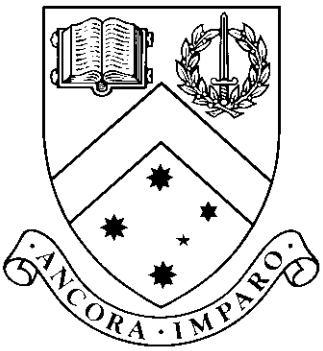
Service-Oriented Mobile Social Network in Proximity

by

Chii Chang, MIT



Thesis

Submitted by Chii Chang

in fulfillment of the Requirements for the Degree of

Doctor of Philosophy (0190)

Supervisor: Sea Ling

Associate Supervisor: Satish Narayana Srirama

Caulfield School of Information Technology
Monash University

November, 2013

Copyright Notices

Notice 1

Under the Copyright Act 1968, this thesis must be used only under the normal conditions of scholarly fair dealing. In particular no results or conclusions should be extracted from it, nor should it be copied or closely paraphrased in whole or in part without the written consent of the author. Proper written acknowledgement should be made for any assistance obtained from this thesis.

Notice 2

I certify that I have made all reasonable efforts to secure copyright permissions for third-party content included in this thesis and have not knowingly added copyright content to my work without the owner's permission.

© Copyright

by

Chii Chang

2013

Contents

List of Tables

List of Figures

List of Abbreviations

AMSNP	Adaptive Mediation Framework for Mobile Social Network in Proximity
API	Application Programming Interface
BPMN	Business Process Modelling Notation
CC	Content Consumer
CloudUtil	Cloud Utility Service
CORBA	Common Object Request Broker Architecture
CP	Content Provider
CPI	Cost-Performance Index
CPU	Central Processing Unit
CRA	Central Registry-based Architecture
DCOM	Distributed Component Object Model
DHT	Distributed Hash Table
DNS	Domain Name System
DPWS	Device Profile for Web Services
EC2	Elastic Compute Cloud
ESB	Enterprise Service Bus
FOAF	Friend Of A Friend
FTP	File Transfer Protocol
GAE	Google App Engine
HTTP	Hypertext Transfer Protocol
IaaS	Infrastructure as a Service
ID	Identification
IETF	Internet Engineering Task Force

IP	Internet Protocol
ISP	Internet Service Provider
JSON	JavaScript Object Notation
LAN	Local Area Network
LBSN	Location-Based Social Network
MANET	Mobile Ad Hoc Network
MCC	Mobile Cloud Computing
mDNS	Multicast Domain Name System
MHMWS	Mobile-Device-Hosted MWS
MP2P	Mobile Peer-to-Peer
MSN	Mobile Social Network
MSNP	Mobile Social Network in Proximity
MWS	Mobile Web Service
OMG	Object Management Group
OS	Operating System
OWL	Web Ontology Language
P2P	Peer-to-Peer
PaaS	Platform as a Service
PBMSN	Proximal-based MSN
PSC list	Previous Interacting Service Consumers
QoS	Quality of Service
RAM	Random Access Memory
RD	Reputation Rating Data
RDF	Resource Description Framework
REST	Representational State Transfer
ROM	Read-Only Memory
RR	Recommended References
RSS	Rich Site Summary
SaaS	Software as a Service

SAWSDL	Semantic Annotation for WSDL and XML Schema
SDM	Service Description Metadata
SLP	Service Location Protocol
SMTP	Simple Mail Transfer Protocol
SNS	Social Network Service(s)
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SP	Service Provider
SPR	Service Provider Ratings
SSDP	Simple Service Discovery Protocol
TCP	Transmission Control Protocol
UDDI	Universal Description, Discovery and Integration
UDP	User Datagram Protocol
UPnP	Universal Plug and Play
URL	Uniform Resource Locator
W3C	World Wide Web Consortium
WADL	Web Application Description Language
WAN	Wide Area Network
WfMS	Workflow Management Systems
WLAN	Wireless Local Area Network
WS	Web Service
WS-BPEL	Web Services Business Process Execution Language
WSDL	Web Services Description Language
XML	Extensible Markup Language
XMPP	Extensible Messaging and Presence Protocol
Zeroconf	Zero-configuration networking

Service-Oriented Mobile Social Network in Proximity

Chii Chang, MIT
[redacted]
Monash University, 2013

Supervisor: Sea Ling
Associate Supervisor: Satish Narayana Srirama

Abstract

Accessing online Social Network Services (SNS) such as Facebook, Twitter, Google+ from mobile devices (e.g., Smart phones) has become a daily activity to many users. With the powerful and rich capabilities of recent mobile devices, users can produce heterogeneous content such as photos, videos, audio, and share them with their social groups. From the users' perspective, mobile devices are no longer simple client-side medium to acquire content from online content providers, but they have become content providers as well.

Embedded Web server enhances the usefulness of mobile devices in terms of content sharing. With embedded Web server, mobile devices can provide various mobile Web services (MWS). Moreover, users are capable of establishing a new form of social network in the mobile peer-to-peer (P2P) manner, in which a group of proximal users, who whether they know one another or not, can perform various SNS activities within the same wireless network. The term—Mobile Social Network in Proximity (MSNP)—has been used in this thesis to illustrate such an environment. MSNP is a composite social network environment in which MSNP participants can either share content data directly from their MWS or they can simply conduct their MWS to provide links that redirect content requesters to retrieve content from the providers' SNS spaces.

In recent years, realising MSNP became an interested topic to many researchers. Since the fundamental infrastructure of MSNP is based on mobile P2P network,

enabling MSNP inherits many common challenges of mobile P2P applications in terms of mobility issues, hardware resource constraint issues, security-related issues, and so on. While the existing MSNP-related works were designed for particular applications and platforms, this thesis aims to provide a solution to enable a generic service-oriented MSNP environment, in which the mobile users can participate in the environment using heterogeneous devices and platforms.

In this thesis, two related major challenges—service discovery and resource management of MSNP—have been addressed.

Service discovery in MSNP faces latency issues which are caused by the long makespan of service discovery and identification of trustworthiness of the service providers. The issues have been analysed in this thesis. Corresponding solutions—*context-aware proactive service discovery for MSNP* and *the lightweight trustworthy service discovery for MSNP*—are presented, together with experimental evaluation.

Since MSNP is a composite environment, in which local services in mobile device and remote online Web services are involved, resource management becomes a crucial task in service discovery and also in other MSNP activities. The hardware resource of mobile device and the availability of remote online Web services, can influence the efficiency of how a mobile device performs its activities in MSNP. This thesis has analysed the challenge in resource management of MSNP and proposed a corresponding mediation framework—*Adaptive Mediation Framework for MSNP (AMSNP)*—that applied workflow technique and Enterprise Service Bus architecture to manage resources dynamically and let the mobile device automatically choose a feasible approach to perform its activities according to the situational changes of resources. A prototype has been implemented and evaluated based on several case studies.

List of Publications

Publications arising from this thesis include:

1. Chang, C., Srirama, S.N., Ling, S., (2013), Towards an Adaptive Mediation Framework for Mobile Social Network in Proximity, Pervasive and Mobile Computing Journal. Elsevier.
   DOI: `http://dx.doi.org/10.1016/j.pmcj.2013.02.004`

2. Chang, C., Srirama, S.N., Krishnaswamy, S., Ling, S., (2013), Proactive Web Service Discovery for Mobile Social Network in Proximity, JNIT: Journal of Next Generation Information Technology, Vol. 4, No. 2, pp. 100–112.

3. Chang, C., Srirama, S.N., Ling, S., (2012), An Adaptive Mediation Framework for Mobile P2P Social Content Sharing, in Proceedings of the 10th International Conference on Service Oriented Computing (ICSOC 2012). Ed: Liu, Chengfei and Ludwig, Heiko and Toumani, Farouk and Yu, Qi. Lecture Notes in Computer Science, Service-Oriented Computing, Springer Berlin Heidelberg, Vol. 7636, pp. 374-388.

4. Chang, C., Ling, S., Krishnaswamy, S., (2011), ProMWS: Proactive Mobile Web Service Provision using Context-Awareness, in Proceedings of the 2011 IEEE International Conference on Pervasive Computing and Communications Workshops, 21 March 2011, IEEE, Piscataway NJ USA, pp. 69-74.

5. Chang, C., Ling, S., (2010), Towards an Infrastructure-less SOA for Mobile Web Service Composition, in Proceedings of the 7th ACM International Conference on Pervasive Services, 13 July 2010 to 15 July 2010, Association for Computing Machinery, New York NY USA, pp. 180-185.

6. Chang, C., Ling, S., Krishnaswamy, S., (2010), Research Challenges in Mobile Web Services, in Enabling Context-Aware Web Services: Methods, Architectures, and Technologies, eds Quan Z. Sheng, Jian Yu and Schahram Dustdar, CRC Press, Boca Raton FL USA, pp. 495-519.

Service-Oriented Mobile Social Network in Proximity

Declaration

I declare that this thesis is my own work and has not been submitted in any form for another degree or diploma at any university or other institute of tertiary education. Information derived from the published and unpublished work of others has been acknowledged in the text and a list of references is given.

Chii Chang
November 11, 2013

Acknowledgments

Thanks to my supervisors: Dr. Sea Ling and Dr. Satish Narayana Srirama.

I've learned from Chris, not only about how to do research, but also the way of thinking and resolving problems in general. Being his student for over seven years have changed me a lot.

I must thank Satish who granted me the opportunity to work with him in Mobile Cloud Lab, University of Tartu, Estonia. Satish provided me with a lot of support and guidance. My research skills have been greatly improved. Without Satish, this thesis couldn't even start. Also thanks to the nice colleagues of Mobile Cloud Lab.

Additionally, thanks to University of Tartu, and the student tutor who helped me in the beginning of my life in Tartu.

I also like to thank Luke Albert Steller who provided me supports with the semantic service technology, and thank the colleagues of DSSE and staff at the Caulfield School of IT for their friendly and helpful support.

Finally, thanks to my family's support.

Chii Chang

Monash University

November 2013

Chapter 1

Introduction

1.1 Preamble

Accessing Social Network Services (SNS; e.g., Facebook, Twitter, Google+, LinkedIn and Foursquare, etc.) has become a common daily activity for Internet users. The marketing report (comScore; 2012) shows that on average, a Facebook user accesses Facebook's services for over seven hours per month, and with around 80% usage traffic derived from mobile applications. Recent smart mobile devices, such as smartphones,[1] tablets,[2] handheld media players,[3] smart cameras[4] and smart watches,[5] are capable of letting users produce various digital content, and share/upload the content to many SNS directly via wireless network connections. This has increased the number of people using mobile SNS applications.

Although the marketing report has shown that mobile SNS applications have successfully become the most popular applications for mobile users, mobile users have been restricted in the virtual communities of online SNS and are not aware of the social opportunities available to them. While mobile users spend most of their time accessing the online SNS, they have missed many opportunities to interact with others for new friendships, business opportunities, or information sharing (Borcea

[1] e.g., iPhone, Google Nexus, Windows Phone
[2] e.g., iPad, Amazon Kindle Fire, Samsung Galaxy Tab
[3] e.g., iPod Touch, Samsung Galaxy Player, Creative ZEN Touch 2
[4] e.g., Polaroid SC1630 Smart Camera, Samsung Galaxy Camera
[5] e.g., SONY smart watch

et al.; 2007). Consequently, applications such as those by MobiSoC (Borcea et al.; 2007), MobiClique (Pietiläinen et al.; 2009), MoSoSo (Tsai et al.; 2009), Uttering (Allen et al.; 2010) and Spiderweb (Sapuppo; 2010) have been proposed to enable a new breed of Mobile Social Network (MSN) functions, which can assist mobile users to interact with proximal people and perform various social activities such as searching for new friends who have common interests, exchanging content of common interests and establishing conversations. In this thesis, such a proximal-based MSN environment is termed a **M**obile **S**ocial **N**etwork in **P**roximity (MSNP).

An MSNP should not be seen as a replacement of existing SNS but as its complement (Pietiläinen et al.; 2009). MSNP leverages online SNS with a proximal mobile wireless network connection by providing location-based social networking opportunities. It can be applied in various social scenarios. For example, with the assistance of MSNP applications, an attendee in a highly populated conference can easily find someone who has common interests based on information derived from public profiles and their public information on online SNS. Another example: visitors who attend a big exhibition such as Comiket[6] in Japan or Comic-Con in USA[7] and Australia,[8] may be at a loss as to where they can find something they are interested in. With MSNP applications, they can rapidly discover the information about any point-of-interest shared by other MSNP application users based on their preferences in the same exhibition. MSNP also provides opportunities for active MSNP application users to bring more visitors to their online SNS spaces. For example, Twitter users can actively advertise content to MSNP application users who are potentially interested in the content, in order to bring more followers to their Twitter.

Although several software frameworks have been proposed to enable MSNP, most of these works are tightly coupled systems. In the past decade, service-oriented architecture (SOA) has become the mainstay of networked application development.

[6]See `http://www.comiket.co.jp/index_e.html`
[7]See `http://www.comic-con.org/`
[8]See `http://www.ozcomiccon.com/press.aspx`

Within SOA developments, the standardised Web service technologies[9] that provide platform independent common intercommunication interfaces have been broadly applied in various networked distributed computing areas to enhance the interoperability of different machines with heterogeneous software platforms. Web service has also been applied to numerous mobile applications either by utilising mobile applications as Web service clients (Kang et al.; 2007; Tian et al.; 2007; Yoshikawa et al.; 2003; Zahreddine and Mahmoud; 2005), or embedding HTTP Web servers to provide Mobile Web Service (MWS) directly from mobile devices (Gehlen and Pham; 2005; Srirama et al.; 2006; Koskela et al.; 2007; Pawar, Subercaze, Maret, van Beijnum and Konstantas; 2008). Utilising Web services can enable loose coupling and platform independent features for MSNP environments.

This thesis focuses on realising a loosely coupled, service-oriented MSNP environment based on Web service standard technologies. Service-oriented MSNP provides an open standard environment. With open standards, mobile application developers can easily implement compatible applications for mobile users to participate in MSNP without being bound to a particular device or software platform.

1.2 Motivation

In order to clarify the main objectives of this research, we use two scenarios to describe the motivations of this research project.

1.2.1 Scenario 1

> *Imagine an extrovert mobile user Jason is waiting for a train. He notices that another man Peter, is standing near him and seems to not be doing anything like reading or playing with his smartphone. So, Jason wants to start a conversation with Peter to pass time, since the train will not come for another 15 minutes. Jason starts the conversation: 'The train*

[9] See `http://www.w3.org/2002/ws/`

is delayed again.' Peter looks at Jason and gives a short response: 'Yeah ...'. It seems Peter does not want to talk to a stranger because the topic does not reflect his interests. Hence, Jason stops the conversation.

In the above scenario, if Jason is able to retrieve information about Peter, he may find a topic of common interest with him, so that their conversation can be continued. From the privacy point of view, ordinary people may not want to let strangers know about them. However nowadays, many people now have SNS accounts. It is very common that each SNS account has a corresponding public profile webpage about the user. Many people do not mind sharing some information about themselves to the public in their SNS profile pages. Such information can include what television programmes the user likes, what books the user reads and what games the user likes to play. Jason and Peter may both have public accessible SNS profiles that allow them to discover their common interests. However, the question is: 'How could Jason find Peter's SNS profile online?' Most existing SNS do not provide a feature to let a user discover the profile of another person who is located in the physical proximity in the real world. Before we discuss a possible solution, let us look at another scenario.

1.2.2 Scenario 2

Comiket, also known as the Comic Market, is the world's largest self-publishing convention held twice a year in Japan. Nakamura (2013) reported that the last Comiket (C83), which was a three-day event that took place at the end of December 2012, had roughly 550,000 visitors with roughly 35,000 artists who were selling their artworks, including comic books, animation DVD/Blu-ray, computer games, comic and animation related products or music CDs and underground DJ-remixed CDs. The visitors of the convention are not only local residents of Japan but include many foreign visitors from around the world. The report also mentioned that there were roughly 210,000 attendees on the last day of the market.

> *At such a high population event, without proper preparation or research, some foreign visitors may not be able to find what they want, and due to the crowded environment, soon, the visitors may quickly lose their passion and leave the venue with empty hands. This means that for lesser known artists, their works may have poor sales because the venue has too many sellers and competitors, and many visitors rarely pay attention to one particular exhibition spot.*

The above scenario indicates that there is a need for both visitors and exhibitors to interact, to share information at runtime, so that visitors can discover what they like, and that an exhibitors can advertise their works to visitors in the venue. Although the related information may be found on the Internet, not everyone knows where to discover such information.

In reality, some visitors may ask their surrounding visitors about where they can find what they are interested in. However, there are many challenges related to such a process. First, if they do not speak a common language, the visitor may have a difficult time finding someone to ask. Second, the visitor may have found many people to speak to, but most are unlikely to stop and respond because they are busy shopping. Third, even though someone may have responded, the answer may not satisfy the visitor's query.

Today, many people carry smartphones. With the rich capabilities of recent smartphones, people can share information using mobile applications without direct conversations in person. Moreover, with semantic computing technology, cross-language information sharing is also possible. In the Comiket scenario above, visitors could share information with one another using mobile applications. For example, a visitor who has made preparations and knows which exhibitions they plan to visit may share the information about the exhibitions (e.g., the artist's website URL) and their locations in the venue, and the topic/category of the artworks selling in those spots. Such an information provider could upload the information on their online space, such as their SNS page or blog. By sharing information

like URL links to surrounding users, they can potentially bring more visitors to an online space. As for the lesser known artist, they can also use mobile applications to advertise their information (e.g., their exhibition and URL link to their online portfolio such as a pixiv[10] page) to proximal visitors. The more unprepared visitor can use their mobile applications to retrieve and compile all the information shared by their proximal information providers and discover what they want. However, the question is: 'How can the information be advertised and be retrieved?' As mentioned before, the Comiket venue has over 200,000 visitors and 35,000 exhibition spots, and the environment is crowded. When a venue has a large number of information providers, the requester may discover too much unwanted information. Moreover, the artist may have a difficult time discovering who to advertise to without being scammed.

1.2.3 Discussion

The above two scenarios result in the same fundamental question: 'How to make discoveries in a proximal public mobile network using mobile applications?'

In order to allow mobile users to discover information about their proximal users in a public wireless network environment, which consists of a large number of participants, a classic network system would utilise a global remote central server to perform autonomous discovery and matching based on the information of a user's physical surrounding and preference profile, and a semantic matchmaking mechanism. Such a system requires users to install client-side applications in order to let the central server trace the users' current locations and perform the matchmaking process on-the-fly. However, because the centralised architecture has the potential single point of failure problem, and because the connection between the global server and the mobile client cannot be guaranteed, there is a need to support decentralised MSNP as a substitution when the global central server is not available, or

[10]See `http://www.pixiv.net/`

as the replacement of the centralised architecture to avoid the single point of failure problem.

Decentralised MSNP solutions can be categorised into two types:

- Pure Mobile Peer-to-Peer (MP2P)—A pure MP2P-based MSNP, which operates purely within mobile devices without remote resources, has been proposed by a number of researchers (Xing et al.; 2009; Allen et al.; 2010; Qureshi et al.; 2010) to enable location-based social networking without the Internet. A pure MP2P-based MSNP operates in a Mobile Ad Hoc Network (MANET) manner, which heavily relies on participants' collaboration for performing social network activities. Each participant has to maintain and share information about other participants in order to support the communication in the network.

- Hybrid MP2P—Hybrid MP2P-based MSNP solutions (Yang et al.; 2008), (Pietiläinen et al.; 2009; Sapuppo; 2010), which aim to leverage online SNS with an MP2P mechanism, can be seen as an extension of online SNS to support location-based communication. On the one hand, it utilises MP2P service discovery mechanisms to ensure participants to discover each other in proximal range. Conversely, content (e.g., a user's public profile, public posts, multimedia content) to be shared by participants is kept on their SNS spaces and the information about content is disseminated within the MP2P network.

This thesis proposes an MSNP solution based on the hybrid MP2P approach. As mentioned in the previous section, much research has been undertaken in regards to enabling proximal-based MSN. However, they resulted in tightly coupled solutions. Although, some works were proposed as open-source frameworks (Toninelli et al.; 2011; Yu et al.; 2011) and their tightly coupled solutions have benefited data transmission and communication maintenance, when the environment grows, different platform-based devices will participate in the network. Hence, developing a native MSNP application for each platform is time consuming and also inefficient. MSNP requires *platform neutral interoperability* for which standard Web service solutions are more appropriate. Further, when an MSNP application development is based on

a tightly coupled solution, the application will bind with a single non-standard communication mechanism, which reduces the communication opportunity in MSNP.

Imagine an MSNP environment with a verity of mobile device users. Each user intends to use their MSNP application to interact with one another. However, due to each MSNP application being implemented in different technology, the opportunity of discovery and interaction between mobile users is much less. For example, a user who is using an MSNP application based on JXTA (Juxtapose)[11] will be unable to communicate with a user who is using the Universal Plug and Play (UPnP)-based[12] MSNP application, because the way they perform discovery is different. Moreover, when the environment grows, the number of operation types and content types increases. In order to fulfil the need, semantic annotation may be applied to describe the operations provided by each participating device. However, due to the heterogeneity issue, the semantic discovery mechanism is difficult to be implemented. Conversely, if the entire system is Web service compliant, the overall interoperability can be highly improved.

1.2.4 Challenges

The benefit of applying Web service standards in MSNP is explicit. However, enabling decentralised Web service-oriented MSNP faces a number of challenges:

Service Discovery Latency—In MSNP, each user's mobile device is a Web service client and also a Web service provider (Srirama et al.; 2006). Since Web service has been applied as the common communication interface, and in most cases, MSNP participants do not have pre-knowledge about other peers, in order to support the service discovery process, each MSNP peer can use semantic Web standards, such as SAWSDL (Semantic Annotation for WSDL and XML Schema)[13] and OWL (Web Ontology Language),[14] to describe its services.

[11] See `http://java.sun.com/othertech/jxta/index.jsp`
[12] See `http://www.upnp.org/`
[13] See `http://www.w3.org/TR/sawsdl/`
[14] See `http://www.w3.org/TR/owl2-overview/`

While performing service discovery, an MSNP peer has to retrieve and process the other participants' service description metadata at runtime to enable dynamic Web service binding. Moreover, an MSNP peer is required to perform trust control processes on mobile devices in the MP2P network environment. Such a discovery process can cause high latency when the environment consists of large number of mobile Web service providers. Furthermore, the dynamic nature of MP2P requires the service discovery process to be fast in order to enable further interaction processes, because MSNP peers are extremely mobile. Each can move out from the current Wi-Fi network and join another, or can switch between 3G/4G mobile Internet.

Trust—Suppose a content requester discovers a Web service from a previous unknown content provider who can provide content of interest to the requester. Should the requester use the service to retrieve the content? In a basic process, the decision can be made by the human manually. However, if a more advanced autonomous service discovery operation is required, the task becomes critical. Imagine an MSNP participant intends to mashup a particular content from all content providers who provide the corresponding services. The process becomes inefficient if it requires the human user to manually select which providers' services they want to use. Hence, an autonomous decision making mechanism is more efficient, but trust is a major concern. Fundamentally, supporting trust in a Web service environment such as applying WS-Trust[15] requires a global entity to manage the trust-related data. However, because service discovery in MSNP is based on MP2P topology, it is impossible to establish a global central management party for supporting trustworthiness (Qureshi et al.; 2010). Hence, each MSNP participant has to manage the trust by itself. In a common SNS such as Facebook, a user can define different levels of content accessibility according to different social groups. A similar approach can be applied in a MSN solution (Kourtellis et al.; 2010). However,

[15]See `http://docs.oasis-open.org/ws-sx/ws-trust/v1.4/ws-trust.html`

for a new MSNP participant, who does not have many contacts, it is hard to define such access control.

Resource-Awareness—Cloud services such as the Platform as a Service (PaaS) based Google App Engine (GAE)[16] or the Infrastructure as a Service (IaaS) based Amazon EC2[17] provide flexible solutions for the need to facilitate high performance operations temporally. Considering the resource limitations of mobile devices and the dynamic nature of the MP2P environment, communication becomes a crucial challenge to both content provider and content consumer. In order to enhance the overall performance of MSNP communication, the service discovery process ideally should use different approaches depending on the environment status. For example, some tasks such as a semantic service/content matchmaking process may be offloaded to remote cloud services. However, distributing tasks to a cloud is not always an efficient solution, because utilising cloud service incurs extra costs such as network latency and the cost of using the service. In some cases, retaining the communication within the local wireless network is more efficient when both performance and cost are considered, especially when there are only a few MSNP peers involved. Conversely, when there are many MSNP peers involved, it may be more efficient to distribute more tasks to the cloud services. Hence, there is a need to design a framework that is capable of dynamically changing its resource management approach at runtime to adapt to different situations, while the MSNP peer is performing MP2P social network activities.

1.3 Research Objectives and Contributions

A typical MSNP phenomenon is the ability of sharing content produced by mobile devices. Content mashup, derived from a Web 2.0 technique, represents a content-driven service composition that enables MSNP participants to retrieve content of

[16]See `https://developers.google.com/appengine/`
[17]See `http://aws.amazon.com/ec2/`

a particular subject from multiple content providers and combine it into a single customisable presentation.

The dynamic nature of MP2P environments and the decentralised, ungoverned topology bring various challenges to developing a mashup-enabled service-oriented MSNP system. In particular, the service discovery and interaction between MSNP participants face latency, trust and resource management challenges as described in the previous section.

The research presented in this thesis focuses on developing a generic framework to enable loosely-coupled service oriented MSNP operation in a public MP2P environment. This research project intends to achieve the following objectives:

- To investigate, develop and validate approaches to overcome the latency-related and trust-related problems of the autonomous service discovery in service-oriented MSNP operated in dynamic public MP2P environment.
- To investigate, develop and validate an approach to resolve the resource management problems of service-oriented MSNP operating in a dynamic public MP2P environment.
- To design, develop and validate an adaptive mediation framework and its programming interface, in order to leverage and manage the autonomous service discovery mechanism for MSNP and its associated resources catering for dynamic changes in the MSNP environments.

To accomplish the objectives, we propose an **A**daptive **M**ediation framework for mobile **S**ocial **N**etwork in **P**roximity (AMSNP), which supports the following features:

- *Context-aware proactive service discovery*

  In order to overcome the service discovery latency issue in unstructured loosely-coupled service-oriented MSNP, the framework supports a proactive service discovery mechanism based on a context-aware user preference prediction scheme.

Further, considering the importance of trustworthiness in MSNP environments, the proposed framework also supports trust control based on social context and participants' reputation information.

- *Decentralisation*

  This avoids the single point of failure issue and enables MSNP to operate in dynamic public MP2P environments. The framework enables MSNP participants to discover and interact with one another without relying on any third-party servers' assistance, such as relying on a remote centralised discovery server or relying on super-peers, which are commonly seen in existing related research projects such as MobiSoC (Borcea et al.; 2007), MobilisGroups (Lubke et al.; 2011), MobiSoft (Kern et al.; 2006) and MoSoSo (Tsai et al.; 2009).

- *Resource-awareness*

  Enabling a decentralised loosely-coupled service-oriented MSNP environment faces resource-constraint issues because such an environment utilises extensively message-driven communication techniques. The proposed framework applied cloud service-based task offloading mechanism to reduce the resource usage of mobile devices. MSNP activities involve various communication tasks and resources, such as mobile device embedded services, SNS, and cloud services. A predefined static task (e.g., always offloading a particular process to a predefined cloud service) is not always the most efficient approach for that particular process because of the dynamic nature of the MSNP environment. In order to support dynamic reconfigurable tasks for performing MSNP activities, a corresponding strategy is proposed in this thesis.

The proposed framework is based on the Enterprise Service Bus (ESB) architecture. ESB is a software infrastructure that can easily connect resources by combining and assembling services to achieve a Service Oriented Architecture (SOA) (Robinson; 2004). The framework is controlled by a Business Process Execution Language (BPEL) based workflow system to dynamically manage and reconfigure tasks. The

runtime reconfiguration involves the autonomous decision making mechanism, which is based on utilising an adaptive workflow task execution scheme developed in this research project by composing fuzzy-set and cost-performance index models.

In summary, this research project aims to propose a mobile device-hosted loosely-coupled service-oriented adaptive mediation framework for unstructured public MSNP environments. The framework consists of the following mechanisms:

- *Proactive service discovery*, which:
  - utilises context information to enable autonomous service discovery
  - supports trustworthy service discovery based on social information.
- *Resource management*, to support dynamic reconfigurable tasks and to help MSNP participants adjust their approaches of performing MSNP activities dynamically at runtime to reflect situational changes.

1.4 Research Scope

This thesis focuses on developing a generic high-level mediation framework rather than proposing a low-level fundamental MP2P communication protocol.

The proposed framework is implemented and evaluated on real mobile devices—the Apple iPhone4S and iPod touch. The MSNP environment is based on simulation, in which a different number of mobile Web service provider peers are deployed on laptops and desktop computers within an IEEE 802.11n Wi-Fi network environment.

Providing MSNP in public wireless network environments can involve privacy and security issues. However, such issues are not in the scope of this thesis.

1.5 Thesis Outline

This thesis is structured as follows:

- Service-oriented mobile social network in proximity (MSNP) involves technologies of mobile Web service provisioning and mobile peer-to-peer (MP2P)

interactions. In order to understand the background of MSNP, **Chapter 2** describes a number of related technologies in the domains of Web service architectures, message exchange protocols, mobile Web service, Web service publish/discovery in MP2P environments, and MWS integration in the models of composition and mashup. Some content of this chapter has been previously published in Chang et al. (2010).

- Developing service-oriented MSNP needs to address a number of issues and challenges. **Chapter 3** identifies and describes the challenges, and reviews a number of related solutions from existing literature. In the follow up, we introduce our proposed MSNP architecture to meet the requirements of a service-oriented MSNP application.

- MSNP is based on a public MP2P environment, in which the dynamically joining/leaving participants may not have prior knowledge about each other. In order to perform social activities, participants have to perform a discovery process at runtime. It is inefficient if a user has to search for a particular content provider manually by browsing all the participants' MWS. Hence, the MSNP application should support an autonomous service discovery mechanism to assist users' searching activity. Autonomous service discovery in a service-oriented public MP2P environment faces the latency issue compounded by a large number of message exchanges, and the uncertainty of trustworthiness may make the user hesitant to access the MWS for the content. **Chapter 4** presents an approach for autonomous service discovery using environmental context information and social information to support trustworthy proactive MWS discovery in MSNP. Partial contents of the proposed scheme in this chapter have been previously published (see Chang et al.; 2011; Chang, Srirama, Krishnaswamy and Ling; 2013; Chang, Srirama and Ling; 2013).

- An MSNP participant is required to adapt to different situations because of the dynamic nature of MP2P environments. **Chapter 5** presents a mobile

device-hosted service-oriented workflow-based mediation framework for MSNP participant to adapt to dynamic changes. The fundamental portion of the framework is based on the Enterprise Service Bus architecture which supports changes to runtime resources without the need to re-launch the application. In order to adapt to different situations, the workflow task adjusts the execution behaviour at runtime. The workflow engine dynamically selects the best approach to complete the mobile user's request based on cost and performance, calculated by combining fuzzy set and cost performance index. The proposed framework in this chapter has been previously published in Chang et al. (2012) and Chang, Srirama and Ling (2013).

- In order to evaluate the proposed theory and design, **Chapter 6** presents a case study using an MSNP scenario. The prototype of the proposed framework has been developed and its functionalities have been evaluated. The evaluation result is described in detail together with a discussion of the findings. Partial contents of this chapter have also been previously published (see Chang et al.; 2011, 2012; Chang, Srirama, Krishnaswamy and Ling; 2013; Chang, Srirama and Ling; 2013).

- **Chapter 7** concludes this research project and provides suggestions for future research directions in this domain.

Chapter 2

A Review of Mobile Web Services

2.1 Introduction

Developing a service-oriented Mobile Social Network in Proximity (MSNP) for public wireless network environments requires loosely coupled standard protocols to support the platform neutral interoperability of MSNP participants. Generally, the interoperability of MSNP can be classified into two types: *physical interoperability* and *logical interoperability*.

Physical interoperability denotes the fundamental mechanism that enables MSNP participants' devices to discover each other and to exchange data with one another in wireless network environments. In general, Wi-Fi and Bluetooth are the two most common technologies to enable wireless network data transaction in proximal range. To enable the interoperability, a standard protocol, which helps MSNP participants discover each other automatically when they join the network, is required on top of the fundamental network environment.

Logical interoperability denotes the mechanism that helps MSNP participants to understand what functions are provided by one another. An MSNP participant who intends to search for a particular content shared by other MSNP participants may discover hundreds of MSNP participants in the current network. Suppose each MSNP participant device provides the human readable document describing its functions to let other participants understand what content/service is provided.

It is inefficient if a requester has to browse all the other participants' description documents to discover which participant is providing what the requester wants. There is a need to support a standardised, machine-readable functional description document to support autonomous discovery and filtering.

The MWS system (Srirama et al.; 2006) which represents a Web service standard based system, is a combination of the Mobile Peer-to-Peer (MP2P) network and Web services. On the one hand, an MWS system provides an open-standard-based technology to support physical interoperability of mobile network entities. On the other hand, it utilises a wide-range of Web service standard technologies to support various needs of logical interoperability including service description metadata, communication message format, quality of service control and privacy/policy, security.

This chapter presents an overview of the current MWS approaches and their related technologies to analyse their appropriateness in applying to MSNP. Partial content of this chapter has been previously published in Chang et al. (2010).

This chapter is organised as follows: Section 2.2 describes an overview of current Web service standard technologies. Section 2.3 reviews existing MWS solutions and their compatibility with MSNP. Section 2.4 provides an overview of mobile Web service integration approaches. Section 2.5 summarises the content of this chapter.

2.2 Web Service

Service Oriented Architecture (SOA) is a collection of design principles and methodologies to design and develop software entities as interoperable services. In the early age of SOA, Object Management Group's (OMG)[1] Common Object Request Broker Architecture (CORBA)[2] and Microsoft's Distributed Component Object Model (DCOM)[3] were the two common technologies to realise SOA. At the end of last century, Web service was introduced to provide a platform neutral, loosely coupled

[1]See `http://www.omg.org/`
[2]See `http://www.corba.org/`
[3]See `https://www.microsoft.com/com/default.mspx`

SOA, and has become the main technology to realise SOA in today's distributed computing.

According to World Wide Web Consortium's (W3C)(2004) definition:

> *'A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP (Simple Object Access Protocol) messages, typically conveyed using HTTP (Hypertext Transfer Protocol) with an XML serialisation with other Web-related standards.'*

A Web service is a platform independent interface to support interaction of different software entities. Figure 2.1 illustrates a basic Web service interaction model that consists of a service provider application operating on a Windows system and a service consumer application operated on a Mac OS (Operating System).

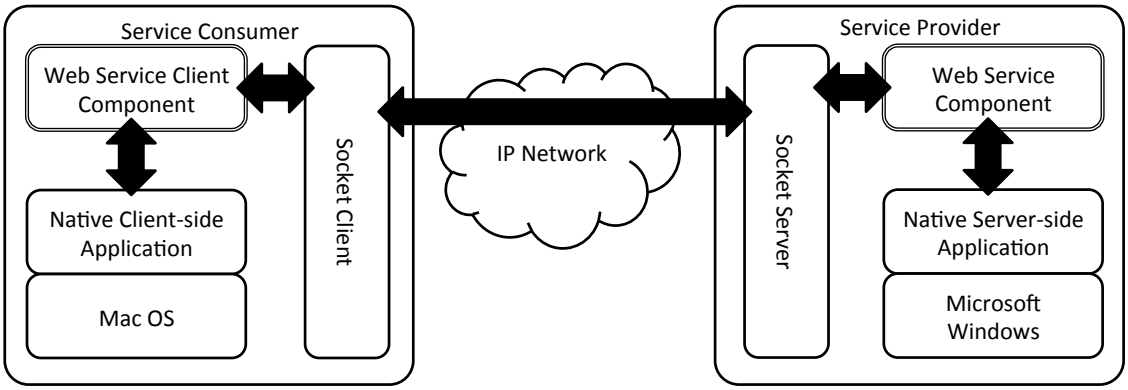


Figure 2.1: Basic Web service interaction model

The service provider application has applied the Web service component module to let the service consumer interact with the service provider by using standard protocols such as SOAP via an Internet Protocol (IP) network. A basic Web service component module:

- provides service description metadata, which describes what functionality the service provider provides.

- handles the incoming request messages sent from clients via the common protocol.

- handles the outgoing response message produced by the service-side component (e.g., parse the result data to XML-format).

A Web service provides various standardised formats to support needs in security (e.g., WS-Security),[4] policy (e.g., WS-Policy)[5] and semantic (e.g., OWL,[6] SAWSDL)[7] and so on that are lacking in CORBA and DCOM. These supports add high value to a Web service.

2.2.1 Web Service Infrastructure

In general, a Web service infrastructure is based on three basic entities: service provider, service requester and service registry (Papazoglou; 2008; Issarny et al.; 2011). As the technology evolved, and as Web services have been applied in various distributed computing systems, the service registry, which indicates that the system relies on a central repository to support service discovery mechanism, is no longer a necessary component in a Web service-based system.

Based on the Web service architecture document of W3C 2004, a Web service system has three basic entities: service provider, service requester and service discovery entity. The service discovery entity has been categorised into three basic forms: central registry, index, and P2P. Each is described below.

2.2.1.1 Central Registry

A classic Web service infrastructure has three entities: provider, requester and registry. Service providers publish their service description metadata (e.g., WSDL) to the remote central registry service, which is usually a service provided by a stable stationary server machine, and requesters search for particular services from the

[4] https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss
[5] See http://www.w3.org/2002/ws/policy/
[6] See http://www.w3.org/TR/owl-features/
[7] See http://www.w3.org/TR/sawsdl/

registry service (see Figure 2.2). A typical example of central registry is Universal

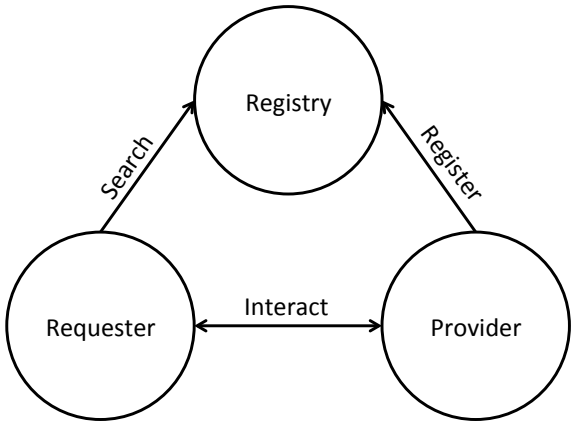


Figure 2.2: Central registry-based Web service infrastructure

Description, Discovery and Integration (UDDI) which was introduced as a core element of Web service infrastructure at the early stage to provide a central registry for global Web service providers to publish and list their services. However, UDDI did not gain much interest from the industries, and ended up losing support from major companies such as IBM and SAP.[8]

2.2.1.2 Index

An index-based Web service infrastructure is similar to central registry-based architecture in terms of service description metadata publishing. The major difference between index-based architecture and central registry-based architecture relates to who controls the published service metadata. In an index-based Web service architecture, there can be multiple index service providers providing the indexing mechanism, which actively searches services and also allows other service providers to publish/register their services.

Figure 2.3 illustrates an example of index-based Web service architecture. An index service can have its own approach to manage service descriptions and provide its own solution to help requesters search for services. Google search[9] is an example of an index-based service.

[8] http://soa.sys-con.com/node/164624
[9] http://www.google.com

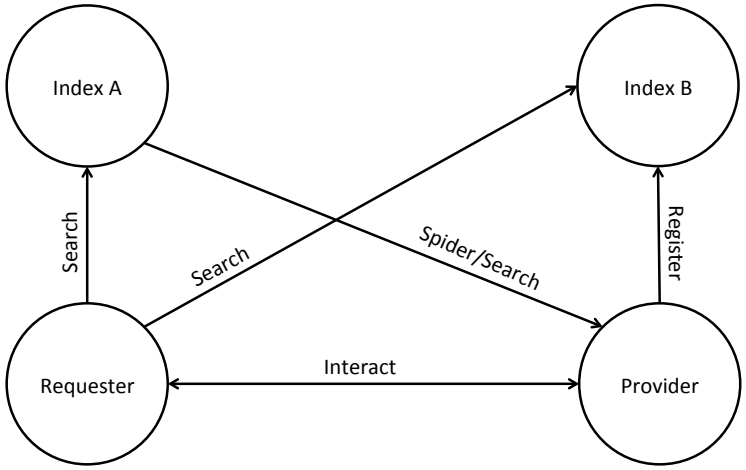


Figure 2.3: Index-based Web service infrastructure

2.2.1.3 Peer-to-Peer

P2P-based Web service infrastructure is a dynamic decentralised network in which a stable registry service or index service is not used. A typical example of P2P-based Web architecture is a group of software entities operating in a MANET environment. In such an environment, peers discover/interact with one another using a specific routing algorithm.

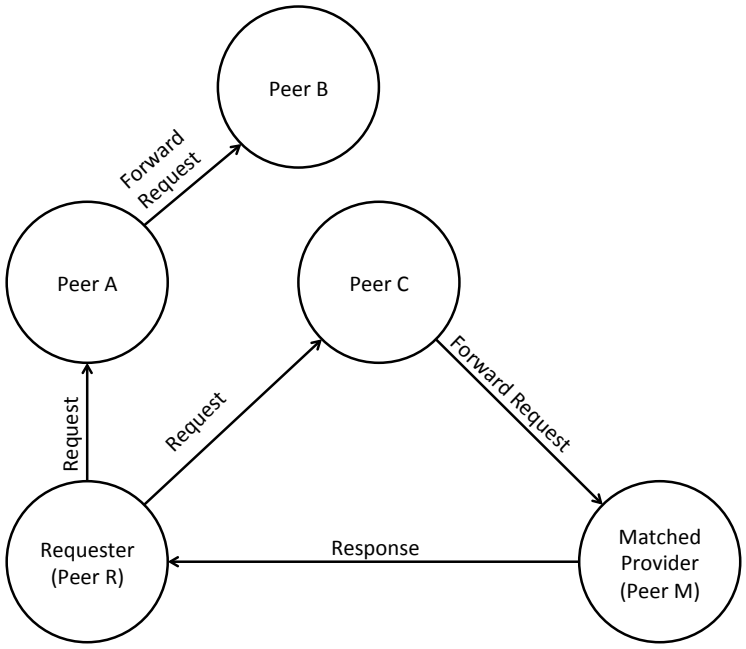


Figure 2.4: P2P-based Web service infrastructure

A generic approach, which is shown in Figure 2.4, utilises the request forwarding approach, in which a requester (Peer R) sends its request message to other connected

peers (Peer A and C in Figure 2.4), and if the recipient can meet the request, it will send a response message to the initial requester, otherwise the recipient (e.g., Peer C) will forward the request message to another peer (e.g., Peer M), which is not connected to the initial requester peer. In the example shown in Figure 2.4, Peer M can meet the request. Hence, it sends a response message to Peer R. Conversely, if the recipient (e.g., Peer B) cannot meet the request and has no further connected peer to forward the request to, the request message will be ignored by the recipient.

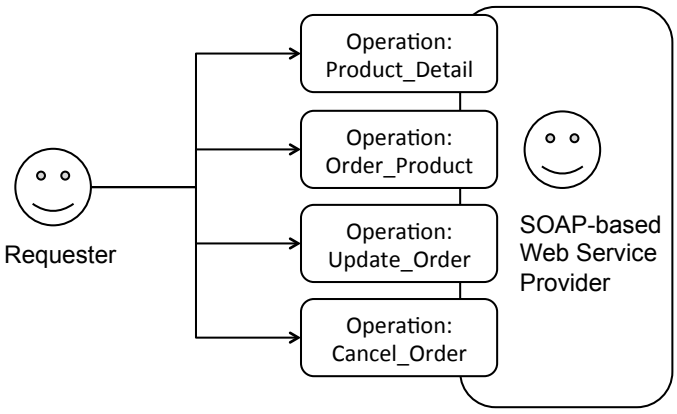
This example only shows a basic simple IP network environment in which each peer has its own global static IP address. In a more complex and dynamic environment in which peers do not have static IP addresses, the request/response message routing will require an advanced algorithm to support the communication.

2.2.2 Simple-Object Access Protocol and Representational State Transfer

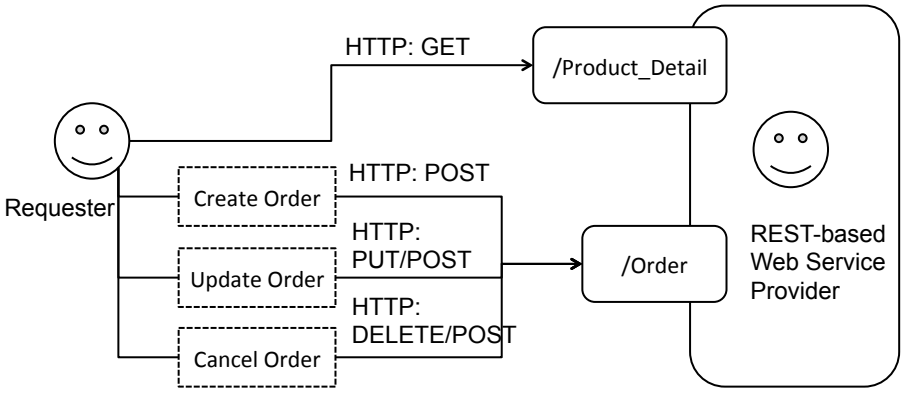
In 1998, Microsoft designed an object-access protocol called SOAP to replace its previous SOA solution, DCOM. Later, SOAP was submitted to W3C and has been included as a part of the Web service standard family. A SOAP message is an XML-formatted document used for both Web service request and response. A SOAP message is described as an envelope, which consists of a header (optional) and a body. For a SOAP request message, the body describes which operation the message sends to and where the response should go.

