



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

D1.1: System Architecture Specification

Due date of deliverable: 31/03/2011
Actual submission date: 31/03/2011
Revision submission date: 14/07/2011

Start date of project: October 1st 2010
Duration: 36 months
Project Manager: Professor Jörg Ott
Revision: v.1.1

Abstract

This deliverable describes the architecture for the service platform designed and developed in the SCAMPI project. In particular, the report explains the central concepts of the project as well as the envisioned usage scenarios and the work plan to implement the architecture.

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	21.01.2011	First version of the deliverable	Mikko Pitkänen (AALTO), Teemu Kärkkäinen (AALTO), Jörg Ott (AALTO), Silvia Giordano (SUPSI), Franck Legendre (ETHZ), Sacha Trifunovic (ETHZ)
v.0.2	7.3.2011	Second iteration of the deliverable	Mikko Pitkänen (AALTO), Teemu Kärkkäinen (AALTO), Franca Dellmastro (CNR)
v.1.0	31.3.2011	Final version of the deliverable	Mikko Pitkänen (AALTO), Martin Potts (Martel), Monique Calisti (Martel)
v.1.1	8.7.2011	Major revision of the deliverable addressing the 6 th month SCAMPI review recommendations and extending the previous architectural specification	Mikko Pitkänen (AALTO), Teemu Kärkkäinen (AALTO), Jörg Ott (AALTO), Franck Legendre (ETHZ), Sacha Trifunovic (ETHZ), Monique Calisti (Martel)

Table of Contents

1	OVERVIEW OF THE SCAMPI APPROACH	4
1.1	POSITIONING SCAMPI FROM THE STRATEGIC PERSPECTIVE	5
1.2	KEY DISTINGUISHING FEATURES OF SCAMPI	6
1.3	SCAMPI AND STATE OF THE ART	6
1.4	SIMPLE USAGE SCENARIO.....	8
1.5	KEY CONCEPTS	8
2	USE CASE SCENARIOS.....	12
2.1	COLLABORATIVE AND PERVASIVE YOUTUBE	13
2.2	MOBILE ONLINE AND ON-THE-MOVE SOCIAL NETWORKING	13
2.3	PERVASIVE SENSING OR DIGITAL SENSE.....	14
2.4	OPPORTUNISTIC TWITTER.....	15
3	FUNCTIONAL SPECIFICATION OF THE SCAMPI SERVICE PLATFORM.....	17
3.1	THE THREE LAYERS OF THE SCAMPI ARCHITECTURE.....	17
3.2	SCAMPI APPLICATIONS.....	18
3.3	SCAMPI PLATFORM SERVICES.....	19
3.4	OPPORTUNISTIC STACK.....	21
3.5	COMPONENT BREAKDOWN OF FUNCTIONAL ARCHITECTURE	21
3.6	SCAMPI COMMUNICATION MODEL.....	23
3.7	SCAMPI SECURITY MODEL	26
3.8	USING OAUTH (FACEBOOK, TWITTER, ETC.) FOR IDENTITY INITIALIZATION	27
3.9	PLATFORM-LEVEL SECURITY AND PRIVACY CONSIDERATIONS	30
4	CONCLUSION AND NEXT STEPS.....	31
	REFERENCES	35

1 OVERVIEW OF THE SCAMPI APPROACH

The SCAMPI project will build a service platform for social-aware mobile and pervasive computing. The aim of the project is to enable end users to benefit not only from the resources available on their own devices, but also to opportunistically exploit resources offered within their environment, including those on other users' devices, in a trustable and secure way. The core vision is that all resources available in the environment (residing on users devices, fixed elements such as WiFi hotspots, fixed cameras, etc.) will be pooled together, and exploited opportunistically and on-demand by users. Applications will thus provide to the users a much richer functionality than what is available on any individual and isolated device.

With respect to conventional service oriented approaches, SCAMPI explores several new directions. First of all, in SCAMPI, services abstract resources that are available also on users' personal devices, and are contributed voluntarily by the users. Moreover, the *networking* environment is assumed to be way less stable than the conventional Internet environment, thus requiring technical solutions to be completely redesigned.

The service-oriented human pervasive network we envision is a network that joins together features of traditional pervasive networks and opportunistic networks. As in pervasive networks, we assume that the mobile users will be immersed in an environment with a multitude of **devices that are able to share** networking and computational resources. As in opportunistic networks, we assume that **the network is extremely dynamic** with unstable topologies, and therefore it is necessary to use opportunities to form contacts with other devices to carry on networking and computing tasks. Access to the conventional Internet infrastructure will also be supported, with such access being seen as a service offered opportunistically. More specifically, infrastructure access is viewed as a service, which can be composed for achieving a required goal, for example, transferring a piece of content to some user not in physical proximity. This approach integrates access to the infrastructure in the overall service-oriented design.

The SCAMPI project builds on **pervasive and opportunistic communication** paradigms. We move from a pure networking perspective, and investigate the definition and implementation of a **service-oriented platform for mobile and pervasive networks**. Therefore, instead of addressing pure networking issues, such as efficient message delivery between endpoints, the project investigates **collaborative ways to enable shared services** by exploiting the hardware and software resources available on users' devices.

The **human factor** is a key element of our technical approach. On one hand, because we will investigate innovative examples of **social-oriented services** enabled by the SCAMPI solutions, and on the other hand, because **information about the social behavior of the users**, i.e., social awareness, **will be exploited** as core contextual information for the technical solutions developed within the project.

The resources used by SCAMPI can be of several types. They can be **physical resources**, such as a CPU, a camera, a microphone, or a sensor. Or, they can be **soft resources**, such as a self-standing piece of code providing a service component.

1.1 Positioning SCAMPI from the Strategic Perspective

The SCAMPI approach does not aim at reinventing the wheel in the areas of online social networking, opportunistic networking, or service-oriented architectures. Instead, the approach **combines the powerful features from the above domains** to create an innovative architecture for social-aware opportunistic and pervasive networks that builds on top of a service oriented model.

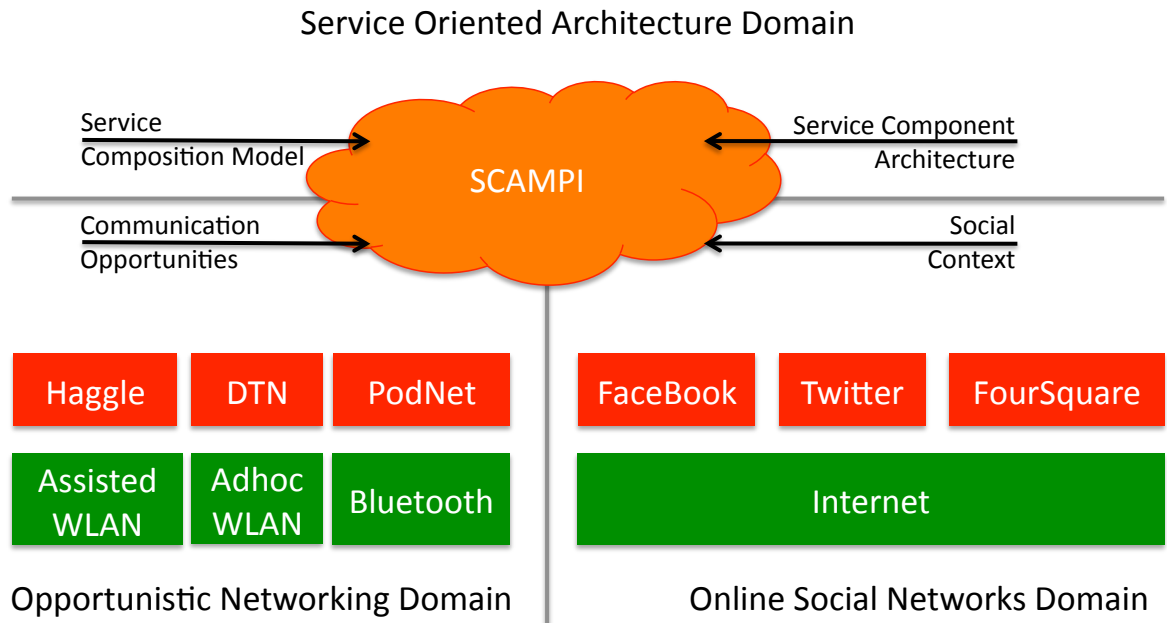


Figure 1: Positioning SCAMPI from the strategic perspective.

Figure 1 illustrates the positioning of SCAMPI in relation to existing networking and service creation paradigms. SCAMPI takes two key features from the **service-oriented architecture domain**, namely 1) creating complex services by **composing** them from simple service components and 2) following a component-based approach where each **service component encapsulates** a minimal, but meaningful self-standing service. From the **opportunistic networking domain** SCAMPI takes the ability to communicate by opportunistically using available resources. These can be, for example, peer-to-peer connections in a WLAN that is assisted by an access point, or purely ad-hoc communications over WLAN or Bluetooth. On top of these physical opportunities, several different network stacks can be used to provide messaging infrastructure. Depending on the chosen stack, the communication models can vary. For example, DTN stack supports unicast and multicast delivery to end-points (in theory) located anywhere with and without DTN infrastructure, whereas the PodNet stack focuses regional content dissemination in pure ad-hoc environments. From the **online social networks (OSNs) domain** SCAMPI takes the ability to infer social context and relations between the actors of the network. In the SCAMPI context, declared friendship can be imported from an existing online social network, for which many popular implementations exist with open APIs [10,11], and then used during an opportunistic encounter to determine if the meeting nodes belong to the same social context. This information can be used, for example, to provide implicit trust relationship, to take forwarding or service invocation decisions, or to enable higher service level offerings to well known nodes.

1.2 Key Distinguishing Features of SCAMPI

The main distinguishing feature of SCAMPI is providing services over opportunistic networks. We recall that opportunistic networks have not only to deal with a distributed environment but also with mobility and **communication prone to disconnections**; since SCAMPI does not require infrastructure, **our mobile devices are both network providers and network users**. This results in one of the **most challenging and complex environments one can consider to run any services**. One assumption we make is that SCAMPI should be able to handle purely opportunistic networks, namely without any infrastructure support, in addition to infrastructure-assisted ones.¹

Such a strong assumption is not necessary in all cases: in urban areas, for example, cellular infrastructure networks are usually omnipresent and WLAN hotspots are plentiful. Even then, however, users may not be able to access such networks at all times (because of coverage, lacking roaming agreements, or overload) or may not want to (for cost reasons). The very central idea of SCAMPI to aim for infrastructure-less operation at its core will cater for users in the broadest possible sense – and it enables that **our solution can also be used under extreme conditions (emergency and disaster situations)**. We however can relax this assumption when implementing our solutions as a published software application.


Summarizing, the main added value of SCAMPI is to propose a networking platform which can **provide the basis for services in one of most challenging environments** that has been considered until now. The numerous aspects that distinguish the SCAMPI approach from the current State of the Art solutions are explained in the following section with comparison to different scientific and practical solutions. Moreover, the remainder of the document provides detailed explanation about numerous challenges that current approaches face in the challenging scenarios and approaches that the SCAMPI architecture takes to solve them.

1.3 SCAMPI and State of the Art

As noted above, **SCAMPI builds upon several well-established complementary fields** – social networking, opportunistic communication, and service-oriented architectures – and **integrates elements of those to provide a novel way of communication and service provisioning**. SCAMPI also follows the state of the art in application, service, and interface design, both taking advantage and integrating existing services and extending their concepts for mobile opportunistic operation. The following examples highlight some shortcomings that present state of the art approaches in their respective domains would face when used in challenging scenarios addressed by the SCAMPI project.

The SCAMPI approach will overcome these shortcomings. **FaceBook** is the most popular platform for online social networking, but **does not work during disconnections**, particularly when both communicating parties cannot connect to a centralized service hosted by the FaceBook service provider. However, the social network structures can be used to imply trust and reachability relation between their users. **DTN** enables communications during disconnections; however, the DTN model focuses on providing plain message forwarding functionality, but **does not provide guidelines for rich service provisioning**. **Android**

¹ This implies that SCAMPI will support traditional infrastructure-based (read: Internet-based) operation as well, but this is the well-understood common way of running communication services today and hence not in the focus of SCAMPI.

v.1.0	<i>D1.1: System Architecture Specification</i>	 7
-------	--	--

platform provides a service-based architecture for mobile application development, however, the services are local components and Android **does not fully support service composition across devices**, for example, LocationManager service cannot be accessed from collocated device. Moreover, services provided by a pocketable Android smartphones are **pervasive but do not encourage collaboration**. Following positions the SCAMPI approach further in relation to state of the art.

