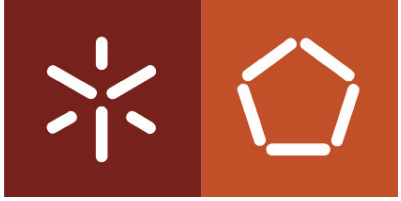
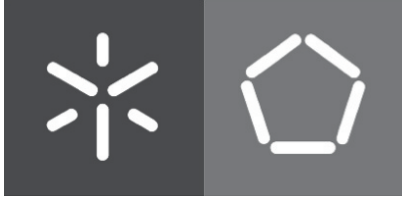


Universidade do Minho
Escola de Engenharia

Rui Filipe Pedro Quelhas

Improving Opportunistic Communications with Social Context

Outubro de 2011



Universidade do Minho
Escola de Engenharia

Rui Filipe Pedro Quelhas

Improving Opportunistic Communications with Social Context

Dissertação de Mestrado
Mestrado em Engenharia Informática

Trabalho efectuado sob a orientação de
Professor Doutor Joaquim Melo Henriques Macedo
Professor Doutor António Luís Duarte Costa

Outubro de 2011

DECLARAÇÃO

Nome

Rui Filipe Pedro Quelhas

Endereço Electrónico: rfpquelhas@gmail.com **Telefone:** 910236439

Número do Bilhete de Identidade: 12572710

Título dissertação ■ / tese □

Improving Opportunistic Communications with Social Context

Orientador(es):

Professor Doutor Joaquim Macedo e Professor Doutor António Costa

Ano de conclusão: 2011

Designação do Mestrado:

Mestrado em Engenharia Informática

Universidade do Minho, 31 / 10 / 2011

Assinatura: Rui Filipe Pedro Quelhas

The Internet lives where anyone can access it.

Vinton Cerf

Acknowledgements

First of all, I want to thank both my supervisors, Professors Joaquim Macedo and António Costa, for their guidance throughout the last year, all the brainstorms they kindly promoted and insights about the research process that have helped a lot a novice like me. They introduced me to this wonderful world of delay-tolerant and opportunistic networks which proved to be very challenging but also extremely rewarding during all this year-long process.

Next, I should also thank all the folks at the Delay Tolerant Networking Research Group (DTNRG) mailing list, that spent some of their precious time helping me out with some concerns and questions that resulted from my research. A special word to Jörg Ott, that despite his busy schedule as the head of the group himself, was most of the times available to answer some of my foolish doubts. Thanks also to Lloyd Wood and Frederic Guidec, for providing me with their different visions, respectively about some problems that people do not usually talk about at the group and some other alternatives to the work being produced by it. Another special word, this time to Hervé Ntareme, who was kind enough to publish some code that became very useful during the development of my work.

Last, but not least, a word to all my friends and relatives, particularly those that put up with me during the last year (I will not mention names because they were many and I do not want to forget anyone), my brother Nuno, my parents Isabel and Aurélio that always acknowledged my effort and agreed with the fact that I should put all my focus on this work.

Resumo

As redes oportunistas ou tolerantes a atrasos surgem como um conceito promissor na busca pela Internet do futuro, propondo superar as limitações de ambientes móveis com topologias dinâmicas, onde o comportamento dos nós na rede origina atrasos e interrupções na comunicação. Baseiam-se numa lógica onde algumas suposições erradas, tomadas pelos modelos tradicionais TCP/IP são consideradas caraterísticas e não exceções na comunicação. Para tal, introduz um novo modelo, *store-and-forward*, baseado na troca de mensagens, definindo que estas devem ser armazenadas pelos nós quando necessário e expedidas durante oportunidades de contato com outros nós. No entanto, o modelo assume diferentes variações ao nível da arquitetura de rede e consequentemente várias estratégias de difusão, incapazes de suportarem diferentes aplicações e cenários de utilização.

Este trabalho baseia-se no conceito de Pocket Switched Networks (PSNs)—ambiente formado por dispositivos de rede transportados por pessoas—para propor a SociAL-based OppOrtunistic Networking (SALOON) *framework* baseada no Bundle Protocol, integrada em sistemas móveis, capaz de promover uma convergência do grafo de representação dos contatos entre os dispositivos na rede e uma abstração do grafo social de cada pessoa que os transporta. Assume-se que esta ferramenta possa funcionar como plataforma de teste para o desenvolvimento não só de estratégias de difusão oportunista sensíveis ao contexto social mas também de novas aplicações capazes de potenciar este tipo de comunicação. O objetivo é disponibilizar um sistema de referência, no qual o trabalho relacionado com computação oportunista possa ser feito de uma forma uniforme por toda a comunidade.

Palavras-chave: contexto social, redes oportunistas, computação móvel.

Abstract

Opportunistic or Delay-tolerant Networks (DTNs) appeared as a promising concept towards what could be the future Internet. They pretend to overcome limitations faced by mobile environments with dynamic topologies, where the behavior of the network nodes leads to communication delay and disruptions. Following a rationale that undermines some wrong key assumptions made by traditional TCP/IP models, and considering them as features, not as exceptions, they introduce a novel message communication model, *store-and-forward*, on which nodes store those messages as needed, forwarding them later, when contact opportunities arise. However, it still does not gather widespread acceptance, because it fits on different architectural proposals, not defining communication strategies adaptable to different scenarios nor proper support for the development of heterogeneous applications.

This work builds on the concept of Pocket Switched Networks (PSNs)—an environment formed by nodes consisting of devices carried by persons—to propose the SociAL-based OppOrtunistic Networking (SALOON) framework, based on the Bundle Protocol and deployed on mobile computing systems, capable of promoting a convergence of the graph representing the physical encounters between those devices and an abstraction of the social graph of each user carrying it. It is intended to work as a testbed for the development of novel context-aware, particularly social-based, opportunistic routing protocols as well as to provide application support on top of a standard delay-tolerant architecture. This framework aims to encourage the community to develop future research around opportunistic computing in a more uniform fashion.

Keywords: social context, opportunistic networks, mobile computing.

Contents

List of Figures

List of Tables

Acronyms

ADU application data unit.

API Application Programming Interface.

BP Bundle Protocol.

CAR Context-aware Adaptive Routing.

CCSDS Consultative Committee for Space Data Systems.

CHE Computing for Human Experience.

CLA Convergence Layer Adapter.

CSCL Computer Supported Collaborative Learning.

CSCW Computer Supported Collaborative Working.

DARPA Defense Advanced Research Projects Agency.

DNS Domain Name System.

DO Data Object.

DoD US Department of Defense.

DoDWAN Document Dissemination in Wireless Ad hoc Networks.

DTN Delay-tolerant Network.

DTNRG Delay Tolerant Networking Research Group.

EASE Exponential Age SEarch.

EID end-point identifier.

EWMA Exponentially Weighted Moving Average.

GPS Global Positioning System.

HiBOp History Based Opportunistic Routing.

HIS Haggle Information Space.

HT History Table.

HTTP Hypertext Transfer Protocol.

IETF Internet Engineering Task Force.

IP Internet Protocol.

IPN Interplanetary Internet.

IPND IP Neighbor Discovery.

IPNRG Interplanetary Internet Research Group.

IRTF Internet Research Task Force.

ISO International Standards Organization.

IT Identity Table.

JPL Jet Propulsion Laboratory.

LTP Licklider Transmission Protocol.

MANET Mobile Ad-hoc Network.

MoSoSo Mobile Social Software.

MSCA Mobile Social Computing Application.

MV Meet and Visit.

MVC Model–view–controller.

OSN Online Social Network.

P2P Peer-to-peer.

PRoPHET Probabilistic Routing Protocol using History of Encounters and Transitivity.

Propicman Probabilistic Routing Protocol for Intermittently Connected Mobile Ad hoc Networks.

PSN Pocket Switched Network.

RSVP Répondez S'il Vous Plaît.

RTT round-trip time.

S/MIME Secure/Multipurpose Internet Mail Extensions.

SALOON SociAL-based OppOrtunistic Networking.

SDK Software Development Kit.

SDNV Self-Delimiting Numeric Value.

SIP Session Initiation Protocol.

SMS Short Message Service.

SSO Single-Sign-On.

TCP Transmission Control Protocol.

TCPCL TCP Convergence Layer.

UDP User Datagram Protocol.

URI Uniform Resource Identifier.

URL Uniform Resource Locator.

USB Universal Serial Bus.

VANET Vehicular Ad-hoc Network.

WCC weakly connected component.

Chapter 1

Introduction

Opportunistic or Delay-tolerant Networks introduce a new model for communication in mobile environments, which are most of the times challenged by restrictions such delay and disruptions caused by the dynamic nature of their topologies. This model is based on an operational behavior where the network nodes exchange messages between, and store messages from, each other in an opportunistic fashion, i.e. when a contact opportunity between them arises. Pocket Switched Networks (PSNs) constitute a specific subset of these environments, where nodes correspond to devices that are actually carried by persons, and communication is thereby driven by human behavior. That behavior is most of the times tied to some specific social parameters that determine a "relationship" between the users forming the network. Such a context can be made available by external sources like Online Social Networks (OSNs) where nowadays, most people have their own particular social profile which is kept associated with their friend' or acquaintances' profile and any piece of online content generated between them. This proves to be an important metric to determine the strength of social ties, and consequently with whom each user tend to contact more or less frequently.

1.1 Research Problem

Despite the plethora of challenged environments envisaged as proper Delay-tolerant Network (DTN) application scenarios, the reference architectural model [11] for these type of communications is still somewhat tied to its historical context, specifically to deep-space networks (which spawned it in the first place). This makes it not perfectly suitable to other particular scenarios such as terrestrial mobile networks, where the communication is most of the times purely opportunistic, and cannot be scheduled, as it happens, for instance, with satellites. However, considering that terrestrial communications are usually tied to human interaction, and that humans by themselves, are most of the time, predictable, there are some strategies that should allow to improve those communications based prior behavior. Such behavior is consistently characterized by social properties that can be evaluated and measured through some specific channels and tools.

This kind of human-based communication in an opportunistic environment is referred to by some authors as PSNs, which have been subject to a great research effort during the last years. However, these have resulted in a segmentation of different proposals, particularly regarding the underlying network architecture, leading to a state where the core research issues are not being dealt with efficiently. The work on opportunistic routing or forwarding strategies, and application support for these kind communications, is a clear example of everything that is being done wrong in this area. Lately, we are witnessing the appearance of diverse "solutions" for these problems, which seem to be pretty limited in terms of practical results, particularly, because of the absence of proper datasets and testing frameworks. These tools are actually available out there, however, they lack support for the underlying social properties described above, thus not providing the human-based perspective needed to properly address this very special kind of networks.

1.2 Motivations and Contributions

To overcome some of the issues described above, the author proposes a novel social-based opportunistic communications framework meant to enable a convergence of the research work over PSNs. The main goal is to provide a way to develop adequate communication services that spawn from routing, forwarding and management protocols into full-fledged DTN-enabled applications, following some state-of-the-art standards. The framework is built over both existing and original middleware components packaged in a multi-layered flexible system capable of being deployed on particular mobile environments, where communication is usually achieved through specific devices, in this case, smartphones.

This document intends to introduce the contributions produced throughout the development process of the framework that try to answer some of the most common issues found during the research phase. The work builds on the main objective of improving opportunistic communications with social context indicators, which it does with the creation of the framework itself. This leads consequently to a set of more particular contributions that comprise the following:

- Evaluation of a proper DTN architecture and respective implementation, given the available technology to produce the desired environment. On-demand enhancement of any specific feature lacking in existing middleware components attached to the system;
- The cross-layer convergence of context indicators. From the properties of opportunistic contacts at a networking level up to a well-defined characterization of possible relationships underlying those contacts provided at an application level;
- A new metric to measure the importance of each contact, called *affinity* which determines the level of that convergence (*physical* and *social*) and allows a better classification of the nodes in the network;
- A novel social-layer component supported by plugins attached to specific context providers, particularly OSNs, with the implementation of

a sample proof-of-concept.

- Evaluation of the context indicators available, regarding their influence on the calculation of the *affinity* property;

In general, the framework is intended to provide a standard-based functionality that allows:

- providing mechanisms based on the available context for the improvement of opportunistic context-based networking protocols or applications; and

- opening a window for the creation of proper datasets, that in turn, allow a supported research over this topic.

1.3 Outline

This document is organized as follows: Chapters 2 and 3 describe the state-of-the-art in social networking (specially its mobile counterpart, tied to a particular context-awareness) and opportunistic or delay-tolerant communications. The former introduces some background information and available technology on both concepts as the latter provides an analysis of some groundbreaking approaches presented by other authors which also establish a convergence between them; Chapter 4 presents and describes the architecture of the SociAL-based OppOrtunistic Networking (SALOON) framework, a software tool that leverages social-based context information in a testbed for the development of novel opportunistic protocols, services and applications; Chapters 5, 6 and 7 provide a bottom-up analysis of all components comprised by SALOON, starting from the available DTN-enabled middleware component and the communication between nodes up to the interfaces connected to external context providers; Chapter 8 states the main conclusions and remarks achieved with this work by summarizing its contributions and promoting additional work for the future.

Chapter 2

Background

This chapter provides an analysis of some core background concepts, presented throughout the literature, that provide a foundation for most of the contributions proposed with this work. It begins by describing the evolution of Online Social Networks (OSNs), from the ground up, into a prevailing model as seen by the status quo, based on mobile communication software immersed in the local context. On the other hand, it also provides an introduction to the concepts of Opportunistic and Delay-tolerant Networks (DTNs), highlighting proper terminology, possible application scenarios and the resulting research for suitable network architectural proposals. Finally, it presents a survey on the available and most widespread implementations of those architectures. Some, already providing their own particular vision about what they consider the role of context information and social behavior in the opportunistic environments they are built upon.

2.1 Mobile Social Networking

Software for social networking was first considered as a set of computer tools and applications that supported group interaction, having its roots on previous developments in Computer Supported Collaborative Working (CSCW) and Computer Supported Collaborative Learning (CSCL). This definition is extended by Lugano et al. that describe the status-quo with social services

able to support interconnections among networked mobile users [47]. They also state their core objectives as to enhance social relationships by improving support to usual face-to-face interactions and by establishing a way to enrich our network of friends or acquaintances. The emergence of wireless communication technologies like WiFi and Bluetooth brought to a need for new mobility approaches promising to unlock a whole new world of opportunities on multiple research and market fields. Mobile devices became key outlets for social software, and a mobile revolution was already on its way.

As show in Figure 2.1 this revolution lead to the appearance of hybrid and purely mobile social services. The former, trying to roughly offer customized access to their web counterparts through portable devices as the latter actually started to provide additional functionality based on the device's features, striving to create a new kind of social experience. Also the application paradigm evolved from communities based on chat-rooms and forums (global in nature) to even more individualized and profile-based websites and communities of interest (way more local in nature). Although some older design concerns may still prevail, most of social software is clearly converging towards pureplay mobile models, even some considered to fit an hybrid scope are being extended to fulfill new mobility requirements.

The original purpose of mobile-phones was to connect people with each other, so they represent a deeply social device by nature. Besides being pervasively connected to a global network, they are also pervasively connected to our persons. This is currently paving the way on social networks' progress, where beyond transferring the current trends from online communities there's now a need to achieve new functionality by means of mobility and the context surrounding relationships between users themselves, and their devices.

2.1.1 Context-awareness

There are many definitions of context in the literature and generally they tend to aim at specifying it in a very large sense. However, lately we are actually acknowledging some efforts to tailor them for mobile and ad-hoc settings, narrowing it to a much smaller scope. A definition fitting this extent

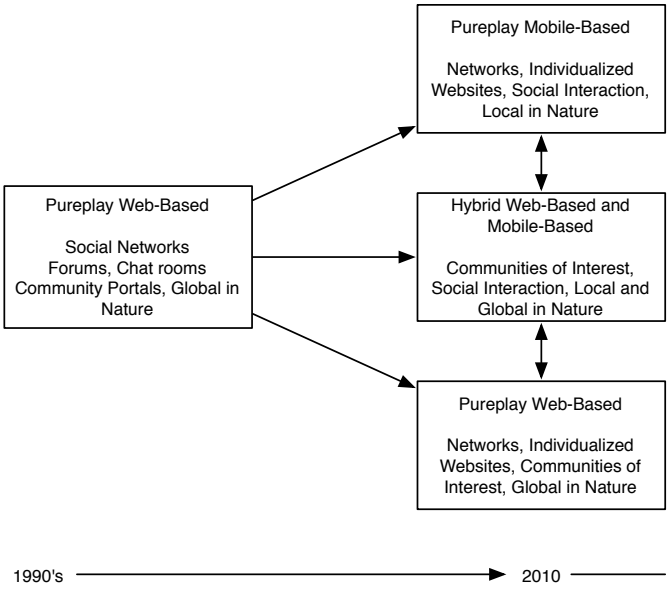


Figure 2.1: The evolution of social networking services over the past two decades. Figure adapted from Ziv and Mulloth [84].

is presented by Th. Eugster et al., that consider distributed objects running on small independent communication-enabled mobile devices [78]. These objects are referred to as peers, which context is characterized by a set of parameters, external to the object implementation, that might influence its behavior. They divide context information in two main categories:

Individual context Represents a peer's egocentric view of the world, gathering information about the object's own directly accessible environment. Such context parameters typically consist on device state information (e.g. storage capacity, available memory and energetic constraints) as well as some physical values measured by it in the surrounding environment (e.g. location, temperature).

Social context Represents the peer's awareness to the existence of other peers, gathering information from them. By detecting their presence, we can estimate their proximity and relevance, thus exchange information based on these or other parameters using profile data or even refer to services they might offer.

This kind of analysis suits well from a network standpoint, but it falls short to describe how this information can be leveraged by applications to augment an ubiquitous environment. Dey [19] refers to context as any information that can be used to characterize the situation on an entity, being it a person, a place or an object that is considered relevant to the interaction between a user and an application, including the user and application themselves. Beach et al. [3] emphasize the value of personal-social-profile and external sensing besides the already mentioned device-related data. They build on the ability to retrieve users' individual information such as interests, preferences as well as friendships and relationships, thus fusing it with device sensing data obtained with built-in accelerometers, microphones, cameras and digital compasses alongside information collected from external sensor networks (e.g. embedded in local smart spaces) encompassing parameters like temperature, humidity, infrared, audio and video. The key idea is to combine these three data streams for a full-fledged context-aware adaptation, which leads them to envision some application examples such as taste-tailored juke-boxes and video screens — like the ones on the movie Minority Report[1].

By merging and managing all these available data, we are allowed to answer questions like *a*) *Where do I stand?*; *b*) *Am I actually available?*; and *c*) *Who is around? And how close?*. Services and applications that leverage this kind of knowledge fit in the concept of Mobile Social Software (MoSoSo) [47, 72], with some of them being described at section 2.1.2.

2.1.2 Applications and Services

From the author's perspective, there are two different categories of applications in the MoSoSo realm. Those built over consumer/market needs and those that resulted from scientific processes within the academia. The former usually have a focus on entertainment and user experience, thus focusing on high-level lifestyle and leisure features. The latter tend to act as support tools to aid the deployment of novel ubiquitous concepts by allowing to collect experimental data about the users' physical connections, mobility patterns

[1] `http://www.imdb.com/title/tt0181689/`

and about the communication restrictions that characterize these pervasive environments. However, they usually have in common the fact of being based on the location-awareness paradigm, and allow for some kind of interaction with nearby peers.

Commercial-oriented Mobile Social Computing Applications (MSCAs) promote functionality around homophilic matchmaking (meeting people that share our interests), nearby friend tracking, instant messaging and content sharing (photos, videos), and other serendipitous interactions. These evolved from early special-purpose, local-connected electronic devices and services [38, 77] into mobile-phone software, working via Short Message Service (SMS) like Dodgeball [20, 84], or more recently, via internet, such as Foursquare[2], Gowalla[3], Instagram[4], Yobongo[5], among many others [43]. In other words, they are moving into a preset that is intrinsically tied to a need for global connectivity (all-around internet access), be it through cellular data connections or WiFi.

On the other hand, research-based projects on MoSoSo build on these concepts of serendipity[6] and homophily[7] to promote an analysis of how mobile and implicit (local) social networks are established and maintained at a lower level. They do this by means of internet-agnostic services that communicate with each other via local available network interfaces and using ad-hoc multi-hop and Peer-to-peer (P2P) protocols like Bluetooth. On top of this, they tend to provide more straightforward interaction functionality, at least compared with commercial MSCAs. Popular examples of these kind of applications and respective projects are Social Serendipity [22], Whozthat? [2] and VENETA [81]. These are also mobile-phone applications and they follow a model where the physical contacts between devices using the service are registered, and each instance has its own social profile—created internally or through external sources like OSNs—used to promote user matchmaking or

[2] `http://foursquare.com`
[3] `http://gowalla.com`
[4] `http://instagr.am`
[5] `http://yobongo.com`
[6] `http://en.wikipedia.org/wiki/Serendipity`
[7] `http://en.wikipedia.org/wiki/Homophily`

other kind of social recommendations.

2.1.3 Disruptive Visions

As we can learn with the facts presented at subsection 2.1.2, the development of MoSoSo draws on a similarly optimistic vision of our location and nearby related or compatible people. The design of these systems privileges a type of urban experience that is centered on an idealized category of users, what ends up being a too-much restricted vision of the social concept.

Thom-Santelli [79] proposes some alternative strategies, not entirely based on the concepts of serendipity and homophily, in order to widen this limited scope and promote more heterogeneous experiences. She builds on some disruptive guidelines such as *a*) moving beyond the typical user, i.e. considering the outliers or even the skepticals and resisters to the technology (e.g. employee reviews against customer reviews in a restaurant); *b*) drawing from critical practice and tactical intervention, i.e. not favoring only spaces of consumption and entertainment (almost like the concept of gentrification[8]), defying the accepted meanings and providing a mechanism to expose social, political and cultural questions that usual engineering approaches might miss (e.g. Homeless Vehicle[9,10], a shopping-cart providing space for homeless people to sleep, designed with a rocket shape to draw people's attention); *c*) leveraging homophily for diversity, i.e. snowball sampling[11] approaches where the technological trust builds around an initial community resulting later in a diverse seed group (e.g. Area's Immediate Reading[12] project, where people carry portable air quality monitoring device for visualizing area's pollutant levels); and *d*) designing for multiple interpretations, i.e. supporting usability by not constraining use where the systems ultimate meaning and purpose are open to user interpretation (e.g. TXTmob[13], a group messaging application first designed for activists to plan protests being eventually used

[8] `http://en.wikipedia.org/wiki/Gentrification`
[9] `http://www.homelessvehicle.com`
[10] `http://www.designboom.com/eng/archi/wodiczko.html`
[11] `http://en.wikipedia.org/wiki/Snowball_sampling`
[12] `http://www.pm-air.net`
[13] `http://www.appliedautonomy.com/txtmob.html`

by authorities to promote its safety or even by regular users for day-to-day communication).

Likewise, Sheth [71] acknowledges a need for a breakthrough of social computing on the ubiquitous web, heralding the concept of Computing for Human Experience (CHE). This vision is built on a suite of technologies that serves, assists and cooperates with humans in a nondestructively and unobtrusively manner. He believes in the future emergence of technology-rich human surroundings for more sophisticated and seamless human-interactions compared to today's precursors such as automotive accident avoidance systems.

He describes a scenario where a farmer is observing an unfamiliar disease on his crop and goes seeking information on how to manage it. A CHE system would analyze a picture of the crop (provided by the farmer) and correlate contextual background domain knowledge, including local weather and soil conditions (by location-awareness). It would then broadcast the information on forums and OSNs, particularly to interested or specialized individuals, tracking responses, prioritizing authoritative sources, pulling additional information, filtering spam and presenting summaries of crowd-sourced data. Finally, directing the information to the farmer, prompting for progress feedback and informing the community when the data is not accurate enough, or wrong [71].

Mobile networks continue to force humans to communicate in an overly engineering-centric fashion. For instance, if we want to exchange an email with someone who happens to be nearby us, it is likely the communication will be performed by some servers spread across the world, what seems to be a burden and certainly results in an unnecessary waste of network resources. Also these architectures continue to face mobility and related disconnections as a challenge instead of an opportunity to communicate [13]. However, it seems clear that we are closer to a shift of the status-quo, as we observe an increasingly interest on novel network architectures based on opportunistic or delay-tolerant communications.

2.2 Opportunistic and Delay-tolerant Networks

Opportunistic networks are emerging as a promising concept towards the next-generation internet. Like Mobile Ad-hoc Networks (MANETs), they establish that existing nodes are enabled to communicate with each other even if a route connecting them never exists. However they also diverge from them by not supposing the nodes need to possess or acquire any knowledge about the network topology [62]. Delay-tolerant Networks (DTNs) [11] assume an architecture consisting of network portions located apart from each other forming the so-called DTN regions, operating internally by their own protocol stack (possibly with Internet-like connectivity) and having only occasional external communication opportunities. The regions are bounded by gateways in charge of providing interconnection among them by means of a novel overlay component called the Bundle Protocol (BP) [70]. Despite some background and architectural differences, both these proposals consider a communication model based on messages that are not forwarded on-the-fly because intermediate nodes store them when no forwarding opportunity exists, a paradigm commonly referred to as *store-carry-and-forward* [13].

