IMPLEMENTATION OF A MIDDLEWARE ARCHITECTURE

FOR COLLABORATION AMONG MOBILE DEVICES

IN OPPORTUNISTIC NETWORKS

by

SAMAYA MADHAVAN

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE AND ENGINEERING

THE UNIVERSITY OF TEXAS AT ARLINGTON

DECEMBER 2012

ACKNOWLEDGEMENTS

I thankfully acknowledge the guidance, assistance and support of all the wonderful PICO Lab mates and in getting my thesis to its present form. My profound thanks in particular are due to: My Thesis Supervisor Dr Mohan Kumar, for his unstinting support on my efforts, constant encouragement, for his wisdom and above all, his unmatchable patience; My Thesis Defense Panel of Dr Mathew Wright and Dr Yonghe Liu – for their uncompromising critics that made me strive towards perfection; My dear friends, Shiraz Qayyum and Lynda Thomas – for their constant encouragement, support and appreciation of my efforts; My beloved brother, Varun Madhavan – who constantly pepped me up and made me realize the value of education and hard work; my gifted parents – mother Subhashini Madhavan and father Madhavan Sadagopan, who had gave me the best of their everything to me, more so their belief in me even when I did not in myself.

NOVEMBER 19TH 2012

ABSTRACT

IMPLEMENTATION OF A MIDDLEWARE ARCHITECTURE

FOR COLLABORATION AMONG MOBILE DEVICES

IN OPPORTUNISTIC NETWORKS

Samaya Madhavan, MS

The University of Texas at Arlington, 2012

Supervising Professor: Mohan Kumar

Opportunistic networks provide a viable platform for mobile communications due to the ubiquity of smart phones. These networks are characterized by lack of end to-end reliable connections. In such networks, establishing collaboration among nodes is a challenge. In this thesis, a middleware architecture for opportunistic communication and collaboration has been designed, developed, and implemented. The middleware architecture supports a protocol for efficiently exchanging information between mobile nodes during an opportunistic contact. Message formats for differentiating the various kinds of messages that

will be transferred across the network are defined to maintain consistency and reduce redundancy. The architecture consists of several modules equipped with system software to differentiate, compute, update and merge information acquired at each participating node. The middleware has been developed with a view to support a variety of application services on opportunistic networks. In particular, the middleware performs service composition utilizing basic services available in different devices. A sample language translation application has been implemented involving several android devices to test service composition in an opportunistic network created among mobile devices. Finally experimental results are carried out to measure success rates of service composition. The experimental studies include results with the following scenarios: parallel service requests; varying service composition lengths; varying content size; varying number of nodes;

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Opportunistic Networks

Opportunistic networks are typically those networks that are operated under wireless environments. Nodes in these networks are handheld devices. These networks do not have any centralized server and require no additional infrastructure for it to function.

Such networks are formed on the fly and are continuously prone to change in their topology; for example people in a coffee shop or at gym. These kinds of networks are gaining wide popularity because of the fact that the information acquired through this network is freely available from the neighboring nodes and do not require access to the web. Also, these days smart phones are available with a wide range of resources and sensors such as camera, GPS and so on which would make opportunistic networking rich in terms of the kind of information that can be acquired from the network. There are many areas of research in opportunistic networks such as services, routing, context, trust and security, information management, resource management etc.

This paper will present the analysis of service composition in opportunistic network by using Bluetooth connections.

1.2 Problem Definition

Services in a network refer to some combination of programming and data. Services can be divided into or can be made up of low-level services. In opportunistic networks, nodes can have one or more of these lower or high level services and at the same time there could be other services that a node is looking for from the network in a particular required sequence. These low level services can be composed in a particular sequence to form required high level services. This process is termed as *service composition*. Service composition depends on a number of attributes such as the number of people

available in the network, how many times they meet, how long they meet, number of required services available and so on. In this paper, we analyze and contribute to answer how efficient and successful services can be composed in such transient and constraint prone environment.

1.3 Proposed Solution

Based on experimental analysis of how many other nodes a particular node encounters and for how long they meet, a message format is proposed with an aim to efficiently acquire and pass messages in the network. This format incorporates a way to forward information it has acquired regarding what other devices have and is looking for in addition to such details about its own resources.

Further a generic middleware for handling any kind of service composition is developed. A mechanism to identify matches between available and required services to compose services in a most efficient way is included in the middleware. Composition of services occurs through a simple handshake method. Several experimental studies were performed and the times taken for those compositions were used to analyze the performance of the middleware.

1.4 Thesis Goal

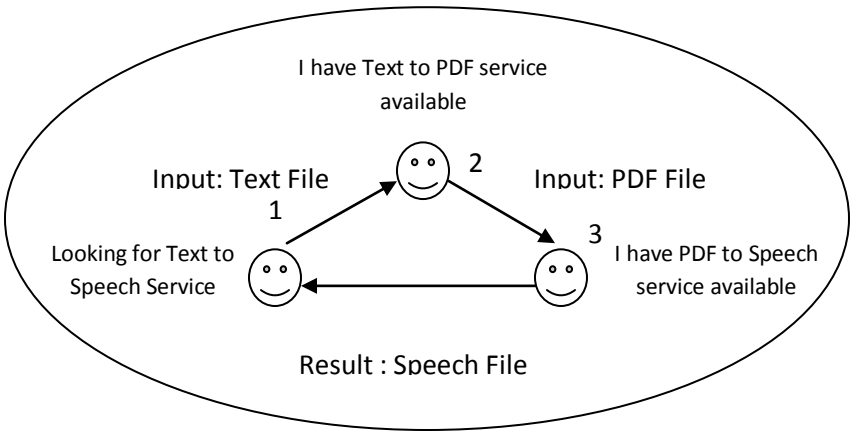

Figure 1-1 Service Composition at a classroom scenario

1.5 Thesis Outline

In Chapter 2, necessary background information and related work is outlined. The background work in terms of the challenges involved in opportunistic networking and social communities has been described in the first part. Also, the work related to developing a middleware, protocols for exchanging messages and service composition has been discussed.

In Chapter 3, the architecture of the middleware is described. This section elaborates how the different messages that are passed in the network are handled in each node. The middleware is divided into three parts namely the application layer, the device layer and the network layer. The application layer contains services that a node can to offer to other devices in the network. The device layer contains several modules each of which has the system software for generic service composition. The network layer has the logic to communicate without human intervention. In this thesis the Bluetooth is used for communication. The middleware architecture can be implemented using the Wi-Fi platform as well.

In Chapter 4, the importance of message handling in opportunistic networks is elaborated followed by the different types of messages that are passed during service composition. A format to differentiate these messages and also the various other fields corresponding to different types of messages is explained. A real-time example involving different types of messages is also described.

In Chapter 5, the implementation detail of the entire architecture that is developed in the Android operation system is explained. The architecture is developed and implemented by adapting the protocol described in chapter 3 and 4. The middleware thus developed is a pluggable unit and can function for any service composition with very minimal code change. A language translator application that would compose services opportunistically has been explained in detail. The device level functionalities involving each module have been elaborated. Finally the Bluetooth transfer module has been described. Experimental studies were performed by varying the number of nodes, lengths of composition, content size and so on and the times taken for those compositions were used to analyze the performance of the middleware.

In Chapter 6, the conclusion and the future work of this thesis is outlined

CHAPTER 2

BACKGROUND AND RELATED WORK

The rising popularity of smart phones and the robust demand to have more applications onto these smart phones has made resource management a huge challenge. Collaborating amongst nearby devices to share information available can help reduce resource constraints significantly. However establishing the network to discover and exchange information among the devices is beset with challenges. In Section 2.1 we review the current state-of-the- art of opportunistic networks and in Section 2.2 we discuss some of the efforts that are related to the work proposed in this thesis.

2.1. Background

2.1.1. Challenges in opportunistic networking:

Opportunistic networks are one of the most interesting evolvement in the recent years. They are accompanied by several challenges across various levels of networking. Chung-Ming Huang et al. [1] discuss about several research-oriented challenges that would occur in opportunistic networking. This paper aims at providing background information about opportunistic networks and its current status in research field. A high level view of the current challenges is described layer wise with respect to the protocol stack. The layers are divided as the network layer, transport layer, bundle layer and application layer.

The network layer issues are broadly classified as forwarding based and flooding based. In forwarding based approach the message is unicast to other devices while in flooding based approach, the message is broadcast. Unicast is done either to the destination device directly, which is called *direct transmission* or to a device that is close to the location of the destination node, called *location based.*

Unicasting can also be done to devices that have knowledge about the location or path to reach the destination node. Such type of unicasting is said to be *knowledge based.*

Although unicasting in general avoids message flooding and network congestion, it is accompanied with challenges as well. In *direct transmission,* the time taken to find the destination node is unknown and so applications that are delay tolerant or with limited delay tolerance cannot benefit using this scheme. The *location-based* transmission is proven to be an efficient communication pattern in comparison with the broadcasting strategies however they require an infrastructure to identify nodes that are relatively closer to the destination node. Constructing and updating such infrastructure in a network that is prone to continuous change in structure can be costly. Under the *knowledge based* approach several schemes are discussed.

Each of these approaches has oracles that collect different types of information such as topology, available resources, most frequently used message and so on. Based on this knowledge the choice of next device to forward a message is made smartly and thus reduces message overhead. Broadcasting is broadly divided as epidemic and prediction routing. Epidemic routing includes schemes that forward messages to all nodes in contact. This scheme increases redundancy of messages in order to make sure that a message is not lost and finally reaches the destination.

