = \mathbb{K} \times \mathbb{F}$ contacts another peer in class $B \in \mathbb{C}$, the two peers may exchange items stored in their respective caches. Such exchanges can lead to, *e.g.*, (a) the departure of a peer, because it obtains the item it requests, or (b) the change of its cache contents, as new items replace old items in the peer's cache.

In particular, given that the current state of the system is $\mathbf{N}(t)$, when a peer of class $A \in \mathbb{C}$ contacts another peer in $A' \in \mathbb{C}$, the peer in A is converted to a peer in $B \in \mathbb{C} \cup \{\varnothing\}$ and the peer in A' is converted to a peer in $B' \in \mathbb{C} \cup \{\varnothing\}$ with the following probability

$$\Delta_{A,A' \to B,B'}(\mathbf{N}(t)),$$

independently of any other event in the history of the process $\mathbf{N}(t)$ so far. In the above, we use the notation $\varnothing$ to indicate that a peer exits the system. We call the above Δ functions the *conversion probabilities* of the system. Conversion probabilities depend on the global state $\mathbf{N}(t)$ at the time of contact. We make the following technical assumption:

ASSUMPTION 1. *For every $s > 0$, and for every $A, A' \in \mathbb{C}$ and $B, B' \in \mathbb{C} \cup \{\varnothing\}$, $\Delta_{A,A' \to B,B'}(\mathbf{N}) = \Delta_{A,A' \to B,B'}(s\mathbf{N})$.*

In other words, the conversion probabilities are *invariant to rescaling*: if all peer classes are increased by the same factor, the conversion probabilities will remain unaltered. Let

$$\zeta^{A}_{A',A' \hookrightarrow B',B''} = \mathbb{1}_{B'=A} + \mathbb{1}_{B''=A} - \mathbb{1}_{A'=A} - \mathbb{1}_{A''=A}, \tag{4}$$

be an indicator function capturing how a conversion $A', A'' \to B', B''$ affects the population of class A. For example, (4) states that conversions can increase N_A by at most 2, when both classes A', A'' are converted to A.

We require that conversions follow what we call the "*grab-and-go*" principle: whenever two peers come into contact, if the first stores the second peer's requested item, the latter will retrieve it and exit the system. In other words, content exchanges that lead to departures are always enforced. Formally, the "grab-and-go" principle can be defined as:

$$\begin{aligned} &\textstyle\sum_{B' \in \mathbb{C} \cup \{\varnothing\}} \Delta_{(i,f),(i',f') \to \varnothing,B'}(\mathbf{N}) = 1 \text{ if } i \in f', \text{ and} \\ &\textstyle\sum_{B \in \mathbb{C} \cup \{\varnothing\}} \Delta_{(i,f),(i',f') \to B,\varnothing}(\mathbf{N}) = 1 \text{ if } i' \in f. \end{aligned} \tag{5}$$

The simplest interaction that satisfies the "grab-and-go" principle is the *static-cache* policy: peers never alter the contents of their caches for as long as they are in the system, other than as dictated by the "grab-and-go" principle. Formally, the static-cache policy can be stated as:

$$\Delta_{(i,f),(i',f') \to B,B'}(\mathbf{N}) = 1, \text{ where}$$
$$B = \begin{cases} \varnothing, & \text{if } i \in f' \\ (i,f) & \text{o.w.,} \end{cases} \quad \text{and } B' = \begin{cases} \varnothing, & \text{if } i' \in f \\ (i',f'), & \text{o.w.} \end{cases} \tag{6}$$

Table 1: Summary of Notation

$\mathbb{K}$	Set of items
C	Cache capacity
$\mathbb{F}$	Set of possible cache contents
(i,f)	Class of users requesting $i \in \mathbb{K}$ and storing $f \in \mathbb{F}$
$\mathbb{C}$	The set of classes $\mathbb{K} \times \mathbb{F}$
$N_{i,f}(t)$	The number of users in class (i,f)
$\mathbf{N}(t)$	The system state
$N(t)$	Number of peers in the system
$\lambda_{i,f}$	Arrival rate of peers in class (i,f)
λ	Aggregate arrival rate
$\hat{\lambda}_{i,f}$	Normalized arrival rate for class (i,f)
β	Decay exponent of the contact rate
μ	Contact rate constant
$\Delta_{A,A' \to B,B'}$	Conversion probabilities
$\zeta^{A}_{A',A' \hookrightarrow B',B''}$	Effect of conversion $A', A'' \to B', B''$ on class A
$\delta_{A',A' \hookrightarrow B',B''}$	Limit points of the conversion probabilities
$n_{i,f}(t)$	Fluid trajectory of class (i,f)
$\mathbf{n}(t)$	Fluid trajectories of the system state
$n(t)$	Sum of fluid trajectories
$n_{i,\cdot}$, $n_{\cdot,i}$	Demand and supply for $i \in \mathbb{K}$
$n^*_{i,\cdot}$, $n^*_{\cdot,i}$	Optimal demand and supply for $i \in \mathbb{K}$

Of course, there are many other conversion probabilities that satisfy the "grab-and-go" principle. In particular, (5) tells us nothing about how peers interact with each other when neither of them stores the other's requested item. Rather than leaving caches static, as in (6), such events can be exploited to change the number of replicas in the system, *e.g.*, to reach some global optimization objective, like increasing system stability or reducing the system sojourn times. We do precisely this in Section 6: we design interactions between peers (*i.e.*, determine the conversion probabilities) in a way that such a global optimization objective is met.

4. MAIN RESULTS

Having described our system model, we now present our main results. To begin with, we establish that, for arbitrary conversion probabilities the dynamics of our system can arbitrarily well approximated by a fluid limit (Section 4.1). We then describe the stability region of the static-cache policy (Section 4.2). Finally, we establish conditions under which interactions that follow the "grab-and-go" principle minimize sojourn times (Section 4.3).

4.1 Convergence to a Fluid Limit

Our first main result states that the evolution of the universal swarm through time can be approximated arbitrarily well by the solution of an ordinary differential equation (ODE). This result is very general: we prove convergence to such a fluid limit for *all* $\beta \in [0,2)$ and *all* conversion probabilities satisfying Assumption 1.

We begin by formally defining the notion of a fluid limit of the universal swarm. We say that the vector

$$\mathbf{n}(t) = [n_A(t)]_{A \in \mathbb{C}}$$

is a *fluid trajectory* of the system if, for every class $A \in \mathbb{C}$, the functions $n_A : \mathbb{R}_+ \to \mathbb{R}_+$ satisfy the following ODEs:

$$\dot{n}_A(t) = \hat{\lambda}_A + \sum_{\substack{A',A'' \in \mathbb{C} \\ B',B'' \in \mathbb{C} \cup \{\varnothing\}}} \zeta^{A}_{A',A' \hookrightarrow B',B''} \mu_{A',A''}(\mathbf{n}(t)) \delta_{A',A' \hookrightarrow B',B''}(\mathbf{n}(t)), \tag{7}$$

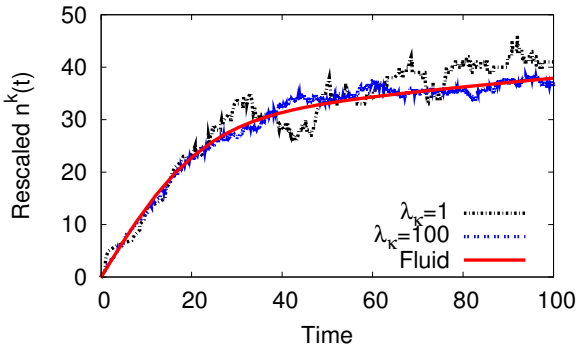


Figure 1: Comparing the rescaled trajectory of the original system to the fluid trajectory using the static-cache policy. We simulated a system where $\mathbb{K} = \{1, 2, 3\}, C = 1$, and $\beta = 0$, for $\lambda_k = 1$ and $\lambda_k = 100$ respectively. The rescaled trajectory clearly converges to the fluid trajectory as λ_k increases.

where $\hat{\lambda}_A$, $\mu_{A',A''}$, and $\zeta^A_{A',A''\hookrightarrow B',B''}$ are given by (2), (3), and (4) respectively, and $\delta_{A',A''\hookrightarrow B',B''} : \mathbb{R}_+^{|\mathbb{C}|} \to [0, 1]$ are any functions that satisfy the following property:

$$\delta_{\cdot\to\cdot}(\mathbf{n}) \in [\liminf_{\mathbf{n}'\to\mathbf{n}} \Delta_{\cdot\to\cdot}(\mathbf{n}'), \limsup_{\mathbf{n}'\to\mathbf{n}} \Delta_{\cdot\to\cdot}(\mathbf{n}')].$$

The δ functions are unique and coincide with the conversion probabilities if and only if the latter are continuous. In this case, the ODEs (7) also have a unique solution. For any $\mathbf{n}^* \in \mathbb{R}_+^{|\mathbb{C}|}$, let $S(\mathbf{n}^*)$ be the set of all fluid trajectories of the system with initial condition $\mathbf{n}(0) = \mathbf{n}^*$. The following theorem establishes two facts. First, $S(\mathbf{n}^*)$ is non-empty—*i.e.*, fluid trajectories exist for all initial conditions. Second, under appropriate rescaling, a trajectory of the universal swarm $\{\mathbf{N}(t), t \in \mathbb{R}_+\}$, can be arbitrarily well approximated by a fluid trajectory.