In relation to scientific state of the art, SCAMPI consortium both follows and actively participates to define the leading edge of the research in the field of communications and networking research. This is proven by several publications by the project members in top quality publications in the field of communications, see up-to-date publications list in [12]. Moreover, the project has already peer-reviewed publications that discuss the social aspects of rich mobile computing service provisioning [13] and that have introduced also practical implementations [14]. In addition to own contributions, several SCAMPI members bring relevant experience from recent projects that are currently considered state of the art in the field of opportunistic computing. These include, for example, the work in the recent Hagggle [1] and ANA [20] projects, as well as ongoing efforts in PodNet [3], DTN [16], and work on Floating Content [15]. In relation to these projects, **the key innovative direction that SCAMPI pursues is the focus on the service composition**, instead of just enabling opportunistic communication or content access and distribution. In addition to efforts within the consortium, the project members follow and cooperate with relevant cooperators in the fields of communication and opportunistic computing research as well as social behavioral research.

In relation to social networking applications, SCAMPI extends the underlying networking functionality to **support applications suitable for opportunistic environments** such as location-based games, location-based content dissemination or any mobile application in general; therefore social-networking applications are one potential subclass to run on using SCAMPI. Beyond social network applications, such as Facebook or Youtube, SCAMPI aims at **supporting services at large** with innovative concepts such as social filtering and opportunistic cloud computing, which have not yet been proposed elsewhere. Now regarding the positioning of SCAMPI vs. existing social-networking platforms, the following explains added value of SCAMPI considering the case of Twitter, which is a centralized online service. The SCAMPI approach aims at integrating the SCAMPI architecture into a Twitter application so that tweets can be exchanged in an opportunistic environment, for example, emergency or disaster situations but also in gatherings such as concerts or sport events where the infrastructure network is usually saturated or even out of order. Of course, **the approach still supports the online features** of Twitter but also **provides users with the means to leverage opportunistic communications**. Moving from a centralized approach to a decentralized environment (with mobility and disconnections) requires reconsidering all networking aspects (from routing to caching to security and privacy management), challenges that are described in further detail in SCAMPI technical annex. In terms of added value, opportunistic networking **enables offloading infrastructure networks, sustain communications in environment where the infrastructure is overloaded or simply down**. But most importantly, it **relocates user communications to the real-world and within their social network**. As most user-to-user communication on social networking platforms is geographically localized [17], that is, one communicates mostly with co-located peers, using online servers that are centralized and also remote can be seen as an aberration.

In relation to practical level state of the art, the SCAMPI architecture extends on and integrates the state of the art software solutions from several topical domains. The service platform provided by SCAMPI integrates to service based mobile computing architectures that are seen in modern smartphones such as Android. On the social networking context, software components and services from the most widely used OSN platforms such as authentication APIs of FaceBook [10] are integrated as part of the SCAMPI platform. Moreover, service models are developed that support the operation of current OSNs such as Twitter to work in opportunistic and only partially connected environments, as explained in Section 2.4. The main innovative departure of SCAMPI in relation to practical state of the art is to **enable rich service composition also in disconnected environments**.

1.4 Simple Usage Scenario

A simple scenario is when two mobile smart-phone users meet and engage into a SCAMPI interaction.

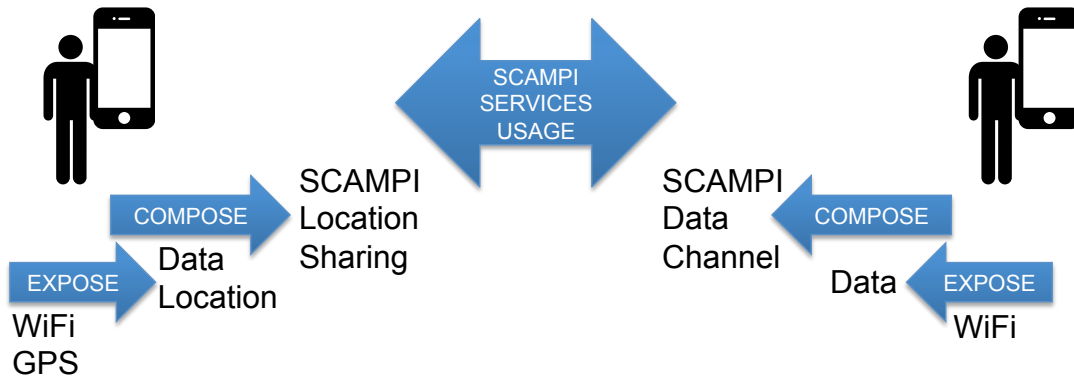


Figure 2: A simplified usage scenario of the SCAMPI platform.

Figure 2 presents a simplified situation where two SCAMPI users **opportunistically meet and share resources**. One user possesses resources for location and communications while the other user has resources only for communications. The user with the location resources then composes a location sharing service and declares this as **a SCAMPI service** that is accessible for users in the proximity. The other user without local location resources can then access the location information through the declared SCAMPI service. In a more realistic scenario, the set of resources are richer in a single SCAMPI device. The number of users involved in the resource sharing can also be significantly larger, for example, several participants in a concert or in a sports event.

1.5 Key Concepts

The main architectural concepts of the SCAMPI platform, shown in Figure 3, are introduced in the remaining of this sub-section.

At the bottom of the conceptual architecture the resources layer comprises the multitude of resources available in the environment, for example, those available in users' modern smart phones. Access to device's **physical resources** is abstracted through a service interface. Such abstraction occurs for example, when accessing the WiFi chipset through the socket interface or GPS information via a location service interface. The network (each network interface) is viewed as a special resource, which is managed through an opportunistic and pervasive networking stack.

The resources can be also **soft resources**; such as piece of code that implements particular program logic or content that can be provided to other peers. The main difference between the physical and soft resources is that soft resources can be moved from one device to another by the SCAMPI platform. The physical resources are on the other hand tied to the particular devices where they are installed.

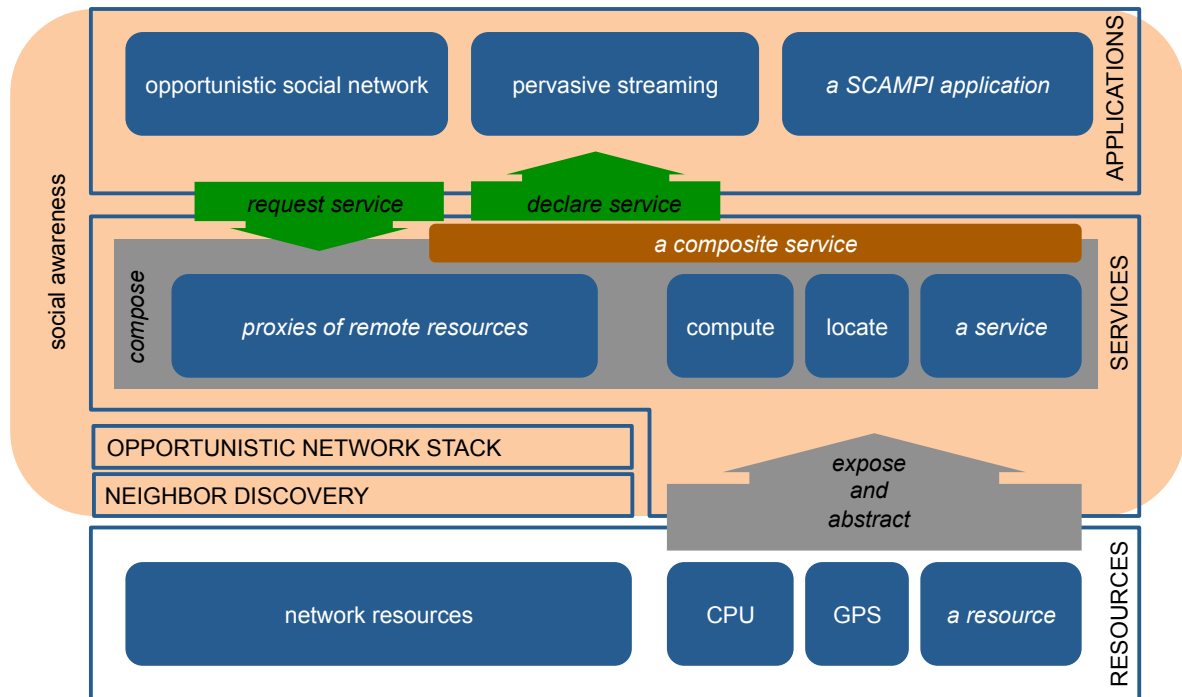


Figure 3: Conceptual overview of the SCAMPI architecture.

In the middle layer, the **platform operates on services** and part of the details about physical resources may be abstracted out. Abstraction can be carried out to the extent that the resource can even be physically located on remote device. The opportunistic networking stacks manage the discovery of - and access to - such devices. These stacks provide the **basic enabling networking services** to the node, such as opportunistic multi-hop forwarding, unicasting, multicasting, and so on. These technologies are able to cope with the challenged networking environment we mainly focus on, characterized by possibly long disconnections and partitions, and for which, in general, it is not possible to provide stable routing abstractions as the one available in static ad hoc networks. Also, these technologies allow the exploitation of direct contacts among mobile nodes as well as opportunities to connect to the fixed infrastructure, when available and needed.

At each individual node, abstractions will be available both for locally available resources, as well as for resources that can reside on other devices, and which can be accessed opportunistically by the node. As of the latter, proxies will be defined, which describe these remote resources. Descriptions will also provide information about their availability (e.g., how often the device providing a resource is directly encountered).

The SCAMPI architecture can compose a service out of single or multiple local or remote resources. As a result of composition, the composite service is declared to applications that use the SCAMPI platform.

Applications request services from the composition block, which after composition declares **a composite service** to the requesting applications. A special case of the composite service is a simple service component that abstracts a single resource.

Services within the SCAMPI architecture are semantically meaningful actions or series of actions that nodes can perform in response to requests by applications. Service requests are made by applications and may be satisfied by the local node or transparently by remote nodes through opportunistic networking.

Service composition is a process of combining local or remote resources, logic and state. The composition itself is a non-generic process that often is service-dependent and thus not implemented, but rather supported by the SCAMPI service platform.

While resources are not directly exposed through the SCAMPI architecture, a minimal service may provide **access to raw resources** such as readings from GPS hardware. Other services can then use this access to provide a composite service such as a location service.

Nodes providing SCAMPI services are often personal devices within opportunistic communication range of the user. Providing services to other nodes can be costly in terms of resource usage, in particular energy. This is in contrast to classical networking where powerful centralized nodes provide services to clients at a trivial per-transaction cost.

The nature of the devices, the short-range communications and the high resource cost of providing services imply significance of the **social context**. Users might not be willing to spend resources providing a service to anyone at any time, but rather change their behavior based on the social context. For example, a user might not wish to provide a file storage service to anyone they randomly meet on the street, but do wish to provide the service to the participants of the same meeting.

The use of remote resources in opportunistic environments as sketched above will require **rich contextual information**. Given that the availability and stability of remote resources in the near future can only be estimated at the local node (in the general case), and the topological information about the network will be unstable and imprecise, contextual information shall be used to provide more precise estimates of resource availability in SCAMPI. From this standpoint, another distinguishing point of SCAMPI will be the exploitation of **social awareness as key contextual information** for the service infrastructure. By social awareness we mean awareness about the social structures of users relationships, the probability of users to communicate and get physically in touch with other users, etc.

In the SCAMPI vision, social relationships will not only be a key enabler for applications, like in conventional Online Social Networks, but will also be exploited to design technical solutions inside the service platform. Social awareness can be exploited **to predict which remote resources will be available as well as when** they will be available. This will make it possible to accumulate information for the respective proxies, and to predict which resources/modules could be exploited by nodes in the near future, and therefore which services could be provided to applications and within which time and at which cost. It must be noted that social awareness in SCAMPI will also be used as an enabler for innovative social-oriented services. For example, readings from **multi-modal sensing**, i.e., joint readings from sensors available on smartphones, such as cameras, microphones, etc., can be used to detect the current activities of the user. This information can be exploited to design services such as

enhanced status updates, in which the status of a user as resulting from multi-modal sensing is automatically advertised to the members of the users social communities.

As another example, the same information can be used to determine what the user is doing and/or where it is moving at a given point in time, information that can be further exploited to understand the current **social context** of the user, and its suitability with respect to a given task to be carried out by the platform, for example, if the user is visiting a particular landmark in a city, she will be a good candidate to take a snapshot of the landmark with her camera. This will complement the information coming from social structures of relationships among users. While the social structures provide **long-term, statistical information** about which resources will be available when, multi-modal sensing provides **situated information** about the current status of the users. These two pieces of complementary information, are expected to provide much more powerful predictive tools, which the SCAMPI service platform will exploit, as well as enable innovative service-oriented framework to support applications.