In prediction routing, based on a preformed prediction about a node's closeness to the destination node, messages are multi-cast to selected nodes as opposed to random flooding. This reduces redundancy of messages and network traffic significantly although there is extra cost involved in forming the prediction about location of nodes.

In the transport layer, as TCP requires robust end-to-end connection to guarantee reliable transmission, it does not scale well for opportunistic networks. Ramdass et al. [3], propose the '*Licklider Transmission Protocol*' (LTP), in which lost messages are retransmitted to guarantee reliability. In TLP, the bundle layer has the logic to store and forward messages in opportunistic network. Challenges in this layer involve factors such as forwarding messages to one node per group versus flooding to all nodes in a

group. Networks that have no end-to end connection can take forever for messages to reach from source to destination. Delay tolerant applications have only limited tolerance for delay. Delays in communication under disconnected networks can be longer than this tolerance for delay. Discussions about several variations of typical application level protocol like HTTP and SMTP have been suggested to make them suitable for opportunistic networks.

2.1.2. Opportunistic communities:

Pietilänen et al. [2] propose the concept of *temporal community*, in which each cluster of node that have high inter and intra contact times are classified into a temporal community. It has been found that these temporal communities have close relations to social communities that are formed in real life. It has been proven through experimental results that nodes that have high contact rates still serve to be less efficient nodes for data dissemination as opposed to nodes that belong to several social communities.

Nodes in a social network are divided as *social* or *vagabonds* [4]. *Social* nodes are those, which participate in a community on a regular basis. Those other nodes, which randomly participate in a community, are termed as *vagabonds*. Zyba et al. [4] show using experimental results that these *vagabond* nodes that are generally ignored in considerations actually play a vital role in opportunistic communications. It has been proven that for efficient message dissemination in a social network it is the number of nodes that participate that counts more than the social correlation that each nodes have in the network.

Allen et al. [5] consider the example of micro-blogging to identify interest groups and methods to communicate among them. Micro-blogging is a concept adapted from twitter where in information of very less spatial scope and validity is passed through mobile nodes. This scheme adapts a *push policy* where in information is pushed to nodes irrespective of their interest towards the message. During initial passes interest of neighboring nodes are learnt and is proven to result in less message over-head than the pull method.

2.2. Related Work

2.2.1. Middleware Architecture

Boldrini et al. [6] proposed a *context and social aware* middleware architecture, which collects context about nodes in the network in terms of user information like name, address, places visited, services maximum used and so on; in terms of service like understanding the type of contents each node is interested in order to make decisions about selecting nodes to pass contents; in terms of device context like battery power of the device, types of sensors nodes have and so on. Implementation of the middleware and the context sharing components has been done including the haggle architecture. Results show that with respect to devices that do not use the context and social aware middleware there is a 200% increase in successful content transfers and 99% reduction in network traffic.

A middleware to support reliable transmission in disconnected network has been designed and implemented using mobile agents by Carvalho et al. [7]. The Flex-Feed framework was developed to support military situations where reliability, robustness, and timely delivery of messages are extremely vital. At the application level Flex-Feed takes service request as inputs from users and in the implementation level it efficiently identifies and combines services in an efficient way. Based on the information provided by the client requesting for a service, the centralized or distributed '*coordinator*' constructs a path between the source and the clients termed as the *data distribution tree.* In this framework, complex requests are sent in the form of graphs with details about requests mentioned through edges and nodes.

Cameo [8] is a middleware platform aimed at making context-aware applications more robust in pervasive environments. The architecture is divided into the *Local Resource Management* (LRM) and the *Context Aware* (CA) framework. The LRM is responsible for interacting locally with the device's hardware and software while the CA layer is responsible for managing and distributing context aware information using forwarding protocols. Cameo begins by exchanging context information of neighboring nodes

through hash values. Nodes are then selected for transfer based on the interests analyzed through the hash values. A sample tourist application trying to exchange several contents has been developed over the android platform using the Cameo architecture.

To support mobile social networking, Pietiläinen et al. [9] developed a middleware that exploits the opportunistic contacts made by mobile nodes called MobiClique. Implementations of MobiClique use FaceBook paradigms to explain the process of mobile social networking using this architecture. Mobiclique periodically spans to check for neighbor nodes and exchanges complete profile information during the first exchange. During subsequent meeting only updates are transferred. If the neighboring nodes that meet are connected through the virtual social network, updates in both profiles are exchanged. MobiClique nodes have functionalities to either send unicast messages to other MobiClique users in its friend list or can multicast to people in a particular interest group.

2.2.2 Protocol for message exchange

The Lightweight Directory Access Protocol [10] is an application level protocol that defines a number of operational commands to access directory information of other nodes and runs over TCP/IP. The client sends session initiation command to the server. LDAP also comes with several other commands for information collaboration. Bind, search, add, delete, modify and unbind are few of those commands. Those applications that involve a sequence of reads and writes to the directory can make of this protocol as this involves very less overhead.

Jini [15] is a java-based protocol for registering and looking up services in a distributed environment. Every service in a network registers with a lookup service from where clients lookup to see if there exists one that it needs. *Discovery* and *Join* are the two commands that a service uses to discover a lookup service and join it and *lookup* command is the one that a client uses when it reaches the lookup service in search of a service.

In Service Location Protocol [16], the user is not required to know the name of the network host to identify a service. Instead it lets the user name the service along with a set of attributes to identify a particular service. SLP associates this description to the network host. In a comparatively large-scale environment, SLP uses directory agents to advertise services available in nearby clients.

2.2.3. Service Composition

Umair et al. [11] have developed a middleware for service composition that selects the low level services to compose high level services from the near by nodes. In this algorithm, a service directory is used to select to create a path of composition. Experimental studies were performed using mobility traces to prove that composing a service using several low level services is better than searching for an exact service match. Also, it has been proved that following multiple hops to reach a node is a better choice than waiting to reach a node in a single hop.

The service composition mechanism used in [12] is different from the usual request availability match mechanism. It classifies nodes hierarchically, with nodes that are abundant in resources falling into the highest category and nodes with relatively low resources falling in the lower level. Nodes that present in the higher level help nodes that are present in the lower level with computations of different kinds required for the composition.

MoSCA [13] is a middleware that provides services where service compositions acquired through other mobile nodes are transparent to its users. The context of other mobile in terms of location and services they have has been observed for a period of time and compositions are selected based on these predictions.

SpiderNet [14] is a service composition framework that aims at providing high quality services in the distributed environment. It follows a directed acyclic graph path to provide quality. Also, it caches selected compositions to support failure recovery in P2P communications.

CHAPTER 3

MIDDLEWARE ARCHITECTURE

In Chapter 2, we saw several work related to the middleware architecture that has been developed so far. In this chapter, we have designed generic middleware architecture for efficient service composition. The architecture that is designed is intended to support collaboration among devices in opportunistic networks. It is also extended to support sequential service composition under ad-hoc environments. There are several other factors that will affect service composition. For example the type of service to be composed, size of each service, length of composition and so on. With variation in services and service composition, the system to handle them will vary as well. However in this chapter, we develop a generic middleware designed handle such tasks as message passing, information exchange and service composition in opportunistic networks. It is to be noted that throughout this thesis the terms node and device has been used interchangeably. Also the term information and message has been used interchangeably and both refer to message packets exchanged during contact. Collaboration in opportunistic networks involves exchange of several types of messages. During an opportunistic contact, a device may transmit a message comprising its own context information and that of the other devices device it encounters in the network. The messages differentiated in order to avoid confusion in the network and also preserve them from getting lost. These different kinds of messages that will be passed can potentially flood the network. This network congestion will also lead to many unsuccessful service compositions as delay in delivery causes these messages to be dropped from the network after certain time to live period. Unless a robust system is there to handle these messages, the efficiency of the network will slow down as time progresses.

