UNIVERSITY OF NAIROBI

SCHOOL OF COMPUTING AND INFORMATICS

IMPLEMENTATION OF SERVICE DISCOVERY FOR ANDROID OPERATING SYSTEM BASED MOBILE AD-HOC NETWORKS

EUGENE KARIBA KAMAU

REG NO: P53/65266/2013

MSC DISTRIBUTED COMPUTING TECHNOLOGY

SUPERVISOR: DR KAHONGE

Submitted to the School of Computing and Informatics in partial fulfilment of the requirements for the award of the degree of Masters in Distributed Computing Technology of University of Nairobi

DECLARATION

This research project being presented is my original work and to the best of my knowledge the work has not been submitted or presented for any other award in any university.

……. ……

Signature Date

Eugene Kariba Kamau

P53/65266/2013

This project has been submitted in partial fulfillment of the requirement of Master Degree in Distributed Computing Technology of the University of Nairobi with my approval as the University Supervisor

…… …….

Signature Date

Dr Andrew Kahonge

ABSTRACT

A mobile ad hoc network is an autonomous collection of mobile devices that communicate with each other over wireless links and cooperate in a distributed manner in order to provide the necessary network functionality in the absence of a fixed infrastructure. In order to enable communication between nodes that are not directly within each other's send range, intermediate nodes act as routers that relay packets generated by other nodes to their destination. Nodes are free to join or leave the network and they may move randomly resulting in rapid and unpredictable topology changes. MANET (Mobile Ad hoc Networks) nodes operate in energy constrained, dynamic, distributed multi-hop environment and nodes need to reorganize dynamically in order to provide the necessary network infrastructure and centralized administration. An Android ad hoc network facilitates implementation of applications that require collaboration such as entertainment multiuser games and p2p communications. Android ad hoc networks can facilitate ubiquitous computing in areas such as home applications where wireless sensors and actuators are embedded in consumer electronics. The Android OS runs in smartphones, tablets, wearable devices, smart TVs and most recently in autos for navigational systems, information systems and entertainment application. There has been enthusiasm in the implementation of Android ad hoc networks. The unique characteristics of MANETs require for specialized implementations for Service and resource discovery, addressing and Internet connectivity and security and node cooperation among others.

This research presents an implementation of Service Discovery middleware for Android Mobile ad-hoc networks. The implementation overlays a Mobile Ad-hoc Network implementation. We propose a routing implementation that is based on the Ad-hoc On-Demand Distance Vector protocol, with modifications to reduce control message overhead. We also developed an implementation of a Semantic approach for description of services on mobile devices, that facilitates semantic service discovery in Android OS Mobile Ad Hoc Networks. The implementation was tested by simulating the routing implementation. An Android based prototype was developed and tested on a range of devices. We deduce from the protoype that it is possible for android devices to share resources and services in ad hoc networks.

ACKNOWLEDGMENT

I wish to express my deepest appreciation to my project Supervisor Dr Andrew Mwaura Kahonge and the MSC Distributed Computing Technology fraternity for making this project a success.

I also wish to acknowledge and appreciate my classmates and other researchers in the field of Distributed Computing whose contributions to the field of Mobile Computing have laid a foundation for this work.

Finally I would also like to express my deepest gratitude to my friends and family for their unending support throughout my studies.of Distributed computing whose contributions to the field have made this work a success.

TABLE OF CONTENTS

List of figures

List of Tables

ABBREVIATIONS

MANET – Mobile Ad-Hoc network

TTL – Time to live

AODV- Ad-hoc On Demand Distance Vector Routing

RREQ- Route Request Packet

CHAPTER 1

INTRODUCTION

1.1 Background

A Mobile Ad-Hoc Network (MANET) is an autonomous transitory association of mobile nodes that communicate with each other over wireless media. The main characteristics of an ad hoc Network are a lack of centralized infrastructure and administration. Mobile Ad hoc networks are formed in an arbitrary manner by a collection of nodes as the need arises.

Characteristics of a MANET are: -

- Autonomous and infrastructure less: Nodes do not depend on any established infrastructure or centralized administration.
- Multi-hop routing: No dedicated routes are necessary for MANET. Every node acts as a router and forwards each other's packets to enable information sharing.
- Mobility: Each node is free to move about while communicating with other nodes. The topology of MANETs is dynamic in nature due to constant movement of the participating nodes, causing the intercommunication patterns of the participating devices to change continuously.

In this project, we shall use application level ad-hoc networking. This will not require modification of the Android Kernel. Details of the implementation for ad-hoc connectivity not related to service discovery will not be discussed in detail.

Inherent characteristics of MANETS include resource poor devices, limited bandwidth, high error rate and a continually changing topology. These characteristics impact on network design decisions.

Our implementation of a MANETs addresses the following:

- Routing implementation
- Service and resource discovery
- Addressing and Internet connectivity
- Power management

1.2 Problem Description

There has been an increasing interest in developing ad hoc networks for Android OS devices owing to the pervasiveness of android devices. However, research in Android OS ad hoc networking has mainly focused on addressing routing issues. There are currently no notable attempts to address service discovery even in notable implementations of Android MANETs. Part of this could be due to the fact that ad hoc communication between mobile devices is a recent research dimension.