2.2.1 Taxonomy and Terminology

The concept of DTN was first considered for a deep-space communication architecture within the research on Interplanetary Internet (IPN). It was primarily targeted at tolerating long delays and predictably interrupted communications over long distances. By the time its first specifications were being published, there were already some scientific community members suggesting the intention to extend the IPN work to other types of setups, including terrestrial wireless networks [24]. On the other hand, the concept of Opportunistic network appeared actually as an extension to this type of terrestrial environments, specifically MANETs.

However, in the literature, the terms Opportunistic Networks and DTNs are used interchangeably. There is no commonly agreed-upon terminology nor clear separation of concepts between the two. Pelusi et al. [62] explore more carefully this situation and propose some means to distinguish them,

concluding that one happens be a subset of the other. They advocate that both differ on some conceptual visions, particularly: DTNs' regions can form internet-like topologies where routes are typically computed via legacy techniques that consider the link unavailability as another component of link cost; on the other hand, opportunistic networks promote the absence of a priori knowledge about network topology and that routes should be computed at each hop while a packet is forwarded. They state that Opportunistic Network constitutes a more general definition that encompasses the one of DTNs.

Nonetheless, some authors consider the possibility of DTN regions being themselves some sort of opportunistic network, and given the initial scope of DTNs it is easy to face, for instance, the contact between two satellites, as being opportunistic itself. So, by this standpoint, opportunistic networks could also be seen as a subset of a DTN [45]. Also, there are those who refer to DTN solely as the research community responsible for studying delay-tolerant and opportunistic communications, specifically represented by the Internet Research Task Force (IRTF) group known as Delay Tolerant Networking Research Group (DTNRG) [16]. Throughout the rest of this document, the author uses both terms indefinitely despite a very specific application scenario being used as base for its work.

2.2.2 Challenged Environments

As a pioneer on assessing the DTN architecture additional value, Fall [23] enumerates some key assumptions incorrectly made by usual TCP/IP models namely *a*) the existence of end-to-end paths between data source nodes and its destination peer(s); *b*) reasonable and manageable communication round-trip times; and *c*) small packet drop probability. Most of these, among others, are actually violated by a class of challenged networks, fitting some situations already described, that includes the following:

Terrestrial mobile networks Environments that may become unexpectedly partitioned due to node mobility or changes in signal strength, maybe on a periodic and predictable manner (e.g. a commuter bus traveling from place to place providing nearby clients communication

with distant parties it might further visit).

Exotic media networks High latency systems that may suffer predictable interruption and outage due to environmental conditions and use of *store-and-forward* services (e.g. deep space satellite, underwater communications).

Military ad-hoc networks Systems operating on hostile environments affected by mobility issues and intentional jamming where data traffic must compete for bandwidth with high priority services (e.g. communication between platoon soldiers within the theater of operations during a war).

Sensor/actuator networks Environments characterized by a tremendous scale (possibly thousands of millions) of devices with extremely limited power, memory and CPU capability where communication is often scheduled to conserve power and nodes are addressed only in aggregate.

Some of these systems are characterized by very significant link delay, others by non existence of end-to-end routing paths and even lack of continuous power or memory resources. They present substantial operational and performance challenges to approaches based on TCP/IP. Previous solutions based on extensions developed for this model to support link-repair mechanisms for end-to-end reliability or by using internet-proxy attachments, also did not seem to solve all the issues described. The former because it still required the use of Internet Protocol (IP) specifically with Transmission Control Protocol (TCP)—that by nature, do not deal well with delays and disruptions (specially due to overheads on connection establishment)—and the latter because it only works as bridge between challenged networks and connected ones (internet) not providing any specific tool to improve the data traffic within them. Hence, Fall argues that to mitigate these issues and to achieve interoperability between the diverse networks, a novel approach was necessary and so he suggests the use of a general purpose message-oriented reliable overlay architecture providing the service semantics of asynchronous message delivery [23].

2.2.3 Networking Proposals

DTNs were first envisioned by Vint Cerf and a group of scientists from NASA's Jet Propulsion Laboratory (JPL) which work (started in the early 1990's) gave birth to the IPN initiative that resulted later in the formation of the Interplanetary Internet Research Group (IPNRG), within the IRTF. Considering some of the already mentioned issues (such as low power, asymmetric bandwidth, high delays and bit-error rate) applicable in deep-space communication environments (satellites and spacecrafts) the group came to the conclusion that simply extending the Internet protocol suite was just not feasible. So they propose a new approach, calling it *bundling*, which builds a store-and-forward overlay network above the lower layers of underlying networks [49]. Later, people like Kevin Fall started to investigate how could the IPN architecture apply to other situations where communication was also subject to delays and disruptions, leading the group to publish a draft describing the IPN architecture as fitting on a more broad concept of DTNs [10]. This novel vision marked the beginning of the transformation process from IPNRG to what is known today as DTNRG.

In 2004, the US Department of Defense (DoD), under the Defense Advanced Research Projects Agency (DARPA) issued a call for proposals in "disruption-tolerant networking" focusing more on other types of disruptions than high-delays, particularly assuming a military character, such as radio shadowing or frequent passage in and out of base station range [25, 83]. Despite the paradigm shift within the IRTF, research on deep-space and interplanetary communications promoted by the IPN initiative is still very active with International Standards Organization (ISO) specifiations being defined by the Consultative Committee for Space Data Systems (CCSDS), the same applies to DARPA [83]. So, there are yet several different but overlapping groups woking toward the common goal of developing DTN protocols. Figure 2.2 tries to illustrate how all these research projects co-exist.

During the first years of its existence, the DTNRG worked towards a more generalizing architecture, refining the initial versions and publishing their findings with RFC 4838 [11]. Since then, the group is working mainly

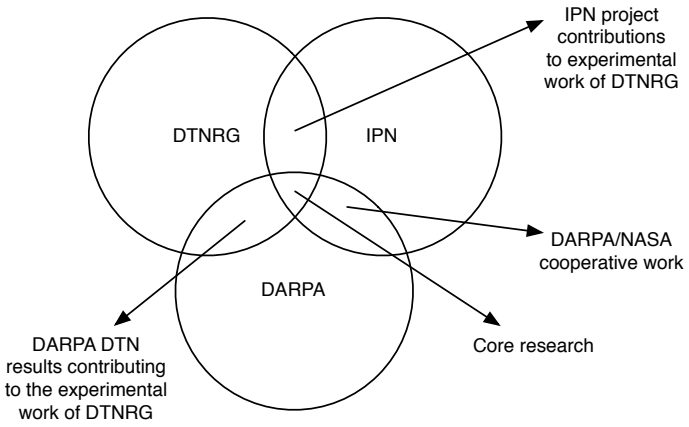


Figure 2.2: Several different organizations with overlapping research work on DTNs. Figure adapted from the one presented by Farrell et al. [25].

over two protocols [25, 49], specifically *a*) the Bundle Protocol (BP), with a specification—RFC 5050 [70]—describing the end-to-end protocol and abstract service description for exchanging messages (as *bundles*) in DTNs; and *b*) the Licklider Transmission Protocol (LTP), a point-to-point protocol providing reliability over links characterized by extremely long round-trip times (RTTs)—typically for satellite communications. Since deep-space and military environments are not within the scope of this work, the author chooses to not elaborate much more on proposals tied to this type of systems such as the ones resulting from DARPA, CCSDS alongside the work performed over the LTP specification. Rather focusing specially on BP and its more broad application scenario coverage and tailored conformance to the DTN architecture, as seen by the DTNRG.

The *Bundle Protocol* runs as an overlay agent on top of the Transport Layer (Figure 2.3), by means of a local *convergence layer* whose design matches the underlying network conditions and protocols [82], in order to provide application abstraction over the identified target environments' networking issues. It provides means to cope with intermittent connectivity, an ability to take advantage of scheduled, predicted and opportunistic contacts between the network nodes, and late binding of overlay end-point identifiers (EIDs) to convergence layer specific addresses (e.g. IP) employing per-

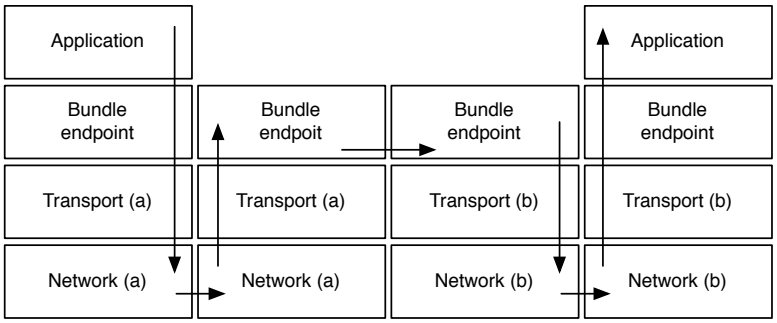


Figure 2.3: Bundle Protocol as an overlay component. Figure adapted from the one presented by Farrell et al. [25].

sistent storage to better handle disruptions and perform store-and-forward functions [49]. The overlay includes namely *a*) transfer or reliable delivery responsibility; *b*) optional end-to-end acknowledgment; *c*) diagnostic and management features; and *d*) flexible naming schemes, based on Uniform Resource Identifiers (URIs) for the late binding encapsulation of various different schemes in the same overall naming syntax. BP is agnostic to existing communication layers and focus itself on virtual message forwarding rather than packet-switching [49]. It only supports internetworking indirectly, by means of a plethora of convergence layers and the application of flexible naming-resolution schemes. The control of mappings, propagation of routing information, discovery of nodes and application identifiers must be accomplished by mechanisms provided from such layers [82]. To perform this way, it attaches itself to any traditional networking stack through specific adapters for each convergence layer, as illustrated by Figure 2.4.

Devices implementing BP are called DTN nodes and they can run DTN-enabled applications that are modeled around sending and delivering application data units (ADUs). This format is of arbitrary length (but subject to implementation limitations), the network might or might not preserve their relative order and likewise, they might or might not be marked to allow fragmentation. Typically, ADUs are sent or delivered to applications in its complete form and the BP transforms them into one or more protocol data units called bundles that are then forwarded by DTN nodes, targeting

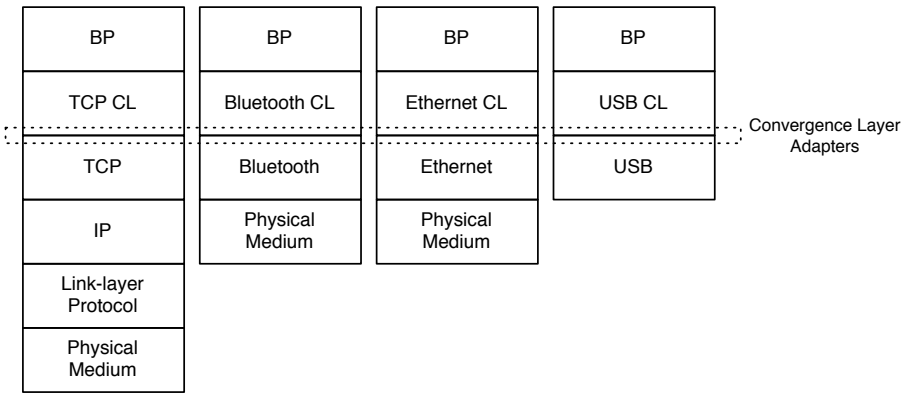


Figure 2.4: Examples of Bundle Protocol-enabled network stacks.

the destination given the respective EID. This purely consists on a network-centric perspective.

In an opportunistic routing environment like a terrestrial mobile ad-hoc DTN, forwarding decisions are not taken upon arrival of bundles but rather when one or more contacts become available via the outgoing interface(s). This concept determines the basic rule to implement the already mentioned store-and-forward mechanisms on each node of the network. The strategy consists roughly in a process where each contact provides information to the routing logic, if it is actually available and is capable of delivering the message—bundle or ADU(s)—to another node, be it the final destination, or a closer intermediate one. Besides that, the contact might also provide additional routing metrics and information about the bundles it contains itself, all of which alongside the data available in the header of the bundles in transit are extremely influential on forwarding (which bundles to transmit) and scheduling (by which order) decisions. To perform this extended logic, DTN nodes need to implement a more advanced routing architecture, possibly based on the conceptual model illustrated by Figure 2.5.

Despite the efforts to standardize it, the architecture was still closely tied to deep-space communications and thus not fitting to well other heterogeneous environments. This lead some skeptics within the research community to study and develop distinct solutions, particularly based in data-centric models, where messages are forwarded based on their content and not their

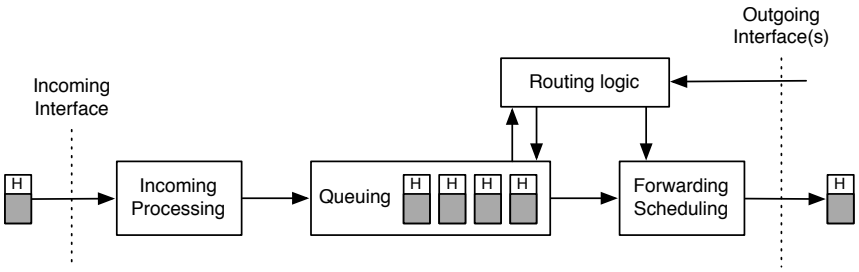


Figure 2.5: DTN conceptual routing model. Figure adapted from the one presented by Ott and Pitkänen [61].

destination. Several projects emerged promoting an abstraction over the infrastructure-related issues and focusing more on patterns observed in alternative challenged environments (specially terrestrial mobile networks) that were more intrinsic to the communication between nodes. These were essentially around mobility models, forwarding schemes, data-dissemination and security. However, the majority of these projects mainly aimed to solve one or few specific issues [15].

An exception, from this standpoint, is Haggle[14] [69, 75, 76]. Haggle is a framework that builds over the concept of Pocket Switched Networks (PSNs) where users (humans) carry mobile devices, moving between several locations as a part of a normal schedule, whereas in some situations they spend some time on connected places where they can communicate with other nodes via the internet (the so-called *islands of connectivity*). However, it specially considers users to also move occasionally within wireless range of other devices (either stationary or carried by other users) and are able to exchange data directly with them. The Haggle approach is thus more oriented to the human way of communicating (specifically focusing on community behavior), rather that related to the technological aspect of the communication [15]. Its design was guided by a set of interrelated principles which the authors [69] believed were fundamental implications of the situation faced by users in this kinds of environments.

Most of these, were grass-roots DTN, but a few, particularly related to the exposure of content to applications, were introduced to facilitate the

[14] `http://www.haggleproject.org/`

novel data-centric operation. For its distinct vision, Haggle describes a layerless architecture by seamlessly integrating a general application framework with the network protocols themselves, unlike the traditional stack model of the BP. This architecture was first conceived comprising three internal modules—delivery, user data and protocols—supported by another in charge of resource management. As shown in Figure 2.6, later, these components were redesigned into a more general purpose system where all modules are managed through a core control component and have access to data within an abstraction referred to as Haggle Information Space (HIS). Applications can inject new data into the HIS in the form of context information, register interest in incoming content or search the HIS for it.

The *control module* maps the core processing operations of the framework, handling the HIS data flow and providing external management Application Programming Interfaces (APIs). It works as an interface to other two additional components, the *forwarding module* and the *classifier*. The former performs neighbor discovery using the available network interfaces and connects to neighbors identified as potential next-hops. The latter evaluates incoming data from established connections and distributes it for security check and for actions performed by the *control module*. These components operate on top of a set of underlying managers (Figure 2.7), each one responsible for a set of specific features such as *a*) data encapsulation; *b*) naming schemes; *c*) content discovery; *d*) protocol selection; *e*) routing or forwarding; *f*) resource management; and *g*) security. Managers communicate with each other via publish-subscribe, using Data Objects (DOs) that specify the desirable intent, input data available and output data expected.

Beyond the architectural distinctions, there is also some terminology that varies between the BP and Haggle, for instance, the former defines the message type as being a possibly complete ADUs that could be fragmented into one or more protocol messages, which are referred to as bundles [11], while the latter establishes ADUs as being comprised of elementary DOs. Despite the fact that Haggle as been widely used as a testbed for opportunistic networking experiments, the reference DTN architecture and the BP are closer to become an Internet Engineering Task Force (IETF) standard and were cho-

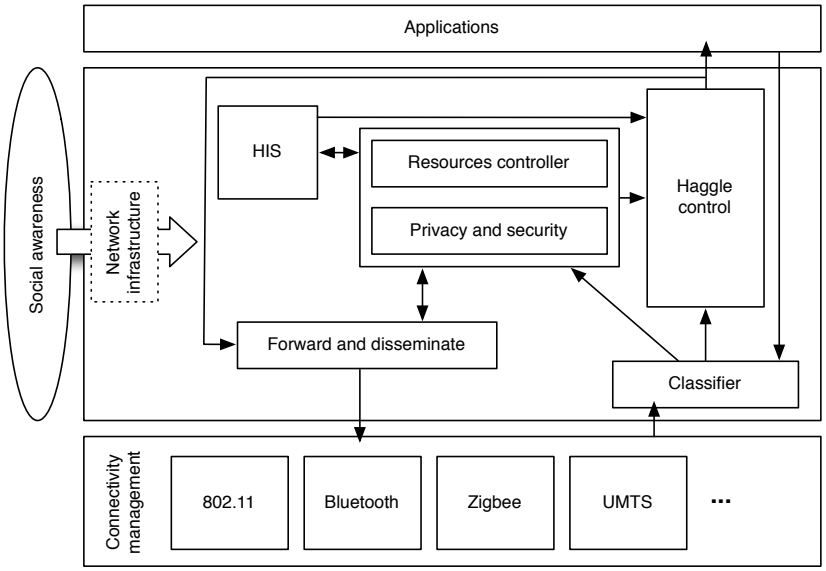


Figure 2.6: Haggle top-level architecture. Figure adapted from the one presented by Conti et al. [15].

sen as the basis of the SCAMPI[15] project that is considered to be Haggle's successor. These differences are also noticeable on the various implementations of each model, specifically, in terms of support for both the underlying networking processes (such as routing and forwarding) and the upper-level applications.

2.3 Middleware and Application Support

As stated in section 2.2, the deployment of DTN architectural implementations only makes sense if those are somehow integrated with legacy networking components to match the underlying communication processes. These implementations operate as middleware between the communication agents provided by the lower layers in the network stack of a specific device, and the applications, running on that device, that leverage this networking functionality. Also, following the same described line, those implementations spawn from two distinct approaches, *network-centric* and *data-centric* models.

[15] `http://www.ict-scampi.eu/`

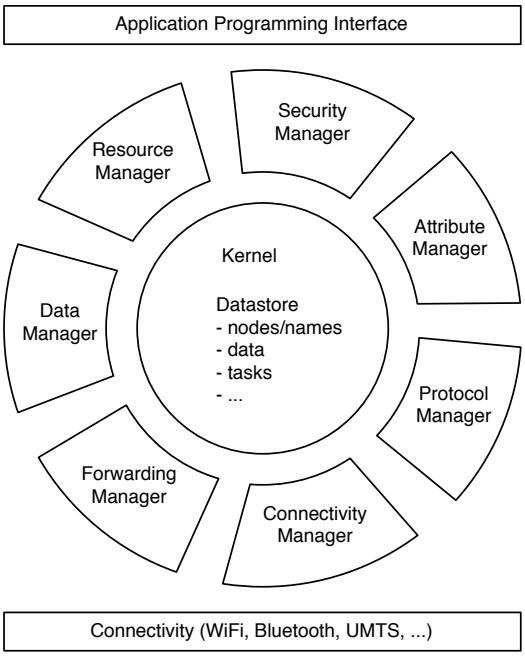


Figure 2.7: Haggle internal components. Figure adapted from the ones presented by Conti et al. [15] and Su et al. [76].

2.3.1 Network-centric Implementations

This category encompasses all the implementations that describe an opportunistic network architecture where messages are forwarded based on their final destination. It is the approach followed by the BP specification [70] which describes and end-to-end connection-oriented model representing a channel between two endpoints (identified by their respective EID). Probably due to the fact that there is a tangible specification available, this model is used in a considerable number of implementations.

The one that serves as main reference is DTN2[16], particularly because it is maintained by the DTNRG members themselves and follows closely the ongoing work on the formal BP specification. DTN2 was designed without any particular scenario in mind and leverages all sorts of legacy networking technologies, fitting better in traditional unconstrained systems (specially Unix-based). Despite some authors claim that it feasible to run DTN2

[16] `http://www.dtnrg.org/wiki/Code`

on more resource-constrained presets [8], the community began working on more lightweight, portable and extensible versions of a BP implementation, particularly to use on small routers and other communication devices like mobile-phones.

Since these platforms are usually tied to their own operating ecosystems, there are not available "one-size fits all" solutions. For routers, IBR-DTN[17] seems to be the most popular alternative, but it is only available for OpenWrt[18] devices. On mobile-phones, some experimentations, such as DASM[19], were made for Symbian[20] platforms, and were followed more recently by the release of an Android[21] implementation called Bytewalla[22]. Other less-common solutions based on BP are also referenced at the DTNRG website[23].

Due to the fact that they all spawned from the reference implementation, which as some limitations itself (because it only defines functionality approved by the DTNRG), most solutions only replicate a subset of its features, thus are prone to even more limitations. This can be noticed, for instance, on the available routing and forwarding strategies included, that are most of the times restricted to schemes like traditional static table-based, epidemic or Probabilistic Routing Protocol using History of Encounters and Transitivity (PRoPHET). Also due to their network-centric character, these implementations usually follow a design that drives away from a more user-based perspective leaving out mechanisms intended to manage any kind of context information that goes beyond the network level. This vision results consequently in the development of limited applications to work on top of this middleware, particularly those that are biased towards direct communication between peers, such as email exchange [37].

[17] `http://www.ibr.cs.tu-bs.de/projects/ibr-dtn/`
[18] `http://openwrt.org/`
[19] `http://www.netlab.tkk.fi/u/thyyryla/dtn/`
[20] `http://symbian.nokia.com/`
[21] `http://developer.android.com/index.html`
[22] `http://www.tslab.ssvl.kth.se/csd/projects/092106/`
[23] `http://www.dtnrg.org/wiki/Code`

2.3.2 Data-centric Implementations

Another way of looking into opportunistic communications is by considering an architectural model where messages are spread throughout the network not based on their final destination but based on what each message represents. This content-based approach determines that a message is passed to every node that declares its interest on it. It assumes a publish-subscribe functionality where messages do not search for their targets but the other way around. Since this is a rather novel approach, there are not many implementations of it out there. However there is a remarkable reference that seems to have some traction in the research community, the Haggle project.

The project defined not only a pioneer approach by introducing this kind of architecture but it also created its own implementation. Unlike any other solution described earlier, Haggle builds on the fact that each message is related to its content and to whoever generated it in the first place. Being strictly tied to the PSN concept, it also determines human-behavior as one of the most influential factors to drive communications in an opportunistic network. It is consequently, by design, immersed on a broader set of context data that spawns from the networking processes into the applications themselves.

Haggle presents itself as a natural fit for the development of both novel context-aware opportunistic communication protocols and applications. Some of these protocols are referred to at section 3.3. As for applications deployed on Haggle, these go from ad-hoc social networking [63, 64] and photo-sharing to medical triage tagging in disaster scenarios [15, 48]. There are other similar middleware implementations which in turn describe their own formal architecture, such as Document Dissemination in Wireless Ad hoc Networks (DoDWAN) [29, 30], U-Hopper [9], or 7DS [53]. These are somewhat limited compared to Haggle since they have resulted from smaller projects, thus they also find it difficult to gather a greater interest from the community, and consequently support a larger set of additional components and applications. There are, however, mentions to the creation of some content-based utilities around file-sharing, news groups and service advertisement or discovery.

Table 2.1: Examples of DTN-enabled middleware implementations both from a network-centric and a data-centric perspective.

DTN-enabled Middleware	
Network-centric	Data-centric
DTN2	Haggle
IBR DTN	DoDWAN
DASM	U-Hopper
Bytewalla	7DS

Middleware implementations fitting the same category usually follow some common specific patterns resulting from their general perspective over delay-tolerant networking. On one hand, most network-centric implementations have in common the fact they are built over the reference specification of the BP, and tend to deploy a particular subset of the features defined in that same specification. On the other hand, data-centric ones tend to promote a similar functionality that enables an awareness to context information, which proves to be critical to these kind of content-based dissemination approaches, allowing a better classification of the data available in each message processed by a node. Table 2.1 summarizes the different categories and middleware implementations outlined throughout this section.

2.4 Summary

This chapter introduced some background concepts that provide a suitable base for this work. It assessed the evolution of social networking services over the last years, establishing the role of context information, and applications built around this concept as defined by the status-quo. This was followed by an introductory guide to opportunistic or delay-tolerant networks with a proposal for proper taxonomy and terminology, a description of the most common application scenarios and the most mentioned architectural models in the literature. The chapter ends with a short survey over the available support technologies that arose from the work on those concepts. Some of

these technologies are proposed as tools that leverage some synergies resulting from joint approaches of opportunistic communications integrated with social-based context.