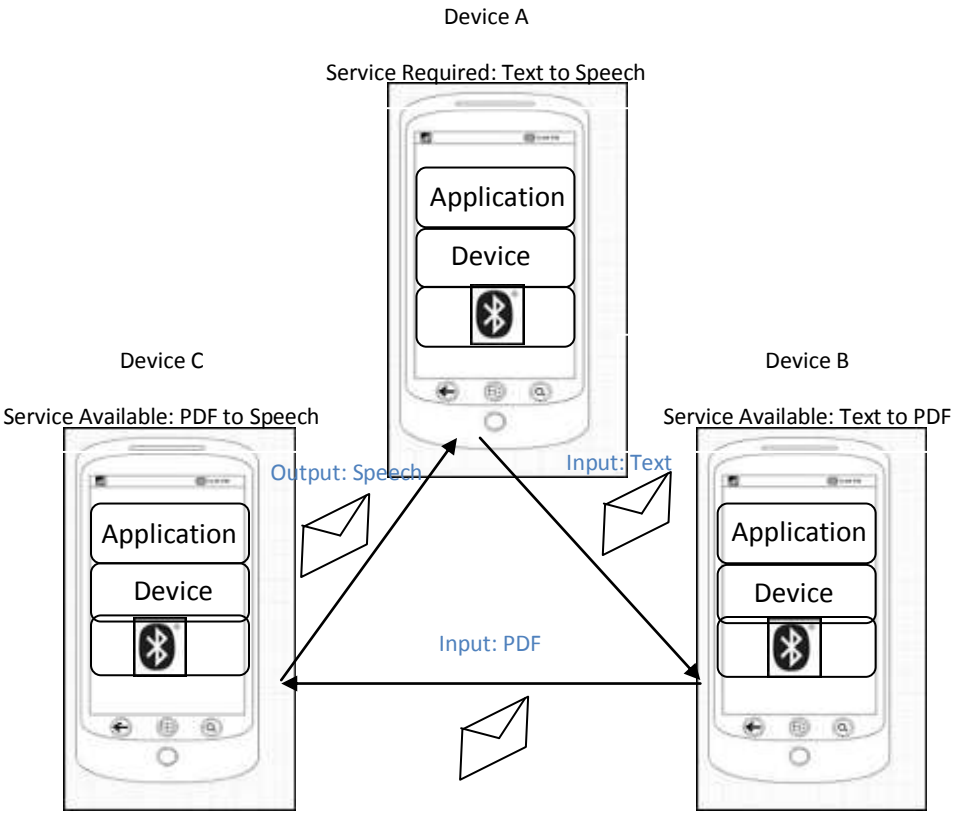

Figure 3-1 Overview of service composition

Figure 3.1 illustrates sample service composition among three devices. Each device has three layers: the network, device and the application. Device A makes a request to compose text to speech service. It is assumed that the low level services available for composition are Text to PDF and PDF to Speech. The service that A is looking for composition is a high level service which is made up of low level services Text to PDF and PDF to Speech in the same sequence. Hence the length of composition here is two. When A opportunistically meets B it sends the request message. On receiving this information B composes Text to PDF service that it has in its local device and forwards the result composed so far to device C that it meets opportunistically at a later time instance. Similarly C composes the PDF to Speech service and forwards it back to A which made the request. This marks a service composition complete.

3.0 Protocol for collaboration

The middleware protocol for collaboration has been established to ensure consistency in message exchange across the network. When two devices meet, messages are exchanged pair wise. Message exchange happens irrespective of whether the two devices that make a contact have or need any information from each other.

A transfer is marked successful only if the entire sequence of protocol is followed. Following such sequential process to exchange messages makes retrieving of lost messages easy. The protocol proposed in this section is similar to the *Lightweight Directory Access Protocol* (LDAP) [10].

As shown in Figure 3.2, every device is layered into the network layer, device layer and the application layer. Each of these layers is explained in detail in further sections. In this section their involvement in the exchange of messages is discussed. Prior to any transfer the application layer in each device informs its device layer with information about what it is looking for and what it has to offer for other device. When two devices make a contact they exchange the *start* message, which initiates the automatic Bluetooth transfer in both devices. This sets up the communication port between the two devices. On successfully establishing this platform to transfer messages, these devices exchange their authentication information for security and privacy purposes. This authentication information is forwarded to level where the actual authentication of the device occurs. This message in the sequence is referred to as the *bind* message. Once the authentication is completed, the exchange of actual information occurs. This information is typically the metadata about each service that exists in and is required by the devices that meet opportunistically as well as information about devices that it meets previously in the network. On receiving the metadata, each device updates this information with its already existing information in its local device. The device level then scans to see if there is any match between the required and available information between the two devices in the contact. The device that is looking for a service is called *seeker* and the device that offers a service is called the *provider.*

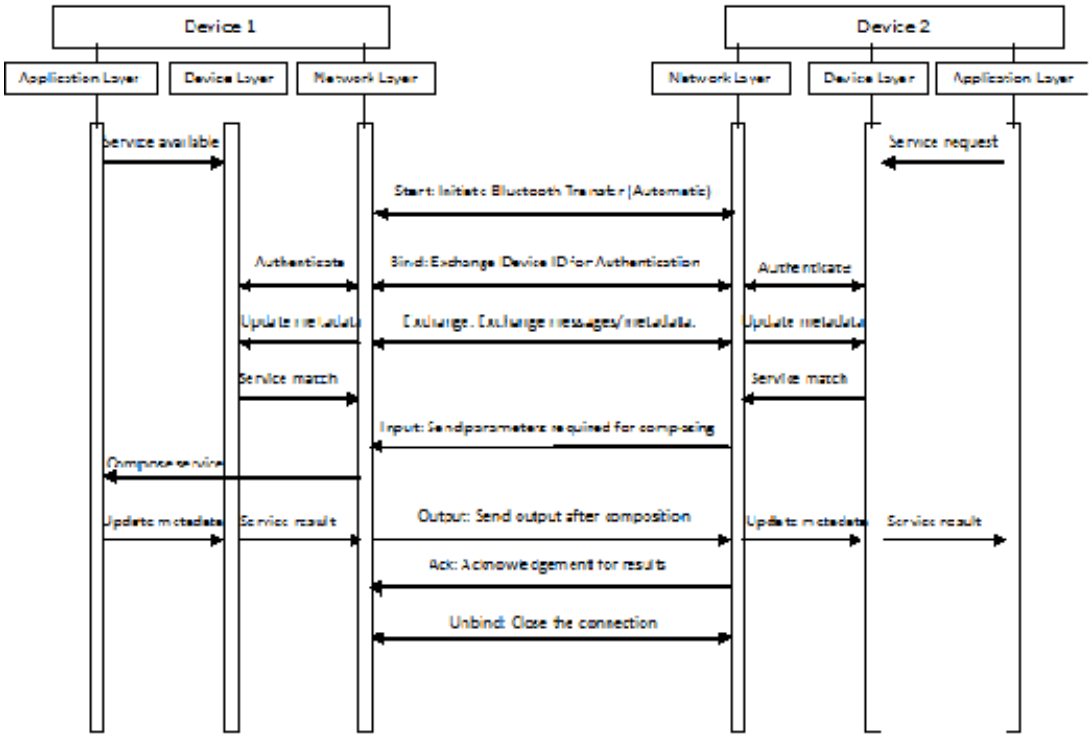

Figure 3.2 Protocol for Collaboration.

On finding such a match, the device that is looking for information sends the *input* message to the other device. This typically contains all the information that is required to compose the service. The input information is not sent along the metadata because this will make the message bigger in size and thus slower the transfer process. . However, if the input length is small it can be forwarded along with the metadata as well. This will reduce the number of message exchange for service composition.

The service provider then composes the service at the application layer. The metadata is now updated to account for the information that is composed now. The seeker will no longer require this service as the service has just been composed. The *output* message containing the result of the composition is sent back to the requesting device. On receiving this information, the device updates its

metadata locally the same way as the sending device. The result is then forwarded to the application layer that initially made the request

On receiving the result, the device sends back *ack* message to the composing device to mark successful information transfer. Following this both device exchange *unbind* message. The unbind message is not the opposite of bind message. Unbind marks closure of a connection and a successful exchange of information. Thus the delegation of composition to another device for composition is totally transparent to the user requesting the information. This is the typical handshake that happens during exchange of any information between two devices.

3.1 High-level Architecture

In figure 3.1 an overview of the architecture supporting service composition is shown. As explained in chapter 1, opportunistic networks are typically operated in wireless environment. Message transfer at network layer occurs either under wireless environments that are based on IEEE 802.11 or through network operated by Bluetooth. The device layer gets incoming information from the network layer and processes it depending on the kind of input it receives. The device layer and the application layer interact with each other from time to time to exchange information about services it is looking for and has to offer.

The different layers that form the middleware and their components are shown in Figure 3.3. The middleware is separated layer wise. The application layer, device layer and the network layer are the three different layers that form the structure. The application layer is the top most level, which lies on the device layer. This layer consists of low and high level services that form the application. This layer is explained in detail in section 3.2

The device layer forms the major component of the middleware and has all the logic required for service composition in it. It consists of the *security and privacy* (SP) module, *message resolver* (MR)

module, the *Service Manager (*SM) module, the *Service Locator* (SL) module, the *Service Composer* (SC) module and the *Message Merger* (MM) module. All these modules work together to update and maintain the different kinds of messages that are described later in chapter 4 in the most efficient way. Also these modules have the logic for service composition of services that are available in their local devices. Each of these modules shown in the device layer of figure 3.3 is described in detail in section 3.3.

The network layer refers to the underlying hardware and the implementation required for communication to occur successfully in an opportunistic environment. The network can be operated through Wi-Fi (802.11) or through Bluetooth. Section 3.4 describes the various constraints that are related to the network layer.

3.2. Application Layer

The application layer contains those applications that are available in a device. These applications are made up of services. The application layer updates the device layer with the services it is ready to offer and the services it is looking for from the network. This layer also helps the device layer in composing available services. Applications are composed of one or more services. The structure of different services and its variations are discussed in the section below.

3.2.1 Services

In addition to services that form applications available in their device, they also have services that they are ready to offer to other devices. As discussed in Section 1.2 applications are made up of one or more of low or high level services composed in a particular sequence. The number of low (or high) level services that completes a composition varies from application to application. This is referred to as the length of services in a composition.