The basic role of Ad Hoc Networks is to allow users to exchange data and consume each other's services and resources. Thus as the interest in developing ad hoc networks on the Android OS continues to rise there is need to address the enabling technologies for these MANET platforms. Existing MANET projects focus on implementing routing techniques. The importance of service

discovery in any Mobile ad hoc networks cannot be overstated. This research contributes to ongoing efforts to implement MANETS for android devices by proposing an implementation for Service Discovery in the Android MANET.

1.3 Significance

The Android OS is open source and has proven to be a good platform for driving ubiquitous computing. Smartphone Ad Hoc networks can be used to facilitate development of Collaborative applications such as for communication and mobile gaming. Ad hoc networks for Android OS are also an enabling technology for ubiquitous computing. Ubiquitous computing would facilitate communication between devices where infrastructure based connections are not available or desirable.

Over 285 billion Android powered devices had been sold by the year 2015 (CNET,2015).

There are numerous applications of android devices in home automation (Androidcentral.com, 2016). Apple introduced Mesh Networking in iOS devices (Elgan, 2014). So far, this has facilitated development of mesh based applications.

Google recently released a framework that will facilitate development of ubiquitous applications for smart devices. There have been discussions on the possibility of introducing a service discovery framework in Android OS (Plus, 2013).

1.4 The need for service discovery in MANETS

The basic role of Ad Hoc Networks is to allow users to exchange data and consume shared services. Collaborative applications also need service discovery capabilities. Our research covered approaches in developing Service Discovery for Mobile ad hoc networks. We propose an approach for developing Service discovery, using semantic techniques for service discovery and an efficient routing mechanism based on Ad-hoc On Demand Distance Routing Vector and neighbor knowledge.

1.5 Objectives

The objective of this project is to propose a framework for Service Discovery in an Android OS mobile ad hoc network. Approaches for the implementation of Service Discovery mechanism have been explored. A prototype was then implemented. The aim was to apply proven techniques for implementation of Service Discovery strategies to:

i. Develop a Service registration language appropriate for Service description, registration, advertisement and service invocation in an Android ad hoc network

ii. Develop a lightweight Service Advertisement and Discovery protocol for Android ad-hoc networks. The scope of this implementation will be mobility support and internode communication.

iii. Develop a pattern for implementing services that will facilitate the invocation of services by nodes in an android based ad hoc network.

1.7 Research questions

. Some of the questions we sought to answer were:

- What are the approaches for developing Service Discovery mechanisms for MANETs that are best suited for Android OS MANETS?
- What is the performance of each approach taken in implementing Service registration, Service advertisement, Service discovery and provide mobility support?
- What are the biggest challenges in having Android devices share their service and resources and how can they be overcome?

1.8 Scope

This project discusses an approach for implementing Service Discovery functionality for an Android OS ad hoc network. We have also developed a prototype middleware application.

The prototype is implemented as an android library project. Other applications will connect to it via Application Interface(API) calls. The middle-ware will be optimized for integration to other MANET applications during development. The component will be developed in isolation and will be tested on existing an MANET project to establish the ease of integration into projects.

We shall use suitable quality of service parameters to determine the best approach.

CHAPTER 2

LITERATURE REVIEW

2.1 Related Projects

We discuss in brief notable implementations of mobile ad hoc networks for Android smartphones.

SPAN (Smart Phone Ad-Hoc Networks) (3wdroid.org, 2014)

This is an Android Based Mesh network project initiated by a USA based not for profit organization. It was inspired by events that followed the Haiti earthquake. Though the cell phone infrastructure survived it was overwhelmed by the load of international workers causing it to crash (Patterson, 2014). The open source project hosted on GitHub currently has been downloaded between 10,000-50,000 times on the Google play store (https://play.google.com/store/apps/details?id=org.span&hl=en). This project does not have a built in implementation for Service and Resource discovery.

The Serval project (Gardner-Stephen, 2011)

This is a practical mesh telephony platform (Gardner-Stephen, 2011) inspired by a Mesh network called the Village Telco (Village Telco, 2008). The Serval project uses Android devices.

The motivations for the project are: -

- Current cellular service costs are high- Gardner-Stephen states that the high prices are due to the high cost of telephony support infrastructure. Service discovery functionality will facilitate development of a communication service that uses the devices as it's infrastructure.
- Unrest disaster and emergency Gardner-Stephen explains that extraordinary events can disable, disrupt or overwhelm mobile telecommunications infrastructure.
- Wireless mesh networks as a complement to infrastructure -oriented cellular services.

Although the current implementation addresses most of the technical aspects of MANET implementation, the issue of Service and Resource discovery is not addressed.

MAINGATE (Mobile Ad-hoc Interoperable Network Gateway)

MAINGATE (Military aerospace.com, 2011) is a mobile ad hoc network gateway developed and tested for the USA military. MAINGATE has been applied in forming Smart-phone ad hoc networks.

