Proactive Web Service Discovery for Mobile Social Network in Proximity

[1]Chii Chang, [2]Satish Narayana Srirama, [1]Shonali Krishnaswamy, [1]Sea Ling
[1,] *Faculty of Information Technology, Monash University, Australia*
{chii.chang; shonali.krishnaswamy;chris.ling}@monash.edu
[2,] *Institute of Computer Science, University of Tartu, Estonia, srirama@ut.ee*

Abstract

Online social networking has become a popular activity in recent years. Participants are willing to share various content not only to private groups but also to public communities. Meanwhile, the increasing smart mobile device users bring the online social networking to mobile terminals. Mobile users' roles have been switched from content consumers to content providers. Various mobile devices generated content such as video records, pictures, and local events were shared to online communities. However, in the current stage, content sharing is still relying on centralised control or tightly-coupled technologies. In this paper, we applied standard-based mobile Web service approach to realise a loosely-coupled service-oriented proximal mobile social network, which allows users to use heterogeneous devices to share content to public societies in proximity.

Keywords: *Mobile Web Service, Context-Aware, Prediction, Semantic Web, Mobile Social Network*

1. Introduction

Online social networking has been a common daily activity for worldwide Internet users. Social network services such as Facebook[1], Twitter[2], YouTube[3], Flickr[4], and many other similar services enable participants to communicate and share content to each other. The social content sharing is not only limited to a group of participants who knew each other in their real-life but also includes sharing to public societies. For example, various Twitter and YouTube users intend to share their content publicly.

Meanwhile, the increasing smart mobile device users bring online social networking to mobile terminals. Mobile devices such as smart phones, handheld media players, and portable gaming consoles have been employed as one of the major median for users to participate in social networking. The capabilities provided by mobile devices enable users to collect a variety of content such as photographs, recorded videos, his/her current location from Global Positioning System and map services, that can be shared to their social networking groups. With pervasive wireless network connectivity, mobile users can form a mobile social network in proximity (MSNP) to share their content on-the-fly to proximal participants within the wireless network range.

Numerous researchers [1], [2], [3], [4] have proposed mobile social network approaches in recent years. The fundamental goal of MSNP is to share content with surroundings in an unstructured decentralised wireless network environment. Such a topology is also known as a mobile peer-to-peer network. There is a conceptual MSNP example such as the StreetPass[5] mechanism of Nintendo 3DS[6] which allows devices to share the avatar information to each other automatically when they pass by. However, since MSNP is still in its early stage, existing approaches are tightly-coupled to individual platforms. In our perspective, an ideal MSNP is platform independent in which participants can share their content freely without considering which platform they are using on their devices. Moreover, an ideal MSNP is highly dynamic in which mobile devices are capable of discovering the interested content for the users on-the-fly without prior knowledge about the content providers.

A loosely-coupled service-oriented MSNP system throws a number of questions such as how to discover most relevant content from participants for a content requester? How a content provider

[1] http://www.facebook.com
[2] http://www.twitter.com
[3] http://www.youtube.com
[4] http://www.flickr.com
[5] http://www.nintendo.com/3ds/hardware#/8
[6] http://www.nintendo.com/3ds

disseminates his/her content to proper participants without becoming unwanted spam? Moreover, mobile devices are resource constraint. In particular, heavyweight processing can consume too much of resources of a mobile device and also cause high latency. Such an issue can cause users to dislike to participant in the MSNP.

This paper aims to resolve above questions/problems by applying context-aware semantic mobile Web service approach for a service-oriented mobile social network in proximity. In order to identify the most relevant content for a mobile user, we propose context-aware user query prediction scheme. We applied Semantic Annotation for Web Service Description Language and XML Schema (SAWSDL)[7] to mobile Web service (MWS) framework for semantic service discovery. Moreover, we studied a number of service discovery approaches for MSNP with mathematical models and analysed their performance.

The remainder of this paper is structured as follows:

Section 2 describes the background of mobile Web service provisioning and context-aware proactive service discovery. Section 3 explains the proposed context-aware user-preferred-query prediction scheme. Section 4 describes proposed service discovery approaches for MSNP. Section 5 describes the components required for each mobile Web participant in our proposed MSNP environment. Section 6 presents our experimental evaluation. Finally, Section 7 provides the conclusion remarks of this work, and the future research direction.

2. Background

2.1. Mobile Peer-to-Peer Web Service Discovery

Various mobile Web service (MWS) solutions [5], [6], [7] have been introduced in past few years. Hosting a Web server on mobile device to provide content to other participants is no longer a challenge. However, the essential challenge still remains when the communication needs to be established in an unstructured decentralised manner. MWS discovery in peer-to-peer manner involves two parts, physical service discovery and logical service discovery.

Physical peer-to-peer service discovery in an unstructured decentralised MWS environment can be realised by open-source platform independent technologies such as Devices Profile for Web Services (DPWS)[8], JXTA[9] or IETF Zero configuration networking (Zeroconf)[10]. Zeroconf has three primary features: (1) dynamically assigns numeric network addresses for devices when they join the local network without a Dynamic Host Configuration Protocol (DHCP) server; (2) applies multicast domain name system (mDNS) to automatically translate the network addresses and hostnames of devices without a Domain Name System (DNS) server; (3) supports DNS service discovery to enable automatically locating network services without a directory server. There are numerous implementations and libraries to enable Zeroconf network in various platforms, such as UPnP[11], Bonjour[12], Avahi[13], Mono.Zeroconf[14], and so on. These technologies are feasible to enable the physical MWS discovery in a P2P manner. The challenge of service discovery in an unstructured decentralised MWS environment is usually related to the logical service discovery part.