SOAP was designed as a message layer protocol, which does not rely on any underlying data transport protocol. A SOAP message can be transmitted using HTTP, Simple Mail Transfer Protocol (SMTP), File Transfer Protocol (FTP) or any other data transport protocols (Takase et al.; 2008). While SOAP has been widely adapted to implement Web service applications, an alternative option of Web service implementation—Representational State Transfer (REST)—has gained much interest within industries.

REST was introduced by Roy Fielding in his PhD thesis (Fielding; 2000), and is currently an alternative to SOAP-based Web service design models. The major difference between REST and SOAP is the request style. In SOAP, request messages have been formatted in an XML document. In REST, a request is simply sent to a URL with an HTTP request method: `GET`, `POST`, `PUT` or `DELETE`. Figure 2.5(a)



(a) SOAP-based Web service invocation



(b) REST-based Web service invocation

Figure 2.5: Web service invocation comparison

and (b) illustrate the difference between REST and SOAP using a product order service example. In SOAP (Figure 2.5(a)), each function is an individual operation. In REST (Figure 2.5(b)), three functions, `create order`, `update order` and `delete order`, are sent to the same URL with different HTTP methods and parameters. For example, to get product detail from a SOAP-based Web service (see Listing 2.1), the request message consists of two parts: a HTTP request method and the SOAP message envelope.

Listing 2.1: SOAP-based request example

```
GET / HTTP/1.1
Host: www.example.com
Content-Type: application/soap+xml; charset=UTF-8
Content-Length: {length}

<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope" >
  <env:Header>
    <!-- Header information here -->
  </env:Header>
  <env:Body>
    <m:getProductDetail xmlns:m="http://www.example.com/">
      <productID>21</productID>
    </m:getProductDetail>
  </env:Body>
</env:Envelope>
```

Conversely, a RESTful Web service (REST-based Web service model) request will be a simple HTTP `GET` request with the corresponding URI, which is shown in Listing 2.2.

Listing 2.2: REST-based request example

```
GET /product/21
Host: www.example.com
Content-Type: application/x-www-form-urlencoded
```

A RESTful Web service requires much less data transaction between requester and provider (Pautasso et al.; 2008). However, if additional content needs to be attached with the request message such as standardised semantic annotations, which involves substantial URI namespace management, a SOAP-based Web service is more flexible than a REST-based Web service (Zur Muehlen et al.; 2005). For example, in a P2P-based MSNP environment, a requester intends to retrieve an 'event picture of current location' from other participants by multicasting the request message to other peers. If the requester uses a SOAP request message, the request message can embed semantic annotations to help other participants to understand the request method. Conversely, such flexibility is not supported by REST because HTTP method-based REST does not use standard document-based messages

for communication, and it may result in a tightly-coupled system if a customised approach is applied.

2.2.3 Web Service Description and Semantics

One of the core Web service elements is Web Service Description Language (WSDL). A WSDL document is an XML-based metadata for describing the functions/operations and the corresponding input/output data types of a networked service. A WSDL document is allowed to embed additional information (e.g., schema types) to support the description of data types without extra documentation.

The previous version WSDL 1.1[10] was originally designed for the generic purpose of describing services (not only HTTP-based services). It only supports `GET` and `POST` of the HTTP methods. Its lack of compatibility to REST motivates researchers to develop alternative solutions such as Web Application Description Language (WADL), which was introduced specifically for describing RESTful Web services. WADL was submitted to W3C as the candidate standard, but later, W3C released WSDL 2.0[11], which fully supports the need to describe RESTful Web services.[12] The XML-formatted WSDL document provides the syntax to describe functions of a Web service, but it cannot express the meaning of the functions (Akkiraju and Sapkota; 2007; Berners-Lee et al.; 2001).

Web service semantics, which represents the meaning and purpose of Web service interaction (Booth et al.; 2004), is an important element of the Web service lifecycle (Akkiraju et al.; 2005), especially when the implementation supports MP2P and ubiquitous environments, the core environment of MSNP.

[10]See `http://www.w3.org/TR/wsdl`
[11]See `http://www.w3.org/TR/wsdl20/`
[12]See `http://www.w3.org/TR/wsdl20-adjuncts/#http-binding`

2.2.3.1 Semantic Web Services

Service discovery in MP2P and ubiquitous environments usually rely on the standardisation of service description and messages, in which prior knowledge of interacting entities is required (Heflin; 2004). However, an important goal of ubiquitous computing is to enable participants (e.g., devices) to interact automatically at runtime without prior knowledge. Such a goal requires a mechanism that helps participants to understand each other's services/functionality.

Autonomous service interaction in a dynamic environment requires the service to be in a machine-interpretable form (Akkiraju et al.; 2005), in which a contract to describe syntax and terms needs to be agreed between service providers and requesters, because it is common to see different parties use different terms to describe their Web service operations/functions (Berners-Lee et al.; 2001), or use the same terms to describe different Web service operations/functions. Although applying a standard to describe services can easily solve the problem, due to the dynamic nature of MP2P and ubiquitous environments, applying a prior standardisation for terms in the environments is an unmanageable task (Heflin; 2004).

Instead of utilising standard terms to describe services, semantic Web systems utilise external knowledge bases (e.g., ontology) to describe the meaning of the terms used in service description documents. Hence, the autonomous service discovery can be done without relying on standard terms used in service description documents (Akkiraju and Sapkota; 2007; McIlraith et al.; 2001).

2.2.3.2 Web Ontology Language

'An ontology defines the terms used to describe and represent an area of knowledge' (Heflin; 2004). In semantic Web services, ontology is used to define the meaning of terms used in service description documents. Different to XML, an ontology does not provide a description of syntax or data structures (Motik et al.; 2009).

Web Ontology Language (OWL) (McGuinness et al.; 2004) is a language designed specifically to describe the semantics of World Wide Web resources including

Web pages and services.[13] An OWL document is usually written in the Resource Description Framework (RDF)[14] format, and usually consists of three basic components (Heflin; 2004):

- *Classes*, which represent general things/individuals in various domains. For example, a 'shopping centre' is a class.
- *Attributes*, also known as properties, describe what elements are included in classes. For example, a class 'shopping centre' may have attributes: 'food court', 'clothing department' and 'electronic department'.
- *Relations*, which describe the relationships between classes. For example, an 'electronic department' is in a 'shopping centre'.

Although the standard-based ontology description document OWL is written in XML-based RDF files, a recent effort from W3C's RDF working group[15] introduced a W3C Working Draft, the JavaScript Object Notation (JSON)[16]-based approach to enable ontology: JSON-DL.[17] JSON is an alternative format for data exchange in Web applications. Many RESTful Web service applications such as Google App[18] and Yahoo Web service[19] support JSON to reduce data size. The drawback of JSON is its lack of standardisation in supporting semantics. Since the emergence of the JSON-DL standard (although the current version is still in its draft stage), it is anticipated that in the near future ontology can be implemented in RESTful Web services effectively.

[13]See http://www.w3.org/2003/08/owlfaq
[14]See http://www.w3.org/RDF/
[15]See http://www.w3.org/2011/01/rdf-wg-charter.html
[16]See http://www.ietf.org/rfc/rfc4627.txt
[17]See http://www.w3.org/TR/2012/WD-json-ld-syntax-20120712/
[18]See https://developers.google.com/google-apps/
[19]See http://developer.yahoo.com/javascript/json.html

2.2.3.3 Semantic Annotations for WSDL and XML Schema

By default, a WSDL document does not support semantics to leverage ontology and Web service descriptions. Hence, in the past years, many works such as WSML (Bruijn et al.; 2006), METEOR-S (Patil et al.; 2004), OWL-S (Martin et al.; 2005), SWSL (Battle et al.; 2005) and WSDL-S (Akkiraju et al.; 2005), have been introduced to provide semantic Web service description. Later, W3C also introduced SAWSDL (Akkiraju and Sapkota; 2007), which was based on WSDL-S, as the standard/recommendation to support semantics in WSDL documents.

SAWSDL supports semantic annotation in both operation description level and data description level. It has three major components, which can be embedded as XML attributes (Farrell and Lausen; 2007):

- *modelReference*, which maps the XML element in WSDL (e.g., operation, interface and input/output) to the semantic model described in external ontology.

- *liftingSchemaMapping*, which maps the XML schema data type to its semantic model described in external ontology.

- *loweringSchemaMapping*, which maps the semantic model to the corresponding XML schema data type.

Additionally, SAWSDL provides *attrExtensions* specifically for WSDL 1.1 to support semantic annotation in attributes extensions.

A semantic Web service description plays an important role in a dynamic MP2P environment in which peers need to discover each other at runtime. By applying semantics, a service requester can describe its search requirements with terms from the semantic models. When a service provider receives a request message with semantic annotations, it can perform reasoning to identify whether its service semantically matches the request or not, instead of using keyword-based syntactical matching (Akkiraju and Sapkota; 2007).

2.3 Mobile Web Service Provisioning

The term MWS has been described differently by various researchers. Some researchers describe MWS as Web services designed for mobile device-based client-side applications (Yoshikawa et al.; 2003; Zahreddine and Mahmoud; 2005; Kang et al.; 2007; Tian et al.; 2007). Although such Web services provide mobility support for client-side mobile applications, the Web service providers still remain static. In this thesis, we prefer the MWS definition of recent works (Gehlen and Pham; 2005; Srirama et al.; 2006; Pawar et al.; 2007; Dorn and Dustdar; 2007; Kim and Lee; 2009; Zhang et al.; 2010; AlShahwan and Moessner; 2010; Elgazzar et al.; 2011) which describe MWSs as mobile-device-hosted Web services, in which the Web services are provided by mobile devices such as smartphones, PDAs and handheld media players.

MWS has been implemented in various scenarios (Srirama; 2008) mainly to support loosely coupled interoperability in pervasive/ubiquitous systems that consist of heterogeneous participating devices including static computers and portable mobile devices used by mobile users. Such a context information (e.g., user's current location, activity, environment) sharing distributed system involves numerous heterogeneous devices (Dorn and Dustdar; 2007). MWS provides the flexibility for developers to implement the system without worrying about different platform-based mobile devices used by participants in the system, as MWS provides a common interface to support communication between participants' devices.

Providing Web services from mobile devices is not a straightforward task because one has to deal with resource constraint issues common in mobile devices, and connectivity issues resulting from device users' mobility. This section provides a review of works in mobile Web service provisioning. However, before we review the works, we first describe a number of specific terms used.

Mobile Host—A mobile host, which represents an application operating on a physical mobile device (e.g., a smartphone), is capable of providing networked service to receive/respond request messages from remote applications via a

common protocol of a wireless network. A mobile host can provide one or more mobile services.

Mobile Service—Mobile service is a generic term to describe a computational operation or activity that functions on a mobile host to receive and respond to requests from remote applications.

Mobile Web Service—A MWS represents a mobile service that follows the Web service standards for its operations. A MWS provides WSDL as its service description metadata, and is communicable via a protocol of Web service standards such as utilising SOAP or utilising HTTP methods as a RESTful Web service.

2.3.1 Mobile Web Service Implementation

Fundamentally, a mobile host that intends to play the role of Web service provider must support two basic Web service components (i.e. WSDL, and SOAP or REST) described in the Web service architecture (Booth et al.; 2004).

Commonly, WSDL and SOAP/REST are supported by components built on top of fundamental network socket servers. Mobile OS such as Symbian, iOS (formerly iPhone OS) and Android OS, hosting Web servers on mobile devices to provide networked services in an IP network are no longer challenging. Many open-source tools are available for developers to implement Web servers on mobile devices. For example, Nokia Mobile Web server[20] enables a Symbian OS-based device to host Web servers; CocoaHTTPServer[21] and Mongoose Web Server[22] enables a HTTP Web server deployed on iOS devices; and kWS,[23] iJetty[24] and Android Web Server[25] enable Android OS-based mobile devices to provide HTTP Web services.

[20]See http://research.nokia.com/page/231
[21]See https://github.com/robbiehanson/CocoaHTTPServer
[22]See http://code.google.com/p/mongoose/
[23]See https://play.google.com/store/apps/details?id=org.xeustechnologies.android.kws
[24]See http://code.google.com/p/i-jetty/
[25]See http://code.google.com/p/android-webserver/

By implementing additional XML processing components, a basic mobile Web server can provide Web service mechanism and achieve the fundamental requirement of Web service provisioning. The main challenges of MWS implementation derive from the resource constraint issues in terms of Central Processing Unit (CPU) power, memory and battery power. Normally, the CPU in a mobile device consumes less battery power, so its processing performance is much less than a generic desktop/laptop computer's CPU. Even though recent high-end mobile devices have quad-core CPU and 1G/2G RAM, the processing performance still cannot compete with a desktop/laptop computer CPU. Hence, if a mobile host has to participate in a semantic Web environment, in which a large number of semantic annotated XML documents need to be processed by the mobile host, the overall performance can be poor. Battery-life is another crucial challenge, because while the CPU technology may evolve, battery-life is difficult to improve. A mobile host that serves a large number of clients can quickly run out of its battery power.

2.3.2 Mobile Web Service Architectures

Depending on the scenarios, MWS systems can be implemented based on various architectures. In general, MWS systems can be categorised into two basic architectures: central registry-based and P2P (decentralised).

2.3.2.1 Central Registry-based Architecture

Central Registry-based Architecture (CRA) relies on a single or a group of static servers to provide registry service for participants in the MWS system to discover each other. The fundamental architecture is similar to the centralised Web service architecture described in the previous section. However, since MWS providers are mobile nodes, which commonly do not have static IP addresses, MWS providers may need to frequently update their IP address information, and the registry server will need to frequently track the status of MWS providers to identify their connectivity states.

CRA is suitable for global range-based MWS system such as remote patient monitoring systems (Ong and Center; 2006; Gehlen; 2007) or mobile learning systems (Chatti et al.; 2006) in which mobile hosts collaborate with static servers in a distributed system.

The challenge in CRA is how the registry servers can handle message overheads. A single point of failure will cause the entire registry service fail. There are solutions such as context-aware registry (Han et al.; 2005) to distribute registration into several categories handled by different servers. However, if the "portal" crashes, the entire discovery process cannot be performed.

2.3.2.2 Peer-to-Peer Architecture

Peer-to-Peer (P2P)-based MWS can be further categorised into two models: flooding and document routing (Gehlen and Pham; 2005). The flooding model is the most basic approach to enable mobile P2P network interaction. Request messages are sent directly from the requesting peer to its directly connected peers using broadcast technique. The document routing model is based on a sharing and synchronising Distributed Hash Table (DHT) (Rescorla and Resonance; 2006) within the network.

P2P architecture can avoid the single point of failure issue. However, due to the fact that MWS systems heavily rely on XML-formatted messaging (WSDL and SOAP), MP2P-based MWS systems face more data transport overhead challenges than the central registry-based solution (Zhu et al.; 2010). Therefore, an MP2P-based approach requires developers to discover the best protocol for their scenarios. Although there are various industrial specification and standards for MP2P service discovery (described in the next section), for best practice, application developers usually need to customise to best suit their applications. For example, in order to realise standard Web service interaction in a JXTA network, Srirama et al. (2007) have proposed a mediation framework to leverage JXTA and SOAP Web services.

The network topology of MP2P can be direct or indirect P2P. Direct P2P means peers can directly connect without a third party facility such as routers. Direct

P2P usually refers to MANET established by Bluetooth, Wi-Fi ad hoc mode or Wi-Fi direct (Gehlen and Pham; 2005). Indirect P2P uses an additional facility to set up the environment. Such a facility can be a static (static network router) or nomadic access point (mobile device with Wi-Fi roaming function). Either way, the fundamental topology only provides the physical connection establishment. In order to interact, peers need to know what functions are provided by one another. This leads to the subject of service discovery frameworks.

2.3.3 Mobile Web Service Discovery and Interaction

Because mobile hosts are nomadic nodes, the connectivity of MWS nodes is unstable. Hence, MWS interaction requires a dynamic service discovery mechanism. This section provides a comparison of existing popular mobile service discovery technologies.

2.3.3.1 Apache River

Apache River[26] (or 'River' for short), also known as Jini, is a centralised MP2P service provisioning technology. Services in River are instances of Java objects, which are described by using Java Object Class Definition (Java Interface) (Obiltschnig; 2006), and the attributes of the Java objects are also described as individual Java objects.

The service discovery architecture in River relies on the central repository approach. The central repository, which is called *lookup service*, manages a registration table letting other nodes publish/discover services. A River environment can consist of multiple *lookup services* (Obiltschnig; 2006). Service discovery in River consists of two phases. First is the bootstrap phase in which River nodes perform UDP multicast to find a *lookup service* in the network. A River node can also use unicast to find a *lookup service* if the node knows where the *lookup service* is located. After that, in the second phase, the publish/search phase, River service provider nodes

[26]See `http://river.apache.org/`

can publish their services on the *lookup service*. As for service clients, they can search the published service on the *lookup service*. When a service client finds a service provider it needs from the *lookup service*, it can directly communicate with the service provider using special Java objects (known as *proxy objects*).

River has two disadvantages (Meshkova et al.; 2008):

- Reliance on a centralised repository, which faces a potential single point of failure problem.
- Dependence on the Java Virtual Machine (JVM). River is a platform specific technology that is only available in the JVM environment.

Originally, River was not a WS-* compliant technology. It requires additional components to integrate Java objects to WS-* compatible data. A related work has been introduced by van Halteren and Pawar (2006). Because the fundamental technology of River is tied to the JVM, building a Web service system on top of River introduces more complexity compared to XML message-based technologies such as JXTA and UPnP, which are described in the following paragraphs.

2.3.3.2 JXTA

JXTA, which was introduced by Sun Microsystem, is a collection of MP2P communication protocols based on super-peer P2P network topology. It has been widely applied in MWS provisioning research works (Srirama et al.; 2006; Schmidt et al.; 2007; Doulkeridis and Vazirgiannis; 2008; Amoretti et al.; 2008) because of its platform and language independent nature.

JXTA is a complex but powerful decentralised MP2P communication technology. It is applicable to various MP2P applications from the range of local area to global Internet. Each peer in a JXTA network has a 128 bit unique ID, and each peer can publish its service description (advertisement) as a XML-formatted file to its network.

There are two main types of peer in JXTA:

- Edge peer, which represents a generic participant of a JXTA network.
- Super peer, which assists JXTA peers to discover each other. Super peer is further categorised into two subtypes:
  - Relay peer, which acts as a broker to help the cross firewall communication.
  - Rendezvous peer, which acts as a repository for JXTA peers in the same network. A rendezvous peer stores advertisements (service description in distributed hash table form) published by service provider peers in its network, and it can synchronise advertisements with other rendezvous peers.

JXTA peers interact with one another using the routing technologies called "pipes". Pipes are asynchronous, unreliable and unidirectional communication protocols. There are three types of pipes supported by JXTA, including two point-to-point pipes—unicast and secure unicast—and a propagate pipe, which lets a message to be sent from one to multiple nodes. JXTA peers can also use pipes to perform service eventing by subscribing to particular service provider peers.

Since JXTA is an XML-based technology, integrating SOAP-based Web service in a JXTA network is less complex than Apache River. Previous research effort contributed by Srirama et al. (2008) has demonstrated a complete solution for utilising Web service in a JXTA environment.

JXTA has two limitations derived from its fundamental design. First, JXTA is not a pure P2P model (Junginger and Lee; 2005). A JXTA network cannot be established without rendezvous peers. Edge peers cannot communicate directly without assistance from rendezvous peers. Second, JXTA requires predefined topology to set up the environment (Kurmanowytsch et al.; 2003), which is less flexible than the other three technologies described in the following sections.

Although JXTA has been a popular technology in the research domain, it did not gain much interest in industries and in commercial fields when it is compared to

UPnP and Bonjour (described in the following paragraphs). The official support by Oracle (after its purchase of Sun Microsystem) has been terminated since the end of 2010. At the time of writing this thesis, the status of JXTA is still uncertain. The decision to let the project move to an open-source community is pending (JVerstry; 2010).

2.3.3.3 Bonjour

Bonjour, which is Apple's[27] technology to achieve Zero-configuration networking (Zeroconf),[28] is a collection of networked service discovery protocols for decentralised mobile P2P environments. In 2011, Apple has submitted Bonjour to Internet Engineering Task Force (IETF) as an ongoing standard protocol (Mac Developer Library; 2013).

Bonjour consists of three main elements:

- Addressing—When a Bonjour-enabled peer joins a local network, it utilises a self-assigned link-local addressing mechanism to automatically assign an IP address to itself and also test the IP address's validity. If the IP address has been used by another peer in the network, it will pick another IP address and test it until a valid IP address is available.

- Naming—Bonjour peers can self-assign service names for their services, and use Multicast Domain Name System (mDNS) queries to identify whether the name has been used or not. If a same name service has been published in the network, the later published service will self-assign a number after its original name to avoid redundant service naming in the network. Beside the original mDNS mechanism, Bonjour also provides an additional element called Bonjour mDNSResponder, which lets a service provider listen to queries on the network. Queries that have been specified for a service name will be automatically directed to the service provider peer.

[27] `http://www.apple.com`
[28] See `http://www.zeroconf.org/`

- Discovery—The service discovery of Bonjour is based on mDNS querying, which is called 'browsing'. A peer, searching for services, utilises multicasting to browse published services by specifying the service type and domain, which are defined in the browsing method.

Bonjour is usually applied in a local network (either LAN [Local Area Network] or WLAN [Wireless Local Area Network]) for networked applications to discover services published by networked devices in the same subnet dynamically without a central repository. Devices in a Bonjour network publish their services based on specific types following the format of <service name >.<application protocol >.<transport protocol >.<local >.
(e.g., myfilesharing._ftp._tcp.local.).[29] The service name is in a human readable text format, which is chosen by the service publisher. If a redundant name appears in the network, the later published service name will be automatically assigned with a number following its original name. The application layer name represents the Domain Name System (DNS) service type, which can follow the registered types found in the official DNS service type list[30] or it can be any name chosen by the application developer if the network is only applied in a specific system. The transport layer name can be Transmission Control Protocol (TCP) or User Datagram Protocol (UDP). The last name—'local.'— represents the service published in the local area network, which is also known as Bonjour link-local. It is possible to establish a Bonjour network in the Internet, which is known as wide-area Bonjour. However, it requires a central DNS service,[31] which means the architecture is no longer decentralised.

Initially, Bonjour was designed as a transport protocol. It does not provide service description mechanisms, neither does it provide a standard messaging protocol

[29] See https://developer.apple.com/library/mac/#documentation/Cocoa/Conceptual/NetServices/Articles/domainnames.html#//apple_ref/doc/uid/TP40002460-SW1

[30] http://www.dns-sd.org/ServiceTypes.html

[31] http://dyn.com/support/bonjour-and-dns-discovery/

for service invocation processes. However, it is implied that Bonjour is flexible to be applied to various network applications including Web services.

Bonjour is open-specification. A tool (Application Programming Interface [API]) for implementing Bonjour is available in almost all the major computer platforms including desktop OS such as Windows, Linux, Mac OS, Unix, BSD, and mobile OS such as iOS and Android OS.

2.3.3.4 Universal Plug and Play

The basic concept of Universal Plug and Play (UPnP) is based on two types of entities: control point, which represents service client, and controlled device, which represents service provider. A peer in UPnP can act as a control point, or a controlled device, or both. A controlled device can provide one or more services. This basic concept is similar to JXTA or Bonjour.

UPnP can be seen as a different version of zero-configuration networking technology with additional features to achieve a more complete mobile P2P service provisioning solution. UPnP covers similar service discovery mechanisms provided by Bonjour. In the service discovery stage, UPnP also supports self-assigned addressing, naming and multicast discovery, which is realised by a slightly different technology called Simple Service Discovery Protocol (SSDP). Additionally, UPnP provides its standard protocols for service interaction stage similar to what JXTA has provided.

UPnP consists of the following additional elements for service interaction:

- Description—Each controlled device provides a URL to let control points retrieve an XML-formatted description metadata, which describes what action(s) is provided by the device. The metadata can also describe the manufacturer's information (e.g., vendor name, product/model ID) of the device.

- Control—After a control point processes a controlled device's description metadata, the control point can access/invoke the operation of the controlled device

by sending SOAP messages. Basically, the interactions between UPnP peers use SOAP messages.

- Eventing—Similar to JXTA, UPnP also supports eventing. UPnP supports the Generic Event Notification Architecture (GENA) technology, which allows control points to subscribe to controlled devices to listen to the notifications/messages of the subscribed controlled devices.

- Presentation—UPnP commonly operates in HTTP protocol. A controlled device can then provide HTML pages letting the users of control points manually access the controlled device via Web page-based interfaces.

Since the service interaction of UPnP is based on XML-formatted messages, it is possible to apply Web service in UPnP by mapping the service description metadata from UPnP's own format to WSDL. However, because the fundamental concept of UPnP is slightly different to generic Web services, the mapping process can face technical challenges. For instance, a UPnP device description metadata describes the actions of the device, but a WSDL describes services in terms of input types and output types, which are not applicable in UPnP.

2.3.3.5 Device Profile for Web Services

Device Profile for Web Services (DPWS)[32] is a collection of Web service standard protocols introduced with the same intention as UPnP. The main difference between DPWS and UPnP is that DPWS was introduced to fully support Web service standards and has been approved by OASIS[33] as a part of the Web service standard family in 2009.

DPWS utilises WS-Addressing and WS-Discovery to achieve the mechanism of mobile P2P service addressing, naming and discovery in a local network. WS-Discovery lets service clients in a DPWS environment find services by using a multicast SOAP-over-UDP technique. The SOAP message for service discovery in DPWS

[32]See `http://docs.oasis-open.org/ws-dd/ns/dpws/2009/01`
[33]See `https://www.oasis-open.org/`

is called Probe message. Probe message defines what type of service the client is looking for, and where the response message should be sent, by following the WS-Addressing standard. Service providers in DPWS actively listen to Probe messages sent by the multicast protocol based on WS-Discovery. If a service provider receives a Probe message that matches its service, it will respond with another Probe message to the initial sender.

Service providers in DPWS describe their services by using WSDL. When a service client receives a Probe message from the matched service provider, it will send a Get Metadata message to the service provider to retrieve the service provider's WSDL for further service invocation processes. The service invocation process in DPWS is the same as UPnP, which uses SOAP messages.

DPWS also supports eventing by following WS-Eventing, which allows service clients to subscribe and receive *Subscribe* messages (defined in WS-Eventing) from the service provider.

Although DPWS is a WS-* standard based technology, it only supports SOAP-based communication. The recently popular RESTful Web service architecture is incompatible with DPWS.

Currently, DPWS still lacks implementation APIs. Although a few research organisations such as SOA4D[34] and WS4D[35] have introduced versions of APIs written in C/C++ and Java, and WS4D has also provided its API for the Android OS platform, there is no API for the iOS platform, which has a large number of users nowadays.

Table 2.1 summarises and compares the technologies described in this section.

[34]See `https://forge.soa4d.org/`
[35]See `http://ws4d.e-technik.uni-rostock.de/`

	Apache River	**JXTA**	**Bonjour**	**UPnP**	**DPWS**
Topology	Centralised	Semi-centralised	Decentralised	Decentralised	Decentralised
Scope	Local	Internet	Local	Local	Local
Addressing	—	—	Self-assigned addressing	DHCP; Self-assigned addressing	WS-Addressing
Naming	—	128bit unique ID	mDNS	DNS; IP	WS-Addressing
Discovery	lookup service	Super-Peer Advertisement	DNS-SD	SSDP	WS-Discovery; WS-Metadata Exchange
Description (WS Required)	Java Object Class Definition	XML	—	XML	WSDL
Invocation (WS Required)	Java RMI	Pipes	—	SOAP	SOAP; WS-Transfer
Eventing	—	Pipes	—	GENA	WS-Eventing
Presentation	Java Class	—	—	HTML	—
Platform	JVM	Cross Platform	Cross Platform	Cross Platform	Cross Platform

Table 2.1: Comparison of mobile P2P service provisioning technologies

2.4 Context-Aware Mobile Web Services

2.4.1 Context-Aware Computing

Context awareness was introduced by Schilit et al. (1994). Context was described as the environmental phenomenon influencing the system process and enhancing the adaptive autonomous actions of the system. Later, Dey (2001) defined context as:

> *'Any information that can be used to characterise the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves.'*

Chen et al. (2000) identified a common definition of context in their literature study:

> *'Context is the set of environmental states and settings that either determines an application's behaviour or in which an application event occurs and is interesting to the user.'*

Context aware computing has been a research field of interest in the Web services area for many years. However, more issues need to be considered in the mobile environment than in a static environment. In general, context can be categorised into two types: static context and dynamic context (Moore et al.; 2008). Static context is a user-customisable context that considers a user profile containing the user's personal information and preferences. Dynamic context can be considered as environmental factors in which the user is passive and is unable or has less control. The system relies on external entities to be observed and sensed, and to derive meaningful information.

Context can be further classified into current context and past context (Yang; 2006). Current context represents the runtime environmental factors of a system. When current context data is acquired, the current context becomes past context. Past context, which is also known as *context history* (Chen et al.; 2000), can then be used in the behavioural or preference prediction processes.

2.4.2 Context-Aware Mobile Web Service

Applying context-awareness in mobile Web service environments basically serves two purposes: (1) customisable service/content provisioning; (2) resolving drawbacks of wireless network communications.

In mobile Web applications (not mobile hosts), a number of approaches (Han et al.; 2005; Lee et al.; 2005; Kang et al.; 2007; Tian et al.; 2007; Sheng et al.; 2008) have applied different contexts as request parameters in order to provide feasible service to the service consumer. Han et al. (2005) have considered that applications

supported by the device and data transmission delay are the context influences in their system. Based on these contexts, the central controller service will redirect the request to the specific service provider in order to provide quality content to the requester. In other works proposed by Lee et al. (2005) and Kang et al. (2007), a service provider pushes service to subscribed clients autonomously based on clients' current context. The work proposed by Tian et al. (2007) has mainly focused on Quality of Service (QoS)-Aware service provision in which the service broker is placed between service provider and consumer. The service broker handles a client's request using a QoS requirement described by WS-QoS and discovering the feasible service provider who matches the requirement. The broker directly interacts with the service provider and retrieves the result for the client. The approach proposed by Sheng et al. (2008) uses a multi-agent technique to handle the client's context and act as a client's proxy to interact with Web service providers.

In MWS, Doulkeridis et al. (2007) aim to support context-aware service direction/discovery in which the approach considers user-related context, an available resource, a location and time context. This work involves a semantic model that structures the service direction recommendation based on both client and server-side context.

Different from a traditional Web service environment in which the service provider and clients are usually connected in a stable LAN or Wide Area Network (WAN), mobile nodes in MWS environments are usually dynamically connected to the wireless network such as Bluetooth, Wi-Fi, wireless broadband or satellite Internet depending on the user's location and network availability. In order to provide the best network connection for MWS nodes, context-aware vertical handover middleware (Pawar, Wac, Van Beijnum, Maret, van Halteren and Hermens; 2008) are proposed for supporting the prediction of the vertical handover. The middleware is capable of sensing the MWS provider node's movement, direction, and surrounding network. It enables the Mobile-Device-Hosted MWS (MHMWS) provider node to measure the network availability and perform the autonomous decision-making process based on

the utility algorithm for the vertical handover. Moreover, the system is capable of ensuring that the MHMWS client-service interaction will not be interrupted due to network switching.

Context-aware systems can be classified into two types: centralised and decentralised (distributed) (Chen et al.; 2000). All the works described in the previous paragraphs are centralised models. Centralised models rely on a central service in the network as a context information repository, whereas decentralised models do not rely on central context repositories. In a decentralised model, context information needs to be disseminated to participating peers and each peer needs to provide mechanisms to manage and process context information. Each model has its advantages and disadvantages derived from its fundamental network topology (Gehlen; 2007). For instance, the centralised approach faces the single point of failure challenge, and the decentralised approach faces the latency challenge due to the large amount of message exchange in the network.

Some authors such as Doulkeridis et al. (2007) classified their work as a decentralised model. In fact, the JXTA-based framework, which is a super-peer-based model, is still a centralised model. When the environment relies on super-peers to manage context information for their groups of peers, single point of failure can still happen, resulting in the failure of the the entire context-awareness mechanism.

Currently, most mobile Web service architectures do not incorporate a "complete" decentralised context-aware model. Existing decentralised context-aware models are based on standalone technologies such as mobile agent frameworks (Gunasekera et al.; 2010; Rhodes et al.; 1999), or standalone routing protocols (Palazzi and Bujari; 2011; Wang et al.; 2005), or standalone middleware frameworks (Palazzi and Bujari; 2011). These works were not designed to be compatible with Web services. Apart from standalone technologies, Gehlen (2007) has proposed a WS-Eventing based context-aware MWS approach, which is fully compatible with Web services. However, a detailed performance evaluation was not described in his work.

Additionally, context-aware MWS provisioning may involve semantic related technology such as utilising ontology metadata to manage context information. At the time of writing this thesis, no work has addressed a context-aware semantic MWS architecture for pure P2P networks. A peer in such an environment is required to process many documents including parsing WSDL, SAWSDL, XML Schema, OWL and reasoning semantic metadata and context information metadata. These processes can cause high latency for resource-constrained mobile devices. One possible solution is to improve the semantic metadata process scheme (Steller and Krishnaswamy; 2008). Another solution is to offload partial tasks to external cloud services, which is described in the next section.

2.5 Mobile Web Service Integration

Offloading processes to a resource-rich remote service provider can improve performance of MWS. A previous work proposed by Asif et al. (2008) partitioned tasks to a remote server to relieve XML document processing at the mobile host. The work is an example of static configuration-based offloading. Recently, cloud services aim to provide a more elastic mechanism. This section provides an overview of cloud computing and mobile cloud computing for MWS integration.

2.5.1 Cloud Computing

In 2006, Amazon launched a new form of networked services—Elastic Compute Cloud (EC2). In the same year, Google's CEO, Eric Schmidt, started to use the term *cloud computing* (Bogatin; 2006) to describe a new computing paradigm. Since then, cloud has become a commonly accepted term to represent the evolved *Resource as a Service* computing models. The resources in cloud computing represent hardware and software computing resources that are accessible remotely over the Internet (cloud).

A service in cloud computing, also known as cloud service, is a collection of existing technologies with new models. Fundamentally, there are three basic service models of cloud computing:

IaaS—An IaaS service is the most basic type of cloud service. The IaaS provider provides computer hardware resource for remote client to access for a period of time. A representative IaaS is Amazon EC2, which allows its client to choose the hardware resource capability (e.g., CPU, RAM, ROM, which is known as a virtual machine), and software resources (e.g., OS). Initially, EC2 lets clients choose which OS and what software they want to install from a list of available software on the provider side. Client can also upload his/her own software image and install it in the virtual machine to achieve his/her needs. Basically, IaaS is very useful for outsourcing purposes. Users can rent a high performance virtual computer to process some tasks remotely for a period of time without purchasing his/her own computer hardware.

PaaS—PaaS is very similar to traditional Web hosting. In fact, it is difficult to differentiate between them. PaaS can be seen as the advanced Web hosting service that provides additional features to achieve complete solution for the need of Web application development. Traditionally, developing and launching a Web application requires many resources such as Web server, database server, host space and security service. A traditional Web hosting service rarely provides all the required features, and commonly only supports applications written in one particular programming language. Recently emerged PaaS providers such as GAE, provide a more complete solution for Web application developers to programme, launch and test applications on the remote GAE server. GAE supports multiple programming languages and Google provides many APIs for developers to implement their applications.

Software as a Service (SaaS)—SaaS provides applications to end users via the medium of the Internet. An example is Google Docs,[36] which is a combination

[36]See `http://docs.google.com/`

of virtual storage and document processing applications. Users can simply use Web browsers to access Google Doc to create and to edit documents stored in the server-side without installing any client-side applications.

Besides the three basic cloud services described above, there are many different types of cloud services in the market. One is the cloud storage service, which is a combination of virtual storage service and Web application. Recent cloud storage services usually provide online virtual hard drive space for file hosting, file synchronisation and client-side applications.

Commonly, when a user installs the cloud storage client-side application to his/her device, a directory/folder is created in the local storage of the device. The directory/folder is then configured to synchronise to its corresponding virtual storage space at the server-side. If any change is made from the client-side (e.g., a file is put into the directory/folder), the virtual storage space will automatically synchronise its content with the client-side. Therefore, the synchronisations can be achieved for all the user's devices that have installed the appropriate client-side application. An additional feature of cloud storage is file sharing. For example, the popular cloud storage service Dopbox's client-side allows the user to obtain a static URL of a file in his/her Dropbox space and use the URL to share the file. Other cloud storage services also provide the file sharing feature in different forms. Some recent representative cloud storage services include Dropbox,[37] Amazon S3,[38] Amazon Cloud Drive,[39] Skydrive[40] and Google Drive.[41]

iCloud's cloud storage service is different from the other cloud storage services. Although iCloud is a collection of cloud computing services, it does not provide flexibility like the other cloud storage services do. iCloud only synchronises particular files (e.g., music files in iTunes) to the cloud storage. Users are unable to use

[37]See `https://www.dropbox.com/`
[38]See `http://aws.amazon.com/s3/`
[39]See `http://www.amazon.com/gp/feature.html?ie=UTF8&docId=1000828861`
[40]See `https://skydrive.live.com/`
[41]See `https://drive.google.com/?`

the iCloud's storage feature as they use other cloud storages (synchronising with a folder).

Since iCloud is a collection of cloud computing services, it also provides other services such as synchronisation of Calendar, Reminder and Notes to all the user's devices, as well as a device tracking service that displays the current location of the user's device (e.g., Macbook, iPhone, iPod or iPad) on a map.

2.5.2 Mobile Cloud Computing

Mobile Cloud Computing (MCC) represents a composition of mobile computing and cloud computing. The definition of MCC is distinguished by application domains. Larosa et al. (2011) have noted that *'currently, there is no universal definition of mobile cloud computing'*.

Generally, MCC has two definitions. On the one hand, MCC represents a system consisting of cloud services and mobile client applications in the form of a client-server model. In such a system, mobile clients utilise cloud services as some of their computing resources to offload computational tasks (e.g., data processing) or store/synchronise data (Dinh et al.; 2011; Fernando et al.; 2013). Conversely, MCC represents a system consisting of cloud services and mobile hosts. In such a system, mobile devices are not only clients of cloud services, but also service providers that participate in a distributed system. For example, Mobile Cloud Middleware (Flores et al.; 2012) utilises MWS for asynchronous messaging.

Computation offloading (or offloading for short) is the most common way of utilising cloud services in the mobile application domain. Offloading in MCC is a form of computational process migration but it is different from the traditional grid computing system in which all resources are managed as one system. Offloading in MCC is not like a classic client-server system in which client-side applications always migrate particular processes to remote server. Offloading in MCC denotes that some computational processes are migrated from the current managed system

environment to a remote external environment as an instance process handled by the more powerful computers for a period (Kumar et al.; 2012).

The fundamental purpose of offloading is to improve performance and to save energy consumption on the mobile device. A mobile client application accesses a cloud service and migrates its computation tasks to the cloud service for a period depending on its needs. In the case of utilising the IaaS-based cloud service, which is the virtual machine-based service, the service provider lets clients offload programs and data, isolated and protected.

Although cloud services can improve the performance of mobile applications, its efficiency depends on many factors such as the availability of network bandwidth in the mobile client's current environment and the amount of data exchange between mobile client and cloud service (Kumar et al.; 2012). Hence, some researchers have proposed a dynamic reconfigurable runtime architecture using context-awareness for MCC (Papazoglou et al.; 2007; Papakos et al.; 2010). These works adapt context-aware mobile service mechanisms to MCC to enable adaptation of a runtime reconfiguration of MCC.

Besides the above fundamental challenges in the mobile computing domain, MCC also faces a challenge in reliability. In October 2012, both Amazon and Google's cloud services failed due to memory leaks (Williams; 2012a). Google's cloud service failure also caused a chain-effect in which Dropbox and Tumblr (micro blogging website) have been affected (Williams; 2012b). In fact, such a single-point-of-failure issue always exists in distributed computing but it appears that when this happens in cloud service system, it causes more serious problems. This remains an in-progress research domain.

2.6 Summary

This chapter presented a literature review of the state of the art developments in the MWS domain. The review started with the fundamental knowledge of Web service standardised by the W3C (2004). The common Web service infrastructures, which

include central registry-based, index-based and P2P-based, were described. Then, different types of Web service communication protocols—SOAP and REST—were compared. Web service description methods associated with Web semantics were also described.

Midway through this chapter, technologies for enabling MWS were discussed, including discussions on how MHMWS providers publish themselves and how they can be discovered. Related technologies—Apache River, JXTA, Bonjour, UPnP and DPWS—were reviewed and compared. Afterwards, this chapter reviewed a number of MWS approaches that applied context-aware models to enhance overall performance in terms of providing customisable/personalised services to mobile users, and resolving the drawbacks/limitations of mobile network communication for MWS. Finally, how cloud computing has been applied in the mobile application domain was explored, in particular how the cloud computing has been applied in the mobile application domain.

In summary, this chapter covers the fundamental background of this thesis and the technologies to realise service-oriented MSNP.

Chapter 3

Towards Mobile Social Network in Proximity

3.1 Introduction

In the past several years, a number of researchers have proposed the proximal location-based social network system architectures (Kern et al.; 2006; Pietiläinen et al.; 2009; Sapuppo; 2010) and platforms (Pernek and Hummel; 2009; Tsai et al.; 2009; Xing et al.; 2009; Rana et al.; 2010; Toninelli et al.; 2011). While many works (Borcea et al.; 2007; Lubke et al.; 2011; Brooker et al.; 2010; Yu et al.; 2011; Yang et al.; 2008) provide central mediation services to either replace existing SNS or to leverage existing SNS with proximal location-based social interaction capability, some researchers (Xing et al.; 2009; Pietiläinen et al.; 2009; Rana et al.; 2010; Toninelli et al.; 2011) intend to support decentralisation to overcome the limitations of the centralised systems.

Fundamentally, a common design model for a centralised proximal location-based social network system is based on a Web service-oriented architecture (e.g., MobiSoC [Borcea et al.; 2007], MobilisGroups [Lubke et al.; 2011]) in which the system can enable heterogeneous mobile applications to participate in the environment using standard interface and protocol. However, Web-service oriented architecture is less

common in decentralised proximal location-based social network systems, because supporting Web service-oriented architecture requires a large number of message-driven processes (e.g., processing XML-formatted semantic metadata on mobile devices), which can cause high latency in the bootstrap and service discovery phase. A decentralised system does not have a powerful static central server for processing a large number of complex metadata, and most mobile devices have limited processing power. Considering the overall performance, most decentralised proximal location-based social network applications were designed as tightly-coupled systems, such as the Jini-based system (Brooker et al.; 2010), which uses Java objects as the means of communication without getting involved with complex metadata parsing as in Web service-oriented systems.

Applying loosely coupled Web service architecture enhances the interoperability of heterogeneous platforms and applications in proximal location-based social network environments. Application developers can follow the Web service standard technologies and protocols to develop different applications for various platforms, which are still able to interact with each other.

In the last decade, numerous researchers (Pratistha; 2002; Srirama et al.; 2006; Doulkeridis et al.; 2007; Pawar et al.; 2007) have proposed mobile-device-hosted Web service (MWS) solutions. By applying MWS to the proximal location-based social network system, the system can achieve loose coupling, which highly improves its flexibility and interoperability. However, applying MWS to a proximal location-based social network faces many challenges in terms of trust, latency and resource management, which will be described in the following sections of this chapter.

In this thesis, Mobile Social Network in Proximity (MSNP) represents the decentralised proximal location-based social network operating in a public wireless network environment. In this chapter, we first provide an overview of the background of MSNP in Section 3.2. Following in Section 3.3, we review and compare a number of related frameworks for enabling MSNP. Section 3.4 describes our proposed

decentralised service-oriented MSNP architecture. Section 3.5 provides a discussion of the proposed architecture.

3.2 Background

MSNP is derived from MSN and Location-Based Social Network (LBSN). The root of MSNP is MSN, which represents the generic social network application designed for mobile clients. A generic MSN application refers to the mobile OS version of a SNS client, such as Facebook Android OS version. LBSN is a subset of MSN, which emphasises the geographical-based social content sharing such as the real time check-in feature provided by Facebook, or a user's comment about a local restaurant shared with his/her friends using Foursquare. The main difference between LBSN and MSNP is the physical geographical coverage. The location-based content or information shared in LBSN can refer to any place in the world, and is shared by the mobile device user's community globally on the Internet. Conversely, MSNP emphasises on the proximal social interaction in which the social content or information is shared with the mobile device user's nearby participants in a fairly close range using the short range wireless network communication technology (e.g., Wi-Fi or Bluetooth).

This section first summarises the background of MSNP's roots—MSN and LBSN—and then identifies the features of MSNP. Finally, we describe the requirements of enabling MSNP and propose a service-oriented MSNP architecture.

3.2.1 Mobile Social Network

In general, *'a mobile social network is a virtual community for individuals to connect with others using mobile devices, such as mobile phones and PDAs'* (Tang and Kim; 2011). Kayastha et al. (2011) have defined MSN as:

> *'A heterogeneous network where mobile users carrying mobile devices interact and share user-centric information with each other using socially aware algorithms to achieve better QoS. Therefore, MSN is a user-centric mobile communications system in which the methods of social network analysis (SNA) can be applied to analyse the structure and ties among mobile users with the objective of improving the efficiency of publishing and sharing information.'*

MSN can be classified into two basic types—Web-based MSN and decentralised MSN (Kayastha et al.; 2011):

- *Web-based MSN* is often referred to as a centralised SNS that requires Internet access from mobile client applications. Examples are Facebook, Twitter and Sina Weibo,[1] which provide mobile OS client applications. Based on the design of service provisioning, Web-based MSN is further classified into three subtypes (Zhong et al.; 2008):
  - *Website-based MSN*. Numerous social networking websites provide 'mobile-friendly' versions of a user interface when they are accessed by the mobile devices' Web browser application. Fundamentally, they are still the original SNS websites. The Twitter mobile version website[2] is an example of such a service.