The Android Operating System has also found acceptance in implementing pervasiveness. The capabilities of the Android operating system enable its application in a variety of devices besides

Smart phones. Ad Hoc networks are in many ways an enabling technology for ubiquitous computing.

2.2 Service Discovery

Service Discovery is defined as a process allowing networked entities to (Ververidis and Polyzos, 2008):

- Advertise their services.
- Query about services provided by other entities.
- Select the most appropriately matched services.
- Invoke the services.

2.3 Characteristics of a service discovery protocol

The typical characteristics of any Service Discovery framework are (Mian et al, 2009):

- Service description
- Service registration
- Service discovery mechanism-comprising search and selection)
- Routing integrations
- Mobility support

2.3.1 Service Description

A shared service requires a description to enable other services or devices to use it. Service Description includes service properties such as service ID, server properties and network properties. XML and its derivatives such as DARPA Agent markup language and web ontology language OWL are the most common service description approaches. The description is a plain text file that contains all of the service's information.

2.3.2 Service registration

This includes information on:

- Duration of the service description
- Distance and number of hops the information will travel according to the service provider.

2.3.3 Service Discovery

Discovery is the process of nodes learning about services available on other nodes. It entails search and discovery. Search mechanisms are responsible for finding registered services. Search depends on the Service registration mechanism. The mechanisms may be: -

- Active search -In SDPs that register services in the local cache search takes place in the nodes by flooding the search message.
- Passive search-this happens in SDPs that register their services in only the local cache. The search is done in the local cache.

Another approach to search is selectively forwarding search requests. In this method, the Service Discovery Protocol first searches a node's local cache. If it fails to find a service, the node selectively forwards the request to other nodes on the basis of the ontology descriptions (Chakraborty et al, 2006). When a node doesn't have enough information to selectively forward a request, it simply broadcasts the request to its neighboring nodes.

The search strategy employed affects network performance as flooding the network for instance can create a lot of traffic in the network.

2.3.4 Service Selection

A service request could have multiple references. Selection can be done manually by the user or automatically, using a criteria-based algorithm (Main et al, 2009). It can be based on the proximity of a node, the current load of the node, bandwidth requirements etc.

2.3.5 Routing

Service discovery protocols (SDP) that register or search in all of the network's nodes flood query or registration messages. Service Discovery Protocols seek to minimize the number of messages that nodes need to generate. SDPs can limit the flooding to n hops or use multi cast routing to specially formed groups of service providers, selective forwarding based on ontology, or selective forwarding based on the notion of potential (Mian et al, 2009). They can also form overlay networks and have specific routing mechanisms within these overlay networks.

2.3.6 Mobility Support

In mobile ad hoc networks nodes frequently move and can change their positions with respect to each other. In SDPs without directory nodes, all nodes keep information of only their own services. In such a case, the SDPs automatically handle mobility by flooding to all nodes or multi-casting query messages to a few nodes citing (Cheng, L. and Marsic, I. 2000). When a service provider advertises its services, it also announces the timeout, or when the service information will expire. The mobile nodes cache this information. The service provider must re announce the service information periodically. Another method for handling mobility is reducing the advertisement diameter and advertisement time interval to compensate for the effects of rapidly changing vicinities (Chakraborty, D. et al. 2002).

2.4 Managing service descriptions

Service Discovery Protocols' manage service descriptions in various ways:

- Some SDPs store the service description in the local cache. In this case the search must cover all nodes to find the service.
- Another approach is to store the service description on all nodes in the networks. This is done by service advertisements. In this case a service search covers only the local cache.
- The service description can also be stored in a subset of nodes by distributing advertisements to n hops (citing Chakraborty, D. et al, 2004 and Lenders, V. et al, 2005). This can also be done by distributing advertisements to multicast groups of nodes (Helal et al, 2003).

The authors observe that by restricting advertisements that flow, the protocols reduce memory requirements and increase the probability of discovering a service in the service provider's vicinity. When the service descriptions are stored only in immediate neighbors, the advertisement diameter can be just one hop.

Service Discovery Protocols can also store service descriptions in special selected nodes, called directory nodes. A directory or directory node is an entity that stores information about services available in the network, facilitating service discovery Cho C. and Lee D. (2205, cited in Mian et al, 2009).

2.5 Service Discovery Protocols Architectures

There are three largely accepted categories of Service Discovery Protocols Architectures (Zakarya and Rahman, 2013, Satish N et al, 2011 and Mian et al, 2009):

- Directory Based Architectures
- Directory Less Architectures
- Hybrid Architectures

2.5.1 Directory Based architectures

A node can be a server, a client or a service directory. Service providers register their services to service directories and service requesters are informed about the available services in the network only through these directory nodes (Satish N et al., 2011).