Chapter 3

Social DTN: Some Synergies

The combination of these two different concepts of social networks and opportunistic communication results in a series of benefits (synergies) that might lead to improvements on the design of even more efficient networking protocols and applications. DTNs' asynchronous *store-and-forward* nature has stimulated the development of distributed replication, caching, storage and retrieval strategies. These approaches promote offline data access that involves an huge deal of cooperation, which by one hand, fits perfectly on environments with an underlying social nature, and by the other also poses a direct benefit to any application that requires an ongoing availability of that data. From the opposite standpoint, social networking is associated with a well-defined theory describing explicit base models that prove to be extremely influential to streamline the data flow on opportunistic networks.

3.1 Store, Carry and Forward

Store-carry-and-forward operation does not relate solely to the asynchronous establishment of hop-by-hop paths that enable messages to flow between a source and a destination within a network suffering from possible delays and disruptions. As the name states, it also relies strongly on providing support for data storage, specifically by leveraging this property for application support using intermediate nodes that might possibly act as some sort of ad-hoc

router. Relying on this principle, Ott and Pitkänen [61] promote a scheme allowing for straightforward content storage, caching, retrieval and propagation of bundles, seen as ADUs, using embedded *application-hints*. These hints introduce a more sophisticated mechanism for content specification and differentiation and so can be used to influence bundle queuing and forwarding at DTN nodes.

The aforementioned authors consider two partly overlapping scenarios: resource retrieval with caching and distributed resource storage in conjunction with retrieval. The former describes that a response to a specific request (e.g. Hypertext Transfer Protocol (HTTP) *GET*) issued by some node in the network, with both being possibly replicated and consequently traveling through various intermediate nodes (keeping the copies with the appropriate discarding policies), allows to react more quickly/effectively to forthcoming requests for the same data using nearby nodes acting as caches avoiding global network access requirements and also issue faster retransmissions and requests in case of packet loss. The latter extends this scenario to also cover store operations, describing a request (e.g. HTTP *PUT*) to save specific content into the network (possibly on a server) that is again forwarded and replicated through intermediate nodes (along multiple paths) who in turn can make it available to other interested nodes later in time, consequently forming a distributed storage. This notion of availability is supplemented with a form of redundancy provided with distributed backup than can also be achieved by this mechanism and can help prevent information loss (in case of hardware or network failures) [61, 66]. To achieve the inherent operational requirements, the authors propose an extended message-based content storage architecture for DTN nodes based on the one illustrated by Figure 3.1. It provides interfaces to perform cache lookups for resources stored as bundles in the queue or the cache for retrieving them in response to passing requests, and also a caching interface for storing resources based on specific selection policies [65].

The approaches described above seem to fit more properly on scenarios where some sort of reliable and globally connected source or sink is available and the opportunistic storage is leveraged for optimization purposes. The

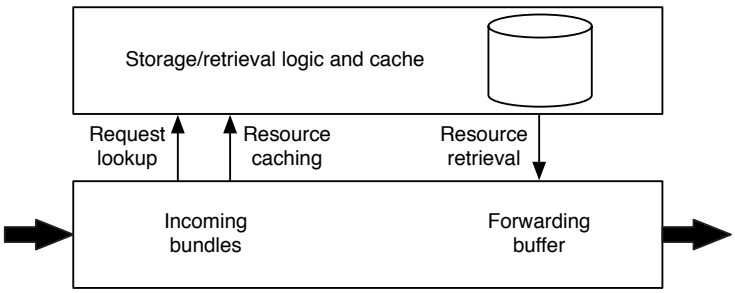


Figure 3.1: Architecture for distributed DTN storage module. Figure adapted from the one presented by Pitkänen and Ott [65].

role to provide that communication bridge might perfectly be performed by one or more (fixed or mobile) nodes acting as Internet access gateways, while other mobile nodes help moving ADUs from or to the latter. This does not mean that it is not suitable to consider serverless storage exclusively in disconnected mobile nodes. However, this requires additional awareness on resource availability, consistency, write access management, and privacy issues, among others [66], something that is also being currently addressed with ongoing research on Opportunistic Computing [13, 15].

This logic is implemented through the use of an additional BP block [70] called *application-hints* that indicates the operations to perform on each bundle and the information required to execute those *a*) the application protocol, to perform resource matching and allow for selective support; *b*) resource metadata (identification, content-type, encoding, lifetime, and any other additional information); and *c*) operation semantics.

Along the same line, Pitkänen and Ott also propose an extended model to deal with bundle fragmentation [65]. Starting from a simplified setup, they describe a process where a node issues a single request and the first reached node (not gateway), holding fragments of the resource (thus aware of its total size), generates a partial response and becomes responsible for requesting the remaining fragments. The model is further extended to allow for erasure-coded transmission. Erasure coding allows to reduce the number of packets in transit by "copying" and sending only part of the message to the next-hop node who in turn is able do decode the full message using only

that partial information. It ensures a better performance by consequently reducing the network overhead. Such a cooperative storage environment created around the nodes forming a sparse mobile community within a specific network, leads to redundancy, storage and availability in bundle communication, consequently increasing the response probability, the cache hit rate and, at the same time, reduce the response latency.

Again, in [65], the authors draw important conclusions about the consequences of having multiple copies of resource located in the network. One is that a resource update will result in temporary inconsistencies, something that is actually aggravated due to the disconnected nature of mobile DTNs. So, it is critical to provide mechanisms to allow for the validation of a resource's "freshness" and establish some sort of version control scheme, something that is claimed to be straightforward since it could perfectly be similar to the set of features provided by web caches. Also, this redundancy could raise some security issues that can also be mitigated using the corresponding functionality available with BP itself. The access control can be provided by encrypting the resource using secure bundle payloads, authentication and integrity have to rely on signed payload only (to avoid possible signature invalidations on re-addressing), what can be done using other bundle security mechanisms. It is also possible to extend these with application-layer encapsulation like Secure/Multipurpose Internet Mail Extensions (S/MIME).

All these properties fall down to mechanisms capable of overcoming a considerable number of networking constraints presented by mobile environments. The introduction of social indicators in the communication process, ends up enhancing even more their importance. Since they are by themselves tied to cooperation, integration and privacy constraints, the use of a social layer can be, in fact, extremely beneficial to allow an adequate management of this kind of issues. Networking by itself, can be assessed by extracting some structural patterns that prove to have an huge impact, particularly on routing and forwarding. These patterns go from base models extrapolated from the social networking theory to indicators about the mobility of nodes and additional parameters introduced with the physical contacts between them.

3.2 Structure of Social Networks

As any kind of specific network, social networks can be represented by graph models that exposes their mathematical base and allow to extract some important properties about the connections they define. The first model to represent this kind of networks was the *random graph model*, where nodes could be connected arbitrarily with a specific probability. It later evolved into a more realistic representation defined as a complex network. These networks present some non-trivial topological features including:

- heavy tail in degree distribution, i.e. only a small set of nodes presents an high degree (number of connections) unlike the remaining, that are in fact connected to those;
- clustering bias, i.e. a considerable set of nodes are tied between them, leading to the formation of network clusters;
- assortativity or dissortativity among vertices, i.e. the preference of nodes to attach (or not) to others based on known similarities (homophily) or differences;
- well-defined structure, i.e. the nodes seem to be connected in an almost deliberate way mimicking communities or even hierarchies;
- reciprocity, i.e. nodes have the tendency to form mutual connections between each other.

Most of these properties also fit a more narrow concept termed scale-free networks which are *graphs* where the degree distribution follows a power-law function, implying that they has no characteristic scale. This is strictly tied to the heavy tail property and describes a network where most nodes have few connections, while a small set of them (called *hubs*) are far more connected and thus are responsible for creating short paths between any two others [47]. Scale-free networks also exhibit a small-world property [51], which is a phenomenon observed along time ago by Milgram suggesting that human-society is a network characterized by short path lengths. It resembles

the *six degrees of separation* concept, where everyone is on average approximately six steps (friendships) away from any other person on earth through a chain established by friends-of-a-friend [18].

Using datasets to represent the social graph from Online Social Networks (OSNs) like Flickr[1], LiveJournal[2], YouTube[3], some authors have tried to develop a model to describe the structure and evolution of online social relationships [41, 52]. They all manage to verify the small-world and scale-free character of those networks by extracting a model centrally supported by *hubs* that form something they call *giant connected component* or *densely connected core* that degrades itself into layers or regions of less connected nodes (forming some sort of isolated communities)—*middle region*—and even totally isolated nodes which they call *singletons*, underlining clearly the power-law property. Besides that, they also observe the clustering pattern (thoroughly analyzed by Backstrom et al. [1]) and the prevalence of short paths between nodes. *Hubs* are considered as a weakly connected component (WCC) since even if they are removed from the graph, nodes can still maintain their connectivity, at least within their community.

Kumar et al. describe the user's role in this evolution [41] by a process in which when early adopters join the network, it begins with an initial growth of their degree (due to an early interest in exploring the system) and is followed by an exponential increase in of the graph size and diameter (since new members join more quickly than friendships are established), settling finally into a period of ongoing organic growth with both membership and friendship links increasing steadily. They then separate the users (nodes) in different categories:

- *stars*, i.e. charismatic users who link themselves to a considerable number of other peers, who in turn might have few connections, becoming popular nodes (*hubs*) in the network;
- users that transpose a real-world friendship network into a virtual one, resulting in the creation of isolated *middle region* groups;

[1] `http://www.flickr.com/`
[2] `http://www.livejournal.com/`
[3] `http://www.youtube.com/`

- user that apply for membership just to briefly explore the system and understand its basics, but quitting soon after, thus not establishing any type of relationship (*singletons*).

When representing networks with moving nodes, some additional metrics arise, particularly about the nature and duration or any other connectivity parameter extracted from the encounters of nodes over time. These are highlighted by Mtibaa et al. [54] that report an experience conducted in a conference environment in which they compare an initial social graph, comprising the friends of the participants attending the event, to a contact graph representing the temporal network created by opportunistic contacts happening when special devices carried by the participants come into communication range (via Bluetooth discovery). They observe major similarities between both graphs and state that forwarding paths between a source and a destination within the graph can be efficiently constructed using nodes close to each other in a social sense, or through the most contacted ("popular") nodes in the network.

Degree and centrality also have their corresponding versions in a mobile network and can also influence clustering, formation of larger connected components and network diameter. Degree could be defined as the average number of devices that a node meets via opportunistic contacts during a Bluetooth scan, and centrality as an occurrence ratio of a node inside the shortest paths connecting pairs of other nodes. High degree nodes, i.e. with many social links, also see more opportunistic contacts often outside their social circles but this proximity leads to the creation of new social connections. As far of centrality goes, a multi-hop context is highly affected by mobility, so formation of hierarchical relationships between the nodes is also noticed. The degree itself is an accurate centrality measure, and can be extended by *closeness* and *betweenness* [17]. The former refers to a possible shortest path between two nodes (how long it will take information to spread from a given node to other nodes in the network), as the latter defines the extent to which a node lies on the paths linking other nodes (has control over information flowing between others).

The characteristics of contacts between nodes are strictly dependent on the perceived social distance between them, i.e. if they represent a direct friendship, friend-of-a-friend link and so on. Contacts between friends seem to be longer but occur much less often than between "strangers", nodes with higher centrality are bounded to more contacts (independently of the social distance) while those with lower centrality depend on closer social ties. Centrality and social distance also play an important role on choosing delay-optimal forwarding paths whereas delays tend to be lower for transmissions between socially-tied nodes and when using those with higher centrality. Those paths are evaluated using specific heuristics that determine beneficial hop-by-hop transmission if and only if *a*) intermediate nodes are within a specific distance of each other in the social graph; *b*) the next-hop is within a specific social distance of the destination node; *c*) next-hop's centrality is greater than or equal to the current's; and *d*) the next-hop is socially closer to the destination than the current one. The combination of these rules improves selectivity, i.e. a larger set of possible fitting paths, allowing a more flexible choice, which however could be attached to some tradeoffs like higher delays and network overhead.

The use of traces obtained by measuring these properties is an effective mechanism to evaluate mobile network protocols and algorithms within a simulation environment. To avoid carrying out the complexity of these kind of experiments, researchers usually rely on datasets possibly created on previous related studies. However, these datasets are: *a*) sometimes not available publicly as they are property, for instance, of telco companies, and thus are kept secret as they may represent a source of competitive advantage; *b*) strictly related to very specific and restricted scenarios and its somehow difficult to generalize their validity; and *c*) generated by specific metrics that cannot be customized. Besides, the fact that it is important to have a mathematical model to provide a more formal character to the simulation itself leads to consider the use of alternative methods like synthetic models. However, the generation of synthetic traces is usually based on random individual movements and result in behavior that is most unhuman-like, which makes it quite difficult to assess to what extent they map real world situations [57].

An hybrid approach can however allow to mitigate those issues, for instance, designing synthetic models starting from real traces. By capturing the key statistical properties of datasets and reproduce them through an appropriate mathematical instance, it is easier provide more realistic input data for simulators [57].

With these such traces, it is possible to estimate parameters like:

- *contact duration*, i.e. the time interval during which two nodes are in communication range;
- *inter-contacts time*, i.e. the time interval between two consecutive contacts made by the same nodes.

Such indicators are particularly important in delay-tolerant mobile ad-hoc networks, since they allow to define the frequency and the probability of being in contact with the recipient of a message or a potential message carrier in a given time period, and to estimate the amount of data able to be transmitted during the process. This allows to determine a robust connectivity model for a given network and identify particular delivery properties that affect the data in transit.

With all this, it becomes easier to understand the impact of human relationships on the mobility and connectivity patterns between nodes in an opportunistic network. The characterization of human contacts allows a refinement of mathematical tools capable to prove that the distribution of contacts follows, again, a power-law function. In this case, that means contacts occur frequently within small sets of nodes, and that from a broader perspective, contacts between nodes all over the network tend to be rare [12]. There is an high factor of clustering which unveils the need to rely on somehow "special" nodes that act as gateways and are able to move data between clusters. This appears in contrast to a single unboundedly large network of socially unrelated individuals, which the small-world experiments proved to be easily downscaled [68].

This contradiction is due to the fact that in real-world social networks, humans tend to organize themselves into communities and these communities do not hold together consistently through time. Something that reflects

perfectly on a person's everyday life where during the day it is close to its colleagues at work and at night is more closer to its family at home. The delivery properties are then influenced by both time correlations and location of nodes. This fact support the findings of Karagiannis et al. [39], who state that beyond a characteristic time of about 12 hours, the distribution of contacts exhibits exponential decay. It is also predictable that such decay is again observed on longer time intervals, specifically if we consider the usually different behavior of users on week days and during weekends [50].

The clustering effect resulting from the formation of groups or communities in the social graph, to which Sastry et al. [68] refer as *cliques*, as some implications on the process of choosing the appropriate next-hop node. These implications are specifically based on the importance associated to different kinds of contacts, those within a *clique* (frequent) and those need to reach outside that *clique* (rare). When removing the possibility for those rare contacts, the number of connected nodes falls sharply, and on the other hand, removing the most frequently occurring contacts does not affect connectivity greatly. Similarly, and given this inadequate ratio of contacts between "friends" and "strangers", Miklas et al. Miklas et al. agree that if the concern is to propagate quickly across a mobile network, the focus needs to be on stranger encounters since they present rare opportunities to reach more people. Otherwise, if the goal is to achieve a setup of more stable and predictable network links, then the focus must be on friend encounters.

Every kind of mobile social structure seems to evidence most of these properties [32]. However, there is a detail that is not properly addressed in the literature, the existence of "familiar strangers" [50]. These might be, for instance, two persons encountering regularly without ever interacting (exceptional relationship of social nature). Nodes can act both as good carriers of each others data, something that leads them to be actually considered as "friends" by some authors. Something like this, can have serious security and privacy implications, so these situations need to be carefully addressed.

Despite the circumstances, these "familiar stranger" contacts can be easily distinguished, avoided or properly handled, for instance, using data provided by the applications themselves. However, researchers seem to frequently ig-

nore them. On the other hand, the community property is seen as quite influential to build mobile social network structures. There are some interesting proposals that promote better approximations to human movement patterns, and thus more consistent mobility synthetic models. One example is the work from Musolesi and Mascolo [56], who present a model that allows collections of hosts to be grouped together, and to move within a specified topographical space in a way that is based on social relationships among the individuals. It allows for the definition of different types of relationships during a certain period of time (i.e. a day or a week) and for a non-random association of nodes to particular areas (either as a stationary origin or as a result of movement) that might be of their particular interest.

Such system is achieved by a representation of the social graph on which the edges' weights are expressed as a measure of the strength of social ties and the likelihood of geographic colocation [57]. A global probability value is used to represent a relationship between the nodes in an *Interaction Matrix* that for a specific probability threshold results in another matrix, referred to as a *Connectivity Matrix* which models the degree of social interaction between two people using values in the range $[0, 1]$ (0 indicating no interaction at all and 1 indicating a strong interaction). The movement of hosts (initially isolated with their respective community) in a simulation grid, is driven by a non-random goal defining the their final destination. This goal is initially chosen randomly inside the node's community (associated to a specific space within the grid) and eventually on other communities in two alternative mechanisms, a deterministic or a probabilistic fashion. In the former, the host moves to the space exerting the highest social attractivity, i.e. its importance in terms of social relationships (for that particular host or community) as an evaluation of the strength of ties that it maintains with nodes within that place. The latter allows for a more realistic selection of destinations based on a probability value proportional to the place's attractivity, but not guided by the strength of social ties with any specific node within that place.

This section described a lot of properties, that can be made available at specific points in the network (by the nodes themselves or by some existing infrastructure) under the form of context information. This can lead to

the formation of a cross-layer channel that correlates data from networking operations (the context surrounding the physical encounters between nodes) and information provided by the application layer (social context of each node). In an opportunistic environment, this kind of knowledge proves to be extremely useful particularly to design and deploy more advanced and intelligent routing or forwarding algorithms.

3.3 Routing and Forwarding

The design of efficient routing strategies for opportunistic and delay-tolerant networks is generally a complicated task due to the absence of knowledge about the topological evolution of the network [62]. Such a compelling challenge has been object of an enormous research effort during the past few years, resulting in a plethora of routing protocols that exploit the store-carry-and-forward paradigm. In this kind of networks, routing and forwarding are usually seen as strictly intertwined tasks performed at the same time (routes are actually built while messages are being forwarded hop-by-hop), so the terms are found to be used interchangeably in the literature. Also, the protocols themselves do not seem to be uniformly classified, what is probably due to the fact that a lot of different metrics need to be evaluated in order to perform a clear separation between them. The issues at stake go from conceptual things like their operational environment and additional knowledge they can leverage in the process, to performance-related parameters such as the reduction of delivery delay, increase of delivery ratio, resource-usage optimization and scalability [21, 62].

Pelusi et al. present a general taxonomy (illustrated by Figure 3.2) that divides routing/forwarding protocols in two main classes: those designed for completely flat ad hoc networks— *without infrastructure*—and those that exploit some of form of additional connectivity support to opportunistically forward messages — *with infrastructure*. Strategies fitting in the former case are then further separated into *dissemination-based* and *context-based* algorithms. The former are essentially forms of flooding the information through the network, as the latter represent approaches that use the knowledge of

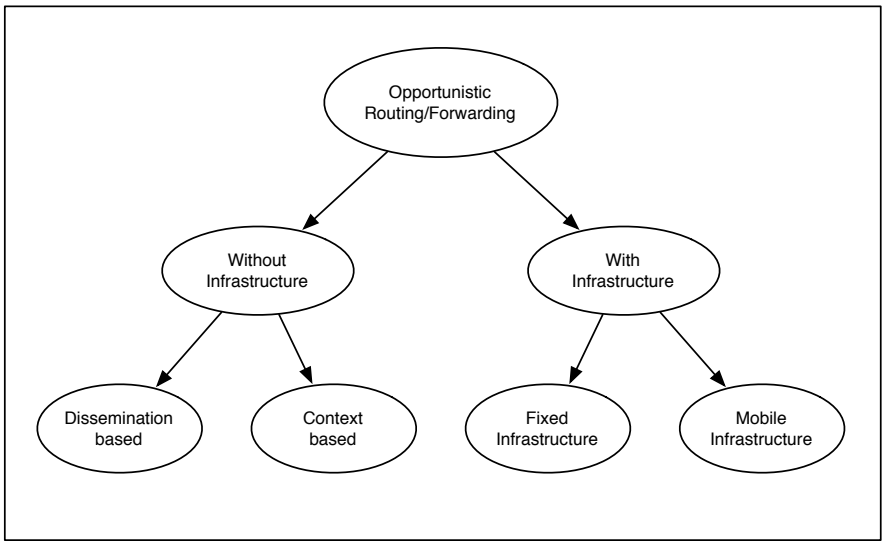


Figure 3.2: A taxonomy for opportunistic routing and forwarding. Figure adapted by the one presented at [62].

the context in which nodes are operating as a way to better identify the best next-hop at each forwarding stage and consequently perform better routing decisions. When referring to approaches supported by some kind of networking infrastructure, i.e. special nodes that outperform the other ones in terms of resource and even network capacity, the authors also make a separation into two different categories, those relying on *a*) nodes of a fixed infrastructure located at specific geographic points; and *b*) nodes of a mobile infrastructure that move around the network following either predetermined or completely random paths [62].

Infrastructure-based routing fits well in very specific scenarios, when there is a need to rely on some kind of base-station or data ferries to act as gateways. The former represents a fixed infrastructure setup that has been successfully applied for instance on underwater environments using the *Infostation* model [26] with both node-to-station only or node-to-node communication. The latter consists in a mobile infrastructure system, where special carriers move around the network area, following either predetermined or arbitrary routes, and gather messages from nodes they pass by. This mobile system has been deployed on deep-space environments (with satellites acting

as carriers) and also on remote and poorly-connected regions (for instance, with a special bus being used as forwarder). Models based on additional infrastructure go a little out of the scope of this work, and so they are briefly described here. However, for more information, one can check a more complete survey provided by both Pelusi et al. [62] or D'Souza and Jose [21].

Protocols designed for opportunistic routing without infrastructure first appeared as pure dissemination-based strategies, basically exploiting some form of flooding. These protocols follow an heuristic where when there is no knowledge of possible end-to-end paths nor of appropriate next-hops, a message should be sent anywhere (diffused all over the network), hoping it will eventually reach the destination passing node by node. Flooding schemes clearly generate high overhead and may suffer high contention, what can consequently result in congestion states and buffer exhaustion on the network nodes. So, in order to mitigate these issues, dissemination protocols operate with flood control techniques that either limit the number of copies allowed to exist in transit, or limit the maximum number of hops a message can travel [14, 62]. A popular example of this kind of protocols is Epidemic Routing [80] where messages are spread into the network by means of pairwise contacts between nodes. It defines a process where those nodes exchange summary vectors containing a compact representation of messages stored in their buffers, and each requests from the other those messages it is currently missing.

In fact, Epidemic Routing is not based on a pure dissemination strategy (infection of contacted nodes) since it provides a flood control mechanism that bounds the maximum number of hops a message is allowed to travel [14, 62, 80]. Other proposals to enhance the protocol, at least in terms of resource usage, are also commonly described, like for instance, the prioritization of stored bundles [74]. A popular technique to achieve controlled flooding, particularly for limiting the number of packets in transit, are sometimes based on network coding schemes which operate as follows, let A, B and C be three nodes forming a network, such as any message traveling between A and C has to be relayed by B. Let A generate a message m_a addressed to C, and C generate a message m_c addressed to A. As the relay

node, B broadcasts a single message containing $m_a \oplus m_c$, from which both nodes A and C can decode their respective parts (Figure 3.3) [14, 62].

Source coding is another similar alternative that aims to increase delivery reliability and reduce worst-case delay. The coding process is performed by the source node that, as already described in section 3.1, sends only a specific part of a message which the destination is able to decode [74]. Other drastic way to reduce the overhead of blind message flooding is implemented by Spray&Wait [73], which separates the delivery in two phases, the *spray phase* and the *wait phase*. During the spray phase, a given number of copies of the message is spread over the network both by the source and by those nodes that have first received the message from it. In the wait phase, each node holding a copy of the message stores it and eventually delivers it to the final destination when (in case) it comes in reach [14].

Even controlled flooding finds it hard to deal with the tradeoff between message transmission delays and network overhead. It actually highlights the fact that a low delay is a consequence of the existence of an high number of copies in transit. This is something that novel "informed" routing approaches are proposing to solve, or at least mitigate. Their "informed" character comes from the fact that they are able to overcome the lack of knowledge about the network topology structure and evolution by leveraging the context in which nodes operate and communicate between each other. These protocols operate in a way that makes them able to establish accurate decisions about the choice of next-hop nodes by looking for those that show increasing match with the known context attributes. The more similarity they present, higher is also the delivery predictability towards the final destination that they are able to provide.