[1] See `http://www.weibo.com/`
[2] See `http://mobile.twitter.com`

  - *Mobile app-based MSN*. Many SNS providers have implemented the client-side applications natively for popular mobile OS. These native SNS applications have re-designed user interfaces to provide a better user experience for the mobile users. However, many do not provide the full functionalities of their original SNS websites.
  - *Original MSN* are MSN services originally designed for mobile devices. These services utilise mobile devices' functions such as real-time location-aware technology to provide location-based services or even to support location-based social network services. Foursquare is one example of an original MSN service, which was initially developed for mobile OS such as iOS and Android. The details of location-based social network will be discussed in Section 3.2.2.

- *Decentralised MSN* represents an environment consisting of a group of users utilising their mobile applications to form a social network group dynamically without a centralised server. In this environment, users can perform real time interaction with one another based on their common interests without knowing each other in advance. They can exchange and share content or information when they connect with one another in the opportunistic network topology of mobile wireless network (e.g., Wi-Fi, Bluetooth) environments. Currently, decentralised MSN has not yet become a commercial product for the market, but researchers have been investigating and designing this type of MSN for many years. MobiClique (Pietiläinen et al.; 2009) is one such research project that intends to develop a decentralised MSN.

In the early years, Web-based and mobile app-based MSN were the mainstay for MSN applications. However, in recent years, original MSN applications have become popular. Many such applications utilise location-based services to provide real-time location-aware social interaction to mobile users. The success of LBSN applications motivated the providers of classic Web-based MSN and mobile app-based MSN to

also support similar location-aware features. In the next section, we discuss the location-based social network (LBSN) in detail.

3.2.2 Location-based Social Network

LBSN is a composition of MSN and Location-based service (LBS). LBS, such as the Google Map for mobile[3], which enables mobile device users to retrieve geographical surrounding information based on their current location retrieved from real time location tracking services, has become one of the most popular mobile applications today. A location in LBS can be provided in two ways: *absolute*, which shows the physical latitude-longitude coordinates; and relative, such as 100 meters east of Melbourne Central Shopping Centre, or a *symbolic* notion such as a *home*, *school* or *library* (Zheng; 2011). LBS providers deliver various geo-related information to their consumers to help them search for a particular place, or to direct them to the place based on their preferred travel method (by walking, by driving/riding, or by public transport etc.), and also, to provide them information about the place submitted by other users. For example, a user can retrieve other people's public comments about a local restaurant on Google Map. Such a feature is the result of the composition of MSN and LBS, which is also known as location-based social network (LBSN).

Today, SNS providers have widely combined LBS with their MSN services to support LBSN. LBSN is a subset of MSN, since researchers (Zheng; 2011; Lindqvist et al.; 2011; Shankar et al.; 2012) often describe LBSN as an environment in which *people carry mobile devices and use location/position sensors/services to enable LBSN*. A less common terminology used to refer to such an environment is Geosocial Network (Carbunar et al.; 2012), which serves the same purpose and the same software architecture.

Depending on a user's privacy preference, content shared in LBSN can be public (visible to any worldwide users) or private (a group of selected people in the user's

[3]See `http://www.google.com/mobile/maps/`

social network). An example of public visible content in LBSN can be the commenting on and ranking of a restaurant that has been posted by a Google+ user (either with his/her anonymous or real name) on Google Latitude/Map. Private LBSN content can be a picture captured in a specific place and posted on a user's SNS space with tagged place name and time, and is only visible to the user's social group.

In general, LBSN services can be categorised into two types:

- *Purpose-driven.* In such an application, a user can explicitly request a server for another user's current location. Examples include AT&T FamilyMap,[4] Glympse[5] and Verizon Family Locator[6] (Lindqvist et al.; 2011). These services are usually provided for small private social groups users, such as for family members tracing each other's current location.

- *Social-driven.* This is a more commonly seen feature of today's MSN services. Social-driven services usually provide the following features:

  - *Check-in.* When a LBSN user arrives at a place, they uses their LBSN application to obtain a list of possible place names based on his/her current location. Then, the user selects one of the place names and posts the check-in information to share within his/her social network.

  - *Geo-tag content.* Geo-tag content can be a simple text comment about a place (restaurant, shopping centre, train station etc.), or a picture or video about the place that is shared within the user's social network. The content is tagged with a specific time and a specific location information with a hyperlink to redirect readers to see where the place is on Google Map, or with a piece of embedded map image showing the explicit location of the place on the map. People usually refer to the comments provided by their social groups to learn about new places. Zheng (2011)

[4]See `https://familymap.wireless.att.com`
[5]See `http://www.glympse.com`
[6]See `http://products.verizonwireless.com`

mentioned that LBSN, such as geo-tag content, not only simply adds physical location and time to content, but also provides knowledge about people's interests and history. The user's social network participants can refer to such geo-related information to discover their common interests and get to know each other better.

– *Establishing new connections.* The user experience survey provided by Lindqvist et al. (2011) indicated that some Foursquare users do use LBSN to meet with new friends. As mentioned in the study by Zheng (2011), public content shared in LBSN lets participants discover people who have common interests based on the places they have checked-in, and the comments they posted about the place. For example, a Foursquare user-A, who likes to try different new restaurants, will notice that user-B also likes to perform the same activity. Hence, user-A may contact user-B and start sharing the common interest together.

While mobile users spend most of their time accessing the Internet-based LBSN, they have missed many opportunities to interact with *physically surrounding people* in the real world for new friendships, business opportunities or information sharing (Borcea et al.; 2007). Although it is possible to establish new connections by using existing LBSNs, *the global Internet-based LBSNs do not provide the capability to enable real-time proximal-based new social connections in the user's current presence.* Consequently, works such as MobiSoC (Borcea et al.; 2007), MobiClique (Pietiläinen et al.; 2009), MoSoSo (Tsai et al.; 2009), Uttering (Allen et al.; 2010) and Spiderweb (Sapuppo; 2010) were proposed to enable a new breed of mobile social network applications, which can assist mobile users to interact with proximal people and perform various social activities such as searching for new friends who have common interests, exchanging content of common interest and establishing a conversation. In this thesis, we term such a proximal-based MSN environment MSNP.

3.2.3 Mobile Social Network in Proximity

A unique feature of MSNP compared to the other MSNs is that MSNP provides the opportunity for people to establish new social interaction with strangers in a public environment. MSNP lets people who do not know each other in proximity, but probably should, gain the opportunity to know each other (Sapuppo; 2010).

The fundamental notion of MSNP derives from two works: MobiClique (Pietiläinen et al.; 2009) and Local Social Network (LSN) (Sapuppo; 2010), in which participants use their mobile devices (e.g., smartphone) to exchange content with one another in physical proximity directly within a short range wireless network environment (i.e., Wi-Fi, Bluetooth). Since the author of MobiClique did not specifically give a name to such an environment, and the name LSN does not encompass the important elements—that is, *mobile devices* and *proximal mobile network connection* in such an environment—to highlight them and to distinguish the environment from generic MSN and LBSN, we use the term MSNP.

MSNP has two basic principles (Pietiläinen et al.; 2009; Sapuppo; 2010):

- *Decentralised operation.* MSNP operates in a MP2P network, in which participating mobile devices do not rely on intermediation entities to assist their communications (Sapuppo; 2010). Such a requirement avoids the single point of failure issue derived from centralised architecture, and the communication between participants can partially work without Internet connection.

- *Leveraging existing social networks.* MSNP applications link to their users' SNS (e.g., Facebook, Twitter, Google+) to enable common profile exchange capability (Pietiläinen et al.; 2009; Sapuppo; 2010) or to share online content by providing URL links. With this feature, MSNP applications are capable of performing common interest matchmaking and content recommendation for their users.

Content exchange or sharing is the most basic activity in a social network application. For example, the most common Facebook or Twitter activity is to post

text/feed or multimedia content (i.e., video, images, music) to share with subscribers (or friends). Other activities such as requesting to join a friend-list (or subscribing) or recommending friends to be added are also related to content exchange (user profile exchange). Hence, the fundamental capability of an MSNP application is to enable content exchange and sharing.

In the past several years, numerous works were proposed to enable proximal-based mobile social networking. Most of these works were tightly coupled systems. As mentioned by Kayastha et al. (2011), in general MSN, a standard interoperation interface has become an issue. Existing works lack common protocol and interfaces to seamlessly exchange information on social relationships, and data retrieved from different MSN. The standard is required not only for data exchange but also to support context-awareness and privacy and trust control. Hence, to support such a need, in this research project, we aim to provide a loosely-coupled service-oriented MSNP solution.

3.3 Requirements of Service-Oriented Mobile Social Network in Proximity

Enabling service-oriented MSNP involves a number of fundamental requirements. This section provides an overview of these requirements.

3.3.1 Content Management

Content management involves how an MSNP participant shares his/her regularly updated content in his/her device. In a classic P2P approach such as in the work of Buchegger et al. (2009), content is stored in the local storage of the participants' own devices. In the work of Datta (2010) and Sharma and Datta (2011), contents can be replicated in numerous existing participants to reduce the transmission of the original content provider. For instance, in a decentralised social network—SuperNova (Sharma and Datta; 2011)—super-peers can provide storage to assist

participants to distribute their updated content. Recently, cloud resources were utilised in various networked applications to reduce the burden caused by data transmission. In the works proposed by Au Yeung et al. (2009) and Seong et al. (2010), users' content can be stored in any trusted Cloud storage (e.g., Amazon Simple Storage Service [Amazon S3][7] or Dropbox)[8] by the user's own choice. A simple URL can be provided in the corresponding metadata to redirect a client to the content stored in the Cloud storage.

3.3.2 Identification

Since a central management party is not available in MSNP; managing the identification of each participant is a challenging task. An improper solution may cause the redundant identifications to be allocated to different participants. While some relevant works (Buchegger et al.; 2009; Datta; 2010) assumed that a unique identification is allocated to each participant who joins the network, some researchers have specified more completed solutions. Au Yeung et al. (2009) utilised Web ID, which was based on the format of Uniform Resource Identifier (URI). For example: `http://msnp.org/b-card#johan`. The work based on Extensible Messaging and Presence Protocol (XMPP)[9] (Lubke et al.; 2011) mentioned that XMPP offers the possibility for identification in a format similar to email. For example: `user@server.org/resource`. Finally, Seong et al. (2010) have applied OpenID[10] to ensure the ID redundancy issue will not occur in the network. An OpenID is an email address that can be used to login to numerous associated online services such as YouTube,[11] Picasa,[12] Gmail,[13] Flickr,[14] Facebook and MySpace.[15]

[7] See `http://aws.amazon.com/s3/`
[8] See `http://www.dropbox.com/`
[9] See `http://xmpp.org/`
[10] See `http://openid.net/`
[11] See `http://www.youtube.com`
[12] See `http://picasa.google.com/`
[13] See `https://mail.google.com`
[14] See `http://www.flickr.com/`
[15] See `https://myspace.com/`

3.3.3 Access Control and Trust

Access control in a decentralised social network usually relies on each participant to manually setup the content access permission (Datta; 2010). Access control in the work of Sharma and Datta (2011) was based on the weight of the collaborative trustworthiness rating. Kourtellis et al. (2010) have defined four access polices based on four social information: relations, labels, weights and location. However, these works did not consider that the trust should be context-dependent (Wang and Vassileva; 2007). A requester—X—might trust Y's music-sharing service, but X might not trust Y's picture-sharing service because X has concerns that Y's pictures may contain undesirable content.

In MSNP, a reliable central management party for supporting trustworthiness is not available. Hence, each MSNP participant needs to manage the access control by themselves. In a common SNS environment such as Facebook, a user can define different levels of accessibility to his/her content, and the user's contacts who were assigned different social groups can only access/see the content authorised to them. A similar approach has been applied in a related work of MSNP proposed by Kourtellis et al. (2010). However, for a new MSNP participant who does not have many contacts, it is hard to clearly define such access control.

3.3.4 Bootstrap and Service Discovery

Bootstrapping is a challenging task in MSNP since a new participant may not have any knowledge about the other participants. For a tightly-coupled solution such as MobiClique (Pietiläinen et al.; 2009), which utilises Facebook's user profiles to establish relationships between participants, it is not applicable to participants who do not have a Facebook account. Au Yeung et al. (2009) applied Friend Of A Friend (FOAF) documents to support friendship discovery. Each device contains a FOAF document to describe the social relationships of its user. However, the FOAF approach does not benefit a new participant who does not have any friend in the environment. A highly distributed decentralised social network—SuperNova

(Sharma and Datta; 2011)—has adapted super-peers to assist new participants in the network. Super-peers are active participants who intend to assist new participants in their early stages to find new friends/subscriptions by disseminating their profiles.

Eventually, a super-peer will gain reputation by performing such assistance and may further gain commercial benefit. Each super-peer can provide different kinds of services. Some super-peers provide a user-list, which allows a new participant to discover the existing participants who have common interests or who can provide some content of interest to the new participant. Some super-peers are recommenders who intend to actively recommend new friends/subscriptions to the new participant based on the participant's interests.

The preceding description provides the conceptual approach for bootstrap based on existing decentralised social network systems. The fundamental challenge is how to realise the discovery mechanism in MSNP. A common approach is to utilise the Distributed Hash Table (DHT) technique (Buchegger et al.; 2009; Datta; 2010; Xing et al.; 2009). In a classic P2P network, each participant maintains its DHT, which contains information about the other participants, such as their network addresses, provided services or even some text messages (Datta; 2010). Besides these static description-based approaches, Seong et al. (2010) and Li et al. (2012) have also applied semantic technologies to enhance the discovery process.

3.3.5 Adaptive Resource Management

One of the challenges of MSNP application is resource management (Brooker et al.; 2010; Yu et al.; 2011; Rana et al.; 2010), which derives from the fundamental mobile network topology. Since most existing related frameworks (Pietiläinen et al.; 2009; Xing et al.; 2009) were designed as tightly coupled systems, in which the participants of such environments do not face many resource constraint issues, they apply specific communication protocols to support the best performance of their systems. Conversely, when loose coupling is required, in which the MSNP participants utilise standard message-driven protocols such as SOAP and OWL to communicate

(Toninelli et al.; 2011), the participating mobile devices will face resource intensive challenges derived from disseminating and processing a large number of standard formatted documents. In order to reduce the resource usage of mobile devices, Pernek and Hummel (2009) utilised a remote centralised server for MSNP participants' discovery processes. Such an approach potentially faces the single point of failure issue. Instead of relying on a central repository server, the decentralised social network platform—Contrail (Stuedi et al.; 2011)—utilised the cloud services to offload the communication tasks to the runtime established cloud services (see Chapter 2, Section 2.5.2 for mobile Cloud computing offloading). Such an approach can reduce the burden of mobile device processes and also avoid the single point of failure issue.

3.4 Enabling Proximal-based Mobile Social Network

Before we introduce our proposed design of MSNP, we first review and compare a number of existing MSN solutions that were proposed to also support proximal-based social network interaction. These existing works can be categorised into client-server models, semi-decentralised models and decentralised models, and we call them Proximal-based MSNs (PBMSN) to distinguish them from our MSNP. A PBMSN provides proximal-based social interaction, but not necessarily following the two principles of MSNP described in Section 3.2.3, which are *decentralised* operation and *leveraging existing social networks*.

3.4.1 Client-Server Proximal-based Mobile Social Network

As Figure 3.1 shows, client-server PBMSN (Borcea et al.; 2007; Banerjee et al.; 2009; Schuster et al.; 2010; Sapuppo; 2010; Lubke et al.; 2011) utilised portal-like central services to support users to discover and to interact with other participants based on the information retrieved from miscellaneous SNS. To discover proximal

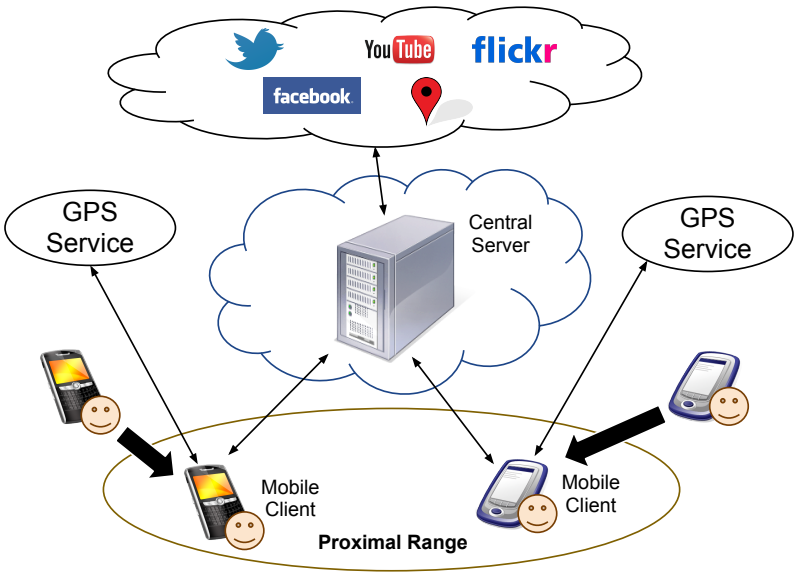


Figure 3.1: Client-server PBMSN

participants, a client-server model facilitates a location-tracking service (e.g., GPS, Google Maps)[16] to continue tracing each participant's current location. Benefiting from a broad range of distributed resources, client-server models are capable of providing high performance and effective services to mobile users. However, client-server models have a number of drawbacks such as the bottleneck problems and the location tracing approach requiring users' devices to frequently send the current locations to the central server, which tends to consume a lot of the battery life of the mobile devices.

Client-server PBMSN has the following advantages and disadvantages:

Advantages

- *Efficient discovery and matchmaking processes.* Client-server PBMSN benefits from the support of the powerful central server to process data. It is very efficient in performing the matchmaking process for users to discover and establish new social networks based on their location information and common interest profiles.
- *Easy maintenance.* Since mobile-side applications are simple client-side software, it is much easier to maintain and to perform the version update

[16]See `http://maps.google.com`

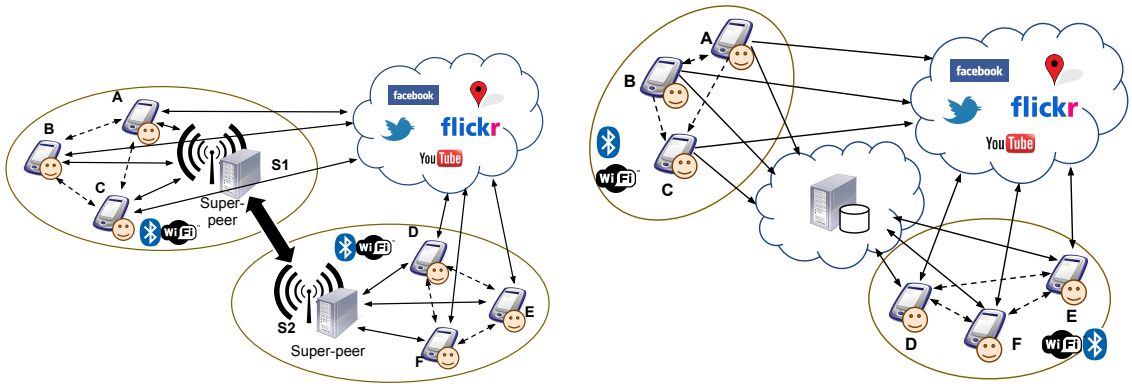
of the system when compared to the pure mobile P2P-based decentralised model.

Disadvantages

- *Less control in user privacy.* Client-server PBMSN requires users to upload their data to the central repository. Existing SNS such as Facebook have provided various privacy settings for users to control who can see their content. However, some users have concerns about how the central repository service provider protects their data against malicious hackers or repressive governments (Stuedi et al.; 2011).
- *Single point of failure of server-side.* It will lead to the failure of the entire MSN. Further, mobile device users' movements are dynamic in nature. Their wireless Internet connection cannot be guaranteed. When a mobile user loses his/her connection with the remote central server, his/her mobile device will not be able to discover other mobile social network users in his/her proximity due to the application purely relying on the server-side operations for the discovery process.
- *Always-on data transmission channel.* To provide proximal user discovery and profile matchmaking, a client-server-based system usually requires the mobile client application to retain its communication channel between itself and the remote server in order to let the server track the mobile user's position. Such an always-on data transmission channel consumes a lot of the battery life of a mobile device.

3.4.2 Semi-decentralised Proximal-based Mobile Social Network

A semi-decentralised PBMSN consists of partial centralised nodes and a MP2P network. The semi-decentralised model can be further classified into two models: super-peers model (see Figure 3.2(a)) and central repository model (see Figure 3.2(b)).



(a) Super-peers model (b) Central repository model

Figure 3.2: Semi-decentralised MSN Model

- In a super-peers model, each MSN environment requires at least one active participant to assist participants' discovery process. Figure 3.2(a) illustrates an example of super-peers model in which two pre-connected super-peers—S1 and S2—act as brokers to assist two groups of users located in different WLAN discover on another. Peer A, B or C are capable of requesting or advertising SNS content with peer D, E or F via S1 and S2. The two related works of Kern et al. (2006) and Tsai et al. (2009) have applied a JXTA framework to enable such an environment.

- A central repository model (McNamara and Yang; 2008; Yang et al.; 2008; Pernek and Hummel; 2009; Sapuppo; 2010; Brooker et al.; 2010) utilises static central servers to assist MSN participants to discover their proximal peers based on certain criteria (see Figure 3.2(b)). The central server is either implemented in a particular location (e.g., Jini-based model, see [Brooker et al.;

2010]) or is accessible via the Internet as a global broker (Sapuppo; 2010). Participants' social activities are still operating in the direct MP2P network.

Semi-decentralised PBMSN has the following advantage and disadvantage:

Advantage

- *Reduced burden.* The burden of the central server is greatly reduced because either the super-peers or registry servers are used for discovery only. Other social activities are done by P2P or distributed to other entities.

Disadvantage

- *Single point of failure.* A super-peer model will fail if an environment does not have a super-peer, powerful enough to handle all the discovery processes. A static repository model will also face similar problems.

3.4.3 Decentralised Proximal-based Mobile Social Network

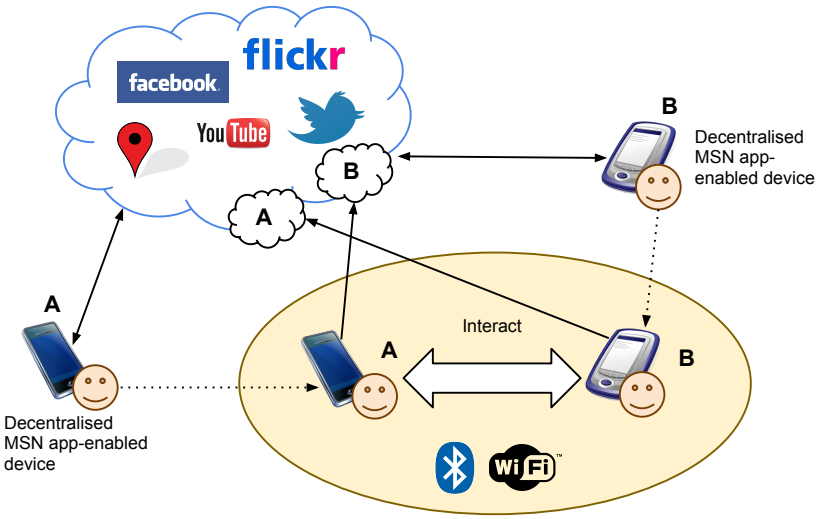


Figure 3.3: Decentralised PBMSN

A decentralised PBMSN aims to overcome the limitations of the centralised solutions by utilising a MP2P network approach. Figure 3.3 illustrates a decentralised

PBMSN scenario. As the figure shows, each participating peer equips a common decentralised MSNP application. The application associates with its user's SNS in order to share content to proximal peers when they meet each other in proximity with the capability of Wi-Fi or Bluetooth communication.

Existing decentralised PBMSN are still in their early stages. Pietiläinen et al. (2009) and Xing et al. (2009) focused on how to enable the SNS activities in MP2P network environments. Others have focused on how content can be shared. In Uttering, Allen et al. (2010) have modelled the user interest profiles, and also introduced a formal mathematical scheme to decide how the content can be proactively pushed to the friends/contacts who have potential interest in the content. In MobiSN, Li et al. (2012) have proposed ontology-based formal semantic models to enable content sharing using a semantic content matchmaking scheme. The approach enables user-interests content routing in decentralised PBMSN based on user profile similarity measurement strategies. However, these two works did not provide a generic architecture or framework for enabling PBMSN.

Further, most existing decentralised PBMSN solutions (Pietiläinen et al.; 2009; Xing et al.; 2009; Rana et al.; 2010) are tightly-coupled, which have limited flexibility and scalability. Ideally, participating in MSNP should be flexible. Users can use the application by their own choice just like participating in a popular online content sharing network such as BitTorrent.[17] Moreover, developers should also have the flexibility to implement their own applications to interact with such MSNP environments. Hence, existing decentralised PBMSN solutions have not yet met the fundamental requirement of MSNP.

A decentralised PBMSN has the following advantages and disadvantages.

Advantages

- *Full control of private content.* Participants in decentralised PBMSN have full control over sensitive materials such as their profiles. They can

[17] See `http://www.bittorrent.com/`

decide where the data is to be stored (e.g., in their own Cloud storage) and with whom and how the data is shared.

- *No single point of failure.* Decentralised model-based MSN can potentially avoid single point of failure. Activities can still be performed partially in a decentralised model-based MSN even when the Internet is not connected.

Disadvantages:

- *Resource intensive.* The model can be resource intensive when the message-driven service-oriented solution is applied. The service discovery processes can also face latency issue.
- *Complexity.* Development is more complex compared to the centralised model

3.4.4 Comparison of Existing Works

Table 3.1 summaries and compares the above mentioned existing MSN middleware frameworks. Works such as Uttering (Allen et al.; 2010) and MobiSN (Li et al.; 2012), which were proposed for resolving specific challenges in PBMSN, are not included in this comparison since they are not the completed framework.

The comparison is based on the following criteria, denoted by the columns in the table:

- *Architecture* (**Archi.**) represents the base model of the framework. The three basic PBMSN models are: client-server, decentralised (DC) and semi-decentralised (Semi-DC).
- *Proximal Discovery* (**PD**) is the means by which participants discover one another in their proximity.
- *Auto Match* (**AM**) denotes how the system enables autonomous discovery and filtering based on user profile or context.

- *Trust* (**Tru.**) specifies whether the system support trust control.
- *Reducing Latency* (**RL**) represents whether the system provides a strategy to reduce latency in the bootstrap and discovery phase.
- *Resource-Aware* (**RA**) denotes whether the system supports a scheme to adapt dynamic changes at runtime to effectively select the most appropriate approach for social network activities.
- *Loose coupling* (**LC**) denotes whether the system supports loosely coupled interoperability for heterogeneous mobile devices and applications.

Work	**Archi.**	**PD**	**AM**	**Tru.**	**RL**	**RA**	**LC**
MobiSoC (Borcea et al.; 2007)	Client-server	Centralised Eventing	User Profile and Location	No	No	No	SOAP
MobilisGroups (Lubke et al.; 2011)	Client-server	Centralised	Manual Location Profile	No	No	No	XMPP
Smart Campus Project (Yu et al.; 2011)	Client-server + Minor DC	Bluetooth	Manual	No	No	No	No (OSGi)
SPN (Yang et al.; 2008)	Client-server + Minor DC	Bluetooth	Manual	No	No	No	No
Jini-based MSN Project (Brooker et al.; 2010)	Semi-DC	Jini	Manual	No	No	No	No
SocioNet (Pernek and Hummel; 2009)	Semi-DC	Bluetooth	FOAF Profile	No	No	No	Web Service
MobiSoft (Kern et al.; 2006)	Semi-DC	JXTA	FOAF RDF	No	No	No	No (Tracy2 + JXTA)
MoSoSo (Tsai et al.; 2009)	Semi-DC	JXTA	Manual	No	No	No	No
Spider Web (Sapuppo; 2010)	Semi-DC	Bluetooth	Manual	No	No	No	No
Proximiter (Xing et al.; 2009)	DC	OLSR	Manual	No	No	No	No
Mobi Clique (Pietiläinen et al.; 2009)	DC	Bluetooth	Social Profile	No	No	No	No
Cloud Semantic MSN (Rana et al.; 2010)	DC	Mobile Agent	Semantic	No	No	No	No
Yarta (Toninelli et al.; 2011)	DC	SLP	Semantic	No	No	No	RDF

Table 3.1: Comparison of MSN Frameworks

An ideal MSNP framework should support the following capabilities:

- *Decentralised*, which can avoid single point of failure issues.
- *Autonomous discovery*, to support a mechanism to improve the discovery result.
- *Trust*—an MSNP system should support trustworthiness to help users interact with people who are not in their contact list.
- *Loose coupling*, to enhance the interoperability of a heterogeneous platform.
- *Latency reduction*—a loosely-coupled MSNP system faces latency challenges. The system should provide a proper strategy to reduce the latency of the message-driven discovery.
- *Resource-awareness*—MSNP activities should be performed by using different approaches. Different approaches require different resource usage. An MSNP system should support being resource-awareness, which adapts to dynamic changes.

Existing related frameworks are still in their early stages. None of the frameworks mentioned in Table 3.1 supports all the above ideals of MSNP.

A purely centralised framework such as MobiSoC (Borcea et al.; 2007) and MobilisGroups (Lubke et al.; 2011) potentially harbours the risk of single-point-of-failure. Some centralised solutions such as Smart Campus Project (Yu et al.; 2011) and SPN (Yang et al.; 2008) support minor decentralised communication capabilities by utilising Bluetooth technology when the central server is not available. However, such a solution is insufficient, because by simply utilising Bluetooth-based discovery, it can result in high latency especially when the environment grows.

Most existing works also lack support for heterogeneous platform interoperability. As the table summarised, most frameworks were proposed in the form of stand-alone technology. Within these frameworks, some have applied standard service-oriented

technologies. MobiSoC is a Web service-based framework that applied SOAP communication. MobilisGroups has utilised IETF XMPP, which is a popular centralised standard communication protocol. SocioNet is also a Web service-based framework. Yarta has utilised standard protocol—Service Location Protocol (SLP)—for proximal mobile P2P discovery. SLP is a similar technology to Bonjour which has been described in Chapter 2, Section 2.3.3. Yarta also applied standard semantic discovery technology based on RDF to realise autonomous discovery.

Within these related frameworks, Yarta is the closest framework to achieve the basic capabilities described previously. It is capable of avoiding a single point of failure, and it supports heterogeneous platform interoperability and autonomous discovery. However, Yarta has not provided a strategy to reduce latency caused by applying standard semantic discovery technology in a MP2P network. The evaluation result of Yarta's prototype has indicated that this is an issue. Further, Yarta has no support for resource-awareness, in which the discovery and interaction scheme should adapt to the resource changes and environmental factors.

Overall, existing works did not address trustworthiness, which is an important aspect of MSNP because MSNP allows users to interact with new people who are not in their existing contact list. Without a proper strategy for trust, people will hesitate to use MSNP. Further, applying trust in MSNP can also cause additional latency in the bootstrap and discovery phase because MSNP is based on MP2P topology in which the involved data for performing trust control is distributed (e.g., stored in each MSNP participants' backend Cloud storage) and require the mobile application to retrieve them at runtime via the unstable mobile Internet. Hence, reducing latency for discovery phase becomes a priority challenge which needs to be resolved in MSNP.

Our work aims to fill the gaps of existing PBMSN solutions to enable MSNP. The goals of our work are to:

- Support loose coupling to enable heterogeneous interoperability for MSNP.
- Support trustworthy control for MSNP in the bootstrap and discovery phase.

- Reduce latency for MSNP in the bootstrap and discovery phase.
- Provide a strategy to enable system reconfiguring of its resources and task for MSNP activities at runtime to adapt to dynamic changes.

In the next section, we describe our proposed service-oriented MSNP architecture to achieve the above goals.

3.5 Design of Service-Oriented Mobile Social Network in Proximity

3.5.1 System Overview

MSNP represents an environment in which mobile users utilise their mobile devices to perform social activities with each other in proximal distance. The fundamental aim of MSNP is to enable communication in a fairly close range so that the participants can potentially meet each other. Figure 3.4 illustrates an MSNP environment. In order to improve the interoperability, Web service has been utilised as the common communication interface.

In an MSNP environment, each mobile device is a mobile Web service consumer and also a provider (Srirama et al.; 2006). When two peers join the same wireless network, they utilise standard communication technologies such as DPWS or Zeroconf to exchange their Service Description Metadata (SDM). For peers who do not have Mobile IPv6,[18] we expect each to have its own back-end cloud storage to synchronise its IP address as a small text file in its cloud storage (or alternatively utilising public DNS servers if available). The URL of the text file is described in a peer's SDM. Hence, when the peer (e.g., Figure 3.4, P2 or P4) moves out from the current network, the other peers (e.g., Figure 3.4, P1 and P3) in their previous network can still interact with P2 or P4 via mobile Internet.

[18]See `http://tools.ietf.org/html/rfc6275`

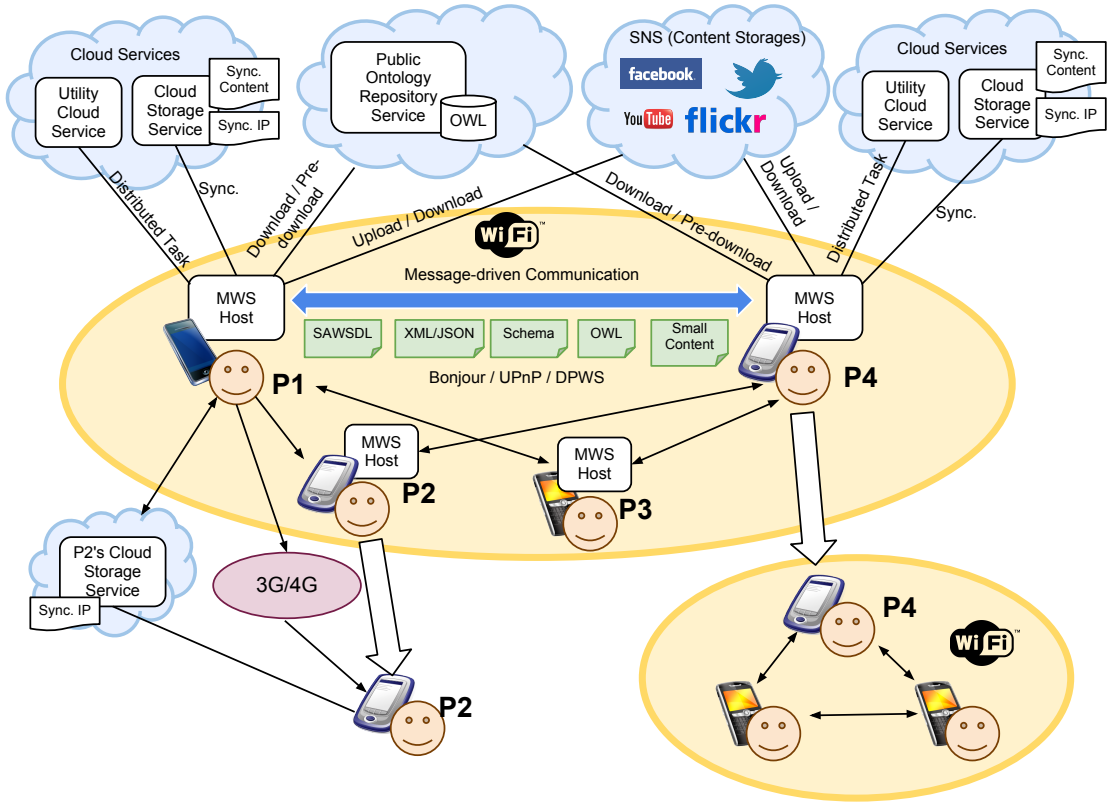


Figure 3.4: Service-Oriented MSNP Architecture

Since P1 and P3 have previously exchanged their SDM with P2 and P4, they have cached the SDM of P2 and P4 in either their local memory or synchronised it to their cloud storages. When P1 and P3 receive requests from other peers in the same network that are performing service discovery, P1 and P3 can also provide P2 and P4's SDM to these requesting peers. Instead of having the SDM directly sent to the peers by P1 and P3, P1 and P3 can synchronise the cached SDM to their cloud storages, and simply provide the URL link to the requesting peers.

A similar concept can be applied to content sharing and mashup, say for example, P1 intends to mashup the content provided by P2 and P3. When P1 invokes P2 and P3 for the content, P2 and P3 will simply reply with the corresponding metadata documents, which contain the description about where the content can be retrieved from in the Internet. For example, P2 has uploaded the content to a SNS as public accessible content. Hence, P2's response metadata will contain the URL link of the uploaded content.

Taking into account that mobile devices usually have limited processing power, it is reasonable for an MSNP peer to delegate some of its processes to its backend Cloud Utility Service (CloudUtil). In Figure 3.4, for example, P1 utilises its backend CloudUtil for semantic service discovery. Further, CloudUtil can also be used to directly access the content uploaded by other MSNP peers in Social Network Services (SNS) to discover useful content for P1's mashup (if the content has been described in Rich Site Summary [RSS][19] feed format).

A content provider in MSNP can also actively push recommendations to other participants based on the participants' service preferences. Due to privacy concerns, MSNP peers may prefer not to share their private information. However, when a list of available services (described semantically) is provided to the participants, the participants can simply reply which service type they are interested in.

3.5.2 Basic Capabilities of Mobile Social Network in Proximity Participant

Depending on the user's preference, the user's device can only support minimal MSNP capabilities if the user intends to save resource usage. Conversely, an active and more advanced MSNP user may prefer to enable full MSNP capability, so that his/her device will support all the capabilities described in Table 3.2.

We classify the MSNP capabilities into two types (see Table 3.2): Content Consumer (CC) and Content Provider (CP). Moreover, we also rank the capabilities into three levels. Progressing from the lowest level to the highest level represents how capable the application is in order to assist the user in MSNP environments. The minimal capability is the level 1 content consumer (Lv1CC), which can discover MSNP participants near the user, and let the user manually interact with other MSNP participants via user application. This minimal MSNP application allows its user to perform minor social activities such as requesting an MSNP participant

[19]See `http://tools.ietf.org/id/draft-nottingham-rss-media-type-00.txt`

Capability Level	**Content Consumer (CC)**	**Content Provider (CP)**
Lv1	• Discover proximal MSNP participants via a common protocol • Interact with proximal MSNP participants via a user interface	• Join and publish/advertise in MSNP via a common protocol • Provide a basic human readable interface
Lv2	• Filter content/service providers automatically based on user preferred service types • Runtime retrieve and parse MSNP participants' description metadata files	• Support standard description metadata • Support semantics
Lv3	• Record user preference and corresponding context • Discover and filter content/service providers automatically based on user preference • Host service socket to enable autonomous interaction • Reconfigure processes to adapt to situations automatically with regards to hardware resources and environmental factors	• Enable subscription (e.g., WS-Eventing) • Advertise automatically based on receivers' preference (requires all capabilities including Lv1CC, Lv2CC, Lv3CC, Lv1CP and Lv2CP)

Table 3.2: Basic MSNP Capabilities

to exchange a digital business card or exchanging profile files towards discovering someone who has common interests. Additionally, the Lv1CC can also access some services provided by MSNP participants such as services that provide URL links to some online public accessible content. However, every action performed in a Lv1CC application requires manual input, which is inconvenient when the environment consists of a large number of providers. Lv1CC does not use any additional mechanism, and can be simply realised by existing WLAN-based P2P discovery technologies such as UPnP and Bonjour.

A more advanced CC application (Lv2CC) in MSNP is capable of providing a partial autonomous discovery mechanism based on a user's input. The autonomous discovery requires both CC and CP to support semantic computing and utilising standard metadata (e.g., OWL, RDF, SAWSDL). For example, suppose an MSNP participant—Alex intends to find out who has common interests with her. She uses her MSNP-enabled device to search for such a person. In order to discover such a person, Alex's MSNP application will first communicate with each MSNP participant using a common protocol (either by applying multicast to the standard formatted semantic request message or utilising client-server style invocation to retrieve and process metadata files) to discover the participants who have common interests with her. Second, Alex's MSNP application will then retrieve the matched participants' public profiles and display them to Alex. Lv2CC can highly reduce the amount of discovery results and effectively filter unwanted results shown on the user's application.

Finally, a full capacity CC application (Lv3CC) in MSNP provides numerous mechanisms to enable autonomous discovery, reducing latency of discovery, and supports autonomous process adaptation depending on context such as resource usage (e.g., device process power, throughput, battery power) and environmental factors (e.g., the number of participants in current network, network speed). In order to support advanced autonomous communication, a Lv3CC application is

required to host a service socket component (e.g., HTTP Web server) to enable automatic machine-to-machine interaction with other MSNP participative devices.

The advanced CP (Lv3CP) application is capable of automatically discovering targets that are of interest to its user. For instance, based on the previous scenario about Alex, who is searching for users who have common interests, if Alex's MSNP application supports all the capabilities described in Table 3.2, in which her device is always recording her activity preferences and corresponding context information, the device is capable of performing autonomous action without Alex's manual input. Her MSNP application can automatically find the matched MSNP participants and then advertise Alex's business card or public profile to them. This feature is most useful for users who intend to find more friends using MSNP and bring more visitors to the users' SNS sites or related online content (e.g., YouTube videos). From content consumer's point of view, automatically receiving recommendations or advertisements based on the consumer's preference can highly reduce the latency caused by the discovery process.

3.5.3 Fundamental Elements of Service-Oriented Mobile Social Network in Proximity

In this section, we describe a number of elements reflecting the design requirements of a service-oriented MSNP.

3.5.3.1 Content Sharing and Management

Distinguishing from a centralised system (e.g., Facebook) in which all the participants' contents are stored in a common repository, and different from a pure MANET-based social network in which each participant provides his/her content directly from his/her device, our design is based on decentralised online social network solutions that utilise distributed cloud storages to store large size content such as images, videos and audio. Each participant's device communicates with proximal

participants using metadata-based messages. If a participant's device has been requested for a particular content, it will respond with metadata, which describes the URL of the corresponding content in the participant's SNS space or cloud storage. Afterwards, the requester can retrieve the content from the Cloud storage without requiring heavy data traffic between the requester and the provider.

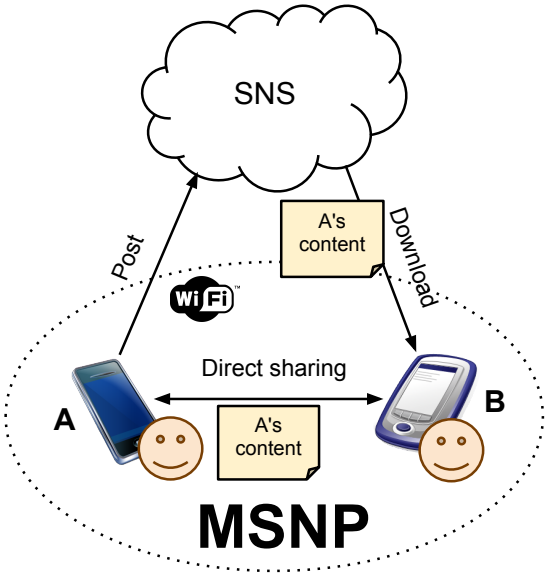


Figure 3.5: Content sharing

For example, as Figure 3.5 shows, Peer B intends to retrieve the content provided by Peer A, there are two possible options to complete the task: In the first option, Peer A may allow content to be retrieved by invoking Peer A's MWS directly, or in the second option, Peer A may have posted the content in its SNS space. From Peer A's SDM, which was obtained from a service discovery process, Peer B can find the content provided by Peer A from Peer A's SNS space and download it via the Internet without requesting Peer A directly.

In addition to the above activity, other social network activities such as 'feedback' or 'comment' in regards to a content, can be sent to the original content provider through the Web service request/response process, and the content provider can add the feedback to the content locally and also synchronise the updated content to the cloud storage. A user can create multiple directories for different authorisation levels. Each directory contains multiple content data. Ideally, a simple indexing

metadata should be available for the user's social contacts/subscribers to identify and be notified of any updates.

3.5.3.2 Network Topology

A fundamental question of MSNP is how participants can use heterogeneous devices to discover each other in the real world physical proximity. This question can be resolved by applying platform independent MP2P service discovery technologies such as Bonjour/Zeroconf, UPnP or DPWS (see Chapter 2, Section 2.3.3). In our work, MSNP is established using Bonjour technology. Implementation of Bonjour is based on Wi-Fi or Bluetooth networks. The recently approved Wi-Fi Direct[20] technology will enable a Bonjour environment without static Wi-Fi hotspots in the near future. Although Bonjour has been chosen to implement our prototype, we argue that our proposed work is compatible with other technologies such as UPnP and DPWS because our focus is not on the fundamental network communication layer, but on the high level architecture and the MWS interaction design.

3.5.3.3 Unique Identification

In Bonjour, a participating device can dynamically obtain an IP address, and is capable of selecting its own service name. If a duplicated name has been published in Bonjour, the later published device will automatically be assigned a number after its selected name (e.g., '`MobileWebService (2)`'). However, such a mechanism is not satisfactory and proper for participating devices in MSNP. The selectable names of devices in Bonjour can be changed easily and there is no corresponding solution to let participants to identify the trustworthiness of one another. Based on our study (see Section 3.3), OpenID appears to be a more feasible option. We expect each participating device in MSNP will use its user's OpenID as its unique identifier in Bonjour.