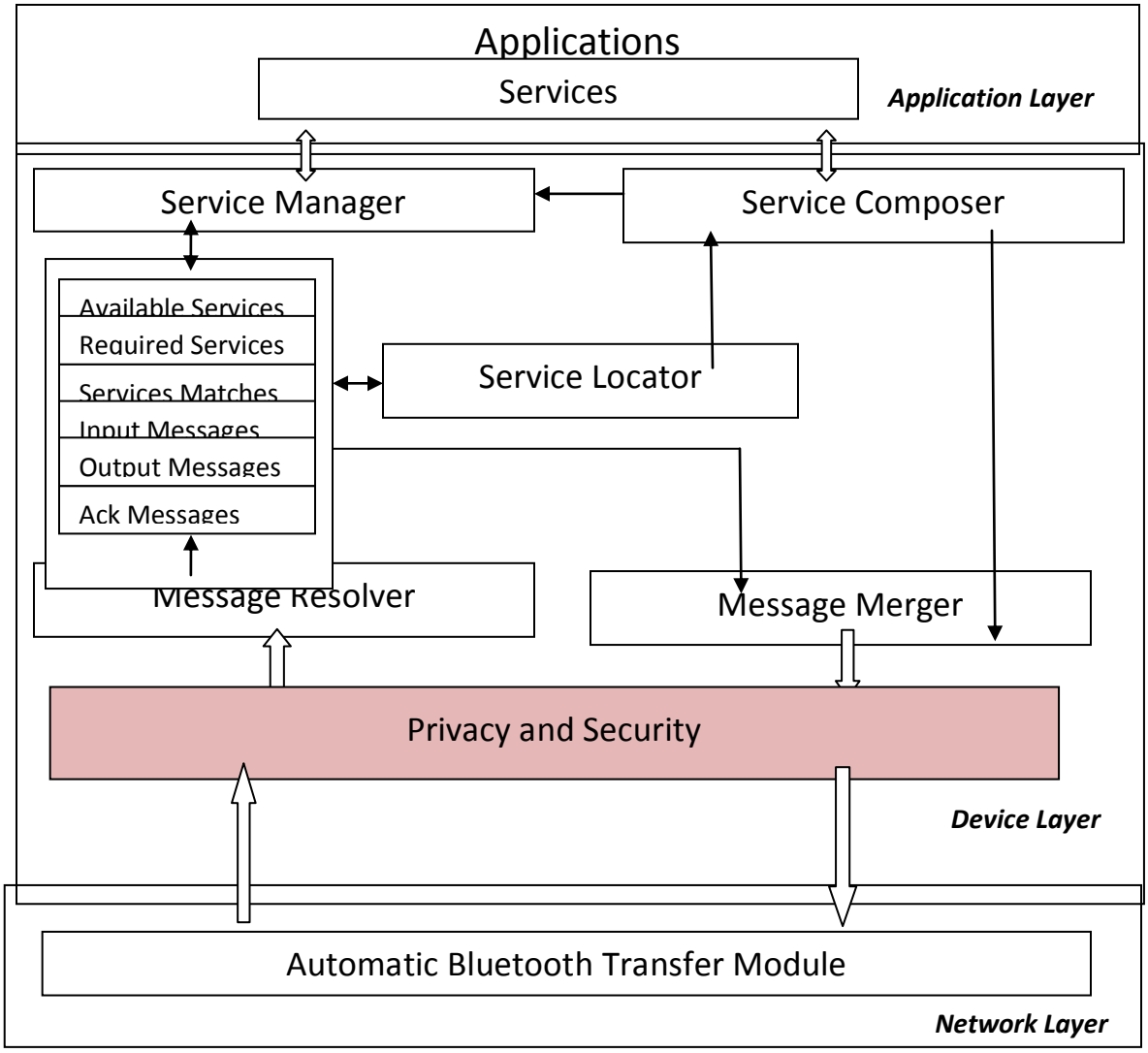

Figure 3-3 Architecture.

Table 3.1 shows service composition involving different length (number) of services required to complete a composition. In real time, the number of services that will be involved to mark a composition complete will vary from application to application. Table 3.1 shows a sample language translation

application involving different lengths of services in its composition. Let us assume that the low-level language translation services available in the network are English to Spanish, Spanish to French, French to German, German to English.

Table 3-1 Service compositions with different length of services.

High level service to be composed	Individual services involved in composition	Length of Service
Spanish to French	Spanish to French	1
French to English	French to German, German to English	2
English to German	English to Spanish, Spanish to French, French to German	3

The services module initially informs the SM module about the services that a device is ready to offer to other devices in the network for composition. Services might also be available for composition at a given time instance and another time instance the user (device) might choose not to offer it. Thus the services module constantly updates the SM module from time to time with the status of services available (and not available) for sharing. This module is explained in detail in section 3.3.2.

Finally, the services module also helps the SC module with the logic for service composition. When a device has a request to compose a service that it has agreed to offer for another device, the SC module triggers the services module to get the logic for composing that part of the service. The SC module is described in detail in section 3.3.4.

3.3. Device Layer

3.3.1. Message Resolver

When device B receives a message from device A, the message contains a combination of services that device A has and is looking for. Before Device A makes an opportunistic with device B, it would have met other devices. Details about services these devices have and is looking for is also passed on to device B through device A. Additionally these messages will also contain information about the required and available matches that the service locator finds. There could also be messages containing results of partially or completely composed services. Thus this incoming message needs a resolving agent to resolve these different types of messages for further processing. The message resolver has the logic to resolve these messages to different category from where the service locator and service manager carry on further processing.

3.3.2. Cache/Service Manager

From the time a request for composition is sent into the network, the message is passed across the network through several devices. Thus the service manager module has the logic to check and see if there are messages existing in the device beyond their time to live or away from the specified spatial location and drops them. Also there could scenarios where the request to compose has already been met or that the required service cannot be found in the network. The service-managing module removes these messages as well. As mentioned in section 3.3 there can also be scenarios when the application layer decides not to offer a particular service anymore or a new service for composition. The service manager module also handles such updates.

3.3.3. Service Locator

On receiving a message after the message resolver resolves a message the service locator checks to see if there is a match between the lists of services devices are looking for and list of services the devices have to offer for composition. If there is a match, a message is created and is sent to the

message-merging module for suitable merging. If the service composition involves the device by itself (either offering service or requesting service) the control is transferred to the service-composing module where service composition actually occurs.

3.3.4. Service Composer

On receiving service composition request from the service locator, the service is executed on the local device and the results are sent to the message-merging module for further processing of the message.

3.3.5. Message Merger

Every device will cache information about the services it has and it is looking for and services that other devices are looking for and has to provide. On receiving a new message the service locator and service manager checks to see if there can be any service composition or updates in messages. The result of service composition is ready to be sent to other devices as well. Once the processing is done on the incoming message there are new updates ready to be sent to other devices. This new information now needs to be merged with the outgoing message. The message-merging module has the logic to update the outgoing message with new information.

3.3.6 Security and Privacy

As mentioned in the handshake protocol, before any exchange of actual information, authentication of device happens at both the devices that meet opportunistically. This module is responsible for identifying and differentiating a valid node from other adverse nodes. Privacy of the node refers to information that a node decides to share (or not) with other nodes in the network for example, a node might consider its location a private information and may chose not to share it with other nodes. This module handles filtering of information that a node considers private from being sent out in the network. The algorithm and implementation of security and privacy comes in a huge variety of options and is beyond the scope of this thesis. The design and implementation of this module is beyond the

scope of this thesis.

3.4 Network Layer

The network module has the underlying logic to communicate the data from each device over to the network for further composition of service. Since opportunistic contacts are not completely predictable and such contacts don't often have a lot of inter contact time the exchange of message should be as automatic as possible. Having human intervention can slow and also reduce the number of exchange as many such possible exchanges might go unnoticed or might be too late before the response occurs from the user.

3.4.1 Bluetooth

In this thesis, the network connections and message exchange happens through Bluetooth. It has been assumed that the initial pairing between two devices is done manually. The device identification information of receiver is set in the sender. Once the manual pairing of device is done, the exchange of messages occur automatically and involves no human intervention. The automatic Bluetooth exchange module has two parts namely the sender and the receiver. Since in this thesis, message passing is always bidirectional during every opportunistic contact, the sender and the receiver modules reside in both nodes that make the contact. The automatic Bluetooth module is triggered every time a message is sent or needs to be received in a particular node. The automatic Bluetooth module was developed as part of the PICO lab work.

CHAPTER 4

MESSAGE FORMAT

This chapter discusses about the different kind of messages that will be available in the network and the way they are classified. Section 4.1 describes how messages are carried within an opportunistic network. Different types of messages are described in 4.2 and a sample file with different kinds of messages is discussed in 4.3.

Opportunistic networks are characterized by lack of end to-end reliable connections. In such environments, handling messages become much more difficult and all the more important because unless there is a robust message handling system communication will not be successful. Also unless a suitable message-updating algorithm is implemented, the network will be flooded with repetitive and/or outdated messages.

There could be messages that are being passed in the network which are not updated due to several copies of the same message being dispersed along the network and they would have been updated at a different node. For such reasons messages, should be time stamped at the time of creation or any modification.

Let us consider a scenario as in figure 4.1 where Node A wants to deliver a message M_{AE} to node E. At time instance T_1 when A meets B it passes on message M_{AE} and when B meets C and D at time instance T_2, it passes on message M_{AE} that it received from A to them as well. Now at node C let us assume that this message is updated (M_{ACE}) and is passed on to node D. Now node D contains both M_{AE} (out dated copy) and M_{ACE}. Now copies of $M_{AE \text{ and }} M_{ACE}$ are distributed in parallel in the network. Thus when node D encounters node E in time instance T_3 both versions of messages reach the final

destination. Ideally, M_{AE} should be marked out dated and be removed from the network and only copies of M_{ACE} should be distributed in the network making the network consistent with the messages and also remove redundant messages that would flood the network.