Some authors choose to establish an even more clear separation between this kind of context-aware protocols based mainly on their flexibility and ability to adapt to different kinds of scenarios and network environments. A popular nomenclature distinguishes partially context-aware from fully context-aware strategies [14]. The former exploit some particular piece of context information to optimize the forwarding task, usually strictly related to the environment they are developed to operate on. Thus, they fit perfectly and

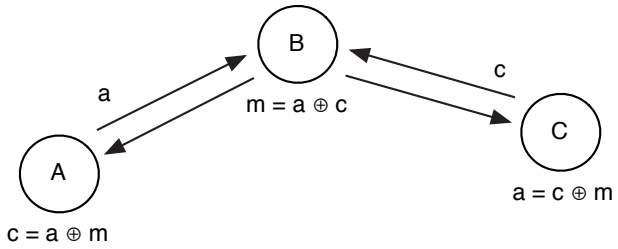


Figure 3.3: Network coding operation performed by intermediate nodes. Figure adapted from the one presented by Pelusi et al. [62].

perform extremely well for that specific scenarios. The latter do not define any restrictions about what kind of context attributes they can manage, and become more adaptive as they are able to learn and exploit everything about the context in which they operate. However, they are all based on the same basic assumption—nodes assume a different importance within the network since each one presents a different delivery probability towards the destination that must be considered during the transmission process.

3.3.1 Partially Context-aware Routing

Partially context-aware routing protocols usually leverage context information that can ben obtained from the network itself. They assume that each node presents its own delivery probability towards a message's destination endpoint given some "historic" facts that can be obtained and made available by each device forming the network. These facts are usually related to previous meetings the node had, specifically the number of times it has contacted with others, the time interval from their last encounter (inter-contact time) and its duration. Such parameters might be useful to determine the "social" character of each node, namely its centrality (degree, betweenness, closeness). Also, if the device allows it, it is also possible to obtain the geographic location where those contacts have happened.

A popular example of a partially context-aware approach is the one defined by PRoPHET [46]. It introduces an extremely obvious, but useful transitivity law stating that if a node A frequently meets a node B, and B

frequently meets a node C, then A and C have high delivery predictability to each other. The underlying strategy is basically similar to Epidemic Routing but with a more intelligent flooding control mechanism. During a contact, nodes exchange, besides the summary vector, another vector containing their delivery predictability towards each node they have encountered previously and consequently for the messages they have stored in their buffers. So, nodes only request from each other those messages for which they provide an higher predictability. At a glance, PRoPHET merely leverages the frequency of contacts that each intermediate node has with the destination endpoint.

These historic facts are used in other alternative approaches to extract additional information about the nature of the registered contacts. Exponential Age SEarch (EASE) [28] alongside Spray&Focus [73] use the time lag from the last meeting with the destination to estimate the probability of delivering the messages. The latter works as an extension to the process defined by Spray&Wait to mitigate starvation on the destination endpoint. This could happen due to the fact that message copies are kept indefinitely by some nodes during the *wait phase* until the destination is directly reachable, thus intermediate nodes must be properly selected in order to avoid those situations. MobySpace [42] uses the mobility patterns of nodes as context information, by grouping those nodes in an high dimensional Euclidian space where each axis represents a possible contact between two of them and the distance along the axis measures the probability of that contact to occur. Nodes are properly placed in that space based upon possible similar sets of contacts and similar contact frequencies.

Bubble Rap [34, 36] builds on an algorithm that exploits the social character of the underlying network. It analyzes more in depth the implications that the existing historic information might have on determining the nodes' social behavior and consequently their mobility patterns. The basic approach defines a fragmentation of the network in specific communities (cliques) where the "global" connectivity is supported by the most "popular" nodes within the network (centrality). Nodes might belong to different communities, so they assume a different rank for each case depending on the number of their social links (degree). The protocol defines a process where messages are "bub-

bled" up in a community towards higher-rank nodes, until a contact with the destination community, or the destination node itself is found.

Another social-based forwarding scheme is proposed by SimBet [17], that also builds on the concept of small-world dynamics but exploits the exchange of pre-estimated betweenness metrics and locally determined social "similarity" to the destination node. It assumes that a node with high betweenness can facilitate interactions between the nodes that it links. The authors present it as the first approach that does not assume a full sociocentric character of the network, and thus have developed it by assuming a possible existence of egocentric nodes. Being the degree easily measured by counting the number of contacts for each node, and the closeness uninformative due to the nature of ego networks, they describe betweenness as the metric that provides the most accurate social pattern in such conditions. They exploit it in order to capture how much a node connects other nodes that are themselves not directly connected. The betweenness calculation is also backed with a similarity calculation, i.e. by considering that the probability of two nodes being connected is higher when nodes in question have a common neighbor (clustering phenomenon).

FairRoute [67] presents an alternative approach with a focus on scalability and reliability by promoting load balancing and resource optimization mechanisms in addition to the throughput maximization and best path selection already assessed by the majority of previous proposals. It builds on the fact that most social-based strategies promote forwarding using an heuristic that biases towards connectivity and consequently the probability of the highly connected nodes to receive messages is inevitably increased leading to an unbalanced load distribution. FairRoute seems to perform better than those forwarding schemes based exclusively on the centrality of each node, by means of a more comprehensive next-hop selection process. The underlying algorithm favors peers that appear to be good candidates to deliver the message by their *perceived interaction strength*, which represents the relationship strength between two nodes measured in terms of each other's list of previous contacts, its duration and the elapsed time since they happened. It also tries to limit communication with those nodes that have less importance, from

a node's standpoint, leveraging contacts between two nodes with equivalent social status (homophily), something the authors define as *assortativity*.

PeopleRank [55] was also developed based on the fact that each node assumes a different importance within a network, which can be somehow evaluated considering its "popularity", given that popular nodes are likely to meet other nodes more regularly or at least more often. It uses a mechanism similar to, and inspired by Google's[4] PageRank [5] algorithm to measure the relative importance of a web page in a set of other pages, in this case, to determine the most suitable nodes for which to forward stored messages. The strategy explores social relationships between the "nodes" by explicit friendships (as defined in OSNs), personal communication (extracted contact patterns), or based on common interests. Each parameter assuming a particular importance in the absence of the others, which can be dynamically established considering the specific scenario using a *damping factor*. This process fits well in a somewhat "centralized" implementation where the social graph is known a priori, or at least can be accessed on-the-fly with some infrastructure support, since it can be cumbersome to transpose it to a pure opportunistic setup. This leads the authors to propose an alternative distributed version where nodes only exchange information about their current *rank* value and the number of social graph neighbors they have.

These strategies favor the influence of social metrics on the communication between each node within the network. They are then considered to be partially context-aware because they work perfectly on human driven scenarios such as Pocket Switched Networks (PSNs), whereas those properties are not available in other setups like sensor networks. Each presents its own preference for a specific subset of metrics, wether as a limitation or as to highlight some particular property which its authors find most relevant. Table 3.1 establishes on which metrics each protocol is actually based. Most are pure metrics tied to the graph theory, some are sort of an hybrid (e.g. degree could represent contact frequency) and the *assortativity* metric encompasses properties such as the contact location, duration, inter-contact times, or any

[4] `http://google.com`

[5] `http://infolab.stanford.edu/~backrub/google.html`

Table 3.1: Social-based metrics as behavior of partially context-aware opportunistic routing and forwarding protocols.

	Clustering	Degree	Closeness	Betweenness	Assortativity
PRoPHET		✓			
EASE		✓			✓
Spray&Focus		✓			✓
MobySpace		✓	✓		✓
Bubble Rap	✓	✓		✓	
SimBet	✓	✓	✓	✓	
FairRoute					✓
PeopleRank		✓			✓

other property that can be used to determine the homophilic binding between nodes.

Historic data is also used in other routing protocols that fitting on different environments, where nodes represent devices that are not exactly carried by humans, especially vehicular networks. Meet and Visit (MV) [7] and MaxProp [6] are somewhat extensions to PRoPHET as they exploit not only the frequency of meetings between nodes but also the frequency of their visits to specific physical places as context information. This might works well for Vehicular Ad-hoc Networks (VANETs) due to the fact that the movement of vehicles is much more predictable then the movement of humans, since they have well-defined paths when traveling by road. This kind of predictive process might not be the better option to deal with routing on human mobile networks. The same applies to other challenged environments like sensor, deep-space and underwater networks, which rely on other particular protocols. More information about these and other similar DTN routing strategies is presented by Conti et al. [14] or D'Souza and Jose [21].

3.3.2 Fully Context-aware Routing

Fully context-aware protocols are not only designed to support and leverage any kind of context information around them that can be used to optimize

routing and forwarding tasks, but also to provide mechanisms to handle and use that context information [14]. They may not be as efficient as partially context-aware protocols in the conditions for which those were developed, but they can be much more adaptive thanks to their learning features [15]. These protocols operate based on an scheme where each node is tied to some sort of profile that contains the most relevant context information, which is previously agreed upon the most common communication requirements of the particular network environments. The profile can encompass data about the node's social behavior such as personal identification information, contacts or relationships, about the device itself such as location, battery life, available memory, storage and processing capacity, among others. To the best of the author's knowledge, there are few protocols fitting this category, and those described in the literature usually have their own share of constraints.

Context-aware Adaptive Routing (CAR) [58] is described as framework to gather and manage context information and assumes an underlying MANET protocol to perform routing and forwarding tasks. Its setup is only capable to forward messages synchronously to nodes at the distance of one hop, that are connected inside the same "cloud". This cloud represents some sort of community from which the messages can only travel via store-and-forward. To reach nodes outside the cloud, a sender looks for a node in its own cloud with the highest probability for delivering the message to the destination. The selected node stores the message waiting to get in touch with the destination itself or with a node (from other cloud) with higher delivery probability (similar to the Bubble Rap approach).

By design, it determines that each node computes its delivery probability proactively and further disseminates it inside its ad-hoc cloud. Context information is leveraged only for determining suitable next-hops inside the cloud, at that specific time. All the available (and pre-defined) parameters are combined using Kalman Filter prediction and multi-criteria decision [40] in order to achieve more realistic values for the delivery probabilities [58]. The main focus of CAR is to define algorithms to combine context information (which is assumed to be already available) [14]. It is expected to be general enough to accommodate for different kind of data and easy to inte-

grate with tangible underlying routing mechanisms. CAR can then see some of its features blended in other specific context-aware forwarding protocols.

One such case happens on Gently [59] which extends the framework provided by CAR with a socially-aware forwarding scheme called LABEL [35]. The LABEL approach was actually one of the first proposals to employ social characteristics into opportunistic routing, being first introduced as part of the Bubble Rap specification. It exploits social clustering to provide asynchronous message delivery to members of a specific community. The protocol basically defines a system where each node is labelled with information about their affiliation or group. This allows other nodes in the network to be aware of a node's community and thus forward to it any message directed to a node belonging to the same community. The strategy is as simple as waiting to reach a node (at a single-hop distance) that has the same label as the destination endpoint and then deliver the message directly.

Gently builds on these two complementary approaches and defines a more complex mechanism, implementing some additional features on top of it. It uses LABEL to establish each node's affiliation, allowing an host to represent the whole community and act as an anycast relay. This is done by dividing nodes in different classes, being these singletons or belonging to a community with various elements. Each community is identified by a class that also defines its cardinality (number of elements). Anycast is performed by forwarding a message to a specific community through some node (that belongs to that community) which is responsible to replicate it (in an epidemic fashion) among the remaining members. Other characteristics of Gently are the ability to allow the forwarding of multiple copies and customizable periodic retransmissions.

In Gently, CAR is used to reach a community (properly identified through LABEL) and route the message inside it, or when social information is not available. If the destination node is in reach (inside the same "cloud"), the message is delivered synchronously. Otherwise, if there is a node belonging to the destination community, the message is forwarded to it via LABEL, and then will be routed inside the community via CAR by means of a suitable carrier (with higher delivery probability). If there is not any reachable node

from the destination's community, the message is forwarded to the best carrier inside the current community. In a worst case scenario, when no further information is available, the message is routed through an available node with the highest delivery probability towards the final destination (CAR-like).

There are other proposals that do not rely on CAR and define their own processes to promote the context management and implement appropriate underlying routing mechanisms. History Based Opportunistic Routing (HiBOp) [4, 5] is a truly opportunistic routing protocol capable of handling a collection of any kind of context data, that can go from information describing the community in which the user (node) lives, its history of social relationships and so on. It defines a process where nodes share their own data during contact opportunities as to learn the context they are immersed in. A message is then forwarded through nodes that share more similarities with the destination. Each node locally stores an Identity Table (IT), containing the user's personal but public information, that is exchanged with other nodes at reach. This allows every node to have a snapshot of the *Current Context* by grouping its own IT and the set of current neighbors' ITs.

HiBOp builds on the fact that humans are most of the time "predictable", and so it considers useful to provide a way to collect information about the context seen by each node in the past. This allows for an insight on the user's habits and experiences which are kept in a History Table (HT) that records each attribute as seen in a node's *Current Context*. The protocol operates as follows: the source node specifies any subset of the destination's IT in the message header which establishes the predefined set of attributes that can be computed throughout the network (can vary on different "versions" of the protocol); intermediate nodes between endpoints hand over the message to next-hop nodes which provide a greater match with the destination attributes (highest delivery probability). HiBOp is flexible enough to allow for tuning the relative importance of both the current and historic context. However, it defines a clear restriction by which only the source node is capable of replicating the message, in order to tightly control the tradeoff between reliability (ratio of correctly delivered messages) and message spread [14].

Probabilistic Routing Protocol for Intermittently Connected Mobile Ad

hoc Networks (Propicman) [60] is another fully context-aware protocol that establishes a forwarding approach based on the context information that is kept in a profile associated to each carrier (similar to HiBOp). Likewise, that information is used to describe a node's social environment and relationships alongside its perceived mobility patterns. It defines that forwarding is performed within a two-hop distance, so the sender selects the best pair(s)—allowing multiple copies—of next-hop nodes that provide the higher delivery probability. The choice of two-hop routes is, according to some seminal research works like the one by Grossglauser and Tse [27], the most suitable to exploit more routing information with minimal additional costs and risks. The algorithm allows multiple message copies in transit and also establishes that the sender should always keep one of them, at least for backup purposes.

It distinguishes itself from HiBOp by introducing some innovative design aspects such as:

- routing with zero knowledge, i.e. routing decisions are based only on the information that each node has about the final destination and not about their intermediate peers;
- profile flexibility, i.e. context information can be somehow categorized by assigning different specific weights to each of the attributes of a node's profile, which leads the authors to consider them as *evidences* of different importance;
- security mechanisms, with integrity and privacy requirements ensured by default alongside a proper encryption the message content, accessed only by the recipient using a specific shared-key and hashing of *evidences*.

The underlying forwarding algorithm starts with the source node sending an header containing the information it "knows" about the destination to all the other nodes at a distance of one hop. This header is a concatenation of all the hashed pairs evidence/value associated to each profile attribute (concerning the destination) selected to be processed. When receiving the header, each neighbor computes its delivery probability and repeats the previous step

also for all its single-hop neighbors. The probability is calculated comparing the matching evidences and their corresponding weight and is transmitted back to the first-hop node which is responsible to determine the full two-hop delivery probability and report it to the source. The sender then chooses the best two-hop route(s) for which to forward the message.

These so-called *informed* protocols have been developed to promote a better balance for the tradeoff between communication delay and network overhead. Their separation on two different categories also reflects the different impact they might have on those metrics, especially considering that some approaches are way to generic to perform well on some highly specific scenarios. This can be noticed mostly in terms of delay since most proposals are usually based on single-copy transmission (or at least pretty much controlled by the source node itself) and perform the selection of next-hop routes in an highly weighted manner. However, from a general point of view, it is perfectly acceptable that fully context-aware strategies can deal better with the above-mentioned tradeoff due to their high flexibility and adaptability. Figure 3.4 tries to illustrate how can all the different routing categories described compare to each it terms of the referred metrics.

An interesting fact about most of the protocols described, particularly those from the fully context-aware class, is that they are usually compared in terms of performance with only seminal approaches like Epidemic Routing or PRoPHET. This is not so weird since these are actually the ones that were developed without any kind of specific scenario in mind, and thus can be perfectly compared as general-purpose opportunistic routing strategies. However, this makes it hard to assess which of these *informed* protocols is actually the better. Likewise, not only they are sometimes proposed by the same authors but are also usually deployed and tested using the Haggle architecture, which might mean that they are not as generic as their authors claim.

Of course, these routing strategies are based on decisions that cannot occur only at a network level. The application layer itself plays a critical role on feeding context information and providing the tools to properly manage it. Also, the calculation of the delivery probabilities could encompass an

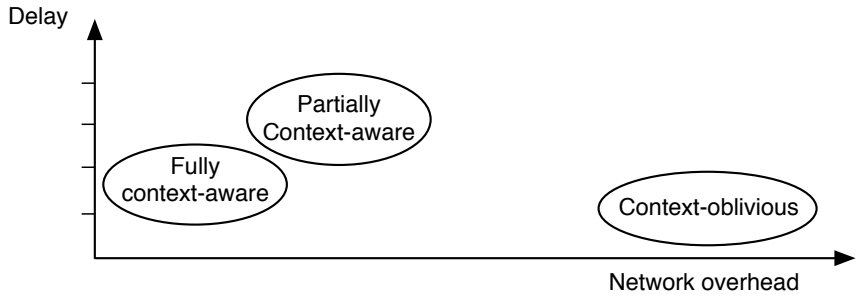


Figure 3.4: Delay and network overhead comparison between the presented categories of opportunistic routing/forwarding. Figure adapted from the one presented by Conti et al. [15].

heterogeneous set of properties, some being quite complex, that sometimes need to be correlated in order to achieve more practical results. Additional middleware components are needed in order to provide support for both the routing logic, the applications themselves and, by the same time, to allow for an easy management of this cross-layer context information exchange. This is the base for most of the contributions proposed by this work and leads to its main proposal.

3.4 Summary

This chapter provides an overview of some of the most relevant synergies that result from a joint application of the presented background concepts. It looks at how store-and-forward mechanisms can have direct consequences over an underlying contextual-layer. Then a comparison between some properties of social networks when viewed as complex and scale-free systems and opportunistic behavior is established, assessing on how to leverage those properties on the development of suitable routing or forwarding algorithms for delay-tolerant environments. These synergies are one of the main motivations for this work and constitute the basis of most of the contributions being presented through the next chapters.

Chapter 4

Architecture Proposal

This chapter introduces SociAL-based OppOrtunistic Networking (SALOON), a framework developed to work as a testbed for the deployment of novel opportunistic routing protocols and applications based on the context generated at two different scopes, the networking and application level. It starts by providing an overview of the architecture in which the system is built upon, and later, describes each component forming it with a formal analysis of its internal organization. Some considerations about the design and other planning choices and decisions are made in a later section.

4.1 Overview

SALOON was planned based on some well-defined premisses. The first one considers that nowadays, the most likely application scenarios for opportunistic environments are sensor-based or human-based communications. A sensor network tends to be pretty much homogeneous, since the nodes forming it are usually similar between each other, so it becomes easier to assess their communication issues, that are particularly related to resource constraints. Network environments formed by persons i.e. Pocket Switched Networks (PSNs) are much more challenging regarding those issues, due to their heterogeneity in terms of density, number of nodes and available technology. By the other hand, they also create an additional value since human actions are driven

by specific aspects that can be used to improve the communication process, leading to more capable and adaptable networking strategies. These aspects are mostly of social character, and constitute the major building block of the framework.

Also, it is assumed that the recurrent form of networking in a PSN consists in using some kind of mobile communication device, particularly, and again based on the status quo, one that is or resembles a *smartphone*. This brings us to another premise, which considers that, given the constant evolution of these devices, the plethora of network interfaces and communication technologies—increasing widespread support and decreasing infrastructure requirements—provided by them, its almost guaranteed that there will be chances to connect to a global network (Internet), as rare as they might be. This opportunities must be leveraged to enrich the context inherent to the interaction between the persons forming the network, their devices and others nearby. This is something similar to the concept of *islands of connectivity* described earlier.

Considering these premisses, the framework must assume an adaptive structure fitting a couple of different setups. In a purely opportunistic scenario it must contemplate only communications between a delay-tolerant service, running on the device, that connects it with every other supporting the same service. Other possible state of the system is when the device is able to connect to the Internet and engage with a special-purpose component that provides the additional social context. This should also happen in a situation where only a few nodes (mobile devices, or any kind of infrastructure) are exposed to that global connectivity and thus are capable to act as gateway. The context should be formed by a set of indicators sufficient to establish a social profile of each user and characterize relationships maintained with others, in order to classify contacts that occur in the opportunistic setup [50]. In other words, to separate those made between perfect strangers, familiar strangers, friends, friends-of-friends or even family and determine other things like their location or any other communication pattern.

In order to satisfy all these possible situations, the system was built based on the network architecture illustrated on Figure 4.1, where three differ-

ent communication environments are assumed. One purely opportunistic, thus *disrupted* from any kind of Internet access, other where some or all nodes forming the implicit local network are globally *connected*, and a third one where that connectivity is provided by some local infrastructure that serves as an Internet *gateway*. For the last two scenarios, there is also the possibility of ad-hoc communication between *connected* nodes and their counterparts. The architecture builds on a client-server model, making a clear separation between two different service instances, one set of communication functionality (both connected and opportunistic) running on the mobile nodes, and another on infrastructure components (in a local connected server or in the *cloud*). This works as a recommendation, but the author believes the architecture should be open to different operational models (like a purely device-based one), however there is a reason behind this approach which is discussed more in-depth at section 4.3.

With the objective of classifying a contact between nodes, based essentially on a possible social relationship maintained (or not) by them or any other previous contact they might had in the same circumstances, comes the need to unify two different panes. Those panes, i.e. a representation of all their social relationships and all their previous contacts can be seen as graphs built over a set of indicators that are obtained from two separate sources. The system considers a model where the social graph can be generated by means of appropriately *connected* applications and the contact graph should be created by specific network services that register the patterns of each physical contact occurred between a pair of nodes.

Social context indicators can be made available to the system, for instance, by interfacing with Application Programming Interfaces (APIs) provided by Online Social Networks (OSNs). It is important to understand the weight of each indicator in a social relationship, however the system is general enough to accommodate for different metrics made available by different context providers, in this case, the particular APIs for each OSN. So, these context providers are accessible through individual plugins that are attached in the mobile system and provide a communication channel with their counterparts available online which are, in turn, responsible for handling the

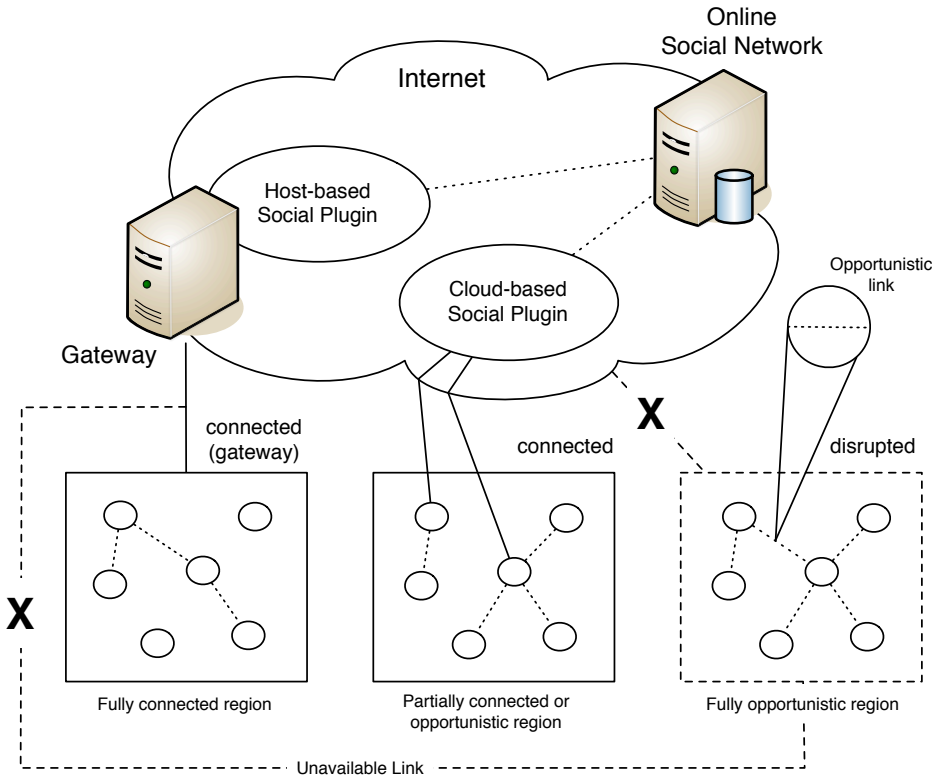


Figure 4.1: Overview of the system architecture.

computation of all the context information made available by each API.