The core enabling technology of the SCAMPI architecture is the ability for nodes to **opportunistically communicate** with surrounding peers. There are many approaches to creating communication opportunities, including direct device-to-device communications, multi-hop communications using Mobile Ad-hoc Networking (MANET) protocols and space-time messaging using Delay-Tolerant Networking (DTN) technologies.

Each approach has its own characteristics in terms of expected delays, feasible data volumes and frequency of communication opportunities. Different SCAMPI services may require **different classes of communication services** in order to perform optimally. For example, a video-streaming service may require either a direct connection or a stable MANET path between the service provider and the user, while a photo-sharing service may work perfectly well over asynchronous DTN style messaging.

Creating communication opportunities between applications and services is one of the key tasks of the SCAMPI service platform. However, the platform is not tied to any particular approach to opportunistic networking and can instead be adapted to use the communication opportunities provided by several different available networking stacks. Examples of such stacks are Huggle [1], DTN [2], PodNet [3], DodWAN [4].

Targeting a possibly **challenged networking environment**, the use of remote resources by SCAMPI is significantly re-thought with respect to conventional Internet-based distributed systems. Remote resources can be available with different stability levels; depending on how long and how frequently the local and remote nodes are in touch, either directly or through an opportunistic multihop path. The stability of remote resources is a key parameter that the SCAMPI platform considers to deliver services to applications. In some cases, high stability is required, for example, in the case of a real-time streaming service in which the camera of a remote node is used to generate the stream (or a portion thereof), it is clearly important that the local and the remote node be in touch at all times. However, if the local node needs to run a particular piece of code on a remote device, lower stability levels can be sufficient. For example, the local node can upload input parameters to the remote node, and fetch the execution results the next time the two nodes are within reach of each other. Specifically, the local node and the remote nodes where resources are physically available do not necessarily need to be connected at all times.

2 USE CASE SCENARIOS

This section illustrates rich use cases for SCAMPI applications and discusses scenarios where the SCAMPI service platform can be used for **opportunistic service creation**. The scenarios introduce four different types of applications that the SCAMPI platform supports and discusses their usage in different networks with varying node densities and usage environments. The sparse networks present networks where each user is in contact with only limited number of users, while in dense networks each user can simultaneously interact with large number of users. Rural networks cannot rely on existence of any infrastructure services and the communication must rely on direct device-to-device communications whereas in urban scenarios infrastructure support is often at least an option.

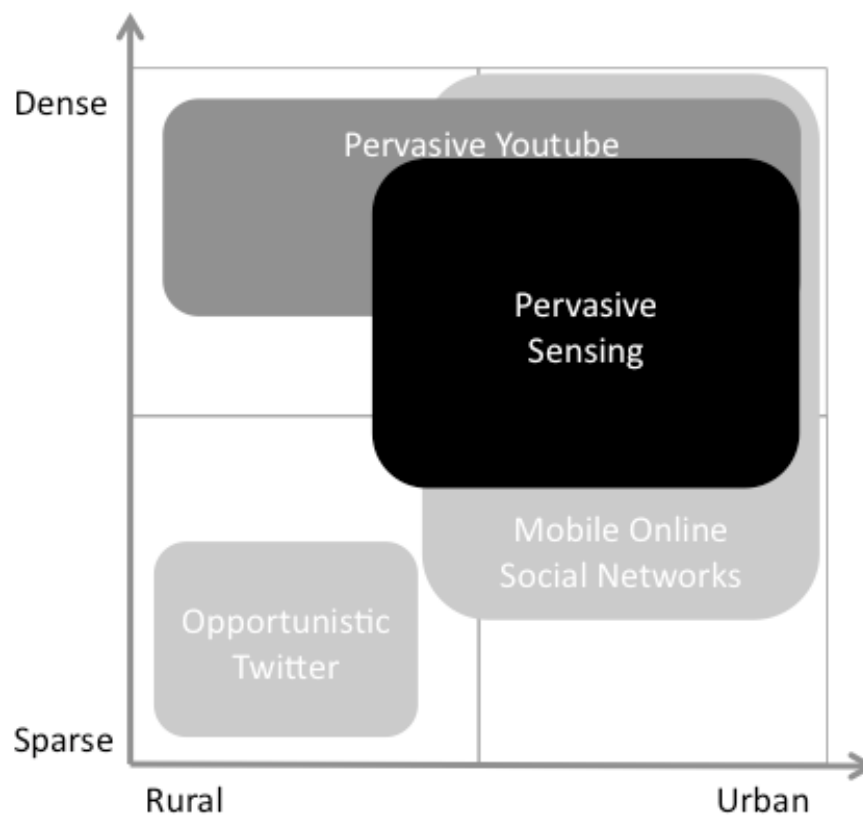


Figure 4: Positioning of SCAMPI use cases.

Figure 4 shows the different applications mapped in envisioned usage scenarios. First, **Pervasive YouTube** is good fit to dense networks, where several networks can participate in providing a media content as service. Benefits of dense networks are two fold. First, lot of content can be accessed from co-located devices. Second, duty cycling can be used between media providers to compose stream so that a single provider does not get its (battery) resources exhausted. **Pervasive sensing** is similarly seen as best fit dense and urban networks where its benefits arise from automatically interpreting communications and context, or buzz, in rapidly changing environment. **Mobile online social networks** will act as information gateway in urban scenarios and targets to information sharing in events where density is high as well as in sparse networks formed by opportunistically communicating tourists. **Opportunistic Twitter** provides an example adaptation of a popular application on top of

SCAMPI platform, which allows the otherwise network dependent application to work in scenarios where support for network connectivity is missing and nodes may be only few.

The above also classifies the applications into groups where different types of **scalability issues** can occur. In sparse scenarios the number of nodes is often few and messages can be exchanged with any encountered node. The application messages that are encapsulated into self-contained units can be passed on like envelopes as in traditional mail system, and do not require state keeping in intermediaries, thus allowing good scalability. Same holds for the rural scenarios, where access is available only to those users in sight and the number of users is significantly smaller than in globally operating online social networks. Thus state related to communication context can be handled in rural areas by individual smartphones. In dense scenarios the scalability concerns become more serious, but again it is possible to limit the communication to colocated users in the proximity of the user and the state keeping can be handled in communicating smart phones. In more static and dense scenarios, mechanisms from MANET routing can be used for optimizing routing and service discovery in scalable manner. The urban usage scenarios can use the existence of infrastructure support, in particular access to the Internet, to support creating very large-scale communications. As an example, the SCAMPI users can benefit from service provided by Facebook to mediate messages to millions of potentially interested users that connect to the service.

2.1 Collaborative and Pervasive YouTube

This SCAMPI video service allows the sharing of user generated multimedia content, that is, content generated by users using the recording capabilities of their mobile devices, combined with other multimedia content that may or may not be user generated. In addition, our pervasive video service allows the combination of video inputs from several colocated users attending a same event, for example, sport, lecture, or exhibition, resulting in a collaborative video experience. The sharing of content relies on opportunistic communications, for example, ad hoc WiFi, and leverages real-life social networks for the dissemination. The SCAMPI platform exploits information about users' interests as well as about the social communities they belong to in order to **proactively disseminate content** towards places where it is likely to be of interest. Information about social communities will hint at types of content to be disseminated within particular communities, as well as to provide information to efficiently move and replicate content in the network. Content sought after will also be generated by users in a participatory fashion, and distributed across the social communities defined by common interests. For instance, Alice is attending a concert. During the concert, she collaborates to mix video recordings of the concert with other concert attendees. Back at her university, she shares this collaborative video. By sharing this video, **Alice implicitly declares her interests** in the music band. In return, Alice would receive clips related to the band from other students. However, clips possibly combining different content types need to be in a format suitable to her device. To this end, the SCAMPI platform will look for available services in the surroundings of Alice's location, possibly availing of resources on other user devices to re-encode content. The proposed video service enables collaborative content generation that can be shared directly among personal devices in a peer-to peer fashion, as opposed to its online and centralized counterpart, YouTube.

2.2 Mobile Online and On-the-Move Social Networking

This SCAMPI application extends the reach of online social networking applications to

opportunistic and pervasive networks. It gives ability for users to become involved in participatory social interactions using applications such as content distribution, flea markets, microblogs, and round-based games. While opportunistic networks (through ad hoc communications) **leverage users' real-life social network** and locality, online networks (through WiMax or 3G) **leverage users' distant and virtual social networks**. Therefore, the SCAMPI architecture integrates wired and mobile opportunistic networks seamlessly to support hybrid on-the-move and on-line social networking applications. This can be applied to the pervasive video service described previously but also to any other online social networking application such as Flickr, EBay, and Facebook. With such architecture, for example, Alice is able to find easily her concert tickets by putting an ad in the hybrid flea-market application, which extends Ebay to opportunistic domains. She finds sold tickets straight away through the local opportunistic network by people she encounters everyday at her university. Once at the concert, Alice wants to post the collaborative video to her Facebook account but since the infrastructure-based wireless networks such as EDGE/3G are overloaded, that is, have reached capacity limits, **SCAMPI hybrid services** decide to postpone the publication and resort to opportunistic communication. While commuting back from the concert, Alice plays a round-based game with another commuter through ad hoc connectivity and a distant game opponent through 3G network without being aware of their physical locations. When back at home her mobile device has still enough battery to avoid being recharged since she was mostly using local ad hoc communications with a higher throughput.

2.3 Pervasive sensing or digital sense

This SCAMPI application provides a social information service that **uses the users device sensing capabilities** such as radio interfaces, microphones, or cameras, to provide information about the user behavior and its social interactions as well as information about the physical environment and local social communications. Leveraging on these inputs, this digital sense allows **further extending social-oriented applications and services to pervasive contexts**. The user behavior and social interactions can be exploited to update the current user status based on the inferred activity, for example, from accelerometer readings, the features of their physical surroundings, for example, inferred by fusing readings from microphones and cameras. Social networking applications can leverage this to tag users according to their status and decide whether they might be interested in certain types of content for example. Pervasive sensing can also enable location-based applications. The information about local social communication can help users to keep up to date with what happens in their surroundings when non-verbal communication contains a lot of valuable information. This service exploits information collected in the environment and processes it to build a real time image about the social environment. Eventually, information about the physical environment, that is, urban sensing, uses readings from cameras, microphones, accelerometers, etc. available from a user' device but also other users around as well as sensors that can be fused to provide multi-modal sensing information to applications. Such information can be used to create a number of distributed digital maps, ranging from maps of current people density or street light intensity, for example, to avoid walking into empty, non well lit, possibly dangerous parts of newly visited city, to maps of ongoing events in an area with music type, approximate queue line, etc. Finally, as an example of the social-oriented applications described above, Alice became aware of the concert since people around her were communicating about going to the concert, and not willing to actively keep it private.

She was at first planning a hike in the park but the local weather station collecting sensor measurements predicted rain showers and she eventually decided to look for tickets through the hybrid flea-market application since the concert was sold out. While at the concert, her device inferred the situation and changed her status automatically to offline mode in order to not be disturbed by friends willing to chat with her. While collaboratively creating a concert video, the sensors of its devices and surrounding devices taking part in the collaborative video decides on the elected video source based on the current quality they provide.

2.4 Opportunistic Twitter

We are currently developing an opportunistic Twitter client, called *Twitteth*, for Android that uses broadband access (2G/3G) when available and switches to opportunistic communications when the infrastructure is no longer available. Our objective is to support disaster and emergency situations so that distressed people would still be able to communicate. Extending the reach of Twitter to opportunistic domains highlighted many issues. The main challenge to overcome is to move from a centralized online platform to a distributed opportunistic and delay-tolerant environment especially when the original service has not been thought from a distributed perspective. **Figure 5** below shows how the application works in normal and in opportunistic modes. *Twitteth* accesses user data using OAuth and the API provided by Twitter, after the user grants access by logging in. *Twitteth* supports most basic Twitter functionality like showing a timeline (i.e., the most recent tweets of the followed users), sending tweets, re-tweeting and replying to and setting favorites, etc. In normal operation, *Twitteth* queries Twitter servers for new tweets every 2 Minutes. The tweets obtained from the queries are cached locally in a database file.