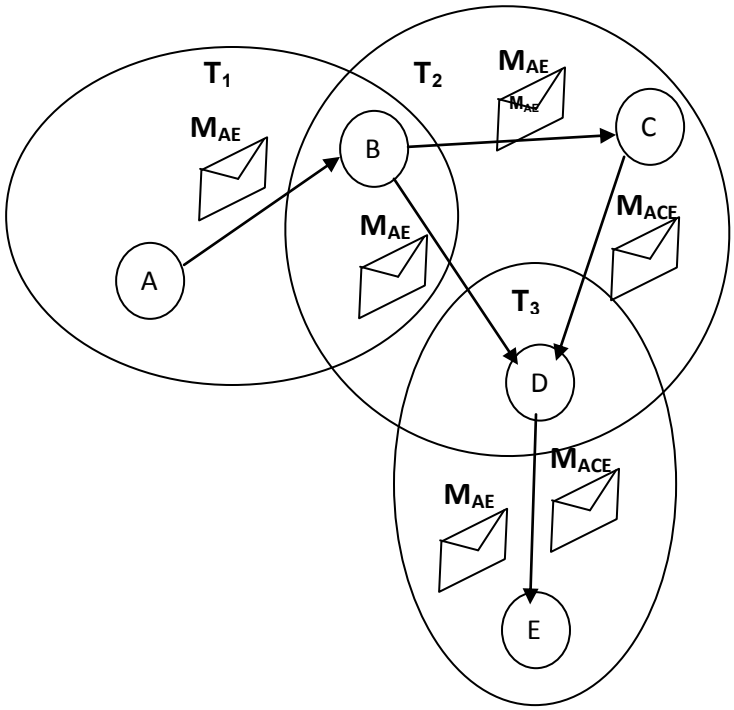

Figure 4-1 Message flooding and inconsistency in opportunistic network

Nodes play an important role in connecting the network and passing along information in the right order and time. The messages that these nodes pass can be of a number of types. These different types of messages are discussed in detail in 4.2 but depending on the kind of the information they are carrying these messages would vary as well.

4.1. Message Propogation

In a network with no direct end to end connection it is very easy for information intended for certain node to get lost. This makes it more important to design a robust mechanism to support delivery of information acquired from various nodes to reach destination nodes that could be several hops away.

For example in Figure 4.2, let us consider a network where A has a message to be delivered to C (M_{AC}), B has a message to be delivered to D (M_{BD}) and C has a message to be delivered to D (M_{CD}). Also, let us assume that A meets B at time instance T_1 and passes on message $M_{AC.}$ Now when B meets C at time instance T_2, unless B passes on M_{AC} which it received from A along with its own message M_{BD} to C, M_{AC} would have got lost at B and thus would never reach C- the intended destination node. Similar is the case at node C when at time instance T3 it meets node D. Node C in addition to receiving message $M_{AC,}$ it should also pass on message M_{BD} in addition to its own message $M_{CD.}$ Finally node D should successfully receive M_{CD} and $M_{BD.}$ This marks a complete and successful message propagation scheme.

The above-described example is how opportunistic network could be in its simplest form. In reality at a given time instance there would be several nodes to which such messages would be broadcast. For example at time instance T1 let us assume B was in a classroom. All the students in classroom would have received messages received by B from A. Thus several copies of M_{AC} are now dispersed in the network. Now if B at time instance T2 goes to a gym with many people around, several copies of M_{BD} and M_{AC} is passed on as well. Similarly node B also receives messages from this network. This makes the network very complicated and could lead to messages not reaching the destination node.

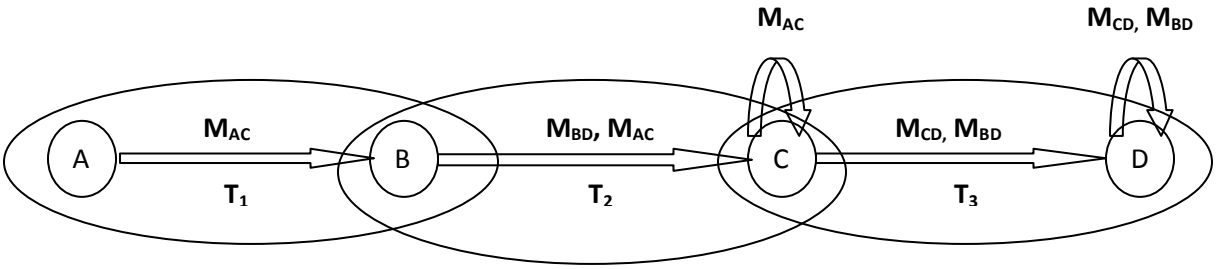

Figure 4-2 Opportunistic networks at different time instance

4.2. Message Types

Communication in an opportunistic network happens through message passing. In a service composition environment messages are of several types. It is important to mark a clear difference between these various kinds of messages to avoid confusion about what kind of information they carry. Section 4.2.1 gives a high level over view of these message types.

4.2.1. Message Identification

In addition to sending request messages of services, a node wants from the network or messages about services they have to offer, they also act as relay nodes to forward messages they have received from the network. Such messages could be request or response message sent by other nodes in the network. In addition there could be messages that indicate a match between the required and available services through information acquired from the network. Also when devices with a match between the required and available service encounter they establish an authentication hand shake and exchange service input and output message. An acknowledgement message confirming successful service composition is also a kind of message that circulates among the network.

Table 4-1 Message types along with their Message ID.

Message ID (MID)	Message Type
0	Available
1	Required
2	Match
3	Input
4	Output
5	Ack

To differentiate these messages from each other our scheme attaches a message ID to the header of each of them as explained in Table 4.1. The entire structure of the message is explained in 4.2.2. In this section the message ID is explained in detail.

When a message indicates a service that is available in a device DID it is prefixed with message ID 0. When a device encounters a message prefixed with 0, it understands that the service details included in the message are of those ready to be offered for composition. Similarly those messages that indicate services that it is looking for from the network are attached with message ID 1. Through opportunistic communication if a device identifies a match between two devices that offers and seeks the same service, they are classifies as message ID 2. When devices with a match in service encounter each other, the device, that requires a service, sends a message with the required inputs and is identified by message ID 3 and similarly after executing the service it returns back a message with the results and are marked with ID 4. Finally to update and acknowledge other devices in the network an acknowledgement message with ID 5 is sent across.

4.2.2. Device Identification

As discussed in Section 4.1, a device in addition to forwarding messages also acts as a relay node in forwarding message it has acquired from other nodes. These messages have to carry information notifying the corresponding device ID from which a message originates. Thus a column indicating this is included in the message format. In general, devices have a unique identification number (UID) by which they are identified. These UIDs are mapped to convenient pseudo names for suitable identification. The detail of how these devices are identified is discussed in detail chapter 5. Device identification is generic information that is included along all types of messages.

4.2.3. Timestamp

As illustrated in Figure 4.1, over time the network may have acquired messages that are outdated. Unless these messages are updated in a timely fashion, the network will get flooded with

messages that will lead to redundant and less efficient service composition. To avoid such inconsistencies each message is generated with a field that contains information about the time at which it is created. When a message is altered at a different node with updated information the timestamp is also updated with the time at which the update happens. When two messages with information about the same service are received at a node the message with the latest timestamp is considered and information is updated. Also there is a pre-fixed time to live through which a message is considered valid. At every node when a message is received, its timestamp is compared against the time to live. Periodically, information present inside a node is validated for time to live and updated accordingly. Any service request message present beyond time to live is considered not available in the network and ignored. The timestamp field is also generic across all type of messages.

4.2.4. Message structure

Based on the different message IDs mentioned in 4.2.1 the structure of the message differs. Along with the message IDs these messages contain other suitable information required for composition. In general, all these messages contain information about the type of message (4.2.1), the device from which the message comes from i.e. the device ID (4.2.2) and the time at which the message is created (4.2.3). Information that is specific to each type of messages is discussed in the further sub divisions as follows.

4.2.4.1. Service request/available message format

Messages that are sent with information regarding services that devices are looking for or ready to offer carry similar information and come under this category. This section elaborates those fields that are common and different between these two types of messages and their purpose. Figure 4.3(a) and (b) shows the format of a typical service request/available message. The MID as explained in Section 4.2.1 for service availability is 0 and for service request is 1. The DID and timestamp is the same as explained in Sections 4.2.2 and 4.2.3. A service availability message contains the name of the service that a node is

ready to offer. Similarly the complete service required column for a service request message contains the name of the service that a node is looking for. The service request message format has an additional field that specifies the next low-level service that it is looking for from the network apart from a field that specifies the complete service that a node initially requested for.

4.2.4.2 Service match message format

Messages that contain information regarding a match between two or more devices that it is looking for and ready to offer the same service is called match messages. This section elaborates the fields that are required for gathering information regarding such messages. As explained in Figure 4.3(b) the message format contains MID with value 2 and Current Time field as explained in section 4.2.1 and 4.2.3 respectively.

The device identification field is also much similar to what is explained in section 4.2.2. As the matched message must contain information about the seeker and provider, the DID field now contains *DID of available* and *DID of required.* Additionally this message also contains a field with the name of the service that matched.