The choice for this kind of approach, is due to the fact that each OSN reflects its own particular vision of the social graph of its users. Albeit there are currently some projects, such as Google's Social Graph API[1] and FOAF[2], that try to promote a unified structure capable of representing the relationships between users on mainstream social services, these do not seem to be acknowledged by the services themselves. Moreover, since the size of those graphs or the subset of context information could vary between services, or even from other plugins, the system should be agnostic to this variations and offload any heavier computational process to a more resource-capable agent. For obvious reasons, particularly, to assure realtime access to the most recent data representation, this process is only triggered while the

[1] `http://code.google.com/apis/socialgraph/`
[2] `http://www.foaf-project.org/`

system is on a *connected* state.

Any context information directly generated by the physical encounters between the devices forming an implicit local network must be promptly registered by the device itself by means of an appropriate service. It should provide a neighbor discovery mechanism with which it identifies any contact opportunity and holds a set of indicators that allow to describe the properties of that contact. Those indicators must then become available to external processes such as applications, in a seamless fashion, by providing a simple channel to establish a correlation with any other additional context information, particularly, the social indicators described earlier.

4.2 System Components

A plugin is composed of two different parts. One running on the server side, responsible for providing an interface with a social context provider, and the other running pervasively on each device and working as an interface with the opportunistic context provider. The server-side component which from now on is referred as *fixed* should operate as a proxy, interacting with existing services such as OSNs. The mobile component consists of a service deployed on specific delay-tolerant middleware, and also separated in two different parts. One corresponds to a mechanism that operates on the discovery of neighbors to register specific information regarding each contact made between nodes, the other listens on the device's network interfaces for opportunities to connect to the Internet, and as that happens, it establishes a communication channel with the *fixed* component.

The *fixed* component has the clear role of providing a suitable representation of the social graph. It is a simple web-service that provides an offline communication layer with a specific context provider. This solution allows to maintain a continuous up-to-date version of the social indicators which can be cached and immediately retrieved by the mobile agent when a connection opportunity arises. To provide an adequate abstraction of the data exchanged between each component, the service needs to define a simple API for the communication between each counterpart.

This API should follow a similar design regardless of the context provider being used. The author proposes a communication format like the one illustrated on Figure 4.2 where a session is established between both agents, and the client (mobile component) makes an Hypertext Transfer Protocol (HTTP) GET request to the web service that includes as parameters its identification token for the given context provider supported by the plugin and any additional authentication or authorization parameters that might be required. Then, the server responds with a structure intended to represent what the author calls the *affinity graph* (more on this on Chapter 6). Again, to avoid unnecessary computation on the client-side, this structure should encompass simple data elements, for instance, organized in attribute-value pairs, and should be made available within an easy-parsable format (e.g. JSON).

Which indicators to request from the context provider and how to manage them, are problems specific to single plugin implementations. When the source of this indicators is an OSN, the process should follow a natural order where for each registered *friend* of the connected user, the service requests the additional parameters it needs to measure their relationship. Any other situation goes out of the scope of this work and so it is not described here. The process illustrated on Figure 4.2 is supposed to describe the basic functionality, but it should be noted that, the web-service must not retrieve information from the context provider as response to requests made by the mobile agents, to avoid leaving them waiting indefinitely.

A better approach is to allow the mobile agents to first register with the web-service, making it in charge of managing the context indicators related to social graph of their respective users. These indicators are then cached in the server and updated in a configurable time-interval, or when the provider allows for proper subscriptions and sends change notifications to the web-service which, in turn, should act accordingly to also update those indicators. This way, subsequent requests made by the mobile agents (again, over an adequate time interval) are able to be promptly satisfied.

Each plugin must also specify individually how the structure containing the social indicators should be handled by the mobile agent. As will be clear

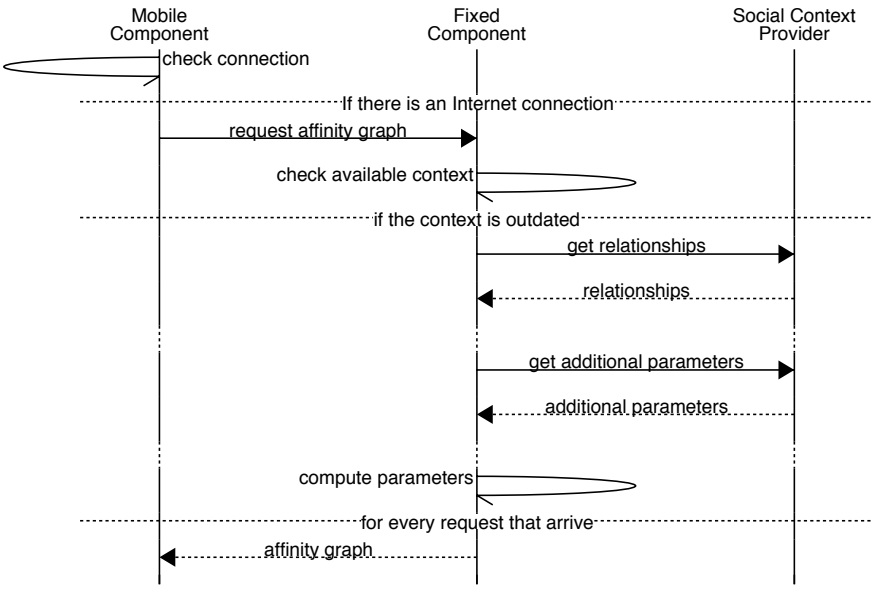


Figure 4.2: Communication process between the plugin components.

later, the goal is to operate on a single value that can describe the *weight* of the relationship between two users, however, each plugin has its own different needs and sometimes it is useful to leverage some of the individual parameters used in the calculation of that value. So, the author recommends that each plugin defines its own persistence model and implementation that allows to store and cache each of those parameters, again, bearing in mind the resource limitations of the mobile devices.

Besides handling the information retrieved from the web-service, the plugin's mobile component should also be able to integrate itself with the existing DTN-enabled middleware. This is somewhat important in way that the retrieved context indicators should be available to be processed by any network tool (routing, forwarding or even management protocols) or application deployed over the middleware. Also, instead of creating additional mechanisms to advertise that social data to other nodes in an opportunistic network, the framework should leverage any existing middleware component capable of providing things like service announce and discovery.

To deliver a seamless mechanism for the provision of these data, when nodes are in contact, they should exchange a token containing references or

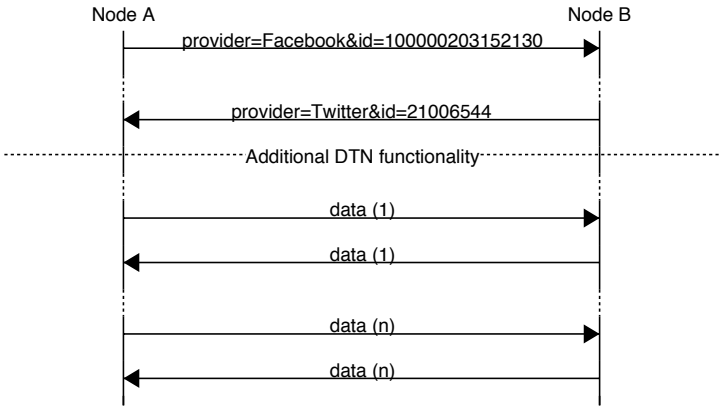


Figure 4.3: Advertising context providers between nodes.

links to any external, preferably online, context providers. The global context is determined on a per-user basis, so the system should allow mobiles agents to be aware of any additional indicator that extends their contact properties and social relationships. Each agent can build a *profile* for any other it comes in contact with, and later connect it to the specific external context sources, in order to update that *profile* whenever an opportunity arises. A way to advertise those connections is by exchanging with other nodes the user's identifier for a given OSN just like it is illustrated on Figure 4.3.

4.3 Design Considerations

Planning and designing the system was a process driven by two main perspectives, availability of resources and data. By one hand, since the goal was for it to be built over a very specific device, i.e. a mobile phone, there should be a perfect awareness about the device's limitations and constraints. The same applies to any particular software or tool needed for its deployment. This relates specifically to the delay-tolerant implementations available for such particular usage. The pervasive component was strictly bounded, operation-wise, by the middleware it could be built upon, and resource-wise, by the overall limitations related to the device itself.

In addition to that, the system is separated into a *fixed* and a *pervasive* component as way to promote a better integration with infrastructure-based

Table 4.1: Examples of social context providers available on the device (local) or through an external (global) source.

Social Context Providers	
Local	Global
Location Services	Social Networks
Address Books	eCommerce Platforms
Calendars	Event Planners
Photo Albums	Dating Services

services such as OSNs, which in this case, are used as the main source of application-wide social context indicators. There are other alternative solutions (Table 4.1) that can generate somewhat related data, for instance, any contact or address book provided by the device's operating system. However, a source like that could only deliver a very restricted subset of social indicators which could even allow to classify the relationship between each subject connected by that context, but would fail to support any capability to determine the value of those relationships. In other words, they would find it extremely difficult to evaluate the strength of the ties connecting the subjects. These resource and data-access limitations lead the author to recommend this kind of strategy, however, the framework is general enough to accommodate full-fledged mobile device-based *plugins*, so the choice is left up for anyone who pretends to build additional plugins.

Nowadays, mainstream OSNs like Facebook or Twitter tend to provide a rich set of contextual parameters about the social graph of each user. So, to avoid overwhelming the pervasive component with computation of a set parameters that could assume an uncontrollable scale, the system defines a mechanism where the mobile counterpart only deals with a well-defined context structure that is built on the server, and requested only when needed. This structure should encompass the least data possible, and thus, the *fixed* component focuses on calculating single values that can express the weight of each set of indicators for the relationships registered by each service. Those values should then be used to determine a global indicator that determines

the *affinity* each user has with all the elements of its social graph. As it is ready to serve separate social plugins, the system does not force the use of any special strategy to match these indicators (since they also vary throughout different APIs), however, a specific scheme was used in the development of a proof-of-concept plugin, which is described in section 7.1.

4.4 Summary

This chapter makes a global architectural assessment of the SALOON framework. It starts by defining the basic network functionality and operation requirements of the system and ends up with a description of its most relevant components. All the design choices are substantiated later taking into account some specificities of the system itself. The next chapters provide a more in-depth architectural description as a bottom-up approach, starting with the middleware implementation that provides support for opportunistic communications on the framework.

Chapter 5

DTN Support

PSNs can only be maintained if all the devices forming the corresponding opportunistic environment provide support for the respective underlying communication protocols. In other words, each node must provide an interoperable and DTN-enabled network stack. SALOON aims to provide a research tool prone to be used by others in the scientific community, so, it should promote a design based on well-defined standards. This chapter describes how this specific issue was addressed by choosing an opportunistic communication stack based on the Bundle Protocol (BP) deployed over a widespread mobile ecosystem such as the Android operating environment. It provides a more in-depth analysis about some specific features of the BP, particularly those that are specified as part of the chosen Android implementation.

5.1 How it Bundles

Choosing to build a system based on the BP, other then any different architecture, was rather easy. The protocol has been maturing over the last years with the work driven by the Delay Tolerant Networking Research Group (DTNRG) and nowadays it provides a stable specification that spawned a series of consistent implementations. BP implements store-and-forward by providing some key capabilities that include:

- custody-based retransmission;

- coping with intermittent connectivity;
- taking advantage of scheduled, predicted and opportunistic contacts;
- late binding of overlay network end-point identifiers (EIDs) to constituent internet addresses;

All these features were created over a networking model based on collaborative communication between nodes in disconnected environments where they "discover" each other through specific neighbor-awareness mechanisms and exchange information by means a convergence layer that sits on top of traditional stacks and works for various subnetworks. These building blocks define the basic functionality provided by the BP. The plethora of underlying protocols over which they could be built, leads them to be treated as individual research problems, per se. As an example, there are separate convergence layers for transport protocols like Transmission Control Protocol (TCP) or User Datagram Protocol (UDP) as for link-layer technologies like Ethernet and WiFi or even direct physical mediums like Universal Serial Bus (USB). The same happens with neighbor-discovery agents developed individually to work with IP, Bluetooth or many other communication tools.

Since there could be specific convergence layers for different protocols, they are usually called Convergence Layer Adapters (CLAs). Each CLA works as a glue to stick applications on the store-and-forward overlay network to the underlying subnetworks in order to send and receive bundles via the available internet protocol(s). Each *adapter* is expected to provide some basic services to a node (or BP agent), namely: *a*) sending a bundle to all nodes reachable by a specific EID and by the same convergence layer protocol; and *b*) delivering to the agent any bundle sent by a remote node via the same convergence layer protocol.

CLAs provide bundle conveyance over specific connection types, defining encapsulation of bundles as well as procedures for connection setup and teardown. A node establishes a connection with the intended peer, typically by using the services provided by the operating system, for instance, a socket connection between pairs of addresses and ports in a TCP communication

or master/slave RFCOMM channels in Bluetooth. Bundles are directed to a node's singleton EID which denotes its bundle-layer address and is advertised by an initial contact header exchanged between agents. Once the connection is established and configured, bundles can be transmitted in either direction. Each bundle is packed into one or more logical segments of formatted data which encompass specific information such as an appropriate header, the segments' length and the bundle data byte range.

Due to the nature of the underlying protocol, some CLA implementations provide additional features such as bundle arrival confirmation or transmission interruption using specific acknowledgements, and bundle fragmentation. The connection between nodes is maintained with a periodic transition of *keepalive* messages and is closed when nodes are no longer in radio range, thus not receiving those messages from each other, or when specific *shutdown* messages are issued by one of them. This teardown process might also be prone to other appropriate additional acknowledgments, depending on the underlying protocol.

Any message exchange in an established connection should happen according the rules defined by the available routing and forwarding protocols. BP by itself does not define any specific standard routing scheme, however, since the early days, most implementations are being shipped at least with a static table-based and/or an epidemic router. However, routing and forwarding can be considered as a separate component from the CLAs, since their only purpose is to determine paths and policies for message expedition consequently providing a selection of the appropriate nodes with whom to communicate. CLAs just create a suitable bundle communication channel between nodes that fulfill the requirements met by each *decision* taken on those routing schemes.

In most delay-tolerant scenarios, the identity and meeting schedule of participating nodes is not known in advance. Neighbor discovery provides the ability to dynamically search for all the bundle agents operating within the same network. This mechanism usually consists on a process where each node advertises its presence via a small beacon that indicates how it can be reachable, in this case, providing its own EID. In addition to aid in the

identification process, these beacons can also be used to advertise specific services that each node provides, including the available CLAs, routing protocols, among others. A node indicates to its peers a proper way to create a direct communication channel between them, which is built based on its EID and an available convergence layer protocol.

Beacons should be sent and received by nodes that pretend to advertise their presence and detect the availability of other neighbors. Their nature is determined by the conditions offered with the underlying transport protocol, which does not happen with their content, that should be agnostic to those. For instance, on capable mediums like Ethernet or WiFi, beacons should be sent as link-layer broadcast messages, and the same should happen whenever possible. This proves to be an important point, because broadcast beacons are designed to reach unknown neighborhoods within the local network domain, thus allowing for a larger operational scope of the opportunistic environment. Multicast and unicast beacons are also useful, but for more particular situations where there is a need to select specific nodes with which to communicate. A full-fledged discovery mechanism should provide support for all these kind of features.

DTN discovery mechanisms must be designed as efficient and extremely lightweight service components. They must be capable of supporting arbitrary length EIDs, and include CLA information in order to maximize the utility of each beacon message without requiring multiple round-trip times (RTTs) in order to perform complex protocol negotiation. This is what distinguishes them from other discovery protocols that are, for instance, based in mapping systems, such as Domain Name System (DNS) or Session Initiation Protocol (SIP), that allow service location and host resolving by keeping updated versions of the bindings [82]. These services are global in nature, and they only work in well connected environments that provide continuous access to the mappings.

5.2 An Android Implementation

The initial version of SALOON was built over a specific middleware, Bytewalla, which is a grass-roots implementation of the BP designed to bring DTN-based communications to the Android platform. Android was selected as the reference platform because of its openness and great industry support, but the architecture presented in Chapter 4 should be extensible to other different mobile computing platforms. As far as the author knows, it is the only available implementation for this platform and is probably the most recent of all the other alternatives for different operating technologies. Obviously, it is intended to comply with a terrestrial scenario where the network if formed by Android devices, particularly carried by people (PSN). Bytewalla is maintained by a research group in the Telecommunication Systems Laboratory at the KTH Royal Institute of Technology[1] and it is being developed in separate versions, each one introducing new specific sets of features:

- the first version introduced a basic setup for delay-tolerant communication based on the reference specification [70];
- the second version provided a security solution for the initial architecture by introducing some data encryption mechanisms under the specific resources and expected performance;
- the third version was developed for a real world scenario of opportunistic communications and introduced things like neighbor discovery and additional routing mechanisms; and
- the fourth version is a direct extension of the third and provides enhanced routing functionality with the addition of various forwarding strategies and policy queueing mechanisms;

The second version was developed within a different branch and thus, was not followed by the subsequent implementations which are also built over the first version of the system. SALOON uses the latest version, so it does not

[1] `http://www.kth.se/`

implement the security functionality described above. Security is a very broad topic, not only in computer science in general, but also particularly in opportunistic communications which makes it go out of the scope of this work. The framework is intended to be used for testing purposes, so this is something that does not pose any limitation to its planning or development.

Bytewalla only implements features that were subject of thorough work and resulted in published specifications by the DTNRG. It does not provide every state-of-the-art dtn-enabled component but it successfully delivers a clean platform with well defined standard features and provides an adequate set of tools to build a testbed like the one being proposed. The most common BP functionality such as late-binding with EIDs, custody transfer, bundle processing and generation, Self-Delimiting Numeric Values (SDNVs)[2] encoding, and endpoint registrations, is entirely supported. On the points highlighted in section 5.1, it provides off-the-shelf implementations of a TCP CLA[3], Internet Protocol (IP) neighbor discovery[4] and a version of Probabilistic Routing Protocol using History of Encounters and Transitivity (PRoPHET)[5].

The TCP Convergence Layer (TCPCL) provides a way to enable TCP-like communication between nodes, which mimics its connection-oriented nature offering a state-full communication context and flow control mechanisms like bundle acknowledgments and refusal. A TCPCL session encompasses the connection establishment, transmission of bundles (fragmented or not), possibility of errors and rejections, ending with a connection teardown. It is very similar to the traditional TCP operation, as one can see on Figure 5.1. There are other alternative CLA implementations compliant with official specifications such as versions for UDP[6] and Saratoga[7]. However, these two in particular, were developed with a focus on space-based and satellite communications and thus are not very useful to a terrestrial environment like the

[2] http://www.dtnrg.org/wiki/SDNV
[3] http://tools.ietf.org/id/draft-irtf-dtnrg-tcp-clayer-02.txt
[4] http://tools.ietf.org/id/draft-irtf-dtnrg-ipnd-01.txt
[5] http://tools.ietf.org/html/draft-irtf-dtnrg-prophet-07
[6] http://tools.ietf.org/id/draft-irtf-dtnrg-udp-clayer-00.txt
[7] http://www.ietf.org/id/draft-wood-dtnrg-saratoga-09.txt

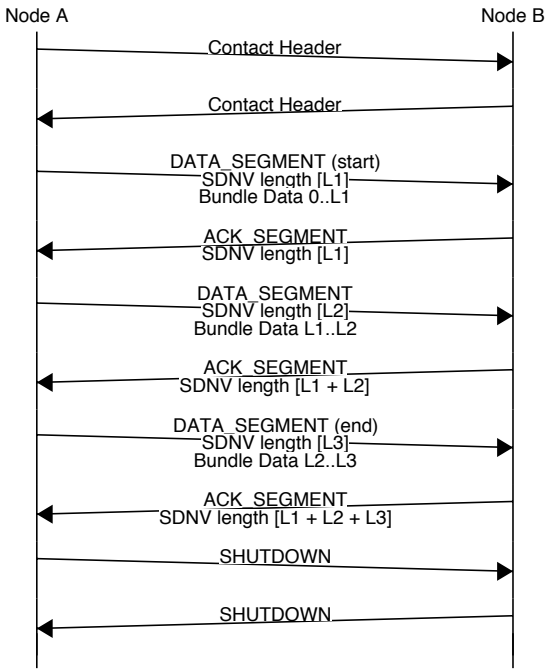


Figure 5.1: TCPCL session between two DTN-agents.

one envisaged by Bytewalla.

No improvements or additions were made to the Bytewalla system, regarding the convergence layer functionality, since the author believes that the TCPCL is mature enough to accommodate all the communication needs of the framework. However, the same does not apply to the neighbor discovery mechanism provided alongside. Despite referring to the latest specification draft of the IP Neighbor Discovery (IPND), the implemented version of the protocol does not seem to provide the necessary extensions to allow the service discovery functionality, as it is described in section 5.1.

In fact, Bytewalla implements a very rigid beacon format, the *discovery header*, illustrated on Figure B.1 (see Appendix B). It only specifies the presence of some convergence layer properties such as its type, TCP-only in this case, the respective address and port, and additionally, a local EID, the time-interval for which to expect updates, and the length of each variable field. The CLA type is represented by a proper byte-code which all nodes are capable to map into a tangible descriptive value. This format does not

follow the one specified by any official IPND[8] draft, lacking both mandatory and optional fields.

IPND's latest specification describes a beacon format which produces the protocol version number and respective flags, the *message* sequence number and a set of length fields packaged as SDNVs. Besides these mandatory fields, the format also allows the addition of two optional blocks containing information about the services available at each node, and a set of Bloom Filters[9] which enable a compact representation of bidirectional links between nodes. This *standard* format is depicted on Figure 5.2.

Bloom Filters are a whole different research issue that falls out of the scope of this work, and since they are not part of the Bytewalla implementation, which provides the basic required functionality, the author has chosen to ignore them, and does not describe them further. On the other hand, service discovery proves to be an useful functionality for advertising the profile references for any external context provider such as depicted on section 4.2. This mechanism is then implemented on-demand over the discovery component already provided on Bytewalla, in other words, somewhat ignoring the recommendations presented by the IPND specification.

SALOON builds its own service discovery mechanism on top of the base *discovery header* format. It consists of an additional block which can be used to represent available routers and different CLAs, but is at the same time, sufficiently general to accommodate unanticipated services provided by the advertising node, such as the profile tokens being considered. The block's format is inspired by the one presented on the latest IPND draft, which is illustrated on Figure 5.3. Basically, it comprises some sort of an associative structure where each available service is arranged by its name and a set of parameters, delimited by well-defined lengths in the form of SDNVs.

The draft defines that the *Service Name* field should be a string identifying the canonical name of the service, and the *Service Parameters* field can be a set of attribute-value pairs, e.g.

```
address=192.168.1.3&port=4453
```

[8] `https://datatracker.ietf.org/doc/draft-irtf-dtnrg-ipnd/`
[9] `http://en.wikipedia.org/wiki/Bloom_filter`

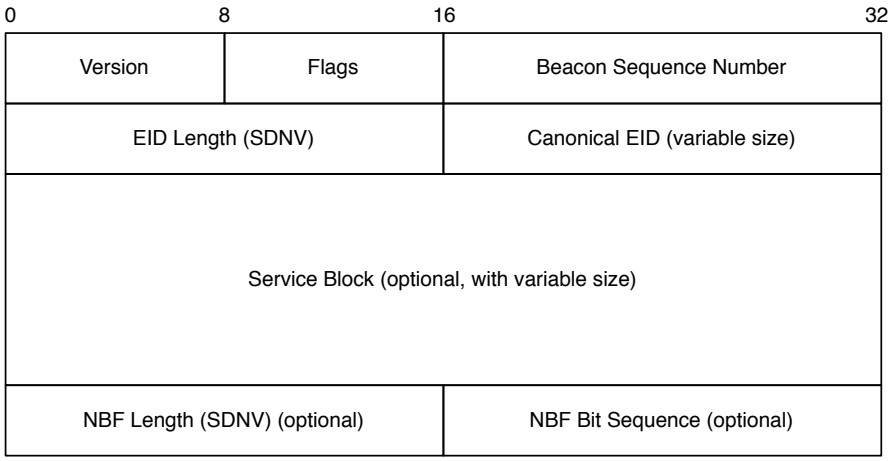


Figure 5.2: Neighbor discovery beacon format as specified by the official IPND draft.

The framework uses this fields to package a name for the external context provider, in this case, a specific OSN, e.g. "Facebook", and the respective profile references (access, authentication or authorization) required to identify the user's profile, e.g.

```
id=100000203152130&access_token=ABC123
```

Each node should be able to map a service to its respective plugin, what can be done by also passing the plugin's *fixed* component Uniform Resource Locator (URL) as a service parameter.

Bytewalla provides a mature implementation of a pure opportunistic routing protocol like PRoPHET. Since the goal of SALOON is to create a testbed to allow an enhancement of the available routing schemes, and specially, the introduction of different ones, the author uses the protocol as a reference to analyze and promote some contributions. He looks for common patterns in various schemes, not working specifically or directly with PRoPHET Nonetheless, these three different components (convergence layers, neighbor/service discovery and routing/forwarding) are the support beams of all the networking processes within the framework.