Additionally to normal Twitter client functionality, *Twitteth* has a setting to enable an opportunistic mode. Upon enabling this mode, the tweets of the user internally are assigned the special status of *opportunistic tweets* and are stored locally in a separate database for opportunistic spreading and later publication to the Twitter servers. In opportunistic mode, the device periodically scans for reachable Bluetooth devices. For the prototype, we choose a scanning interval of 120 Seconds. The scanning interval presents a tradeoff between energy consumption and delay (as a longer scanning interval misses short connection opportunities). We choose Bluetooth for opportunistic communication, because it is the ad hoc communication technology, which requires the least user interaction with current Android versions. Besides, WiFi ad hoc is still not supported on most popular mobile OSes (Android, iOS). Once two phones are close enough to connect to each other, they exchange their global opportunistic tweets, thereby spreading them epidemically. At the moment, we do not consider any limitations for the epidemic spreading such as hop count or geographic distance.

The current version relies on a light version of a opportunistic stack supporting node discovery and broadcast dissemination (see bottom of Figure 5). We are currently thinking of ways to add trust and security features to the service platform (see middle of Figure 5) as well as a supporting service for the dissemination of sensor data (e.g., accelerometer, GPS, etc) through *Twitteth*. One major issue for now is to allow the relaying of personal Twitter messages by opportunistic peers as this implies giving away his/her twitter credentials to a potentially non-trusted peer. OAuth was not designed for such scenarios and we are currently working to find out adapted security solutions.

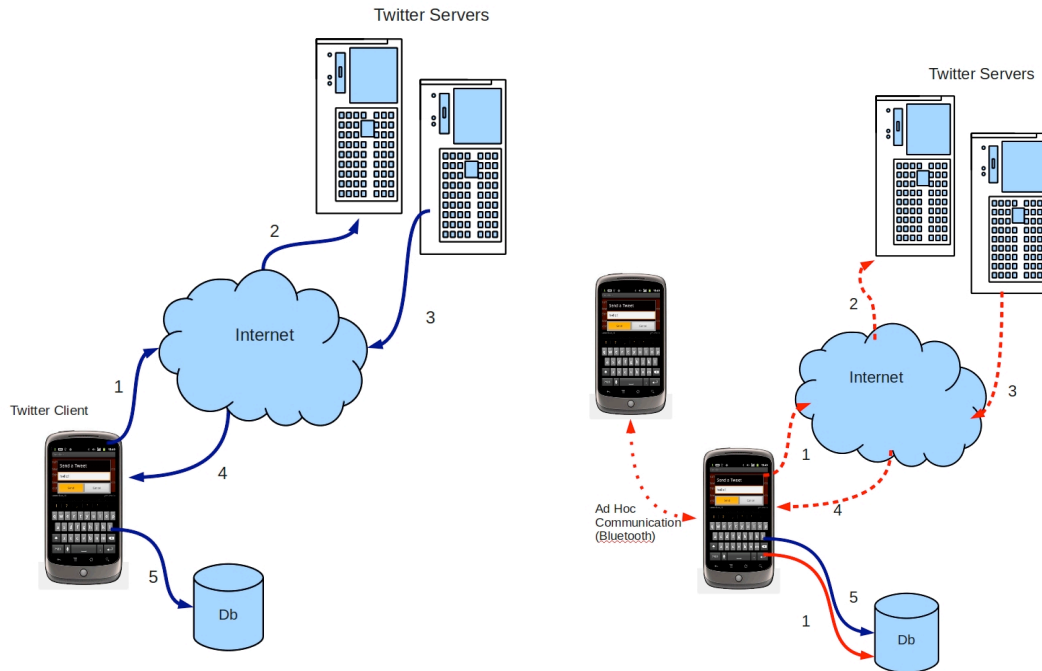
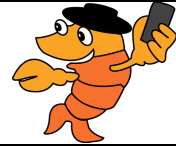


Figure 5 - Two modes of operation. *Left figure:* Continuous blue lines refer to normal operations when the fixed communication infrastructure is available. *Right figure:* When in opportunistic mode, dashed red lines refer to periodic attempts to connect to (i) opportunistic peers and (ii) the infrastructure, whilst continuous red lines refer to normal opportunistic operations.

3 FUNCTIONAL SPECIFICATION OF THE SCAMPI SERVICE PLATFORM

The core functionality of the SCAMPI service platform is structured into three layers. The following diagram explains how these layers relate to each other and the methods and message types that the layers use to compose services.

3.1 The three layers of the SCAMPI Architecture

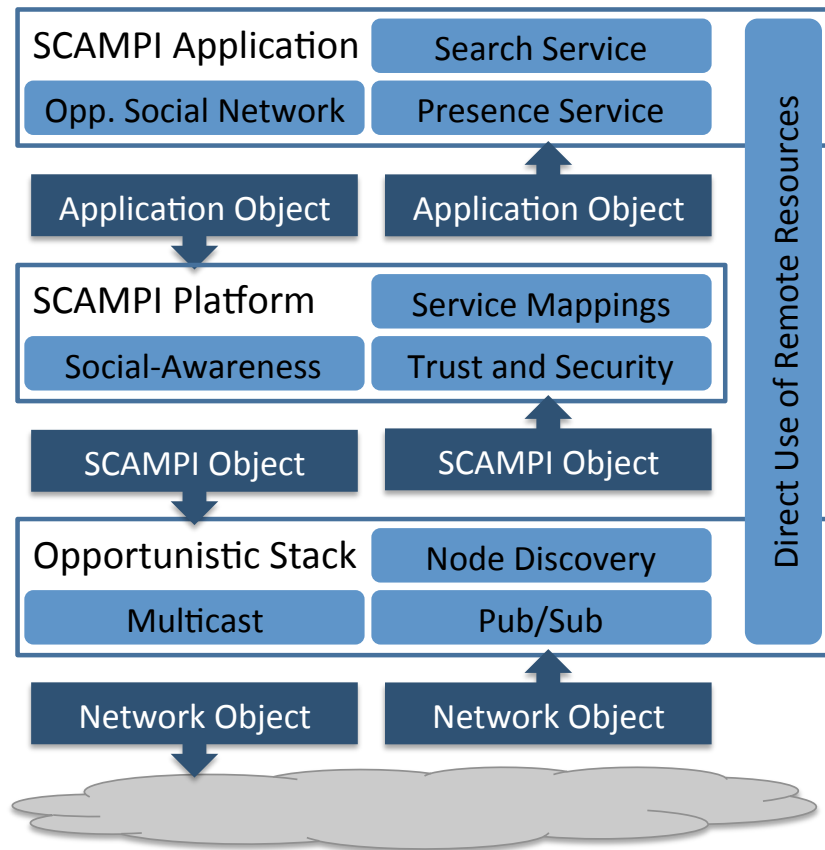


Figure 6: Three layer of the SCAMPI architecture.

Figure 6 illustrates the three functional layers of the SCAMPI architecture. At the top of the architecture are the **applications**, which act as an interface between the user and the SCAMPI platform. The application layer hosts services that include **application specific logic**, for example, knowledge on how certain types of queries are evaluated or how events from other nodes affect the application.

In the middle is the SCAMPI **service platform**, which provides **service-supporting mechanisms for distributed service composition**. The platform receives service requests from the application and returns to the application the best matching services that are either found locally or from remote hosts. To decide which service are the best match, the platform can use, for example, social-awareness and trust information.

The remote services can be either **accessed through** the SCAMPI platform, or the platform can act as an **enabler to locate and initialize** a communication channel for services. **Direct access** from applications to remote services may be needed, for example, in case of media

streaming, when it is not feasible to package stream data into self-contained SCAMPI service objects, but direct access to stream is preferred instead.

At the bottom of the figure is the **opportunistic networking** stack, which can use different actualizations, for example, DTN, PodNet, DoDWAN, MANET, to provide the opportunistic networking functionality. The main purpose of this layer is to **find and locate SCAMPI nodes** in the surrounding network and enable mediating service objects between them. The opportunistic stack receives from the service platform application data as SCAMPI objects, which contain information that assists in making intelligent networking decisions based on, for example, the current social context.

3.2 SCAMPI Applications

A **SCAMPI application** owns all state and logic that is specific to the application instance. Whereas the service layer below provides matching and state keeping for the SCAMPI services, the application layer can interpret the contents of the service messages, and compose them together to create meaningful **application states**. A user interface component provides a view on the state and the means to interact with the application.

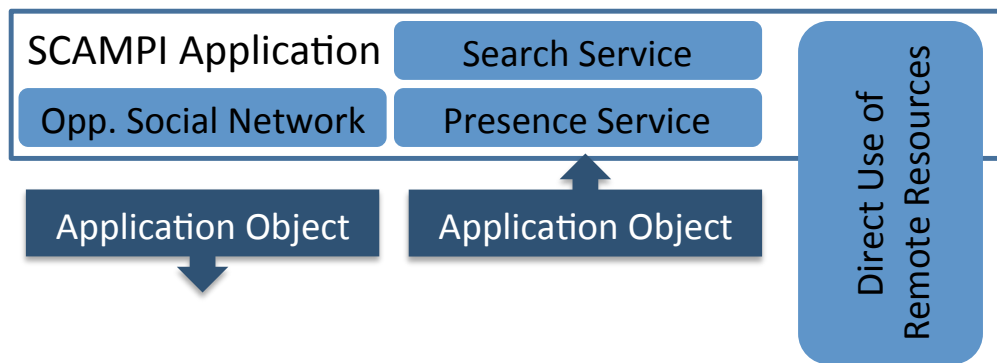


Figure 7: SCAMPI application layer.

The application interacts with other SCAMPI applications by using the services of the SCAMPI service platform. When seeking services, the **application declares service requests** that identify the sought services. After the service request is sent, the application listens for matching application data objects returned by the service platform. Upon reception of such an object, the application creates its **view on the distributed state**. Applications can also announce services to other nodes. A service announcement carries the identifier of the service interface, for example, a class name of the interface in the SCAMPI reference implementation.

Following this approach each SCAMPI application can request and declare services by using the following service commands.

- **seekService(ServiceInterface, ApplicationID, DataObject);** a method to seek for a SCAMPI service is parameterized by the identification of the service interface and the identification of the searching application. The optional DataObject parameter may contain the related application data that is passed to a service upon the arrival of service request to the service. An example of passing a data object with a service request is to send a request seeking gateway delivery to the Internet with the data for which the delivery is sought.

- **provideService(ServiceInterface, ApplicationID, DataObject);** a method to announce a service is parameterized by the identification of the service interface and the identification of the announcing application or service component. The optional DataObject parameter may contain the related application data that is passed to a requesting application upon the arrival of service announcement to the seeking application. An example of passing a data object with a service announcement is to send content with a service announcement that can be directly delivered to registered seekers such as subscribers to a news channel.

The application level service components are implemented as **self-standing components** that can be composed together to form a multitude of services. To achieve this goal, the applications need to be able to exchange data objects between services. Thus, the interaction between the application and the service components follows the principles of inter process communications (IPC) where data objects can be marshaled and un-marshaled for remote delivery between processes, or services in this case. However, it is up to the implementation in different platforms whether the components actually run in different processes or as different threads within a single process. The initial reference implementation uses platform-specific IPC messaging, but several alternative message formats are explored to allow the portability of the applications to different platforms. These include Apache Thrift [6], JSON [7], MIME [8], as well as a custom SCAMPI defined XML schema.

3.3 SCAMPI Platform Services

The SCAMPI service platform provides **support for distributed service composition** and the creation of applications. The **platform services** consist of service-supporting functionality that is **generic** and can serve wide array of applications. Such generic functionality can be, for example, trust and security, social-awareness, service subscription management for remote and local execution, and incentive-supporting services, among others. Most importantly, the SCAMPI service layer understands the generic semantics of services such as their availability and whether a message indicates the seeking - or the provision - of a service.

The SCAMPI service platform receives **service requests and declarations** from the application components. The task of the platform is to use the service platform's state information to find the best possible provider for the sought services. When an application requests a service, the platform tries to map the request to service declarations that are announced locally, or seen from the other nodes. If matching service announcements are not seen, the request can be propagated further to other nodes as a SCAMPI object and a service subscription is maintained open to later return the response messages to the requesting application. Upon the arrival of response the service layer notifies the requesting application and passes the sought data to the application. When seeing a service announcement from the application or local service components, the service platform may proactively distribute the service to other nodes.

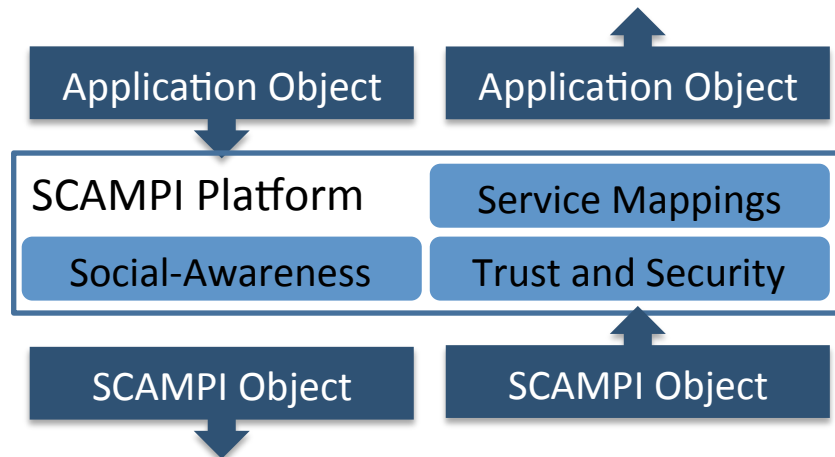


Figure 8: SCAMPI service platform.