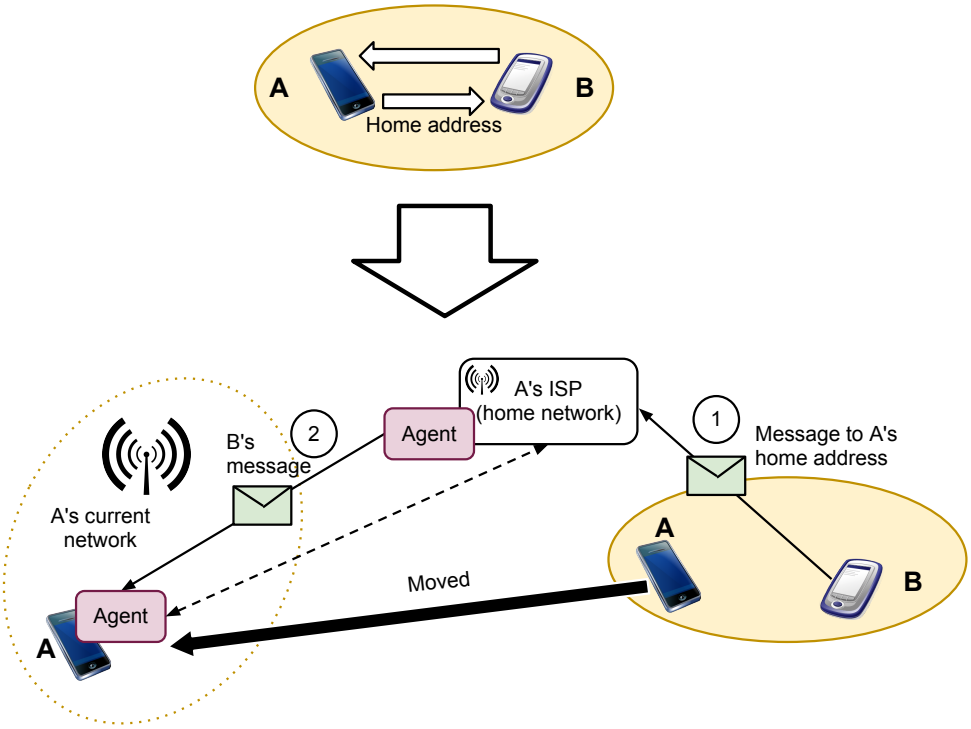
[20]See `http://www.wi-fi.org/Wi-Fi_Direct.php`

3.5.3.4 Internet Protocol Mobility

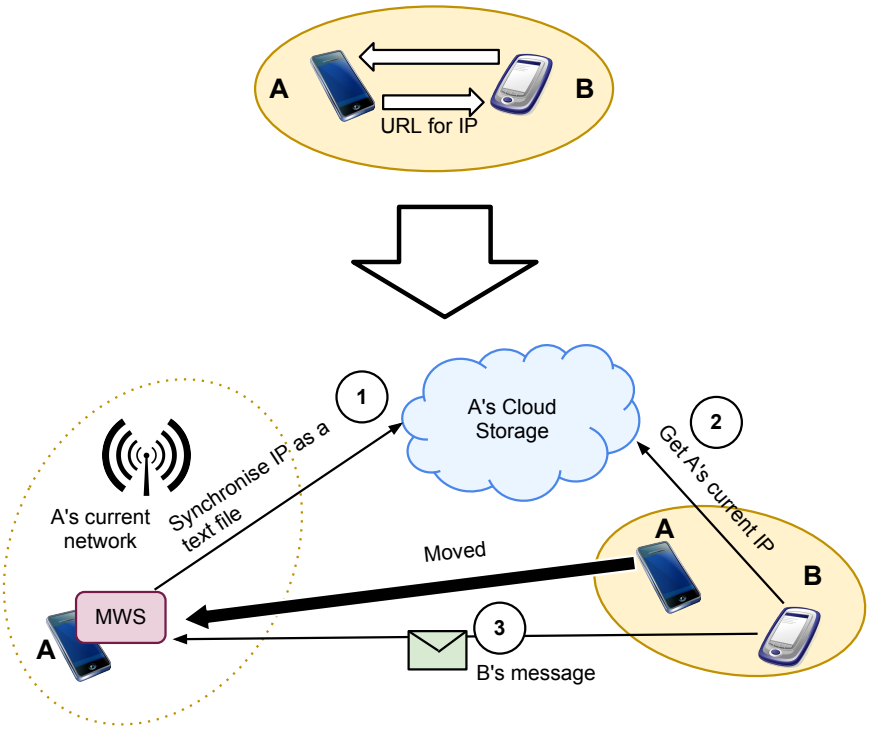
Mobile devices usually do not have static IP addresses. Although in Zeroconf (which is used in our work) each device has been dynamically assigned an IP address to participate in the network, the IP address is only valid while the device remains in the network. If the device moves out of the current network, its communication with the other participants' devices will be interrupted, and further reconnection is impossible unless a new IP address can be notified to the other participants. Alternatively, if each mobile device has a Mobile IPv6 address at its disposal, the dynamic IP is no longer an issue.

Figure 3.6(a) illustrates a scenario in which two MSNP peers—Peer A and Peer B—are in physical proximity with a Wi-Fi connection, and they have established a connection by exchanging SDM, which contains the IP address of each other. In this scenario, Peer A's IP address is a static mobile IPv6 address (also known as a 'home address'). Afterwards, Peer A moves to another location with 3G/4G connection. Peer B intends to retrieve content from Peer A. Since Peer A's Internet Service Provider (ISP) has provided Peer A with a mobile IPv6 solution, Peer B's request message can still reach Peer A because in the mobile IPv6 environment, data intended to reach Peer A's IP is actually received by the broker server of Peer A's ISP (Step 1). The broker server will forward the data to Peer A via software agents (Agent in Figure 3.6(a)) that maintain the communication channel between Peer A and the broker server of Peer A's ISP (Step 2), which is also known as communication within a 'home network'.

Ideally, mobile IPv6 can resolve the IP mobility issue. However at the time of writing this thesis, most mobile devices in the market do not have static Mobile IPv6 addresses. In order to overcome this limitation, we propose an approach utilising cloud storage services to dynamically synchronise the device's IP address (see Figure 3.6(b)). This approach serves a similar solution as a mobile IPv6 solution. Each participant in an MSNP has its own cloud storage (e.g., Dropbox) by its own choice. Figure 3.6(b) illustrates a scenario in which Peer A and Peer B were in the same



(a) Utilising ISP provided Mobile IPv6



(b) Utilising Cloud storages

Figure 3.6: IP mobility support

network and exchanged their SDM with each other. In order to let Peer B maintain communication with Peer A when Peer A moves to another network, they will perform the following steps:

1. Peer A obtains a new dynamic mobile IP address; it synchronises its new IP address as a text file to its cloud storage

2. Peer B retrieves the text file that describes Peer A's IP address from Peer A's cloud storage

3. Peer B identifies Peer A's current IP address from the text file, and then sends its request to Peer A

3.5.3.5 Adaptive Discovery and Trust in Mobile Social Network in Proximity

Utilising distributed hash table (DHT) is a common approach to describe published services in a P2P network. However, in order to minimise network transactions, information disseminated to each peer is usually limited.

A generic discovery task in MSNP is to search for specific service/operation types that can return the user required content. In order to discover such content, one needs to first identify whether the content provider's Web services can respond to the required content. Such information is not clearly described in DHT, and the requester needs to retrieve the provider's WSDL (and related documents such as OWL-DL[21] and XML schema) in order to identify whether the provider's service can respond with the required content or not.

In SNS such as Facebook, a user may enter a number of keywords to search for a particular *content provider*. However, the efficiency of the keyword-based search is restrictive (Srirama et al.; 2008). When the environment grows, a keyword-based search can result in a large number of matched services. Manual browsing to select a feasible service from such a list is not ideal for mobile application users. In order

[21]See `http://www.w3.org/TR/owl-guide/`

to overcome such a problem, we applied semantic Web technology to improve the discovery process in MSNP to autonomously discover and filter search results.

The communication between participants involves with trust issues. For example, a content provider's content may not be consistent with its description metadata, or the service provided by a participant may exhibit malicious behaviour. Since a reliable central management party for supporting trustworthiness is not available in MSNP, the environment requires a decentralised trust solution letting each MSNP participant to manage the access control by itself.

A number of works (Kourtellis et al.; 2010; Qureshi et al.; 2010; Wang and Vassileva; 2007; Golbeck; 2005; Yang et al.; 2011) were proposed to support distributed trust control based on reputation-based schemes for MP2P environments. As mentioned previously, MSNP lets users interact with new people who do not know each other beforehand. In such an environment, trust control also faces latency problems derived from retrieving and processing required data for trust control from various distributed resources. For example, in order for an MSNP participant to identify trustworthy content provided by another MSNP participant's MWS (i.e. the content is matched to its description or not), the content requester may perform the reputation-based scheme that involves retrieving reputation ranking data from a number of sources at runtime. Such a task can cause latency problems due to the unstable mobile network bandwidth and the data provider's throughput in the mobile network.

Another task that can cause runtime latency is the semantic service discovery process. Since the process can sometimes be time consuming on resource constraint mobile devices (Steller and Krishnaswamy; 2008), there is a need to reduce the latency caused by performing trust-based discovery and semantic service matchmaking. In our work, we have applied context-aware user preference-based mobile Web service discovery scheme to enable proactive autonomous discovery in MSNP. The scheme reduces latency by using the user's context information to improve the

performance of the overall discovery process. The details of this scheme is described in Chapter 4 of the thesis.

3.5.3.6 Resource-Awareness

Considering the resource limitations of mobile devices and the dynamic nature of MP2P environments, communication performance becomes crucial to both content provider and content consumer. In order to enhance the overall performance of MSNP communication, some tasks such as semantic service and content matchmaking processes may be distributed to remote cloud services (e.g., GAE[22], Amazon EC2).[23] However, distributing tasks to cloud is not always ideal, because of the dynamic nature of mobile networks. Utilising a cloud service can also incur additional costs such as network latency (mobile Internet bandwidth is unstable) and the price of using the service.

In some situations, retaining communications within local wireless networks is more efficient when both performance and cost are considered, especially when there are only a few MSNP peers involved. Conversely, when it involves many MSNP peers, it may be more efficient to distribute more tasks to the cloud services. Hence, there is a need to design a scheme that is capable of dynamically changing its approach at runtime to adapt to different situations while the MSNP peers are performing MP2P social network activities. In our project, we have designed a Web service standard and workflow-based scheme to support runtime adaptive task reconfiguration. The scheme is capable of dynamically selecting feasible resources to perform various MSNP tasks at runtime without relaunching the application. In order to enable the dynamic resource allocation for the adaptive task reconfiguration, we have developed a framework called **A**daptive **M**ediation Framework for Mobile **S**ocial **N**etwork in **P**roximity (AMSNP). The details of the workflow-based scheme and AMSNP are described in Chapter 5 of the thesis.

[22]See `https://developers.google.com/appengine/`
[23]See `http://aws.amazon.com/ec2/`

3.6 Summary

This chapter describes the background of MSNP and the proposed service-oriented MSNP architecture. The chapter started with a review of the origins of MSNP (i.e., MSN and LBSN). We also highlighted the differences between MSNP and its predecessors.

We then discussed the requirements of enabling service-oriented MSN, and reviewed and categorised the related works on MSNP. A detailed comparison of these works based on a number of criteria was presented to identify the gaps in existing works when they were applied in service-oriented MSNP.

Finally, we presented our proposed service-oriented MSNP architecture. The architecture attempts to avoid the single point of failure issue and to resolve the dynamic mobile IP issue. Moreover, we described the individual components used by the service-oriented MSNP applications. These components are purported to be capable of reducing service discovery latency and supporting resource-aware adaptive reconfiguration in the service discovery process. The details of these components for service-oriented MSNP applications will be described in chapters 4 and 5 of the thesis.

Chapter 4

Discovery and Trust in Mobile Social Network in Proximity

4.1 Introduction

Service-oriented Mobile Social Network in Proximity (MSNP) lets participants establish new social interactions with strangers in public proximity using heterogeneous platforms and devices. Such characteristic faces challenges in discovery latency and trustworthiness.

In a classic service-oriented environment, service discovery is based on the request-response model in which a service requester first retrieves service providers' service description metadata (SDM) files via a common data transmission protocol and then processes the SDM in order to identify whether the service providers' services can fulfil the requester's need or not. The SDM files can combine with semantic annotations and associate with corresponding semantic description metadata (such as OWL) to enable a more scalable interoperability.

Request-response model is the classic architecture to enable semantic service-oriented architecture. However, it may cause high latency in MSNP because when the environment consists of a large number of participants, a content requester who searches for a particular service provided by other MSNP participants will need to

retrieve and process a large number of SDM files together with associated semantic metadata files. Performing such a task on a resource constraint mobile device can be time consuming. In addition to service discovery, once the requester discovers a list of service providers who can fulfil his/her needs, for security and privacy reason, he/she needs to identify whether the service providers are trustable or not before he/she interacts with the service providers.

Performing trust management control in MSNP also faces challenge in latency because a stable third party entity to determine the trustworthiness is not available in MSNP. A requester who intends to determine the trustworthiness of a stranger's service needs to refer to other participants' past experience with the stranger's services. Intuitively, mobile participants may have synchronised their trust-related data to their backend cloud storages so that these data can be retrieved indirectly and will not be affected by their movement. However, for the requester who needs to collect and process those trust-related data, his/her overall discovery performance will be affected and will result in a high latency.

This chapter analyses the service discovery models of MSNP and presents corresponding solutions to improve the service discovery performance of MSNP. We firstly present and analyse the basic service discovery models of service-oriented MSNP in Section 4.2. In Section 4.3, we apply a context-aware user preference prediction scheme to enhance the semantic service discovery process. In Section 4.4, we address the trustworthiness issue in MSNP and propose a scheme to reduce the latency of the trustworthy service discovery for MSNP. Section 4.5 summarises the chapter.

4.2 Service Discovery in Service-Oriented Mobile Social Network in Proximity

Before we proceed with our discussion, we define and reiterate the terminologies in a service-oriented MSNP:

- A *device* represents a mobile device such as a smart phone, a handheld media player or a small tablet computer. A device can be operated by any operating system and is capable of participating in mobile P2P network using software applications.
- An *agent* represents an MWS-enabled software agent. The term—agent is derived from the software agent described in W3C Web Service Architecture document (World Wide Web Consortium; 2004), in which an agent performs Web service activities for its human user. In MSNP, an agent can perform functions for both MWS client and server.
- A *user* is a human user who holds a mobile device that is embedded with mobile Web service (MWS)-enabled software agent.
- An *MSNP participant* represents an entity which participates in an MSNP environment. Each MSNP participant consists of: a *device*, an *agent*, a *user*.

In MSNP, each agent would have pre-downloaded a fair number of public common ontologies that have been published on cloud resources (e.g., Swoogle[1], or FUSION[2]). A public common ontology describes numerous common service types and data types semantically. Each SAWSDL-compliant agent describes its services using semantic annotations that map to the corresponding ontology types. Benefiting from the public common ontologies and semantic annotation, an MSNP service requester's agent can identify whether a service matches to the functionality it needs from the service provider's WSDL and related documents (e.g., XML Schema). In the following subsections, we discuss three basic service discovery models in MSNP.

[1] `http://swoogle.umbc.edu/`
[2] `http://www.seerc.org/fusion/semanticregistry/`

4.2.1 Pull-based Service Discovery

Pull-based service discovery in service-oriented MSNP represents the most basic service discovery mechanism that is supported by the existing mobile P2P protocols (e.g., UPnP, Bonjour, DPWS, etc.) without making a significant assumption, such as expecting the requester *agent* to provide MWS to let other *agents* advertise service description metadata (SDM) to it.

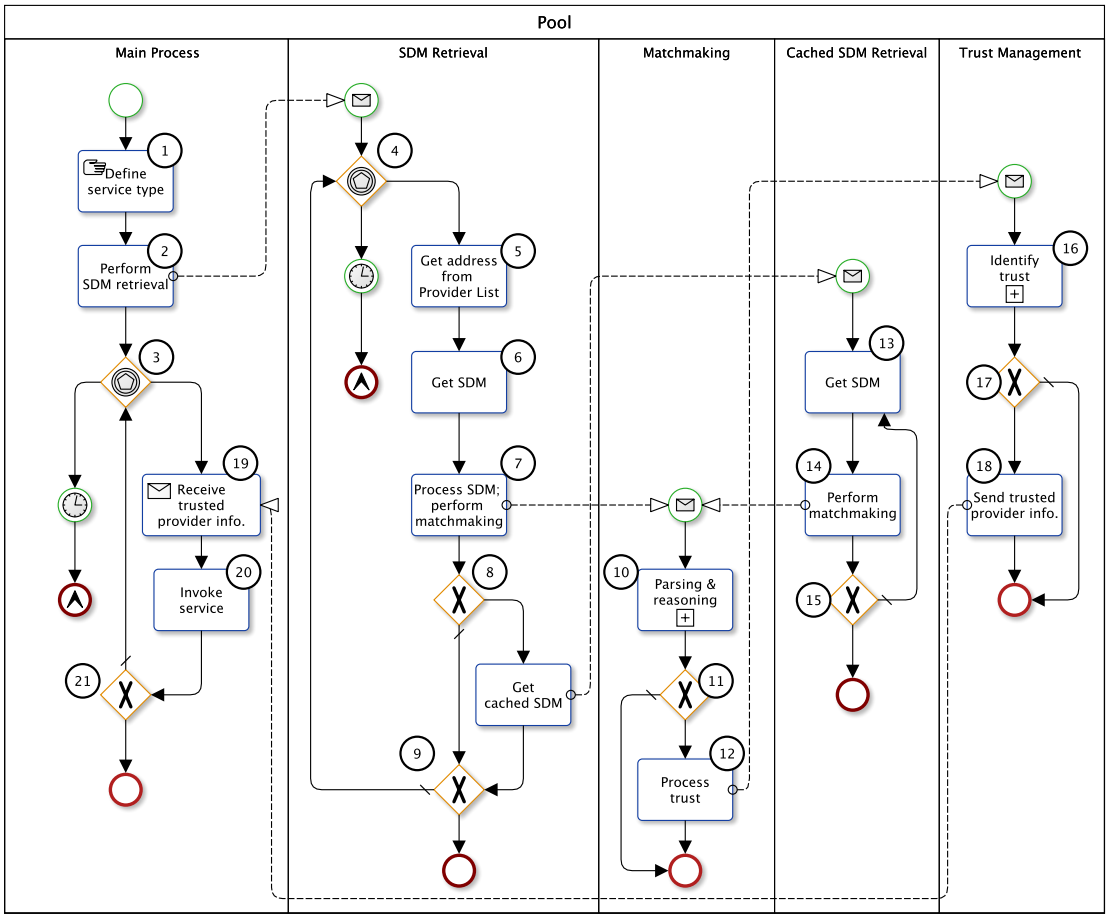


Figure 4.1: Pull-based service discovery in MSNP

Figure 4.1 illustrates the process flow of the pull-based service discovery model described in Business Process Modelling Notation (BPMN).[3] A brief description of the BPMN symbols used in the figure can be found in Appendix A.

The discovery process consists of five subprocesses:

[3] http://www.bpmn.org/

- **Main Process**. When a user joins an MSNP environment, he/she manually defines his/her need (task 1) and then requests his/her *agent* (denoted by $agent_{rqt}$ to discovery the corresponding service provided by other MSNP participants' *agents* (task 2). The $agent_{rqt}$ launches the SDM retrieval subprocess and keeps the main process thread on stand by waiting for the result from the Trust Management subprocess (task 19). When a trusted provider information is passed to the main process, $agent_{rqt}$ will invoke the corresponding service from the provider *agent* to retrieve the result (task 20).

- **SDM Retrieval**. The SDM Retrieval subprocess is set to a finite timestamp (mark 4). When time is up, this subprocess will be terminated. The main activity of this subprocess is to retrieve SDM from each MSNP participant's *agent* in the requester's current environment (task 5, 6). After retrieving and processing the SDM, $agent_{rqt}$ may find out that the provider *agent* is also providing a service which returns a list of other *agents*' SDMs which were fetched when the provider *agent* performed discovery previously when it joined the current environment. If such a *cached SDM* service is available, $agent_{rqt}$ will launch the **Cached SDM Retrieval** subprocess.

- **Cached SDM Retrieval** subprocess retrieves one or more cached SDMs from the provider (task 13). The cached SDMs can either be retrieved from its provider *agent* directly or be retrieved from the provider's cloud storage depending on the provider's preference. The retrieved SDM is also passed to the Matchmaking subprocess (task 14).

- **Matchmaking**. The retrieved SDM and its associated documents will be processed by the Matchmaking subprocess. The Matchmaking subprocess uses semantic reasoning algorithm and XML document parsing technologies to identify whether the provider can provide a corresponding service to fulfil the request or not (task 10). If the provider can provide the corresponding service, $agent_{rqt}$ will perform **Trust Management** (task 12).

- **Trust Management** subprocess identifies whether the provider's service is trustworthy or not (task 16). If the provider's service is trustworthy, $agent_{rqt}$ will perform the service invocation to retrieve result from the provider (task 18).

A drawback of this simple model is the latency issue. Because SDMs are described in XML format, resource-constraint mobile devices are usually unable to process a large number of XML documents effectively.

4.2.2 Push-based Service Discovery

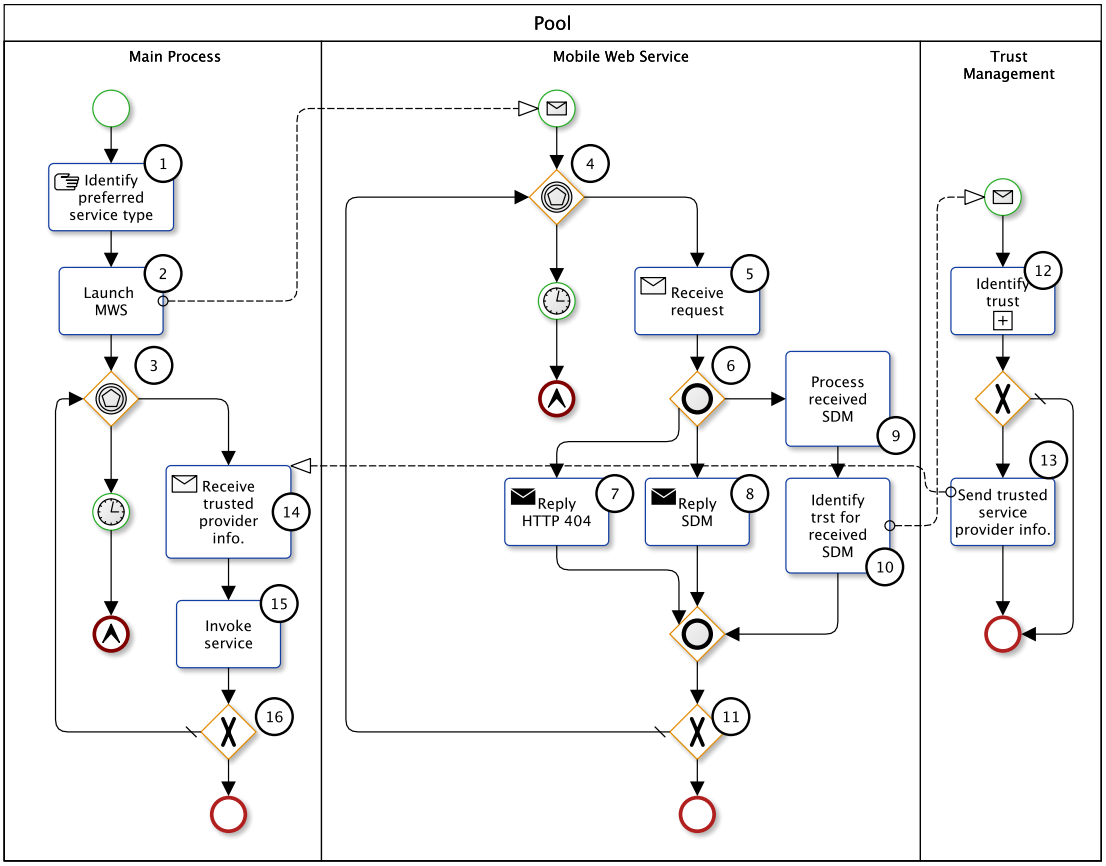


Figure 4.2: Push-based service discovery in MSNP

Push-based service discovery approach involves a requester *agent* ($agent_{rqt}$) utilising passive mechanism to receive SDMs advertised by the other active MSNP participants' *agents*. Figure 4.2 illustrates the process flow of push-based service

discovery approach. The behaviour of the main components in this approach is described below:

- **Main Process** in push-based approach differentiates from the pull-based approach in that $agent_{rqt}$ does not actively invoke the other participants' *agents* to retrieve their SDM. Instead, $agent_{rqt}$ launches an MWS provider (task 2) to passively receive SDM advertised by the other *agents*.

- **Mobile Web Service** subprocess permits the requester to be passive. In this subprocess, $agent_{rqt}$ provides MWS to let other participants' *agent* retrieve $agent_{rqt}$'s SDM (task 8). Based on $agent_{rqt}$'s SDM, other *agents* directly push their SDM to $agent_{rqt}$. When $agent_{rqt}$ receives an SDM, it performs the matchmaking task to identify whether the SDM's provider can provide the required service type or not (task 9). If the SDM's provider can provide the required service type, $agent_{rqt}$ will perform the Trust Management subprocess to identify the provider's trustworthiness (task 10).

- **Trust Management** subprocess is the same as the Trust Management process described in the previous pull-based model.

4.2.3 User Preference Associated Push-based Service Discovery

In addition to the two basic service discovery approaches described in the previous sections, we propose a new service discovery approach for MSNP—the user preference associated push-based service discovery (*PrefPush*). The relative works of *PrefPush* have been previously published in (Chang et al.; 2011; Chang, Srirama, Krishnaswamy and Ling; 2013; Chang, Srirama and Ling; 2013). *PrefPush* in MSNP relies on participants' *agents* actively advertising their SDM to one another. A requester participant's agent—$agent_{rqt}$ will perform the following three subprocesses to enable *PrefPush*. Figure 4.3 describes the process flow of the *PrefPush*-based service discovery model in BPM notation.

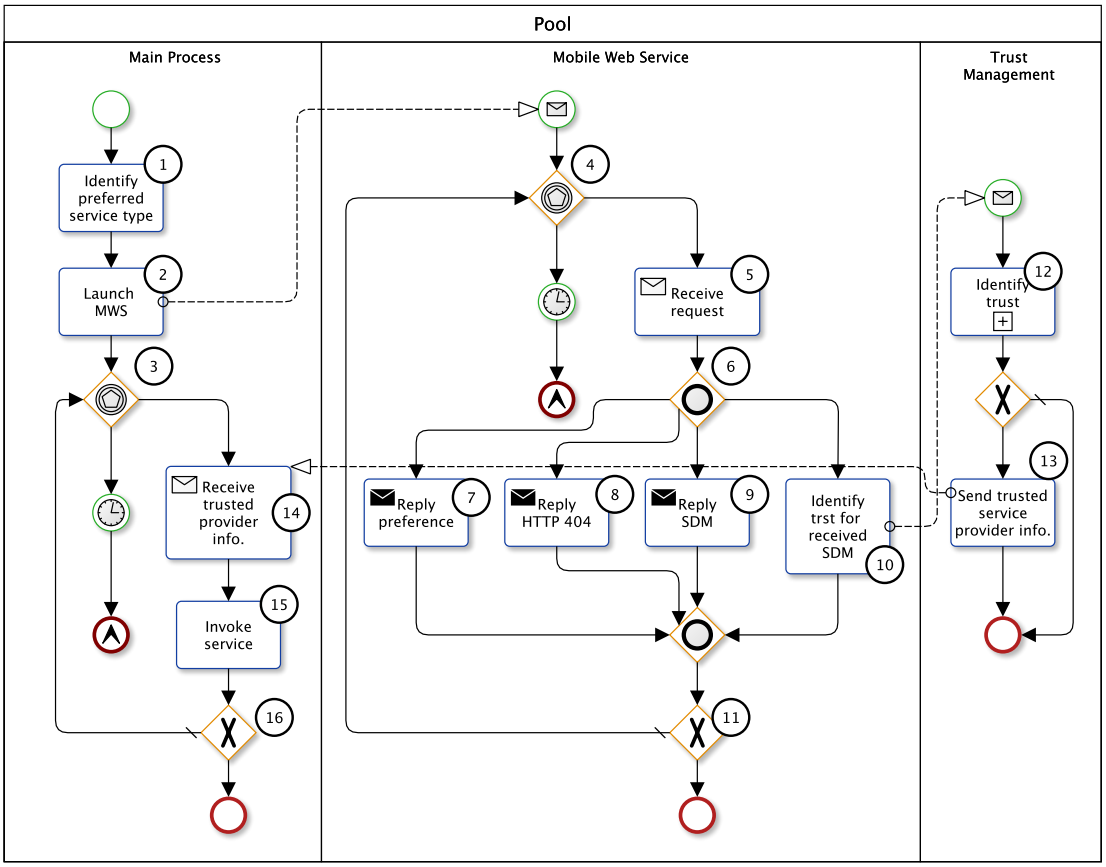


Figure 4.3: *PrefPush*-based service discovery in MSNP

- **Main Process** is slightly different from the pull-based model. In the *PrefPush* approach, when a user joins the environment, $agent_{rqt}$ can autonomously identify its user's preferred service in the current environment based on the user's past request records and the current environmental context information (task 1). The details of the approach to enable the autonomous identification of the user's preferred service will be described in the next section. Alternatively, user can manually define his/her preferred service type prior or on demand at runtime. Once the preferred service type is identified, $agent_{rqt}$ launches its Mobile Web Service (MWS) server-side mechanism to let other participant's *agents* actively interact with it (task 2). Afterwards, $agent_{rqt}$ puts the main process on stand by waiting for the result from the Trust Management subprocess (task 14). When a trusted provider information is returned from the Trust

Management subprocess, $agent_{rqt}$ will invoke the trusted provider's service to retrieve the result.

- **Mobile Web Service** subprocess enables the requester to be passive. In this subprocess, $agent_{rqt}$ provides MWS to let other participants' *agents* retrieve $agent_{rqt}$'s SDM (task 9). Based on the $agent_{rqt}$'s SDM, other participants' *agents* can also request $agent_{rqt}$ for its user preferred service type (mark 7). The other participants' *agents* who retrieved $agent_{rqt}$'s user preferred service type information, can determine whether their services can fulfil the need or not. If they can, they can post their SDM to $agent_{rqt}$ as advertisement. Note that, at this stage, we do not expect the other participants' *agent* to directly post the content corresponding to the $agent_{rqt}$'s user preferred service type, because $agent_{rqt}$ does not know whether the provider is trustworthy or not. Hence, $agent_{rqt}$ expects a SDM advertisement rather then the corresponding content. Once $agent_{rqt}$ receives a SDM, it performs the Trust Management subprocess to identify the trustworthiness of the provider (task 10).

- **Trust Management** subprocess is the same as the Trust Management process described in the previous pull-based model.

The major difference between *PrefPush* and the previous two approaches (*Pull*, *Push*) is that in the *PrefPush*-based model, $agent_{rqt}$ does not need to perform any SDM retrieval process or perform semantic matchmaking process. Since the matchmaking process is done by other participants' *agents*, the overall discovery makespan of *PrefPush* can be much lower than the other two approaches.

However, the drawback of *PrefPush* is that it assumes other participants' *agents* remain active and will advertise their SDM to the others. Since such an assumption cannot be guaranteed, it is more feasible to perform both pull-based and *PrefPush*-based model at the same time for the service discovery.

4.2.4 Hybrid-based Service Discovery

Hybrid-based service discovery model in MSNP combines both pull and *PrefPush*-based service discovery models. Figure 4.4 illustrates a hybrid-based service discovery model. The service discovery process consists of three main parallel tasks: pull-based service discovery, push-based service discovery, and the trust management.

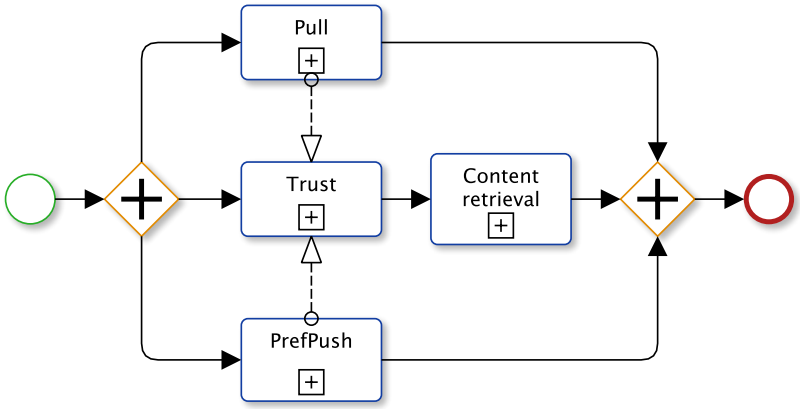


Figure 4.4: Hybrid-based service discovery in MSNP

When an MSNP user enters an MSNP-enabled environment, he can either manually launch the application to search for his/her preferred service provider, or his/her agent can automatically triggers service discovery mechanism at background based on the user's preference computed from the scheme described in Section 4.3. The agent will use a client-side MWS mechanism to search service providers and also launch an MWS server to allow other participants' agents to actively advertise their SDM to it.

The pull-based service discovery task and *PrefPush*-based service discovery task will pass SDMs retrieved from other service provider agents to the trust management task. If the provider is trustworthy, the agent will interact with the provider to retrieve the result/content as described in the previous two models.

4.3 Context-Aware Proactive Service Discovery in Mobile Social Network in Proximity

Push-based service discovery in MSNP can be greatly improved by applying a proactive autonomous discovery mechanism. As Figure 4.2 shows, in the first task in the Main Process, if it relies on user manually entering the preferred service type, after the service type is entered, the user has to wait until the result return. However, if the agent can predict the user preferred service type, it can support the *PrefPush*-based service discovery approach in which the agent can autonomously start the discovery process when the user enters the MSNP environment. By doing so, when the user starts using the application, the agent has already discovered and identified a list of trusted services provided by the others. Moreover, some results may have already been pre-fetched by the agent if the user has granted the agent to do so.

In this section, we present our proposed context-aware proactive service discovery scheme for MSNP. The proposed scheme enables the agent to perform SDM prefetching and content prefetching to reduce service discovery latency. The relative works of this scheme have been previously published (Chang et al.; 2011; Chang, Srirama, Krishnaswamy and Ling; 2013; Chang, Srirama and Ling; 2013).

Before the details of the proposed scheme is discussed, we provide the background of the proposed scheme.

4.3.1 Background of Proactive Service Discovery

The fundamental part of the user preference associated push-based service discovery approach proposed in this thesis is based on the autonomous data prefetching mechanism. In general, the prefetching mechanism consists of the elements described in the following sections.

4.3.1.1 Prediction

The prediction mechanism aims to predict a user's request based on various factors. Factors in a prefetching approach for Web browsing (Jiang and Kleinrock; 1998; Tuah et al.; 2003; Bürklen et al.; 2006) include user's browsing history, interests, navigation behaviour, and the popularity of the available contents/resources. By analysing these factors and comparing them with presently available contents, the probability of user's interest in a content can be computed.

In mobile and pervasive computing environments, more factors need to be considered (Drew and Liang; 2004; Choi et al.; 2005; Feng et al.; 2006; Drakatos et al.; 2009; Jin et al.; 2007; Boldrini et al.; 2010). These are user's current location, moving direction, hardware resources, network bandwidth, and many others. Based on these factors, corresponding policies or rules can be designed and applied to a decision-making scheme to predict and anticipate a mobile user's future request more accurately.

In Web-based systems, Jiang and Kleinrock (1998) have introduced a prediction module to track user's access history continuously. Based on the historical records, the system can compute the probability of user's browsing actions, and determine what content needs to be prefetched. An extended approach proposed by Tuah et al. (2003) applied compound access graph to perform prediction based on the most recent browsing histories and the relationships between web pages. A mobile environment based prefetching scheme proposed by Bürklen et al. (2006) has considered the location factor. The prediction result was calculated based on the user's searching histories in specific locations. These techniques were proposed for Web systems and their prediction decision modules only considered static factors. They did not consider dynamic factors such as the user's preference in different situations and events.

A number of researchers (Drew and Liang; 2004; Choi et al.; 2005; Bürklen et al.; 2006; Drakatos et al.; 2009) have proposed location-based and movement-based prediction scheme for cache prefetching. These works predict the probability of user's

future query by analysing the user's present and future location (based on his/her movement prediction), the corresponding query history records, and the predefined user preference profiles. However, in reality, a user's preference can dynamically be changed at runtime due to more other factors. Moreover, the pre-defined static user preference profiles and rules are difficult to fulfil unanticipated situations (Chen; 2005), unless the user is willing to adequately define many different preferences manually for all possible situations. In most cases, a user is unable to define his or her probability for events accurately (Heckerman; 1996). Therefore, a proper adaptive scheme is required in the prediction mechanism.

4.3.1.2 Adaptation and Context

Adaptivity is an important concern in the autonomous data prefetching approaches, especially in the resource-constrained mobile computing environments in which network bandwidth, and hardware resources (i.e., cache size, energy) are limited. Without a properly designed strategy, a prefetching scheme may incur excessive resource costs (Yin et al.; 2002; Drakatos et al.; 2009; Pallis et al.; 2008).

A number of researchers have proposed approaches to improve the adaptivity of their prefetching schemes in different aspects. An earlier work proposed by Jiang and Kleinrock (1998) was concerned with system resource usage. In their approach, the prefetching behaviour was dynamically adjusted based on the access performance. In the work of Pallis et al. (2008), a policy and proxy-based prefetching strategy was proposed. Service consumers have been categorised into different cluster groups based on their interests, so the proxy can prefetch data more efficiently, and reduce the bandwidth cost. Yin et al. (2002) proposed a value-based adaptive prefetch (VAP) scheme, in which each data item has been assigned a value. Based on the assigned value, the current remaining power level, the access rate, the update rate, and the data size, the system can evaluate the cost of prefetching, and adjust the prefetching decision dynamically. Hu et al. (2003) proposed the Sliding Cache technique for adaptive prefetching, in which the cache space was dynamically changed

based on the usage of the cached data item. Their evaluation showed that the approach can reduce the frequency of prefetching processes, and the results showed that the lesser the frequency of prefetching the lower the energy cost.

Improving the accuracy of prefetching is one of the most important aspects to improve adaptivity. Drakatos et al. (2009) proposed a context-aware cache management prefetching strategy. The proposed cell-based mobility scheme is capable of detecting user's movement, and predicting use's future location. Based on the predicted future location together with query patterns (previous query records), the system is able to prefetch data item more accurately. The authors have also mentioned that if a user's preference model has been applied, the accuracy of prefetching can be explicitly improved. However, further detail in this respect was not elaborated in their works.

User preference profiling is one of the major aspects to improve the accuracy of the prediction strategy. When accuracy is increased, the overall adaptivity is also improved due to the resource costs being reduced. However, existing works (Bürklen et al.; 2006; Choi et al.; 2005; Del Prete and Capra; 2010) did not consider the dynamism of the user's preference. It is near impossible and inconvenient for most ordinary users to manually pre-define various preferences for all possible situations. The system needs to autonomously compute user's preference at runtime not only based on the historical query records, but also taking user's current context into consideration. To overcome this challenge, our proposed strategy aims to dynamically predict user preference at runtime using context-aware mechanisms.

Recall that in Section 2.4.1, we mentioned that Dey (2001) defined context as: *'any information that can be used to characterise the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves.'* In the following paragraphs, we use Dey's definition as the basis to describe context in our system.

4.3.2 Context-aware User Preferred Service Prediction

The main technique that ensures the success of proactive service discovery in our system is the context-aware prediction scheme. The context-aware prediction scheme takes user's current contexts as the basis, and then compares the current contexts to historical records to compute which query requested by the user has the highest probability. Each query recorded by the system has its associated *semantic service type*. By predicting the highest probable query, the system is capable of identifying what *semantic service type* is interested by the user in current environment. In this section, we describe our proposed context-aware prediction scheme.

Definition 4.1: ***Raw Context Data*—B.** $B = \{b_i : 1 \leq i \leq N\}$. A b_i is the data retrieved from context providers such as Global Positioning System, Compass application, image sensor, video sensor, voice sensor, and so on. A b_i will be used as the basic input parameter to describe an interpreted context.

Definition 4.2: ***Interpreted Context*—C.** C is a set of output from a rule-based context interpreting process, in which $C = \{c_j : 1 \leq j \leq N\}$. Each $c_j \in C$ consists of ID, *type*, *value*, and a set of associated raw context data B_{c_j}.

Based on Delir Haghighi et al. (2008)'s work, an interpreting rule consists of context type ($type^{c_1}$), the scope of raw context data value, which includes minimum value and maximum value, and the output represents the interpreted value from this definition. For example, an interpreting rule describes $inputMin$ = "$x12y14$", $inputMax$ = "$x37y22$", *type* = "*location*", *output* = "$MeetingRoom$". When a retrieved location context contains a value: $x15y17$, which is within the scope of $inputMin$, and $inputMax$, the system will consider the location "$MeetingRoom$" as one of the current contexts.

Definition 4.3: ***Query Records*—R.** Each device should maintain a set of query records R, in which $R = \{r_k : 1 \leq k \leq N\}$. R represents the device user's previous queries associated with corresponding contexts. Each record r_k consists

of a query q_{r_k} and a collection of context information C_{r_k} occurred when q_{r_k} is submitted by the user.

A q_{r_k} represents a request query submitted by the user for invoking either an internal embedded Web service on his/her device or an external Web service provided by other mobile device peers within the network. A q_{r_k} consists of ID, *parameters*, and the corresponding *semantic Web service operation type*.

Definition 4.4: *Raw Candidate Queries*—Q. $Q = \{q_l : 1 \leq l \leq N\}$. Q is a set of non-duplicate queries from R:

$$Q = \bigcup_{k=1}^{|R|} q_{r_k} \tag{4.1}$$

When the *Predictor* component receives a set of contexts, it can predict the user's query based on the comparison result between the current contexts and the contexts of each query record. User may also define a preferred query manually by setting a set of context and the corresponding query in a file, which will be loaded in the beginning of the process. If user's definition exists, it will be used as the priority option. Otherwise, the system will perform the prediction automatically.

Let $\tilde{C}$ be a set of current contexts, where $\tilde{C} = \{\tilde{c_i} : 1 \leq i \leq N\}$. By applying Bayes' theorem (Clema and Fynewever; 1973), the probability of a $q_l \in Q$ with one associated context c_i can be computed from (4.2):

$$P(q_l|\tilde{c_i}) = \frac{P(\tilde{c_i}|q_l) \cdot P(q_l)}{P(\tilde{c_i})} \tag{4.2}$$

where $P(\tilde{c_i}|q_l)$ is the probability of $\tilde{c_i}$ when q_l was requested. It is computed from (4.3):

$$P(\tilde{c_i}|q_l) = \frac{|\{r_k \in R : q_{r_k} \equiv q_l \wedge \exists c_x \in C_{r_k}, c_x \equiv \tilde{c_i}\}|}{|\{r_k \in R : q_{r_k} \equiv q_l\}|} \tag{4.3}$$

$P(q_l)$ is the probability number of occurrence of q_l in R, in which $P(q_l) = \frac{|\{r_k \in R : q_{r_k} \equiv q_l\}|}{|R|}$.

$P(\tilde{c}_i)$ is the probability of a random selected query that contains $\tilde{c}_i$ as one of its attributes. It is computed from (4.4):

$$P(\tilde{c}_i) = \sum_{r_k \in R} (P(\tilde{c}_i|q_{r_k}) \cdot P(q_{r_k})) \tag{4.4}$$

By considering all the involved context, the probability of q_l (denoted by $P(q_l|\tilde{C}, R)$ will be refined as (4.5):

$$P(q_l|\tilde{C}, R) = \sum_{\tilde{c}_i \in \tilde{C}} \left(\frac{P(\tilde{c}_i|q_l) \cdot P(q_l)}{P(\tilde{c}_i)} \cdot \frac{1}{|\tilde{C}|} \right) \tag{4.5}$$

The calculation from (4.5) is based on considering the importance of all involved contexts equally. However, the importance of each context must be distinguished by different users. Hence, we apply the *weight of context* (Delir Haghighi et al.; 2008) in our scheme.

Definition 4.5: ***Context Importance Rules*—G.** G is a finite set of rules, where $G = \{g_m : 1 \leq m \leq N\}$. Each g_m consists of a corresponding context c^{g_m} and a corresponding query q^{g_m}, and the weight value denoted by v_{g_m}.

v_{g_m} is a user-defined value in the context importance rules (G) for clarifying the importance of a context type to a query. By default setting, each context type has equal importance (set to 0) to all the queries. For example, a user may consider the location context to be more important to a query for searching the train arrival time. Hence, the user can increase the importance of the location context (e.g., set it to a number greater than zero) to the query to improve the prediction accuracy. Such a setting can also be applied globally. For example, user may prefer the location context should always be the primary consideration. Hence, whenever the prediction is performed, the location context will always be allocated a higher importance value than the other contexts.