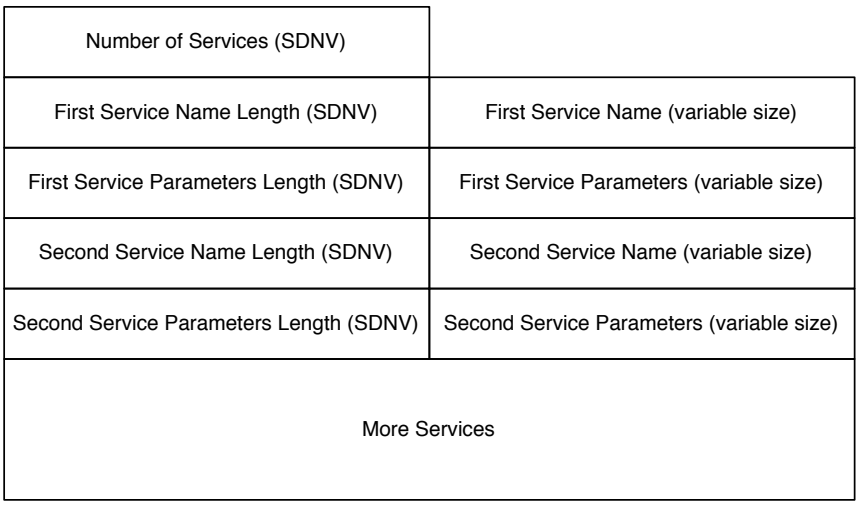


Figure 5.3: The service discovery block format as specified by the IPND draft.

5.3 Summary

This chapter provides a more detailed description of some architectural components related to the DTN-enabled middleware used in the implementation of SALOON. It establishes the critical components at this level, as being related to the convergence layer, neighbor/service discovery and routing support. Finally, it describes how each of these components is made available by a specific implementation of the BP, for the Android platform, over which the framework was built.

Chapter 6

Context Profiling

SALOON's main focus is to provide a context-awareness environment such as the one promoted by some of the data-centric middleware implementations described on section 2.3, removing by the same time, the need to resort to standard-oblivious functionality like those approaches end up promoting. With such a concern, it builds, like already stated, over a reference BP component to which are attached some additional modules that should work in a seamless fashion, by providing appropriate interfaces to every context provider existing in the system. In this particular case, the framework illustrates its functionality based on two different data sources, the contact graph and the social graph of each node within an opportunistic network setup.

6.1 Opportunistic Contacts

Contacts between nodes provide a rich source of context information, in a way that enables a proper evaluation of how to handle the transmission of messages based on constraints like the availability of the network, the urgency of each message, its size or any other transmission restrictions that might arise. Those parameters can be correctly assessed by measuring a set of properties abstracted by the context surrounding an encounter of nodes within the network, such as the:

- frequency of contacts between two nodes;

- duration of each contact;
- time elapsed between contacts;
- geographic location of a contact;
- available network interfaces in each node;
- available bandwidth for each contact;
- average battery level of each node;
- available memory and storage space in each node; among others.

Each of these context indicators can be obtained through physical communication between nodes by *a*) producing the appropriate functionality at each network interface (frequency, durations and bandwidth); *b*) using core services provided by each device like Global Positioning System (GPS) and computational facilities (inter-contact times, locations); and *c*) advertising internal status parameters and mapping of resources to other nodes by means of service discovery mechanisms (network interfaces, battery, storage, etc.). By combining these parameters for each contact between two specific nodes in an mobile network, it becomes possible to *quantify* an underlying relationship tying both of them, and consequently obtain an appropriate weight for that connection in the representative graph. This graph sits over what the author describes as the *physical pane*.

Due to the fact that SALOON builds over a strong social base, some of these parameters are actually obliterated. The framework focuses only on those that allow for an explicit measurement of the type of relationship maintained by each pair of nodes. The author further separates the context indicators presented above in two different categories, *social-based* and *resource-based* (Table 6.1). As stated previously, in section 3.2, the first set of properties that encompasses the frequency, duration, interval and location of contacts, proves to be critical for the classification of those relationships. Resource availability and constraint related context indicators are useful, particularly, for some use-cases on routing and forwarding, but

Table 6.1: Examples of context indicators for the two different categories fitting the physical pane.

Physical Level Context	
Social-based	Resource-based
Contact frequency	Network interfaces
Contact duration	Contact bandwidth
Inter-contact time	Battery level
Contact location	Memory and storage

are by themselves a whole research issue that sits outside the scope of this work, so they are not considered part of the base set of physical level context indicators managed by SALOON.

This concept of a *contact graph*, where the link weight captures the strength of a relationship, is already proposed by other authors such as Hossmann et al. [32]. They also end up extending that model by analyzing, in particular, the role that contact location can have on the identification of inter-community meetings and formation of clusters between nodes with heavier social ties [33]. SALOON's model goes a bit further, focusing not on any group relationship existing within the network, but in the individual relationships themselves.

These physical social-based indicators are used to calculate a specific *affinity* between each node. This value could be matched to their delivery predictability, as defined by some of the context-aware routing and forwarding protocols presented at section 3.3. SALOON uses it to evaluate in what extent it can represent a suitable social classification of each node regarding any other in the network, i.e. if they have a friendship, are acquaintances or strangers. It consolidates this classification and introduces other types of it by leveraging another *affinity* value resulting from an explicit social graph of each user.

6.2 Social Relationships

A *contact graph* allows to measure and evaluate the mobility of nodes, and to some extent the existence of an underlying social relationship between them. Also, the nature of PSNs leads them to assume, most of the times, similar communication patterns to those registered on connections between users in OSNs. The *social graph* of each user, represented by those same nodes, provides another layer of knowledge for the classification of those connections, something that is briefly described by Mtibaa et al. [54].

With the measurement of social relationships between the nodes within a network, comes a series of additional improvements on various opportunistic communication parameters. The ability to separate eligible nodes for message transmission into different social categories like *friends* and *family* allows, for instance, the introduction of a top-level anycast or multicast mechanism, becoming possible to reach a pre-defined group of nodes in a similar fashion, or even the selection of transmission channels providing an higher degree of trust, privacy and integrity. This notion of security also applies for the election of outlying nodes, via transitivity, based on the inherent trust graph built between friends-of-friends.

In addition to the mobility of nodes (who meets whom) and the social relations (who knows whom), there's also a need to assure the existence or not, of explicit interactions between those nodes (who communicates with whom). Most of the research, at this level, in opportunistic networking is largely based on insights from measurements of one of these parameters, but not all three combined. This is mostly due to the fact that the available data finds it hard to map an explicit classification of the underlying relationships and those interactions between their both ends. SALOON addresses this issue by providing a bridge to social services' APIs, via the plugin mechanism described in Chapter 4. OSNs like Facebook register explicit relationships connecting users (family and friends) and produce a set of communication events between them, which should be useful to determine the nature of interactions they might have with each other.

The concept of using these kind of services to assess more efficiently

the communication patterns in opportunistic environments is thoroughly addressed by Hossmann et al. [31]. Their work is based on the deployment of a Facebook application called Stumbl, which uses the service's API to retrieve users' social connections and respective communication events associated to them. The application works by asking users to regularly report whom of their friends (registered on Facebook) they meet face-to-face in a daily basis. It intends to provide a set of empirical results to show how the social relationship between two users and the physical contacts they have with each other converge during time.

Despite the lack of real-world experimentation, the work on Stumbl produced some interesting results about how the social tie type, physical encounters and interactions (generated on Facebook) between nodes, relate to each other. It first found the social tie type has a very strong impact on the frequency, duration and nature of each contact. For instance, meetings between family members are usually long and frequent, between friends are also quite long but less frequent, and between colleagues (as like acquaintances) are much more shorter and less frequent. There is also an explicit dependency between the social tie and the interaction, the collected data establishes that the communication events (Facebook content) generated between friends and family are also much more than those between colleagues or acquaintances, as illustrated by Figure 6.1.

The last, and probably the most important question they try to answer is how the real-world meetings and the Facebook interactions correlate. The fact is that the communication events generated between *stumbl-friends*, i.e. between friends with daily real-world encounters occur about 10 times more often than between other registered friends. This is the main driver for a more than perceived convergence between the *contact-graph* and the *social graph* associated to each node in the network.

The approach proposed by SALOON builds on these evidences to develop its global thesis, which highlights all the properties described as part of a set of social context indicators that should improve opportunistic communications. So, another core contribution of this work relates to the introduction of what the author calls the *affinity graph*. It is some sort of an hypergraph

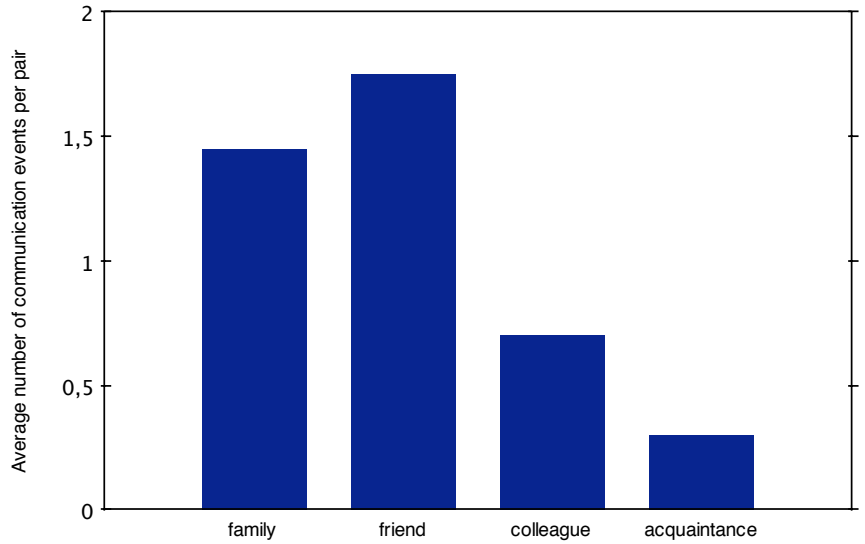


Figure 6.1: Relationship between social tie types and communication events generated on Facebook. Figure adapted from the one presented by Hossmann et al. [31].

that allows to represent the aforementioned convergence between two different layers (social and physical) and is capable of describing formal values for the "real" weight of a relationship between two users within a delay-tolerant network setup.

6.3 Building the Hypergraph

The *affinity graph* is a network representation that intends to provide a richer set of connectivity options for an opportunistic node. As illustrated by Figure 6.2, it works as a conceptual network representation that bridges, for each node, its local implicit communication environment with an overlay network generated by its own social graph. The author calls it an hypergraph[1], since the convergence of the physical and the social layer results in an extension of the graph edges towards other additional nodes, particularly by means of community properties related to the classification of each node and the transitivity principle introduced by friend-of-a-friend and other similar rela-

[1] `http://en.wikipedia.org/wiki/Hypergraph`

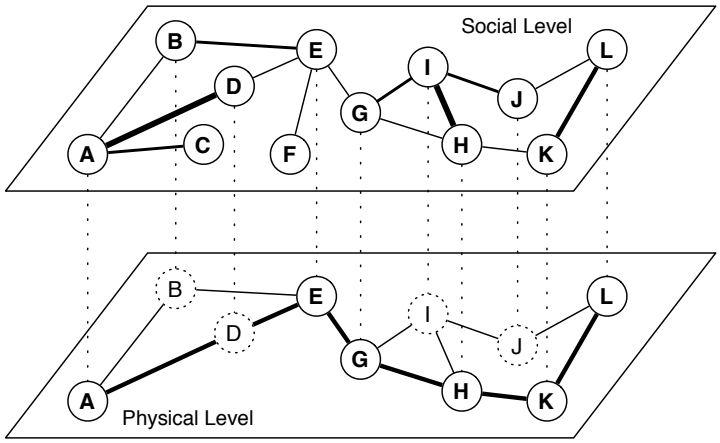


Figure 6.2: Convergence of the social (application-wide) and physical layers.

tionships. It allows to build less-costly (in terms of social distance, security and overhead) message transmission paths between two specific nodes in the same or in a different opportunistic region. Not only that, but it provides the ability to extend the network with, less *trusted* or even previously *unknown* agents by means of a transitivity property exposed, for instance, by friend-of-friends relationships.

6.3.1 Implementation

In order to provide an efficient management of the context indicators associated to the *affinity graph*, SALOON introduces a novel set of underlying mechanisms (deployed on the *mobile component*) coupled with the Bytewalla off-the-shelf infrastructure[2] and the grass-roots plugin system developed for this work's specific proposes. Figure 6.3 describes the top-down placement of the *Context Manager*, a core module in charge of managing the introduced mechanisms. It has the role of providing the following functionality:

1. Interfacing with two specific context sources, plugins and the *Contact Manager*. The latter is a Bytewalla service that detects new contact opportunities and registers (A) the available properties about each of

[2] `http://www.tslab.ssvl.kth.se/csd/projects/092106/sites/default/files/Bytewalla_Final_Report_v1.0.pdf`

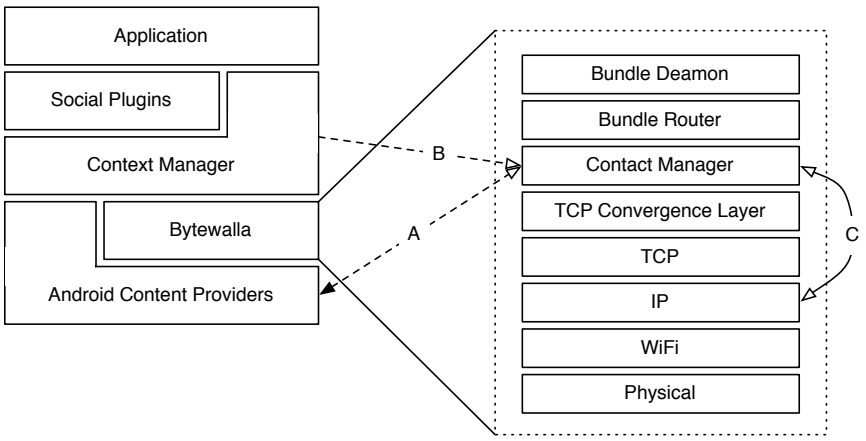


Figure 6.3: A Top-down view regarding the operational behavior of the Context Manager.

those, managing the availability of links and actual connections.

2. Provide an adequate persistence service (A) to store any necessary context parameters obtained from the aforementioned sources and respective providers.

3. Acquire and manage a representation for the *social graph* through the available plugins (B) and for the *contact graph* through the data layer already coupled with the *Contact Manager*, and provide the essential routines to build the resulting *affinity graph*.

4. Work as a listener on the neighbor discovery and advertisement channels (C) and provide a proper bridge to the plugin system to allow the exchange of service-related data, particularly, the access credentials provided by each node to its profile registered at the preferred context providers (OSNs).

One detail worth (a lot) mentioning is the strategic placement of the component to be directly connected with every other layer in the application stack. This cross-layering nature is critical because it allows every context indicator (temporary or persisted) to be directly accessible from both the network and the application layer. An approach like this is essential to support the main use-cases of the SALOON framework, the deployment of

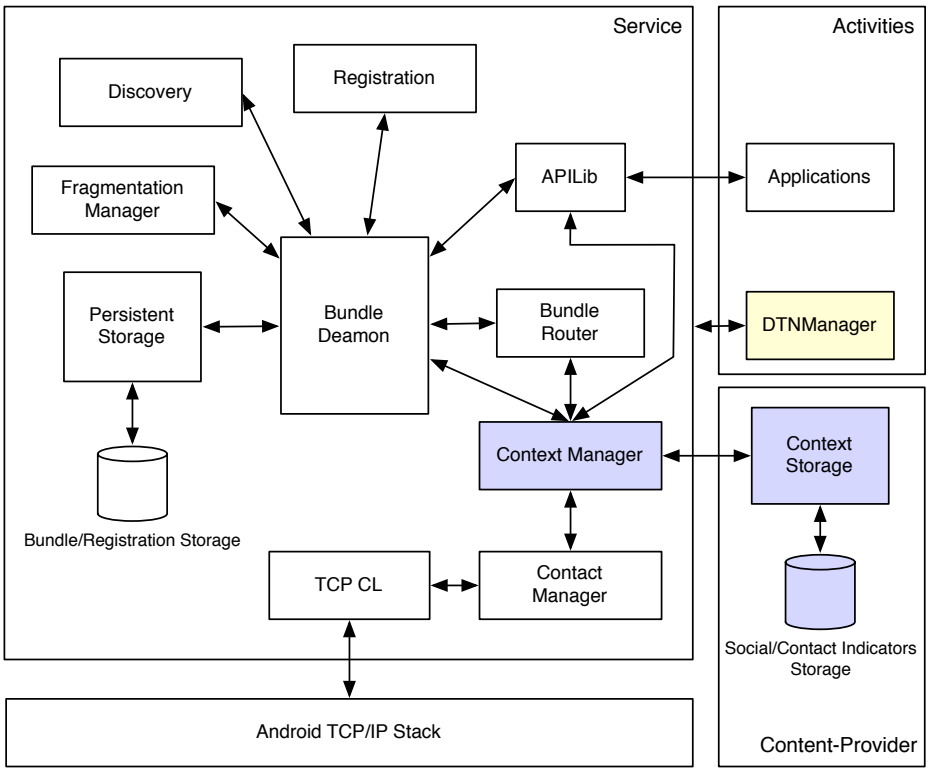


Figure 6.4: Bytewalla's internal layout and the extended contextual components.

context-based opportunistic routing/forwarding protocols or other specific network services and even appropriate consumer-based applications.

The *Context Manager* is integrated internally with the remaining Bytewalla (see Appendix B, Figure B.3) components as depicted in Figure 6.4. An appropriate persistence component was also implemented from the ground-up alongside it (*Context Storage*). Each plugin is also coupled with an existing module called *DTNManager* which provides a front-end interface for the user to configure, monitor and manage the underlying middleware service. The module was extended with a canvas for each context provider on which the user enables/disables any existing plugin and provides access credentials to connect the system to his profile available on the respective providers.

Bytewalla's underlying system (see Appendix B, Figures B.2 and B.3) follows, by default, some standard Android development patterns which are

also, in turn, promoted by the additional components introduced with SALOON. Besides the *DTNManager*, there is another top-level component where applications, or basic Delay-tolerant Network (DTN) services, like *pings* or message transmission tools are integrated by means of proper BP registrations [70]. These front-end utilities are mapped to Android Activities[3], which are the top layer of interaction with users, providing input/output dialogs by means of proper visual components. The core underlying functionality is supported by the *DTNService* (Figure 6.4), implemented as an Android Service[4] which is basically a system-wide process that can potentially execute indefinitely, and provide a long-running availability of DTN features.

SALOON introduces storage facilities for context provisioning through a data model like the one illustrated on Figure A.1 (see Appendix A). A node stores the physical context parameters for a limited number of contacts with each of the others, in this case, resorting only to four different records. The author believes that four records are reasonable enough to determine an up-to-date average value for indicators like the contact duration or inter-contact time, which are in turn, also provided by the storage module and are computed using an Exponentially Weighted Moving Average (EWMA)[5]. This approach was chosen in order to accommodate different needs at different levels. By one hand, average values are enough to build mathematical models to assess low-level communication issues of routing and forwarding, and by the other, tangible values for previous known contact properties (e.g. location) can be extremely useful within the application scope.

During a contact between two nodes A and B, the physical context parameters' provision in B works like the process depicted on Figure 6.5. The *Contact Manager* registers the occurrence of a contact and a subsequent connection established between both nodes and promptly notifies the *Context Manager*. The latter acts as a interface for the storage module, in which it registers some data like the time on which the contact started and its location (obtained via an appropriate location provider, as defined natively

[3] `http://developer.android.com/guide/topics/fundamentals/activities.html`
[4] `http://developer.android.com/guide/topics/fundamentals/services.html`
[5] `http://en.wikipedia.org/wiki/Moving_average`

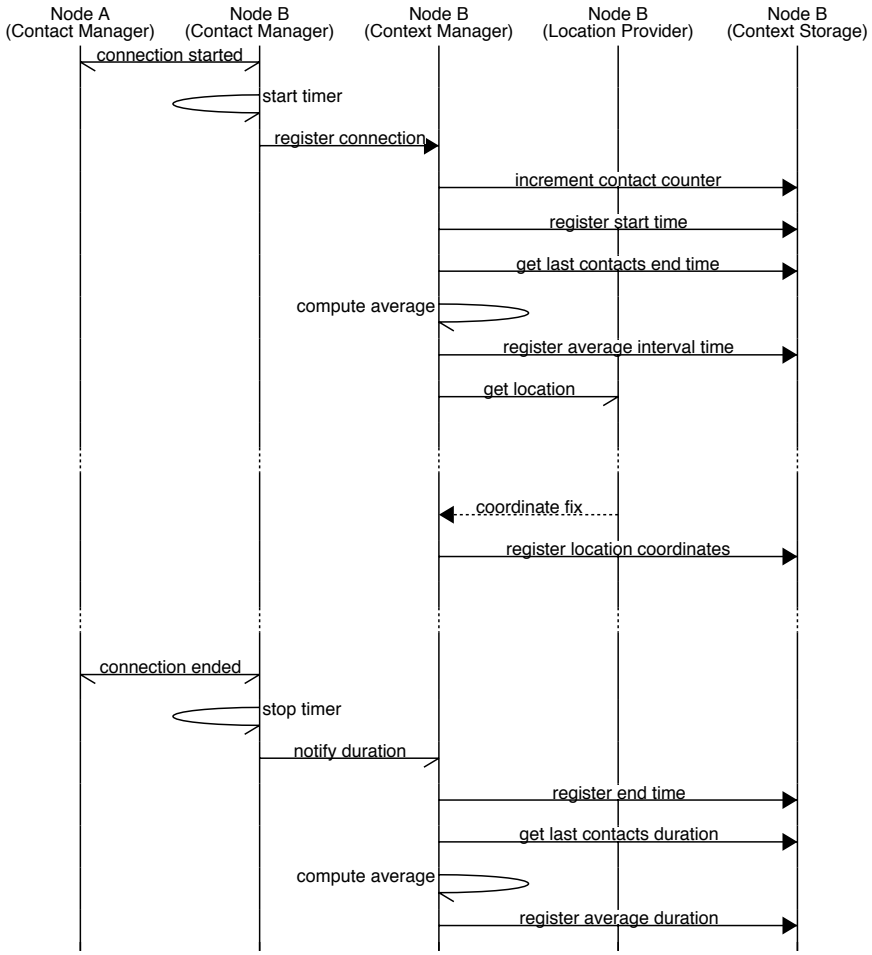


Figure 6.5: Physical context provisioning process in during an opportunistic contact between two nodes.

by Android), removing the oldest entry in the table if the four record limit was exceeded. By the same time, it also retrieves the inter-contact time for the previous registered contacts (when they exist) to which it applies an appropriate EWMA, finally storing the new average value. By the end of the contact, the *Contact Manager* sends the respective termination signal to the *Context Manager* providing it with the registered contact duration. Then, the *Context Manager* repeats the average calculation process, this time for the existing contact durations for previous encounters with node A.

External applications are attached to the system by means of APIs managed by an appropriate internal component (*APILibs*). However, the author

believes that the *Context Manager* generates useful data for other applications not DTN-enabled. In order to avoid additional overhead on the middleware processes, every information related to the context indicators is persisted using Android Content Providers[6], which offer a standard mechanism for applications to share data without exposing the underlying storage, structure and implementation, through the use of Android Intents[7]. This way, the context information is available system-wide and easily accessible to any application or service running on the device.

An application that wishes to operate directly on bundles processed by the middleware should create a specific bundle registration [70] in the *Bundle Deamon* through the same API component. The registration determines that each bundle, after being received by the daemon should be directed to the respective application for further processing. This functionality is completely handled by Bytewalla which creates the necessary resources to accommodate that behavior. It basically saves the registration in an internal storage module that is contextually connected to the daemon and to another storage layer where the actual bundles are saved. The *Registration* component by itself, works as an interface to that storage module and is used by the *Bundle Deamon* to trigger a match between the available registrations.

6.3.2 Use-cases

Back to the *affinity* metric, the author believes it can produce some interesting network effects on how data is transmitted in an opportunistic environment. This is particularly observable on the behavior of routing protocols driven by path selection based on the delivery predictability of each node towards the destination. These two different metrics can then be compared, correlated or used as reference in the absence of their counterparts. Improvements can be achieved not only during the regular communication process, but also in some very specific situations where most of the protocols do not perform properly, for instance, during the formation of a new opportunistic

[6] `http://developer.android.com/guide/topics/providers/content-providers.html`

[7] `http://developer.android.com/guide/topics/intents/intents-filters.html`

region, when nodes initially arrive and start the communication process.