4.2.4.3 Service input/output message format

When two devices with match in service request and availability meet, they exchange information that is required for service composition. The details of the handshake sequence are explained in Chapter 3. This section elaborates the format of such messages. As shown in Figures 4.3(c) and (d), the message format includes MID, DID and timestamp information like in all types of messages. The service input message contains the list of inputs that are required for service to be composed on an opportunistic device and similarly once the service is composed the service output message contains the list of outputs that are required to mark service composition complete. It also contains the name of the service that the input and the output list correspond to. The service output message format has an additional field that

specifies the initial overall service that the node initially requested to associate the final service composition result with.

MID (0)	DID	Available Service	Current Time

(a)

MID (1)	DID	Next Required Low-level Service	Complete Required Service	Current Time

(b)

MID (2)	DID of Avail.	DID of Req.	Required/Available Service	Current Time

(c)

MID (3)	DID	Required Service	Service Input(s)	Current Time

(d)

MID (4)	DID	Service	Service Output(s)	Complete Required Service	Current Time

(e)

MID (5)	DID of Req.	Service Composed	Current Time

(f)

Figure 4-3 (a) Service available (b) Service request (c) Service match (d) Service input (e) Service output (f) Service ack.

4.2.4.4 Service acknowledgement message format

Once a service is executed, the network should be pruned with the latest information. Messages existing in the network prior to service composition need to be updated with the latest time stamped information. For such updates to be done, a service acknowledgement message is sent once a service composition happens successfully. Information on what happens once the acknowledgement message is received was discussed in Chapter 3. MID of service ack message is 5 as described in previous sections. The format also includes information about the service that is composed along with the device for which the service was composed and time of creation.

In the following section we describe a sample service composition request and explain how different types of messages are generated, updated and handled in real-time.

4.3. Service Composition Reflecting Message Types

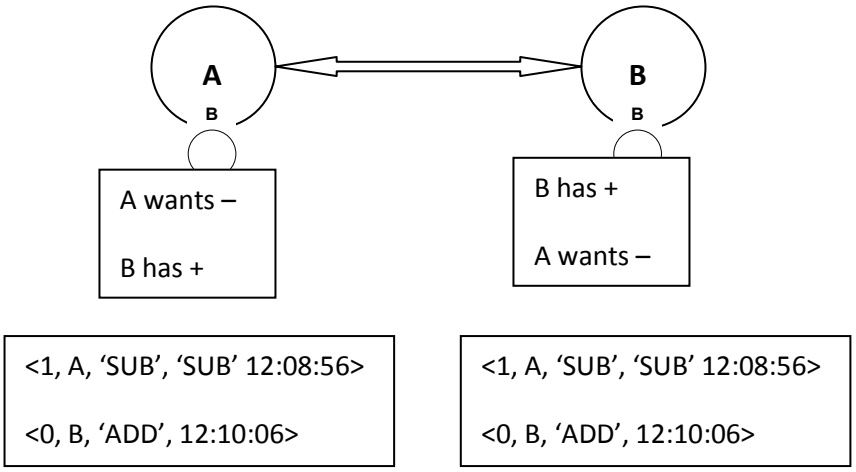

Figure 4-4 Device A and B opportunistically meets at time instance T1.

Let us consider a network where people are looking for different mathematical services such as addition, subtraction, multiplication and so on. For the sake of understanding let us assume that each device has and is looking for one or more of these services. Let us assume A is looking for subtraction (SUB), B has addition (ADD) service to offer the network and C has subtraction to offer. When device A

and B meet, this information is exchanged as shown in figure 4.4. Now in addition to its own information, A and B contains information about each other as well.

The format of service request/available message is also shown in the Figure 4.4 as explained in 4.2.4.1. The message ‘A is looking for subtraction’ is in the format ‘<1, A,’SUB’,’SUB’, 12:08:56>’ where ‘1’ is the MID, ‘A’ is the DID, ‘SUB’ is the name of the next low level service it is looking for, second ‘SUB’ is the complete high level service it is looking for and ’12:08:56’ is the time at which this message was created. Similarly, ‘B has addition’ is represented as ‘<0, B, ‘ADD’, 12:10:06>’ where ‘0’ is the MID, ‘B’ is the DID, ‘ADD’ is the service it has to offer and ’12:10:06’ is the time at which the service is created.
This information after contact and exchange is represented as <0, B, ‘ADD’, 12:10:06>; <1, A, ‘SUB’, ‘SUB’, 12:08:56> and <0, C, ‘SUB’, 12:20:38> and the details are just as explained for Figure 4.4. In addition to these, after meeting with A at time T1, B now has information about A, which is looking for SUB. At T2 when B meets C, B and C generate a match message indicating that A is looking for SUB and C has SUB. This is represented as <2, C, A,’SUB’, 12:30:00> as explained in 4.2.4.2. Here ‘2’ represents MID, ‘C’ represent DID of device in which the service exists, ‘A’ represents DID of device which is looking for the service, ‘SUB’ is the matching service and ’12:30:00’ is the time at which the match was generated.

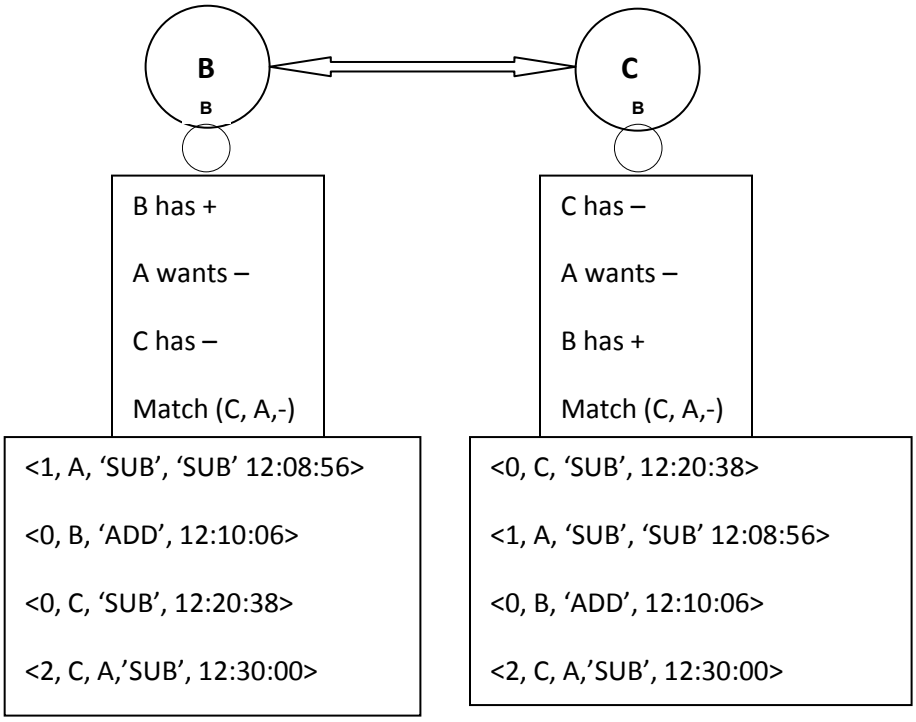

Figure 4-5 Device B and C opportunistically meet at time instance T2.

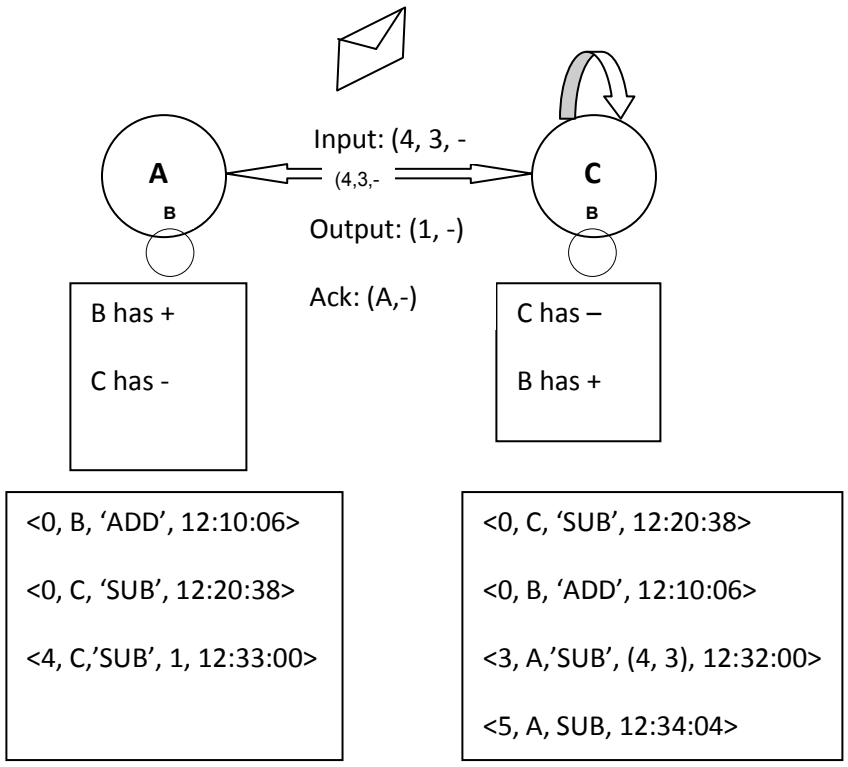

Figure 4-6 Device A and C opportunistically meets at time instance T3.

4.4 Service Discovery in opportunistic environment

Service discovery refers to the process of identifying the availability of services in the network. Since the network is prone to frequent changes with respect to the availability of nodes, it becomes a tougher challenge to identify the different services offered by each node. Also from time to time, nodes offer new services and might decide not to offer a service anymore. Such scenarios add to the difficulty of service discovery under such environment. The lack of end-to-end path makes it impossible to identify all services that are required for the composition before the request is sent.