THEOREM 1. *Let $\alpha \equiv 1/(2-\beta)$. Consider a sequence of positive numbers $\{\lambda_k\}_{k\in\mathbb{N}}$ such that $\lim_{k\to\infty} \lambda_k = +\infty$, and a sequence of initial conditions $\mathbf{N}^k(0) = [N^k_A(0)]_{A\in\mathbb{C}}$ s.t. the limit $\lim_{k\to\infty} {\lambda_k}^{-\alpha}\mathbf{N}^k(0) = \mathbf{n}^*$ exists. Consider the rescaled process*

$$\mathbf{n}^k(t) = \lambda_k^{-\alpha}\mathbf{N}^k(\lambda_k^{\alpha-1}t), \quad t \in \mathbb{R}_+. \tag{8}$$

Then for all $T > 0$ and all $\epsilon > 0$,

$$\lim_{k\to\infty} \mathbf{P}\Big(\inf_{\mathbf{n}\in S(\mathbf{n}^*)} \sup_{t\in[0,T]} |\mathbf{n}^k(t) - \mathbf{n}(t)| \geq \epsilon \Big) = 0,$$

i.e., $\mathbf{n}^k$ *converges to a fluid trajectory in probability.*

The above convergence in probability is illustrated in Figure 1: the rescaled process $n^k(t) = \mathbf{1}^T\mathbf{n}^k(t)$ converges to the fluid trajectory as we increase the scaling factor λ_k from 1 to 100. The proof of this theorem can be found in our technical report [16] and follows closely the argument in [10], so we omit it for reasons of brevity.

Given a fluid trajectory $\{\mathbf{n}(t)\}_{t\in\mathbb{R}_+}$, we denote by

$$n_{i,\cdot}(t) = \textstyle\sum_f n_{i,f}(t), \;\; n_{\cdot,i}(t) = \sum_{j,f:i\in f} n_{j,f}(t), \quad i \in \mathbb{K} \tag{9}$$

the (rescaled) population of peers requesting and caching item i, respectively. We call $n_{i,\cdot}$, $n_{\cdot,i}$ the *demand* and the *supply* of item i, respectively.

4.2 Stability of the Static-Cache Policy

Armed with the above system characterization through its fluid trajectories, we turn our attention to the issue of system stability. Intuitively, we wish to understand what is the *stability region* of our system: what conditions should the arrival rates $\lambda_{i,f}$, $(i, f) \in \mathbb{C}$, satisfy, so that the total number of peers in the system remains bounded?

Surprisingly, a universal swarm evolving under the static-cache policy, arguably the simplest policy satisfying the "grab-and-go" principle, has a very wide stability region. We demonstrate this below by studying (a) the stability of the fluid trajectories of the static-cache policy and (b) the ergodicity of the original stochastic system.

We begin by stating our main result regarding fluid trajectories. We say that the system of ODEs (7) is stable if the fluid trajectories $\{\mathbf{n}(t)\}_{t\geq 0}$ remain bounded for all $t \geq 0$ irrespectively of the initial conditions $\mathbf{n}(0)$. In other words, irrespectively of how many peers are originally in the system, the population never blows up to infinity. Denote by

$$\lambda_{i,j} = \textstyle\sum_{f:j\in f} \lambda_{i,f}, \qquad i, j \in \mathbb{K}, \tag{10}$$

the aggregate arrival rate of peers requesting item i and caching item j.

A sufficient condition for stability of the static-cache policy is stated in the following theorem, whose proof can be found in Section 5.1.

THEOREM 2. *Assume that all rates $\lambda_{i,j}$ are positive and*

$$\textstyle\sum_{j:j\neq i} \lambda_{j,i}/\lambda_{i,j} > 1, \quad \forall i \in \mathbb{K}. \tag{11}$$

Then, for all $\beta \in [0, 2)$, the system of ODEs (7) under the static-cache policy is stable.

There are several important conclusions to be drawn from Theorem 2. To begin with, the stability region remains the same for all values of $\beta \in [0, 2)$: this is quite surprising, as it implies that (7) applies to all the different contact regimes we reviewed in Section 3.4 (contact-constrained, constant-bandwidth, and interference-constrained communications).

In addition, recall that $\lambda_{i,\cdot}$ and $\lambda_{\cdot,i}$, given by (1), are the aggregate arrival rates of peers requesting and caching item i, respectively. Inequality (11) implies that the system can be stable even if $\lambda_{i,\cdot} > \lambda_{\cdot,i}$ for some i, *i.e.*, *peers requesting i arrive at a higher rate than peers storing i.* In particular as long as for every i there exists an item j such that $\lambda_{j,i} > \lambda_{i,j}$, then (11) are satisfied, and the system is stable. Intuitively, if such a j exists, the size of its swarm will grow large enough to provide the upload capacity necessary to serve peers requesting item i.

The above theorem has a direct equivalent w.r.t. the stochastic process $\{\mathbf{N}(t)\}_{t\in\mathbb{R}_+}$:

THEOREM 3. *Assume that all rates $\lambda_{i,j}$ are positive and that (11) holds. Then, for all $\beta \in [0, 2)$, the stochastic process $\{\mathbf{N}(t)\}_{t\in\mathbb{R}_+}$ under the static-cache policy is ergodic.*

We provide a proof of this theorem in Section 5.2.

A very interesting aspect of static-cache stability is in the manner in which the universal swarm becomes unstable when (11) is violated. In particular, recall by (9) that $n_{i,\cdot}(t)$ is the demand for item i, *i.e.*, the size of the swarm of peers requesting item i (in the fluid limit). The following theorem then holds:

THEOREM 4. *Assume that all rates $\lambda_{i,j}$ are positive, and $\beta \in [0,2)$. Then there exists at most one item $i \in \mathbb{K}$ for which*

$$\sum_{j:j\neq i} \lambda_{j,i}/\lambda_{i,j} < 1. \quad (12)$$

Moreover, if such an item i exists, there exist initial conditions $\mathbf{n}(0)$ such that

$$\lim_{t\to\infty} n_{i,\cdot}(t) = \infty, \text{ and } \limsup_{t\to\infty} n_{j,\cdot}(t) n_{i,\cdot}^{1-\beta}(t) < \infty, \ \forall j \neq i.$$

In other words, for $\beta \in [0,1]$, *only one swarm can become unstable*. There can be only one item that satisfies (12), and although the swarm of peers requesting this item grows to infinity, the product $n_{j,\cdot} n_{i,\cdot}^{1-\beta}$ remains bounded. As a result, for $\beta \in [0,1]$, no other swarm than the one satisfying (12) can become unstable. This property is very appealing, as it suggests that even if the arrival rates are outside the stability region, the stability of all but one swarm remains unaffected. Note that, when $\beta > 1$, *i.e.*, in the interference-constrained case, the product $n_{j,\cdot} n_{i,\cdot}^{1-\beta}$ is also bounded; however, this does not imply that other swarms do not grow. Nevertheless, these swarms grow at a slower rate than $n_{i,\cdot}$.

This stability property arises precisely because the universal swarm utilizes available bandwidth for inter-swarm communication. Intuitively, a swarm that becomes unstable has unbounded uploading capacity. As a result, as long as all arrival rates are positive, a fraction of this unbounded capacity can be used serve to other swarms at a very high rate; when $\beta \in [0,1]$, this rate is in fact high enough to suppress the growth of any other swarm.

4.3 Optimality Under the "Grab-and-Go" Principle

Despite the interesting stability properties of the static-cache policy, it is still tempting to see whether we can design more sophisticated policies that achieve a wider stability region. Preferably, given that the system is stable we would like a design that minimizes average sojourn time. In this section, we characterize the minimum sojourn time achievable by any system satisfying the "grab-and-go" principle. We will use this to propose a content exchange policy that minimizes the average sojourn time in Section 6.

By Little's Theorem, minimizing the average sojourn time is equivalent to minimizing $N(t)$, the total number of peers in the system. We approach this problem by studying the stationary points of the fluid trajectories. This is a heuristic: by studying the stationary points of (7), we implicitly assume that the Markov process $\{\mathbf{N}(t)\}_{t\in\mathbb{R}_+}$ exhibits some form of concentration around these stationary points. Nevertheless, we believe that there is important intuition to be gained through our approach; we demonstrate that this is indeed the case through our numerical study of a sojourn minimizing system in Section 6.2.