By applying the *weight of context*, the final formula has been refined in (4.6).

$$P(q_l|\tilde{C}, R) = \sum_{\tilde{c_i} \in \tilde{C}} \left(\frac{P(\tilde{c_i}|q_l) \cdot P(q_l)}{P(\tilde{c_i})} \cdot \frac{1 + v_{g_m}^{c_i, q_l}}{|\tilde{C}| + \sum v_{g_m}} \right) \tag{4.6}$$

where $\sum v_{g_m}$ is the sum of a set of v_{g_m} in which $v_{g_m} \neq 0$. $v_{g_m}^{c_i,q_l}$ denotes one of the defined rule, where $v_{g_m}^{c_i,q_l} \leftarrow g_m \in G, c^{g_m} \equiv \tilde{c_i} \wedge q^{g_m} \equiv q_l$.

In order to let a user have enough control on the autonomous decision-making based on the prediction model, the user can manually define Context Filtering rules. A Context Filtering rule consists of a query type, and a list of contexts that should be ignored in the calculation. For example: A user is searching for the recommended food in the current area. For this search query, weather context and temperature context can be important if the *food seller type* is an outdoor bazaar, or it does not have enough indoor seats when customers are required to queue outside. On the other hand, a similar search method may not be influenced by weather and temperature if the query specifies the search criteria as "`restaurant`" + "`indoor`".

If user defined rules exist, in the prediction algorithm, the current contexts $\tilde{C}$ for a query q_l will be redefined to reflect whether a context should influence the q_l or not. For example, current contexts $\tilde{C}$ consists of $\tilde{c_1}$, $\tilde{c_2}$, $\tilde{c_3}$, and $\tilde{c_4}$. If user has defined that $\tilde{c_4}$ has no influence to *query type* q_y, when the prediction algorithm computes the $P(q_y|\tilde{C}, R)$ (4.6), $\tilde{C}$ will be redefined as $\{\tilde{c_1}, \tilde{c_2}, \tilde{c_3}\}$ excluding $\tilde{c_4}$.

A prediction scheme that relies on the user's historical record usually has a limitation in which the accuracy of the prediction can be low when there is not enough records. One solution is to apply *social context*. *Social context* represents the factors that can potentially influence a user's decision. For example, a friend f of a mobile user u, might have similar interest to u, and f might have been to the same place as where u is currently arriving. Since f and u are similar, they may prefer to interact with the same type of services at that location.

4.4 Trustworthy Service Discovery in MSNP

Imagine an MSNP user—A in Comiket (the self- publishing convention for comic, animation, games and related products) enabling his/her MSNP application running in the background to discover service providers autonomously. Based on A's past MSNP activities records and the search histories on the Web browser with the corresponding context information, A's *agent* predicts that A may be interested in services that provide content related to a particular topic. The *agent* then discovers a list of matched service providers—S, $(S = \{s_i : 1 \leq i \leq N\})$ who provide content related to the topic. In order to filter malicious providers' services from its search results, A's *agent* has to identify which $s \in S$ is trustworthy

A common approach to determine a service provider's trustworthiness is to utilise a reputation-based trust scheme. Figure 4.5 illustrates a basic reputation-based trust scheme for determining a service provider's trustworthiness.

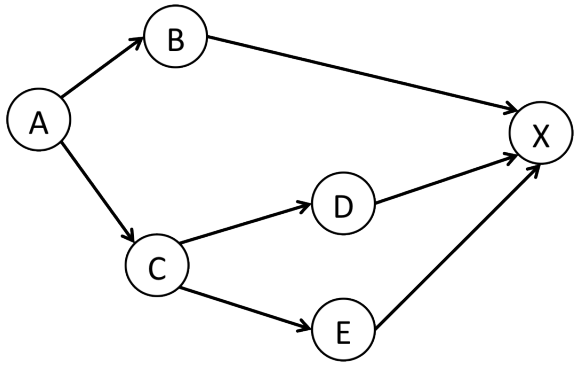


Figure 4.5: Reputation-based trust model example

In the figure, we have a number of participants in the model: A, B, C, D, E and X. A attempts to identify X's trustworthiness based on X's reputation rated by the other peers. A has two direct recommenders—B and C—and two indirect recommenders—D and E (which are leaf peers of C). Since B has direct interaction experience with X, B can report X's reputation rating value to A. Although C does not have interaction experience with X, C can forwards the query to D and E to retrieve their reputation rating of X. By calculating the average of the reputation rating values from direct and indirect peers, and with the weight of trust (i.e., A's

trust weight to B and C; C's trust weight to D and E), A can compute a final rating value of X.

We generalise the trust value of A to X (denoted by $T_{A\to X}$) using the following formula:

$$T_{A\to X} = \frac{\sum_{i\in DirR_A} rate_{i\to X} \times rate_{A\to i}}{\sum_{i\in DirR_A} rate_{A\to i}} \tag{4.7}$$

where $DirR_A$ denotes the direct recommenders of A. $rate_{i\to X}$ is the rating value of X from recommender i, in which either rated by the i itself or computed from the i's leaf peers. For example, C's leaf peers are D and E. Hence, the rating value of X from C (denoted by $rate_{C\to X}$) is computed by the formula below:

$$rate_{C\to X} = \frac{\sum_{j\in DirR_C} rate_{j\to X} \times rate_{C\to j}}{\sum_{j\in DirR_C} rate_{C\to j}} \tag{4.8}$$

where $DirR_C$ denotes the direct recommenders of C, which are D and E.

There are two basic reputation schemes used to select recommenders:

- *Reputation rating based on friends' rating.* A's *agent* can obtain the reputation rating data from the *agents* of A's friends who have direct interaction experience with a service provider X in the past history. For those friends who do not have direct interaction experience with X, they may assist A obtain the reputation rating data from their friends' *agents*, which is known as FOAF. By computing the reputation rating data from friends and FOAF, A's MSNP *agent* is capable of identifying the trustworthiness level of X.

- *Reputation rating based on public recommenders.* In many cases, A's *agent* may not be able to retrieve a fair number of reputation rating data from friends and FOAF. An alternative solution is to refer the rating from the other public proximal MSNP participants (strangers) with information such as:

  - *General rating*—A's MSNP *agent* may retrieve the reputation rating from random selected MSNP *agents* in proximity, and compute the trustworthiness level of X based on the raw reputation rating data. This approach

will cause a lot of overhead in MP2P environment. A better approach is to filter the number of required reputation rating data based on credibility.

– *Credibility*—Authors in the general P2P-based trust solution—PeerTrust (Xiong and Liu; 2004) have justified that one of the main elements used in identifying a recommender's trustworthiness is to compute its credibility. Credibility can be computed based on different approaches depending on the application requirement. The two approaches are:

  * *Based on the trustworthiness of the recommender.* It requires reputation data referred by other participants (e.g., from friends);
  * *Based on similarity.* In this approach, A's *agent* retrieves the reputation rating data from proximal participants' *agents* (relay peers) and compares the rating similarities of A with the candidate recommender peers. A list of peers who have rating similarity with A higher than the threshold (pre-defined by A) will be chosen as the recommenders of the reputation rating. Another related approach is to utilise the profile similarities. As described by Ziegler and Golbeck (2007), people attempt to trust people who have similarities.

4.4.1 Challenges

In the last decade, various trustworthy service discovery schemes have been proposed in the P2P computing area (Singh and Liu; 2003; Xiong and Liu; 2004; Rahbar and Yang; 2007; Aikebaier et al.; 2012). However, trustworthy service discovery in MP2P environments involves two major challenges that were not addressed in the generic P2P-based trustworthy service discovery schemes. They are:

1. *Unstable connectivity*

   Participants in MP2P environments are dynamic objects. Each participant can be randomly connected or disconnected from the network. A classic approach

to identify a service provider's trustworthiness is to utilise reputation scheme in which the service requester collects the service provider's reputation rating from the other participants in the network and perform trustworthiness calculation to identify the trustworthiness level of the service provider. Such an approach is less of an issue in a traditional P2P network, which have stable Internet connection. However, in an MP2P environment, the requester may face problem on collecting the reputation rating data from the other participants by direct invocation because many of them will be dynamically disconnected from the network before they have a chance to receive the request message.

2. *Resource constraint*

   A classic P2P-based approach to identify a service provider's trustworthiness requires a lot of reputation rating data from other participants. Such a requirement is difficult to be fulfilled in MSNP, because unlike traditional desktop computers that have stable power connections, mobile devices have limited battery power to operate continuously. Supporting trustworthiness processes by using classic P2P-based approach involves a lot of transaction overhead. Since mobile device users attempt to save battery power, they may disable the function to assist other participants in trustworthiness processes (e.g., disable the MWS provision mechanism).

As described previously in Section 3.5.1, each MSNP participant has a backend cloud storage. The first challenge described above can be resolved by utilising cloud storages. Each MSNP participant can synchronise its reputation rating data to its public accessible cloud storage. The corresponding URL link of the reputation rating data is described in its service description metadata. Hence, when a requester performs the service discovery process by retrieving other participants' service description metadata, it will obtain the corresponding URL links of each participants' reputation rating data.

Although utilising cloud storage resolves the unstable connectivity problem, retrieving a large number of data from cloud can cause latency issues for the overall

service discovery process. Although today's mobile Internet or WLAN can provide fast Internet speed, the response speed of cloud storage service is still relatively unstable. For example, we have performed a test on data retrieval from Dropbox cloud storage service using an Apple iPod Touch 4 generation in a Wi-Fi network environment, in which the upload and download speed were both 13 Mbps (transaction speed was tested on speediest.net[4] application) to retrieve 250 files by asynchronous call. Each file size is 3,406 bytes. The test result showed that the total timespan was 29 seconds, which is much longer than we expected.

In order to improve the performance of the trustworthiness process in MP2P environment, one solution is to reduce the amount of required reputation rating data. We review a number of existing approaches in the next section.

4.4.2 Related Works

A number of works have been proposed to support trustworthiness in MP2P environments. While works proposed by Li et al. (2010) and Rathnayake et al. (2011) were focusing on how to improve the reliability of trust models by utilising the computation of a large number of trust-related data, resulting in insufficient processing speed in MP2P network (Niu et al.; 2013), some authors (Wu et al.; 2009; Qureshi et al.; 2012; Waluyo et al.; 2012) have proposed lightweight trustworthy service/peer discovery schemes for MP2P environments.

Reducing data transaction is a common strategy to improve the processing speed of trust in MP2P. Wu et al. (2009) have proposed a group-based reputation scheme. Their design is based on super peer MP2P network, in which a super-peer (which is described as Power peer in their work) manages the reputation rating data of a group of mobile peers with similar movement speed. The super-peer is selected based on the peer's performance. A super-peer network such as JXTA can explicitly resolve the transaction overhead issue. However, in a public environment such as MSNP, users may not be willing to let their devices act as super-peers because the

[4]See `https://itunes.apple.com/US/app/id300704847?mt=8&ign-mpt=uo\%3D4`

high frequency of data transaction through their mobile devices can consume too much hardware resources (i.e. CPU, RAM, battery life, etc.).

M-Trust (Qureshi et al.; 2012) reduces reputation data transaction by selecting recommenders based on the confidence of the candidate recommenders. When a new peer joins the network, it collects the reputation rating data from other peers to establish an initial reputation data list. Based on the rating list, the new peer identify the most reliable recommender to provide its trust rating of a particular service provider. A disadvantage of M-Trust is that the system will directly remove a trustworthy peer's recommendation (reputation rating) when the peer is disconnected from the current network (either due to network switching or due to the *time to live* of its recommendation expiring). It would be ideal to provide a strategy to let M-Trust retrieve updates from recommenders in a different network, but this has not been addressed.

Similar to the fundamental strategy of M-Trust, TEMPR (Waluyo et al.; 2012) also improves the trust processing speed by utilising the selective recommender approach. Distinguished from M-Trust, the TEMPR scheme computes direct peers' (candidate recommenders who can directly interact with the requester) trustworthiness based on two scores: (1) the direct peers' trustworthy rating from other unknown peers; and (2) the direct peers' untrustworthy rating from other unknown peers. Once a set of trustworthy direct peers is identified, the requester peer can request them for a service provider peer's reputation rating. The trustworthy direct peers can also use the same approach described above to identify the trustworthy recommenders in their own groups of connected peers who are the indirect recommenders of the initial requester peer. Hence, TEMPR can reduce unnecessary message transactions.

Our work described in the following section can be seen as an extension of TEMPR, designed specifically for service-oriented MSNP. The major difference is that we do not assume strangers' *agents* will always forward messages to assist other participants for the trust processes. Hence, a requester who intends to identify a

provider's trustworthiness has to obtain the reputation rating data by either directly invoking the data provider *agent* (if the *agent* provides the corresponding Web service operation) or by retrieving the data from the data owner's cloud storage (based on the URL links described in the data owner's SDM).

4.4.3 Overview of A Lightweight Trustworthy Service Discovery for Mobile Social Network in Proximity

The aim of the following proposed scheme is to improve the speed of trustworthy service discovery in service-oriented MSNP by reducing transaction overhead in the process and not relying on message forwarding in order to avoid the issues caused by unstable connectivity and resource constraint. The fundamental strategy to reduce the transaction overhead is to utilise the selective trust reputation rating recommender scheme similar to existing works. However, we need to address two additional issues:

- How can MSNP participants share their reputation rating data? and
- How can a requester limit the number of its recommenders in the friend-based reputation model and in a public-based reputation model.

The later sections involve a number of elements. Hence, we define the meaning of the elements first.

Definition 4.6: ***Service Provider*—*SP*.** An SP is an MSNP participant that provides WS. It is defined as a tuple $(ID, services)$ where:

— ID denotes the identity of the SP

— $services = \{service_i : 1 \leq i \leq \mathbb{N}\}$ represents a set of WS provided by the SP. Each *service* has a name denoted by *SName* and a semantic service type denoted by *SType*

Definition 4.7: ***Previous Interacted Service Consumers List*—*PSC* *list.*** $PSC = \{(cid_j, IR_j) : 1 \leq j \leq \mathbb{N}\}$. An SP can optionally provide its PSC list to let the others know who have been using its services. A PSC is defined as a tuple (cid, IR) where

— cid denotes a service consumers' identity

— IR denotes interaction records between the service provider and service consumer, e.g., IR_j denotes a list of interaction records between the SP and the service consumer cid_j

Definition 4.8: ***Service Provider Ratings*—*SPR*.** $SPR = \{(ID_k, Rates_k) : 1 \leq k \leq \mathbb{N}\}$ where:

— ID_k denotes the identification of SP_k

— $Rates_k = \{(service_l^k, rate_l^k) : 1 \leq l \leq \mathbb{N}\}$ is a list of rating values of SP_k's services

— $service_l^k$ denotes one of the SP_k's services

— $rate_l^k$ denotes the rating value of $service_l^k$

Definition 4.9: ***Recommended References*—*RR*.** $RR = \{(SType_m, ID_m) : 1 \leq m \leq \mathbb{N}\}$ where:

— $SType$ denotes a semantic service type

— $ID_m = \{id_o^m : 1 \leq o \leq \mathbb{N}\}$ denotes a list of MSNP participants' IDs that are recommended as the rating reference for $SType_m$ services

Definition 4.10: ***Reputation Rating Data*—*RD*.** Each MSNP participant has a RD file in its device local storage as well as its cloud storage synchronously. An RD file contains two sets of data—SPR and RR.

Listing 4.1 illustrates a simplified *RD* in hash map format.

Listing 4.1: Simplified *RD* example

```
<key>Service Provider Rating</key>
<value>
      <key>SPID</key>
      <value>
            <key>URI</key>
            <value>
                  <key>type</key>
                  <value>semantic type value</value>
                  <key>Rate</key>
                  <value>rating value</value>
                  <key>transaction records</key>
                        <value>
                        <!-- URI, service type, time etc. -->
                        </value>
            </value>
      </value>
      <!-- Other interaction records ... -->
</value>

<key> Recommended References </key>
<value>
      <key>Semantic Service Type</key>
      <value>
            <key>ID</key><value><!-- URL of RD --></value>
            <!-- other IDs ... -->
      <value>
      <!-- Other types ... -->
</value>
```

An *RD* file can be obtained from either friends or other proximal MSNP participants. The prerequisite condition is how the requester *agent* retrieves the *RD* from the other *agents* (either from friends or public proximal participants). In a generic MANET environment, it is commonly assumed that the requester *agent* will collect the *RD* by broadcasting or multicasting its request message to the other participants' *agents*. This is not always applicable in MSNP. Fundamentally, MSNP operates in a dynamic public MP2P environment in which participants may not always be available. For example, when the requester *agent* intends to request a list of friends' *agents* for the *RD*, there may only be a few of them online. Another example, when the requester *agent* intends to request a list of proximal MSNP participants' *agents* for the reputation rating data, many may not even respond to such a request because they may have disabled such an operation to save their battery power.

To resolve the basic data retrieval problem in MSNP, each MSNP participant can utilises one or multiple backend public accessible cloud storage services to provide its reputation rating data to the others. The URL of the reputation rating data can be simply described in SDM. Hence, while the requester *agent* retrieves SDM in the first phase of service discovery process (see Section 4.2), it can already identify where to retrieve the reputation rating data provided by the other proximal participants. As for the friends' RD, since the requester has close connection with them, the requester would have already replicated their SDM files. Therefore, the requester *agent* always know where to retrieve the RD of the requester's friends.

One aspect in MP2P trust that was not addressed in most existing works is how service providers actively participate in the trustworthy service discovery processes. In real world services, providers always attempt to encourage consumers to use their services by using various schemes such as showing customers' rating and reviews of their products and services. Although in an MP2P trust system, service providers should not hold the rating of their own services (Singh and Liu; 2003), they can still provide a list of previous interacting service consumers.

When a requester intends to retrieve a service provider's reputation rating, the service provider can provide a PSC list (see Definition 4.7). The requester can use the *cid* of PSC list to collect RD instead of collecting all the RD of friends or proximal strangers. This approach can reduce unnecessary data transmission. Moreover, MSNP *agents* can identify that a service provider who does not provide the PSC list can potentially be a malicious node unless the service provider is new to the MSNP. If an MSNP participant is new, it may not have any interaction record with any other participants either as a service consumer or as a service provider. Hence, if an MSNP participant is not new, and it does not provide the PSC list, then this MSNP participant's service can be identified as potentially malicious. This notion is based on the reputation system of general online trading/shopping services such as eBay[5] or Yahoo Auction Japan[6]. In the case of only one matched service

[5]See `http://www.ebay.com/`
[6]See `http://auctions.yahoo.co.jp/`

provider found in the network, and it is a new MSNP participant, the *agent* should notify its user and let the user decide whether to invoke the service or not.

Considering the situation when a dishonoured service provider may provide an incomplete PSC list, which only describes a list of good records, the requester *agent* should not refer to the service provider's PSC list to identify the service provider's trustworthiness in the following cases:

- In the case of recommendation from friends: If none of the *cid* found in PSC belongs to the requester's trusted friends, the PSC should not be used.
- In the case of recommendation from public: If none of the *cid* found in the PSC belongs to highly creditable strangers, the PSC should not be used.

The following sections describe the proposed scheme for trustworthy service discovery in service-oriented MSNP.

4.4.4 Selecting Recommenders Based on Friends and Friend of a Friend

Due to privacy issues, the information about a person's trust rating value to his/her friends may not be accessible to other friends. For example, in Facebook, a user can hide all their posts from a friend and the friend will not know. Although the trust rating value is not accessible, the person can still provide a list of friends as Recommended References (RR; see Listing 4.1 and Definition 4.9) for a particular service type. The friends' IDs assigned in RR denote that the owner of RR trusts this list of participants' judgement for a particular service type based on their past experience.

RR is generated and updated when an MSNP *agent* performs service by referring to the RD of its user. RR only contains the IDs of trusted friends for a particular service type. If a friend in this list has given a high rating to a bad service provider, the friend's ID will be removed from the list. As a simple example, the MSNP application lets user manually block a service provider ID. When a service provider

ID is blocked, the friends who gave a good rate to the service provider will be removed from the corresponding Recommended References. On the other hand, when the list is empty and the recommendation was from random picked friend, if a friend's recommended service provider gives satisfactory recommendation to the requester, the friend's ID would be added to the list.

There are two approaches to assign friends to *RR*.

1. Assigning *RR* based on experience. Since an *RD* provides a list of ratings, an *agent* is capable of identify which friend of its user has the highest service interaction experience with a specific service type.

2. Assigning *RR* based on similarity. A user can assigned their friends to *RR* based on how similar their past rating to a particular service type. For example, *user A*'s past rating is very similar to *user B*. *user C* intends to refer *user A*'s rating to service provider—*H* who provides *K* type service. Unfortunately, *user A* does not have experience with *H*. However, since *user B*'s rating is similar to *user A*, *user A* has already assigned *user B* as *RR* of *K* type service. Hence, *user C* will refer to *user B*'s rating to identify the trustworthiness of *H*.

The rating seminaries between two users—*A* and *B*—can be computed by using Pearson Product-moment Correlation Coefficient below:

$$sim(A,B) = \frac{\sum_{s_o \in S}(rate_{A \to s_o} - \overline{rate}_r)(rate_{B \to s_o} - \overline{rate}_B)}{\sqrt{\sum_{s_o \in S}(rate_{A \to s_o} - \overline{rate}_A)^2 \sum_{s_o \in S}(rate_{B \to s_o} - \overline{rate}_B)^2}} \tag{4.9}$$

where S is the set of all the services rated by both A and B. $S = \{services^A \in SPR_A \cap services^B \in SPR_b\}$ and $S = \{s_o : 1 \leq o \leq N\}$. $rate_{A \to s_o}$ is A's rating to service s_o. $\overline{rate}_A$ is the average rating by A to all $services^A$. $rate_{B \to s_o}$ is B's rating to s_o, and $\overline{rate}_B$ is the average by B for all $services^B$.

Note that both approaches require a fair number of friends' *RD* replicated previously. For example, a user can replicate their friends' *RD* at home, then their *agents*

can apply the approaches to identify RR before the user using MSNP application outside.

The following algorithm outline the steps for a requester to identify the trust score of a service/content provider's service $s \in S$.

Algorithm 4.1:

Step 1. *Identify a list of friends who have experience with service—s.*

1.1. Requester retrieves PSC of the provider of s (PSC_s). We expect that the requester has a list of friends' IDs (denoted by FID, where $FID = \{fid_j : 1 \leq j \leq N\}$) stored in the local memory of the mobile device.

1.2. By searching the intersection between all the *cid* in PSC_s and FID, requester can find a list of friends who have service invocation experience with s—$MFID$, where $MFID = FID \cap CID$. If $|MFID| = 0$ then the process goes to Step 3. Otherwise, continue with Step 2.

Step 2. *Identify matched recommended references.*

2.1. As described previously, each MSNP participant has a RD. Let $MRR = \{rr \in RR : SType_{rr} \equiv SType_s\}$, where $SType_s$ is the semantic service type of s that the requester intends to invoke. RR is a list of friends' IDs that are recommended for identifying the reputation of a type of $SType_s$.

2.2. Let $RrID = MFID \cap MRR$. From $RrID$, the requester *agent* can identify the recommended friend(s) for $SType_s$ that also have experience with s, and refer the friend's rating to s. If $|RrID| = 0$, the process goes to Step 3.

Step 3: *Referring recommendation from recommended friend's FOAF.* When the requester's direct friends do not have experience with s, the requester will refer to the reputation rating from FOAF.

3.1. Identify a friend with the highest experience as a recommender and then based on the recommender's RD to find the friend of the recommender who has the highest experience with $SType_s$ and who also has rated s.

3.2. Once the FOAF is found, the requester will refer to the FOAF's rating of s. However, if none of the FOAF has experience with s then the process will proceed to the scheme described in Section 4.4.5—Selecting recommenders based on public.

4.4.5 Selecting Recommenders based on the Public

In this section, we describe the scheme to identify a service provider SP's reputation score based on the public proximal MSNP participants' ratings.

In the following descriptions, *credibility* will be used. Hence, we define the meaning of *credibility* of our scheme first.

Definition 4.9: *Credibility*—Cr. An MSNP participant's Cr, which is rated by the other peers, represents its reputation as a recommender for a type of service. The more MSNP participants' IDs shows up in the RR of every peer's RDs, the higher the MSNP participant's credibility is for being a recommender of the corresponding service type.

Algorithm 4.2 describes the scheme for selecting trustworthy recommenders from public proximal MSNP participants.

Algorithm 4.2:

Step 1: *Generating a candidate recommender list.* While the requester performs the service discovery process to find service providers who can provide the service of interest, the requester is also retrieving the RD of each proximal MSNP *agent*. This step consists of the following two tasks:

1.1. Let $PRRD$ be the set of RDs retrieved from all proximal *agents*. $PRRD = \{prrd_i : 1 \leq i \leq N\}$ where $prrd_i$ denotes the RD of each *agent* p_i. For

each $prrd \in PRRD$, the requester *agent* can identify that whether a p_i has interaction experience with service provider s or not.

1.2. Let MPR denotes the matched $PRRD$ in which $MPR = \{prrd_j \in PRRD | ID \in SPR_j \equiv ID_s\}$. ID_s denotes the ID of service provider s. If ID_s is found in one of $prrd_j$'s SPR but not in the PSC list of the provider of s, then either the $prrd_j$ is dishonoured or the provider of s is dishonoured.

Since the aim of this scheme is to identify the trust of s's provider, the final result will show its reputation score. However, dishonoured rating from the other participants will affect the accuracy of the scheme. Hence, the requester *agent* has to identify a recommender's trustworthiness before referring its reputation rating. Step 2 describes the process to identify a recommender's trustworthiness based on credibility.

Step 2: *Identify the credibility of a candidate recommender.* A proximal MSNP participant's credibility is computed based on the other proximal MSNP participant's rating. Suppose we want to compute a proximal MSNP participant p_i's credibility, we will use $PRRD$ excluding the RD of p_i. We use $CRRD$ to represents such a set of data, where $CRRD = \{crrd_m : 1 \leq m \leq N\}$. Step 2 consists of following two tasks:

2.1. Let Cr_p be the credibility of p, and Cr_p is computed by the equation below:

$$Cr_p = |\{crrd \in CRRD | ID_{rr_o^{crrd}} \equiv ID_p\}| \tag{4.10}$$

where $ID_{rr_o^{crrd_m}}$ denotes an MSNP participant's ID in the RR of $crrd_m$, and ID_p denotes p's ID in MSNP.

2.2. Once the credibility of each $PRRD$'s owner p_i is computed, the process goes to the next step.

Step 3: *Identify the experience of a candidate recommender.* People trust a person who has more experience about a specific subject. In existing works such as TEMPR (Waluyo et al.; 2012), the experience of p is directly related to the number of successful interactions completed between p and the service provider. Here, we consider the experience based on the type of service instead of a particular service provider's service. Because in the real world, a person may not use a service the second time when he/she had a bad experience with the service the first time. However, the person may have a lot of of experience using the same type of service provided by many different providers. Hence, the person's opinion is still valuable. For example, the review of a senior computer machine reviewer, who has over 100 reviews of notebook computers from different brands, is often being considered as more trustable than a junior reviewer who has only reviewed less than 10 number of notebook computers. Based on this assumption, the experience of p in our model is based on p's experience to a particular service type. This step involves the task below:

3.1. Let $STypeEx_{p_i \to s}$ be p_i's experience to $SType_s$. The experience value of p_i to $SType_s$ is computed by:

$$STypeEx_{p_i \to s} = |\{ir_l^{RD_{p_i}} \in IR^{RD_{p_i}} : SType_{ir_l}^{RD_{p_i}} \equiv SType_s\}| \quad (4.11)$$

where

- $IR^{RD_{p_i}}$ is the interaction records of p_i, in which $IR^{RD_{p_i}} = \{ir_l^{RD_{p_i}} : 1 \leq l \leq N\}$.
- $SType_{ir_l}^{RD_{p_i}}$ denotes the service type of the invoked service recorded in $ir_l^{RD_{p_i}}$.

Step 4: *Compute the trust score of a candidate recommender.* The trust score of an MSNP participant is the average of its normalised credibility value and its normalised experience value. The normalised value is computed based on the

overall comparison from all the other participants in P. This step involves the following two tasks:

4.1. For a particular MSNP participant—$\varphi \in P$ as a recommender of a service type (Tr), the trust score Tr_φ of φ is computed by the formula:

$$Tr_\varphi = avg\left(\frac{Cr_\varphi}{\sum_{p_i \in P} Cr_{p_i}} + \frac{STypeEx_{\varphi \to s}}{\sum_{p_i \in P} STypeEx_{p_i \to s}}\right) \tag{4.12}$$

where

- Cr_φ is φ's credibility value.
- $\sum_{p_i \in P} Cr_{p_i}$ denotes the sum of credibility values of all p_i.
- $STypeEx_{\varphi \to s}$ denotes the experience of φ for $SType_s$.
- $\sum_{p_i \in P} STypeEx_{p_i \to s}$ denotes the sum of all p_i's experience for $SType_s$.

4.2. Based on the computation result, the requester can choose a number of MSNP participants that have the highest Tr_φ value to be its recommender to compute the reputation score of s.

In this scheme we did not consider referring to FOAF's reputation ratings because referring to the reputation rating of proximal stranger's FOAF involves a large number of data transaction, which will cause data overhead and latency issues. In MSNP, there is a high chance that a candidate recommender's friend is not available for direct interaction. Hence, the requester has to retrieve the RD of the candidate recommender's friend from the friend's cloud storage.

As mentioned previously in Section 4.4.1, cloud storage services have unstable throughput which can cause latency issue. Performing the public reputation based scheme including FOAFs' RDs will involve a large number of transactions from cloud storage service, which will be deemed too heavy for mobile devices in MSNP environment.

An alternative solution is to use cloud utility services to enhance the overall performance of trustworthy service discovery. However, utilising cloud utility is not

a straightforward task in this case. The cloud-based architecture in the previous friends-based scheme enables proactive activities because friends know each other already. Dynamic factors such as unmeasurable number of proximal MSNP participants and their FOAF in the public reputation-based scheme means that it is preferred that the trust approach should be defined at runtime. It is therefore ideal to support an adaptation mechanism in MSNP when applying cloud services in this scheme.

4.5 Summaries and Discussion

In this chapter, we have presented the context-aware proactive service discovery scheme for service-oriented MSNP to reduce service discovery latency. Apart from service discovery in MSNP, trustworthiness has also been addressed. While existing work in trust management of MP2P environment focused on the trust model, and did not consider data transmission overhead issues, we have presented a lightweight trustworthy service discovery scheme specifically for service-oriented MSNP.

Although the proposed schemes in this chapter can enhance the overall service discovery performance, the dynamic nature of MSNP environment can still lead to unpredictable situations in which mobile devices cannot perform the service discovery effectively. For example, a service advertiser attempting to advertise its service by utilising push-based approach may suddenly run low on hardware resource availability due to the sudden increase in the number of MSNP participants joining the environment. Another example, during trustworthy service discovery, a requester may only have the option to refer to the reputation rating from public proximal recommenders in which the requester has to identify the credibility of each candidate recommender. If the number of candidate recommenders is large, the overall process of determining trust can create serious latency. There is therefore a need to utilise the task offloading mechanism of mobile cloud computing dynamically at runtime when the mobile device is unable to handle an unexpected situation in

service-oriented MSNP. Hence, in the next chapter, we present a solution to enable resource-aware adaptive service discovery scheme to adapt dynamic changes in service oriented MSNP.

Chapter 5

Adaptive Mediation Framework for Mobile Social Network in Proximity

5.1 Introduction

Considering the resource limitations of mobile devices and the dynamic nature of MP2P environment, communication becomes a crucial challenge to both content provider and content consumer. In order to enhance the overall performance of MSNP communication, the service discovery process can use different approaches depending on the environment status. For example, some tasks such as semantic service/content matchmaking process may be distributed to remote cloud services (e.g., GAE, Amazon EC2). However, distributing tasks to cloud is not always an effective solution, because utilising cloud service consumes extra costs such as network latency, price of using the service etc. In some cases, retaining the communication within local wireless network is better when both performance and cost are considered, especially when there are only a few MSNP peers involved. Conversely, when there are many MSNP peers involved, it may be more effective to distribute more tasks to the more powerful cloud services. It would be helpful if there exists a system

which can adapt to different situations when dealing with change in the number of MSNP peers. Hence, we design a framework which is capable of dynamically changing its approach at runtime to adapt to these situations, while a particular MSNP peer performs MP2P social network activities.

This chapter presents the **A**daptive **M**ediation framework for mobile **S**ocial **N**etwork in **P**roximity—AMSNP. A generic framework designed to adapt to dynamic changes in MSNP environments based on resource-awareness (Chang et al.; 2012; Chang, Srirama and Ling; 2013).

The remainder of this chapter is structured as follows: in Section 5.2, we summarise the works on adaptive workflow management systems for mobile services. This is followed by our proposed framework in Section 5.3. Section 5.4 describes the workflow-based adaptive task reconfiguration model. Section 5.5 provides case studies to explain how the proposed workflow model is applied in service-oriented MSNP tasks.

5.2 Adaptive Workflow for Mobile Services

In recent years, a number of works have been proposed to enable proximal-based MP2P social networks. However, existing decentralised MSNs are still in their early stages. Works proposed by Yang et al. (2008); Pietiläinen et al. (2009); Xing et al. (2009) were focused on how to enable the SNS activities in mobile P2P networks. Within decentralised MSN, two works have focused on how the content can be shared. Allen et al. (2010) have modelled the user's interest profiles, and also introduced a formal mathematical scheme to decide how the content can be proactively pushed to the friends/contacts with potential interest in the content. Li et al. (2012) have proposed ontology-based formal semantic models to enable content sharing using semantic content matchmaking scheme. The approach enables the user-interests-based content routing in decentralised MSN by analysing the similarity of user profiles. A common limitation in existing decentralised MSN solutions is that

they were tightly-coupled solutions with limited flexibility and scalability. The AM-SNP framework proposed in our work is a service-oriented solution based on ESB architecture design and standard technologies, which allows fundamental resources used in the participants' interaction to be changed dynamically at runtime.

Workflow Management Systems (WfMS) enable autonomous processes, which can highly reduce user's interference in content mashup and content advertisement scenarios. Researchers (Mecella et al.; 2006; Neyem et al.; 2008) in MP2P area usually apply WfMS in specialised purpose scenarios such as field-work, rescue operations or disaster events, in which the involved mobile nodes are manageable, and collaborate for the same goal. Workflow adaptation schemes in these works have focused on failure recovery or resource allocation. This is understandable because MP2P systems (in particular: MANET) deal with special purpose scenarios rather than general-purpose scenario (Conti and Giordano; 2007) like in MSNP. Few works have been done on proposing workflow systems for MP2P content mashup. Philips et al. (2010) have proposed a workflow system based on a Java API—AmbientTalk for mashup in MP2P environment. The work mainly focused on how to implement the workflow tasks on-top of AmbientTalk. In Avanes and Freytag (2008)'s work, an adaptive workflow scheduling scheme has been presented for mobile ad hoc network in disaster scenario. These works have been designed for similar MP2P environments such as MSNP. However, they do not address issues explicitly raised in this thesis. In this thesis, the workflow adaptivity mainly focuses on how to select the most feasible approach to complete the task of content mashup process based on performance (e.g., timespan of the approach) and costs (bandwidth, battery, transaction-load etc.).

5.3 AMSNP Framework

Our framework design is based on the Enterprise Service Bus (ESB) architecture (Robinson; 2004). ESB is a software infrastructure that can easily connect resources by combining and assembling services to achieve a Service Oriented Architecture (SOA).

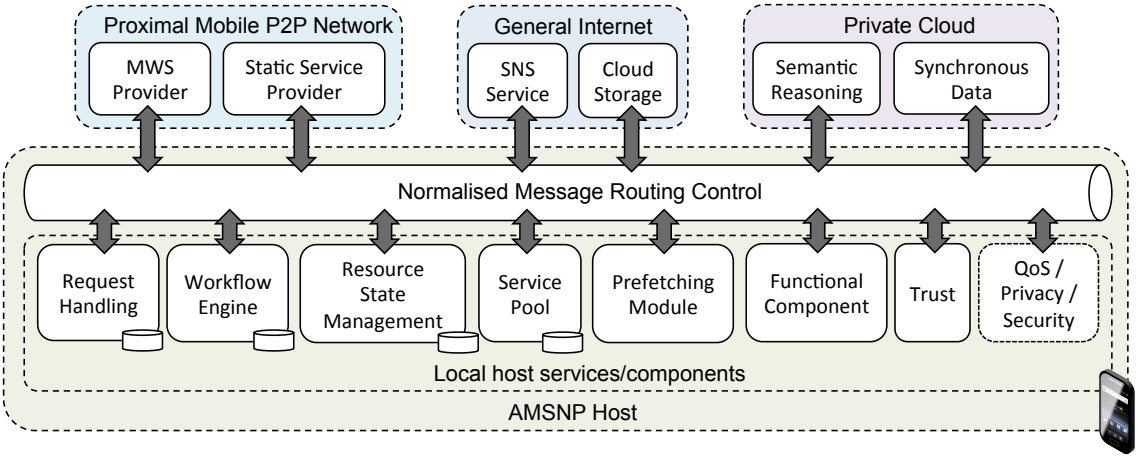


Figure 5.1: Architecture of AMSNP framework

Fig. 5.1 illustrates the architecture and main components of AMSNP. The architecture consists of four major components:

- **Proximal Mobile P2P Network**

  It represents the other MSNP peers within the same network. Depending on the developer's preference, an AMSNP host can support various network communication protocols such as XMPP[1], UPnP[2], Bonjour[3], etc.

- **General Internet**

  Basically, the content generated by the MSNP peers are updated to their SNS (e.g., Facebook, Twitter) or their cloud storages. In our design, the cloud storage services play an important role in MSNP. As mentioned previously, each MSNP peer synchronises its current IP address to its cloud storage in order to resolve the dynamic IP issue of mobile P2P network.

[1] http://xmpp.org/
[2] http://www.upnp.org/
[3] http://www.apple.com/support/bonjour/

- **Private Cloud**

  MSNP peer can utilise a number of backend cloud utility services for distributing tasks in order to reduce the resource usage of the device and also improve the overall performance. For example, the semantic service discovery process requires the MSNP peers to process a number of semantic metadata and matchmaking. Such a task can be distributed to its cloud utility services. Additionally, an MSNP peer can also synchronise some data to its private cloud, possibly in the form of cached service description metadata documents.

- **AMSNP Host**

  This is the largest component. It represents an MSNP peer with embedded AMSNP framework. An AMSNP host itself is built based on the ESB architecture. Each component of AMSNP is a service, and can be launched/terminated at runtime. A function can be performed by a local service within the AMSNP host, or it can be performed by an external service such as a private cloud utility service depending on the definitions of corresponding workflow. The AMSNP system is controlled by the WS-BPEL[4] workflow engine. When the user's application submits a request to AMSNP, the request will be handled by the *Request Handling* component, and a corresponding workflow will be selected. The selected workflow will then be passed to the workflow engine for execution via the message routing control component. Each workflow task is managed by a *Task Agent*. The *Task Agent* will decide how to perform the task after analysing the cost-performance scheme, which is described in Section 5.4.

  The AMSNP host itself contains the following components:

  1. ***Normalised Message Routing Control*** component handles incoming and outgoing messages. It process the message to meet the required format for the receivers of the message.

[4] http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.html

2. ***Request Handling*** component receives request messages from other applications (e.g., a User Application, which provides user interface for users to access AMSNP) and forward the request message to corresponding components via the Normalised Message Routing Control component.

3. ***Workflow Engine*** is the main component to process workflow documents (e.g., WS-BPEL documents) and executes workflow tasks.

4. ***Resource State Management*** service is responsible for continually monitoring the resource usages such as CPU usage, network bandwidth usage, cloud utility service usage, etc. These resource usages are cost intensive, and are the main elements influencing the decision making of the adaptation scheme to be described in the next section.

5. ***Service Pool*** is responsible for managing information on internal services, private cloud services, and services provided by external MSNP peers. It contains a collection of the service descriptions of external MSNP peers, the service descriptions of each internal service and each accessible private cloud utility service.

6. ***Prefetching Module*** encompasses four components for enabling the prefetching mechanism. They are described below:

   - ***Recorder***

     Each time the device user sends a request query to the mediation from the user application, the recorder will record the details of the request, and a set of current context.

   - ***Fetcher***

     It manages prefetched data item. Each data item is stored in a particular local directory in the device storage, and corresponding information can be retrieved later.

   - ***Predictor***

     It uses the prediction technique (to be described in the next section)

to predict the mobile user's query based on current context information.

- ***Context Manager***

  It continuously operates to retrieve up-to-date raw context data from *context providers*, and interprets the collected raw context data to compound contexts based on the pre-defined matching rules. For example, a rule may define a compound context - noise level is loud when the value of environmental context raw data - noise is between 30 and 50. A *context provider* can be an external sensor device, or it can be an embedded application within the same mobile device. Examples are: compass application, map application (e.g., Google map), sound detection application, etc.

7. ***Functional Components*** are miscellaneous utility components such as semantic metadata matchmaking component, message parsing and calculation component, for calculating the Cost Performance Index (CPI) value described in the next section.

8. ***Trust*** handles the trustworthy service discovery processes. The corresponding scheme has been described in the previous chapter.

9. ***QoS/Privacy/Security*** are additional components needed to improve the quality of service and security requirements. They are not within the scope of this thesis. We will consider them in our future work.

5.4 Adaptive Approach Selection based on the Cost-Performance Index Model

Each request received by the *Request Handling* component, has to be processed by triggering a corresponding business process workflow. In a basic workflow document (e.g., WS-BPEL), the endpoint (either a single service or a composite service) for processing each task/activity has been pre-defined in the document. Considering the dynamic nature of mobile P2P environment, the pre-defined endpoint may not be the best selection for the task. For example, a workflow is launched when the network has only 10 or less peers in existence. The workflow defines that the task for service discovery will be fully performed by a local host service of the device without using external distributed services. However, once the workflow is launched, the situation can change. There can be 50 more peers suddenly joining the network. Such a change can make the pre-defined approach no longer feasible. On the other hand, distributing tasks to external service (such as a service deployed on GAE) is not always the best approach because in many cases, performing tasks in the local host is more efficient. These concerns lead us to apply the dynamic adaptation technique, which is capable of identifying the best approach for each workflow task at runtime.

In this section, we propose an adaptation scheme that can decide which *approach* should be chosen for each workflow task at runtime based on the latency (timespan) of the *approach*, and costs. In order to clarify the terminologies used in this scheme, we first provide the following definitions:

Definition 5.1: ***Cost element set*—*E*.** E is a finite set, where $E = \{e_k : 1 \leq k \leq N\}$. Each e_k is a cost element defined as a tuple (n_k, v_k) where:

- n_k is a unique name of e_k
- v_k is the cost value of e_k.

Definition 5.2: ***Workflow.*** A workflow is a set of sequential or parallel tasks—T, $T = \{t_i : 1 \leq i \leq N\}$, to achieve a goal.

Definition 5.3: ***Approach.*** Each task $t \in T$ in a workflow can be completed by a number of pre-defined *approaches*—A_t, $A_t = \{a_j : 1 \leq j \leq N\}$.

Given a workflow with a set of tasks T, an *approach* $a \in A_t$ of a task $t \in T$ is defined as a tuple (p, E) where:

— p is a performance value

— E is a set of cost elements.

$E_{a_j}^{t_i}$ denotes the cost element set of an approach a_j of a task t_i.

The *approach* for a task is selected at runtime after the workflow is launched, and the decision is made based on the cost and performance. Note that for the rest of this thesis, *'approach'* denotes the term described in this definition.

For example, a set of services S, $S = \{s_i : 1 \leq i \leq N\}$ has been discovered that can provide the content requested. The task t of invoking an $s \in S$ to retrieve content, can be either performed by *approach* a_1: using a local host component to retrieve all content or performed by *approach* a_2: distributing the process to a cloud service and then synchronising the result to the user's mobile device, i.e., $A_t = \{a_1, a_2\}$ in this example.

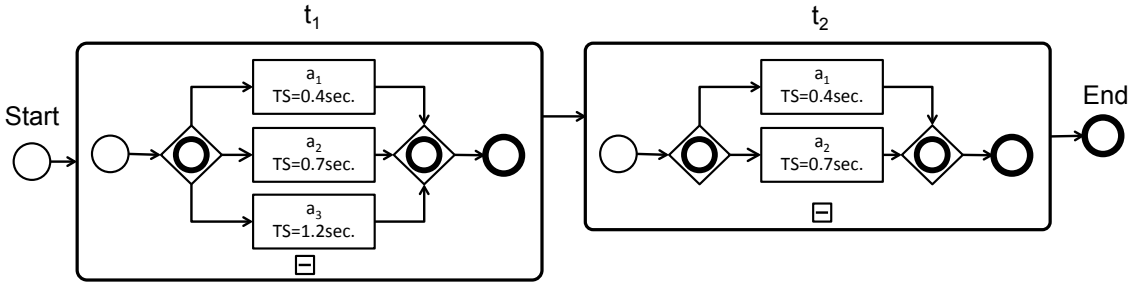


Figure 5.2: Workflow path selection based on timespan

Figure 5.2 shows a sample workflow which has two tasks $T = \{t_1, t_2\}$. For task t_1, there are three selective *approaches*, and for task t_2, there are two selective *approaches*. Each *approach* consumes different timespans, which contributes to the *cost elements* of the *approach*. In order to achieve the goal effectively, the system needs to identify the shortest path (i.e., select the approach that induces the lowest

timespan for each task) to reach the goal. Initially, the shortest path can be obtained by (5.1).

$$timespan = min\left\{\sum_{i\in T, j\in A_i} \tau_j^i\right\} \tag{5.1}$$

where τ_j^i denotes the timespan of *approach* a_j of task t_i.

However, the shortest timespan may not mean that the *approach* selection is the most efficient when cost is considered. Hence, we propose a CPI-based scheme to enable our workflow system to analyse and select the most efficient *approach* at runtime. The scheme combines fuzzy set (Zadeh; 1965) and the *weight of context* (Delir Haghighi et al.; 2008). The reason we use fuzzy set is to compare the performance and cost between *approaches* instead of using static values.