The service layer maps the service objects to network **endpoint identifiers**, which can be used to reach the nodes that have either declared a SCAMPI service or are currently executing a service for this node. Each endpoint identifier can be related to a single or multiple nodes.

Generic service-supporting operations, for example, social awareness, security, and recommendation functionality, can be implemented as sort of filters that intelligently select EIDs that can be used for optimal service composition.

Following this approach the SCAMPI platform asks for a service from the opportunistic stack by using the following service commands.

- **seekService(OwnEID, remoteEID, SCAMPIObject);** a method to seek for a SCAMPI service. OwnEID identifies the seeking node and is used for response routing purposes to guide the sought service items back to seeking node. RemoteEID is the destination EID for the message, which can be a single node, or a multicast entity that identifies group of nodes, for example, all SCAMPI nodes. SCAMPIObject carries the service identifying information as well as the related service and security parameters.
- **provideService(OwnEID, remoteEID, SCAMPIObject);** a method to announce a service that is offered to local and remote applications. OwnEID identifies the providing node and is used for routing purposes to locate the provided service. RemoteEID and SCAMPIObject are as above.

3.4 Opportunistic Stack

The main responsibility of the opportunistic stack is to **create opportunities for communication** between the SCAMPI nodes. The communication stack supports node discovery, opportunistic messaging, and routing.

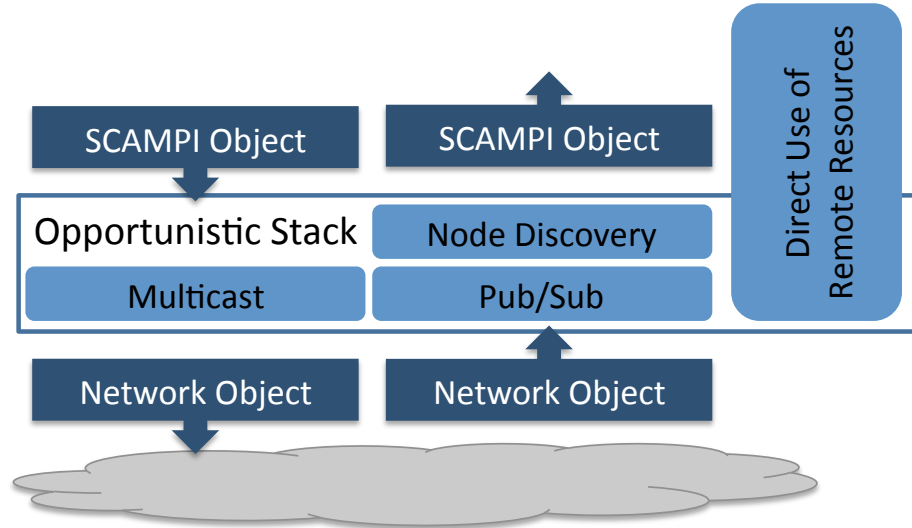


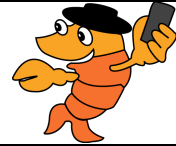
Figure 9: Opportunistic stack to provide SCAMPI support.

Because of the numerous challenges in opportunistic networking, as explained in Section 1.5, the communication stack needs to operate over **heterogeneous networking technologies**. For the more challenging scenarios with only intermittent connectivity, the DTN architecture and its Bundle Protocol are used for asynchronous messaging support. In more challenging scenarios, IP neighbor discovery (IPND) or multicast-DNS protocols provide mechanisms for node and service discovery. Device-to-device connections by using MANET protocols such as the Open Link State Routing protocol (OLSR), can support communications in more stable scenarios, or direct connections can be formed between neighboring peers. Independent of the chosen networking technology, the SCAMPI approach is to support interoperability by using well-documented protocols and include their functionality in the SCAMPI architecture according to published protocol specification. Examples of such protocols include DTN Bundle protocol [RFC5050], IP Neighbor Discovery (IPND) [18], as well as multicast-DNS (Bonjour) service discovery [19].

The opportunistic stack can provide communication service to applications, which allows them to **directly access remote resources**. In such cases the SCAMPI **platform operates as a facilitator**, and provides mechanisms from locating communication channels, such as direct device-to-device connections for media streaming. While data in such communication channels does not flow through the service platform, the meta-data that is included in service establishment messages helps the platform to keep track on the communication between the nodes.

3.5 Component Breakdown of Functional Architecture

This section discusses the functional architecture at the software component level. The discussion explains what parts of the architecture will run within same processes and which will be executed in different processes or requires own threads to execute tasks in parallel.



Main interfaces that enable communication and data exchange between functional components are shown to illustrate the flow of data and control between the components of the architecture and to further highlight which components will gain access to data handled by the platform.

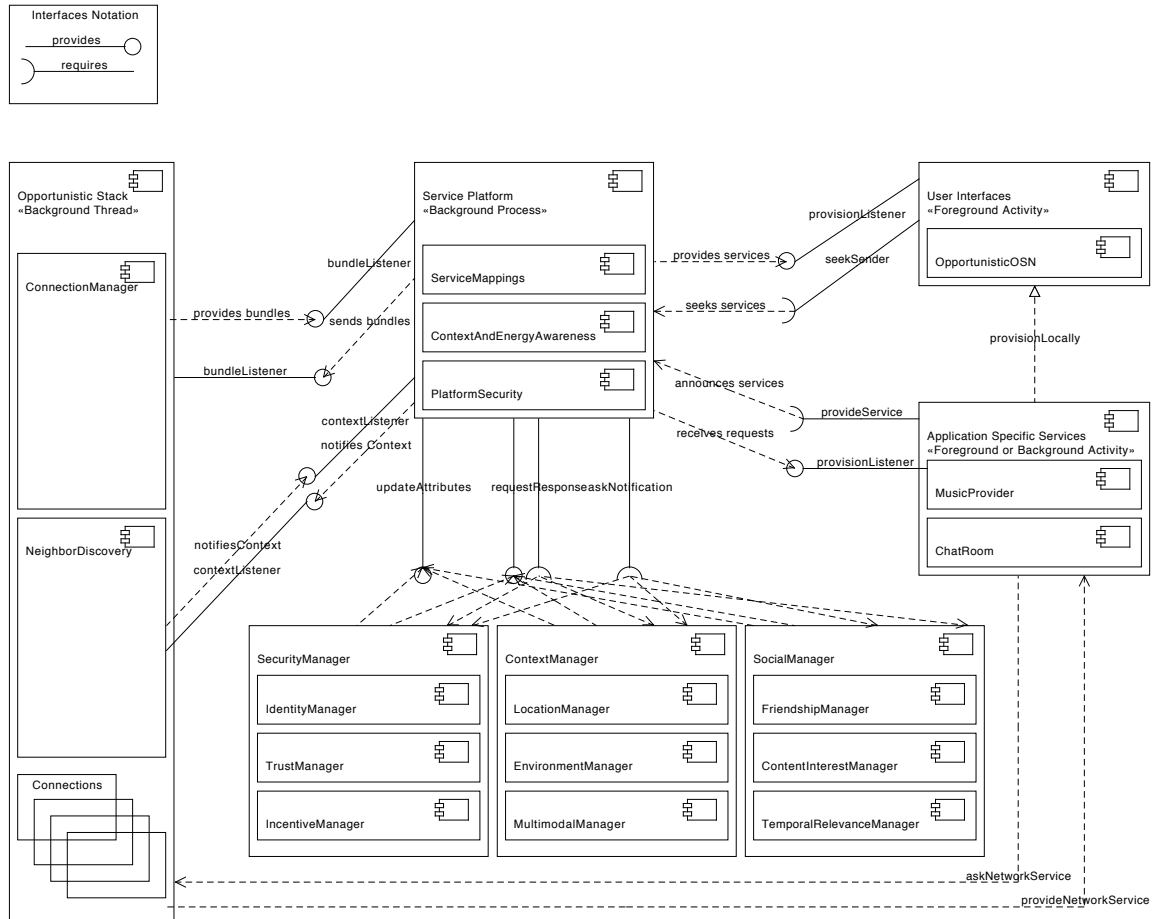


Figure 10: Functional breakdown of SCAMPI components.

On the left in Figure 10, the **opportunistic communication stack** runs in its own thread in the same background process as the SCAMPI platform. The rationale is that the platform must be able to execute at all times and the opportunistic stack must be able to manage contacts and send and receive messages in parallel with the operation of the SCAMPI service platform. The user interface runs as a foreground activity, meaning that a process for executing it is started when user interacts with the platform. The communication between the foreground activity and the platform components takes place through inter process communications (IPC) interfaces. The rationale for separating user interface to separate process is that the platform can independently execute in the background even when user interface does not exist. The execution of the background service can be of course interrupted if user so explicitly requires. Moreover, using simple socket-based IPC mechanisms allows implementing the service platform in manner that its function will be platform independent. As an example, an alternative choice could have been made to use Android interface definition language (AIDL) to provide IPC functionality, but this choice would have limited us to be able to execute the platform only on Android devices. The chosen socket API, however, exists for most of devices that SCAMPI project targets.

In the middle of the figure, the **service platform** component is responsible for functionality that is central to distributed service composition. The service mappings component keeps track on how each service component can be reached. Context and energy awareness and security manager components interact with several subsystems that keep track on state of the SCAMPI environment, including, social and context tracking and security requirements of the platform. The division of all this functionality into several manager components allows us to group functionality into clearly encapsulated entities. The service platform can then combine information from multiple managers to provide rich information for service composition.

On the right in the figure, user interfaces execute in their own process, as explained, to allow the platform to execute in the background even when the user exits the application. Several **application-specific service components** are also planned, such as a chatroom or an Internet radio, that will execute in different process than the service platform, similarly as user interface does. The rationale is that a lot of the functionality required to implement the end user applications needs logic that is not common to all services, but rather specific to each application. Therefore, such specific functionality should be separated from common platform services. The application specific components can run either in the foreground such as the chatroom interface that enables real-time chat or in the background such as Internet radio that provides streaming content.

3.6 SCAMPI Communication Model

The following explains four different simplified use cases where SCAMPI architecture is used to offer a component of service that can be used for composing complex services. Common to all use cases is that the **messages are transferred asynchronously and in a semantically self-contained manner** whenever possible **to meet the requirements imposed by the opportunistic environment**, that is, especially the **lack of stable communication paths**. The asynchronous messaging model with self-contained messages can be easily explained when considering email system as an example. Asynchronous messaging can be seen in traditional email application, where the application send the message, but does not receive confirmation about the reception. This feature removes the need for the recipient to be reachable at the time of sending the message, and therefore real-time messaging connection is not required between end points. When adapting the email system for DTNs [22], each email message is packaged into a single message unit, called bundle, which contains the message and all its headers. This is different to the Internet approach, where message can be divided into multiple smaller fragments in form of IP packets. The benefit of self-contained messages is that any intermediary that receives the packet is able to do application level decisions solely based on the message, for example, a gateway can immediately forward email to the Internet without waiting for further parts of the message to arrive. Further motivation for this messaging model can be found, for example, in DTN Bundle Protocol [5] or DTN Architecture Specification [21]. In cases where such a model is not considered sufficient, for example, with multimedia streaming, the **SCAMPI architecture can use also end-to-end connections between service components**. The following scenarios explain in detail the semantics of communication services used by the SCAMPI platform.

SCAMPI messaging gateway scenario illustrates a SCAMPI service on top of a uni-directional non-acknowledged messaging model. The service seeker sends application data, for example, a digital photograph, to a **gateway service** for delivery to the Internet. The seeker specifies that only services provided by trusted friends are relevant. The SCAMPI

message object carries information related to generic service composition such as the sought service and its trust-level. It does not, however, carry information that is specific to gateway service logic such as the address of the final destination of the message. Such specific information is only needed upon reaching the gateway application, which then reads it from the application data. SCAMPI trust services, common to all applications, block the message from propagating to nodes that have a trust-level below the required value. Moreover, the service platform guides the message towards possible service providers based on social context and inferred service reachability information such as its location.