Recall by (9) that $n_{i,\cdot}$ and $n_{\cdot,i}$ are the demand and supply of item i, respectively. The "grab-and-go" principle (5) implies that the fluid trajectories given by (7) satisfy the following set of equations:

$$\dot{n}_{i,\cdot}(t) = \hat{\lambda}_{i,\cdot} - 2\mu \cdot (n(t))^{-\beta} n_{i,\cdot}(t) n_{\cdot,i}(t). \quad i \in \mathbb{K} \quad (13)$$

The above equations state that the swarm of peers requesting i grows with new peer arrivals and decreases at encounters between peers in the swarm and peers that cache i. However, they do not specify what type of conversions take place during other types of encounters between peers. Nevertheless, a stationary point $\mathbf{n} \in \mathbb{R}^{|\mathbb{C}|}$ of (13) must satisfy:

$$n_{i,\cdot} n_{\cdot,i} - \hat{\lambda}_{i,\cdot} (2\mu)^{-1} (n)^{\beta} = 0, \forall i \in \mathbb{K}.$$

Since peers cache at most C items, the number of cached items must be no more than the total cache capacity, *i.e.*,

$$\sum_{i\in\mathbb{K}} n_{\cdot,i} \leq Cn = C \sum_{i\in\mathbb{K}} n_{i,\cdot}$$

We now pose the following problem: among all stationary points of content exchange policies that satisfy the "grab-and-go" principle, which stationary point has the minimum aggregate peer population? More formally, we wish to solve:

$$\begin{aligned}
\text{Minimize} \quad & \textstyle\sum_{i\in\mathbb{K}} n_{i,\cdot} && (14a)\\
\text{subj. to:} \quad & n_{i,\cdot} n_{\cdot,i} - \tfrac{\hat{\lambda}_{i,\cdot}}{2\mu} \big(\textstyle\sum_{i\in\mathbb{K}} n_{i,\cdot}\big)^{\beta} = 0, \quad \forall i \in \mathbb{K} && (14b)\\
& \textstyle\sum_{i\in\mathbb{K}} n_{\cdot,i} \leq C \sum_{i\in\mathbb{K}} n_{i,\cdot} && (14c)\\
& n_{i,\cdot} \geq 0, \ \forall i \in \mathbb{K}. && (14d)
\end{aligned}$$

When $\beta \in [0,1]$, the above problem is convex [3] and its solution is given by the following theorem:

THEOREM 5. *For $\beta \in [0,1]$ and $\rho_i = \hat{\lambda}_{i,\cdot}(2\mu C)^{-1}$ the unique optimal solution to (14) is*

$$n^*_{i,\cdot} = \sqrt{\rho_i} \big(\textstyle\sum_{j:j\in\mathbb{K}} \sqrt{\rho_j}\big)^{\beta/(2-\beta)} \quad (15a)$$

$$n^*_{\cdot,i} = C\sqrt{\rho_i} \big(\textstyle\sum_{j:j\in\mathbb{K}} \sqrt{\rho_j}\big)^{\beta/(2-\beta)}, \quad \forall i \in \mathbb{K}, \quad (15b)$$

The proof of Theorem 5 can be found in Section 5.4. Note that the theorem does not hold for $\beta \in (1,2]$, as (14) is not convex for these values of β. Moreover, (15) describes the optimal steady state demand and supply but not the size of each individual class. By (15), the optimal supply is proportional to the square-root of the aggregate arrival rates of peers requesting this item. This was also observed in the closed caching system described in [5]; our result can thus be seen as an extension of [5] for an open system with peer arrivals and departures. Finally, by (15)

$$n^*_{i,\cdot} = Cn^*_{\cdot,i} \quad (16)$$

i.e., the demand is C times the supply. In Section 6, we use this to propose a sojourn-minimizing item-exchange policy.

5. ANALYSIS

5.1 Proof of Theorem 2

Using (6), the ODE (7) for the fluid trajectories under the static cache policy assumes the following simple form.

$$\dot{n}_{i,f}(t) = \hat{\lambda}_{i,f} - 2\mu n(t)^{-\beta} n_{i,f}(t) n_{\cdot,i}(t), \quad i \in \mathbb{K}, f \in \mathbb{F}, \quad (17)$$

where $n_{\cdot,i}(t) = \sum_{j,f:i\in f} n_{j,f}(t)$. The above differential equation has an explicit solution in terms of $g_i(t) := 2\mu n_{\cdot,i}(t) n(t)^{-\beta}$, given by $n_{if}(t) = \big\{n_{if}(0) + \int_0^t \hat{\lambda}_{if} e^{\int_0^s g_i(u)du} ds\big\} e^{-\int_0^t g_i(u)du}$. Consider now the ratio $n_{if}(t)/n_{if'}(t)$ for two distinct indices f, f'. In the view of the previous formula, it reads

$$\frac{n_{if}(t)}{n_{if'}(t)} = \frac{n_{if}(0) + \int_0^t \hat{\lambda}_{if} \exp\big(\int_0^s g_i(u)du\big)\, ds}{n_{if'}(0) + \int_0^t \hat{\lambda}_{if'} \exp\big(\int_0^s g_i(u)du\big)\, ds}.$$

Since the function g_i is non-negative, the argument in the integrals is lower-bounded by a positive constant. As a result, it follows by L'Hospital's rule that $\frac{n_{if}(t)}{n_{if'}(t)} = \frac{\hat{\lambda}_{if}}{\hat{\lambda}_{if'}} + O(1/t)$.

This implies that for large t, the individual variables $n_{ij}(t)$ are related by proportionality constraints, and as a result we can focus on tracking a smaller set of variables. Namely, we introduce the variables $u_i(t) := \frac{n_{i\cdot}(t)}{\hat{\lambda}_{i\cdot}}$. Each individual variable $n_{if}(t)$ verifies $n_{if}(t) = \hat{\lambda}_{if}u_i(t) + O(1/t)$, then

$$\begin{aligned}\dot{u}_i(t) &= 1 - u_i(t)n_{\cdot i}(t)n(t)^{-\beta} \\ &= 1 - u_i(t)\big(\tfrac{\sum_{j\neq i}\hat{\lambda}_{ji}u_j(t)}{[\sum_j \hat{\lambda}_{j\cdot}u_j(t)]^\beta} + O(1/t)\big),\end{aligned}$$

where $\hat{\lambda}_{ij}$ as in (10) and $\hat{\lambda}_{i\cdot}$ as in (1). Hence, for large enough T the evolution of u_i within a finite interval $[T, T+t]$ can be arbitrarily well approximated by the ODE:

$$\dot{u}_i = 1 - u_i\textstyle\sum_{j\neq i}\hat{\lambda}_{ji}u_j\big[\sum_j \hat{\lambda}_{j\cdot}u_j\big]^{-\beta}. \tag{18}$$

We therefore focus on (18)—keeping in mind that our analysis below holds for large enough T. We will show that if (11) is satisfied for every $i \in \mathbb{K}$, then $\sup_i u_i(t)$ is bounded for all t. In particular, the following lemma holds:

LEMMA 1. *For $M>0$ large enough, there exist $\delta>0$ and $\epsilon>0$ s.t. if $\sup_i u_i(0)=M$, then $\sup u_i(M\delta)\leq M(1-\epsilon)$.*

PROOF. To show this, for a given M, fix a $\delta > 0$. If $\sup_i u_i(\delta M) < M(1-\delta)$, then the lemma obviously holds for $\epsilon = \delta$. Suppose thus that there exists an i such that $u_i(\delta M) \geq M(1-\delta)$. By (18), for $t \in [0, \delta M]$ we have $u_i(t) \leq u_i(0) + t \leq M + \delta M$ and $u_i(t) \geq u_i(\delta M) + t - \delta M \geq M(1-\delta) - \delta M$. Hence $u_i(t) \in [M(1-2\delta), M(1+\delta)]$. This in turn implies that, for $t \in [0, \delta M]$, $n(t) = \Theta(M)(1+O(\delta))$, where the constants involved depend on $\hat{\lambda}_i$ but not on t. As a result, for $j \neq i$, and $t \in [0, \delta M]$, we have

$$\dot{u}_j(t) = 1 - u_j\Theta(M^{1-\beta})(1+\epsilon_1(\delta)),$$

where $\epsilon_1(\delta) = 1 - \frac{1+O(\delta)}{(1+O(\delta))^\beta} = O(\delta)$. Thus for $t \in [0, \delta' M]$, where $\delta' < \delta$, we have

$$\begin{aligned}u_j(t) &= [u_j(0) + \int_0^t e^{s\Theta(M^{1-\beta})(1+\epsilon_1(\delta))ds}]e^{-t\Theta(M^{1-\beta})(1+\epsilon_1(\delta))} \\ &= u_j(0)e^{-t\Theta(M^{1-\beta})(1+\epsilon_1(\delta))} + \frac{1-e^{-t\Theta(M^{1-\beta})(1+\epsilon_1(\delta))}}{\Theta(M^{1-\beta})(1+\epsilon_1(\delta))}.\end{aligned}$$