A classic MWS provider supports standard WSDL[15] to describe its operations. However, if a peer only searches services by matching the keywords to the vocabularies used in WSDL, the searching can be extremely difficult and can come out an empty searching result, as most often providers use their own vocabulary in describing the services and operations within WSDL. Alternatively, when we assume MWS providers stick to common names for their services, the

[7] http://www.w3.org/TR/sawsdl/
[8] http://docs.oasis-open.org/ws-dd/ns/dpws/2009/01
[9] http://java.sun.com/othertech/jxta/index.jsp
[10] http://www.zeroconf.org/
[11] http://www.upnp.org/
[12] http://developer.apple.com/opensource/
[13] http://avahi.org/
[14] http://www.mono-project.com/Mono.Zeroconf
[15] http://www.w3.org/TR/wsdl20/

results of this keyword-based search will be quite huge, and still being a problem [8]. Therefore, semantic Web service (SWS) has been introduced to support a more scalable and flexible service description mechanism. There have been a number of specifications to support SWS, such as WSDL-S[16], OWL-S[17], and a more recent W3C[18] approved standard: SAWSDL. The role of SAWSDL is to bridge the types/classes described in ontologies (e.g., OWL[19] documents) with the actual operation of the Web service endpoint. SAWSDL adds semantic annotation in WSDL or XML schema documents to map the Web service operation types or the data types of the schema documents to their corresponding meaning in ontologies. Ontologies describe the meaning of operation or data. In a semantic Web, it is usually expected that the participants of a network will share a fair number of common ontologies to express the resources they provide and use [9]. Benefiting from the knowledge of common ontologies, MWS requesters are capable of automatically identifying the functions and content provided by MWS providers without users' interference.

2.2. Proactive Service Discovery and Context-Awareness

One of the major mechanisms of our work is to enable autonomous service discovery and to filter unwanted services in advance without human interference. Such a mechanism is known as service recommendation or proactive service discovery (PSD). PSD represents an autonomous mechanism that is capable of discovering suitable services for the user based on numerous rules or dynamic factors. Such factors are known as context information, which can be retrieved from various software or hardware sensor components. For example, recent smart devices (e.g., iOS devices[20], Android OS[21] devices) are capable of retrieving a user's current location, moving direction, weather and temperature in the user's current environment and etc. By defining corresponding rules, raw context information can be interpreted to high-level context information to illustrate a meaningful situation. There are a fair number of context interpreting or reasoning solutions existing in the pervasive computing research area for many years. Representatives include Context Toolkit [10], and Context Spaces [11].

A typical PSD approach is to adapt pre-defined rules that describe which service is suitable for the user based on user's preference in current situation or related context factors. However, in reality, a user's preference can dynamically change at runtime due to different context factors. The pre-defined static user preference profiles and rules are difficult to fulfil unseen situations [12], unless the user is willing to adequately define many different preferences manually for all the possible situations. Furthermore, in most cases, a user is unable to define his/her probabilities for events accurately [13]. Consequently, researchers [14], [15] have applied machine-learning mechanism to minimise the need of user's manual setting. These works adapted Bayes' theorem [13] to predict the user preferred query based on historical user query records and associated context factor records. Once the user preferred query is computed, system can match the query to corresponding semantic service type to identify the most suitable service for the user automatically.

3. Context-Aware User Query Prediction

The primary goal of our system is to perform an effective service discovery mechanism for user to discover his/her interested content in MSNP. In order to realise such a need, the user's mobile device needs to be able to identify the user's interested service in current situation. Prediction is a technique that has been applied in various searching mechanisms including classic file systems and Web searching systems [16], [17]. The context-aware prediction scheme in our system takes user’s current contexts as the basis, and then compares the current contexts to historical records to compute which

[16] http://www.w3.org/Submission/WSDL-S/
[17] http://www.w3.org/Submission/OWL-S/
[18] http://www.w3.org
[19] http://www.w3.org/2004/OWL/
[20] http://www.apple.com/ios/
[21] http://www.android.com/about/

query requested by the user has the highest probability. In this section, we describe our proposed context-aware prediction scheme. Firstly, we explain each element involved in the scheme.

Raw Context Data (*rc*) — is the data retrieved from context providers such as Global Positioning System, Compass application, image sensor, video sensor, voice sensor, and so on. An *rc* will be used as the basic input parameter to describe an interpreted context.

Context (*c*) — is an output from a rule-based context interpreting process. Based on [19], an interpreting rule consists of *context type*, 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="meeting_room". When a retrieved location *rc* contains a value: x15y17, which is within the scope of inputMin, and inputMax, the system will consider a context: location="meeting_room" as one of the current contexts.

Query Records (*R*) — Each device should maintain a set of query records *R*, in which $R=\{r_i : 1 \leq i \leq n\}$. *R* representing the device user's previous queries associated with contexts. Each *r* in *R* consists of a query and a set of context (*C*). Each record in r_i consists of a query q_{ri}, and a collection of context information C_{ri} occurred when q_{ri} was submitted by the user.