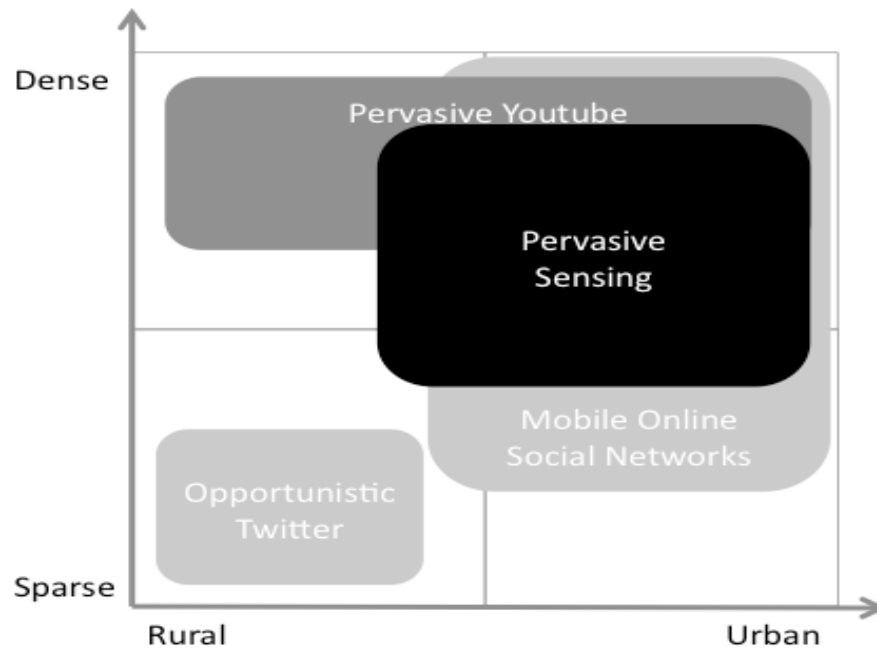


Figure 11: Simple message gateway scenario.

Content search service use case illustrates the content search service in the SCAMPI environment. The service seeker creates a **content search query** and asks SCAMPI to distribute this to a selected set of users, for example, all friends of the user. Searchers from other SCAMPI friends are evaluated and responses are sent back to the sender of the query. Search definition and evaluation takes place at the application layer, since it requires logic that is specific to the content search application in question. This logic can, for example, relate to the formulation of queries in SQL or REGEXP language or logic for feature-based extraction, etc. SCAMPI services take care that processing is carried out so that nodes conform to security attributes that are carried in the SCAMPI header. The security service is an example of functionality that can be a **generic service for number of applications** and it is thus independent of the content of the application object (image, music, text, etc.). The social-aware service is able to pick a set of network end-points that it has observed to be part of a friends' group, for example, by learning from the security module that the user can decrypt message from these users or from the location service that the users frequently co-locate with the user. Messages are then guided towards these endpoints. Upon receiving a message from the network, the SCAMPI platform carries out the security operations, notifies the platform services (for example, adding the trust-level of the sending endpoint), and delivers the application data to the intended application.

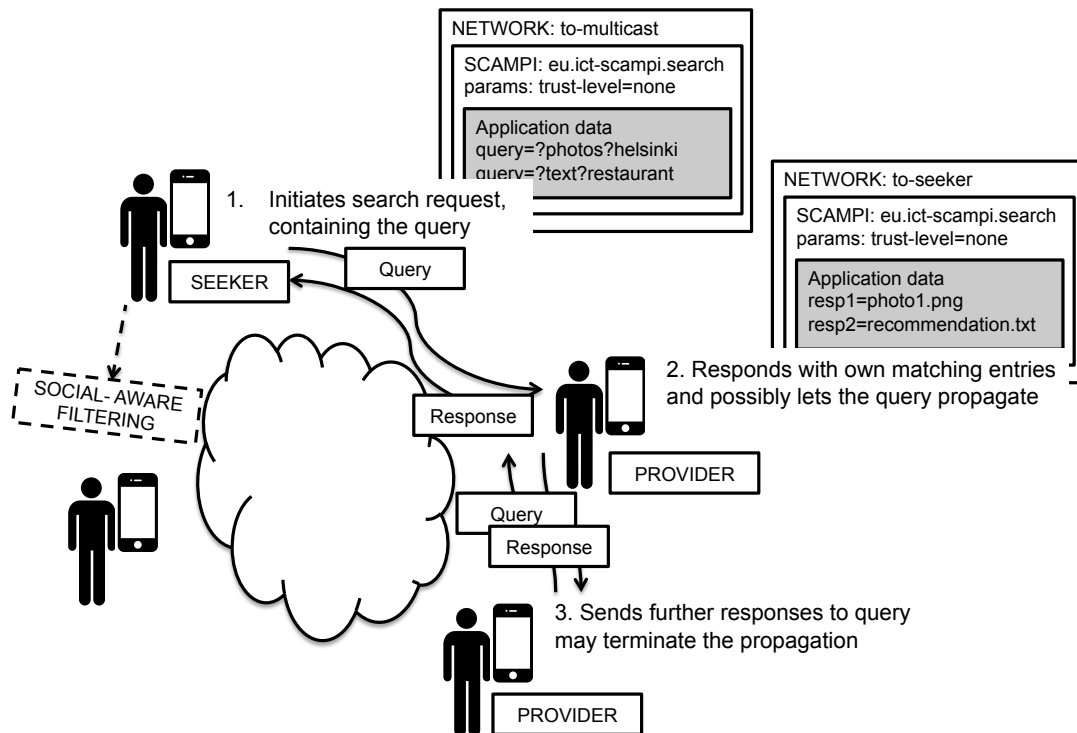


Figure 12: Content search scenario.

Opportunistic media streaming scenario illustrates use case where direct access to resources is needed. A provider declares a **content stream** to neighbors in direct contact. The neighbor can be either a one-hop neighbor, or a neighbor towards which MANET routing can be used to establish end-to-end communication. After getting information about the available service, the seeker can decide to use a streaming client to join the stream that was declared through a SCAMPI service. The stream service can be generic, and not specifically tied to SCAMPI platform. Thus, the service does not have to, for example, work on top of an asynchronous messaging model. In this case, the service invocation and composition is supported by the SCAMPI platform, but the actual data is streamed through direct access to the remote resource.

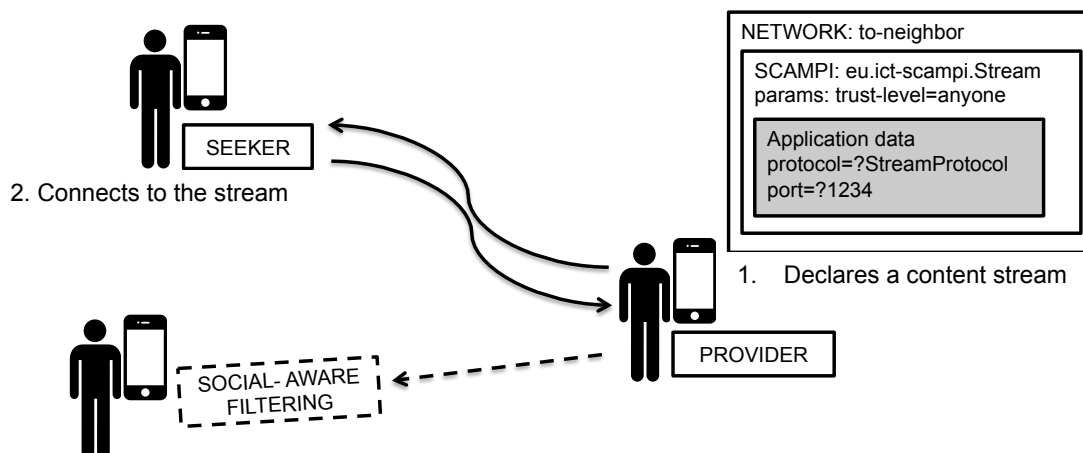


Figure 13: Opportunistic media streaming scenario.

Distributed service optimization scenario is an example of **replication and optimization** in the SCAMPI environment. Service replication can be used to increase the availability of the service when it becomes popular, or to optimize service placement in the network. The seeker initiates request to replicate a service, which can carry the code that implements the service logic together with a state (for example, application's data items) related to the request. The provider then starts the service in a SCAMPI service container and sends an acknowledgement to the seeker. The seeker can decide upon receiving acknowledgements that a sufficient number of service instances are running and decide to terminate the service.

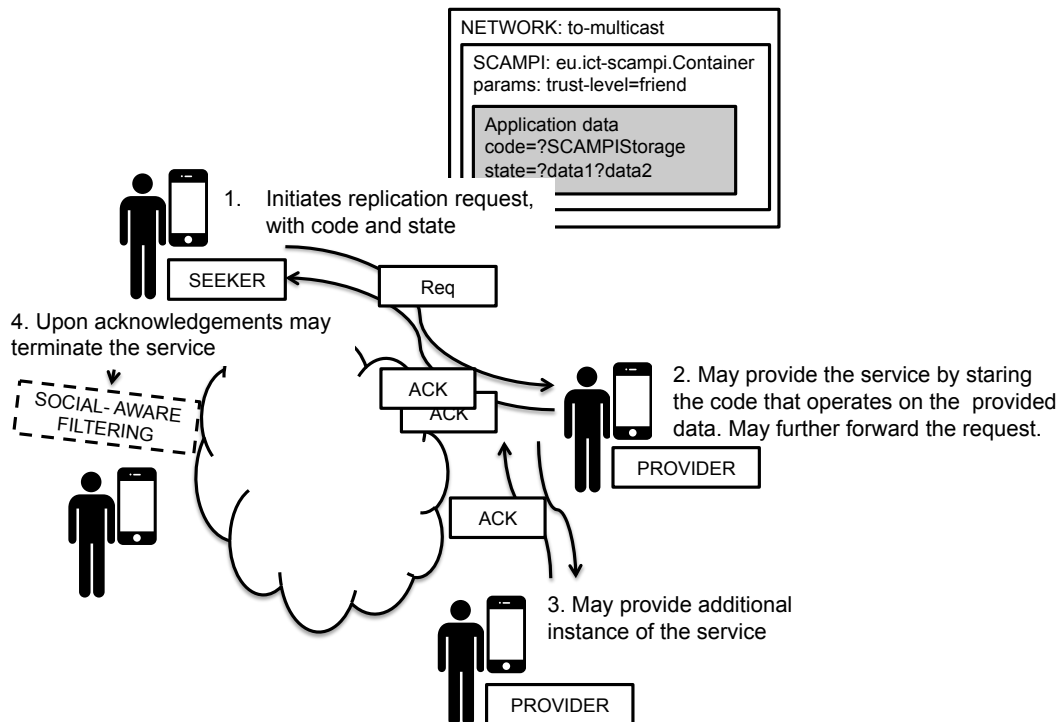


Figure 14: Distributed service optimization in the SCAMPI environment.

3.7 SCAMPI Security Model

The security services are in the core of the SCAMPI service platform. In order to work in most disconnected environments the security functionality must not rely on existence of connectivity to centralized security services. However, the platform can make use of security services offered by the existing state of the art social networking solutions. The following illustrates the use of this functionality by SCAMPI service platform in detail according to what has been evaluated to be beneficial for the platform. The presentation focuses on methods for establishing user identities, but the security APIs are designed and constantly refined to support other types of security attributes, these include, for example, trust and incentive based security providers that are developed within the SCAMPI project.

Social Managers discover and maintain knowledge of social connections between users. Such manager may, for example, define a user group based on a set of phone numbers from the user's phonebook and allow applications to address messages to that group. Further, the manager may provide information, such as routing hints, to other functions of the middleware.

SCAMPI router instance can have **multiple social managers**, each of which has knowledge gathered from separate sources. These instances are by default independent; however, it is possible to link the identity in one manager to an identity in another manager. For example, by linking the Facebook identifier of a user to their phone number, the Facebook contact graph can be used for routing messages destined to a specific phone number.

This obviously raises privacy issues as the SCAMPI router might have knowledge of multiple identifiers for the user through friendship managers, but the user might not want to expose all of those externally or may not want to expose the linking between them. For example, the user may have an anonymous Twitter account and wishes other users to be able to destine messages to the Twitter identity. Linking the anonymous Twitter identity to the user's Facebook identity would reveal the user's real name.

In general, the user does not need to expose any identities externally, simply by requesting random messages from nodes that it meets and checking locally whether the destinations on the received messages match any of its own identities. On the other hand, exposing an identity may allow routing mechanisms to deliver messages to the user more efficiently. This seems to give raise to privacy vs. efficiency tradeoff.