Fix a $0 < \delta' < \delta$, then

$$\begin{aligned}u_j(\delta' M) &= O(Me^{-\Theta(\delta' M^{2-\beta})(1+\epsilon_1(\delta))}) + \frac{1-e^{-\Theta(\delta' M^{2-\beta})(1+\epsilon_1(\delta))}}{\Theta(M^{1-\beta})(1+\epsilon_1(\delta))} \\ &= \Theta(M^{-(1-\beta)})(1+\epsilon_2(M,\delta,\delta')),\end{aligned}$$

where $\epsilon_2 = O(\epsilon_1(\delta) + M^{2-\beta}e^{-\Theta(\delta' M^{2-\beta})(1+\epsilon_1(\delta))})$. From this and (18) we get that for $t \in [\delta' M, \delta M]$

$$\dot{u}_j(t) = 1 - u_j\hat{\lambda}_{i,j}\hat{\lambda}_{i,\cdot}^{-\beta}M^{1-\beta}(1+\epsilon_3(M,\delta,\delta')),$$

where $\epsilon_3 = O(\delta + O(M^{\beta-2}) + O(M^{\beta-2}\epsilon_2) = O(\delta) + O(M^{\beta-2}) + O(e^{-\Theta(\delta' M^{2-\beta})(1+\epsilon_1(\delta))})$. From this refined bound on the ODE, we can repeat the steps above to get that for $t \in [\delta'' M, \delta M]$, where $\delta' \leq \delta'' < \delta$, we have

$$u_j(t) = \hat{\lambda}_{i,\cdot}^{\beta}\hat{\lambda}_{i,j}^{-1}M^{\beta-1}(1+\epsilon_4(M,\delta,\delta',\delta'')),$$

where $\epsilon_4 = O(\epsilon_3) + O(M^{2-\beta}e^{-\Theta(\delta'' M^{2-\beta})})$. As a result, for $t \in [\delta'' M, \delta M]$,

$$\begin{aligned}\dot{u}_i(t) &= 1 - u_i\frac{\sum_{k\neq i}\hat{\lambda}_{ki}u_k}{n^\beta} \\ &= 1 - u_i(t)\frac{\sum_{k\neq i}\hat{\lambda}_{k,i}\frac{\hat{\lambda}_{i\cdot}^\beta}{\hat{\lambda}_{ik}}[M^{\beta-1}(1+\epsilon_4)]}{[\hat{\lambda}_{i\cdot}M(1+O(M^{2-\beta})(1+\epsilon_4)]^\beta} \\ &= 1 - u_i\sum_{k\neq i}\frac{\hat{\lambda}_{k,i}}{\hat{\lambda}_{i,k}}[M(1+\epsilon_5(M,\delta,\delta',\delta''))]^{-1},\end{aligned}$$

for $\epsilon_5 = O(\epsilon_4) + O(M^{2-\beta})$. Let $\gamma_i = \sum_{k\neq i}\frac{\hat{\lambda}_{ki}}{\hat{\lambda}_{ik}} > 1$, by (11). Then

$$u_i(\delta M) = u_i(\delta'' M)e^{-\gamma_i(1+\epsilon_5)(\delta-\delta'')} + M\frac{1-e^{-\gamma_i(1+\epsilon_5)(\delta-\delta'')}}{\gamma_i(1+\epsilon_5)}.$$

By a Taylor expansion, $u_i(\delta M)$ becomes

$$\begin{aligned}&u_i(\delta'' M)[1-\gamma_i(1+\epsilon_5)(\delta-\delta'')+O(\delta^2)]+M[(\delta-\delta'')+O(\delta^2) \\ \leq& M[1+\delta''+(\delta-\delta'')[1-\gamma_i(1+\epsilon_5)]+O(\delta^2)]\end{aligned}$$

as $u_i(\delta'' M) \leq M(1+\delta'')$ by (18). Assume now that M is large, and set $\delta = \Theta(M^{(\beta-2)/2})$ and δ', δ'' to be proportional to δ, such that $\delta' < \delta'' < \delta$ and $\delta'' + (\delta - \delta'')[1-\gamma_i] < 0$. It then follows that $\epsilon_5 = O(\delta)$. Hence for large enough M (and small enough δ) $u_i(\delta M) = M[1+\delta''+(\delta-\delta'')[1-\gamma_i]+O(\delta^2)] < 0$ and the lemma follows. □

Hence, outside a bounded set, $\sup_i u_i$ has to decrease (*i.e.*, is a Lyapunov function), and the theorem follows. □

5.2 Proof of Theorem 3

We now establish that under condition (11), the original Markov process $\mathbf{N}(t)$ is ergodic. To this end, we shall rely on the *fluid limit* approach. That is to say, we shall identify a Lyapunov function F, and establish that, for initial condition $\mathbf{N}(0)$ such that $F(\mathbf{N}(0)) = M$, then , for large enough M, it holds that

$$\mathbf{E}F(\mathbf{N}(\delta M)) \leq (1-\epsilon)M, \tag{19}$$

for suitable positive constants $\delta, \epsilon > 0$. Unsurprisingly, the line of argument parallels that of Theorem 2's proof, with some additional elements introduced to take care of the random fluctuations in the process.

The Lyapunov function to be considered is

$$F(N) := \sup_{i\neq j} N_{ij}/\lambda_{ij}.$$

Define the event $\Omega_{ij} = \{N_{ij}(M\delta) \geq \lambda_{ij}M(1-\delta)\}$. We first establish the following intermediate result.

LEMMA 2. *On the event Ω_{ij}, for some positive constants $\gamma, c > 0$, for all $k \neq i$, with probability $1 - e^{-\Theta(M)}$ one has*

$$N_{ij}(t) \in [\lambda_{ij}M(1-c\delta), \lambda_{ij}M(1+c\delta)],\ t \in [0, M\delta], \tag{20}$$

$$N_{ik}(M\delta) \in [\gamma M, \lambda_{ik}M(1+c\delta)], k \neq j. \tag{21}$$

PROOF. Let E_{ik} denote the unit rate Poisson processes used to generate the arrival times of type (ik)-users in the system. Consider the event $\Omega_1 = \{|E_{ik}(\lambda_{ik}M\delta) - \lambda_{ik}M\delta| \leq M\lambda_{ik}\delta/2, k \neq i\}$. Then using Chernoff bounds, it is readily seen that its probability is at least $1 - e^{-\Theta(M)}$.

To establish (20), it suffices to note that

$$N_{ij}(M\delta) - E_{ij}(\lambda_{ij}M\delta) \leq N_{ij}(t) \leq N_{ij}(0) + E_{ij}(M\delta),$$

and on the event $\Omega_1 \cap \Omega_{ij}$, the left-hand side is at least $\lambda_{ij}M(1-(5/2)\delta)$ and the right-hand side is at most $\lambda_{ij}M(1+(3/2)\delta)$. (20) thus holds with $c = 5/2$.

Consider next $k \neq j$. We introduce now the notation $D_i(t)$ to represent the number of departures of users requesting object i in time interval $[0,t]$. On the event Ω_1, necessarily $D_i(M\delta) \leq rM$ for some suitable constant r. Indeed, $D_i(M\delta)$ cannot exceed $N_{i\cdot}(0) + \sum_{k\neq i} E_{ij}(M\delta\lambda_{ij})$, which in turn is no larger than $\sum_{k\neq i} M\lambda_{ik}(1+(3/2)\delta)$ on Ω_1, given the initial condition $F(N(0)) = M$.

Introduce now $D_{ik}(t)$ to represent the number of departures of type (ik)-users during time interval $[0,t]$. This process is generated as follows: at each jump time T of the counting process $D_i(\cdot)$, conditional on the past of the process before time T, a type (ik)-user is chosen to leave the system with probability $N_{ik}(T^-)/N_{i\cdot}(T^-)$. An explicit construction of this selection mechanism can be made by attaching a uniform random variable U_n to each jump point T_n of the process D_i in $[0, M\delta]$, and by letting

$$D_{ik}(t) = \sum_{n:T_n \geq t} \mathbb{1}_{U_n < N_{ik}(T_n^-)/N_{i\cdot}(T_n^-)}.$$

As previously established, on the event $\Omega_1 \cap \Omega_{ij}$, one has $N_{i\cdot}(t) \geq M\gamma$ for all $t \in [0, M\delta]$, and $D_i(M\delta) \leq rM$. This entails that, on this event, $D_{ik}(M\delta) \leq \sum_{n=1}^{rM} Z_n$, where $Z_n := \mathbb{1}_{U_n < \left(X - \sum_{\ell=1}^{n-1} Z_\ell\right)/M\gamma}$, and $X := N_{ik}(0) + E_{ik}(\lambda_{ik} M\delta)$. Indeed, type (ik)-departures are more likely if arrivals occur at the beginning of the interval $[0, M\delta]$.

This yields a first lower bound:

$$N_{ik}(M\delta) \geq Y := X - \sum_{n=1}^{rM} Z_n. \tag{22}$$

To simplify this further, one can note that the resulting random variable Y is stochastically reduced if one replaces X in both this expression and the definition of the random variables Z_n by a lower bound. On the event Ω_1, such a lower bound consists in $M\rho$ with $\rho = \lambda_{ik}\delta/2$.