Exchange of messages to identify the services offered by each device in its neighboring network can involve a lot of message passing in the network and will be time consuming. Maintaining such service

discovery learning oracles can become a very costly affair as well due to the every changing shape of the network as discussed in Chapter 2.

In this architecture, service discovery occurs concurrently with service composition. As discussed in the previous sections, messages that carry MID 0, refer to those services that are available in device DID. Each message comes along with a timestamp referring to the time at which the message was created. After certain time period these messages are dropped from every node and considered not available. If the node continues to generate messages referring to availability of a service that was previous dropped, it still exists in the network with updated timestamp of creation. If at any point of time a node decides to offer a new service, the information is passed on using similar messages to nodes that it encounters opportunistically. Such service availability message are merged along with other service composition essential messages and thus does not involve any extra significant cost for advertising the services. On receiving such a message, the service resolver resolves messages that refer to availability of a message. It checks to see if there are services that are available beyond their time to live or adds messages if there are new services the application layer decides to offer. Also the service matcher checks to see if there is a match between messages that refer to availability of a service and messages that refer to services that are required by any device.

CHAPTER 5

IMPLEMENTATION AND EXPERIMENTAL ANALYSIS

As discussed in Chapter3, the different layers that make up the system are the application layer, the device layer and the network layer. The device layer contains the logic to support the different messages passed in the network discussed in chapter 4. The device layer developed must be robust to support the different types of transitions the messages will undergo in the network and also must support any type of service composition request it receives from its application layer.

A real time service composition has been developed as part of the thesis work. The middleware along with the message formats have been implemented to make sure that it supports any kind of service composition. With such generic middleware developed, any application that be composed from low level services can utilize it in an opportunistic network with minimal code changes.

Figure 5.1 shows the block diagram of how the middleware architecture acts as a pluggable unit for any kind of application that can be opportunistically composed. Any application that can be composed of one or more low level services uses the middleware to compose them from the network. Thus the middleware developed is the same for any type of applications. As shown in the diagram let us assume there is an application 1 that can be divided into *n* low level services identified in the network. For this application to be composed opportunistically, it needs to implement the middleware along with its components on it device. Such changes from one application to another involve very less code level changes. Also the system supports several applications to be composed at the same instance.

Using the service composition module developed, further experiments were conducted to research success rates in opportunistic networks. In Section 5.1 a sample application level module that was developed is discussed. Section 5.2 discusses the evaluations in detail.

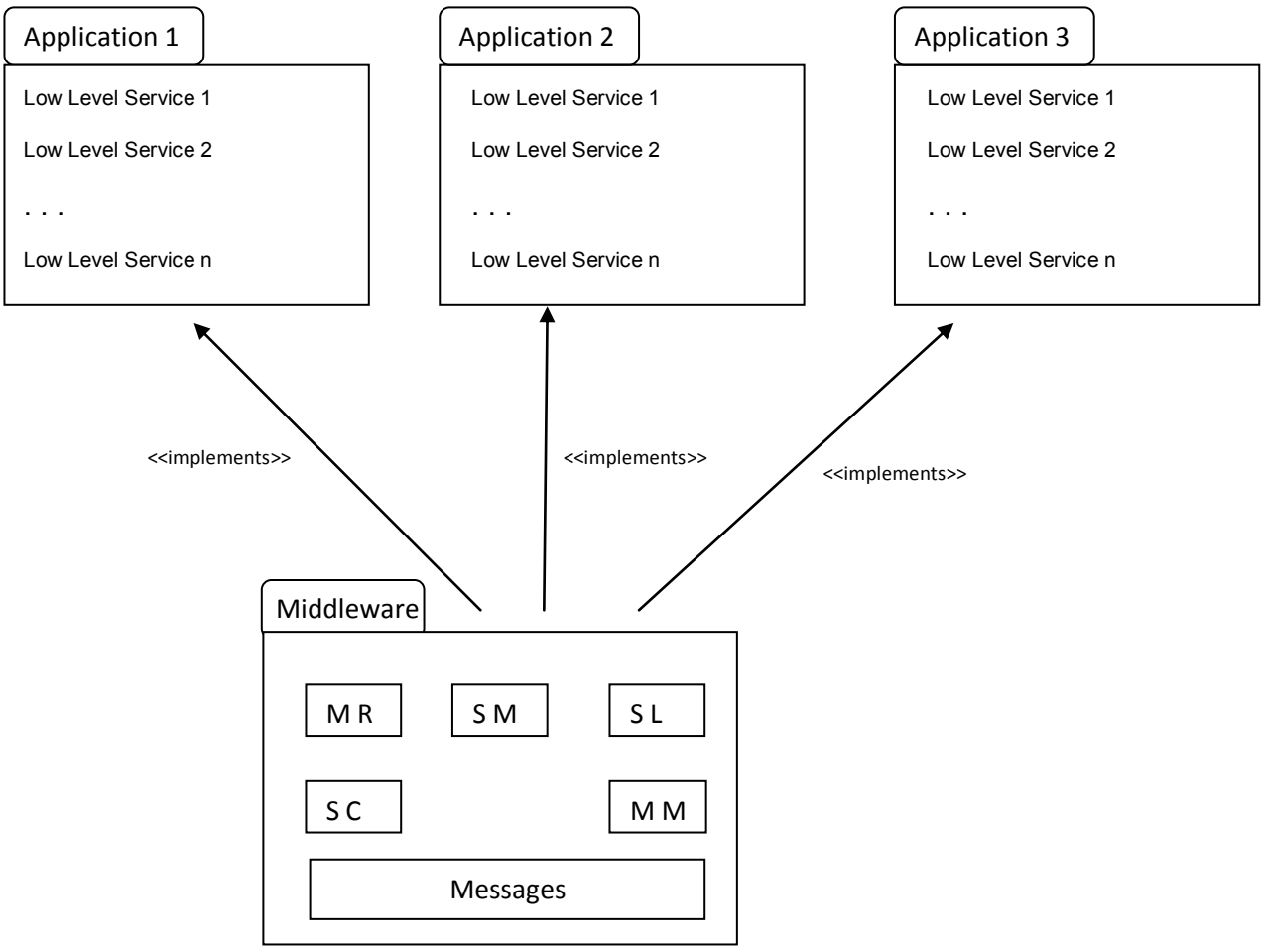

Figure 5-1 Generalized middleware with various pluggable applications.

5.1. Service Composition Application Details

For experimental and analyses purposes, a Bluetooth language translator application was developed using the android 2.1 update 1 operating system. A device making a translation request does not necessarily require the translator to exist on its device. When a request to language translation is

made, the request is analyzed and decomposed into suitable low-level language translators that will be available in the network. The request is then forwarded via Bluetooth by the automatic Bluetooth module. On receiving the service composition request, the middleware has the respective modules to check availability of service, compose, merge with other request, result messages and forward it for further composition. Several experimental results are discussed in Section 5.3.

5.1.1. Assumptions

In the application developed few assumptions have been made. Each device is assumed to be having the knowledge of different low level services that exist in the network before service composition requests are forwarded. Thus when a service request is forwarded, the device computes the ways through which the service can be composed and includes the first hop composition before forwarding the request message. The following sections discuss the individual and over all information that the devices have.

Four Motorola devices were used during the development of the application. The devices are named A, B, C and D for easy identification. Table 5.1 shows the various services each device has and is looking for from the network.

5.2 Experimental Analysis

In this section, several experimental scenarios have been discussed along with their relevant results and analysis. Each experiment was carried out by varying just the parameter to be analyzed and by keeping the other parameters constant. Each of these scenarios is explained followed by their experimental results and suitable analyses. The experimental studies include the following scenarios: varying service composition lengths; varying content size; varying number of nodes; varying services with same length; parallel service requests;

Table 5-1 Sample overall and device level details in an opportunistic language translator setup.

Available low level services in the network: E-S, S-F, F-G, G-E Abbreviations: E-S: English to Spanish, S-F: Spanish to French, F-G: French to German, G-E: German to English.

	A	B	C	D
Looking for high level service:	E-G	S-F	F-E	F-S
Decompose to low level service:	E-S S-F F-G	S-F	F-G G-E	F-G G-E E-S
Length of Composition:	3	1	2	3
Available low level services in each device :	S-F G-E	E-S F-G	E-S G-E	S-F F-G

5.2.1. Varying length of services required for composition.

To analyze how the times taken to compose services vary with difference in service length, experiments were conducted by composing language translation services of length one, two and three. Four android devices were used for this experiment where every time request to compose is generated by just one device. Length of service is increased by one and ten such trials were conducted to measure the average and standard deviation each time.

Table 5-2 Average and Standard Deviation varying service lengths – Time taken in seconds.

Services Length	Average	S.D
1	22.6	3.83551
2	47.7	4.66786
3	63.3	4.02216

Table 5-2 shows the average and standard deviation of time taken in seconds to compose services of different length. From the table we see that the time taken to compose a service doubles every time with increase in composition length. However, the time to compose these services will vary significantly with the service that is being composed and also the availability of services in neighboring nodes as seen in section 5.2.3.

5.2.2 Varying number of nodes for composition.

Experiments to understand the time taken to compose services with varying number of nodes is conducted. The same language translation service of length two has been composed over an ad-hoc network with length two, four and six. Ten trials were repeated for each of these scenarios to obtain the average and standard deviation.