Facebook Manager simply exports the user's identity and a list of the user's friends from the Facebook. It would then identify messages destined to the user's Facebook identity and allow applications to destine messages to the Facebook identities of the user's friends. More advanced manager might build a graph of the social connections and then calculate routes on the graph which can then be used by routing modules when deciding which messages to forward over a contact.

Physical Contact Manager, besides importing explicitly known friendship information from a service such as Facebook, similar knowledge can be built from physical contact information. If each SCAMPI router has a unique, random identity, a physical contact manager can observe which identities it comes in contact frequently and assume the user has a social relationship with those identities.

Implicit social managers will also be an area of active research and they can operate, for example, using declarations that are based on radio co-location [9] or multimodal censing. Development of these components is done in interaction with work packages WP2 and WP3 and possible algorithms for deducing social context are improved throughout the project.

3.8 Using OAuth (Facebook, Twitter, etc.) for Identity Initialization

The following examples illustrate in more detail how the Facebook-based social manager can be used in SCAMPI to establish transitive trust. The first example explains how an OAuth-based authorization service provided by Facebook enables to authorize a user to access other users' friendship information. The second example explains how the same service APIs can be used to initialize identities, so that they can be used for social aware opportunistic forwarding mechanism used in SCAMPI.

The examples explain identity management in scenarios where user A is sending to user C, and A meets user B who is friend with user C and therefore a suitable candidate to relay the message to C.

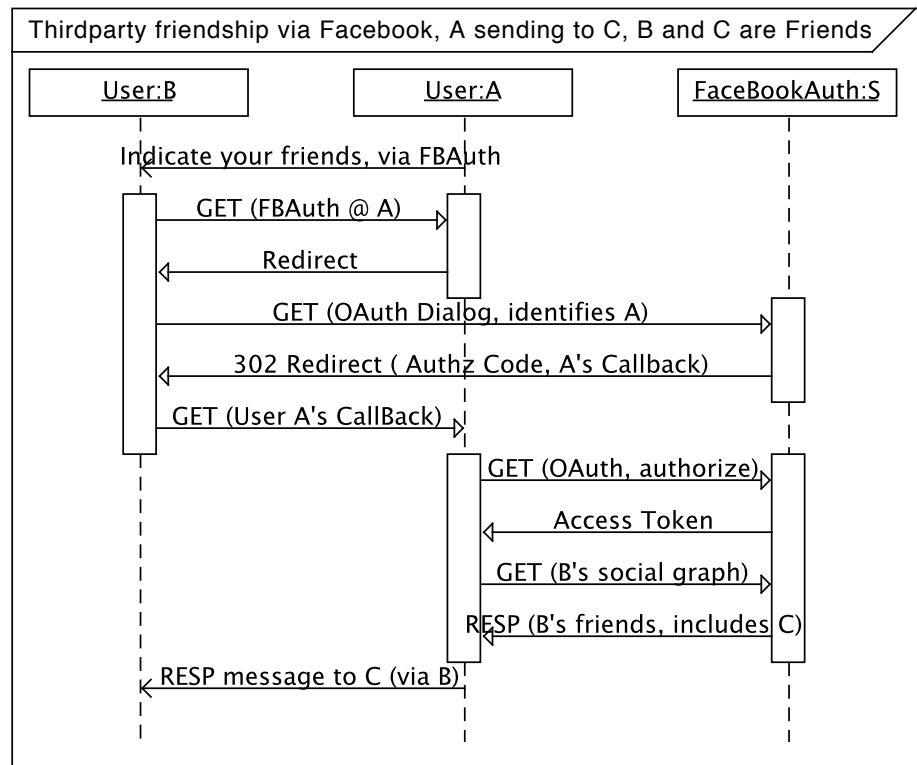


Figure 15: Using Facebook to establish identity and trust.

Figure 15 shows scenario where user A asks B to indicate it's friendship graph through a social context manager, which is in this example provided by Facebook, but similar APIs are provided, for example by Twitter and others. A detailed explanation of the authorization flow is provided by the Facebook Authentication API documentation [10]. After the B has authorized A to asks B's contacts from Facebook, the user A can fetch the list of contacts and see if the destination of the message that is being sent is in the list. If the destination appears in the list, A can request B to carry the message towards the destination.

The above approach has two main benefits. First, many of the users of social networking applications have accounts in one or several popular online social networks, such as Facebook or Twitter. These services collect and maintain information about the users' social network structures and provide a namespace where each user is uniquely identified with persistent identifier, that is, if C appears in A's network with certain identifier she appears in B's network with the exactly same identifier if B and C are friends. The second benefit is more practical: the most popular online social networks offer open and well established interfaces to query the information and even provide software libraries that allow straightforward integration of functionality to multiple platforms, such as Android or iPhone, that are relevant to SCAMPI.

However, the above approach has one major weakness considering opportunistic scenarios that are targeted in SCAMPI. This becomes obvious when looking at the protocol exchanges of the Facebook authentication service. In a challenging communication scenario, where neither or just one of the users A and B is able to connect to the Internet, the complete sequence of messages that is required for authorization becomes difficult or impossible to carry out. Therefore, the following illustrates how the same service provider can be used to initialize information in manner that is usable in opportunistic scenarios too.

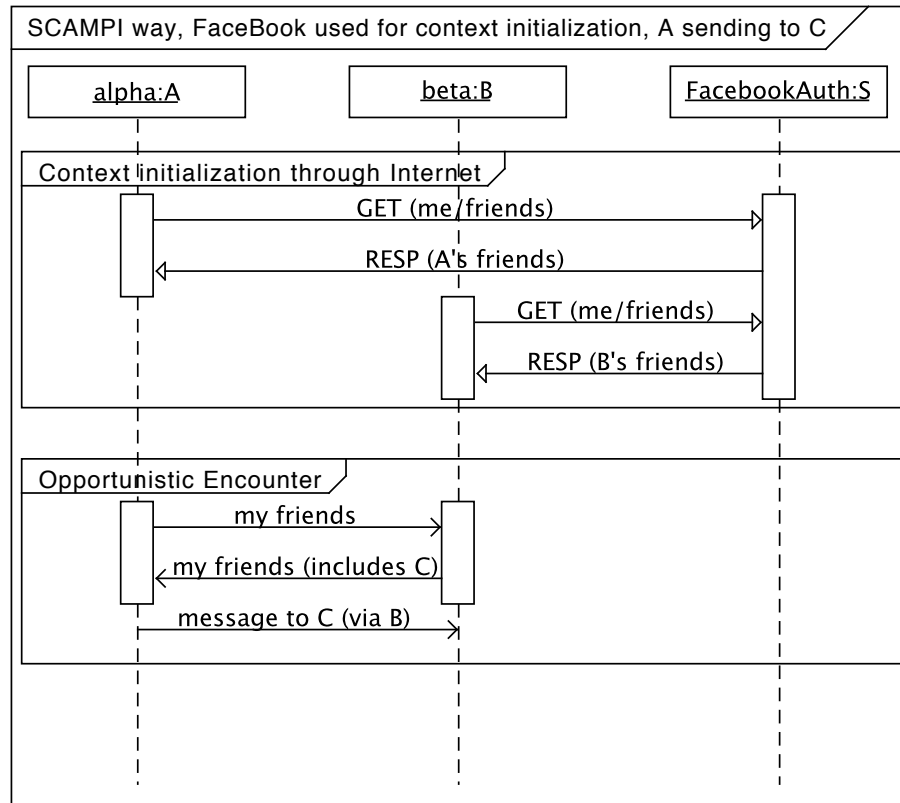
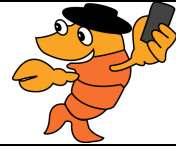


Figure 16: Initializing identity and trust for opportunistic encounters.

Figure 16 illustrates identity and social graph initialization by using Facebook authentication APIs. In the example, both users A and B initialize their friendship graphs from the Facebook service while they are connected to the Internet. This can take place, for example, in the morning while being connected at home. The social graph information is then stored on the users' devices and can be passed to other SCAMPI users upon encountering them. The proposed approach has at least two aspects to consider. When B declares her friends to A, it exposes some personal information about her social graph; however, the information is similarly exposed in Facebook too. If B does not like to expose this information, she can still participate in SCAMPI interactions, but forwarding performance can be worse due to a lack of information. Second, scalability would become an issue if complete friendship graphs would be further forwarded and maintained up-to-date throughout the network. In the above scheme, it is, however, enough to maintain each user's own graph and those of currently connected neighbors. The intersection of the user identifiers in these graphs can then serve as basis for two-hop routing algorithms that has been shown to be effective for message delivery in opportunistic networks. Sending out one's friendship graph can be seen as potential privacy issue, but experience from real life shows that people do declare openly their friendship information in online social networks such as Facebook. If user decides not to disclose friendship graph it can still participate to interactions, but does not receive preferred service based on friendship. This provides both incentives to reveal social context as well as compatibility towards users who are not willing to expose social graph information.

3.9 Platform-level Security and Privacy Considerations

The SCAMPI service platform enables producing, consuming, and storing user data on their devices. While it is clear that this can be seen as a threat to security, this is not regarded as significantly larger problem than in currently existing online social networks. Therefore, we see that for the initial phases of the architecture the following mechanisms that are used in currently existing solutions are applicable for our architecture.

Delivery of trusted binaries is supported by the software distribution and installation mechanisms used by the most widely used smartphone platforms and their application stores. The trust in these distribution channels is established by signing applications binaries by using a secret owned by a trusted source for binaries. Such a mechanism can be used to secure SCAMPI users against installation of unsecure or maliciously modified software components.

Accessing users' data based on explicit consent can be efficiently enforced with existing API tools in both online social networks and mobile platforms. For online social networks, good examples can be found in the authorization levels in Facebook that are enforced by the service provider to efficiently limit third party applications' access to personal user data. For mobile platforms, good examples can be found from Android mobile platform, which enables multitude of mechanisms of securing the user data starting from the consent about application's rights that are asked from user. This consent can, for example, require the user to allow application to access location of the user. The Android platform also enables developers to efficiently limit access to data that is stored on user's device by separating data storage on content providers that require application specific access rights. These examples highlight that some of the questions that need to be addressed in SCAMPI are already well addressed in existing implementations of related platforms. However, also new mechanisms need to be further developed to support similar operation outside the aforementioned platforms where such APIs might not exists. This is why the SCAMPI architecture demands that the core APIs need to be designed in platform independent manner. As a concrete example, if the security model would strictly rely on Android's content provider model the platform would be difficult to instantiate on pure Java or iPhone platform.

Accessing resources on users' device can be limited by similar platform security mechanisms as are used for limiting access to data. These mechanisms enforce the platform to ask for user's consent before allowing the installed application to access resources in user's devices. The access can be limited per resource, for example, allowing application to access location information, but not to allow the application to use device's network connectivity.

Research challenges for SCAMPI security solutions still remain, as the project targets significantly more challenging scenarios than the already existing solutions are providing. The research on these solutions is conducted in WP3 and their development is iteratively being carried in cooperation of the architecture implementation. Project deliverable D3.1 at M12 will contain more detailed analysis on the security models for the SCAMPI scenarios.

4 CONCLUSION AND NEXT STEPS

This deliverable first of all introduces to the core SCAMPI vision and distinguishing features in a broad strategic perspective. By identifying in which way the SCAMPI approach positions itself in the wide networking/service provisioning context also with respect to the state-of-the-art work, this document discusses the key technical and technological concepts at the core of the SCAMPI architectural foundation. This is then further articulated by presenting and discussing four main strategic usage scenarios for the SCAMPI technology.

Furthermore, this document describes the initial foundation of the SCAMPI architecture that comprises three functional layers:

- At the top of the architecture are the **applications**, which act as an interface between the user and the SCAMPI platform.
- In the middle is the SCAMPI **service platform**, which provides service-supporting mechanisms for distributed service composition.
- At the bottom are protocol stacks for various **opportunistic networking** implementations.

The rationale for these 3 layers has been documented and their functionality and purpose have been explained through examples of use cases. Furthermore, the currently identified messages and parameters to be passed between the layers have been defined and the structure of the SCAMPI Objects has been presented. The first realization of the SCAMPI platform is ongoing and implements this layered approach.

It is planned that the ongoing research and performance measurements carried out in the other work packages influence the architecture in an iterative way. WP2 studies the basic models and mechanisms (algorithms) necessary for predicting the performance and realizing and sharing services in opportunistic, pervasive networks, and WP3 conducts research about supporting mechanisms and elements for enabling and securing these services. This combination both advances the research on opportunistic networking adding mechanisms that exploit the pervasive characteristic for supporting shared services, as well as introduces new models and algorithms for the services themselves. Similarly, WP4 on empirical performance evaluation of SCAMPI services feeds back inputs to the architecture design from its experimentally-driven validation activities.