The challenge of directory based architectures in Mobile ad hoc networks is that no single node is always reachable. This implies that centralized directories may not work. Also scalability would be an issue as the resource constrained nodes would not be able to handle responses for a large number of nodes. There is need to have a strategy for nodes to learn what services are available on every other node in the network that is a global directory. One approach is to use a full replication strategy for directory nodes in order for every directory to store all services available in the MANET, irrespective of their location (Ververidis and Polyzos, 2008).

To further enhance this approach, nodes acting as directories are in constant communication with each other and replicate service information among them. This results in a backbone of directory-enabled nodes.

An alternative to the aforementioned backbone based approaches for implementing distributed directories is a clustering approach. Each cluster of service providers is formed

based on physical proximity and semantic proximity of the descriptions of provided services.

2.5.2 Directory less Architectures

In this approach there are no Service Directory nodes to mediate communication between service requesters and service consumers. Clients contact service providers directly by flooding the network with service queries. This results in a high message overhead. The flooding by query messages additionally consumes lot of bandwidth, computational and battery resources, which are already scarce in MANETs thus making this architecture unsuitable for MANETS (Zakarya and Rahman, 2013).

Service providers broadcast service advertisements and Service requesters broadcast service requests. Both processes may take place at the same time in the network. A basic problem in non-directory based approaches is how to determine the frequency of service advertisements in order to reduce network load and avoid redundant transmissions (Satish et al, 2011).

2.5.3 Hybrid Approaches

This is a hybrid of directory based architectures and directory less architectures. It has both the benefits and weaknesses of the two individual approaches.

Service providers register their services with Service Directories if they locate any in their surrounding area. Service requesters send their queries to the service directories as they are aware of. If they are not aware of any service directory, they broadcast the queries to the whole network. Service replies may come both from Service Providers and Service Directories.

2.5.4 Overlay-based vs. Overlay-less

(Mian et al, 2009) describe another category for classifying service discovery protocols, overlay network. The overlay network in multi hop ad hoc network Service Discovery Protocols has the advantage of being able to control the multi cast of service query or advertisement messages. This controlled multi-cast restricts and greatly reduces the network traffic.

2.6 Service Discovery Implementation

We discuss two important aspects of service discovery implementation. These are service discovery architecture and service matching.

2.6.1 Service Matching

Existing service matching techniques in SDP protocols use simple matching schemes with service matching being done at syntactic level. The weaknesses of this approach are mentioned in this document. Later in this document we shall also discuss how the shortcomings can be mitigated by applying semantic techniques in service matching.

2.6.2 Service Discovery Architecture

Service discovery architectures can either be request-broadcast-based or advertisement based.

Broadcast Based

In broadcast based architectures a service discovery request is broadcast throughout the network and if a node has the service it responds with a service reply. The disadvantages of this approach are poor scalability in large diameter networks and those with many devices. Further, there is poor resource utilization as even nodes that do not have the service or are not in the route are involved.

Advertisement Based

In advertisement based service discovery architectures nodes advertise their services in the network. Nodes interested in discovering services cache the advertisements. Advertisements contain service metadata and location data. The nodes memory cache grows, thus imposing memory constraints. This approach is also inefficient in terms of bandwidth usage as the network is flooded.

These architectural approaches haves the challenges, namely network load, network wide reachability and scalability, as demonstrated by Chakraborty et al,2006. They recommend that service matching and discovery architecture implementation should not be decoupled. They further propose a hierarchical grouping of services to mitigate some of the challenges.

2.6.3 Case Studies

In this section we discuss previous implementations of service discovery frameworks.

Case Study 1: Enhanced Service Discovery in Bluetooth (Avancha S et al)

This implementation applies semantic techniques in service discovery. The implementation consists of several core components:

Service Ontology

The service ontology describes relationships between different entities. It also facilitates inexact matching by specifying rules and constraints on attributes. The implementation uses DAML+OIL for service description. This implementation does not guard against the limitations mentioned earlier.

Reasoning engine

The reasoning engine has a knowledge base that stores information about service instances. Large amounts of data are loaded into the engine. The implementation is able to perform pattern matching to discover whether it can answer service requests directly. Upon failure, it evaluates other possible solutions which match the query attributes within an error range.

This implementation requires a large cache in order to store information all services. This introduces a memory constraint on nodes. In addition, it imposes the need for a centralized directory where service descriptions are maintained. This is a bottle neck in many MANETS.

We will seek to address these challenges in the solution we shall develop.

Case Study 2: Konark (Helal, S. et al)

Konark is a service Discovery architecture designed for ad hoc peer to peer networks and targeted towards device independent services and m-commerce oriented software services. It is geared towards managing service discovery across homogeneous devices.

This implementation is optimized for service discovery in homogenous devices. Services are grouped in a hierarchical tree thus facilitating minimal payloads for advertising devices.

The authors argue that existing Service Discover Protocols place emphasis on devices as services rather than a device independent software services. Konark assumes an IP levee connectivity in the ad-hoc network. Each device maintains a local repository that maintains the local services it is offering.

Service information is disseminated via advertisement and discovery requests. The rate of advertisement is determined by preset parameters or network dynamics. Advertisements can be either location or time based. Service advertisements contain time to live information. Clients are therefore able to cache service information. As observed before in this document caching service information results in large caches in the devices that introduce resource constraints on devices.