Let D_{t_i} be a set of timespan value for the selective *approaches* (A_{t_i}) of task t_i, where $|D_{t_i}| = |A_{t_i}|$, $D_{t_i} = \{d_j : 1 \leq j \leq |A_{t_i}|\}$, in which d_j represents the timespan of a_j, $a_j \in A_{t_i}$. Let L be the longest timespan in D_{t_i}, where $L = max\{d_j \in D_{t_i}\}$. The performance value of each *approach* R_{a_j} is computed by (5.2):

$$R_{a_j} = \begin{cases} 1 & \text{iff } d_j \equiv L \\ (L+1) - d_j & \text{otherwise} \end{cases} \tag{5.2}$$

Let $\tilde{A}_{t_i}$ be the fuzzy set of A_{t_i}, $\tilde{A}_{t_i} = \{\tilde{a}_j : 1 \leq j \leq |A_{t_i}|\}$. We need the normalised fuzzy number of the ranking values. Hence, the fuzzy number of an *approach's* ranking value (denoted by $\tilde{R}_{a_x}$) is:

$$\tilde{R}_{a_x} = \frac{R_{a_x}}{\sum_{a_j \in A_{t_i}} R_{a_j}} \tag{5.3}$$

where R_{a_x} is the performance value of a_x derived from (5.2), and $\tilde{R}_{a_x}$ is the normalised fuzzy number of the performance value of a_x, in which $0 \leq \tilde{R}_{a_x} \leq 1$.

The cost element set (Definition 5.1) must be comparable between different related *approaches*. If *approach* a_1 for task t_1—$E_{a_1}^{t_1}$ contains the value of *'battery cost'*,

then the *approach* a_2 for task t_1—$E^{t_1}_{a_2}$ must also contain such a value. Accordingly, the overall CPI between different *approaches* can be compared.

Since we are comparing the cost elements between different *approaches*, the normalised value of a cost element $\tilde{v}_{e_x}$ is computed by (5.4).

$$\tilde{v}_{e_x} = \frac{v_{e_x}}{\sum_{e_k \in E^{t_i}_{a_j}} v_{e_k}} \tag{5.4}$$

and the average value of the total cost of a_j (denoted by $\Upsilon^{t_i}_{a_j}$) is computed by (5.5).

$$\Upsilon^{t_i}_{a_j} = \frac{\sum_{e_k \in E^{t_i}_{a_j}} \tilde{v}_{e_k}}{\left|E^{t_i}_{a_j}\right|} \tag{5.5}$$

By applying the basic CPI model, the cost-performance value—δ of an *approach*—a_j is:

$$\delta^{t_i}_{a_j} = \frac{\tilde{R}_{a_j}}{\Upsilon^{t_i}_{a_j}} \tag{5.6}$$

However, the importance of the weight of an e_k is different for different users. For example, when the device battery-life remains 50%, the user may consider that saving the battery life of his/her mobile device is more important than spending money on using cloud services for computational needs. In this case, the weight of the battery life cost element will be higher than the weight of the bandwidth cost of the cloud service. Therefore, the normalised value of an e_k needs to be refined as $\tilde{v}_{e_k} \cdot w_{e_k}$, where w_{e_k} denotes the weight of e_k, and the cost is re-defined as follow:

$$\widehat{\Upsilon}^{t_i}_{a_j} = \frac{\sum_{e_k \in E^{t_i}_{a_j}} \tilde{v}_{e_k} \cdot w_{e_k}}{\sum_{e_k \in E^{t_i}_{a_j}} w_{e_k}}, w_{e_k} \geq 1 \tag{5.7}$$

Finally, the cost-performance value of a_j is re-defined:

$$\delta^{t_i}_{a_j} = \frac{\tilde{R}_{a_j}}{\widehat{\Upsilon}^{ti}_{aj}} \tag{5.8}$$

5.5 Applying the Proposed System to Mobile Social Network in Proximity Scenarios

In this section, we use two examples to show how the workflow system can be applied to MSNP scenarios. These examples are described in BPMN.[5] The notation has been chosen to describe the workflow process because it can be mapped to Web Services Business Process Execution Language (WS-BPEL),[6] which has been used in our prototype to control the processes.

Firstly, we describe the general behaviour of a *Task Agent*.

As mentioned in the previous section, in AMSNP, workflow tasks are managed by *Task Agents*. Figure 5.3 illustrates the general behaviour of a *Task Agent*.

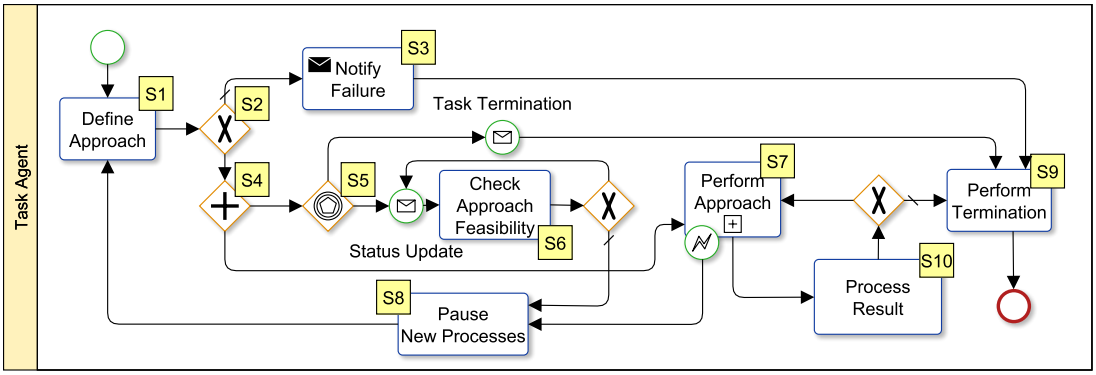


Figure 5.3: General Behaviour of Task Agent

When a task is launched, the first step (S1) defines a feasible *approach* (see Definition 5.2) based on the CPI scheme described in the previous section. If a feasible *approach* cannot be found (S2), the *Task Agent* will send the failure notification (S3) to the workflow engine and perform *task termination* (S9). On the other hand, if a feasible *approach* has been defined, *Task Agent* will perform the *approach* (S7) and also start receiving incoming messages (S5). There are two types of messages that can be received by the *Task Agent*:

- *Task Termination*—when the *Task Agent* receives this message from the workflow engine, it will immediately perform *task termination* (S9) in which all the

[5] http://www.omg.org/spec/BPMN/2.0/
[6] http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.html

related services started by the *Task Agent* will be stopped, and the *Task Agent* will notify the workflow engine to release the *Task Agent* from memory.

- *Status Update—Task Agent* is always receiving the device resource status update from the *Resource State Management* component for a period of time (e.g., every 60 seconds). When *Task Agent* receives the *Status Update*, it will review the feasibility of the current *approach*. If the current *approach* is still feasible, the process will continue. Otherwise, *Task Agent* will pause incoming new jobs (S8) and re-define its approach.

While the *Task Agent* is performing its *approach*, if any error occurs, for example, the *approach* uses *CloudUtil* for semantic service matchmaking and the *CloudUtil* is suddenly unavailable, the *Task Agent* should pause the current process (S8) and re-define its *approach*. The result generated by the executed *approach* should return to *Task Agent*, so *Task Agent* can process the result (e.g., forward the result to the task requester) (S10). Each time a result is returned, *Task Agent* will check whether the task should continue or not. If the task has been notified to terminate or it has completed, *Task Agent* will perform *task termination* (S9).

5.5.1 Service Description Metadata Prefetching Scenario

In this section, we use an example to show how the proposed workflow system can be applied to a *service description metadata prefetching* scenario. In the scenario, an MSNP peer (*peerX*) has just arrived in an MSNP-enabled environment. *peerX*'s device has been previously set to auto-discovery mode. Hence, it triggers the *SDM prefetching* process. Recall that this is performed by the *Prefetching Module* of AM-SNP host described in Section 5.3. In the following paragraphs, we call it *prefetching*. The basic sequence of *prefetching* is static and consists of four tasks (see Figure 5.4):

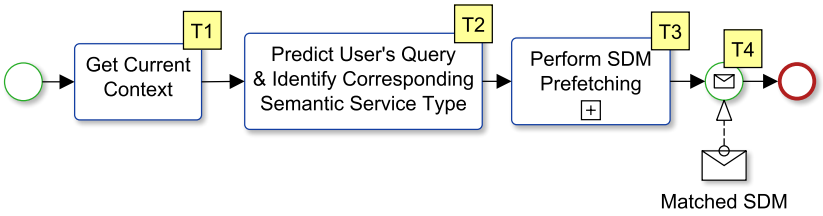
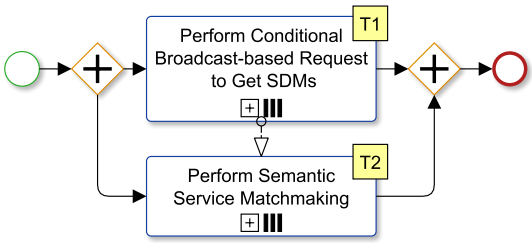


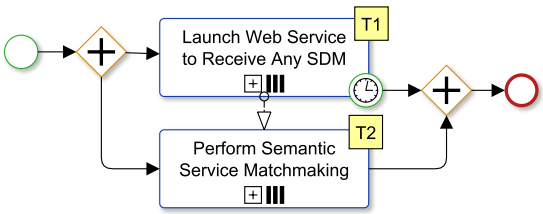
Figure 5.4: Service Description Metadata Prefetching

- T1—retrieve current environmental context information via the *Context Manager*.
- T2—pass current context information to the *Predictor* in order to identify which semantic service type is of interest to the user in the current environment.
- T3—trigger a subprocess to discover which proximal MSNP peers' services correspond to the target service type identified in T2.
- T4—receive matched SDM and send it to local memory.

T3 can be performed by three different *approaches*, which are:



(a) Pull



(b) Push

(c) PrefPush

Figure 5.5: *Approaches* of the SDM Prefetching Task

1. **Pull-based Service Discovery (*Pull*).** *Pull approach* (Figure 5.5(a)) consists of two parallel sub-tasks. Task T1 represents a process in which *peerX* will send a simple HTTP request to a selected group (e.g., within the same subnet) of proximal MSNP peers to retrieve their SDMs.

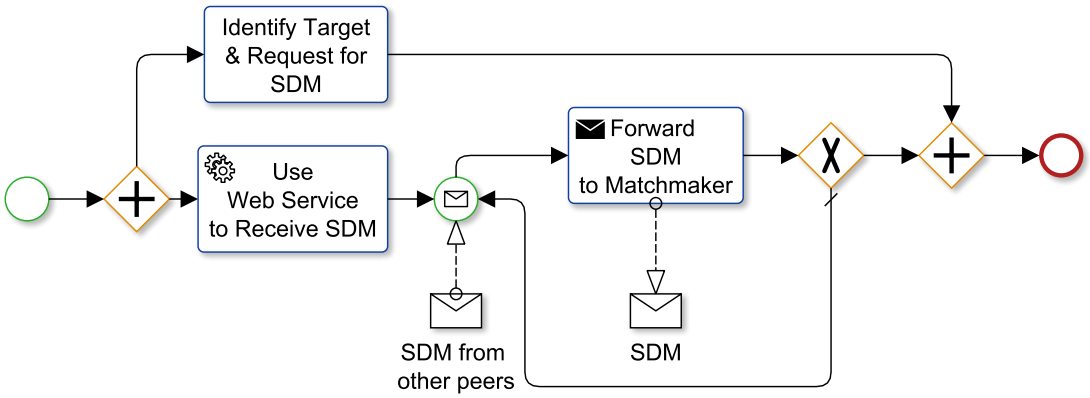


Figure 5.6: SDM Retrieval Subprocess

Figure 5.6 illustrates the details of T1. The SDM retrieval process of T1 is different from the classic HTTP request/response process. In T1, *peerX* asynchronously sends out request messages without waiting for responses, and a Web service will be launched to receive SDM from MSNP peers who received *peerX*'s request messages. It is to ensure that the communication will not be discontinued because of unforeseen circumstances (e.g., temporary disconnection or switch network). When a SDM is received in T1, the SDM will be forwarded to T2 for the semantic service matchmaking process. If the SDM contains the matched service type, it will be sent to the *Task Agent* which handles the original *prefetching* task.

2. **Push-based Service Discovery (*Push*).** Similar to *Pull*, *Push approach* (Figure 5.5(b)) also consists of two parallel tasks for retrieving SDMs and for semantic service matchmaking. T1 of *Push* also forwards the received SDM to T2 for the matchmaking process. The difference is that in T1 of *Push*, *peerX* only launches a Web service to receive incoming SDM (for a period of time) without requesting for it. This *approach* assumes that the other MSNP peers

will randomly perform *'blind flooding'* to send their SDM to proximal peers regardless of the proper Web service operation type for receiving the SDM. This *approach* requires $peerX$ to perform an additional job to identify what data it has received.

3. **Preference-Assisted Push-based Service Discovery ($PrefPush$).** $PrefPush$ only contains one subprocess, which launches a Web service to receive incoming requests for $peerX$'s SDM, *preferred service type*, and *matched SDM*. When it receives a *matched SDM*, it will forward the SDM to the *Task Agent* who handles the original *prefetching* process. In this *approach*, an active MSNP peer ($peerA$) can perform the following process sequence to *push* its SDM to $peerX$:

   1. $peerA$ retrieves $peerX$'s SDM in order to identify which Web service operation provided by $peerX$ is for replying the preferred service type of $peerX$.
   2. $peerA$ retrieves $peerX$'s preferred semantic service type.
   3. $peerA$ performs semantic service matchmaking process to identify if any of its service can fulfil $peerX$'s need.
   4. If $peerA$ can provide the service that $peerX$ needs, $peerA$ will send its SDM to $peerX$. Otherwise, $peerA$ will not perform further action.

   Compared to the previous two *approaches*, $PrefPush$ can reduce the overall latency caused by performing semantic service matchmaking, because the matchmaking task is done by the other MSNP peers.

Although in this example, we only describe three *approaches* for service discovery. In real world practices, other different *approaches* need to be defined in order to adapt to different situations. Since the focus of this chapter is on the mediation framework, we will not describe all the possible service discovery *approaches* here.

5.5.2 Content Advertising Scenario

In this section, we use an example based on a scenario in which an MSNP peer (denoted by *PeerX*) intends to proactively advertise content recommendation metadata (containing information about the service provider that has provided the content which is of potential interest to the receiver) to other MSNP peers. In this example, the workflow consists of two parallel tasks:

- *discovery* (T1)—find peers which are interested in the provided content. T1 consists of two sub-tasks: *Peer Discovery* and *Preference Matchmaking*. *Peer Discovery* denotes the process of discovering physical peers in MSNP environment and retrieving the content/service preference metadata from the peer. The result of *Peer Discovery* will be sent to *Preference Matchmaking* to identify the peer who is interesting in the provider's content/service or not.
- *advertising* (T2)—send the content recommendation metadata to the matched peers.

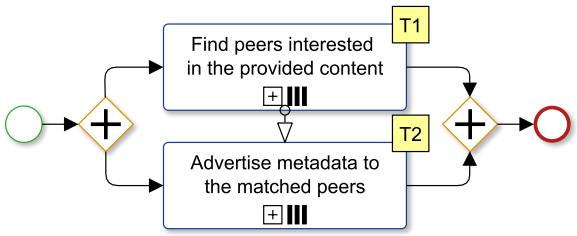
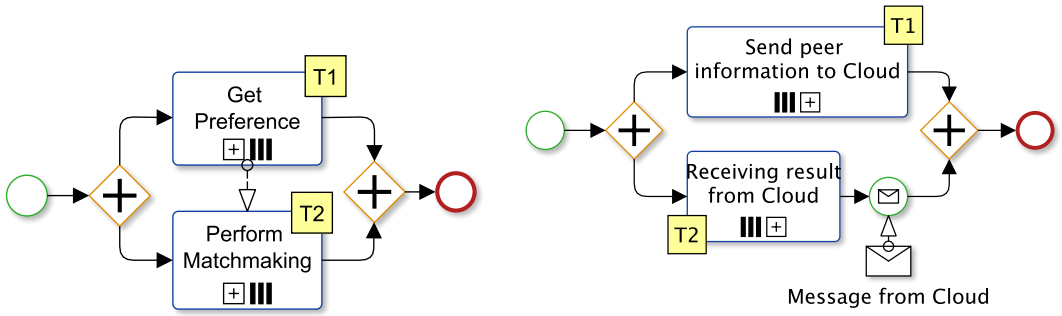


Figure 5.7: Content advertising workflow

In this example, two *approaches* have been defined for task T1 in Figure 5.7, which are mobile-based discovery (Figure 5.8(a)) and cloud-based discovery (Figure 5.8(b)). Each is a sub-workflow and consists of two parallel tasks. For the *approach* in Figure 5.8(a), the task agent will perform a subprocess (Figure 5.8(a)—T1) to retrieve the service preference metadata from each MSNP peer in the network. The response message received by Figure 5.8(a)—T1 will be passed to Figure 5.8(a)—T2 for service matchmaking process. As for Figure 5.8(b), which is the cloud-based *approach*, the mobile host will send a request to its Cloud Utility service (CloudUtil)



(a) Mobile-based discovery (b) Cloud-based discovery

Figure 5.8: *Approaches* of the discovery task (T1 in Figure 5.7)

when an MSNP peer is found (see Figure 5.8(b)—T1). The request message contains the basic information about the peer (e.g, the URL to retrieve its current IP address), the CloudUtil will retrieve and process the service preference metadata from each MSNP peer to find out which peer is interested in the content provided by the *PeerX*. The parallel task (Figure 5.8(b)—T2) was launched at the same time as Figure 5.8(b)—T1 to receive result from the CloudUtil.

The result of Figure 5.8(a)—T2 or Figure 5.8(b)—T2 will be sent back to the original workflow. When the original workflow receives the response, the result from service matchmaking will be sent to the task agent which manages the advertisement task (Figure 5.7—T2).

5.6 Summary and Discussion

This chapter proposed a resource-aware workflow-based adaptive mediation framework AMSNP (Chang et al.; 2012; Chang, Srirama and Ling; 2013) for service-oriented MSNP. The framework enables an MSNP participating device to dynamically change its behaviour to adapt to different situations when it receives a user's request. The adaptation mechanism utilises the proposed CPI scheme to support the device to automatically select a feasible approach for each task within a request handling process by comparing the dynamically changed cost and performance of the approaches.

Workflow systems provide flexibility and scalability to MSNP processes. The adaptation scheme introduced in this chapter enables the system to select a feasible approach to complete the workflow task. It also potentially brings a new form of MSNP communication. For example, an active peer in an MSNP environment can provide a recommended routing approach (described in WS-BPEL) to a new peer joining the network. The new peer can automatically execute the WS-BPEL workflow process to perform service discovery or content retrieval without the need for users manual control.

Chapter 6

Prototype Implementation and Evaluation

6.1 Introduction

In the previous chapters, we have described the proposed solution to enable service-oriented mobile social network in proximity (MSNP). In Chapter 3, we have presented the design of the architecture of the service-oriented MSNP. In order to reduce service discovery latency, In Chapter 4, we have presented the context-aware user preference associated push-based service discovery scheme and the schemes to support lightweight trustworthy service discovery. In Chapter 5, we have introduced the workflow-controlled mediation framework to support resource-aware behaviour for MSNP.

For proof-of-concept, we have developed and evaluated a prototype consisting of the components composing the mechanism described in our schemes. This chapter presents the evaluation methods and results of our prototype. In order to show the detailed evaluation of each proposed scheme, we have tested each component individually. Partial contents of this chapter have been previously published (Chang et al.; 2011, 2012; Chang, Srirama, Krishnaswamy and Ling; 2013; Chang, Srirama and Ling; 2013).

This Chapter is organised as follow: In Section 6.2, we explain the details of how the prototype was implemented and the details of the testing environment. Section 6.3 shows how much the service discovery process has improved by applying the user preference associated push-based service discovery scheme. Section 6.4 presents the evaluation of the context-aware user preference prediction scheme described in Section 4.3. Section 6.5 presents the evaluation of the trustworthy service discovery scheme described in Section 4.4.3 to 4.4.5. Section 6.6 presents the evaluation of the workflow component for supporting dynamic approach selection described in Section 5.4. Finally, we summarise and discuss our evaluation results in Section 6.7.

6.2 Prototype Implementation

We implemented the components of the AMSNP framework to evaluate the three main contributions of this thesis:

- the context-aware user preference prediction scheme for proactive service discovery, as described in Section 4.3;
- the lightweight trustworthy service discovery scheme, as described in Section 4.4; and
- the resource-aware adaptive approach selection scheme, as described in Section 5.4

The prototype was developed using the objective-C programming language and was tested on Apple iPod Touch 4th generation and Apple iPhone4S. The implementation detail of the components are described in the following sections.

6.2.1 Mobile Web Service

The basic mechanism which lets MSNP *agents* participant in the service-oriented MSNP is Mobile Web Service (MWS). As described in Section 3.5.2, depending on the user preference, an *agent* can either support the simple MWS client-side

mechanism only to discover and invoke services or support both MWS client-side and MWS server-side mechanisms. The implementation of the MWS mechanism are described below.

6.2.1.1 Mobile Web Service Provider

The MWS provided by each MSNP *agent* in the prototype is RESTful Web service. An advanced MSNP content consumer or a content provider is able to be an MWS host. In the prototype, an MWS host consists of two main components:

- ***HTTP Web server***

  As described in Section 2.3.1, many APIs were developed to deploy HTTP Web server on mobile OS. We used CocoaHTTPServer[1] API to enable the HTTP server mechanism. The advantages of using CocoaHTTPServer are: (1) it supports asynchronous socket communication, which can improve the speed of data transaction; (2) it supports Bonjour service publication. Web services provided by CocoaHTTPServer are discoverable by the Bonjour service discovering mechanisms. In the prototype, we use Bonjour as the main mechanism for MP2P service discovery. The HTTP Web server in prototype will respond with a SAWSDL document when the request message does not specify a particular path/operation name.

- ***Semantic Web service protocol***

  SAWSDL and OWL documents play an important role in MSNP. In order to enable autonomous service discovery and filtering, an MWS host is required to be able to process XML-formatted SAWSDL and OWL. We used Google Data API[2] to process XML-formatted data. Google Data API provides a fully functioned XML parsing mechanism. The SAWSDL and OWL data used in the prototype were written manually because there is no tool available to generate SAWSDL automatically from the source code written in Objective-C.

[1] `https://github.com/robbiehanson/CocoaHTTPServer`
[2] `https://developers.google.com/gdata/`

6.2.1.2 Mobile Web Service Client

The basic functions of an MSNP *agent* are: to discover other MSNP *agents* in its current network; and to invoke Web services provided by the other *agents*. In order to support the two functions, we have implemented two components:

- ***Web service invocation component***

  It supports two asynchronous HTTP method invocation mechanisms (`GET` and `POST`), which is compatible with RESTful Web services.

- ***MP2P service discovery***

  As mentioned previously, the prototype used Bonjour technology to support MP2P service discovery. Each MSNP *agent* has a Service Pool component (see Section 5.3) to monitor the current network. The Service Pool component utilises `<NSNetServiceBrowserDelegate>` to monitor the published MWS (by MSNP *agent*) in the Bonjour network. It manages a list of pushed MWS names. Depending on the discovery approach, it may automatically retrieve the SDM of each newly joined MSNP using the Web service invocation component.

6.2.2 Components of Adaptive Mediation Framework for Mobile Social Network in Proximity

One major benefit of ESB is its flexibility in dynamic reconfiguration of resources. In order to realise such a benefit, the components of AMSNP have been deployed as localhost Web services, accessible from `http://localhost` domain, so that the system can dynamically launch and release the components.

The following paragraphs describe the main components that have been implemented for evaluating AMSNP:

- ***Workflow Engine***

  The Workflow Engine (see Section 5.3) was implemented to support WS-BPEL 2.0. Since there is no existing BPEL API available for Objective-C, we have implemented a simple workflow engine component to process BPEL workflow tasks. The component supports `<sequence>` and `<flow>` of the BPEL 2.0 documents. The Workflow Engine component consists of two sub-components:

  - ***Workflow Manager*** receives a request message which triggers a corresponding pre-defined BPEL workflow to achieve the goal of the request. The Workflow Manager will launch a Task Agent for each task defined in the BPEL. The BPEL document parsing mechanism was implemented using the XML parser of Google Data API for iOS.
  - ***Task Agent*** mainly handles individual task. It decides how to perform the task based on the adaptive resource-aware approach selection scheme. Resource usage can be retrieved from the Resource State Management component.

  The evaluation of the Workflow Engine component is described in Section 6.6.

- ***Resource State Management***

  It monitors the resource usage of the mobile device. It records CPU and RAM usage every second. This component was implemented using the `<mach>` packages of the C programming language.

- ***Prefetching Module***

  Described as the 6th component of AMSNP host in Section 5.3, we have implemented the main functionality of the Prefetching Module—***Predictor***, whose function is to evaluate the context-aware user preference prediction scheme as described in Section 6.4.

- ***Trust***

  A component for evaluating the lightweight trustworthy service discovery scheme—***Trust Evaluator*** has been implemented. The details of the evaluation is described in Section 6.5.

- ***Service Pool***

  The implementation of Service Pool has been described previously in Section 6.2.1.2.

Evaluating the performance of service discovery may involve hundreds of MSNP *agents*. We did not have a large number of mobile devices to realise such an environment. However we have deployed hundreds of MSNP agents in a Macbook Aluminium 2008 version with Intel Core 2 Due 2.4 GHz CPU and 4GM RAM to simulate the environment. The wireless network for evaluation is on IEEE 802.11n 2.4GHz Wi-Fi environment controlled by an Apple Airport router which is Internet connection-enabled.

The following sections provide details on how each component was evaluated in order to present the proof-of-concept of our proposed schemes.

6.3 Proactive Service Discovery Performance

In Section 4.2.3, the user preference associated push-based service discovery ($PrefPush$) approach was described. The $PrefPush$-based service discovery approach utilises the context-aware user preference prediction scheme to let the requester *agent* provides its user preferred semantic service type to other service/content provider *agents* in the network. The approach can reduce the required metadata processing on the requester-side, hence, reducing the service discovery timespan of the requester *agent*.

In order to show that the proposed $PrefPush$-based service discovery approach can provide a better performance than the other two basic approaches—$Pull$ and $Push$. We have performed an experiment in a simulation environment to compare

the performance (timespan of service discovery process) and the costs (CPU usage and RAM usage) of the three approaches.

First, we re-iterate and describe the implementation details of the three approaches below:

- *Pull*, as described in Section 4.2.1, enables an MSNP requester *agent* who intends to search for a *cType* service in an MSNP environment to use active invocation to retrieve the other service provider *agents*' service description metadata (SDM) in order to identify which *agent*(s) can provide the *cType* service. In our setting, we assume each service provider *agent* has its own OWL file to describe its semantic type. Hence, the requester *agent* has to retrieve both SAWSDL and OWL from each service provider *agent*.

- *Push*, as described in Section 4.2.2, allows the MSNP requester *agent* to utilise MWS to passively receive and process SDMs advertised by the other service provider *agents*.

- *PrefPush*, as described in Section 4.2.3, is fundamentally similar to the *Push* approach. However, the requester *agent* is also able to provide user preferred service type to the other service provider *agents*. In this approach, the semantic type parsing process is performed by the service provider *agents*.

In our experiment, we did not include the *Hybrid* approach (described in Section 4.2.4) in the comparison because the main purpose of the *Hybrid* approach, which utilises both *Pull* and *PrefPush* concurrently, is to guarantee that the requester *agent* is still capable of performing service discovery in an MSNP environment when the other *agents* do not support the *PrefPush*-based approach. In other words, it provides a fall back mechanism.

6.3.1 Settings

The experiment was performed mainly on an iPhone4S with IEEE802.11n 2.4 GHz Wi-Fi environment. We simulated the other proximal MSNP participants by deploying a number of MSNP agent hosts on a Macbook (described in Section 6.2).

The MSNP *agent*, which has been installed in the iPhone4S, represents the service requester who is searching for a semantic service type—$cType$ provided by the other proximal MSNP participants' MSNP *agents*. The proximal MSNP participants' MSNP *agents* are of two types: *normal* and *matched*. The *normal* MSNP *agents* do not provide $cType$ services and the *matched* MSNP *agents* can provide $cType$ services. Every MSNP *agent* (including the requester and the others) provides two service description related documents—SAWSDL and OWL. The size of SAWSDL is 6KB and the size of OWL is 12KB.

The experiment consists of two tests: performance and resource costs.

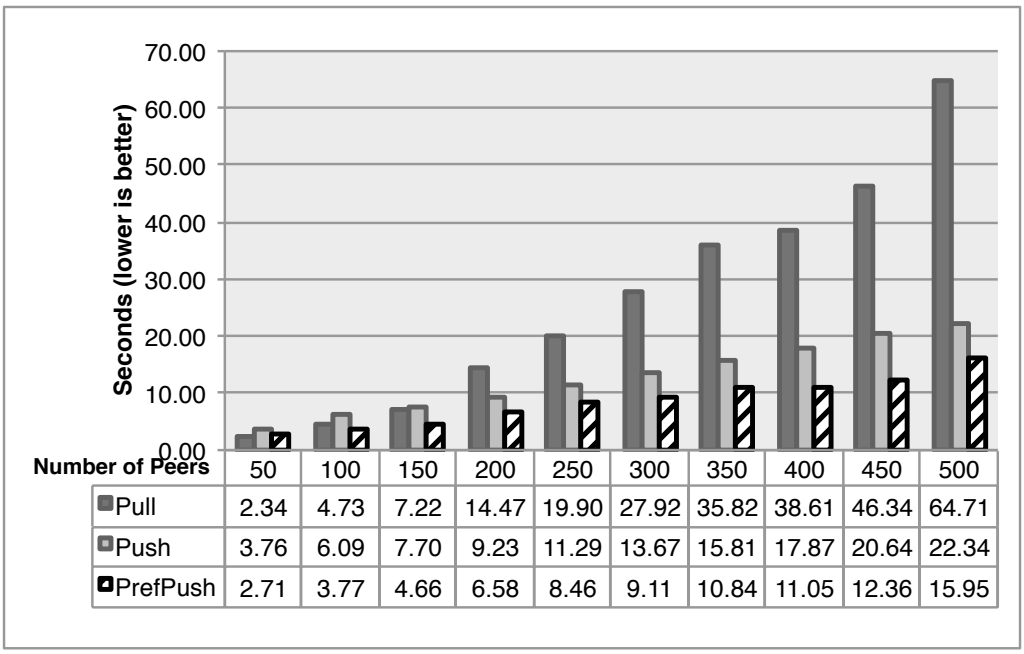
- For the performance, we aimed to compare the timespans to successfully discover matched service providers from all the deployed MSNP *agents* in the network. We deployed a different number of MSNP *agents* on the Macbook to evaluate the three different approaches: $Pull$, $Push$ and $PrefPush$.

- For the resource cost testing, we aimed to compare the CPU and RAM usages between the three approaches. It was done by recording the CPU and RAM usages while performing the service discovery processes.

Note that in the description of Section 4.2.1, we have mentioned SDM caching. SDM caching can reduce transaction overheads for all approaches equally if implemented. Since the aim of our tests is to compare the three approaches solely, we did not include the caching mechanism.

6.3.2 Performance Comparison

This section presents the experimental result of the service discovery timespan comparison among the three approaches, as shown in Figure 6.1.



Number of Peers	50	100	150	200	250	300	350	400	450	500
Pull	2.34	4.73	7.22	14.47	19.90	27.92	35.82	38.61	46.34	64.71
Push	3.76	6.09	7.70	9.23	11.29	13.67	15.81	17.87	20.64	22.34
PrefPush	2.71	3.77	4.66	6.58	8.46	9.11	10.84	11.05	12.36	15.95

Figure 6.1: Timespan Comparison

In the figure, x-axis represents the number of service provider *agents* deployed in the network. Each deployed group has 4/5 *normal agents* and 1/5 *matched agents*. For example, when 500 service provider *agents* were deployed, while 400 out of 500 were normal agent, 100 out of 500 *agents* were *matched agents* who can provide *cType* service. The y-axis represents how long it took the requester *agent* to discover the matched service providers.

The result shows that when there were only 50 service provider *agents* in the network, *Pull* provided the best performance. However, when the number of service provider *agents* increased, the performance of *Pull* worsened because of the increased amount of SDM retrieval and semantic data processing. *Push* utilised MWS to receive SDM from the other service provider *agents*. Although the requester *agent* in the *Push* approach also had to process SDM, the overall timespan was much lesser than *Pull* when the environment consisted of a large number of

service provider *agents*. Among the three approaches, our proposed $PrefPush$ approach outperformed the other two when the environment consisted of 100 or more service provider *agents*.

6.3.3 Resource Usage Comparison

This section presents the comparison of the CPU and RAM usages of the three approaches. In our setting, we deployed 1 *matched* service provider *agent* and 4 *normal* service provider *agents* every 1 second continuously for 100 seconds. We recorded both CPU usage and RAM usage within a period of 100 seconds.

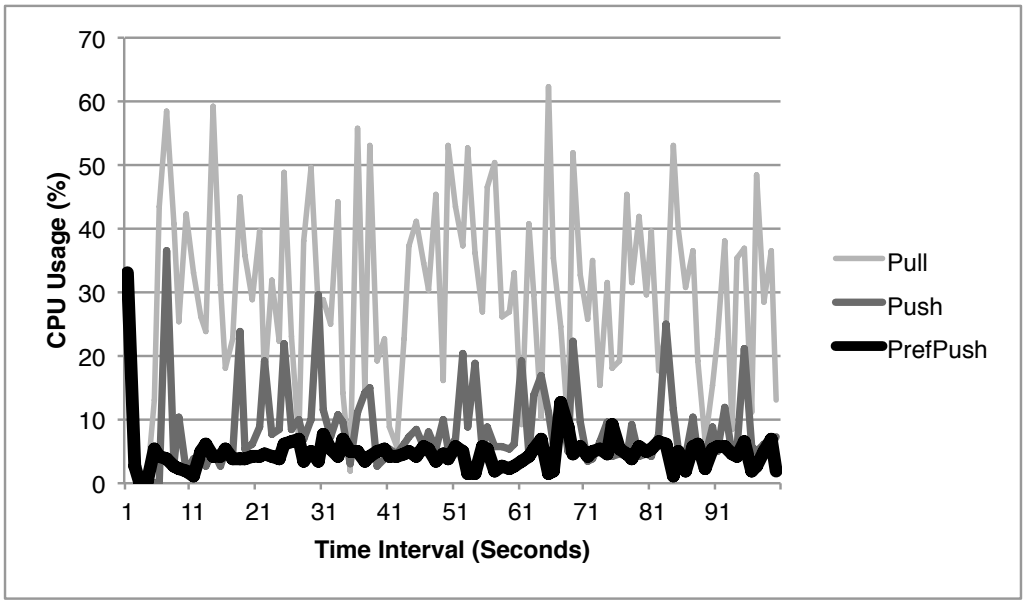


Figure 6.2: CPU Usage Comparison

Figure 6.2 illustrates the CPU usage record comparison among the three approaches. As the graph shows, $Push$ had the highest CPU usage while $PrefPush$ consumed the least CPU resource.

Figure 6.3 illustrates the RAM usage comparison among the three approaches. As the result shows, the RAM usage of the three approaches increased continuously. $Pull$ consumed the highest RAM resource, while $Push$ and $PrefPush$ have very similar RAM resource consumption.

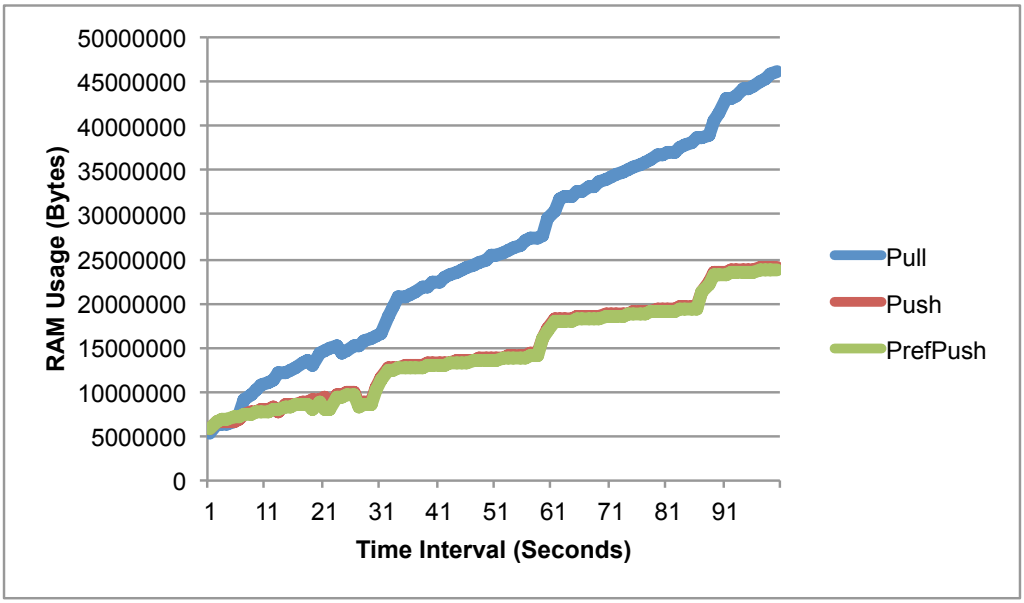


Figure 6.3: RAM Usage Comparison

In order to highlight the difference between $Push$ and $PrefPush$, we enlarged the graph to show the RAM usage between 40 to 60 seconds for $Push$ and $PrefPush$. This is shown in Figure 6.4.

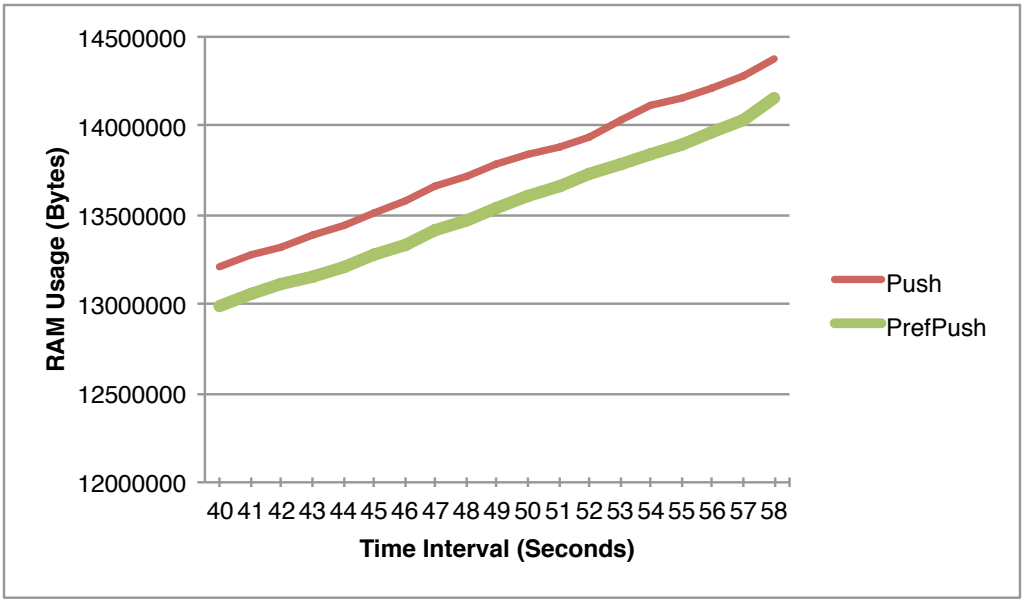


Figure 6.4: Partial RAM Usage Comparison of Push and PrefPush

In the figure, $Push$ shows a slightly higher RAM usage than $PrefPush$.

6.4 Context-Aware User Preference Prediction

The ***Predictor*** enables the context-aware user preference prediction scheme presented in Section 4.3.2 and it is the main component for enabling the proactive service discovery in MSNP. The scheme uses current environmental context information to compare with the past context information associated with the service invocation records to predict what types of services may be of interest to the user in the current environment.

The test focused on evaluating the accuracy of the prediction scheme using two different datasets: the programme generated dataset and the epSICAR-dataset[3].

6.4.1 Evaluating the Scheme on Programme Generated Dataset

In order to test the accuracy of the scheme, we have created a user query record generator to simulate user query records and the associated context information. Table 1 illustrates the basic parameters used in the record generator. We defined five types of records. Each record type describes a particular query type and five types of associated context information values denoted by CL, CT, CA, CW and CP. The record generator will randomly generate a given number of records (e.g., 100, 200, 300, etc.). Each record consists of one query type and five context values. For example, considering the setting in Table 1, the record generator will randomly select a record type from A to E. If the selected record type is A, then the query type will be Q1 and the associated context information will be CL=L1, CT=T1, CA = a random value from A1 to A5, CW = a random value from W1 to W5, CP = random value from P1 to P5. The two static values (L1 and T1) represent the contexts that will influence the user's decision to select Q1.

[3] `http://www.imada.sdu.dk/~gu/`

Record	Query	CL	CT	CA	CW	CP
TypeA	Q1	L1	T1	A1-A5	W1-W5	P1-P5
TypeB	Q2	L1-L5	T2	A2	W1-W5	P1-P5
TypeC	Q3	L1-L5	T1-T5	A3	W3	P1-P5
TypeD	Q4	L1-L5	T1-T5	A1-A5	W4	P4
TypeE	Q5	L5	T1-T5	A1-A5	W1-W5	P5

Table 6.1: Parameters for Prediction Test

Figure 6.5 illustrates the results of our evaluation using the parameter setting in Table 1. The x-axis shows the percentage records we have used as training set to predict the rest of the records. For example, the very first value on the bottom left of the graph shows the accuracy result based on a total of 100 query records, of which 60% records were used as the training set to predict the rest of the records (40%). The prediction accuracy for this is around 84%, as shown on the y-axis.

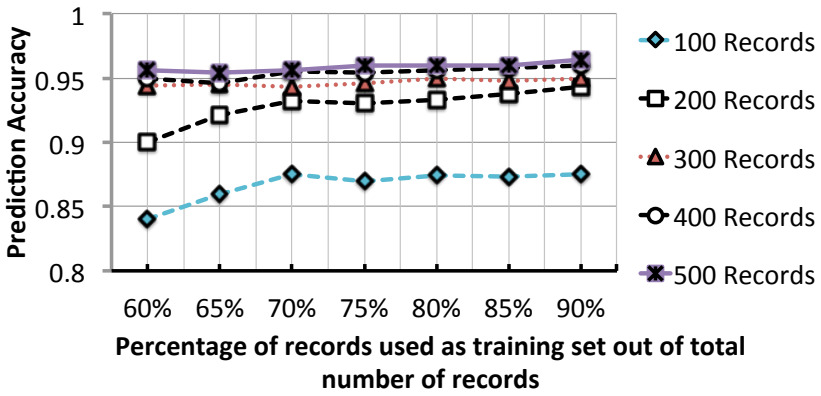


Figure 6.5: Prediction based on random dataset

6.4.2 Evaluating the Scheme on epSICAR Dataset

We have also tested our prediction scheme using a subset of epSICAR dataset. We used 200 sequence records from the dataset. Each record consists of two context attributes: location and action. Each record is also associated with corresponding object (e.g., Hi-Fi Music system for listening music in living room), which can be

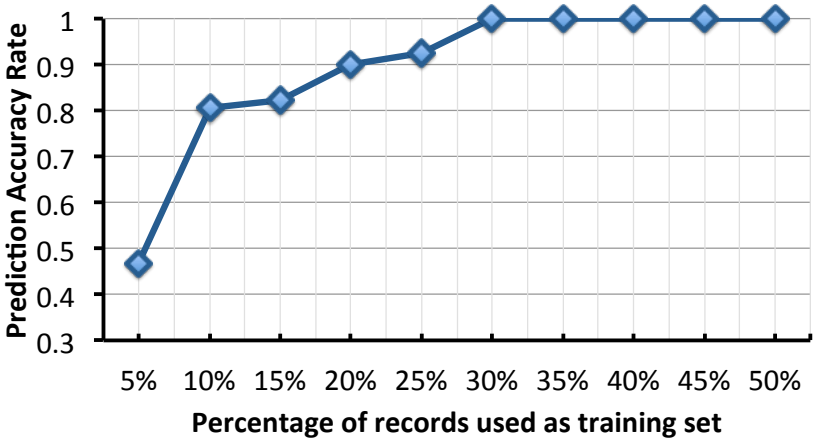


Figure 6.6: Prediction based on real dataset

considered as a service. The test result is shown in Figure 6.6. In the figure, when 30% of the records (i.e., 60 records) were used as training set to predict the rest of the records, the accuracy of prediction was close to 100% rate.

6.5 Trustworthy Service Discovery in MSNP

This section presents the evaluation of the ***Trust Evaluator*** component, which is the main component of the lightweight trustworthy service discovery scheme. The evaluation consisted of two parts:

1. We evaluated the proposed scheme described in Section 4.4.4, in which a requester intends to obtain the trust score of a provider based on the requester's friends and friend of a friend (FOAF) (i.e., recommendation based on friends and FOAF).

2. We evaluated the proposed scheme described in Section 4.4.5, in which a requester intends to obtain the trust score of a provider based on public proximal MSNP users who are non-friends of the requester (i.e. recommendation according to the public).

We describe our evaluation approach below:

(1) For each user record of a trust rating dataset, we considered the user as a *requester* in MSNP who had a set of trust rating records (denoted by *R-set*) which corresponds to the Reputation Rating Data (*RD*) in (Definition 4.9).

(2) From the *R-set*, we separated the records into two subsets: *rating of friends* and *rating of non-friends*.