We evaluate the SCAMPI approach with both trace-driven simulations and experiments involving real users early in the project so that we can (1) control the risk, (2) make the appropriate design choice in order to maximize the success of the project and assess the feasibility of the overall approach, and (3) provide some early evaluation results on the design. Tools that are currently used include:

- The ONE Simulator (<http://www.netlab.tkk.fi/tutkimus/dtn/theone/>), which has been developed and is maintained by AALTO, and which is used as reference simulator.
- The Huggle platform (<http://code.google.com/p/huggle/>), with four partners of the original Huggle project, three of them deeply involved in the Huggle code development and maintenance.
- The Podnet platform (<http://podnet.ee.ethz.ch/>), which has been developed and is maintained by ETH.

- The AALTO platform provides RFC5050-compatible DTN implementation for Android and iPhone which has been developed and is maintained by AALTO.

This process is used for iterative revisions of the architecture (and multiple prototyping cycles) each one offering increasingly better functionality and performance, as well as to alternate design phases with experimentation phases. As recently demonstrated by successful projects in the field (e.g. Haggie, ANA), significant advances are possible only through integrating theoretical *and* experimental results derived from real testbeds.

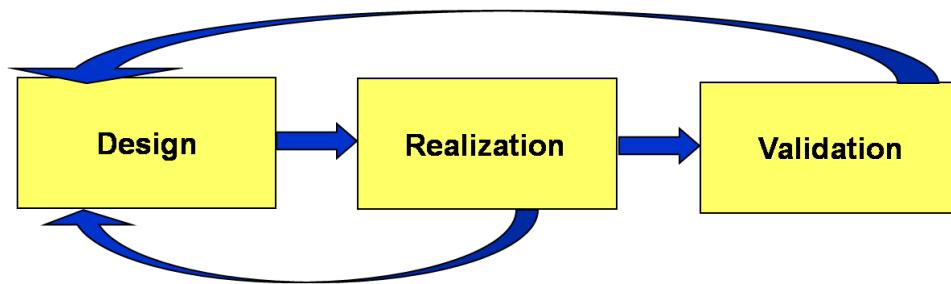


Figure 17: Incremental, feedback based approach of SCAMPI.

The iterative approach described in Figure 17 has been used from the very beginning of the project. During the first year, the main goal of the project has been to develop the models for opportunistic resource sharing, according to its long-term research perspective on the Future Internet. To properly do so, we have also collected data through experiments, in order to better understand user requirements and thus better align the SCAMPI resource sharing models with users' needs and expectations. The main approach we are currently using during the remainder of the first year is collecting and analyzing detailed traces of users interaction patterns and social structures in Online Social Networks. This is carried out in Task 4.1. It is worth noting that the overall SCAMPI objective (that is, a platform for opportunistic resource sharing in mobile environments) is a significant innovation, as nothing of this kind is currently elsewhere available. Even mobile social networks are mostly limited to straightforward applets that provide single-hop connectivity to Online Social Networks running on the legacy Internet, while opportunistic resource sharing is not considered at all. Analysing the current interaction patterns in Online Social Networks provides significant information to grasp user requirements during the initial stage of the project (currently done in Task 1.1). This overcomes the fact that it is impossible at the project onset to already collect requirements in the very same networking environment that SCAMPI will enable; as to date this is simply not existent. By analysing how people interact with each other, and how they form social structures in OSNs, we expect to achieve quantitative evidence of the opportunities for resource sharing that users could exploit once the SCAMPI platform will be in place. Thus, this provides an initial quantitative baseline to start validating the models used in the SCAMPI opportunistic resource sharing platform, and designing the related technical solutions.

As the project progresses, it will be possible to test the SCAMPI functionality at increasing scales (in Tasks 4.2 and 4.3).

To this end, the plan for experimentation of the project includes the use of a pilot application that will be distributed to users. This application will benefit from an increasing set of SCAMPI features that will be integrated as they become available. The application will benefit from an increasing set of SCAMPI features that will be integrated as they become

available (note that such features will also be refined dynamically based on the results achieved during the project lifetime). To facilitate testing at large scales, the application will be made available through distribution platforms such as Apple Store or Android Market (depending on the platform that will be identified as the most suitable). The application will also enable us to complement and refine the collection and understanding of user requirements (in Task 1.1) based on data from Online Social Networks carried out in the first year - as outlined in Figure 18.

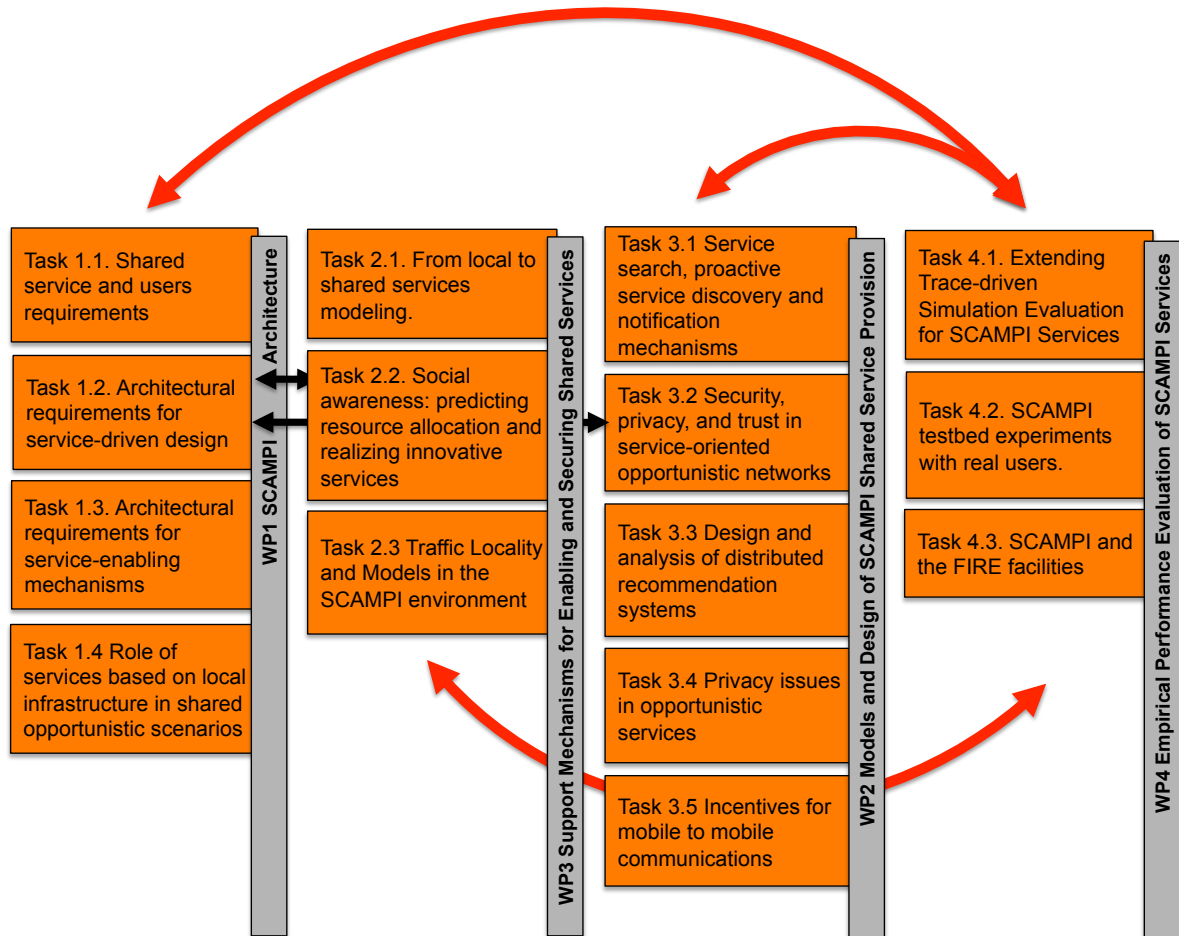



Figure 18: Interaction between technical work packages in the SCAMPI project.

Based on this approach, as far as the collection of user requirements and experimentations leading to a refining of the architecture are concerned, we list the following phases:


- By Month 6: The detailed plan for collecting user requirements throughout the project duration has been put in place; an initial plan for testing SCAMPI in the large scale is also in place and discussions with cooperation with Future Internet Research and Experimentation (FIRE) framework has started including bilateral talks on experimentation with SmartSantander testbed. Two user requirement gathering tools that work as Facebook application have been developed and are collecting data to check the assumptions on the structure of social networks and help fix the relevant parameters.

v.1.0	<i>D1.1: System Architecture Specification</i>	 34
-------	--	---

- By Month 12: Data from Online Social Networks will be analysed; user requirements motivating the SCAMPI models will be derived; a detailed plan for testing SCAMPI in the large scale (which will include an initial design of the reference application) will be in place; user requirements will have been collected; the SCAMPI concept will have been defined and the application (functionalities and developer) will have been identified
- During Year 2: An initial version of the application will be distributed, and data will be collected from its use; user requirements and SCAMPI features will be refined according to the collected data; an application will be sent to one of the FIRE-facilities (e.g., SmartSantander) to make a practical experimentation on the facilities
- By Month 18: An early prototype of this application will be available for beta testing of the user interface and usability aspects;
- By Month 24: The application will be at a stage where it can be distributed to real users through an application store
- During Year 3: The final version of the application will be distributed, and data will be gathered for the final design of the SCAMPI platform

REFERENCES

1. Haggie Architecture: <http://code.google.com/p/haggie/>
2. DTN Architecture: <http://tools.ietf.org/html/rfc4838>
3. PodNet Architecture: <http://podnet.ee.ethz.ch/>
4. DodWAN Architecture: <http://www-valoria.univ-ubs.fr/CASA/DoDWAN/>
5. DTN Bundle Protocol: <http://tools.ietf.org/html/rfc5050>
6. Apache Thrift Framework: <http://incubator.apache.org/thrift/>
7. JavaScript Object Notation (JSON): <http://tools.ietf.org/html/rfc4627>
8. Multipurpose Internet Mail Extensions (MIME): <http://tools.ietf.org/html/rfc2045>
9. Sacha Trifunovic, Carlos Anastasiades and Franck Legendre, “**Social Trust in Opportunistic Networks**”, Second IEEE International Workshop on Network Science For Communication Networks (NetSciCom'10), San Diego, CA, USA, March 2010.
10. FaceBook Open Authentication API:
<https://developers.facebook.com/docs/authentication/>
11. Twitter Open Authentication API: http://dev.twitter.com/pages/basic_to_oauth
12. Publications by the SCAMPI project: <http://www.ict-scampi.eu/publications/>
13. Andrea Passarella, Marco Conti, Eleonora Borgia, Mohan Kumar, “**Performance Evaluation of Service Execution in Opportunistic Computing**”, The 13-th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems, ACM MSWiM 2010.
14. Opportunistic Twitter
15. Mikko Pitkänen, Teemu Kärkkäinen, and Jörg Ott, “**Opportunistic Web Access via WLAN Hotspots**”, Proceedings of the 8th Annual IEEE International Conference on Pervasive Computing and Communications (PerCom), Mannheim, Germany, March 2010.
16. Esa Hyytiä, Jorma Virtamo, Pasi Lassila, Jussi Kangasharju, Jörg Ott, “**When Does Content Float? Characterizing Availability of Anchored Information in Opportunistic Content Sharing**”, Proceedings of IEEE INFOCOM, April 2011.
17. Theus Hossmann, Franck Legendre, Thrasyvoulos Spyropoulos, George Nomikos, “**Stumbl: Using Facebook to Collect Rich Datasets for Opportunistic Networking Research**”, Proceedings of IEEE AOC Workshop 2011, Lucca, Italy.
18. Specification of IP Neighbor Discovery (IPND): <http://tools.ietf.org/html/draft-irtf-dtnrg-ipnd-01>
19. Specification of Multicast DNS Service Discovery Protocol (Bonjour):
<http://files.multicastdns.org/draft-cheshire-dnsext-multicastdns.txt>
20. Autonomous Networking Architecture project: <http://www.ana-project.org/web/>

v.1.0	<i>D1.1: System Architecture Specification</i>	 36
-------	--	---

21. RFC 4838, Delay-Tolerant Networking Architecture:

<http://www.faqs.org/rfcs/rfc4838.html>

22. Tuomo Hyyryläinen, Teemu Kärkkäinen, Cheng Luo, Valdas Jaspertas, Jouni Karvo, Jörg Ott, **“Opportunistic Email Distribution and Access in Challenged Heterogeneous Environments”**, Demo at the Second ACM MobiCom Workshop on Challenged Networks (CHANTS), Montreal, September 2007.