We now control the probability that the lower bound Y in (22) is below a threshold τM for some constant $\tau > 0$, taking $X = M\rho$. We have the following representation: $\mathbf{P}(Y < \tau M) = \mathbf{P}(\sum_{n=0}^{(\rho-\tau)M} V_n \leq rM)$, where the random variables V_n are independent, geometrically distributed with parameter $(\rho M - n)/(\gamma M)$. We omit details, but Chernoff's bounding technique can be used, by evaluating the Laplace transform of the random variable $\sum_{n=0}^{(\rho-\tau)M} V_n$, to show that, for small enough constant $\tau > 0$, the probability $\mathbf{P}(Y < \tau M)$ is at most $\exp(-\Theta(M))$. This concludes the proof of the Lemma. □

We next need the following Corollary.

Corollary: On the event Ω_{ij}, for any $\delta' < \delta$, with probability $1 - \exp(-\Theta(M))$, the following holds for all $k \neq i$:

$$N_{k\cdot}(M\delta') \leq \mathrm{Bin}(O(M), e^{-\Theta(M^{2-\beta})}) + \mathrm{Poi}(\Theta(M^{\beta-1})),$$

where Bin denotes a Binomial random variable, Poi a Poisson variable, that are mutually independent.

Proof. On Ω_{ij}, with probability $1 - e^{-\Theta(M)}$ it holds that $N_{ij}(M\delta') \geq M(1-(5/2)\delta)$. The previous Lemma, suitably modified, therefore applies, and thus, there must exist a constant $\delta'' < \delta'$ such that with probability $1 - e^{-\Theta(M)}$, the following holds: $N_{ik}(t) = \Omega(M),\ t \in [M\delta'', M\delta']$. Consider now the dynamics of $(N_{k\cdot})$. Arrivals occur at a rate $\lambda_{k\cdot}$, and departures occur at a time-varying rate $N_{k\cdot}(t)N_{\cdot k}(t)N(t)^{-\beta}$. The product $N_{\cdot k}(t)N(t)^{-\beta}$ is at least $\Omega(M^{1-\beta})$ on the interval $[\delta'' M, \delta' M]$ by the previous argument. Thus its state at time $M\delta'$ can be upper-bounded by that of a M/M/∞/∞ queue, with initial state $N_{k\cdot}(M\delta'')$ at time $M\delta''$, arrival rate $\lambda_{k\cdot}$ and death rate $\Omega(M^{1-\beta})$. Now, with probability $1 - e^{-\Theta(M)}$, it holds that $N_{k\cdot}(M\delta'') = O(M)$, and the result follows. □

We are now ready to conclude the proof of the Theorem. To this end, we place ourselves on the event Ω_{ij}, and derive bounds on the trajectories $N_{ij}(t)$ for t in the interval $[M\delta', M\delta]$, relying on the previous results.

As we have just seen, with probability $1 - e^{-\Theta(M)}$, the components $N_{ik}(M\delta')$ are of order $\Theta(M)$. Furthermore, following the same lines as in the proof of (21), we can deduce from the fact that $N_{ij}(t) = N_{ij}(0)(1+O(\delta)),\ t \in [0, M\delta]$ that

$$N_{ik}(t) = N_{ik}(M\delta')(1+O(\delta)),\ t \in [M\delta', M\delta]. \tag{23}$$

Let us now introduce dedicated unit rate Poisson processes $\Delta_{k\ell}$ for each user type $(k\ell)$, and consider the representation

$$\begin{aligned} N_{k\ell}(t) = \quad & N_{k\ell}(M\delta') + E_{k\ell}(\lambda_{k\ell}(t - M\delta')) \\ & -\Delta_{k\ell}(\mu \textstyle\int_{M\delta'}^t N_{k\ell}(s) N_{\cdot k}(s) N(s)^{-\beta} ds). \end{aligned}$$

Replacing in the above $N_{\cdot k}(s)$ by an upper bound of order $N_{ik}(M\delta')(1+O(\delta))$, and $N(s)$ by a lower bound of order $N_{i\cdot}(M\delta')(1+O(\delta))$, we obtain a process $N^+_{k\ell}(t)$ that is an upper bound to $N_{k\ell}(t)$, and that is an $M/M/\infty/\infty$ process with arrival rate $\lambda_{k\ell}$ and death rate $N_{ik}(M\delta')N_{i\cdot}(M\delta')^{-\beta}(1+O(\delta))$.

Subsequently, we can also derive lower-bounding processes $N^-_{k\ell}(t)$ by upper-bounding $N_{\cdot k}(s)$ by

$$N_{\cdot k}(s) \geq N_{ik}(M\delta')(1+O(\delta)) + \textstyle\sum_{m\neq i,k} N^+_{mk}(s),$$

and lower-bounding $N(s)$ by $N(M\delta')(1+O(\delta))$ in the argument of $\Delta_{k\ell}$. Note now that the processes $N^+_{k\ell}$ have a stationary distribution that is Poisson with parameter $O(M^{\beta-1})$. Thus with high probability, their supremum over $[M\delta', M\delta]$ is small compared to $N_{i\ell}$, itself of order M. Eventually, we obtain that with high probability, $N_{k\ell}(t)$ admits lower bounds $N^-_{k\ell}(t)$ that are $M/M/\infty/\infty$ processes with arrival rate $\lambda_{k\ell}$ and death rates again equal to

$$N_{ik}(M\delta')N_{i\cdot}(M\delta')^{-\beta}(1+O(\delta)).$$

These lower bounds in turn will provide upper bounds on N_{ik}, by writing

$$\begin{aligned} N_{ik}(t) \leq \quad & N_{ik}(M\delta') + E_{ik}(\lambda_{ik}(t - M\delta')) \\ & -\Delta_{ik}(\textstyle\int_{M\delta'}^t N_{ik}(s)[\sum_{\ell\neq i} N^-_{\ell i}(s)]N(s)^{-\beta} ds. \end{aligned} \tag{24}$$

The argument of Δ_{ik} is lower-bounded by

$$N_{ik}(M\delta')N_{i\cdot}(M\delta')^{-\beta}(1+O(\delta)) \int_{M\delta'}^t \sum_{\ell\neq i} N^-_{\ell i}(s) ds.$$

By the ergodic theorem, applied to the $M/M/\infty/\infty$ processes $N^-_{\ell i}$, this integral reads with high probability with respect to M:

$$(t - M\delta') \sum_{\ell\neq i} \lambda_{\ell i} \frac{N_{i\cdot}(M\delta')^\beta}{N_{i\ell}(M\delta')}(1+O(\delta)).$$

Upon simplification, we have with high probability, replacing in (24) the Poisson processes E_{ik} and Δ_{ik} by their expectation, up to some error vanishing as M increases,

$$\begin{aligned} N_{ik}(M\delta) - N_{ik}(M\delta') \leq \quad & M(\delta - \delta')\,[\lambda_{ik} - -(1+O(\delta)) \times \dots \\ & \dots N_{ik}(M\delta') \textstyle\sum_{\ell\neq i} \frac{\lambda_{\ell i}}{N_{i\ell}(M\delta')}\Big]. \end{aligned}$$

Let $a_{ik}(t) = \lambda_{ik}^{-1} N_{ik}(t)$. The previous equation reads

$$a_{ik}(M\delta) - a_{ik}(M\delta') \le M(\delta - \delta') \times \cdots$$
$$\cdots \times \Big[1 - a_{ik}(M\delta') \textstyle\sum_{\ell \neq i} \frac{\lambda_{\ell i}}{\lambda_{i\ell}} \frac{1}{a_{i\ell}(M\delta')}(1 + O(\delta))\Big].$$

Now, for the index k for which $a_{ik}(M\delta')$ is largest,the right-hand side of the above is no larger than

$$M(\delta - \delta')\big[1 - (1 + O(\delta)) \sum_{\ell \neq i} \frac{\lambda_{\ell i}}{\lambda_{i\ell}}\big],$$

itself strictly smaller than $-M\epsilon$ for some positive ϵ, if we chose δ small enough, when condition (11) is in force. This enables to conclude that, with high probability,

$$\sup_{i,j} N_{ij}(M\delta)/\lambda_{ij} = F(N(M\delta)) \le (1 - \epsilon) F(N(0)).$$

The same bound applies to the expectation of the left-hand side, using a uniform integrability argument. This establishes the desired contraction property of the Lyapunov function F and, hence, the ergodicity of the original Markov process. □

5.3 Proof of Theorem 4

To show that there can be at most one i for which (12) holds, we observe that if it holds for some i, then for any other $j \neq i$, we have $\hat\lambda_{j,i}/\hat\lambda_{i,j} < 1$. This implies that any other j satisfies (11), so no $j \neq i$ can also satisfy (12).