Case study 3: DEAPspace (Hermann et al.)

The aim of the DEAPspace project is to provide a framework for internetworking pervasive devices. The objective is to facilitate a proximity bound computing and transient communication relationships in ad-hoc networks. This project's key research focus is on prompt service discovery without a central service discovery server, compact and extensible discovery server and description, security configuration and device management.

The DEAPspace project revolves around what the authors describe as a model. Its usage model is role based, having service requestors and providers. DEAPspace research encompasses algorithms and service descriptions to along the roles. Devices broadcast their world view. If missing devices occasionally incorporates new elements found in each service broadcast, then missing a broadcast will not cause a problem as the next broadcast by any device that did not miss a particular broadcast will update this information. This mitigates the problem of disconnections and transient devices in ad-hoc networks. Like in several other approaches, the broadcast/advertisement algorithms are fully time aware.

2.7 Semantic Service Discovery

The proponents of web semantic service discovery argued for the augmentation of services with descriptions of their capabilities so as to facilitate the reuse without human assistance or any predefined interfaces or protocols. These concepts can be applied in building service discovery for mobile ad hoc networks.

Semantic service descriptions enable services to be device interpretable. They facilitate services whose properties, capabilities, interfaces and effects are encoded in an unambiguous, machine understandable form. Service data can be captured together with specifications of its properties and capabilities, the interface for its execution, preconditions and effects of its use.

(MCllraith et al) observe that semantic markup of services, in the web sphere, achieves the following objectives:

i. Automatic service discovery. Services can be located that provide a particular service and that adhere to requested properties.
ii. Automatic service execution. Descriptions contain information that facilitates service invocation such as inputs required, the expected output and information of how to execute the service.
iii. Automatic service composition and interoperation. With sematic markup the information required is available at the cooperating device.

Syntactic description of services offers little for automated discovery and execution of services. Most of the times human intervention is required to consume services. Semantic description is necessary to define the meanings of descriptions sufficiently for machine readability.

In Android OS MANETS, this intuitive model of service descriptions is helpful. Nodes are highly heterogeneous and dynamic in many respects, such as location, OS versions, device capabilities and so on. Cheng, L. and Marsic, I. observe that there are two aspects of heterogeneity that need to be considered when studying service discovery in mobile ad-hoc networks. These are

- Application resource heterogeneity- Different nodes may have capabilities such as diverse processing capabilities, memory amounts, power and so on.
- Communication links capability- Wireless links in the network generally have low and volatile bandwidth and large transmission latency. Because of dynamic channel sharing and fading, the available bandwidth of a wireless link changes frequently and abruptly.

In a typical scenario one device would be a smartphone and the next node could be a printer or a smart TV. This heterogeneity necessitates an approach that facilitates seamless integration. A selection of services is required if there is more than one service matching a service request. Non-functional attributes could also come into play. With semantic descriptions complex services can be composed on the fly by aggregation of several atomic services. Semantic service discovery can be viewed as the application of approaches that achieve the above. (Charif et al) sum up the components of semantic service description as a register, a reasoner, a matchmaker, a decomposer and an invoker.

2.7.1 Semantic Service Description

Service description is critical in order for MANET nodes to be able to deploy, compose and consume services automatically. Service description should be able to communicate service, parameters and capabilities such as inputs, outputs and service dependencies.

(Nedos et al) argue that most semantic service description languages and architectures require a globally connected network such as the web. These conditions are not possible in a MANET environment. They propose a model where nodes do not share a common Semantic representation for their services but instead a Semantic agreement is derived through node interaction.

(Chakraboty et al,2002) propose a Group-based Service Discovery Protocol that revolves around a novel service description syntax, where services are grouped based on a universal ontology. Unlike in the previous approach, the service ontology is similar across all the nodes. Services are grouped based on their similarity. The solution they propose extends the Dreggie ontology, that contains a comprehensive ontology to describe a service in terms of its capabilities, inputs outputs, platform constraints, device capabilities of the node etc. In the Dreggie ontology, the

generic class service is functionally classified into two main subgroups, hardware and software service. Each subgroup is further classified in this manner all the way down to a specific service. This they explain eliminates strict syntactic matching based on unique identifiers or interfaces and provides flexibility to do loose coupling.

Nedos et al, state that two problems that service discovery has to address are:

- Discovery queries must be interpreted by nodes with heterogeneous ontologies. They argue that given the random patterns in MANETs it is inappropriate to assume that a global ontology is available in every mobile node. Rather each autonomous node should be allowed to describe its own service.
- Lack of persistent and centralized registries, an implicit scenario in MANETs.

They propose a model where concepts rather than services are advertised and discovered. This they explain provides flexibility to enable the progressive ontology matching and the concept-based discovery of services.

We will use the resource description framework schema to model ontologies and to represent service description and queries. Our service specification will be based on the service profile of OWL-S.