Candidate Query (q_x) — represents the query, which is sent by the user based on current contexts. When the Predictor component receives a set of context, 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. In order to retrieve a list of candidate queries from past query records (*R*), we can use Equation (1) to retrieve the raw candidate queries *Q*.

$$Q = \bigcup_{r \in R} q_{ri} \tag{1}$$

Context Importance (*e*) — is a user-defined value in the context importance rules (CIR) 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.

3.1. Predicting the Weight of Query

The prediction scheme applies *Bayes' theorem* [13] and the *context weight* model [18] to compute the weight of q_x found in *R*. The three main elements of the prediction scheme include the probability of a query, the weight (influence) of a context to a query, and the user's ranking for a context to a query. The following illustrates the formula to compute the weight of q_x based on current contexts (*C*) and query records (*R*).

$$w(q_x|C,R) = \sum_{i=1}^{|C|} \left(\frac{P(c_i|q_x) \cdot P(q_x)}{P(c_i)} \cdot \frac{1 + e(c_i, q_x)}{|C| + \sum e} \right) \tag{2}$$

Where $w(q_x|C,R)$ denotes the weight of q_x computed from summing the influence value of each current context. The influence value of a current context (c_i) to q_x is computed based on matching the current context c_i to the contexts in each r_i in *R*, plus the *context importance* (*e*). $P(c_i|q_x)$ denotes the probability of q_x requested by the user when context c_i occurs, in which $P(c_i|q_x) = |\{r \in R : q_r=q_x \wedge \exists c \in C_r, c=c_i\}| / |\{r \in R : q_r=q_x\}|$. $P(q_x)$ denotes the probability of q_x based on its occurrence in *R*, in which $P(q_x) = |\{r \in R : q_r=q_x\}| / |R|$. $P(c_i)$ is the probability of a randomly selected query that contains c_i as one of its attributes, in which $P(c_i)$ =

$P(c_i|q_x)P(q_x)+P(c_i|q_x')P(q_x')$. $P(q_x')$ denotes the probability of other queries in R excluding the consideration of records that have $q_r=q_x$, in which $P(q_x') = |\{r \in R : q_r \neq q_x\}| / |R|$. $P(c_i|q_x')$ denotes the probability of other queries requested by the user excluding q_x when c_i occurrs, in which $P(c_i|q_x') = |\{r \in R : q_r \neq q_x \wedge \exists c \in C_r, c=c_i\}| / |\{r \in R : q_r \neq q_x\}|$. In order to retrieve the rate of a single c_i to q_x, we compute the value using Equation (3):

$$rate(c_i|q_x) = \frac{P(c_i|q_x) \cdot P(q_x)}{P(c_i)} \cdot \frac{1+e(c_i,q_x)}{|C|+\sum e} \tag{3}$$

Where $\sum e$ is the sum of a set of context importance (e) in the context importance rules (E), in which the defined value of e is not 0. $e(c_i,q_x)$ denotes one of the defined rule, where: $e(c_i,q_x) \leftarrow e \in E, c_e=c_i \wedge q_e=q_x$.

A prediction scheme that relies on the users 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 MSNP 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 in that place. The detail of the prediction model that applies *social context* will be explained in our future work.

4. Service Discovery in MSNP

4.1. Overview of the Architecture

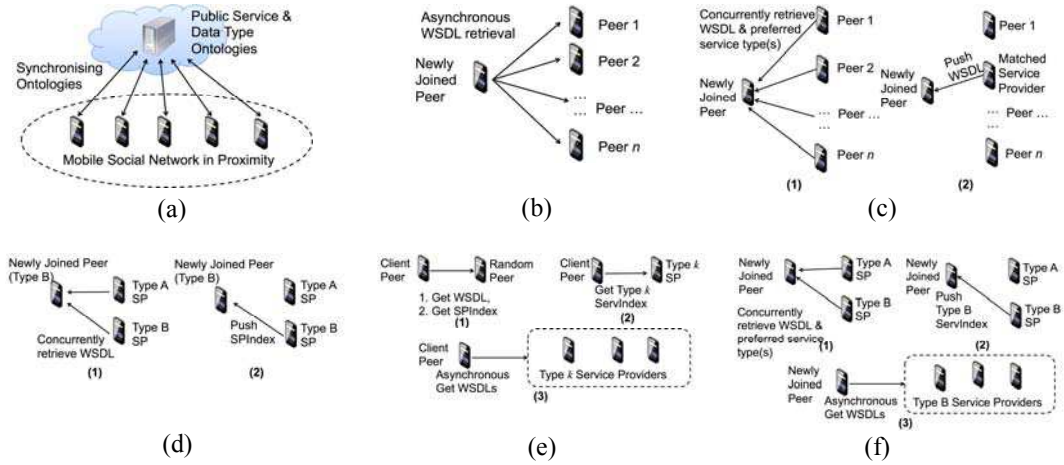

Figure 1. Service Discovery Approaches

Figure 1(a) illustrates the basic knowledge sharing in our system. In order to enable the interaction in such an environment, a common knowledge need to be disseminated in the network. We applied Bonjour to realise the physical service discovery, and applied SAWSDL for logical service discovery using semantic service expression.

In this system, each peer is a Web service requester and can also be a RESTful Web service [19] provider if the participant is sharing content to the others. We expect each peer is SAWSDL-compliant. Moreover, each MWS should have pre-downloaded a fair number of public common ontologies that have been published on cloud resources (e.g., Swoogle[22], or FUSION[23]). A public common ontology describes numerous common service types and data types semantically. Each SAWSDL-compliant MWS provider describes its services using semantic annotations that map to the corresponding

[22] http://swoogle.umbc.edu/
[23] http://www.seerc.org/fusion/semanticregistry

ontology types. Benefiting from the public common ontologies and semantic annotation, a MWS requester can identify whether a service matches to the functionality it needs or not from the service provider's WSDL and related documents (e.g., XML Schema). In following subsections, we discuss four applicable service discovery approaches in MSNP. We also analyse the performance of each approach by using mathematical models.