To prove the remainder of the theorem, we use the notation $z = x \pm y$ to indicate that $z \in [x - y, x + y]$. Suppose that $\gamma_i = \sum_{k \neq i} \hat\lambda_{ki}/\hat\lambda_{ik} < 1$ for some i and assume that $u_i(0) = M > 0$, for some large M. Assume further that for $j \neq i$, $u_j(0) = u_i^{1-\beta} \frac{\hat\lambda_{i\cdot}^\beta}{\hat\lambda_{i,j}}(1 \pm \epsilon)$, for some small $\epsilon > 0$. The following lemma then holds:

LEMMA 3. *For all $\epsilon > 0$, there exists $M_0 > 0$ s.t. for all $M > M_0$ there exists $\delta > 0$ such that if $u_i(0) = M$ and $u_i^{1-\beta}(0)u_j(0) = (1 \pm \epsilon)\hat\lambda_{i\cdot}^\beta/\hat\lambda_{i,j}$, then $u_j(t)u_i^{1-\beta}(t) = (1 \pm \epsilon)\hat\lambda_{i\cdot}^\beta/\hat\lambda_{i,j}$ for all $t \in [0, \delta]$.*

PROOF. Fix some $\epsilon > 0$. The lemma follows by the continuity of the fluid trajectories if $u_i^{1-\beta}u_j$ is in the interior of $\frac{\hat\lambda_{i\cdot}^\beta}{\hat\lambda_{i,j}}(1 \pm \epsilon)$. Suppose thus that it is at the boundary. We consider the upper boundary case, *i.e.*, $u_j u_i^{1-\beta} = \frac{\hat\lambda_{i\cdot}^\beta}{\hat\lambda_{i,j}}(1+\epsilon)$ and show that $\frac{d}{dt}u_j u_i^{1-\beta}$ is negative, so that $u_j u_i^{1-\beta}$ is forced in the interior; the same argument can be used to show that the derivative is positive when at the lower boundary, so we omit this case. Indeed

$$\frac{d}{dt}u_i^{1-\beta}u_j = (1-\beta)\dot u_i u_i^{-\beta}u_j + u_i^{1-\beta}\dot u_j$$
$$= (1-\beta)u_i^{-\beta}u_j(1 - u_i\frac{\sum_{k\neq i}\hat\lambda_{ki}u_k}{n^\beta}) + u_i^{1-\beta}(1 - u_j\frac{\sum_{k\neq j}\hat\lambda_{kj}u_k}{n^\beta}).$$

Note that $u_j(0) = \Theta(u_i^{\beta-1}(0))$, where the asymptotic notation is as $M \to \infty$. Observe that, since $\hat\lambda_{i',j'} > 0$ for all $i', j' \in \mathbb{K}$, we have that at time $t = 0$, $0 < \frac{\sum_{k\neq i}\hat\lambda_{ki}u_k}{n^\beta} = \Theta(u_j u_i^{-\beta}) = \Theta(u_i^{-1})$, and $0 < \frac{\sum_{k\neq j}\hat\lambda_{kj}u_k}{n^\beta} = u_i^{1-\beta}\frac{\hat\lambda_{i,j}}{\hat\lambda_{i,\cdot}^\beta}(1 + o(1))$. We thus have that, at $t = 0$,

$$\frac{d}{dt}u_i^{1-\beta}u_j = \frac{\hat\lambda_{i\cdot}^\beta}{\hat\lambda_{i,j}}(1+\epsilon)\Big(\frac{1}{u_j} - u_i^{1-\beta}\frac{\hat\lambda_{i,j}}{\hat\lambda_{i,\cdot}^\beta}(1 + o(1)) + O(u_i^{-1})\Big)$$
$$= \frac{\hat\lambda_{i\cdot}^\beta}{\hat\lambda_{i,j}}(1+\epsilon)\Big(\frac{1}{u_j} - u_i^{1-\beta}\frac{\hat\lambda_{i,j}}{\hat\lambda_{i,\cdot}^\beta}(1 + o(1))\Big)$$

as $u_i^{-1} = o(u_i^{1-\beta})$ for $\beta < 2$. On the other hand as $u_j u_i^{1-\beta} = \frac{\hat\lambda_{i\cdot}^\beta}{\hat\lambda_{i,j}}(1+\epsilon)$ implies that $u_j^{-1} = u_i^{1-\beta}\frac{\hat\lambda_{i\cdot}^\beta}{\hat\lambda_{i,j}}\frac{1}{1+\epsilon} < u_i^{1-\beta}\frac{\hat\lambda_{i\cdot}^\beta}{\hat\lambda_{i,j}}$, so for M large enough the above quantity is negative. □

Consider now a fluid trajectory in which $u_i(0) = M$ and $u_i^{1-\beta}(0)u_j(0) = \frac{\hat\lambda_{i\cdot}^\beta}{\hat\lambda_{i,j}}(1 \pm \epsilon)$. Then we have that

$$\dot u_i = 1 - u_i\frac{\sum_{k\neq i}\hat\lambda_{ki}u_k}{n^\beta} = 1 - u_i\frac{\sum_{k\neq i}\hat\lambda_{ki}\frac{\hat\lambda_{i,\cdot}^\beta}{\hat\lambda_{i,k}}u_i^{\beta-1}1 \pm \epsilon}{(\sum_k \hat\lambda_{k,\cdot}u_k)^\beta}$$
$$= 1 - u_i\frac{\sum_{k\neq i}\frac{\hat\lambda_{k,i}}{\hat\lambda_{i,k}}\hat\lambda_{i\cdot}^\beta u_i^{\beta-1}(1 \pm \epsilon)}{\hat\lambda_{i\cdot}^\beta u_i^\beta(1 + o(1))},$$

which for large enough M and a small enough ϵ becomes $1 - \sum_{k\neq i}\frac{\hat\lambda_{ki}}{\hat\lambda_{ik}}(1 + o(1)) > 0$. This, along with Lemma 3 implies we can select an $\epsilon > 0$ such that, for large enough M, if $u_i(0) = M$ and $u_i^{1-\beta}(0)u_j(0) = \frac{\hat\lambda_{i\cdot}^\beta}{\hat\lambda_{i,j}}(1 \pm \epsilon)$, then there exists a $\delta > 0$ s.t. $u_i'(0)$ is positive and bounded away from zero uniformly in M and $u_i^{1-\beta}(t)u_j(t) = \frac{\hat\lambda_{i\cdot}^\beta}{\hat\lambda_{i,j}}(1 \pm \epsilon)$, for all $t \in [0, \delta]$. This in turn implies that the above is true for all $t \ge 0$, and, in particular, that u_i diverges to infinity. □

5.4 Proof of Theorem 5

Let us define $x_i = n_{i,\cdot}$ and $\rho_i = \hat\lambda_{i,\cdot}(2\mu C)^{-1}$, $i \in \mathbb{K}$. By (14b), we have

$$n_{\cdot,i} = C\rho_i(\textstyle\sum_{j:j\in\mathbb{K}} x_j)^\beta / x_i. \tag{25}$$

Using (25), we can rewrite (14) as the following equivalent convex optimization problem involving only x_i:

$$\begin{aligned} \text{Minimize} \quad & \textstyle\sum_{i\in\mathbb{K}} x_i \\ \text{subj. to:} \quad & \textstyle\sum_{i\in\mathbb{K}}(\rho_i x_i^{-1}) \le (\sum_{i\in\mathbb{K}} x_i)^{1-\beta}, \quad i \in \mathbb{K} \quad (26a) \\ & x_i \ge 0, \quad i \in \mathbb{K}. \quad (26b) \end{aligned}$$

We can write its Lagrangian function as

$$\Lambda(\mathbf{x}, \varphi, \mathbf{w}) = \textstyle\sum_{i\in\mathbb{K}} x_i + \varphi h(\mathbf{x}) + \sum_{i\in\mathbb{K}} w_i g(x_i),$$

where φ and $\mathbf{w} = [w_i]_{i\in\mathbb{K}}$ are Lagrangian multipliers, $\mathbf{x} = [x_i]_{i\in\mathbb{K}}$, $h(\mathbf{x}) = \sum_{i\in\mathbb{K}}\rho_i x_i^{-1} - (\sum_{i\in\mathbb{K}} x_i)^{1-\beta}$, and $g(x_i) = -x_i$. Hence, any $\tilde{\mathbf{x}} = \{\tilde x_1, ..., \tilde x_K\}$ is optimal if and only if it satisfies the following KKT conditions [3]:

$$h(\tilde{\mathbf{x}}) \le 0, \quad g(\tilde x_i) \le 0, \quad i \in \mathbb{K}, \tag{27a}$$
$$\varphi \ge 0, \quad w_i \ge 0, \quad i \in \mathbb{K}, \tag{27b}$$
$$\varphi h(\tilde{\mathbf{x}}) = 0, \quad w_i g(\tilde x_i) = 0, \quad i \in \mathbb{K}, \tag{27c}$$
$$\frac{d\Lambda}{dx_i}(\tilde{\mathbf{x}}, \varphi, \mathbf{w}) = 0, \quad i \in \mathbb{K}. \tag{27d}$$

We know that $x_i > 0, \forall i \in \mathbb{K}$ from (26a) and (26b). Thus condition (27c) requires $w_i = 0, \forall i \in \mathbb{K}$ and (27d) needs $\varphi \neq 0$. Then by $\varphi h(\tilde x) = 0$ in (27c) and condition (27d), any optimal solution $\tilde{\mathbf{x}}$ must satisfy the following two equations:

$$h(\tilde{\mathbf{x}}) = 0, \quad \frac{d\Lambda}{dx_i}(\tilde{\mathbf{x}}, \varphi, [0]) = 0$$

Solving these two equations leads to the unique solution $x_i = \sqrt{\rho_i}(\sum_{j:j\in\mathbb{K}}\sqrt{\rho_j})^{\frac{\beta}{2-\beta}}, i \in \mathbb{K}$, as shown in (15a), and the Lagrangian multiplier $\varphi = (\sum_{i\in\mathbb{K}}\sqrt{\rho_i})^{\frac{2\beta}{2-\beta}}(2-\beta)^{-1}$.