(3) From the *rating of non-friends* subset, we used the proposed schemes (described in Section 4.4.4 and 4.4.5) to predict what was the *requester*'s rating for each *rating of non-friends*.

(4) We also used the basic schemes (i.e., by simply referring to the ratings from all the *rating of friends* or all the friends of the corresponding users of *rating of friends*) to predict what was the *requester*'s rating for each *rating of non-friends*. Then we compared the results between the proposed schemes with the basic schemes.

(5) Finally, we compared the data transaction costs between the proposed schemes with the basic schemes. We then applied a basic CPI model to compare the schemes.

In order to evaluate the proposed trustworthy service discovery scheme for MSNP, we have used the Advogato[4] dataset to simulate a large number of MSNP users' trust rating data.

Advogato dataset is part of the Trustlet project (Massa et al.; 2009), which collects the trust rating values of social network site users since October 13 2007. Each record in the Advogato dataset consists of:

- The ID of the person who rated another person
- The ID of the person who has been rated

[4] `http://www.trustlet.org/wiki/Advogato_dataset`

- The rating value, which has three possible levels suggested by Massa et al. (2009): Apprentice (represented by a score of 0.6), Journeyer (represented by a score of 0.8), and Master (represented by a score of 1.0).

We have tested our proposed trustworthy service discovery scheme using the Advogato dataset of 26 May, 2013. The original dataset contains many records with empty rating values (Some users have not rated any other users). Since our proposed scheme requires a fair number of rating data to calculate the trust score of a person based on other users' ratings, we have removed users who have less than 10 rating records from the original dataset.

The original Advogato dataset does not specify the relationship between users (i.e., are they real friends or not?). However, from their trust ratings, we categoried the relationship of users into two groups: when two users rated each other as 'Master' level, they are 'friends'. Otherwise, they are 'non-friends'.

The following sections present the evaluation cases and results.

6.5.1 Selecting Recommender Based on Friends and Friend Of A Friend

The aim of this test is to show that the proposed schemes (described in Section 4.4.4) require less transaction cost but still can provide similar trust score measurement result as the basic schemes.

The *basic schemes* use a simpler approach to determine a service/content provider's trustworthiness based on the reputation rating of all the requester's friends or all the requester's FOAF. They are:

- **All Friends (AF)**.

  In this scheme, the requester computes a service provider's trust score based on the average rating values of all the requester's friends who have rated the service provider.

- **All Friends of Friends as Recommended Reference (AFOAF)**

  In this scheme, the requester computes a service provider's trust score based on the average rating value of all Recommended References (RR) of the requester's friends. The RR in this scheme are simply the FOAF who have rated the service provider without additional filtering.

The *proposed schemes*, which correspond to Step 2, 3 and 4 of **Algorithm 4.1** described in Section 4.4.4 are:

- **One High Experience Friend (HEF)**

  In this scheme, the requester computes the service provider's trust score based on one single High Experienced Friend found from the requester's friends who have rated the service provider, and have largest rating records in the friends. HEF corresponds to the description in **Algorithm 4.1**, Step 2.

- **One High Experienced FOAF (HEFHEF)**

  In this scheme, the requester computes the service provider's trust score based on one single High Experienced FOAF who has rated the service provider. The High Experienced FOAF is a friend of a HEF who may not have rated to the service provider, but the HEF has one or more friends who have rated the service provider. This scheme corresponds to **Algorithm 4.1**, Step 4.

- **One Most Similar Friend (MSF)**

  In this scheme, the requester computes service provider's trust score based on one single most similar friend. This scheme corresponds to **Algorithm 4.1**, Step 3.

In this test case, we firstly retrieved a list of user IDs (as requesters) from the dataset. Each user had a list of ratings consisting of the IDs of the persons who had been rated, and the corresponding rating level value. Our test focused on predicting the requester's rating of each 'non-friends' (representing service providers who will be evaluated by the requester) based on 'friends' and 'FOAF'.

We used the above five different schemes to perform the prediction to show that the proposed scheme, which utilises the High Experience Friend's rating and the High Experience Friend's Recommended Reference person's rating (Algorithm 4.1) are efficient approach to measure the trust score of a provider.

We assumed that the requester has replicated friends' *RD* (Definition 4.10) in local memory previously. Hence, at runtime, it can identify recommenders for computing the reputation score of a service provider without retrieving all friends' *RD* directly from the friends' MWS or their cloud storages. The replicated *RD* can only be utilised to identify recommenders. In order to find out the up-to-date reputation rating score from the recommenders, the requester still has to perform the request to retrieve the necessary *RD* directly from the friends' MWS or their cloud storages. Depending on the scheme used, the required *RD*-retrieval process can be different.

Table 6.2 summaries the cases of different schemes that were used for testing and comparison. The Comparable Count in the table represents the total number of rating records that have been used to test the scheme. Because each scheme relies on different criteria, the Comparable Count differs. For example, not all the users have available friends or FOAF's ratings to predict the trust rating of a specific user. Hence, such incomparable records have been excluded in the testing for that scheme.

Scheme	**Comparable Count**	**Prediction Accuracy**	**Average Minimum Transaction Required**
Basic			
AF	1010	0.633569	6
AFOAF	1075	0.642984	36
Proposed			
HEF	1010	0.635335	1
HEFHEF	1010	0.640418	6
MSF	1010	0.579199	1

Table 6.2: Comparison of Trust Schemes' Accuracy and Transaction Costs of Friends and FOAF

The values of 'Average Minimum Transaction Required' in Table 6.2 were computed as follows:

- **AF** scheme requires up-to-date reputation rating values from all friends. The *average minimum transaction required* is equal to the average number of friends of each requester under test, in which the average number of friends each requester has is 6, which is the average number of 'Master' level ratings of each user in the Advogato dataset.

- **AFOAF** scheme requires the highest transaction cost at runtime incurred by retrieving the up-to-date reputation rating values from all FOAFs. The total cost of the required transaction was the number of friends multiplied by the number of FOAF, which is 36.

- In **HEF** scheme, since the requester has replicated the *RD* (Definition 4.10) previously, the replicated old *RD* is sufficient for the requester to identify a HEF at runtime without consuming data transaction cost on retrieving new *RD* via the Internet. Once a HEF is found, the requester only needs to retrieve the up-to-date reputation rating value from the HEF. Hence, in this case, the transaction cost is 1.

- **HEFHEF** scheme requires the minimum transaction values is 6, which is the sum of the transaction cost of retrieving RD from all friends of HEF.

- **MSF** scheme incurs the same transaction cost as the HEF-based scheme.

Figure 6.7 summarises and compares the prediction accuracy and the transaction cost of the five schemes in graphical form.

In order to highlight the overall improvement of the proposed approaches (HEF, HEFHEF, MSF) compared to the basic approaches (AF, AFOAF), we have translated the results into a cost and performance index (CPI) model. Figure 6.8 shows the cost-performance index value of each approach. As the figure shows, when direct friends are available as the recommenders of the reputation rating, the proposed HEF and MSF schemes provide better CPI values than the basic scheme—AF.

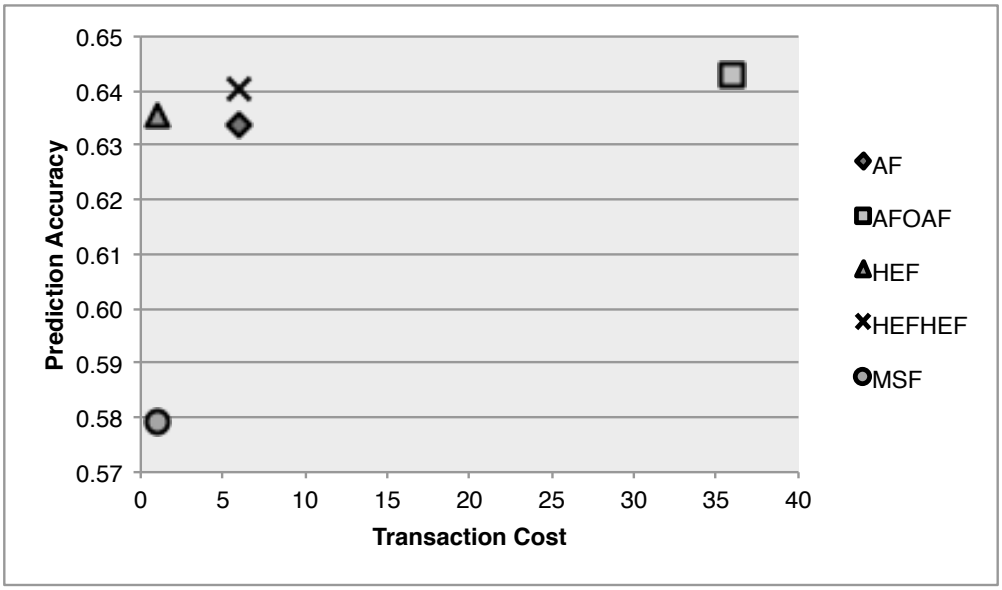


Figure 6.7: Predictive Rating Accuracy Comparison of Different Schemes based on Friends and FOAF

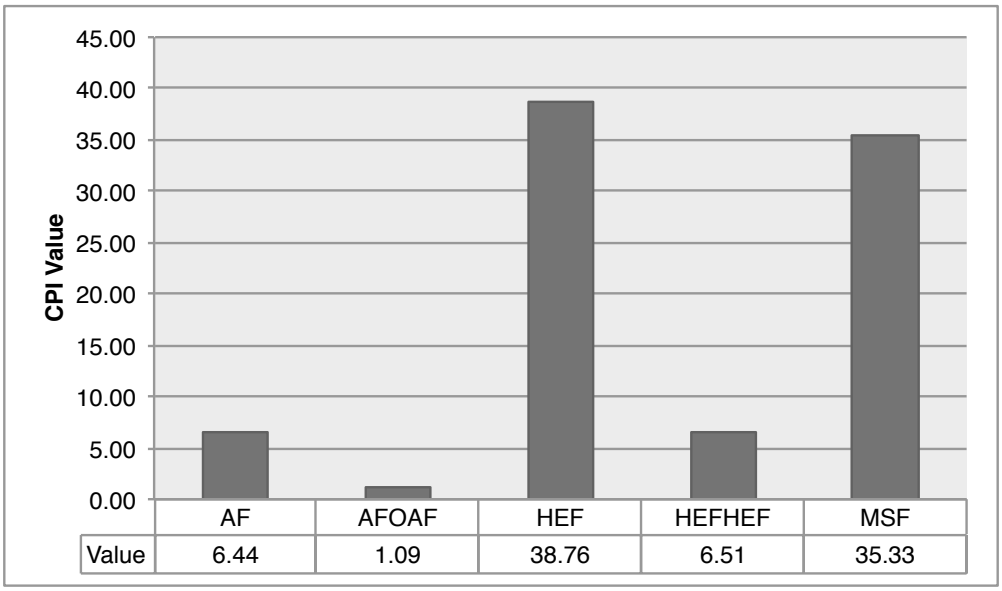


Figure 6.8: Cost and Performance Comparison of Different Schemes based on Friends and FOAF

When direct friends cannot be the recommenders, during which FOAF is needed, the proposed HEFHEF scheme gives a better CPI value than the general AFOAF scheme.

6.5.2 Selecting Recommenders Based on the Public

The test described in this section corresponds to the scheme described in **Algorithm 4.2**, in which a requester is unable to determine a service/content provider's trustworthiness based on friends or FOAF's reputation rating values. Hence, the requester will refer to proximal strangers for the reputation rating values. However, the reputation rating of random selected stranger is unreliable. Hence, we presented in Section 4.4.5 an approach to identify which strangers' reputation rating values are reliable based on the stranger's *experiences* and *credibilities*.

This test aims to show that the proposed scheme can improve the accuracy when the trustworthy service discovery process is based on public proximal MSNP participants' rating scores. Recall that in our setting, each user in the Advogato dataset has 'friends' (people who have been rated as 'Master' level) and 'non-friends' (people who have been rated as 'Apprentice' or 'Journeyer' level). In this test case, we used the 'non-friends' as the proximal strangers of the requester.

The test case compared the proposed scheme with the basic Naïve scheme. The two schemes are summarised below:

- **Naïve Scheme**

  The requester computes a service provider's trust score based on the average rating values of all the requester's 'non-friends' who have rated the service provider. The service provider is excluded from the list of 'non-friends'.

- **Proposed Scheme**

  The requester computes a service provider's trust score based on a selected recommender based on both credibility and experience computed from the 'non-friends' list. Same as the Naïve scheme, the service provider is excluded from the list of 'non-friends'.

We also included two additional schemes—Experience Only (Exp Only) and Credibility Only (Credit Only)—in which the requester selects a recommender based on only experience and based on only credibility respectively. These two schemes

were included because we wish to show that the proposed scheme (based on both credibility and experience) provides better prediction accuracy than the cases of only using one of them to predict the reputation rating value.

When referring to the ratings from the public, the average minimum transactions required were the same, because the requester had to collect all the proximal MSNP participants' rating data in order to identify their credibility and experience. The value—7 is the average number of 'non-friends' that each user had in the Advogato dataset.

In our test, we removed all the friends from the dataset. Each requester derived another user's rating score based on other user's rating values (i.e., public recommendations).

Scheme	**Comparable Count**	**Prediction Accuracy**	**Average Minimum Transaction Required**
Proposed Scheme using both Credibility and Experience	851	0.703078	7
Naïve Scheme	851	0.504942	7
Experience Only Scheme	851	0.686321	7
Credibility Only Scheme	851	0.499681	7

Table 6.3: Comparison of Trust Schemes' Accuracy and Transaction Costs of Public

Table 6.3 shows the tabulated results, and Figure 6.9 the results in graphical form.

Since the transaction cost of all schemes were the same, we did not need to calculate their cost-performance index value to compare their performance in this case. As the result shows, the accuracy of the Naïve scheme was 50%, which means that if the requester computes a provider's trust based on the average trust rating scores from all the proximal MSNP participants, it will only have a 50% chance for the result to match what the requester expects. If the requester computes the provider's

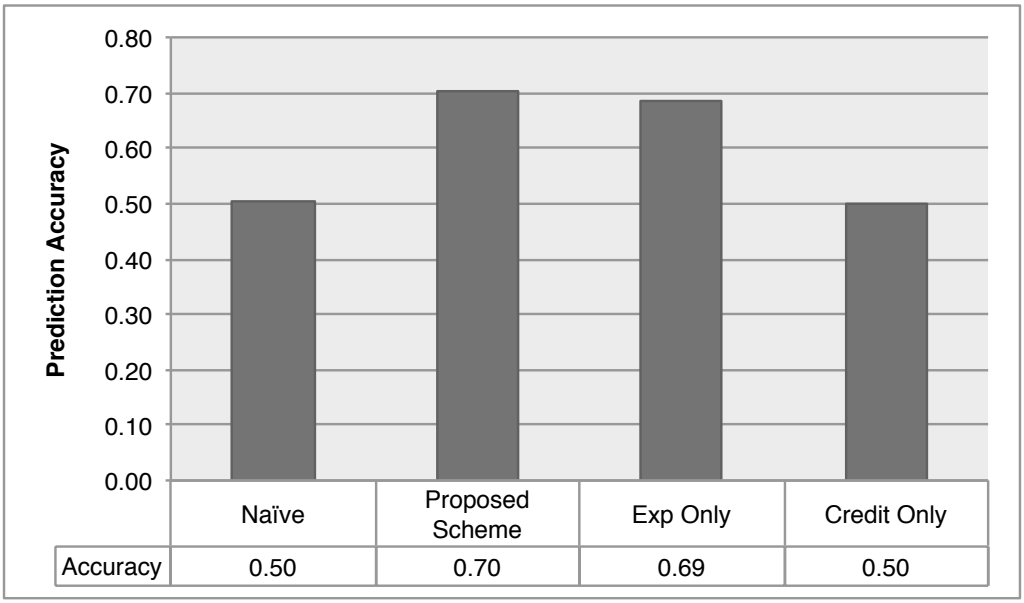


Figure 6.9: Predictive Rating Accuracy Comparison of Different Schemes based on Public

trust score based on the most experienced MSNP participant's rating (Exp Only), there is a 69% chance that the result will match what the requester expects. On the other hand, if the requester only refers to the trust score of the highest credible MSNP participants (Credit Only), there is only a 50% chance that the result can match what the requester expects. Our proposed scheme which combines experience with credibility outperforms the other schemes with a 70% chance. Overall, all these schemes perform better than the Naïve scheme in terms of accuracy.

The proposed scheme is shown to improve the accuracy when the prediction is based on public proximal MSNP participants' rating scores. However, since the rating score was computed based on strangers' ratings, the scheme was unable to reduce the transaction cost like the schemes based on friends and FOAF did. Because the requester did not have strangers' ratings pre-stored in its local memory or its cloud storage, in order to identify and compare the experience of all the proximal MSNP participants, the requester had to collect all the rating data from all the proximal participants' *agents*. Reducing the transaction cost in public-based trustworthy service discovery for MSNP requires further investigation. We consider this as one of our future research directions.

6.6 Adaptive Approach Selection based on the Cost-Performance Index Model

This section describes the evaluation of the adaptive approach selection of the proposed workflow component described in Section 5.4. The test was conducted on iPod Touch 4th generation.

This evaluation considers two case studies described in Section 6.6.1 and Section 6.6.2 respectively.

6.6.1 Service Description Metadata Prefetching

This case study was based on the scenario described in Section 5.5.1, in which a newly joined mobile device ($PeerX$) will predict its user's request query and identify a preferred semantic service type (denoted by $pType$ in this section). Then, $PeerX$ will trigger the *Service Discovery* task to prefetch the corresponding service description metadata (SDM) from other peers.

6.6.1.1 Test Setting

In the beginning of the case study, there were only 10 peers found in the network. After a few seconds, a different number of peers joined the network. Then, the *Task Agent* of $PeerX$ that handles the *service discovery* started calculating the CPI value of each applicable *approach* in order to change the system's behaviour based on cost and performance.

The aim of this case study is to show how the system changes its task *approach* dynamically at runtime based on the CPI model. The focus is on the *service discovery* task. Recall that three applicable *approaches* have been pre-defined for the *service discovery* task.

The technical implementation details of each *approach* are described below:

- ***Pull***

  PeerX launches a Web service to receive incoming SDM files, and also *PeerX* sends a simple `HTTP GET` request to each MSNP peer in the same Wi-Fi subnet. The request message contains *PeerX*'s global IP address (Mobile IPv6 address). Each MSNP peer who receives *PeerX*'s request, will send a `HTTP POST` request to *PeerX* with the peer's SDM file (in SAWSDL) in the request message body. When *PeerX* receives an SDM, it passes the SDM to the *matchmaking* component to identify whither the SDM contains a Web service operation that matches *pType* or not. If *pType* is found in the SDM, the SDM will be stored in the *Temporary Files* folder of the device.

- ***Push***

  PeerX launches a Web service to receive incoming `HTTP POST` requests from other MSNP peers. When *PeerX* receives the request, it checks whither the data is a SAWSDL file or not. If the file type is correct, the file will be sent to the *matchmaking* component and the rest is the same as the *Pull approach*.

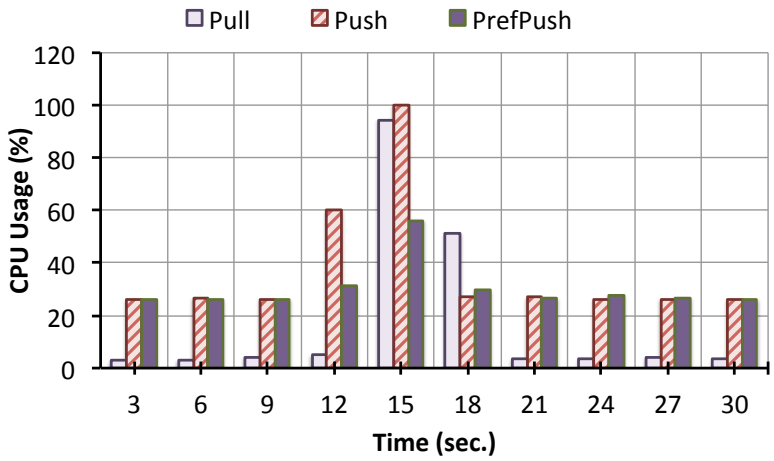
- ***PrefPush***

  PeerX predicts its user preferred service type—*pType* based on current context information and then launches a Web service to respond what type of service is of interest by its user. Based on the preferred service type, other MSNP peers can advertise their SAWSDL files to *PeerX* if they can provide the *pType* service. Concurrently, *PeerX* also provide a Web service to receive the SAWSDL files advertised by other MSNP peers.

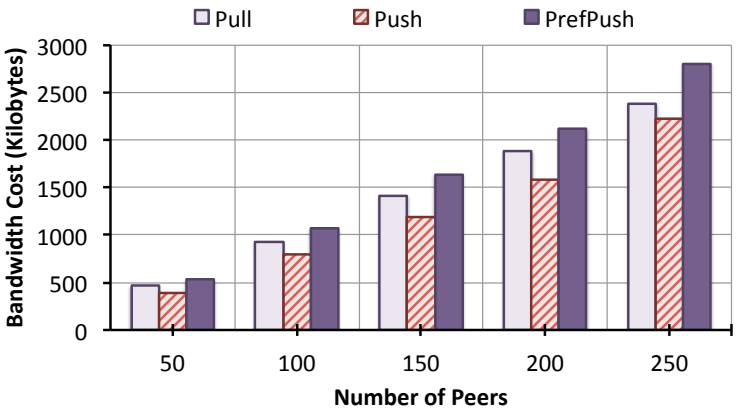
6.6.1.2 Resource Usage and Performance Records

In the case study, we considered two resource cost elements: *CPU usage* and the *bandwidth cost*. In a networked system, CPU usage and network transaction are two elements that consume the most battery-life of a mobile device. Figure 6.10

illustrates the resource cost comparison of the three *approaches* described previously.



(a) CPU usage



(b) Bandwidth cost

Figure 6.10: Cost Records

Figure 6.10(a) illustrates the *CPU usage* comparison of the three *approaches* in a 30 seconds period. In the beginning of the test, there were no new peers joining the network. At the 12 seconds mark, new peers were discovered and the service discovery *approach* was launched. The figure explicitly shows that both *Push* and *PrefPush approaches* consumed the most CPU power even though there were no new peers discovered. On the other hand, the *Pull approach* consumed very little CPU power in the first 12 seconds.

Figure 6.10(b) illustrates the *bandwidth cost* comparison. Overall, the *PrefPush approach* incurred the highest *bandwidth cost*, followed by the *Pull approach*. *Push approach* incurred the least *bandwidth cost*.

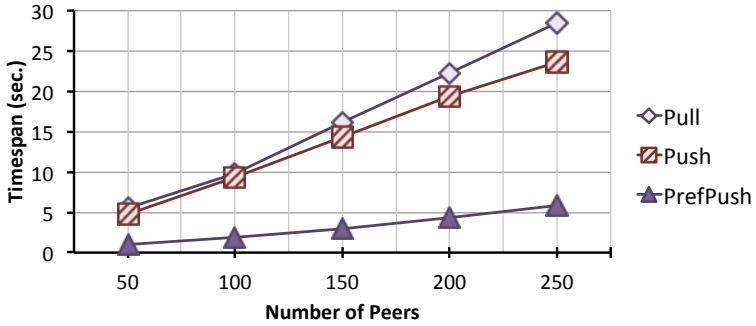


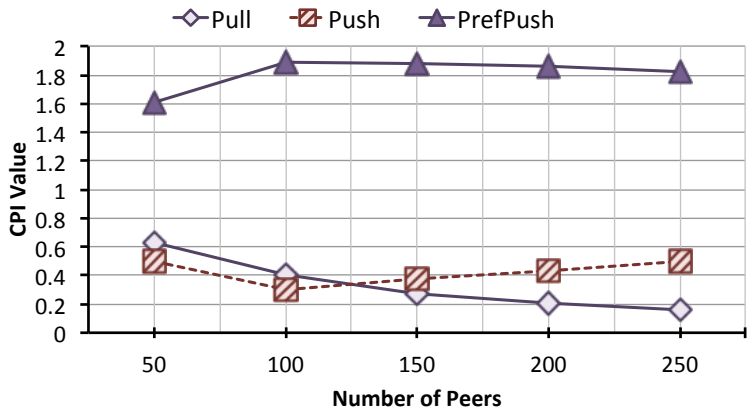
Figure 6.11: Discovery Approaches Timespan

Figure 6.11 illustrates the timespan performance of the three *approaches*. As the figure shows, *PrefPush* had the best performance. *Push* and *Pull* resulted in similar performance when the environment consisted of only a few peers. When there were more peers in the environment, *Push* performed slightly better than *Pull*.

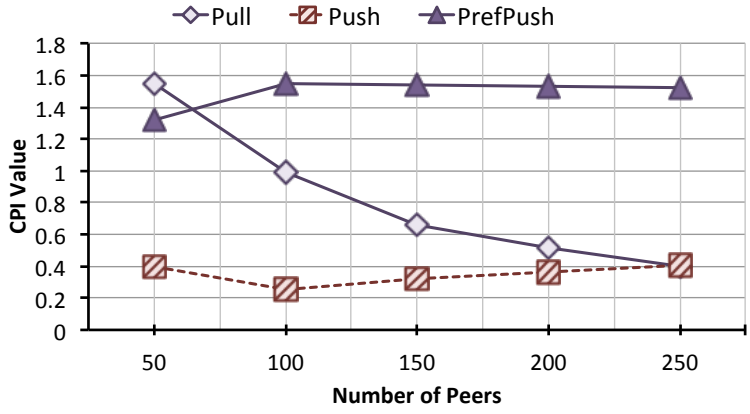
6.6.1.3 Cost-Performance Index Value Comparison

Figure 6.12(a) illustrates the original CPI values of the three *approaches*. In this instance, the weight of each *cost element* has been set equally to 1. While other MSNP peers are capable of supporting the *PrefPush approach*, *PeerX* will select *PrefPush* for its *service discovery* task. However, if *PeerX* did not receive any incoming message for *PrefPush approach* for a period of time (e.g., 5 seconds), it should consider *PrefPush* to be currently not applicable and select another *approach*. As the figure shows, when there were less than 100 new peers joining the network, the *Pull approach* performed slightly better than *Push approach*. Hence, *PeerX* should perform *Pull* in such a situation.

In the next test, we simulated the situation when the battery-life became low (we reduced the value manually). Hence, in this situation, the system will try to reduce the hardware resource usage to save battery-life. i.e., When *CPU usage* is reduced, the battery-life consumption will also reduce. In this case, the weight of



(a) CPI values with equal weight of cost element (higher is better)



(b) CPU usage while idle + 10 (higher is better)

Figure 6.12: Cost performance index testing result

CPU usage has been increased to 10 by the system, which denoted that the *Task Agent* needed to identify which approach requires less CPU power. After the *Task Agent* performed the CPI calculation, the *Pull approach* out-performed to *PrefPush* when there were only 50 new peers found in the environment. If *PrefPush* was not suitable, the *Task Agent* would choose the *Pull approach* instead, unless there were 250 new peers joining the network.

Both Figure 6.12(a) and Figure 6.12(b) show that when the number of peers increase, the CPI value of the *PrefPush* will decrease slightly, and the CPI value of *Push* will increase conversely. This is because the *Push approach* incurs much lesser *bandwidth cost* compared to the other two *approaches*, resulting in a slightly improved CPI value.

6.6.2 Content Advertising

This case study was based on the content advertising scenario described in Section 5.5.2 previously.

In this case study, each peer has a back-end cloud storage using Dropbox, and the peer's current IP address is continually synchronised with its cloud storage, and is retrievable from a static URL address using the `HTTP GET` request. Moreover, since each peer is a Web service provider, the communication does not rely on the common Web service request/response process. Instead, when two peers initiate the communication, they exchange their basic description metadata, which contains the URL information of each peer's current IP address. By doing so, a requester does not need to wait for the response when it sends out the query. Instead the request query contains a specific ID. When the provider completes the request, it invokes the requester node and sends the result (with the specific ID contained in the requester's query) to the requester.

6.6.2.1 Test Setting

In the beginning of the test case, 10 MSNP peers were found. After the workflow was executed, more peers joined the network. Then, the system performed the calculation to identify whether the *approach* should change or not, based on the CPI values of the *approaches*.

In the experiment, three cost elements have been considered: CPU usage of mobile device, network bandwidth cost of mobile device, and network bandwidth of the cloud utility service. The cloud bandwidth cost has been considered because it is one of the limitations of GAE. Note that the cost element of the cloud in this evaluation was only used to show how the system behaved according to the proposed CPI scheme. In reality, the cost of a cloud utility service could involve other factors such as instance creating platform, hardware performance, time of usage, etc.

Mobile devices have limited processing power. In this test, tasks were performed asynchronously. Our experiment involved 250 MSNP peers and the total cost of

using GAE was within its free usage plan limit. If there were more than 250 MSNP peers involved, the device will not be able to perform its tasks within an acceptable timespan. Hence, we did not consider the pure cost elements of cloud like those in Amazon EC2.

In the following test, we used the setting corresponding to the scenario described in Section 5.5.2. We summarise the setting below:

A content advertiser ($PeerX$) intended to advertise $pType$ service to other MSNP peers when it discovered new peers in the environment. $PeerX$'s workflow consists of two tasks:

- **T1**: *discover* peer who was interested in the $pType$ service.
- **T2**: *advertise* its SDM to the peers who were interested in the $pType$ service.

Task T1 can be accomplished by two selectable approaches:

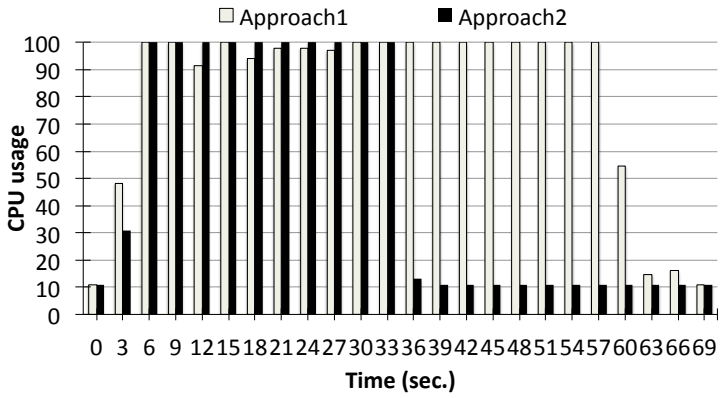
- **Approach 1**—perform all the discovery processes on mobile device.
- **Approach 2**—offload the SDM retrieval process and the semantic matchmaking process to cloud utility service.

T1 and T2 were parallel tasks and their sessions will remain until the workflow was terminated. The entire process can be set for a specific period, and it will terminate when the period expire.

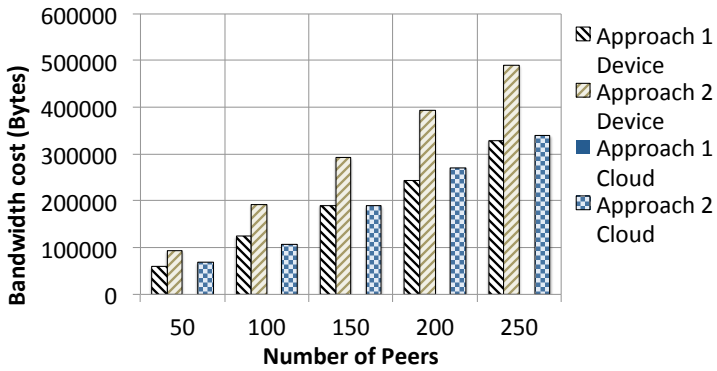
6.6.2.2 Resource Usage and Performance Records

Figure 6.13(a) illustrates the recorded CPU usage of the two *approaches*.

The figure shows that while the application was running, it consumed around 11% of the CPU usage at the 0 second mark. This is because the device was running a Web server and has joined Bonjour network, in which the device needed to continually communicate with the router to update the Bonjour service list. At the 3 second mark, the workflow has been triggered, so the CPU usage went up to 100% at the 6 second mark. For Approach 1, the CPU usage was over 90% at 51



(a) CPU usage



(b) Bandwidth cost

Figure 6.13: Cost records

seconds. On the other hand, for Approach 2, the CPU usage was over 90% at 27 seconds. The CPU usage cost element of our experiment was based on how long the CPU usage remained at over 90%. As shown in the figure, Approach 1 costs 24 more seconds than Approach 2.

Figure 6.13(b) illustrates the bandwidth cost recorded for both device-side and the cloud utility service-side for different members of MSNP peers in the network.

In the figure, 'Approach $\mathbb{N}$ Device' (where $\mathbb{N} = 1$ or 2) denotes the bandwidth cost of the mobile device in Approach $\mathbb{N}$. 'Approach $\mathbb{N}$ Cloud' denotes the bandwidth cost of the cloud utility service in Approach $\mathbb{N}$.

Since Approach 1 did not use the cloud utility service, the cost value of 'Approach 1 Cloud' was always zero.

Figure 6.14 illustrates the process timespan recorded for each *approach* affected by the number of MSNP peers. As the figure shows, with fewer number of peers,

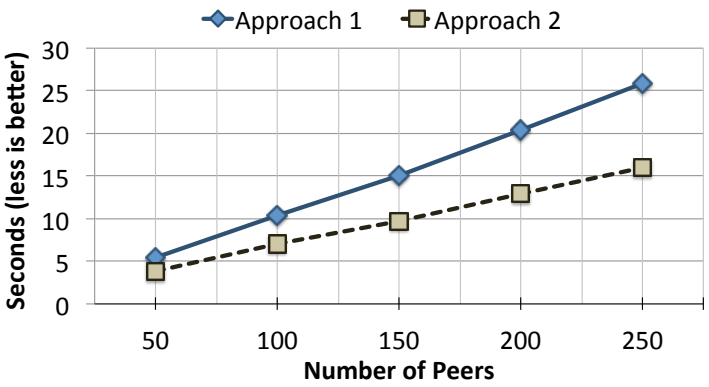
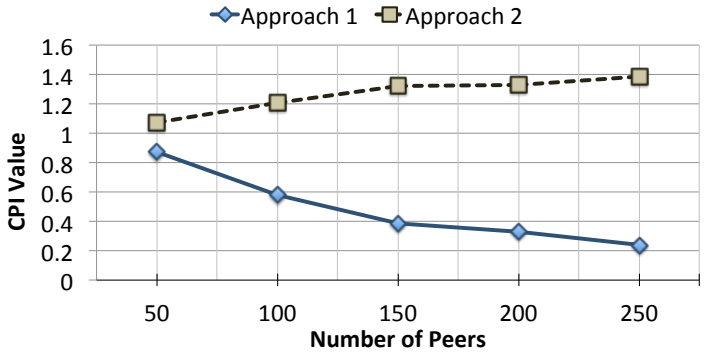


Figure 6.14: Timespan (lower is better)

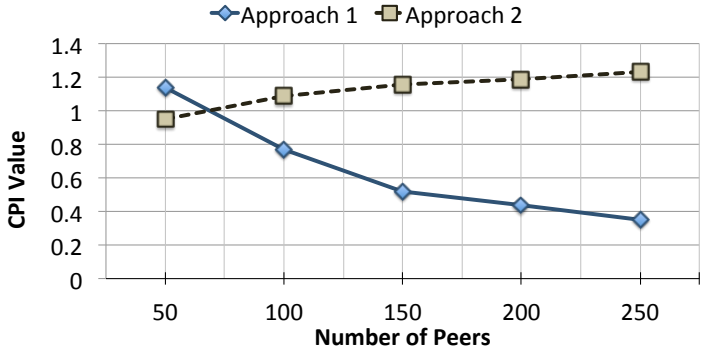
Approach 2 (which distributes the matchmaking process to the cloud) performance did not improve much.

6.6.2.3 Cost-Performance Index Value Comparison

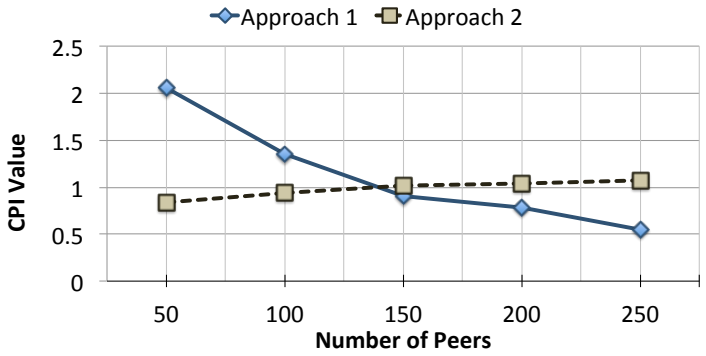
Figure 6.15(a) illustrates the CPI values of both *approaches* influenced by the number of peers. In this case, the weight of each cost element has been set equally to 1. As the number of peers increased, the CPI value of Approach 1 was reduced. In the next case, we assume that the system intended to reduce the cloud bandwidth usage because the available bandwidth for the free-of-charge period was getting low. Hence, the weight of cloud bandwidth was increased by 1. The result (Figure 6.15(b)) shows that when the number of peers was 50, the CPI value of Approach 1 was more than the value of Approach 2. Hence, the workflow remained unchanged in Approach 1. In the final case (Figure 6.15(c)), we assumed that the available bandwidth of free-of-charge period was nearly reaching the end. Hence, the weight of the cloud bandwidth was increased by 5. In this case, the workflow engine only selected Approach 2 when there were 150 or more peers.



(a) CPI values with equal weight of cost element (higher is better)



(b) cloud bandwidth weight + 1 (higher is better)



(c) cloud bandwidth weight + 5 (higher is better)

Figure 6.15: Cost performance index testing result

6.7 Summary

This chapter presents the prototype evaluation of the proposed schemes for enabling service-oriented MSNP. In order to provide the proof-of-concept of the proposed schemes, the corresponding component of each scheme has been implemented and evaluated individually.

We have implemented the mobile Web service (MWS) provider and client components. The components support MP2P service publication and discovery mechanisms to realise a decentralised service-oriented MSNP. The MWS components have been tested on real mobile devices—iPhone4S and iPod Touch 4th generation.

We summaries the evaluation results below:

Proactive service discovery for MSNP. We have implemented and tested the components of the proactive service discovery mechanism for reducing service discovery latency in service-oriented MSNP. The two components are:

- **Predictor** enables the context-aware user preference associated proactive service discovery scheme. The component has been evaluated using two different datasets—random-generated dataset and epSICAR-dataset by applying different percentage of data from the dataset (i.e. training set) to predict the rest of the data in the dataset. The result indicates that our proposed scheme provides the best performance when the training set contains over 300 records.
- **Trust Evaluator** enables the lightweight trustworthy service discovery scheme. We have tested the reputation-based trust scheme on Advogato dataset using two test cases:
  - The reputation of the service/content provider computed based on friends and FOAF; and
  - The reputation of the service/content provider computed based on public proximal MSNP participants (strangers).

  The test results show that the proposed schemes for predicting the reputation of the service/content provider computed based on friends and friend of a friend (FOAF) can reduce the overall transaction cost and are equally reliable to the basic schemes which require large number of reputation rating data. Furthermore, although the proposed scheme for predicting the reputation of the service/content provider based on public

proximal MSNP participants does not reduce the transaction cost, it can improve the chance of finding the reliable recommenders for retrieving the reputation ratings of content/service providers. As mentioned in Section 6.5.2, the solution to reduce transaction cost in public-based trustworthy service discovery for MSNP will be considered as our future research direction.

In order to show that the proactive service discovery for MSNP scheme can improve the performance of service discovery, we have installed the prototype on iPhone4S and deployed different number of MSNP agents on a laptop computer to simulate the MSNP environment. We compared three MWS discovery models—*Pull*-based, *Push*-based and the proposed user preference-associated push-based ($PrefPush$) model. The result shows that the $PrefPush$-based service discovery model can effectively reduce the service discovery timespan in service-oriented MSNP environment.

Resource-aware task reconfiguration. We have implemented and tested the workflow-based AMSNP framework consisting of the resource-aware workflow task reconfiguration component to adapt to the dynamic environmental and resource changes in the MSNP environments. The framework was evaluated on an iPod Touch 4th generation using two case studies—the service discovery metadata prefetching scenario and the service advertisement scenario. The evaluation results have shown that by applying the proposed resource-aware approach selection scheme together with the ESB-based architecture, AMSNP is capable of reconfiguring the approaches of workflow tasks at runtime based on resource and performance.

Chapter 7

Conclusion and Future Research Direction

7.1 Research Contributions and Findings

Service-oriented MSNP provides a loosely coupled platform-independent environment that enables proximal-based mobile social network devices and applications operating on heterogeneous platforms to seamlessly communicate using standard Web service technologies in MP2P network environments.

In order to realise MP2P-based service-oriented MSNP, this thesis firstly introduced a service-oriented MSNP architecture in Chapter 3. The proposed architecture resolves the fundamental challenges of establishing MP2P-based MSNP communications by utilising cloud services. Using backend cloud storage services, MSNP applications can maintain their communications without being affected by the dynamic nature of mobile IP addresses.

One drawback of utilising Web service standards to enable service-oriented MSNP is the service discovery latency caused by the high makespan in processing large number of service discovery-related data such as SAWSDL, XML Schema and OWL in MSNP. In order to overcome the latency issue in the service discovery phase, the

project introduced the context-aware user preference associated proactive service discovery scheme. Based on the user's current context information, the scheme utilises the user's past service interaction records, which consists of the corresponding semantic service type and the context information recorded when the past interactions were performed, to predict what type of service is of interest to the user. The scheme has been implemented as a component of a mediation framework—AMSNP. The component has been evaluated using two methods. In the first evaluation method, the context-aware user preference prediction mechanism has been tested using a random-generated dataset and a real user activity record dataset—epSICAR. The result shows that when the user has over 300 context associated service interaction records available in the system, the prediction can reach 95% accuracy.

Apart from proactive service discovery, to identify trustworthiness of service providers also affects the speed of service discovery. To reduce the timespan of trustworthy service discovery, this project has produced a lightweight, social reputation-based trustworthy service discovery scheme. The proposed lightweight trustworthy service discovery scheme consists of two parts. The first part of the scheme explains how to utilise the reputation rating data derived from friends and friend of a friend to compute/predict a service provider's trustworthiness score. The scheme computes a service provider's trustworthiness score based on selective friends or friend of a friend instead of referencing the reputation rating based on all available friends or friend of a friend. Hence, it minimises the overall required data transaction costs incurred by retrieving the reputation rating data. The evaluation result has shown that the scheme not only maintains the same level of accuracy of trustworthiness prediction but also reduces the makespan of the overall process.

The trustworthy service discovery also aims to support the need in which the requester does not have the reputation rating data from friends or friend of a friend to compute a content/service provider's trustworthiness. The scheme utilises the credibility information and experience information of proximal MSNP participants (strangers) to identify a reliable recommender to provide the reputation rating data

for the requester to compute the trustworthiness of a content/service provider. Although the scheme is unable to reduce the overall process makespan because it requires the requester to collect the reputation rating data from all the proximal MSNP participants, it has improved the prediction accuracy in identifying a service provider's trustworthiness.

To address the resource constraint issue, the project has applied the task offloading mechanism of mobile cloud computing technology to support long-live MSNP activities. However, due to the dynamic nature of MSNP environments, static task configuration is not always applicable. For example, when the environment consists of only a few proximal MSNP participants, a content advertising *agent* may retain its task operation on the mobile device. However, when the number of proximal MSNP participants increases, or when the hardware resource availability of the mobile device is running low, some tasks should then be offloaded to the cloud.

In order to support such a dynamic task reconfiguration, two mechanisms were proposed: a resource reconfiguration component to dynamically reallocate components used by the task on-the-fly, and, a decision maker component which can inform the resource reconfiguration component what resources should be used for the task. To realise these requirements, we have proposed the AMSNP framework, a workflow-based middleware built based on the ESB architecture design with the resource-aware adaptive task reconfiguration mechanism.

While the framework enables long-live proactive service discovery for user's tasks in MSNP environments, the decision maker component lets the workflow engine of AMSNP identify the best approach for each task based on a composite computation model, combining fuzzy set and cost-performance index. The ESB architecture-based AMSNP hosted on the mobile device, allows an MSNP *agent* to dynamically reconfigure internal and external resources and components at runtime in order to adapt to the dynamic changes of environment when it performs MSNP activities. The AMSNP prototype has been implemented and tested on a real mobile device.

In summary, this thesis project has accomplished the following efforts:

- To investigate, develop and validate approaches to overcome the latency-related and trust-related problems of the autonomous service discovery in service-oriented MSNP operated in dynamic public MP2P environment. The corresponding context-aware user preference associated proactive service discovery scheme has been proposed, implemented and evaluated together with the lightweight trustworthy service discovery scheme that are capable of reducing the overall service discovery makespan and providing the low-cost trustworthy service discovery in MSNP.

- To investigate, develop and validate an approach to resolve the resource management problems of service-oriented MSNP operating in a dynamic public MP2P environment. A corresponding resource-aware task reconfigurable workflow system has been proposed, implemented and evaluated to reflect dynamic changes in public MP2P-based MSNP environments.

- To design, develop and validate an adaptive mediation framework and its programming interface, in order to leverage and manage the autonomous service discovery mechanism for MSNP and its associated resources catered for dynamic changes in the MSNP environments. A corresponding ESB-based middleware—AMSNP has been proposed. The prototype has been implemented and tested in which the prototype is capable of reconfiguring resources at runtime based on the workflow engine's instructions.

Table 7.1 compares our AMSNP framework with the existing related frameworks in the literatures described in Section 3.4.

Recall that the columns in Table 7.1 represent the following elements:

- *Architecture* (**Archi.**) represents the base model of the framework. The three basic PBMSN models are: client-server, decentralised (DC) and semi-decentralised (Semi-DC).