4.2. Pull-based Service Discovery

Figure 1(b) illustrates a pure pull-based service discovery approach. In this approach when a peer joins the network, it asynchronously retrieves service description metadata (i.e. WSDL with semantic annotations) from each existing service provider peer in the network. By processing the WSDLs, the newly joined peer (*NJ*) is capable of identifying the semantic service types provided by each service provider. A common problem of this simple model is the latency issue. Because WSDLs are described in XML format, resource-constraint mobile devices are usually unable to process a high amount of XML documents effectively. We can model the performance of this approach as follows:

$$T_{disco}^{NJ} = max_{i=1}^{|P|}(T_{gSD_{pi}}) + \sum_{i=1}^{|P|} T_{psSD_{pi}} \quad (4)$$

Where, the total time for NJ to identify which service provider can fulfil its needs is the maximum time to retrieve the WSDL files from all the service provider peers asynchronously, plus the total time for processing each WSDL.

$T_{gSD_{pi}}$ = time spent for *NJ* to retrieve a *p*'s WSDL.

$T_{psSD_{pi}}$ = time spent for *NJ* to parse one of the WSDL files to identify the services provided by the service provider.

4.3. Push-based Service Discovery

Figure 1(c) illustrates a push-based service discovery approach in which every existing peer is actively interacting with *NJ* when *NJ* joins the network. In this model, every participative peer is required to have embedded Web service provisioning mechanism in order to let other peers to advertise service description to it. As Figure 1(c) shows, when *NJ* joins the network, other existing peers will actively retrieve *NJ*'s WSDL in order to interact with *NJ*. Existing peers will then retrieve *NJ*'s preference profile, which describes what type of service *NJ* is interested. The preferred service type was identified from the context-aware user query prediction mechanism, which was described in previous section. After existing peers retrieve *NJ*'s preferred service type, the peers who can provide the preferred service type (we call them matched service providers, or MPs for short), will actively invoke *NJ*'s operation to advertise their WSDLs to *NJ*. We can model the performance of this approach as follows:

$$T_{disco}^{NJ} = max_{i=1}^{|P|}(T_{gSD}^{pi} + T_{psSD}^{pi} + T_{gPref}^{pi} + T_{psPref}^{p}) + max_{j=1}^{|MP|}(T_{pSD}^{mpj}) \quad (5)$$

Where:

P = all the service providers in the network as described in previous subsection.

T_{gSD}^{pi} = time spent for a *pi* to retrieve *NJ's* WSDL.

T_{psSD}^{pi} = time spent for a *pi* to process *NJ's* WSDL. This process happens concurrently.

T_{gPref}^{pi} = time spent for *pi* to retrieve *NJ's* preference profile.

T_{psPref}^{p} = time spent for a *pi* to process *NJ's* preference profile in order to identify *NJ's* preferred service type.

T_{pSD}^{mpj} = time spent for a matched service provider to push its WSDL to *NJ*.

4.4. Hybrid-based Assistive Service Discovery

In a hybrid-based assistive service discovery approach, we expect that there are numerous peers regularly participating in the network. These active peers intend to interact with others and share information they've collected in the areas. We call such peers as Super Peers (*SPs*). *SPs* are active participants. Whenever a *SP* is in the network, it continuously tracks other peers.

Figure 1(d) illustrates an example about how *SPs* obtain and disseminate information in the network. As Figure 1(d)(1) shows, when a *NJ* who provides "*TypeB*" service, joins the network, *SPs* will concurrently retrieve the *NJ*'s WSDL in order to identify what service NJ provides. Afterwards, the *TypeB SP*, a *SP* who mainly maintains *TypeB* service provider list, will push Super Peer Index (*SPIndex*) to *NJ*. A *SPIndex* is a simple metadata that describes what type of services are currently available on the network, and which *SP* is maintaining the *ServIndex* (a list of service providers' URIs) of each particular service type.

If a newly joined peer is a pure Web service client (*CP*) that is only capable of performing pull-based service discovery, it can seek assistance from the other peers as Figure 1(e) shows.

When *CP* joins the network, it can randomly select one of the existing service provider peers to retrieve *SPIndex* from it. After *CP* retrieves *SPIndex*, it can find the *SP* who provides the *ServIndex* it needs. With the *ServIndex*, *CP* will be able to obtain a complete list of service provider URIs (Uniform Resource Identifiers) who can fulfil *CP*'s preferred service type. We can model the performance of this service discovery approach as follows:

$$T_{disco}^{NJ} = T_{gSD_{rp}} + T_{psSD_{rp}} + T_{gSPI_{rp}} + T_{psSPI} + T_{gSD_{sp}} + T_{psSD_{sp}} + T_{gSvI_{sp}} + T_{psSvI} + max_{j=1}^{|MS|}(T_{gSD_{msi}}) \quad (6)$$

Where:

$T_{gSD_{rp}}$ = time spent for *NJ* to request WSDL from a randomly selected service provider peer (denoted by *rp)*.

$T_{psSD_{rp}}$ = time spent for *NJ* to parse *rp's* WSDL.