By plugging x_i into (25), we can derive the value of $n^*_{\cdot,i}$ as (15b), and (15) is the unique optimal solution of (14). □

6. BARON: GUIDING CACHE REPLACEMENT VIA VALUATIONS

Our analysis in Section 5.4 has identified the optimal stationary points that minimize the average sojourn time. However, we have not described a method for leading the system to such points. In this section, we present BARON to bridge this gap. BARON dictates how peers should exchange content items so that the system converges to the optimal points defined in Theorem 5. We also demonstrate BARON's performance using numerical simulations.

BARON is a centralized scheme. In particular, it requires estimating the demand and supply of each item $i \in \mathbb{K}$, captured by the population of peers requesting and storing i, respectively. In practice, individual peers may maintain estimates of these quantities, *e.g.*, either by gossiping or sampling. However, studying decentralized schemes for estimating the demand and supply is beyond the scope of this paper. As a result, we focus on scenarios in which these quantities are readily monitored through at a centralized tracker.

6.1 Designing BARON

To lead the system to the optimal point, one intuitive way is to first identify which items are over-replicated and which are under-replicated. Whenever two peers come into contact, if one has an over-replicated item i and the other has an under-replicated item j, then the first peer replaces its item i with item j. This replacement increases the current supply $n_{\cdot,i}$ of the under-replicated item.

Valuations in BARON. BARON keeps track of whether an item is currently over-replicated or under-replicated in following way. In particular, for each content item i, BARON maintains a real-valued variable v_i. We will call this variable the *valuation* of item i.

Our choice of valuation is inspired by (16), which states that at an optimal point the supply of an item is C times the demand. Motivated by this, the valuations are given by

$$v_i(t) = Cn_{i,\cdot}(t) - n_{\cdot,i}(t), \quad i \in \mathbb{K}. \tag{28}$$

A positive valuation $v_i > 0$ indicates that item i is currently under-replicated. Similarly, a negative valuation $v_i < 0$ indicates that item i is currently over-replicated.

One appealing property of (28) is that it requires prior knowledge *only* of the cache capacity C; in particular, it does not require knowledge of the arrival rates $\lambda_{i,f}$ of each peer class. Nevertheless, this valuation requires to track the supply and demand for each item.

Content exchange guided by valuations. BARON is a centralized design that relies on a central controller to maintain the valuations (28). In addition, this central controller lists the valuations on a public board, and makes them available to all peers.

The content exchanges between peers are guided by these valuations following a *negative-positive rule*. More specifically, during a contact event between a peer A with cache f and a peer B with f', each peer checks if it has any over-replicated items. If so, it further checks whether the other peer has any under-replicated items that it has not already stored in its cache. If such a pair of items exists, a replacement takes place. In particular, the first peer A replaces the item with the minimal negative valuation in its cache, *i.e.*, peer A removes item i such that

$$i = \text{argmin}\{v_x | x \in f, v_x < 0\}.$$

Then, among the under-replicated items in the peer B's cache f' yet not in peer A's cache f, peer A replicates the item with the maximal positive valuation, *i.e.* peer A selects item j such that

$$j = \text{argmax}\{v_y | y \in f' \setminus f, v_y > 0\}.$$

After retrieving item j from peer B, peer A replaces i with j. Hence its cache f changes to $(f \setminus \{i\}) \bigcup \{j\}$. A similar procedure follows for peer B.

Clearly, there are other ways to design valuations and the rules for guiding content exchanges via valuations. In Section 6.3, we will examine other options for these two design components of BARON.

Based on the above definitions, the conversion probabilities of BARON satisfy Assumption 1 because of the positive-negative rule. As a result, by Theorem 1, we can study the dynamics of BARON through its fluid trajectory.

6.2 Evaluating BARON

We evaluate BARON's fluid trajectories using numerical simulations in MATLAB. Our main observation is that, by guiding the content exchanges through valuations, BARON converges to the optimal stationary points defined in (15), which minimize the average sojourn time.

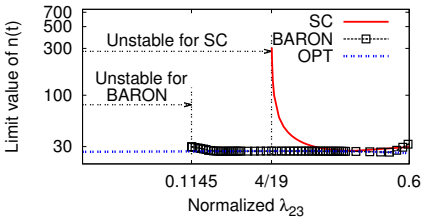
6.2.1 BARON vs. Static-Cache Policy

We compare BARON to the static-cache policy by the examining system stability and optimality when using each design. Then we further use the static-cache policy as an example to demonstrate that only one swarm becomes unstable when instability occurs.

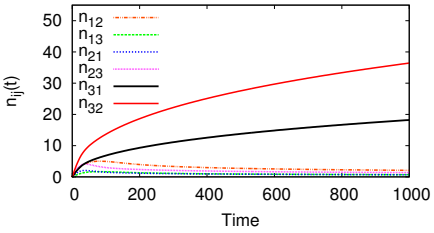
We simulate the following scenario. Assume there are three items $\{1, 2, 3\}$ in the system, and peer's cache size is one. Hence we have six peer classes, where each class of peers requesting item i and caching item j $(\neq i)$ has a normalized arrival rate of $\hat{\lambda}_{i,j}$. Peers requesting one item form one swarm, leading to three swarms in total. We set the contact process parameters as $\beta = 0$ and $\mu = 0.002$. We assume initially no peer is in the system.

Stability. We begin with comparing the system stability under BARON and the static-cache policy. In particular, we aim to understand under which conditions of arrival rates, the system stabilizes when using each design. So we leave $\hat{\lambda}_{23}$ as a free variable, and fix the relative ratios of the other five classes as $\frac{1}{5}, \frac{1}{15}, \frac{2}{15}, \frac{1}{5}, \frac{2}{5}$, respectively of $(1 - \hat{\lambda}_{23})$. To identify the system stability for a given $\hat{\lambda}_{23}$ value, we examine the system's fluid trajectory over a significantly long time ($t \approx 10^5$).

Figure 2(a) shows the rescaled peer population when the system can stabilize as we vary $\hat{\lambda}_{23}$. We see that when using the static-cache policy, the system stabilizes only when $\hat{\lambda}_{23}$ is above $\frac{4}{19}$. This verifies the conclusion in Theorem 2 since $\frac{4}{19}$ is the arrival rate $\hat{\lambda}_{23}$ that violates condition (11). In con-

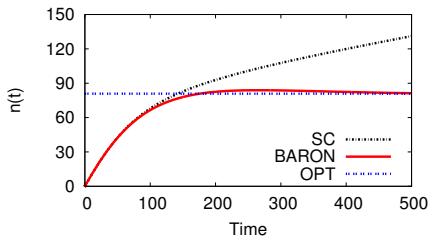


(a) Limit points of the fluid trajectory for variable $\hat{\lambda}_{23}$.

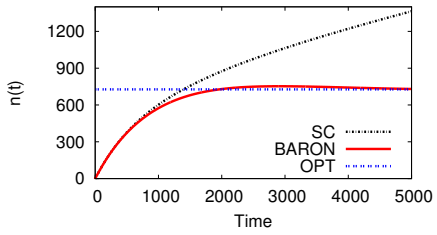


(b) Static-cache policy w/ $\hat{\lambda}_{23} = \frac{4}{19}$.

Figure 2: Comparing BARON and static-cache (SC) policy by varying arrival rate configurations.

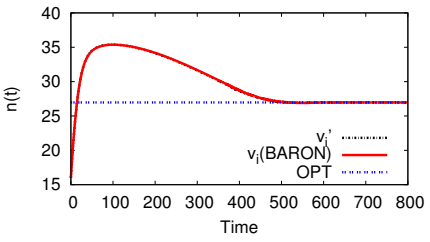


(a) Contact-constrained communication w/ $\beta = 0.5$.

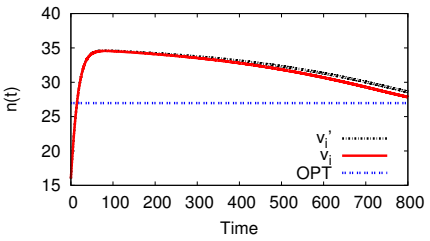


(b) Constant-bandwidth communication w/ $\beta = 1$.

Figure 3: BARON under various β.



(a) Varying valuation designs under NPR.