In Konark (Helal, S. et al,2003) the Ontology of the service description framework is highly related to the Dreggie ontology. Services are described in a tree based structure. The tree has a number of levels that represent service description. The service descriptions become more specific further down the tree. Services are described as ‘all’, ‘generic’ and ‘specific’ based on the tree level.

Figure 1 Konark service tree

The root of a service description document in Konark is the Service type. It represents the start of the service description. The first child is the service name, a user friendly name of the service. The service type defines the type of the service such as printer. The third child is Keywords. These keywords may describe the service type or some characteristic of the service.

Service description concepts

In this section, we discuss key concepts related to frameworks for semantic service description. Semantic service description frameworks can be described with respect to:

- What kind of service semantics are described?
- In what language or formalization are concepts described?
- What kind of reasoning is allowed upon the abstract service description?

The use of ontologies enables the sharing of service concepts and vocabulary re-use. An ontology represents the capabilities of a service and the restrictions applied to its use.

Functional and non-functional semantics

Functional semantics relate to what a service actually does and how it works. They describe the service capabilities. These include inputs. Outputs, preconditions and effects, execution flow and so on. On the other hand, non-functional semantics are those aspects of the service that do not directly relate to its capability such as a service name

Data Semantics

Domain dependent semantics of a service profile are represented in terms of concepts taken from a shared domain, table or application ontologies. If different ontologies are used the service description agents will be required to resolve the heterogeneities for interoperability, or to carry out inter-ontology mapping.

2.7.1.1 Reasoning about Semantic Service Descriptions

The basic idea of formally grounded descriptions of Services is to allow agents to better understand the functional and non-functional semantics through appropriate logic-based reasoning(Klusch,2005). Concept expressions used to specify data semantics are usually built up from basic concepts and roles taken from formal application or domain ontologies.

2.7.1.2 An Overview of Semantic Service Description Frameworks

We discuss several approaches to semantic service description. Sematic Service description in mobile ad-hoc networks has its foundations in Semantic representation of web services.

SAWSDL

The standard language WSDL for Web Services is a W3C recommendation. SAWSDL was developed by adding mechanisms by which semantic annotations can be added to WSDL components. It provides mechanism by which ontological concepts that are defined outside WSDL service documents can be referenced to semantically annotated WSDL description elements.

According to Klusch and Kapahnke (2008) The key design principles of SAWSDL are:

- To be enable semantic annotations of web services using and building on WSDL.
- It is agnostic to semantic (ontology) representation of languages.
- It enables semantic annotations for web services for discovering web services and invoking them.

Based on these design principles, SAWSDL defines attributes for mapping between WSDL components and semantic concepts. They explain that SAWSDL is a mere syntactic extension of

WSDL without any formal Semantics. We observe that SAWSDL does support well logic based discovery and composition of services.

2.7.1.3 OWL-S

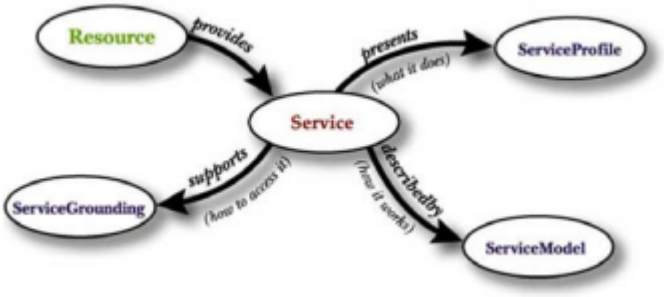

Figure 2 OWL-S structure

OWL-S is an upper ontology used to describe the semantics of services based on W3C standard Ontology. It builds on top of OWL and consists of 3 main upper ontologies:

- The service profile
- The process model
- Service grounding

Service profile

The OWL-S profile ontology is used to describe what the service does, and is meant to be used mainly for service discovery. An OWL-S service profile encompasses functional requirements whose attributes include:

i. *hasInput* and *hasOutput* -These relate to data channels where data flows between processes.

ii. *Preconditions*-Specify conditions that must be asserted in order for an agent to execute a service

iii. *Effects*-Parameters that can be asserted to establish a successful execution of the service

Non-functional requirements include: *Service name, service category, quality rating, text description, metadata* about the service and other known requestors.

The profile class can be sub classed to support the creation of profile taxonomies

Service process model

The OWL-S process model describes the composition of service. It describes the inter-relation of consistent patterns and workflow features. A process in OWL-S can be atomic, simple or composite depending on the level of complexity.

Atomic processes are single process descriptions with exposed inputs, outputs, preconditions and effects. Simple processes provide a means of describing service or process abstractions which have no specific binding to a physical service, and are realized by an atomic process through service discovery and dynamic binding at runtime or through a composite process(Klusch). Composite processes are hierarchically defined workflows, consisting of an aggregation of processes. They are constructed using control flow sequences, inputs, outputs, pre-conditions and effects of their constituent processes.

Service grounding

The OWL-S service grounding provides a binding between logic based and XML based service descriptions for the purpose of facilitating service execution.