$T_{gSPI_{rp}}$ = time spent for *NJ* to request *SPIndex* from *rp*.

T_{psSPI} = time spent for *NJ* to process *SPIndex*.

$T_{gSD_{sp}}$ = time spent for *NJ* to retrieve *sp's* WSDL.

$T_{psSD_{sp}}$ = time spent for *NJ* to process *sp's* WSDL.

$T_{gSvI_{sp}}$ = time spent for *NJ* to request *ServIndex* from *sp*.

T_{psSvI} = time spent for *NJ* to process *ServIndex*.

$T_{gSD_{msi}}$ = time spent for *NJ* to retrieve WSDL from each matched service provider (*msi*) described in *ServIndex*.

Above, we have discussed the service discovery approach for a pure client. On the other hand, if the newly joined peer (NJ) contains embedded mobile Web service provider, the push-based service discovery can be performed. Note that the push-based approach is useful for active MWS providers to perform autonomously advertisment. Figure 1(f) illustrates such a scenario in which *SPs* will actively interact with *NJ* and retrieve *NJ's* preferred service type. In this scenario, *NJ* prefers *Type B* service. Hence, *Type B SP* will push *Type B ServIndex* to *NJ*. Afterwards, *NJ* can retrieve WSDLs of *Type B* service providers for further invocation needs. We can model the performance of this approach as follows:

$$\begin{aligned} T_{disco}^{NJ} &= max_{i=1}^{|SP|}(T_{gSD}^{spi} + T_{psSD}^{spi} + T_{gPref}^{spi} + T_{psPref}^{spi}) \\ &+ max_{j=1}^{|MSP|}(T_{pSvI}^{mspj}) + \sum_{j=1}^{|MSP|} T_{psSvI}^{NJ} + max_{k=1}^{|MP|}(T_{gSD_{mpk}}^{NJ}) \end{aligned} \quad (7)$$

Where:

T_{gSD}^{spi} = time spent for a *sp* to retrieve *NJ*'s WSDL.

T_{pSvI}^{mspj} = time spent for a *sp* who matched to *NJ's* preferred service type to push *ServIndex* to *NJ*.

T^{spi}_{psSD} = time spent for a *sp* to process *NJ's* WSDL.

T^{spi}_{gPref} = time spent for a *sp* to retrieve *NJ's* preference profile.

T^{NJ}_{psSvI} = time spent for *NJ* to process *ServIndex*.

$T^{NJ}_{gSD_{mpk}}$ = time spent for *NJ* to retrieve WSDL from each service provider described in *ServIndex*.

T^{spi}_{psPref} = time spent for a *sp* to process *NJ's* preference profile in order to identify *NJ's* preferred service type.

5. Components of a Mobile Peer in MSNP

As explained in earlier section, mobile peer have to provide several abilities in the discovery procedure. Figure 2 illustrates the components of the mobile peer in our system. It consists of two groups of components: service invocation related components and service provision related components.

5.1. Service Invocation Related Components

- *Context Middleware* — continuously operates individually to retrieve up-to-date raw context data from context providers, and interprets the collected raw context data as compound contexts based on the predefined 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.
- *Context Provider* — can be an external sensor device, or an embedded application within a mobile device. Examples can be a compass application, a map application (e.g., Google map24), a sound detection application, etc.
- *Knowledge Manager* (denotes by KM)—is responsible to manage the annotation of semantic service type and data types used in semantic metadata such as OWL, RDF and SAWSDL. A mobile host uses *KM* to synchronise its semantic knowledge with public ontology services. The detail of selecting appropriate ontology is out of the score of this paper. At this stage, we assume such a mechanism has been implemented, and a number of common ontologies have been shared within the mobile social network. Hence, each mobile peer is capable of identifying the services provided by each other based on semantic service matchmaking.
- *Service Matchmaker* — performs semantic service matchmaking based on the input parameters including a user query, and a list of available services described in URIs and corresponding semantic types. In our system, we expect that all the semantic types used by participants can be found in the common ontologies.
- *Service Discoverer* — is capable of performing physical service discovery within Zeroconf network. It retrieves and maintains a list of service provider domains published in the network. When an external service provider needs to be invoked, the Service Invoker will firstly request Service Discoverer for the target service provider's current address. It is an important prerequisite process because the mobile P2P network is a dynamic environment, each peer can sudden disconnect and may be reconnected again. Their IP address can change frequently. Moreover, some peers may not be able to reconnect to the network again.
- *Service Invoker*—supports dynamic Web service invocation mechanism by dynamically generating a corresponding Web service client based on the descriptions of the service (e.g., WSDL, XML Schema, RDF, etc.). In our system, we expect that each service provider will reply its basic WSDL when it receives a simple HTTP GET (without extra path). Hence, each mobile peer is capable of performing further service invocation to such a service provider.
- *Recommender* — is an always-on local service on the mobile device. It collaborates with the Context Middleware to continuously observe user's contextual information. It applies a number of predefined event rules to decide when to trigger the proactive service discovery operation. It also consists of a mechanism to predict user's preferred query based on context information and the historical records of user's queries. The prediction method has been described in previous section. When user performs searching, the recommended services are

shown on the first few number of user's query results, and the rest services, which have been briefly described in the *ServIndex,* will also be selectable, but distinguish to the recommended services, the service description and related data of these services have not yet been fetched. Invoking these services will take longer time than invoking the recommended services. In order to continuously improve the accuracy of prediction, Recommender records user's queries and the context information when user invokes a service.