(b) Varying valuation designs under LHR.

Figure 4: Examining the peer population under various design options in BARON.

trast, when using BARON, the system has a much larger stability region. More specifically, the system is able to stabilize when $\hat{\lambda}_{23}$ is larger than 0.1145. This demonstrates BARON's effectiveness of guiding content exchanges.

Optimality. We further examine the stationary state that the system converges to when using BARON and the static-cache policy. As shown in Figure 2(a), the system under the static-cache policy converges to a non-optimal state. Moreover, the closer $\hat{\lambda}_{23}$ is to the stability boundary $\frac{4}{19}$, the more peers in the stationary state. In contrast, BARON is able to guide the system to the optimal stationary state if the system stabilizes. This demonstrates that BARON achieves optimality by the use of valuations.

Single swarm instability. Now we examine how peer classes evolve in time when instability occurs under the static-cache policy. Figure 2(b) shows the population of each peer class along the time when $\hat{\lambda}_{23} = \frac{4}{19}$, demonstrating the conclusion in Theorem 4: only *one* swarm can become unstable.

Recall that peers requesting the same item form one swarm. Our main observation is that only the swarm requesting item 3 blows up. This is because item 3 is the one that does not satisfy (11). As this swarm grows, peers in other swarms can obtain their requested items quickly and depart. Hence the supply for item 3 further decreases.

6.2.2 *Dependence on β*

To comprehensively understand BARON's performance, we extend to cases with other β values. In particular, we examine two cases with $\beta = 0.5$ and $\beta = 1$ respectively. As shown in Section 3.4, a larger β indicates a smaller contact rate. The case when $\beta = 1$ is the constant-bandwidth communication scenario where a peer's contact rate is constant regardless of the peer population. We do not simulate the case where $\beta > 1$, because the optimality result identified in Section 4.3 does not hold for such β. We configure the other parameters as in Figure 2 with $\hat{\lambda}_{23} = \frac{1}{6}$.

Figures 3(a) and (b) show the evolution of the total number of peers in time. The main observation is that, while the system does not stabilize when using the static-cache policy, the system under BARON converges to the optimal in both cases. This demonstrates the effectiveness of the valuations under various communication settings. Even though the aggregate contact rate decreases as β increases, BARON is still able to adapt the item supply according to the demand, guiding the system towards the optimal.

Furthermore, as the contact rate becomes smaller when β increases, the system with BARON takes longer time to stabilize to the optimal. This is because the item replacement and replication only occur during contact events. A smaller contact rate slows down the adjustment of the item distribution, leading to a slower convergence.

6.3 Comparing to Other Designs

BARON has two design components – the valuations in (28) and the negative-positive rule. Now we experiment with other designs for these two components, and examine their performance in comparison to BARON.

An alternative valuation v_i' is

$$v_i'(t) = n^*_{\cdot,i} - n_{\cdot,i}(t), \quad i \in \mathbb{K}, \tag{29}$$

i.e., item i's valuation is defined as the distance of its current supply $n_{\cdot,i}$ to the optimal $n^*_{\cdot,i}$ as given by (15b). Note that in (29), computing the optimal supply $n^*_{\cdot,i}$ requires the knowledge of several system parameters, including the arrival rates $\lambda_{i,f}$, $(i,f) \in \mathbb{C}$, and the contact process parameters μ and β. Obtaining the values of these parameters could be difficult in practice.

Moreover, instead of the *negative-positive rule* (NPR) in BARON, another rule of guiding content exchanges via val-

uations is replacing one item with another as long as the other item has a higher valuation and the item is not already stored. We refer to it as the *lower-higher rule* (LHR).

We examine all four combinations of these design options, where BARON is the combination of NPR with valuations v_i defined in (28). We use the same configuration as Section 6.2.2, and assume initially 16 peers request item 3.

Figures 4(a) and (b) plot the trajectories of peer population under various design combinations. We observe that none of the other combinations performs better than BARON.

In addition, in terms of the comparison of design options for each component, we make the following observations. First, the two valuations perform similarly, and v_i performs better than v_i' under LHR. Second, the system with NPR converges to the optimal faster than LHR. This is interesting because NPR is stricter than LHR, and one would expect that LHR leads to a faster convergence by enabling more frequent replications and replacements. Indeed, from Figures 4(a) and (b), we observe that the peer population stays around its peak ($\approx$35) for a longer time when using NPR. Nonetheless, NPR is able to catch up later. While this demonstrates the efficiency of restricting the replacement to over-replicated item only, the analytical reason beneath is worthwhile to further explore.

7. CONCLUSIONS AND FUTURE WORK

In this paper, we made the first attempt towards a systematical understanding of universal swarms, where peers share content across peer-to-peer swarms. We have rigorously proved that such content exchange across swarms significantly improves stability compared to a single autonomous swarm. We also have proved convergence to a fluid limit for a general class of content exchanges; our theorem thus paves the way for the analysis of more complicated exchange schemes than the one described in the present work.

An important future research direction lies in further investigating the parallels between our work and "missing piece syndrome" in single swarms [6]. In particular, once a "one-club" forms in a swarm, each "one-club" peer has idle bandwidth capacity. It can thus contact uniformly at random peers and seeders at other swarms to obtain C items and place them in its cache, and subsequently continue to sample other swarms to see if it can retrieve and/or offer a missing piece. From this point on, our model applies: peers wish to retrieve one item (their "missing piece"), and leave the system immediately once they retrieve it (corresponding to the "grab-and-go" principle).

This is of course a simplification of the above system, as it ignores the "growing" phase when peers acquire all chunks of a file but the last one, as well as the cache-filling phase. However, in light of the stability properties we observed in this work, understanding if, *e.g.*, the stability region increases through such exchanges, is an interesting open question.

Our analysis leaves several additional open questions, including formally characterizing the stability conditions of BARON, and analytically studying BARON's convergence to optimal stationary points. Our model can also be extended in various ways, including multi-item request, heterogeneous cache sizes and contact rates. The case where arrival rates are not strictly positive, and peers arrive with a partially-filled cache are also worth considering.

Finally, while our model assumes peers are cooperative, it would be interesting to investigate the strategic behavior of peers in universal swarm systems.

Acknowledgements

We would like to thank the anonymous reviewers for their helpful suggestions and insights. This work is supported in part by the FP7 EU project "SCAMPI" and the ANR French project "PROSE".

8. REFERENCES

[1] Altman, E., Nain, P., and Bermond, J.-C. Distributed storage management of evolving files in delay tolerant ad hoc networks. In *INFOCOM* (2009), pp. 1431 –1439.

[2] Altman, E., Neglia, G., De Pellegrini, F., and Miorandi, D. Decentralized stochastic control of delay tolerant networks. In *INFOCOM* (2009), pp. 1134 –1142.

[3] Boyd, S., and Vandenberghe, L. *Convex Optimization*. Cambridge University Press, 2004.

[4] Chaintreau, A., Le Boudec, J.-Y., and Ristanovic, N. The age of gossip: spatial mean field regime. In *SIGMETRICS* (2009), pp. 109–120.

[5] Cohen, E., and Shenker, S. Replication strategies in unstructured peer-to-peer networks. *SIGCOMM Comput. Commun. Rev. 32* (2002), 177–190.

[6] Hajek, B., and Zhu, J. The missing piece syndrome in peer-to-peer communication. *Information Theory Proceedings, 2010 IEEE International Symposium on* (June 2010), 1748 –1752.

[7] Hu, L., Le Boudec, J.-Y., and Vojnoviae, M. Optimal channel choice for collaborative ad-hoc dissemination. In *INFOCOM* (2010).

[8] Ioannidis, S., Chaintreau, A., and Massoulié, L. Optimal and scalable distribution of content updates over a mobile social network. In *INFOCOM* (2009).

[9] Ioannidis, S., Massoulié, L., and Chaintreau, A. Distributed caching over heterogeneous mobile networks. In *SIGMETRICS* (2010), pp. 311–322.

[10] Massoulié, L. Structural properties of proportional fairness: Stability and insensitivity. *The Annals of Applied Probability 17*, 3 (2007), 809–839.

[11] Massoulié, L., and Twigg, A. Rate-optimal schemes for peer-to-peer live streaming. *Perform. Eval. 65* (November 2008), 804–822.

[12] Massoulié, L., and Vojnovic, M. Coupon replication systems. *Networking, IEEE/ACM Transactions on 16*, 3 (June 2008), 603 –616.

[13] Qiu, D., and Srikant, R. Modeling and performance analysis of BitTorrent-like peer-to-peer networks. *SIGCOMM Comput. Commun. Rev. 34* (2004), 367–378.

[14] Reich, J., and Chaintreau, A. The age of impatience: optimal replication schemes for opportunistic networks. In *CoNEXT* (2009), pp. 85–96.

[15] Towsley, D. The internet is flat: a brief history of networking in the next ten years. In *PODC* (2008), pp. 11–12.

[16] Zhou, X., Ioannidis, S., and Massoulié, L. On the stability and optimality of universal swarms. Tech. rep., Technicolor, 2011.