It is observed from Table 5-3 that the time taken to compose a service increases with increase in the number of nodes in the network. This observation is opposite to the belief that more nodes in the

network make opportunistic computing faster and successful. It is to be noted that in the experiment it was assumed that the services that are required for the composition are available in the neighboring nodes. However in reality lesser the number of nodes, lesser is the probability of finding the required service for composition.

It is to be noted that with less number of nodes, the deviation is observed to be high. This is because the automatic Bluetooth transfer module in a node is engaged with less time interval for sending and receiving messages and this leads to inconsistencies.

Table 5-3 Average and Standard Deviation varying number of nodes – Time taken in seconds.

No of Nodes	Average	S.D
2	13.1	6.43687
4	47.7	4.66786
6	68.2	3.8239

5.2.3. Varying services of same length

Table 5-4 Average and Standard Deviation varying services of same length – Time taken in seconds.

Services No	Average	S.D
1	47.7	4.66786
2	21.8	6.92499
3	30.4	5.6999
4	62.2	9.98666

Different language translation compositions, each of length two was carried out to analyze the time taken to compose different services of same type and same length. Four Motorola devices were

used for this experiment and each time one device sends request. Four different language translation services of length two were analyzed and each of them was repeated ten times to obtain average and standard deviation.

Results of these experiments can be observed from Table 5-4. From the table we see that although the services were of same type and same length, the time taken to compose them differs from service to service. This difference in time for composition is attributed to the availability of services in the devices. Conditions such as all services available in the same device, services available in the same device making the request can hasten the service composition. While if the service required is in different devices or is not available in the network the time taken to compose them will get delayed or composition will be marked unsuccessful.

5.2.4 Varying size of content to be composed

The size of the content to be composed is varied and experiments have been conducted to analyze the time taken for service composition. Similar to the previous experiments, four Motorola devices were used to compose language translation service of length one. Each time the content to be translated is varied to analyze the time taken to compose service of different content size. In the initial service composition a string input was used to do the translation. The time taken to do the translation in each device is negligible as compared to the time taken for information to be transmitted from one device to another via Bluetooth.

Table 5-5 Average and Standard Deviation of varying content size – Time taken in seconds.

Content	Size of Content	Average	S.D
String	34 bytes	22.6	3.83551
Paragraph	1875 bytes	23.4	3.4144
Page	9380 bytes	22.9	4.0098

From Table 5-5 we note that the time taken to compose service with contents varying between a string and page has no significant change in time. It has been noted through experimental study that the time taken for a content of up to 20 pages has no significant change in composition time. For every 20 pages the composition time increases by one second of transfer time. The composition time at a local device is still insignificant.

5.2.5 Parallel service composition.

In the experiments conducted so far, the request to compose service was initiated by one device while all other nodes work towards composing the service requested. However, in reality almost all nodes participating in service composition will have service composition request. Thus in addition to helping other nodes compose services, each node may have its own service to be composed. In this experiment, four devices make service composition requests in sequence. Each service request is of length varying from one to three. Each device has two low level services available in its device for composition. Ten such trials are run to obtain the average and standard deviations of the time taken for each of those service requests to be composed. The average and standard deviation of the time taken for the all four requests to be entirely composed and the time taken for the acknowledgement of compositions to reach to mark the composition complete is also measured.

From Table 5-6 we see that time taken for services to be composed with parallel request being composed is more or less the same as the time take for those services to be composed when only one device requests for composition.

Table 5-6 Average and Standard Deviation of parallel requests – Time taken in seconds.

Services Length	Average	S.D
1	49.8	4.41714
2	36.7	3.653
3	61	3.3665
Complete composition	121	3.74166
With ACK	134.8	6.14275

5.2.6. Simulation of opportunistic environment.

To simulate an opportunistic environment, Scenario 5.3.5 is repeated with a change in the availability of nodes for composition. The choice of selecting devices to exchange message is randomized. Every node randomly selects one of the other nodes to exchange message each time. Also, each device is set idle for random number (between one and five) of minutes. This ensures that each node meets another random node at least once in five minutes. Such randomization might cause service composition to fail because of unavailability of services to be composed or delay in availability of services. Such situations are common in opportunistic network as well. From Table 5-7 we see that the time taken to compose the same service over a simulated opportunistic network varies significantly.

Table 5-7 Simulating opportunistic contacts - Time taken for 4 parallel requests.

TRIAL	TIME TAKEN IN MINUTES
1	23
2	44
3	26
4	42
5	35

CHAPTER 6

CONCLUSION AND FUTURE WORK

In this thesis, a middleware to support service composition has been designed, developed and implemented. Efficient message passing techniques have been adapted to combine service discovery and requests messages of all devices that a particular device encounters in the network. The middleware has been implemented with an intention to support any kind of service composition. A language translation application has been implemented using the middleware architecture. Several experimental studies were also conducted to test the performance of the middleware using the language translation application over the generic middleware.

Although an opportunistic environment was simulated to test the performance of the middleware, it would be ideal if users around the neighborhood to test the performance of the middleware carried devices. The middleware proposed is designed to be generic and support any kind of service composition but it has been implemented and tested only for the language translation application. More service composition applications can be built to test the performance and make the middleware more robust. Also, currently the service discovery happens along with the service composition. But if the system has the logic to build a service directory and compute various ways and paths to compose a service, it can efficiently compose services and also smartly select paths of composition and use alternatives in case of failure of a node

REFERENCES

[1] Chung-Ming Huang, Kun-chan Lan and Chang-Zhou Tsai, “A Survey of Opportunistic Networks,” in proceedings of 22nd International Conference on Advanced Information Networking and Applications – Workshops.

[2] Pietilänen, Anna-Kaisa, and Christophe Diot. "Dissemination in opportunistic social networks: the role of temporal communities." *Proceedings of the thirteenth ACM international symposium on Mobile Ad Hoc Networking and Computing*. ACM, 2012.

[3] M. Ramadas, S. Burleigh, and S. Farrell, “Licklider transmission protocol - specification,” *Internet-draf*, 2007.

[4] Zyba, Gjergji, et al. "Dissemination in opportunistic mobile ad-hoc networks: The power of the crowd." *INFOCOM, 2011 Proceedings IEEE*. IEEE, 2011.

[5] Allen, S. M., et al. "Exploiting user interest similarity and social links for micro-blog forwarding in mobile opportunistic networks." *Pervasive and Mobile Computing* (2011).

[6] Boldrini, C., et al. "Context-and social-aware middleware for opportunistic networks." *Journal of Network and Computer Applications* 33.5 (2010): 525-541.

[7] Carvalho, Marco, Michal Pechoucek, and Niranjan Suri. "A mobile agent-based middleware for opportunistic resource allocation and communications." *Defence Applications of Multi-Agent Systems* (2006): 121-134.

[8] Arnaboldi, Valerio, Marco Conti, and Franca Delmastro. "Implementation of CAMEO: A context-aware middleware for Opportunistic Mobile Social Networks." World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2011 IEEE International Symposium on a. IEEE, 2011.

[9] Pietiläinen, Anna-Kaisa, et al. "MobiClique: middleware for mobile social networking." *Proceedings of the 2nd ACM workshop on Online social networks*. ACM, 2009.

[10] Howes, Timothy A. "The Lightweight Directory Access Protocol: X. 500 Lite." *Ann Arbor* 1001 (1995): 48103-4943.

[11] Sadiq, Umair, et al. "Modeling and simulation of service composition in opportunistic networks." *Proceedings of the 14th ACM international conference on Modeling, analysis and simulation of wireless and mobile systems*. ACM, 2011.

[12] Kalasapur, Swaroop, Mohan Kumar, and Behrooz A. Shirazi. "Dynamic service composition in pervasive computing." *Parallel and Distributed Systems, IEEE Transactions on* 18.7 (2007): 907-918.

[13] Del Prete, Lucia, and Licia Capra. "MoSCA: service composition in mobile environments." *Proceedings of the ACM/IFIP/USENIX Middleware'08 Conference Companion*. ACM, 2008.

[14] Gu, Xiaohui, Klara Nahrstedt, and Bin Yu. "SpiderNet: An integrated peer-to-peer service composition framework." *High performance Distributed Computing, 2004. Proceedings. 13th IEEE International Symposium on*. IEEE, 2004.

[15] Waldo, Jim. "The Jini architecture for network-centric computing." *Communications of the ACM* 42.7 (1999): 76-82.

[16] Veizades, John, and Charles E. Perkins. "Service location protocol." (19

BIOGRAPHICAL INFORMATION

Samaya Madhavan, born in 1988 in India had her early schooling in Chennai (the erstwhile madras city) and graduated from the College of Engineering, Guindy, Anna University in 2009 where she did her bachelor's in Computer Science and Engineering. Later in 2012 she completed her Masters' in Computer Science at the University of Texas at Arlington. She is currently employed as the Intermediate Programmer Analyst with Transamerica Life and Protection at Bedford, Texas where she earlier completed her internship as QA and Software Automation Testing Engineer.

Born as the second child to Madhavans, Samaya's father Madhavan Sadagopan is an engineer by profession and is currently the Regional Technical Manager and Regional Business Manager of an international inspection organization; mother Subhashini is a graduate in science and is currently a home maker. Samaya's brother Varun Madhavan is currently pursuing his PhD studies at the University of Illinois, Urbana.