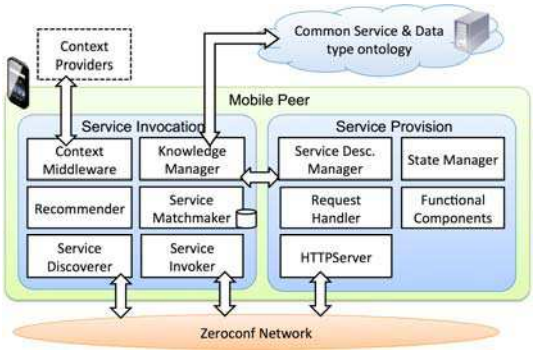

Figure 2. Components of a mobile Web service peer

5.2. Service Provision Related Components

- *Service Description Manager* — is responsible to edit the WSDL of the service at runtime before it is sent to the requester. This mechanism is important because a participant in Bonjour network doesn't have a static IP address, and its selected domain name may change when another participant has used a redundant name. Hence, a mobile host needs to ensure the information described in its WSDL is correct before the document is sent.
- *Request Handler* — is responsible for handling incoming request query and outgoing responses. It forwards the request to corresponding Functional Component for the process.
- *Functional Components* — represent the operations provided by the mobile host.
- *State Manager* — controls the state of mobile host to ensure the server will not be overloaded. For example, if there are over 100 concurrent requests for the HTTPServer, State Manager may inform the HTTPServer to stop receiving more incoming requests until a fair number of processes have been completed. Furthermore, if an incoming query requests the state of the mobile host, the Request Handler will retrieve such information from State Manager.
- *HTTPServer* — is a typical Web server embedded on the mobile host that handles message transmission using HTTP protocol. It also publishes itself in Zeroconf network.

6. Evaluation of the Prototype

We have implemented a prototype of our proposed system. In this section, we describe the evaluation results in two parts. The first part contains the accuracy evaluation of our context-aware user query prediction scheme. The second part describes the performance testing of our prototype in real mobile devices.

6.1. User Query Prediction Scheme Evaluation

Our prediction scheme is generic, and is applicable to general context information. In order to test the accuracy of the scheme, we have programmed a user query record generator to simulate user's 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. Record generator will randomly generate a given number

of records (e.g., 100, 200, 300, etc.). Each record describes one query type, and five context values. For example, assume we use the setting in Table 1, the record generator will randomly select the record type from A to E. Assume 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 = random value from A1 to A5, CW = random value from W1 to W5, CP = random value from P1 to P5. The two static values (L1, T1) represent the contexts that influenced the user's decision to select Q1.

Table 1. Parameters for Prediction Testing (1)

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

Figure 3 illustrates the results of our evaluation by using the parameter setting in Table 1. In axis-x, it shows how many percentage records we have used as training set to predict the rest of records. For example, the very first value on the left-bottom of the graph represents the accuracy result based on using total 100 query records, and 60% of the records were used as training set to predict the rest 40% of the records.

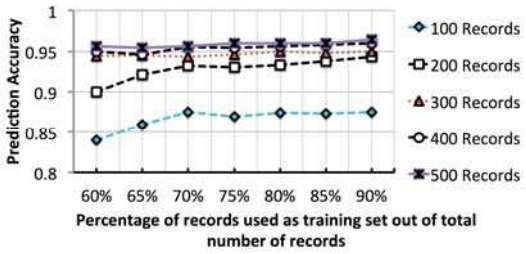

Figure 3. Prediction Result (1)

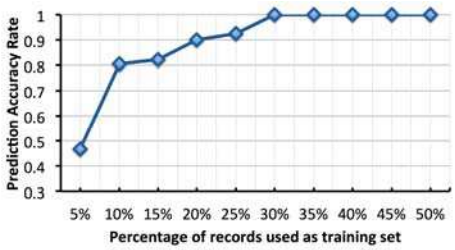

Figure 4. Prediction Result (2)

We have also tested our prediction scheme using the subset of epSICAR-dataset [20]. We used 200 sequence records from the dataset. Each record consists of two context attributes: location and action. Each record also has a corresponding object (e.g., Hi-Fi Music system for listening music in living room), which can be considered as a service. The testing result is shown in Figure 4.

6.2. Performance of Service Discovery in MSNP

Performance is always a concern in mobile Web service environments because resource constraint mobile devices are unlikely to process XML-format metadata based Web service communication effectively. Moreover, high amount of network transmission can potentially consume too much battery-life of mobile devices. We have considered such an issue, and performed testing on battery-life of a MWS provider which embedded on a recent mobile device Apple iPod touch 4th generation, with 99% battery-level, and connected to IEEE802.11g WiFi network (54bps). We implemented Web service clients on a Macbook laptop computer to simulate 20,000 Web service clients. We let each Web service client to concurrently submit a 6462 bytes data to the MWS (that is 20,000 requests sent to MWS at once) using HTTP POST method, and the MWS replies requests asynchronously. After all the clients received the responses from MWS, the battery-level of the iPod touch remains in 91%. Indeed, the battery power may be dependent on devices. Some devices may have poorer battery power, but we can measure that in the near future, battery-life is less an issue for MWS provisioning. In following subsections, we describe our testing result on performance and the comparison of four service discovery approaches described in Section 4.