- *Proximal Discovery* (**PD**) is the means by which participants discover one another in their proximity.

- *Auto Match* (**AM**) denotes how the system enables autonomous discovery and filtering based on user profile or context.
- *Trust* (**Tru.**) specifies whether the system support trust control.
- *Reducing Latency* (**RL**) represents whether the system provides a strategy to reduce latency in the bootstrap and discovery phase.
- *Resource-Aware* (**RA**) denotes whether the system supports a scheme to adapt dynamic changes at runtime to effectively select the most appropriate approach for social network activities.
- *Loose coupling* (**LC**) denotes whether the system supports loosely coupled interoperability for heterogeneous mobile devices and applications.

The table shows that our service-oriented MSNP solution—AMSNP is able to address issues that were not covered by existing proximal-based MSN frameworks. These issues are Trust, Reducing Latency, Resource-Awareness and Loose Coupling.

Work	**Archi.**	**PD**	**AM**	**Tru.**	**RL**	**RA**	**LC**
MobiSoC (Borcea et al.; 2007)	Client-server	Centralised Eventing	User Profile and Location	No	No	No	SOAP
MobilisGroups (Lubke et al.; 2011)	Client-server	Centralised	Manual Location Profile	No	No	No	XMPP
Smart Campus Project (Yu et al.; 2011)	Client-server + Minor DC	Bluetooth	Manual	No	No	No	No (OSGi)
SPN (Yang et al.; 2008)	Client-server + Minor DC	Bluetooth	Manual	No	No	No	No
Jini-based MSN Project (Brooker et al.; 2010)	Semi-DC	Jini	Manual	No	No	No	No
SocioNet (Pernek and Hummel; 2009)	Semi-DC	Bluetooth	FOAF Profile	No	No	No	Web Service
MobiSoft (Kern et al.; 2006)	Semi-DC	JXTA	FOAF RDF	No	No	No	No (Tracy2 + JXTA)
MoSoSo (Tsai et al.; 2009)	Semi-DC	JXTA	Manual	No	No	No	No
Spider Web (Sapuppo; 2010)	Semi-DC	Bluetooth	Manual	No	No	No	No
Proximiter (Xing et al.; 2009)	DC	OLSR	Manual	No	No	No	No
Mobi Clique (Pietiläinen et al.; 2009)	DC	Bluetooth	Social Profile	No	No	No	No
Cloud Semantic MSN (Rana et al.; 2010)	DC	Mobile Agent	Semantic	No	No	No	No
Yarta (Toninelli et al.; 2011)	DC	SLP	Semantic	No	No	No	RDF
AMSNP	**DC**	**Bonjour**	**Semantic**	**Yes**	**Yes**	**Yes**	**RESTful Web Service**

Table 7.1: Comparison between AMSNP and other proximal based MSN frameworks

7.2 Future Research Direction

During this research project, we have identified several subjects for future research directions.

Social-aware service discovery for MSNP—Our context-aware user preference associated proactive service discovery for MSNP scheme requires a fair amount of context information associated service discovery and interaction records in order to predict the user preferred service type. However, for a user who has none or a few records, the scheme is unable to predict the user preferred service type regarding to the context information of the user's current environment. One possible solution is to utilise the context associated service interaction records from the user's social groups such as friends or friend of a friend based on some similarity measurement between the user and his/her friend's profiles. However, a proactive service discovery scheme that relies on social-driven information will incur additional data retrieval cost at runtime, which can increase the overall service discovery makespan. A proper solution to overcome such an issue requires further investigation.

A lightweight trustworthy service discovery for public MSNP—The proposed scheme for lightweight trustworthy service discovery is applicable when the user has a fair amount of reputation rating data from friends or friend of a friend. If the reputation rating data comes from the public, it is inevitable that the requester has to retrieve all the available reputation rating data from the available proximal MSNP participants. The transaction cost can also be high when the environment consists of a large number of proximal MSNP participants.

Mashup in mobile grid computing—It is ideal to have a mobile-hosted framework that can support mobile content advertiser to dynamically generate WS-BPEL metadata for content mashup and disseminate the metadata to other MSNP participants who may be interested in the content. For the content

receiver side, further research need to be conducted to analyse and propose a feasible solution to support trustworthiness, so that the receiver's mobile-host will not accidentally execute a WS-BPEL metadata sent by a malicious peer.

Appendix A

Business Process Modelling Notations

Following notations were used in this thesis. The notations follow the Business Process Model and Notation (BPMN) Version 2.0 standard[1].

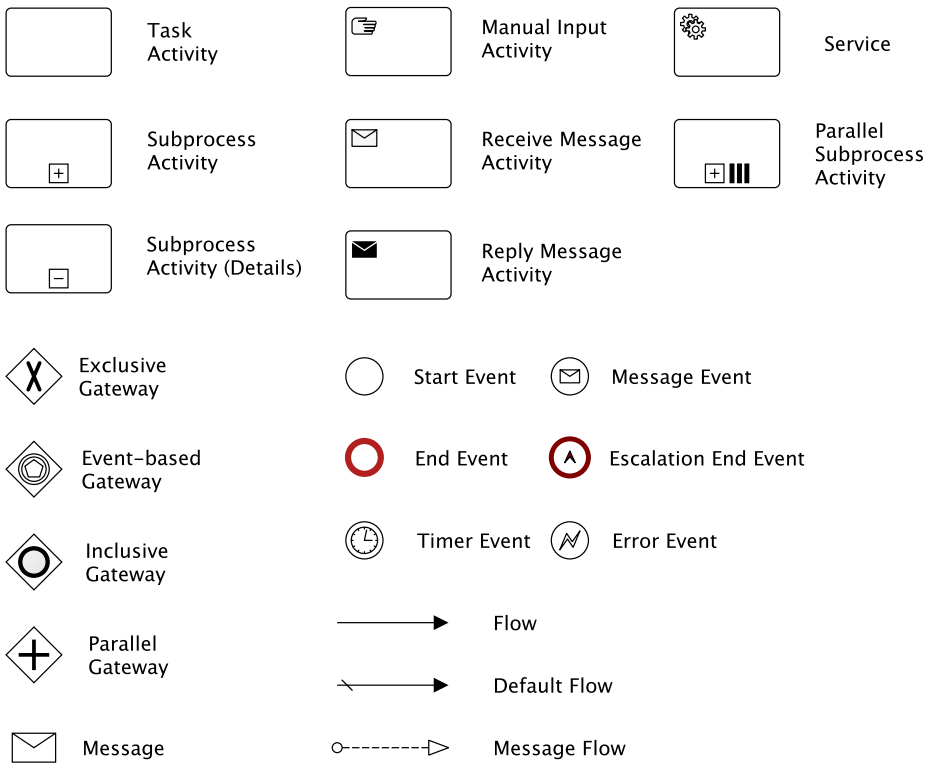


Figure A.1: BPM notations used in the thesis

[1] http://www.omg.org/spec/BPMN/2.0/

References

Aikebaier, A., Enokido, T. and Takizawa, M. (2012). Trustworthy Group Formation Algorithm Based on Decentralized Trust Management in Distributed Systems, *Proceedings of the 15th International Conference on Network-Based Information Systems*, IEEE Computer Society, pp. 58–65.

Akkiraju, R., Farrell, J., Miller, J., Nagarajan, M., Schmidt, M., Sheth, A. and Verma, K. (2005). Web Service Semantics: WSDL-S, *W3C Member Submission* . **URL:** `http://www.w3.org/Submission/WSDL-S/`

Akkiraju, R. and Sapkota, B. (2007). Semantic Annotations for WSDL and XML Schema - Usage Guide, *W3C Working Group Note* . **URL:** `http://www.w3.org/TR/sawsdl-guide/`

Allen, S. M., Colombo, G. and Whitaker, R. M. (2010). Uttering: Social Microblogging without the Internet, *Proceedings of the Second International Workshop on Mobile Opportunistic Networking*, MobiOpp '10, ACM, New York, NY, USA, pp. 58–64.

AlShahwan, F. and Moessner, K. (2010). Providing SOAP Web Services and RESTful Web Services from Mobile Hosts, *Proceedings of the 2010 Fifth International Conference on Internet and Web Applications and Services*, ICIW '10, IEEE Computer Society, Washington, DC, USA, pp. 174–179.

Amoretti, M., Laghi, M., Zanichelli, F. and Conte, G. (2008). JXTA-SOAP: Implementing Service-Oriented Ubiquitous Computing Platforms for Ambient Assisted Living, *Ambient Intelligence* pp. 75–90.

Asif, M., Majumdar, S. and Dragnea, R. (2008). Partitioning the WS Execution Environment for Hosting Mobile Web Services, *Proceedings of the 2008 IEEE International Conference on Services Computing*, Vol. 2 of *SCC '08*, IEEE, pp. 315–322.

Au Yeung, C.-m., Liccardi, I., Lu, K., Seneviratne, O. and Berners-Lee, T. (2009). Decentralization: The Future of Online Social Networking, *W3C Workshop on the Future of Social Networking Position Papers*, W3C.

Avanes, A. and Freytag, J.-C. (2008). Adaptive Workflow Scheduling Under Resource Allocation Constraints and Network Dynamics, *Proceedings of the VLDB Endowment* **1**(2): 1631–1637.

Banerjee, N., Chakraborty, D., Dasgupta, K., Mittal, S., Nagar, S. and Saguna (2009). R-U-In? - Exploiting Rich Presence and Converged Communications for

Next-Generation Activity-Oriented Social Networking, *Proceedings of the Tenth International Conference on Mobile Data Management: Systems, Services and Middleware*, MDM '09, pp. 222 –231.

Battle, S., Bernstein, A., Boley, H., Grosof, B., Gruninger, M., Hull, R., Kifer, M., Martin, D., McIlraith, S., McGuinness, D. et al. (2005). Semantic Web Services Language (SWSL), *W3C Member Submission* **9**.
URL: *http://www.w3.org/Submission/SWSF-SWSL/*

Berners-Lee, T., Hendler, J. and Lassila, O. (2001). The Semantic Web, *Scientific American* **284**(5): 28–37.

Bogatin, D. (2006). Google CEO's new paradigm: 'cloud computing and advertising go hand-in-hand', Online [Last view: 24 June 2013].
URL: *http://www.zdnet.com/blog/micro-markets/google-ceos-new-paradigm-cloud-computing-and-advertising-go-hand-in-hand/369*

Boldrini, C., Conti, M., Delmastro, F. and Passarella, A. (2010). Context- and Social-Aware Middleware for Opportunistic Networks, *Journal of Network and Computer Applications* **33**(5): 525–541.

Booth, D., Haas, H., McCabe, F., Newcomer, E., Champion, M., Ferris, C. and Orchard, D. (2004). Web Services Architecture, *W3C Recommendation* .
URL: *http://www.w3.org/TR/ws-arch/*

Borcea, C., Gupta, A., Kalra, A., Jones, Q. and Iftode, L. (2007). The MobiSoC Middleware for Mobile Social Computing: Challenges, Design, and Early Experiences, *Proceedings of the 1st international conference on MOBILe Wireless MiddleWARE, Operating Systems, and Applications*, Vol. 27, Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering (ICST), pp. 27:1–27:8.

Brooker, D., Carey, T. and Warren, I. (2010). Middleware for Social Networking on Mobile Devices, *Proceedings of the 2010 21st Australian Software Engineering Conference*, ASWEC '10, IEEE Computer Society, Washington, DC, USA, pp. 202–211.

Bruijn, J., Lausen, H., Polleres, A. and Fensel, D. (2006). The Web Service Modeling Language WSML: An Overview, *in* Y. Sure and J. Domingue (eds), *The Semantic Web: Research and Applications*, Vol. 4011 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp. 590–604.

Buchegger, S., Schiöberg, D., Vu, L.-H. and Datta, A. (2009). PeerSoN: P2P social networking: early experiences and insights, *Proceedings of the 2nd ACM EuroSys Workshop on Social Network Systems*, ACM, pp. 46–52.

Bürklen, S., Marrón, P. J., Rothermel, K. and Pfahl, T. (2006). Hoarding Location-based Data using Clustering, *Proceedings of the 4th ACM International Workshop on Mobility Management and Wireless Access*, MobiWac '06, ACM, New York, NY, USA, pp. 164–171.

Carbunar, B., Rahman, M., Rishe, N. and Ballesteros, J. (2012). Private Location Centric Profiles for GeoSocial Networks, *Proceedings of the 20th International Conference on Advances in Geographic Information Systems*, ACM, pp. 458–461.

Chang, C., Ling, S. and Krishnaswamy, S. (2010). *Research Challenges in Mobile Web Services*, Enabling Context-Aware Web Services Enabling Context-Aware Web Services Methods, Architectures, and Technologies, Chapman and Hall/CRC, chapter 18, pp. 495–519.

Chang, C., Ling, S. and Krishnaswamy, S. (2011). ProMWS: Proactive Mobile Web Service Provision using Context-Awareness, *Proceedings of the 2011 IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*, IEEE, pp. 69–74.

Chang, C., Srirama, S. N., Krishnaswamy, S. and Ling, S. (2013). Proactive Web Service Discovery for Mobile Social Network in Proximity, *Journal of Next Generation Information Technology* **4**(2): 100–112. Available online. [Last viewed: 30 June, 2013].
URL: `http://www.aicit.org/JNIT/ppl/JNIT132PPL.pdf`

Chang, C., Srirama, S. N. and Ling, S. (2012). An Adaptive Mediation Framework for Mobile P2P Social Content Sharing, *in* C. Liu, H. Ludwig, F. Toumani and Q. Yu (eds), *Proceedings of the 10th International Conference on Service Oriented Computing (ICSOC 2012)*, Lecture Notes in Computer Science. Service-Oriented Computing, Springer, pp. 374–388.

Chang, C., Srirama, S. N. and Ling, S. (2013). Towards an Adaptive Mediation Framework for Mobile Social Network in Proximity, *Pervasive and Mobile Computing* . Elsevier (In Press) Available online:.
URL: `http://dx.doi.org/10.1016/j.pmcj.2013.02.004`

Chatti, M., Srirama, S., Kensche, D. and Cao, Y. (2006). Mobile Web Services for Collaborative Learning, *Proceedings of the Fourth IEEE International Workshop on Wireless, Mobile and Ubiquitous Technology in Education*, WMUTE'06, IEEE, pp. 129–133.

Chen, A. (2005). Context-Aware Collaborative Filtering System: Predicting the User's Preferences in Ubiquitous Computing, *CHI '05 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '05, ACM, New York, NY, USA, pp. 1110–1111.

Chen, G., Kotz, D. et al. (2000). A Survey of Context-Aware Mobile Computing Research, *Technical report*, Technical Report TR2000-381, Dept. of Computer Science, Dartmouth College.

Choi, I. S., Lee, H. G. and Cho, G. H. (2005). Enhancing of the Prefetching Prediction for Context-Aware Mobile Information Services, *Proceedings of the First international conference on Mobile Ad-hoc and Sensor Networks*, MSN'05, Springer-Verlag, Berlin, Heidelberg, pp. 1081–1087.

Clema, J. K. and Fynewever, M. (1973). The Dynamic Re-evaluation of Alternatives and the Emulation of Human Decision Making, *SIGSIM Simul. Dig.* **4**(2): 18–21.

comScore (2012). comScore Introduces Mobile Metrix 2.0, Revealing that Social Media Brands Experience Heavy Engagement on Smartphones, Online. [Last viewed: 28 June, 2013].
URL: *http://www.comscore.com/Insights/Press_Releases/2012/5/Introducing_Mobile_Metrix_2_Insight_into_Mobile_Behavior*

Conti, M. and Giordano, S. (2007). Multihop ad hoc networking: The reality, *Communications Magazine, IEEE* **45**(4): 88 –95.

Datta, A. (2010). SoJa: Collaborative Reference Management using a Decentralized Social Information System, *Proceedings of the 6th International Conference on Collaborative Computing: Networking, Applications and Worksharing*, IEEE, pp. 1 –9.

Del Prete, L. and Capra, L. (2010). diffeRS: A Mobile Recommender Service, *Proceedings of the Eleventh International Conference on Mobile Data Management*, pp. 21 –26.

Delir Haghighi, P., Krishnaswamy, S., Zaslavsky, A. and Gaber, M. M. (2008). Reasoning about Context in Uncertain Pervasive Computing Environments, *Proceedings of the 3rd European Conference on Smart Sensing and Context*, Springer-Verlag, pp. 112–125.

Dey, A. K. (2001). Understanding and Using Context, *Personal Ubiquitous Comput.* **5**(1): 4–7.

Dinh, H. T., Lee, C., Niyato, D. and Wang, P. (2011). A Survey of Mobile Cloud Computing: Architecture, Applications, and Approaches, *Wireless Communications and Mobile Computing* . Available online. [Last viewed: 26 June, 2013].
URL: *http://unix.cs.plattsburgh.edu:8008/~iaydi001/COURSES/12S/12S-CSC-485-A/READINGS/mobileCloudComputing-WCMC2011.pdf*

Dorn, C. and Dustdar, S. (2007). Sharing Hierarchical Context for Mobile Web Services, *Distributed and Parallel Databases* **21**(1): 85–111.

Doulkeridis, C. and Vazirgiannis, M. (2008). CASD: Management of a context-aware service directory, *Pervasive and Mobile Computing* **4**(5): 737–754.

Doulkeridis, C., Zafeiris, V., Nørvåg, K., Vazirgiannis, M. and Giakoumakis, E. A. (2007). Context-based Caching and Routing for P2P Web Service Discovery, *Distrib. Parallel Databases* **21**(1): 59–84.

Drakatos, S., Pissinou, N., Makki, K. and Douligeris, C. (2009). A Future Location-Aware Replacement Policy for the Cache Management at the Mobile Terminal, *Wirel. Commun. Mob. Comput.* **9**(5): 607–629.

Drew, S. and Liang, B. (2004). Mobility-Aware Web Prefetching over Heterogeneous Wireless Networks, *the 15th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, Vol. 1 of *PIMRC 2004*, IEEE, pp. 687 – 691.

Elgazzar, K., Hassanein, H. and Martin, P. (2011). Effective Web Service Discovery in Mobile Environments, *Proceedings of the 2011 IEEE 36th Conference on Local Computer Networks*, LCN '11, IEEE Computer Society, Washington, DC, USA, pp. 697–705.

Farrell, J. and Lausen, H. (2007). Semantic Annotations for WSDL and XML Schema, *W3C Recommendation* **28**.
URL: `http://www.w3.org/TR/sawsdl/`

Feng, W., Zhang, L., Jin, B. and Fan, Z. (2006). Context-Aware Caching for Wireless Internet Applications, *e-Business Engineering, 2006. ICEBE '06. IEEE International Conference on*, pp. 450 –455.

Fernando, N., Loke, S. W. and Rahayu, W. (2013). Mobile Cloud Computing: A Survey, *Future Generation Computer Systems* **29**(1): 84 – 106. Including Special section: AIRCC-NetCoM 2009 and Special section: Clouds and Service-Oriented Architectures.

Fielding, R. T. (2000). *Architectural Styles and the Design of Network-based Software Architectures*, PhD thesis, University of California, Irvine. AAI9980887.

Flores, H., Srirama, S. and Paniagua, C. (2012). Towards Mobile Cloud Applications: Offloading Resource-intensive Tasks to Hybrid Clouds, *International Journal of Pervasive Computing and Communications* **8**(4): 3–3.

Gehlen, G. (2007). *Mobile Web Services: Concepts, Prototype, and Traffic Performance Analysis*, Mainz. PhD thesis, RWTH Aachen University, Aachen, North Rhine-Westphalia, Germany.

Gehlen, G. and Pham, L. (2005). Mobile web services for peer-to-peer applications, *Second IEEE Consumer Communications and Networking Conference*, IEEE, pp. 427–433.

Golbeck, J. A. (2005). *Computing and Applying Trust in Web-based Social Networks*, PhD thesis, University of Maryland at College Park College Park, MD, USA.

Gunasekera, K., Zaslavsky, A., Krishnaswamy, S. and Loke, S. (2010). Service Oriented Context-Aware Software Agents for Greater Efficiency, *in* P. Jdrzejowicz, N. Nguyen, R. Howlet and L. Jain (eds), *Agent and Multi-Agent Systems: Technologies and Applications*, Vol. 6070 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp. 62–71.

Han, B., Jia, W., Shen, J. and Yuen, M. (2005). Context-Awareness in Mobile Web Services, *Parallel and Distributed Processing and Applications* pp. 519–528.

Heckerman, D. (1996). Bayesian networks for knowledge discovery, *in* U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth and R. Uthurusamy (eds), *Advances in knowledge discovery and data mining*, American Association for Artificial Intelligence, Menlo Park, CA, USA, pp. 273–305.

Heflin, J. (2004). OWL Web Ontology Language-Use Cases and Requirements, *W3C Recommendation* **10**: 12.
URL: `http://www.w3.org/TR/webont-req/`

Hu, H., Xu, J. and Lee, D. L. (2003). Adaptive Power-Aware Prefetching Schemes for Mobile Broadcast Environments, *Proceedings of the 4th International Conference on Mobile Data Management*, MDM '03, Springer-Verlag, London, UK, UK, pp. 374–380.

Issarny, V., Georgantas, N., Hachem, S., Zarras, A., Vassiliadist, P., Autili, M., Gerosa, M. and Hamida, A. (2011). Service-Oriented Middleware for the Future Internet: State of the Art and Research Directions, *Journal of Internet Services and Applications* **2**: 23–45.

Jiang, Z. and Kleinrock, L. (1998). An Adaptive Network Prefetch Scheme, *IEEE Journal on Selected Areas in Communications* **16**(3): 358 –368.

Jin, B., Tian, S., Lin, C., Ren, X. and Huang, Y. (2007). An Integrated Prefetching and Caching Scheme for Mobile Web Caching System, *Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing*, Vol. 2, pp. 522 –527.

Junginger, M. O. and Lee, Y. (2005). Chapter 4. Peer-To-Peer Middleware, *Middleware for Communications* pp. 81–107.

JVerstry (2010). Latest news, Online. Last viewed: 29 June 2013.
URL: `https://kenai.com/projects/jxse/pages/LatestNews`

Kang, E., Park, D. and Lim, Y. (2007). Content Aware Selecting Method for Reducing the Response Time of an Adaptive Mobile Web Service, *Ubiquitous Intelligence and Computing*, Springer Berlin, Heidelberg, pp. 748–757.

Kayastha, N., Niyato, D., Wang, P. and Hossain, E. (2011). Applications, Architectures, and Protocol Design Issues for Mobile Social Networks: A Survey, *Proceedings of the IEEE* **99**(12): 2130–2158.

Kern, S., Braun, P. and Rossak, W. (2006). MobiSoft: An Agent-based Middleware for Social-Mobile Applications, *Proceedings of the 2006 international conference on On the Move to Meaningful Internet Systems: AWeSOMe, CAMS, COMINF, IS, KSinBIT, MIOS-CIAO, MONET - Volume Part I*, Springer-Verlag, pp. 984–993.

Kim, Y.-S. and Lee, K.-H. (2009). A Lightweight Framework for Mobile Web Services, *Computer Science - Research and Development, Springer.* **24**: 199–209.

Koskela, T., Kostamo, N., Kassinen, O., Ohtonen, J. and Ylianttila, M. (2007). Towards Context-Aware Mobile Web 2.0 Service Architecture, *Proceedings of the International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies*, UBICOMM '07, IEEE Computer Society, Washington, DC, USA, pp. 41–48.

Kourtellis, N., Finnis, J., Anderson, P., Blackburn, J., Borcea, C. and Iamnitchi, A. (2010). Prometheus: User-Controlled P2P Social Data Management for Socially-Aware Applications, *Proceedings of the ACM/IFIP/USENIX 11th International Conference on Middleware*, Springer-Verlag, pp. 212–231.

Kumar, K., Liu, J., Lu, Y.-H. and Bhargava, B. (2012). A Survey of Computation Offloading for Mobile Systems, *Mobile Networks and Applications* pp. 1–12.

Kurmanowytsch, R., Kirda, E., Kerer, C. and Dustdar, S. (2003). OMNIX: A Topology-Independent P2P Middleware, *Proceedings of the 15th conference on advanced information systems engineering*.

Larosa, Y., Chen, J.-L., Deng, D.-J. and Chao, H.-C. (2011). Mobile Cloud Computing Service based on Heterogeneous Wireless and Mobile P2P Networks, *Proceedings of the 7th International Wireless Communications and Mobile Computing Conference*, pp. 661 –665.

Lee, H., Choi, J. and Park, S. (2005). Context-Aware Recommendations on the Mobile Web, *On the Move to Meaningful Internet Systems 2005: OTM Workshops*, Springer Berlin, Heidelberg, pp. 142–151.

Li, J., Wang, H. and Khan, S. (2012). A Semantics-based Approach to Large-scale Mobile Social Networking, *Mobile Networks and Applications* **17**(2): 192–205.

Li, J., Zhang, Z. and Zhang, W. (2010). MobiTrust: Trust Management System in Mobile Social Computing, *Proceedings of the 10th IEEE International Conference on Computer and Information Technology*, IEEE Computer Society, pp. 954–959.

Lindqvist, J., Cranshaw, J., Wiese, J., Hong, J. and Zimmerman, J. (2011). I'm the Mayor of My House: Examining Why People Use Foursquare: A Social-driven Location Sharing Application, *Proceedings of the 2011 annual conference on Human factors in computing systems*, pp. 2409–2418.

Lubke, R., Schuster, D. and Schill, A. (2011). MobilisGroups: Location-based group formation in Mobile Social Networks, *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2011 IEEE International Conference on*, pp. 502 –507.

Mac Developer Library (2013). *Bonjour Concepts*, Apple Inc. Last viewed: 29 June, 2013.
URL: *https://developer.apple.com/library/mac/#documentation/Cocoa/Conceptual/NetServices/Articles/about.html*

Martin, D., Paolucci, M., McIlraith, S., Burstein, M., McDermott, D., McGuinness, D., Parsia, B., Payne, T., Sabou, M., Solanki, M., Srinivasan, N. and Sycara, K. (2005). Bringing Semantics to Web Services: The OWL-S Approach, *in* J. Cardoso and A. Sheth (eds), *Semantic Web Services and Web Process Composition*, Vol. 3387 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp. 26–42.

Massa, P., Salvetti, M. and Tomasoni, D. (2009). Bowling Alone and Trust Decline in Social Network Sites, *Eighth IEEE International Conference on Dependable, Autonomic and Secure Computing, 2009. DASC '09.*, pp. 658–663.

McGuinness, D., Van Harmelen, F. et al. (2004). OWL Web Ontology Language Overview, *W3C Recommendation* **10**(2004-03): 10.
URL: *http://www.w3.org/TR/owl-features/*

McIlraith, S. A., Son, T. C. and Zeng, H. (2001). Semantic Web Services, *IEEE Intelligent Systems* **16**(2): 46–53.

McNamara, G. and Yang, Y. (2008). Creating a Mobile P2P File Sharing Environment Over Bluetooth, *Third International Conference on Pervasive Computing and Applications, 2008 (ICPCA 2008)*, IEEE, pp. 863–868.

Mecella, M., Angelaccio, M., Krek, A., Catarci, T., Buttarazzi, B. and Dustdar, S. (2006). WORKPAD: An Adaptive Peer-to-Peer Software Infrastructure for Supporting Collaborative Work of Human Operators in Emergency/Disaster Scenarios, *Proceedings of the International Symposium on Collaborative Technologies and Systems*, pp. 173 – 180.

Meshkova, E., Riihijärvi, J., Petrova, M. and Mähönen, P. (2008). A Survey on Resource Discovery Mechanisms, Peer-to-Peer and Service Discovery frameworks, *Computer Networks: The International Journal of Computer and Telecommunications Networking* **52**(11): 2097–2128.

Moore, P., Hu, B. and Wan, J. (2008). Smart-Context: A Context Ontology for Pervasive Mobile Computing, *The Computer Journal Advance Access* pp. 1–17.

Motik, B., Patel-Schneider, P., Parsia, B., Bock, C., Fokoue, A., Haase, P., Hoekstra, R., Horrocks, I., Ruttenberg, A., Sattler, U. et al. (2009). OWL 2 Web Ontology Language: Structural Specification and Functional-style Syntax, *W3C recommendation* **27**: 17.
URL: *http://www.w3.org/TR/owl2-syntax/*

Nakamura, T. (2013). Inside comiket: I didn't have to line up to get into comic market 83, Online. [Last viewed: 28 June, 2013].
URL: *http://kotaku.com/5972736/inside-comiket-i-didnt-have-to-line-up-to-get-into-comic-market-83*

Neyem, A., Franco, D., Ochoa, S. F. and Pino, J. A. (2008). An Approach to Enable Workflow in Mobile Work Scenarios, *in* W. Shen, J. Yong, Y. Yang, J.-P. A. Barthès and J. Luo (eds), *Computer Supported Cooperative Work in Design IV*, Springer-Verlag, Berlin, Heidelberg, pp. 498–509.

Niu, W., Lei, J., Tong, E., Li, G., Chang, L., Shi, Z. and Ci, S. (2013). Context-Aware Service Ranking in Wireless Sensor Networks, *Journal of Network and Systems Management* pp. 1–25.

Obiltschnig, G. (2006). Automatic Configuration and Service Discovery for Networked Smart Devices, *Electronica Embedded Conference Munich*. [Available online; Last viewed: 26 June, 2013].
URL: *http://www.appinf.com/download/AutomaticConfiguration.pdf*

Ong, S. and Center, N. (2006). A Mobile Webserver-Based Approach for Tele-Monitoring of Measurement Devices, *MobiSys06, Demonstration Paper* . Available Online, [Last viewed: 26 June 2013].
URL: *http://www.sigmobile.org/mobisys/2006/demos/Ong.pdf*

Palazzi, C. and Bujari, A. (2011). Social-Aware Delay Tolerant Networking for Mobile-to-Mobile File Sharing, *International Journal of Communication Systems* **25**(10): 1281–1299.

Pallis, G., Vakali, A. and Pokorny, J. (2008). A clustering-based prefetching scheme on a web cache environment, *Comput. Electr. Eng.* **34**(4): 309–323.

Papakos, P., Capra, L. and Rosenblum, D. S. (2010). VOLARE: Context-Aware Adaptive Cloud Service Discovery for Mobile Systems, *Proceedings of the 9th International Workshop on Adaptive and Reflective Middleware*, ARM '10, ACM, New York, NY, USA, pp. 32–38.

Papazoglou, M. P. (2008). *Web Services: Principles and Technology*, Pearson, Prentice Hall.

Papazoglou, M., Traverso, P., Dustdar, S. and Leymann, F. (2007). Service-Oriented Computing: State of the Art and Research Challenges, *Computer* **40**(11): 38 –45.

Patil, A. A., Oundhakar, S. A., Sheth, A. P. and Verma, K. (2004). METEOR-S Web Service Annotation Framework, *Proceedings of the 13th international conference on World Wide Web*, WWW '04, ACM, New York, NY, USA, pp. 553–562.

Pautasso, C., Zimmermann, O. and Leymann, F. (2008). Restful Web Services VS. "Big"' Web Services: Making the Right Architectural Decision, *Proceedings of the 17th international conference on World Wide Web*, WWW '08, ACM, New York, NY, USA, pp. 805–814.

Pawar, P., Srirama, S., van Beijnum, B.-J. and van Halteren, A. (2007). A Comparative Study of Nomadic Mobile Service Provisioning Approaches, *Proceedings of the 2007 International Conference on Next Generation Mobile Applications, Services and Technologies (NGMAST '07)*, pp. 277–286.

Pawar, P., Subercaze, J., Maret, P., van Beijnum, B. and Konstantas, D. (2008). Towards Business Model and Technical Platform for the Service Oriented Context-Aware Mobile Virtual Communities, *Proceedings of IEEE Symposium on Computers and Communications, 2008. (ISCC 2008)*, pp. 103–110.

Pawar, P., Wac, K., Van Beijnum, B.-J., Maret, P., van Halteren, A. and Hermens, H. (2008). Context-Aware Middleware Architecture for Vertical Handover Support to Multi-homed Nomadic Mobile Services, *Proceedings of the 2008 ACM Symposium on Applied Computing (SAC '08)*, ACM, New York, NY, USA, pp. 481–488.

Pernek, I. and Hummel, K. (2009). SocioNet: A Context-Aware Approach for Lowering the Communication Barrier, *in* R. Meersman, P. Herrero and T. Dillon (eds), *On the Move to Meaningful Internet Systems: OTM 2009 Workshops*, Vol. 5872 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, pp. 444–453.

Philips, E., Carreton, A. L., Joncheere, N., De Meuter, W. and Jonckers, V. (2010). Orchestrating Nomadic Mashups using Workflows, *the 3rd and 4th International Workshop on Web APIs and Services Mashups*, ACM, pp. 1:1–1:7.

Pietiläinen, A.-K., Oliver, E., LeBrun, J., Varghese, G. and Diot, C. (2009). MobiClique: Middleware for Mobile Social Networking, *Proceedings of the 2nd ACM workshop on Online social networks*, WOSN '09, ACM, New York, NY, USA, pp. 49–54.

Pratistha, M. (2002). *Exposing Web Services on Mobile Devices. Thesis*, School of Computer Science and Software Engineering, Monash University.

Qureshi, B., Min, G. and Kouvatsos, D. (2012). A Distributed Reputation and Trust Management Scheme for Mobile Peer-to-Peer Networks, *Computer Communications* **35**(5): 608–618.

Qureshi, B., Min, G., Kouvatsos, D. and Ilyas, M. (2010). An Adaptive Content Sharing Protocol for P2P Mobile Social Networks, *IEEE 24th International Conference on Advanced Information Networking and Applications Workshops*, pp. 413 –418.

Rahbar, A. and Yang, O. (2007). PowerTrust: A Robust and Scalable Reputation System for Trusted Peer-to-Peer Computing, *IEEE Transactions on Parallel and Distributed Systems* **18**(4): 460–473.

Rana, J., Hallberg, J., Synnes, K. and Kristiansson, J. (2010). Harnessing the Cloud for Mobile Social Networking Applications, *International Journal of Grid and High Performance Computing (IJGHPC)* **2**(2): 1–11.

Rathnayake, U., Sivaraman, V. and Boreli, R. (2011). Environmental Context Aware Trust in Mobile P2P Networks, *IEEE 36th Conference on Local Computer Networks (LCN)*, pp. 324–332.

Rescorla, E. and Resonance, N. (2006). Introduction to Distributed Hash Tables, *IAB plenary, IETF* **65**.

Rhodes, B., Minar, N. and Weaver, J. (1999). Wearable Computing Meets Ubiquitous Computing: Reaping the Best of Both Worlds, *The Third International Symposium on Wearable Computers, 1999. Digest of Papers*, IEEE, pp. 141–149.

Robinson, R. (2004). Understand enterprise service bus scenarios and solutions in service-oriented architecture, part 1: The role of the enterprise service bus, IBM Developer Works.
URL: *http://www.ibm.com/developerworks/webservices/library/ws-esbscen/*

Sapuppo, A. (2010). Spiderweb: A Social Mobile Network, *Wireless Conference (EW), 2010 European*, pp. 475 –481.

Schilit, B., Adams, N. and Want, R. (1994). Context-Aware Computing Applications, *Proceedings of the 1994 First Workshop on Mobile Computing Systems and Applications*, WMCSA '94, IEEE Computer Society, Washington, DC, USA, pp. 85–90.

Schmidt, H., Kapitza, R. and Hauck, F. (2007). Mobile-Process-based Ubiquitous Computing Platform: a Blueprint, *Proceedings of the 1st Workshop on Middleware-Application interaction: in Conjunction with Euro-Sys 2007*, pp. 25–30.

Schuster, D., Springer, T. and Schill, A. (2010). Service-based development of mobile real-time collaboration applications for Social Networks, *Proceedings of the 8th IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops 2010)*, pp. 232–237.

Seong, S.-W., Seo, J., Nasielski, M., Sengupta, D., Hangal, S., Teh, S. K., Chu, R., Dodson, B. and Lam, M. S. (2010). PrPl: A Decentralized Social Networking Infrastructure, *Proceedings of the 1st ACM Workshop on Mobile Cloud Computing and Services: Social Networks and Beyond*, ACM, pp. 8:1–8:8.

Shankar, P., Huang, Y.-W., Castro, P., Nath, B. and Iftode, L. (2012). Crowds Replace Experts: Building Better Location-based Services using Mobile Social Network Interactions, *Proceedings of the 2012 IEEE International Conference on Pervasive Computing and Communications (PerCom2012)*, IEEE, pp. 20–29.

Sharma, R. and Datta, A. (2011). SuperNova: Super-peers Based Architecture for Decentralized Online Social Networks, *CoRR* **abs/1105.0074**: 1–20.

Sheng, Q., Benatallah, B. and Maamar, Z. (2008). User-Centric Services Provisioning in Wireless Environments, *Communications of the ACM* **51**(11): 130–135.

Singh, A. and Liu, L. (2003). TrustMe: Anonymous Management of Trust Relationships in Decentralized P2P Systems, *Proceedings of the Third International Conference on Peer-to-Peer Computing*, IEEE, pp. 142–149.

Srirama, S., Jarke, M. and Prinz, W. (2006). Mobile Web Service Provisioning, *Proceedings of the International Conference on Internet and Web Applications and Services (ICIW 2006)*, pp. 120–125.

Srirama, S., Jarke, M. and Prinz, W. (2007). MWSMF: A Mediation Framework Realizing Scalable Mobile Web Service Provisioning, *Proceedings of the 1st International Conference on Mobile Wireless MiddleWARE, Operating Systems, and Applications (MOBILWARE '08)*, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), ICST, Brussels, Belgium, Belgium, pp. 1–7.

Srirama, S. N. (2008). *Mobile Hosts in Enterprise Service Integration*, RWTH Aachen University, chapter Chapter 7: Applications of Mobile Web Service Provisioning, pp. 161–183.

Srirama, S. N., Jarke, M., Zhu, H. and Prinz, W. (2008). Scalable Mobile Web Service Discovery in Peer to Peer Networks, *Proceedings of the the 3rd International Conference on Internet and Web Applications and Services (ICIW 2008)*, IEEE Computer Society, pp. 668–674.

Steller, L. and Krishnaswamy, S. (2008). Optimised Semantic Reasoning for Pervasive Service Discovery, *Service-Oriented Computing (ICSOC 2008)* pp. 620–625.

Stuedi, P., Mohomed, I., Balakrishnan, M., Mao, Z., Ramasubramanian, V., Terry, D. and Wobber, T. (2011). Contrail: Enabling Decentralized Social Networks on Smartphones, *in* F. Kon and A.-M. Kermarrec (eds), *Middleware 2011*, Vol. 7049 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp. 41–60.

Takase, T., Makino, S., Kawanaka, S., Ueno, K., Ferris, C. and Ryman, A. (2008). Definition Languages for RESTful Web Services: WADL Vs. WSDL 2.0, Online. IBM Reasearch [Last viewed: 26 June, 2013].
URL: *http://www.techrepublic.com/whitepapers/definition-languages-for-restful-web-services-wadl-vs-wsdl-20/914073*

Tang, J. and Kim, S. (2011). Context-Driven Mobile Social Network Discovery System, *Multimedia, Computer Graphics and Broadcasting* pp. 106–115.

Tian, M., Gramm, A., Ritter, H., Schiller, J. and Voigt, T. (2007). Adaptive QoS for Mobile Web Services through Cross-Layer Communication, *Computer* **40**(2): 59–63.

Toninelli, A., Pathak, A. and Issarny, V. (2011). Yarta: A Middleware for Managing Mobile Social Ecosystems, *in* J. Riekki, M. Ylianttila and M. Guo (eds), *Advances in Grid and Pervasive Computing*, Vol. 6646 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, pp. 209–220.

Tsai, F. S., Han, W., Xu, J. and Chua, H. C. (2009). Design and Development of a Mobile Peer-to-Peer Social Networking Application, *Expert Systems with Applications* **36**(8): 11077 – 11087.

Tuah, N. J., Kumar, M. and Venkatesh, S. (2003). Resource-Aware Speculative Prefetching in Wireless Networks, *Wireless Networks* **9**(1): 61–72.

van Halteren, A. and Pawar, P. (2006). Mobile Service Platform: A Middleware for Nomadic Mobile Service Provisioning, *IEEE International Conference on Wireless and Mobile Computing, Networking and Communications*, IEEE, pp. 292–299.

Waluyo, A. B., Taniar, D., Rahayu, W., Aikebaier, A., Takizawa, M. and Srinivasan, B. (2012). Trustworthy-based Efficient Data Broadcast Model for P2P Interaction in Resource-Constrained Wireless Environments, *Journal of Computer and System Sciences* **78**(6): 1716–1736.

Wang, J., Reinders, M., Pouwelse, J. and Lagendijk, R. (2005). Wi-Fi Walkman: A Wireless Handheld that Shares and Recommends Music on Peer-to-Peer Networks, *Electronic Imaging 2005*, Vol. 5683, International Society for Optics and Photonics, pp. 155–163.

Wang, Y. and Vassileva, J. (2007). A Review on Trust and Reputation for Web Service Selection, *Proceedings of the 27th International Conference on Distributed Computing Systems Workshops*, p. 25.

Williams, A. (2012a). Amazon web services outage caused by memory leak and failure in monitoring alarm. [Last viewed: 5 June 2013].
URL: *http://techcrunch.com/2012/10/27/amazon-web-services-outage-caused-by-memory-leak-and-failure-in-monitoring-alarm/*

Williams, A. (2012b). Google app engine back up after major service disruption dropbox and tumblr also suffer. [Last viewed: 5 June 2013].
URL: *http://techcrunch.com/2012/10/26/google-app-engine-down-with-major-service-disruption-as-dropbox-and-tumblr-also-suffer/*

World Wide Web Consortium (2004). Web Services Architecture.
URL: *http://www.w3.org/TR/ws-arch/*

Wu, X., He, J. and Xu, F. (2009). A Group-Based Reputation Mechanism for Mobile P2P Networks, *Advances in Grid and Pervasive Computing*, Vol. 5529 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp. 410–421.

Xing, B., Seada, K. and Venkatasubramanian, N. (2009). Proximiter: Enabling Mobile Proximity-based Content Sharing on Portable Devices, *Proceedings of the 2009 IEEE International Conference on Pervasive Computing and Communications*, PERCOM '09, IEEE Computer Society, Washington, DC, USA, pp. 1–3.

Xiong, L. and Liu, L. (2004). PeerTrust: Supporting Reputation-based Trust for Peer-to-Peer Electronic Communities, *IEEE Transactions on Knowledge and Data Engineering* **16**(7): 843–857.

Yang, G., Liu, Z., Seada, K., Pang, H.-Y., Joki, A., Yang, J., Rosner, D., Anand, M. and Boda, P. P. (2008). Social Proximity Networks on Cruise Ships, *MobileHCI-MIRW2008, Mobile Interaction with the Real World*, pp. 105–114.

Yang, S. (2006). Context Aware Ubiquitous Learning Environments for Peer-to-Peer Collaborative Learning, *Journal of Educational Technology and Society* **9**(1): 188.

Yang, X., Guo, Y. and Liu, Y. (2011). Bayesian-Inference based Recommendation in Online Social Networks, *IEEE INFOCOM, 2011*, pp. 551 –555.

Yin, L., Cao, G., Das, C. and Ashraf, A. (2002). Power-Aware Prefetch in Mobile Environments, *Proceedings of the 22nd International Conference on Distributed Computing Systems*, ICDCS '02, IEEE Computer Society, Washington, DC, USA, pp. 571 – 578.

Yoshikawa, T., Ohta, K., Nakagawa, T. and Kurakake, S. (2003). Mobile Web Service Platform for Robust, Responsive Distributed Application, *Proceedings of the 14th International Workshop on Database and Expert Systems Applications (DEXA '03)*, IEEE Computer Society, Washington, DC, USA, pp. 144 – 148.

Yu, Z., Liang, Y., Xu, B., Yang, Y. and Guo, B. (2011). Towards a Smart Campus with Mobile Social Networking, *Internet of Things (iThings/CPSCom), 2011 International Conference on and 4th International Conference on Cyber, Physical and Social Computing*, pp. 162 –169.

Zadeh, L. (1965). Fuzzy sets, *Information and Control* **8**(3): 338 – 353.

Zahreddine, W. and Mahmoud, Q. (2005). An Agent-Based Approach to Composite Mobile Web Services, *Proceedings of the 19th International Conference on Advanced Information Networking and Applications (AINA '05)*, IEEE Computer Society, Washington, DC, USA, pp. 189–192.

Zhang, J., Levy, D., Chen, S. and Zic, J. (2010). mBOSSS+: A Mobile Web Services Framework, *Proceedings of the 2010 IEEE Asia-Pacific Services Computing Conference*, APSCC '10, IEEE Computer Society, Washington, DC, USA, pp. 91–96.

Zheng, Y. (2011). Location-based social networks: Users, *Computing with Spatial Trajectories* pp. 243–276.

Zhong, H., Bi, L., Feng, Z. and Li, N. (2008). Research on the design method of mobile social network services, *International Conference on Information Management, Innovation Management and Industrial Engineering, 2008. ICIII '08.*, Vol. 2, pp. 458 –461.

Zhu, J., Oliya, M., Pung, H. and Wong, W. (2010). LASPD: A Framework for Location-Aware Service Provision and Discovery in Mobile Environments, *2010 IEEE Asia-Pacific Services Computing Conference (APSCC)*, IEEE, pp. 218–225.

Ziegler, C.-N. and Golbeck, J. (2007). Investigating Interactions of Trust and Interest Similarity, *Decision Support Systems* **43**(2): 460–475.

Zur Muehlen, M., Nickerson, J. and Swenson, K. (2005). Developing Web Services Choreography Standardsthe Case of REST VS. SOAP, *Decision Support Systems* **40**(1): 9–29.