2.8 Service Advertisement and Discovery

Service advertisement is handled at the advertisement layer of the MANETs implementation. Service advertisement is the mechanism by which nodes in MANETs communicate their capabilities to their peers. Service discovery is the mechanism by which nodes learn about services from their networked peers.

2.8.1 Service Advertisement

We shall discuss advertisement from two main perspectives:

- Service advertisement description
- Service advertisement communication or routing

There are two general mechanisms for service advertisement. These are:

- Active pull-A mechanism where clients use a discovery process to discover advertised services.
- Passive push- A mechanism where servers use an advertisement process to periodically announce their registered services.

In order for nodes to advertise their services, they need to have the ability to propagate packets across the networks. Various protocols and algorithms have been proposed for service advertisement in mobile ad hoc networks.

Hassanein et al. classify and compare service discovery protocols based on service discovery strategies and service information accumulation approaches. We shall use their approach to classify Service Discovery approaches in any further discussions. The classification is: -

2.8.2 Service Discovery Strategies

i) Service Discovery based on Supporting Layer
   - Network Layer
   - Application Layer

ii) Service Discovery based on Multicast DNS

iii) Service information Accumulation Strategies
   - Without Directory
   - Central Directory
   - Distributed Directory

In network layer protocols, service discovery is coupled with the routing mechanism.

2.9 Message Propagation Mechanisms

The goal of data communication in a service advertisement implementation is iterative data transfer between participating nodes. Broadcast implementations employ multi-hop strategies to

propagate packets. Different metrics can be applied to access the performance of broadcast protocols. These could be the number of messages it generates, the time needed for the broadcast to complete or the reliability (possibility that all nodes will deliver the broadcast).

In any flooding mechanism, one must balance reliability against message overhead. On the one hand, increasing reliability generally involves sending a greater number of redundant messages and thus incurs a higher message overhead. Redundant messages are needed to reach all nodes and to recover from packet loss, hence reducing the overhead will generally decrease reliability.

Excessive broadcasts however can cause a broadcast storm. A broadcast storm is characterized by message redundancy, network contention and packet collisions.

2.9.1 Gossip Protocols

Gossip protocols employ epidemiological strategies. They are probabilistic in nature. A node chooses its partner node with which to communicate randomly. Each node sends out a fixed number of packets to a fixed set of neighbors independent of the number of nodes in the network. Gossip protocols have a high reliability. However, a major drawback is that they generate a large number of messages (Marzullo, K. and Masini, S.).

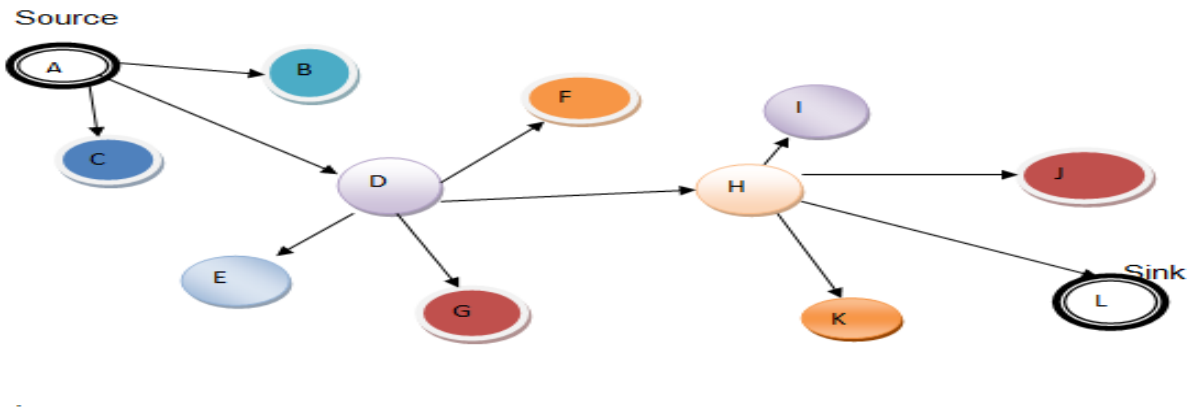

Figure 3 Mondal, S. Gossiping routing protocol(Data is coming from 'A' to 'L' through 'A->D->H->L' path)

A typical gossip algorithm.

F = 1 and the number of hops a gossip message can travel is fixed) and is called blind counter initiate broadcast of m:

send m to B initial neighbors

when (p receives a message m from q)

if (p has received m no more than F times)

send m to B randomly chosen neighbors that

p does not know have already seen m;

Gossip protocols can be classified into two:

- Anti-entropy protocols: -They send an unbounded number of messages in non-terminating runs. Their premise is the eventual delivery of messages and timely delivery is not a priority.
- Rumor Mongering: - They terminate and so the number of messages sent is bounded

Friedman, R. et al summarize the advantages of gossip protocols as below:

- Scalability: -They are scalable with respect to the node load
- Network load balancing: - The load is equally distributed among the peers
- Resilience to node failures: - A node is likely to be contacted by more that one peer therefore the failure of a node would be mitigated.