6.2.1. Environment Setting

As already explained, our testing environment is in IEEE 802.11g WiFi network, and Apple iPod touch 4th generation (Apple A4 800MHz CPU power) is mainly used as a mobile Web service peer. We also used a Macbook laptop (Intel Core 2 Duo 2.4GHz CPU power) to simulate the rest peers. The parameters involved in our analysis have been described in Table 2.

Table 2. Parameters for performance testing

Element	Value	Element	Value
WSDL	3482 bytes	No. of Existing Service Provider Peers	50 to 200
SPIndex	2788 bytes	No. of Super Peers	20
ServIndex	1055 bytes	No. of NJ's preferred service type	1
Preferred Service Profile	177 bytes		

6.2.2. Testing Results

Figure 5(a), (b), (c) illustrate the time-spent on service discovery for each approach described in Section IV that is influenced by the number of matched service providers (denoted by *MP)* and the total number of service providers (denoted by *P)* in the environment. Figure 5(a) denotes an environment with 100 service providers, 5(b) denotes an environment with 150 service providers, and 5(c) denotes an environment with 200 service providers. The number of *MP* doesn't influence the pure-pull approach because in the pure-pull approach, the newly joined peer (denoted by *NJ)* always needs to process all the WSDL from environmental service providers in order to identify which service provider can fulfil its needs. On the other hand, The pure-push approach can improve the latency problem because the task to identify which service provider can fulfil *NJ's* preferred service type, has been assigned to the other active peers who intend to advertise their services to *NJ*. However, a mobile device is resource constraint. With its limited process power, processing high amount of request from other peers can still increase the overall latency for *NJ*. The same issue occurred in the assistive push-based approach, in which a number of super peers are available to assist the service discovery process. Assistive approaches can highly improve the overall performance. However, due to the mobile device's limited process power, the push-based assistive approach results a poorer performance than the pull-based assistive approach. By comparing Figures 5(a), (b), and (c), we can clearly see that the pure-pull approach was highly influenced by the total number of service providers in the environment.

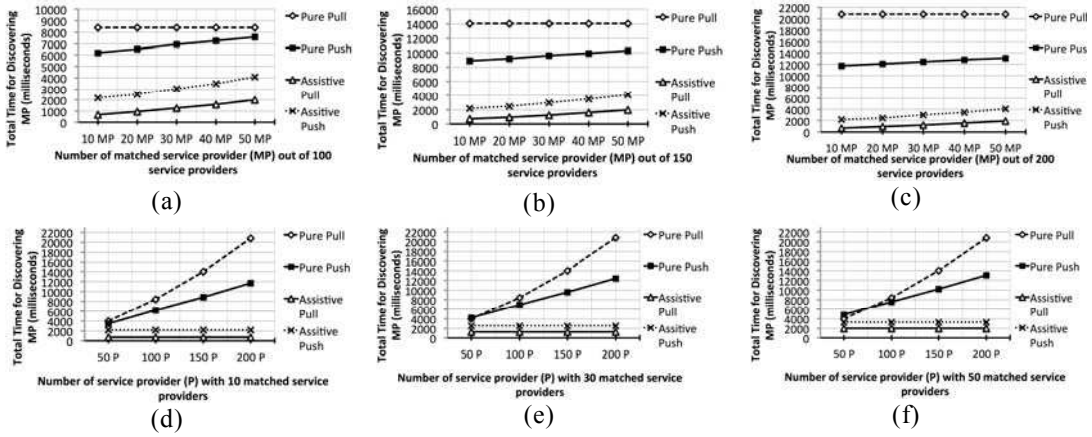

Figure 5. Performance testing result based on number of matched services and the total number of service providers

Figures 5(d), (e), (f) illustrate the latency influenced by the number of service providers in the environment. They clearly show that the pure-pull approach is highly influenced by the number of service providers. An interesting point shown on these graphs is that the pure-pull approach can have a better performance than the pure-push approach when there are only 50 service providers in the environment. It is because processing the concurrent request/response in a resource constraint mobile device, in which the high frequency of data transmission can cause inevitable latency. On the other hand, when there are only 50 service providers in the environment, the pure-pull approach can handle the process well. The number of environmental service providers less affected the assistive-pull approach and the assistive-push approach because in these two approaches, *NJ* did not need to communicate with all the service providers.

Figure 6 illustrates the latency caused by transmitting and processing WSDL files. In a pure-pull approach, a *NJ* needs to retrieve the WSDL file from each service provider, and parse the WSDL in order to identify which service provider can fulfil its needs. Hence, the larger the WSDL size is, the higher is the latency it causes. On the other hand, the pure-push approach benefited by the distributed WSDL process in which each active service provider who intends to advertise its service to *NJ* will process *NJ's* metadata to identify what service *NJ* needs. Hence, the WSDL size does not explicitly affect the overall latency. How- ever, as Figure 6 shows, when the WSDL size is small, the pure-pull approach can result a slightly better performance than the pure-push approach, it was caused as the resource constraint mobile device has limited power to handle high amount of request/response processes when all the existing service providers are con- currently sending requests to it. The figure also shows that the WSDL size doesn't explicitly influence the latency of the assistive-pull approach and the assistive-push approach because in these two approaches, *NJ* didn't need to process high amount of WSDL files to identify the service provider it needs. Due to the same reason as the pure-push approach in which the mobile device has limited power to handle request/response, the assistive-push approach has poorer performance than the assistive-pull approach.