In approaches like the one defined by PRoPHET [44] where the frequency of contacts between nodes plays a crucial role in achieving an appropriate value for the delivery predictability, is even more easier to assess the importance of this contribution. PRoPHET (as other protocols in the same category) defines a process for network formation where nodes build a table of delivery predictabilities towards other nodes they have previously met, so, that table is updated only when a contact with other node is detected. The process is pretty straightforward, each node updates its value for the delivery predictability towards the other taking into account the time passed during their last encounter. This is a rule on PRoPHET, what does not mean that it happens the same way on every other approach. However, as one of the standard protocols defined for delay-tolerant architectures, the author believes that every other alternative should provide a similar basic functionality, thus can be analyzed in a uniform fashion.

In the first encounter between two nodes, their delivery predictability assumes the same default value as defined by each version of the protocol. The problem is that each node has its own importance in the network (particularly due to its social nature), so it should assume a predictability value capable of displaying that importance. The same should happen for any other default action predicted by the underlying build-up mechanism. For instance, each value should be aged (considering the time passed since the last encounter) based on the social nature of each node, and the transitivity property should assume a different importance by the same way.

A reasonable conclusion that could be drawn from this approach is that each protocol should be capable of defining their own formula to achieve an appropriate value for the delivery predictability, i.e. the *affinity*. So, the SALOON approach focuses on providing the necessary tools but does not specify how the thing should be built. One of the reasons it works this way is to avoid any biased behavior towards a particular scheme and allow for other existing routing and forwarding proposals, based on different subsets of contextual parameters, to be tested in the same conditions.

In terms of direct application improvements, the introduction of the *affin-*

ity metric can also produce additional network effects. One particular scenario comes to mind, the extension of Mobile Social Software (MoSoSo) to a more local and implicit preset. Ad-hoc social networks (described in section 2.3) can become a revolutionary concept in a way that they should allow users to meet new people live, and build instant serendipitous relationships. This could be achieved through a set of intelligent recommendation features highlighting homophilic users with common interests (frequent locations, cultural tastes or trends, etc.) and social status, trustable relationships with friends-of-friends, among other functionality. The social nature of the framework should also be useful to the deployment of other potential killer-apps like the ones presented by Lindgren and Hui [45].

Other interesting use-cases where the metric could have a significant impact would be on the creation of trust communities, service-sharing and opportunistic cloud computing inside the same network region [13, 15]. The fact, is that, application scenarios are vast and could result in solving other research issues that are not within the already wide scope of this work. These assumptions are purely theoretic, and they are provided in this document not only as to give some meaning to the framework itself but also to promote additional research problems that might be addressed in the future with the help of this work.

6.4 Summary

This chapter introduces the core concept behind the SALOON framework, context profiling, which builds on the convergence of two different contextual panes, the contact graph and the social graph of each node in an opportunistic network. It introduces each one of those panes and describes what sort of properties could characterize the connections within them. Later, it proposes a *hypergraph* model capable of representing those connections by means of a novel metric called *affinity*. Finally, it describes some specific components added to the framework's infrastructure that provide the capability to deal with these new concepts and allow a better support for a wide range of application scenarios.

Chapter 7

Case Study: Facebook Plugin

All this work around the *affinity graph* would not make sense without an adequate reference plugin to collect useful application-wise (and of course, social-based) context indicators capable of describing the most common patterns of communication events generated between each peer. Since the framework promotes the use of OSNs as the preferred category of context providers for this kind of data, nothing better than plugin for a service like Facebook as a proof-of-concept. Facebook is nowadays a global scale service, used by hundreds of millions of people to communicate with their friends, acquaintances or even strangers (possible new-born friends), promote events, discuss about all kinds of topics and share a wide range of content. The social profile of a person can be built based entirely on the activities she promote in the platform, since much of her life is already transposed to this virtual world.

7.1 Facebook Graph API

Facebook provides an API[1] for authenticated applications to access the profile data when authorized by the respective user. The Graph API, as it is called, exposes a graph representation through a set of *objects* (nodes) and *connections* (edges), for each user registered in the platform, encompassing a plethora of contextual parameters associated to the relationships

[1] `https://developers.facebook.com/docs/reference/api/`

they maintain with other registered users. Each parameter assumes its own distinct importance in a global *weight* for the pairwise connections.

The API establishes a graph traversal process starting with a specific root node that corresponds to an *object* representing the profile of a particular user registered in the platform. So, the developed plugin provides access to a subset of parameters that are exposed directly through that profile, by its associated *objects* or *connections*. The user authorizes an application granting it access to its profile, which is the entry point in the API for the update process of any available context indicator. The first contextual layer faced by any application should be a list of the user's connections, i.e. their implicit relationships (family and friends). The *affinity* is calculated for each of those connections using a subset of contextual parameters.

A user's profile is represented, no wonder, by an object called *User*, which in turn provides a set of connections to other particular associated objects. The plugin extends the social graph towards a specific group of these parameters comprising the user's:

- photos on its own shared album connections or any other *Photo* object directly accessible by the profile (such as other additional photos where he is tagged);
- checkins, at some specific place, represented with *Checkin* objects, accessible through a namesake connection;
- posts made on his or any other user's personal canvas (*wall*), available through a number of different connections such as *feed*, *posts* or *home*, to specific *Post* objects;
- comments or *Comment* objects which are basically responses to posts made by or to the user;
- likes, which are flags on both posts and comments, thus not having any representation object;
- events (as *Event* objects) to which he is attending, again accessible by a namesake connection;

- groups that he has joined, which are defined as *Group* objects reachable through an appropriate direct connection;
- pages representing special entities (*Page* objects) not available through friendship connections but through likes.

To promote a better comprehension about the importance of each parameter, the author defines a particular taxonomy that classifies them, not only in terms of their meaning, but particularly, and more importantly, in terms of what he thinks its their relative weight in a social relationship between two users. As depicted on Table 7.1, this taxonomy builds on a separation of three different categories:

Tagged Context Every situation where one of the users is *tagged* in the other's content (photos, posts or checkins) or any other content (generated by an external user but directly accessible by both profiles) where the two are also *tagged*.

Shared Context Pieces of content generated between both users, such as comments, *likes* and posts made to each other.

Common Context Records which are common to both users, like events they have both attended or are scheduled to attend, checkins at common places and mutual interests regarding music, movies, TV shows, books, groups, organizations and so on.

Each category assumes its own weight on a measurement of the relationship between users. In this case, the plugin promotes an initial partitioning of categories between one that encompasses explicit mutual and shared communication events and other that only can define some similar patterns for both users, given previous (and future predicted) behavior. Based on some of the conclusions provided by Hossmann et al. [31], the author favors the context related to communication events generated directly between the two users, thus, the parameters that in way are capable of determining some sort of synthetic model for real-world encounters between them. Although the plugin enables a customization of these values, it recommends and defaults

Table 7.1: Facebook social context indicator categories.

	Context Categories		
Parameter	Tagged Context	Shared Context	Common Context
Photos	✓		
Checkins	✓		✓
Posts	✓	✓	
Comments		✓	
Likes		✓	
Events			✓
Pages			✓

to a difference of magnitude over 80% for mutual communication and 20% for common or similar patterns.

Mutual communication events comprise those that fit both the *Tagged Context* and *Shared Context*. The author goes even further with this separation by segmenting the 80% global value for this top-level category, into slices with 45% and 35% of magnitude respectively (Figure 7.1). In general, the functionality builds over a much more larger set of context indicators than the ones presented on earlier studies, which also brings a more careful analysis of their role. Therefore, each category internally classifies its related context indicators given their intrinsic nature and own perceived weight.

The plugin favors, particularly, the role of the *Tagged Context*, because the author believes that it comprises the most accurate indicators to assess the scale of physical encounters between two peers. For instance, photo tag context, is measured in two different ways, the number of times each user appears in photos from the other user's albums, and the number of times both of them are tagged in any other registered user's photo. In the former case, the chances of the photo being taken by the respective peer are pretty high, so we can assume their were physically near each other. The same can be directly assessed in the latter case, where they are both explicitly tagged in the same photo, thus, we can also assume they were together at the same place. This is also expected when users tag each other on checkins

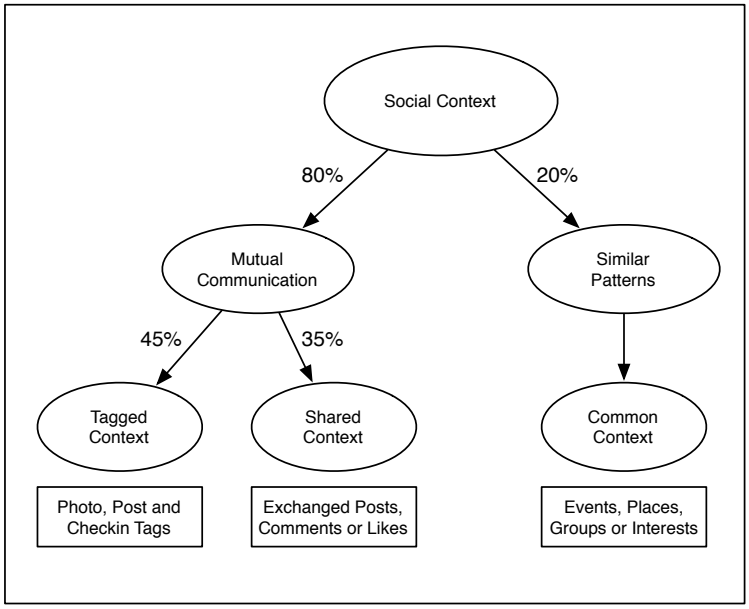


Figure 7.1: Segmentation of context categories based on their importance.

and, even at some extent, on status update posts. Of course, the importance of each parameter is mostly tied to the sense of location, whereas photos and checkins should provide a more reliable source than posts, and so their value decreases in accordance, 50%, 40% and 10%, recommended respectively.

Shared Context comprises essentially what Hossmann et al. consider to be explicit communication events generated between users, such as posts, comments or *likes*. However, and again, the plugin extends this idea by addressing each of those parameters by their perceived importance. To distinguish reciprocal from non-reciprocal connections, i.e. provide a chance to determine if users are actually real-life friends, the author proposes a weighted scheme where posts should assume a greater value against comments and an even greater one against *likes*, for instance, 60%, 30% and 10% respectively. This is considered, particularly to neglect connections as a form of subscription, where an ordinary user is connected to some sort of special user like a celebrity, and ends up generating one way directed content, assuming the celebrity does not bother to interact with him, probably because of the fact that she has endless similar connections to manage.

In a different way, *Common Context* is not seen as the most important

way to measure a relationship of two users that might, be considered good friends or, promote more physical encounters between each other. However, it produces an additional homophilic character, thus a way for users to be matched by their common interests. In this particular case, interests are assessed through groups to which both users belong and by mutual *likes* on special Facebook fan-style pages for movies, music artists, TV shows, organizations, among others. Likewise, the addition of this category also allows for some sort of predictive measurement, since it can comprise data about places (extrapolated from historic patterns) and events—through Répondez S'il Vous Plaît (RSVP)—which users should attend in the future. Again, to highlight the role of location in the perspective of a physical contact, these three different parameters, checkins, events and interests assume a decreasing individual recommended value around 70%, 20% and 10% respectively.

It is important to realize that this plugin only accounts for the context that is directly associated to an explicit connection between two users. There could have been, however, an additional context category comprising the indicators related to friend-of-a-friend relationships, which could be called, for instance, *Transitivity Context*. This category would allow to introduce a new set of communication events, generated by both users, throughout a greater portion of the social graph. The general idea would be to measure transitive relationships by means of context generated at an intermediate user or other different graph node such as fan-pages. However, something like this would carry a lot of scaling issues and introduce undesired overhead which could hamper the progress on other important components of these work. The author decided to leave such features out, possibly for future research work.

7.2 Asynchronous Interface

Since the Graph API exposes a rich set of parameters and it works in an online fashion, i.e. requires an appropriate internet connection for data exchange, the plugin was developed, as recommended on Chapter 4, resorting to an always available server-side component. This component is basically

a lightweight web-service, where each user register its own set of access credentials to authorize and authenticate any subsequent communication with the Facebook API. The choice for this first-time registration process allows the application to maintain an up-to-date and consistent state of the context indicators used by the plugin, by providing an offline access to provider itself. On a request made by the mobile counterpart, the web-service only responds with cached content and avoids any overhead generated by additional network communications with the API.

To avoid any communication restrictions, namely, on the access to some specific data, with the Graph API, the application should prompt the user to provide the necessary permissions needed to allow that access. In this particular case, the user is required to grant access permissions to its profile information, relationships, photo albums, posts, checkins, events, groups and interests. This is part of an authentication and authorization process which results in the generation of a suitable access token that should be used by the application in subsequent requests made to the API. The process is based on the OAuth 2.0 protocol[2] and is handled by a Single-Sign-On (SSO) utility provided by an exclusive Android Software Development Kit (SDK)[3].

An ordinary access token has the limitation of being temporary, and since we are dealing with a specific use-case, where the system would benefit from offline access granted to the web-service, this could result in a serious restriction. Fortunately, Facebook provides an additional permission called *offline access* which when granted by the user, allows the authorized application to access the data available with all the remaining permissions, anywhere in time, by providing it with an long-term access token. The user is prompted to authorize the application when he first enables the plugin through an appropriate interface provided on the *DTNManager* (as described in section 6.3), and every time he removes some required permission (using the account's privacy settings control panel). Otherwise, the SSO functionality ensures the connection with the user profile still exists, without additional authentication from the user. When the authorization is completed, the mobile client

[2] `http://tools.ietf.org/html/draft-ietf-oauth-v2-12`
[3] `https://developers.facebook.com/docs/mobile/android/sso/`

creates or updates (if necessary) the user's registration in the web-service with its profile *id* and the respective access token.

It is only possible for the application to behave gracefully, if it is granted all the required permissions. There is an important tradeoff regarding the strong inverse correlation between the number of permissions an application requests and the number of users that will allow those permissions. The author acknowledges the fact that the plugin requires a considerable set of permission grants, but they are all necessary to achieve good results in the calculation of the *social affinity*, as described in section 7.1.

The registration process fits well with a special feature provided by Facebook, *realtime updates*[4], which enables an application to subscribe for changes in data provided by the API. The subscription process, illustrated on Figure 7.2 is pretty straightforward:

1. The application makes an HTTP *POST* request containing a URL-encoded sequence of two different object types:

   - *user*, when the application wants to be notified about changes in user-specific data; and
   - *permissions*, when the application wants to be notified about changes regarding its own permissions, as defined by the user,

   alongside a list of fields to lookup, a callback URL used for further communications initiated by the API itself, and a validation string;

2. The API reacts by sending an HTTP *GET* request to the web-service including some parameters such as a verification token, used to challenge the application and determine if it is talking to the correct instance;

3. If the verification token matches the validation string sent previously, the application responds with the same code and Facebook registers it as a notification receiver;

[4] `https://developers.facebook.com/docs/reference/api/realtime/`

4. From now then, when a change occurs for the data parameters subscribed by the application, Facebook sends it a notification via an HTTP *POST* request containing a JSON-encoded string (see Appendix C, Listing C.1) that defines the specific subscription type (*user* and/or *permission*), an *id* of each modified object and a list of fields (identified by their name) that were subject to the changes.

5. The application can then respond to those modifications as it sees fit. In this case, it runs a subsequent update process to retrieve the new values of each modified parameter.

As of this writing, Facebook only provides change notifications for a limited set of user connections including only, from the subset established in section 7.1: *feed*, *friends*, *likes* and *checkins*. Then, the application defines two different polling schemes for retrieving updated data:

- partial polling, where only the context indicators related to the available update notifications are retrieved, whenever a new notification arrives; and

- full polling, where all the preferred context indicators are updated in a time window defined as part of the plugin configuration.

The latter involves an update of all the context indicators, just to avoid some bugs detected on the Facebook notification system, which seems to be properly responding only for changes in the application permissions granted by each user.

All the data managed by the web-service is properly cached for each registered user in a specific profile that accounts for every occurrence of the specified context indicators. In this preliminary version of the plugin, the profile is stored in an individual JSON record file per user (Listing 7.1). On every update, the application increments (if that is the case) a counter for each indicator and re-calculates a global value for the application-wide social *affinity*. During a connectivity opportunity, the mobile client can retrieve the data in two different formats, a lightweight object containing only

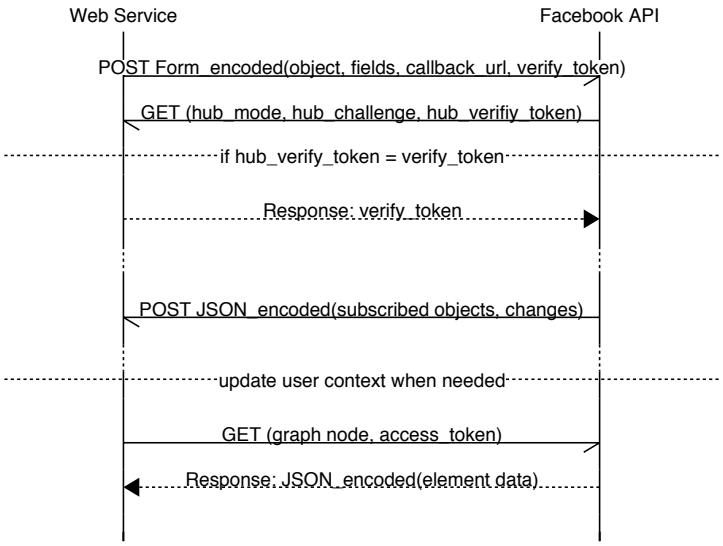


Figure 7.2: Facebook realtime update subscriptions and notifications.

the *affinity* value for each of its relationships, or a more complex one, containing all the data stored as part of its profile in the web-service.

In the mobile counterpart, the data should also be exposed, when possible, by means of an appropriate Android Content Provider (see Appendix A, Figure A.1). This is useful, particularly for a situation where a lot of the context indicators are stored and are meant to be made available to other pervasive applications and services running in the ecosystem. However, the first version of the plugin only considers the minimal data exchange between its two components, thus it only has to store the effective application-wide *social affinity* value towards each other user. The fact that this is only meant to serve as proof-of-concept, and since the framework tries to promote an heterogeneous vision for each context provider and respective plugin, lead to this simplistic approach for its basic structure. It just tries to pragmatically address one of the main issues behind this work: provide a new reference value to match the delivery predictability, as defined by probabilistic routing protocols, and establish a metric for further evaluations.

To accommodate an efficient integration of multiple mobile components, allow a consistent work flow of the web-service and real-time availability of

Listing 7.1: User profile on the Web-service.

```
{
"id": "100000203152130",
"access_token": "AAACEdEose0cBALNCq4z7BYKuWxZAD",
"connections":
[
        {
                "id": "641696741",
                "category": "friend",
                "tagged_photos": 12,
                "tagged_checkins": 8,
                "tagged_posts": 3,
                "shared_posts": 0,
                "shared_comments": 25,
                "shared_likes": 41,
                "common_checkins": 2,
                "common_events": 10,
                "common_groups": 4,
                "common_interests": 3,
                "social_affinity": 0.68
        },
        ...
]
}
```

data, the backend infrastructure was built using Node.js[5]. Node (as it is called most of the times) is designed for extreme scalability in networked applications through a combination of asynchronous I/O, server-side JavaScript (based on Google's V8[6]) and a single execution thread event-driven architecture. It is mainly characterized by a low memory footprint (due to the absence of threads and their increasing cost), high throughput and a simple programming module.

Node proves to be extremely useful to develop asynchronous services like the one described throughout this section. In fact, it fits even better in the vision desired for SALOON for other particular reasons. It provides a seamless integration of JSON data, chosen as the basic information exchange and storage format for most of the framework's components. Also, since

[5] http://nodejs.org/

[6] http://code.google.com/p/v8/

the web-service acts mainly as a proxy between the mobile client and the Graph API, it should be capable to handle the routing of various requests and responses, from or to both of the endpoints. This is done using the *Connect*[7] middleware framework provided natively by Express[8]. Besides the HTTP routing functionality, Express also works as a simple Model–view–controller (MVC) framework that provides a more top-level, thus simpler, approach for the development of networking applications.

7.3 Summary

This chapter constitutes the last step on the bottom-up evaluation of the SALOON framework, and presents its top-level components. Essentially, it describes the main features of a plugin developed as proof-of-concept which interfaces with a particular OSN context source, in this case, the Facebook Graph API. Initially, it provides an analysis of a subset of context indicators chosen as part of the core parameters for the calculation of the *social affinity* (the weight of each connection as registered by the application-level context providers). Later, it illustrates some of the basic features developed as part of the plugin infrastructure and describes how the component is attached to its mobile counterpart.

[7] `http://senchalabs.github.com/connect/`
[8] `http://expressjs.com/`

Chapter 8

Conclusions

Pocket Switched Networks (PSNs) determine a delay-tolerant communication model strictly tied to human behavior, which is in turn extremely influenced by a set of social metrics that measure the relationships between each node in these kind of environments. Such metrics can be obtained by means of some specific context indicators made available through various providers in different setups. This context information can be, in fact, extremely useful on the improvement of opportunistic communication protocols, particularly regarding routing and forwarding processes, and even of the user experience in a set of appropriate applications.

This work introduces the SociAL-based OppOrtunistic Networking (SALOON) framework, which pretends to offer a standard-based testbed for the development of these particular opportunistic networking components. Its basic operational model is a consequence of some basic premises. Given the increasing evolution of wireless and mobile technologies not only on communication devices themselves, but also on the reach of the available infrastructure and network support, opportunistic and delay-tolerant scenarios can be seen as use-cases in very specific situations. By the same fact, it makes sense to consider that every device will have opportunities to connect to the Internet. Those opportunities should be leveraged to enrich the noticeable context on each node by connecting to external or online-available providers. Moreover, this vision should lead to a better convergence on opportunistic

communications research and consequently to a solid compromise about the major role this area must play on achieving a globally connected world [16].

8.1 Insights

Generally speaking, most of the contributions the author proposed to achieve with this work were fulfilled at the end. There are however, some peculiarities that still could be subject of a thorough study, most of them are indeed good topics for future work. The author builds over an explicit issue regarding the social networking phenomenon in opportunistic communications, and tries to promote the fact that the former concept can become extremely influential on the improvement of the latter.

This document builds on a formal thesis that tries to assess how opportunistic environments can benefit from social-based context information, already available or created by the network nodes during the communication process, and how that information can help to understand the way nodes interact with each other. It favors a form of analytical research to vindicate the role of that social-based context on the improvement of some delay-tolerant networking components, particularly, in situations where the network is formed by devices carried by humans, or as their are called, PSNs. Whereas the focus is much over possible improvements on networking facilities such as routing or forwarding, and at a whole different level, on the support for suitable applications.

The work would be even more interesting if there was the possibility to provide additional strategies for the application of concepts such as the *affinity* metric, based on more accurate measures of its impact, particularly on routing and forwarding schemes, which could be assessed more formally using different methodologies such as testing in a proper simulation environment. This kind of argumentative research is actually one of the main drivers of this work. There is a lack (or even total absence, considering that the few existing traces are private) of appropriate datasets beholding most of the proposed context indicators in its genesis. The framework introduced in this document was built also to provide an adequate experimentation system where

these datasets can be generated and then used to a more in-depth measuring starting from the glimpsed insights promoted by the author.

8.2 Future Work

The SALOON framework is intended to constitute a first step towards a unified research method for the development of opportunistic communication tools. The goal is provide a general architecture that allows real-world experimentations and collection of richer datasets, specially based on the managed context indicators, that can provide a better substantiation for empirical conclusions like the ones presented by Hossmann et al. [31], Mtibaa et al. [54] and others. Such datasets would also prove to be critical in the assessment and measurement of the impact, that the social context indicators presented with this work, can actually have particularly on the improvement of opportunistic routing and forwarding protocols, like the author tries to promote with some contributions. Tangible values gathered by means of simulations and adequate testing provide a much more valid technique to analyze which components of the communication are most affected with the introduction of these novel metrics.

In summary, the author would love to see his work triggering the creation of a larger set of components that could be attached to the framework. So, he intends to promote the development of:

- new middleware plugins for various context providers (local or global ones);
- consumer applications that leverage the data available at those providers making them eligible to become a killer-app for opportunistic environments [45];
- novel low-level delay-tolerant networking services, such as routing protocols, capable of making an even more *informed* management of the network context based on different metrics provided by the *hypergraph* model.

The framework itself would benefit from further enhancements, particularly regarding the creation of a security model capable of providing a certain level of data privacy on message transmission and storage, through appropriate encryption mechanisms. Another thing that could be exploited even further is probably the use of additional context indicators, not restricted to a social-nature (e.g. resource-based, like depicted on section 6.1), which would enable a larger scope of development and consequently additional research work on different kind of opportunistic scenarios.

In the same way, the proof-of-concept plugin, that is strictly bound to the context of a direct social relationship between two Facebook users, could be extended to favor other additional indicators capable of describing transitive connections. This would relate not only to the importance of intermediate friend-of-a-friend users but also to the activity promoted by both endpoints of a relationship with an existing and shared friend. This concept of *Transitivity Context*, as glimpsed on section 7.1, could comprise a lot of the already privileged parameters like posts, comments and likes with that shared friend.