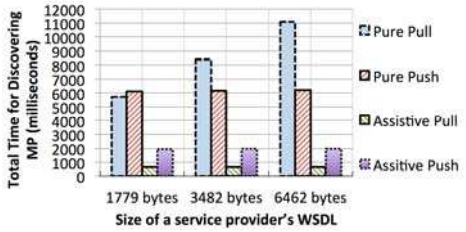

Figure 6. Latency caused by processing WSDL files

7. Conclusions and Future Work

In this paper, we propose a mobile Web service-based approach for mobile social network in proximity (MSNP). The goal of this work is to realise a loose-coupled service-oriented mobile social network for mobile user to share content to the others in proximity within an unstructured decentralised manner. The primary challenge of realising MSNP in a service-oriented manner is to enable dynamic service discovery proactively in order to filter unwanted searching results. Proactive dynamic service discovery involves two important tasks: (1) identifying which service provider can fulfil the user's interests and (2) filtering unnecessary service information to reduce the searching results shown on the user's mobile device. We have applied Web service standard-based semantic Web services to enable the task of identifying service providers. We also describe how the participants can communicate to realise such a semantic mobile Web service environment. For the task of filtering unnecessary service information, we proposed context-aware user query prediction scheme to identify what type of service user is interested in his/her current environment based on historical query records and associated context information records.

For future work, we intend to investigate trust and policy issues. Trust and policy are important concerns for content sharing in a public MSNP. Since the environment is highly dynamic, participants usually have very limited knowledge about each other. A mobile content

provider might provide content that is inconsistent to its service description, or the provided content is not fully suitable for the content requester. For example, an event picture sharing service provider might maliciously include some uncomfortable pictures in its service, some content requesters may not want to receive such content. However, with limited knowledge obtainable in MSNP, requesters can't control the content they receive. We intend to apply cloud resources to support such a need in our future work.

8. References

[1] B. Xing, K. Seada, N. Venkatasubramanian, "Proximiter: Enabling mobile proximity-based content sharing on portable devices", In IEEE International Conference on Pervasive Computing and Communications, pp.1-3, 2009.

[2] A. Sapuppo, "Spiderweb: A Social Mobile Network", In 2010 European Wireless Conference, pp.475-481, 2010.

[3] S. M. Allen, G. Colombo, R. M. Whitaker, "Uttering: Social Micro-blogging Without the Internet", In Proceedings of the Second International Workshop on Mobile Opportunistic Networking (MobiOpp '10), pp.58-64, 2010.

[4] S. L. Sapuppo A, "Local Social Networks", In Proceedings of International Conference on Telecommunication Technology and Applications, pp.15-22, 2010.

[5] S. Srirama, M. Jarke, W. Prinz, "Mobile Web Service Provisioning", In Proceedings of ICIW 2006/AICT 2006, pp.19-25, 2006.

[6] H. Schmidt, A. Kohrer, F. J. Hauck, "SoapME: a Lightweight JavaME Web Service Container", In Proceedings of the 3rd Workshop on Middleware for Service Oriented Computing, pp.13-18, 2008.

[7] F. AlShahwan, K. Moessner, "Providing SOAP Web Services and Restful Web Services from Mobile Hosts", In Proceedings of ICIW 2010, pp.174-179, 2010.

[8] S. N. Srirama, M. Jarke, H. Zhu, W. Prinz, "Scalable Mobile Web Service Discovery in Peer to Peer Networks", In Proceedings of ICIW 2008, pp.668-674, 2008.

[9] J. Hebeler, M. Fisher, R. Blace, A. Perez-Lopez, Semantic Web Programming, Wiley, 2009.

[10] A.K. Dey, "Understanding and Using Context", Personal Ubiquitous Computing 5, pp.4-7, 2001.

[11] A. Padovitz, S. W. Loke, A. Zaslavsky, "Towards a Theory of Context Spaces", In Proceedings of PERCOMW '04, pp. 38-42, 2004.

[12] A. Chen, "Context-aware Collaborative Filtering System: Predicting the User's Preferences in Ubiquitous Computing", In Proceedings of CHI '05, pp.1110-1111, 2005.

[13] D. Heckerman, "Bayesian Networks for Knowledge Discovery", In Advances in Knowledge Discovery and Data Mining, pp.273–305, 1996.

[14] K. Rasch, F. Li, S. Sehic, R. Ayani, S. Dustdar, "Context-driven Personalized Service Discovery in Pervasive Environments", In Proceedings of World Wide Web 14, pp.295-319, 2011.

[15] C. Chang, S. Ling, S. Krishnaswamy, "ProMWS: Proactive Mobile Web Service Provision using Context-awareness", In Proceedings of PERCOM Workshops 2011, pp.69-74, 2011.

[16] Z. Jiang, L. Kleinrock, "An Adaptive Network Prefetch Scheme", Selected Areas in IEEE Journal on Communications, , vol. 16, no.3, pp.358-368, 1998.

[17] N. J. Tuah, M. Kumar, S. Venkatesh, "Resource-aware speculative prefetchingin wireless networks", pp.61–72, 2003.

[18] P. Delir Haghighi, S. Krishnaswamy, A. Zaslavsky, M. M. Gaber, "Reasoning about context in uncertain pervasive computing environments", In Proceedings of EuroSSC'08, pp.112–125, 2008.

[19] A. Rodriguez, "Restful web services: The basics", https://www.ibm.com/developerworks/webservices/library/ws-restful/

[20] epSICAR-dataset http://www.imada.sdu.dk/~gu/