Bibliography

[1] L. Backstrom, D. Huttenlocher, J. Kleinberg, and X. Lan. Group formation in large social networks: membership, growth, and evolution. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '06, pages 44–54, New York, NY, USA, 2006. ACM.

[2] A. Beach, M. Gartrell, S. Akkala, J. Elston, J. Kelley, K. Nishimoto, B. Ray, S. Razgulin, K. Sundaresan, B. Surendar, M. Terada, and R. Han. Whozthat? evolving an ecosystem for context-aware mobile social networks. *Network, IEEE*, 22(4):50–55, July/August 2008.

[3] A. Beach, M. Gartrell, X. Xing, R. Han, Q. Lv, S. Mishra, and K. Seada. Fusing mobile, sensor, and social data to fully enable context-aware computing. In *Proceedings of the Eleventh Workshop on Mobile Computing Systems & Applications*, HotMobile '10, pages 60–65, New York, NY, USA, 2010. ACM.

[4] C. Boldrini, M. Conti, J. Jacopini, and A. Passarella. Hibop: a history based routing protocol for opportunistic networks. In *WoWMoM 2007. IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks, 2007.*, pages 1 –12, 2007.

[5] C. Boldrini, M. Conti, and A. Passarella. Exploiting users' social relations to forward data in opportunistic networks: The hibop solution. *Pervasive and Mobile Computing*, 4(5):633–657, 2008.

[6] J. Burgess, B. Gallagher, D. Jensen, and B. N. Levine. Maxprop:

Routing for vehicle-based disruption-tolerant networks. In *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, pages 1–11, April 2006.

[7] B. Burns, O. Brock, and B. Levine. Mv routing and capacity building in disruption tolerant networks. In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, volume 1, pages 398–408, March 2005.

[8] C. Caini, P. Cornice, R. Firrincieli, M. Livini, and D. Lacamera. Dtn meets smartphones: Future prospects and tests. In *Wireless Pervasive Computing (ISWPC), 2010 5th IEEE International Symposium on*, pages 355 –360, May 2010.

[9] I. Carreras and D. Tacconi. U-hopper: User-centric heteogeneous opportunistic middleware. In *Bio-Inspired Models of Network, Information and Computing Systems, 2007. Bionetics 2007. 2nd*, pages 302 –305, 2007.

[10] V. Cerf, S. Burleigh, A. Hooke, L. Torgerson, R. Durst, K. Scott, K. Fall, and H. Weiss. Delay-Tolerant Network Architecture: The Evolving Interplanetary Internet. Draft, August 2002.

[11] V. Cerf, S. Burleigh, A. Hooke, L. Torgerson, R. Durst, K. Scott, K. Fall, and H. Weiss. Delay-Tolerant Networking Architecture. RFC 4838 (Informational), April 2007.

[12] A. Chaintreau, P. Hui, J. Crowcroft, C. Diot, R. Gass, and J. Scott. Impact of human mobility on opportunistic forwarding algorithms. *Mobile Computing, IEEE Transactions on*, 6(6):606 –620, june 2007.

[13] M. Conti and M. Kumar. Opportunities in opportunistic computing. *Computer, IEEE*, 43(1):42–50, January 2010.

[14] M. Conti, J. Crowcroft, S. Giordano, P. Hui, H. Anh Nguyen, and A. Passarella. Routing issues in opportunistic networks. In B. Garbinato,

L. Rodrigues, and H. Miranda, editors, *Middleware for Network Eccentric and Mobile Applications*, chapter 6, pages 121–147. Springer Publishing Company, Incorporated, 1st edition, 2009.

[15] M. Conti, S. Giordano, M. May, and A. Passarella. From opportunistic networks to opportunistic computing. *Communications Magazine, IEEE*, 48(9):126–139, September 2010.

[16] J. Crowcroft, E. Yoneki, P. Hui, and T. Henderson. Promoting tolerance for delay tolerant network research. *SIGCOMM Computer Communication Review*, 38:63–68, September 2008.

[17] E. M. Daly and M. Haahr. Social network analysis for routing in disconnected delay-tolerant MANETs. In *MobiHoc '07: Proceedings of the 8th ACM international symposium on Mobile ad hoc networking and computing*, pages 32–40, Montreal, Quebec, Canada, 2007. ACM Press.

[18] J. Davidsen, H. Ebel, and S. Bornholdt. Emergence of a small world from local interactions: Modeling acquaintance networks. *Phys. Rev. Lett.*, 88(12):128701 (1–4), March 2002.

[19] A. K. Dey. Understanding and using context. *Personal Ubiquitous Computing*, 5:4–7, January 2001.

[20] J. Dixit. The artful- and mobile-dodger [location-based social networking]. *Spectrum, IEEE*, 42(3):59–60, March 2005.

[21] R. J. D'Souza and J. Jose. Routing approaches in delay tolerant networks: A survey. *International Journal of Computer Applications*, 1 (17):8–14, February 2010.

[22] N. Eagle and A. Pentland. Social serendipity: mobilizing social software. *Pervasive Computing, IEEE*, 4(2):28–34, April–June 2005.

[23] K. Fall. A delay-tolerant network architecture for challenged internets. In *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM '03, pages 27–34, New York, NY, USA, 2003. ACM.

[24] K. Fall and S. Farrell. Dtn: an architectural retrospective. *Selected Areas in Communications, IEEE Journal on*, 26(5):828–836, June 2008.

[25] S. Farrell, V. Cahill, D. Geraghty, I. Humphreys, and P. McDonald. When tcp breaks: Delay- and disruption- tolerant networking. *Internet Computing, IEEE*, 10(4):72 –78, July/August 2006.

[26] D. Goodman, J. Borras, N. Mandayam, and R. Yates. Infostations: a new system model for data and messaging services. In *Vehicular Technology Conference, 1997 IEEE 47th*, volume 2, pages 969–973, May 1997.

[27] M. Grossglauser and D. N. C. Tse. Mobility increases the capacity of ad hoc wireless networks. *IEEE/ACM Trans. Netw.*, 10:477–486, August 2002.

[28] M. Grossglauser and M. Vetterli. Locating nodes with ease: last encounter routing in ad hoc networks through mobility diffusion. In *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, volume 3, pages 1954–1964, March 2003.

[29] F. Guidec and Y. Mahéo. Opportunistic Content-Based Dissemination in Disconnected Mobile Ad Hoc Networks. In *International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies (UBICOMM 2007)*, pages 49–54, Papeete, French Polynesia (Tahiti), November 2007. IEEE Computer Society Press.

[30] J. Haillot and F. Guidec. A Protocol for Content-Based Communication in Disconnected Mobile Ad Hoc Networks. In *IEEE 22nd International Conference on Advanced Information Networking and Applications (AINA'08)*, pages 188–195, Okinawa, Japan, March 2008. IEEE CS.

[31] T. Hossmann, F. Legendre, G. Nomikos, and T. Spyropoulos. Stumbl: Using facebook to collect rich datasets for opportunistic networking research. In *The Fifth IEEE WoWMoM Workshop on Autonomic and Opportunistic Communications (AOC 11)*, Lucca, Italy, June 2011. IEEE.

[32] T. Hossmann, T. Spyropoulos, and F. Legendre. A complex network analysis of human mobility. In *Third International Workshop on Network Science for Communication Networks (NetSciCom 11)*, Shanghai, China, April 2011. IEEE.

[33] T. Hossmann, T. Spyropoulos, and F. Legendre. Putting contacts into context: Mobility modeling beyond inter-contact times. In *Twelfth ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 11)*, Paris, France, May 2011. ACM.

[34] P. Hui and J. Crowcroft. Bubble Rap: Forwarding in small world DTNs in ever decreasing circles. Technical Report UCAM-CL-TR-684, University of Cambridge Computer Laboratory, May 2007.

[35] P. Hui and J. Crowcroft. How small labels create big improvements. In *Pervasive Computing and Communications Workshops, 2007. PerCom Workshops '07. Fifth Annual IEEE International Conference on*, pages 65–70, March 2007.

[36] P. Hui, J. Crowcroft, and E. Yoneki. Bubble rap: social-based forwarding in delay tolerant networks. In *Proceedings of the 9th ACM international symposium on Mobile ad hoc networking and computing*, MobiHoc '08, pages 241–250, New York, NY, USA, 2008. ACM.

[37] T. Hyyryläinen, T. Kärkkäinen, C. Luo, V. Jaspertas, J. Karvo, and J. Ott. Opportunistic email distribution and access in challenged heterogeneous environments. In *Proceedings of the second ACM workshop on Challenged networks*, CHANTS '07, pages 97–100, New York, NY, USA, 2007. ACM.

[38] Y. Iwatani. Love: Japanese Style. *Wired Magazine*, November 1998.

[39] T. Karagiannis, J.-Y. Le Boudec, and M. Vojnović. Power law and exponential decay of inter contact times between mobile devices. In *Proceedings of the 13th annual ACM international conference on Mobile computing and networking*, MobiCom '07, pages 183–194, New York, NY, USA, 2007. ACM.

[40] R. L. Keeney and H. Raiffa. *Decisions with Multiple Objectives: Preferences and Value Tradeoffs*. Cambridge University Press, 1993.

[41] R. Kumar, J. Novak, and A. Tomkins. Structure and evolution of online social networks. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '06, pages 611–617, New York, NY, USA, 2006. ACM.

[42] J. Leguay, T. Friedman, and V. Conan. Evaluating mobility pattern space routing for dtns. In *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, pages 1–10, April 2006.

[43] N. Li and G. Chen. Sharing location in online social networks. *Network, IEEE*, 24(5):20–25, September/October 2010.

[44] A. Lindgren and A. Doria. Probabilistic Routing Protocol for Intermittently Connected Networks. Draft, March 2006.

[45] A. Lindgren and P. Hui. The quest for a killer app for opportunistic and delay tolerant networks (invited paper). In *Proceedings of the 4th ACM workshop on Challenged networks*, CHANTS '09, pages 59–66, New York, NY, USA, 2009. ACM.

[46] A. Lindgren, A. Doria, and O. Schelén. Probabilistic routing in intermittently connected networks. In P. Dini, P. Lorenz, and J. N. d. Souza, editors, *Service Assurance with Partial and Intermittent Resources*, volume 3126 of *Lecture Notes in Computer Science*, pages 239–254. Springer Berlin / Heidelberg, 2004.

[47] G. Lugano, J. Kyppõ, and P. Saariluoma. Designing people's interconnections in mobile social networks. In *Proceedings of the First International Conference on Multidisciplinary Information Sciences and Technologies*, InScit 2006, pages 500–504, Badajoz, Spain, October 2006. Open Institute of Knowledge.

[48] A. Martín-Campillo, J. Crowcroft, E. Yoneki, R. Martí, and C. Martínez-García. Using haggle to create an electronic triage tag. In *Proceedings of the Second International Workshop on Mobile Opportunistic Networking*, MobiOpp '10, pages 167–170, New York, NY, USA, 2010. ACM.

[49] A. McMahon and S. Farrell. Delay- and disruption-tolerant networking. *Internet Computing, IEEE*, 13(6):82–87, November/December 2009.

[50] A. Miklas, K. Gollu, K. Chan, S. Saroiu, K. Gummadi, and E. de Lara. Exploiting Social Interactions in Mobile Systems. In *Proceedings of UbiComp 2007*, pages 409–428, September 2007.

[51] S. Milgram. The Small World Problem. *Psychology Today*, 2:60–67, 1967.

[52] A. Mislove, M. Marcon, K. P. Gummadi, P. Druschel, and B. Bhattacharjee. Measurement and analysis of online social networks. In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, IMC '07, pages 29–42, New York, NY, USA, 2007. ACM.

[53] A. Moghadam, S. Srinivasan, and H. Schulzrinne. 7ds - a modular platform to develop mobile disruption-tolerant applications. In *Proceedings of the 2008 The Second International Conference on Next Generation Mobile Applications, Services, and Technologies*, pages 177–183, Washington, DC, USA, 2008. IEEE Computer Society.

[54] A. Mtibaa, A. Chaintreau, J. LeBrun, E. Oliver, A.-K. Pietilainen, and C. Diot. Are you moved by your social network application? In *Proceedings of the first workshop on Online social networks*, WOSP '08, pages 67–72, New York, NY, USA, 2008. ACM.

[55] A. Mtibaa, M. May, C. Diot, and M. Ammar. Peoplerank: Social opportunistic forwarding. In *INFOCOM, 2010 Proceedings IEEE*, pages 1–5, March 2010.

[56] M. Musolesi and C. Mascolo. A community based mobility model for ad hoc network research. In *Proceedings of the 2nd international workshop*

on Multi-hop ad hoc networks: from theory to reality, REALMAN '06, pages 31–38, New York, NY, USA, 2006. ACM.

[57] M. Musolesi and C. Mascolo. Mobility models for systems evaluation. In B. Garbinato, L. Rodrigues, and H. Miranda, editors, *Middleware for Network Eccentric and Mobile Applications*, chapter 3, pages 43–62. Springer Publishing Company, Incorporated, 1st edition, 2009.

[58] M. Musolesi and C. Mascolo. Car: Context-aware adaptive routing for delay-tolerant mobile networks. *Mobile Computing, IEEE Transactions on*, 8(2):246–260, February 2009.

[59] M. Musolesi, P. Hui, C. Mascolo, and J. Crowcroft. Writing on the clean slate: Implementing a socially-aware protocol in haggle. In *WoWMoM 2008. IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks, 2008.*, pages 1–6, 2008.

[60] H. A. Nguyen, S. Giordano, and A. Puiatti. Probabilistic routing protocol for intermittently connected mobile ad hoc network (propicman). In *WoWMoM 2007. IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks, 2007.*, pages 1 –6, 2007.

[61] J. Ott and M. J. Pitkänen. Dtn-based content storage and retrieval. In *WoWMoM 2007. IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks, 2007.*, pages 1–7, June 2007.

[62] L. Pelusi, A. Passarella, and M. Conti. Opportunistic networking: data forwarding in disconnected mobile ad hoc networks. *Communications Magazine, IEEE*, 44(11):134–141, November 2006.

[63] A.-K. Pietilainen, E. Oliver, J. LeBrun, G. Varghese, J. Crowcroft, and C. Diot. Experiments in mobile social networking. Technical report, Technical Report CR-PRL-2008-02-0003, Thomson, 2008.

[64] A.-K. Pietiläinen, E. Oliver, J. LeBrun, G. Varghese, and C. Diot. Mobiclique: middleware for mobile social networking. In *Proceedings of the*

2nd ACM workshop on Online social networks, WOSN '09, pages 49–54, New York, NY, USA, 2009. ACM.

[65] M. J. Pitkänen and J. Ott. Redundancy and distributed caching in mobile dtns. In *Proceedings of 2nd ACM/IEEE international workshop on Mobility in the evolving internet architecture*, MobiArch '07, pages 8:1–8:7, New York, NY, USA, 2007. ACM.

[66] M. J. Pitkänen and J. Ott. Enabling opportunistic storage for mobile dtns. *Pervasive and Mobile Computing*, 4(5):579–594, 2008.

[67] J. M. Pujol, A. L. Toledo, and P. Rodriguez. Fair routing in delay tolerant networks. In *INFOCOM 2009, IEEE*, pages 837 –845, 2009.

[68] N. Sastry, K. Sollins, and J. Crowcroft. Delivery properties of human social networks. In *INFOCOM 2009, IEEE*, pages 2586 –2590, 2009.

[69] J. Scott, J. Crowcroft, P. Hui, and C. Diot. Haggle: a networking architecture designed around mobile users. In *Proceedings of the Third Annual Conference on Wireless On-demand Network Systems and Services*, pages 78–86, 2006.

[70] K. Scott and S. Burleigh. Bundle Protocol Specification. RFC 5050 (Experimental), November 2007.

[71] A. Sheth. Computing for human experience: Semantics-empowered sensors, services, and social computing on the ubiquitous web. *Internet Computing, IEEE*, 14(1):88–91, January/February 2010.

[72] I. Smith. Social-mobile applications. *Computer, IEEE*, 38(4):84–85, April 2005.

[73] T. Spyropoulos, K. Psounis, and C. S. Raghavendra. Efficient routing in intermittently connected mobile networks: the multiple-copy case. *IEEE/ACM Trans. Netw.*, 16:77–90, February 2008.

[74] T. Spyropoulos, R. N. Rais, T. Turletti, K. Obraczka, and A. Vasilakos. Routing for disruption tolerant networks: taxonomy and design. *Wirel. Netw.*, 16:2349–2370, November 2010.

[75] J. Su, J. Scott, P. Hui, J. Crowcroft, E. de Lara, C. Diot, A. Goel, M. Lim, and E. Upton. Haggle: Seamless networking for mobile applications. In J. Krumm, G. Abowd, A. Seneviratne, and T. Strang, editors, *UbiComp: Ubiquitous Computing*, volume 4717 of *Lecture Notes in Computer Science*, pages 391–408. Springer Berlin/Heidelberg, 2007.

[76] J. Su, J. Scott, P. Hui, E. Upton, M. H. Lim, C. Diot, J. Crowcroft, A. Goel, and E. de Lara. Haggle: Clean-slate networking for mobile devices. Technical Report UCAM-CL-TR-680, University of Cambridge, Computer Laboratory, January 2007.

[77] M. Terry, E. D. Mynatt, K. Ryall, and D. Leigh. Social net: using patterns of physical proximity over time to infer shared interests. In *Extended abstracts on Human factors in computing systems*, CHI '02, pages 816–817, New York, NY, USA, 2002. ACM.

[78] P. Th. Eugster, B. Garbinato, and A. Holzer. Middleware support for context-aware applications. In B. Garbinato, L. Rodrigues, and H. Miranda, editors, *Middleware for Network Eccentric and Mobile Applications*, chapter 14, pages 305–322. Springer Publishing Company, Incorporated, 1st edition, 2009.

[79] J. Thom-Santelli. Mobile social software: Facilitating serendipity or encouraging homogeneity? *Pervasive Computing, IEEE*, 6(3):46–51, July–September 2007.

[80] A. Vahdat and D. Becker. Epidemic Routing for Partially Connected Ad Hoc Networks. Cs, Department of Computer Science, Duke University, Durham, NC, 2000, June 2000.

[81] M. von Arb, M. Bader, M. Kuhn, and R. Wattenhofer. Veneta: Serverless friend-of-friend detection in mobile social networking. In *WIMOB*

'08. IEEE International Conference on Wireless and Mobile Computing, Networking and Communications, pages 184–189, 2008.

[82] L. Wood, W. M. Eddy, and P. Holliday. A bundle of problems. In *Aerospace conference, 2009 IEEE*, pages 1–17, March 2009.

[83] L. Wood, P. Holliday, D. Floreani, and W. M. Eddy. Sharing the dream. In *Ultra Modern Telecommunications Workshops, 2009. ICUMT '09. International Conference on*, pages 1–2, October 2009.

[84] N. D. Ziv and B. Mulloth. An exploration on mobile social networking: Dodgeball as a case in point. In *ICMB '06. International Conference on Mobile Business*, pages 21–21, June 2006.

Appendix A

Data model

A.1 Context Provisioning

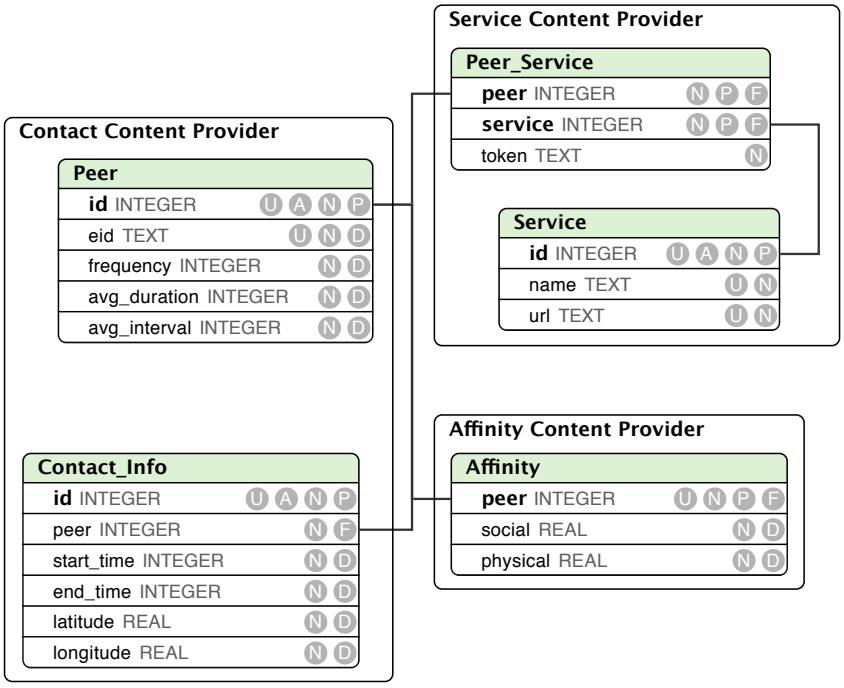


Figure A.1: Standard context data model used by SALOON plugins.

Appendix B

Bytewalla Support

This appendix presents some existing components and functionality provided by the Bytewalla implementation. Figures are adapted from content available throughout the project's deliverables and documentation[1].

B.1 Reference Features

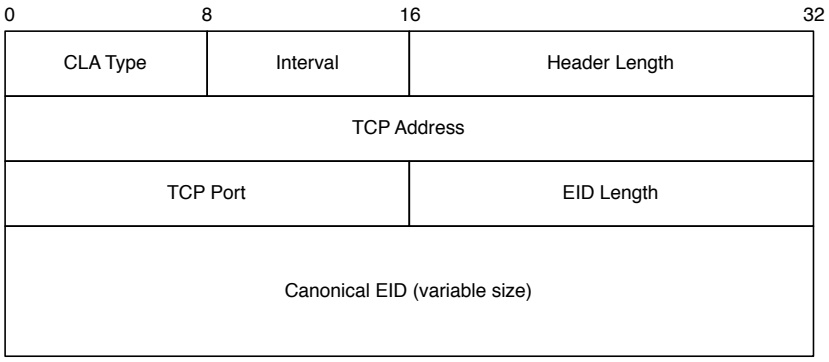


Figure B.1: Discovery header as the beacon format on Bytewalla.

[1] `http://www.tslab.ssvl.kth.se/csd/projects/092106/`

B.2 Major Software Components

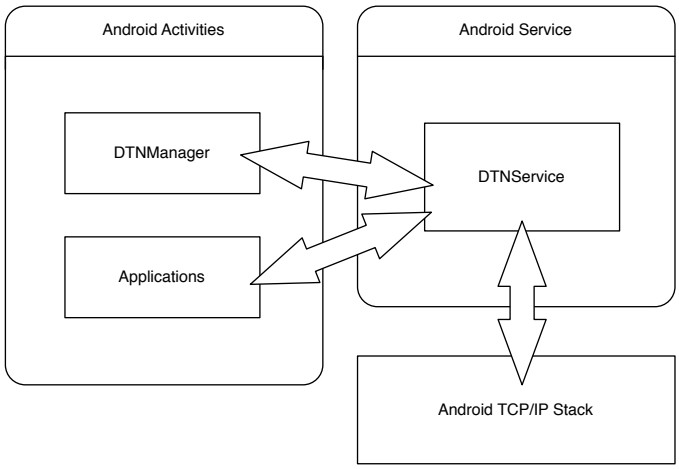


Figure B.2: Bytewalla major software components.

B.3 Internal Design

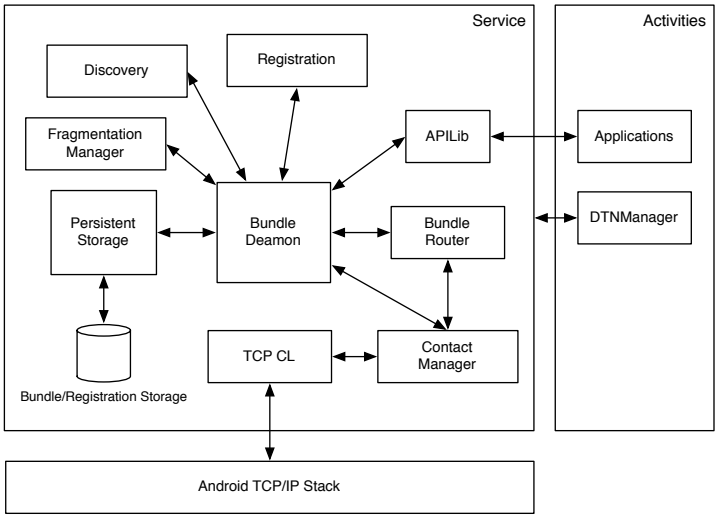


Figure B.3: Bytewalla internal module's design.

Appendix C

Facebook Support

C.1 Realtime update notification

Listing C.1: Facebook realtime updates notification format.

```
{
"object": "user",
"entry":
[
    {
        "uid": 1335845740,
        "changed_fields":
        [
            "name",
            "picture"
        ],
        "time": 232323
    },
    {
        "uid": 1234,
        "changed_fields":
        [
            "friends"
        ],
        "time": 232325
    }
]
}
```