Gossip protocols however have shortcomings in MANETs. They rely on routing which can be expensive in a MANET owing to the absence of dedicated resources and dynamic nature of the network that makes route maintenance difficult. MANET nodes also typically suffer from limitations in memory, processing and power.

In a fixed network setting, the source of a message would select at random the nodes it will communicate with. In a MANET however, not all targets are within the range of the source. In addition, due to mobility, addressing information included in messages could easily become stale. Implementations of gossip protocols therefore have to implement mechanisms to mitigate the problem of message overhead and node mobility.

Gossip protocols implement strategies to improve the efficiency of the broadcast process. (Friedman, R. et al) summarize the algorithms as: -

a. Probabilistic- Each recipient of a message applies a mechanism to determine whether or not to broadcast a packet. (Haas, Halpern and Li) demonstrate that using appropriate heuristics it is possible to save up to 35% message overhead.
b. Counter based- In this approach, each recipient monitors how many neighbors have previously decided to gossip a given message and only forwards it if it hears fewer than m such gossips.
c. Location Based- Here the recipient estimates its distance from the source of a message before, in making a decision on whether or not to retransmit a message

The constant mobility of MANET nodes could also facilitate opportunistic gossip. Nodes would send messages to receivers that are likely to be moving into particular regions of the MANET.

Gossip-based protocols rely on the explicit assumption that each node has enough buffering resources. In order for a gossip protocol to be effective, the participating nodes must have sufficient resources to cache and gossip a message for a sufficient number of times. If a node does not have sufficient resources, it will drop a large number of messages.

Our implementation will run in Android OS devices only. Nonetheless, it will be heterogeneous with respect to device capabilities.

We will take into account all the above mentioned issues in developing a gossip strategy for our implementation.

2.9.2 Flooding

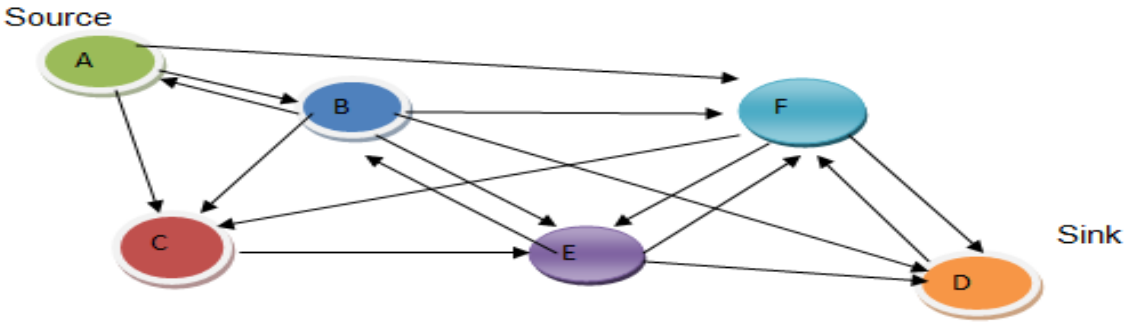

Figure 4 Mondal, S. An illustration of flooding

Unlike gossiping where a node sends a message to its neighbor, in flooding nodes send messages to all the network. A major advantage of flooding is that it requires no costly topology maintenance or route discovery. Paul, D explains that the weaknesses of flooding include:

- The implosion problem
- The overlap problem
- Nodes are blind to available resources

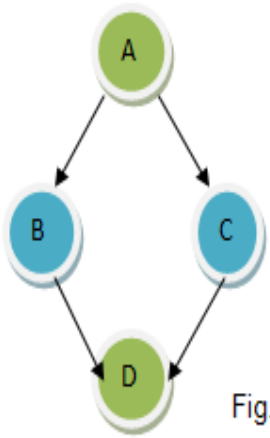

Fig.8. *The Implosion Problem*

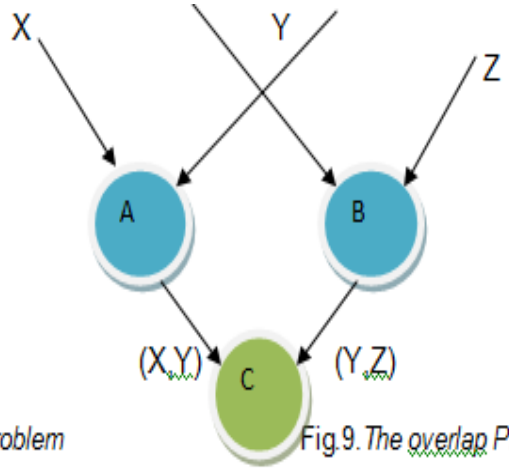

Fig.9. *The overlap Problem*

A typical flooding algorithm is:

Source node send messages to all neighbors

Receiving nodes relay the message to their neighbors.

Continue until all nodes have received the data

Ho, C et al propose flooding as an alternative for reliable multicast. A major challenge in ad hoc networks is in achieving reliable multicast communication given that nodes are mobile and there are frequent node outages and failures. Communication mechanisms should in addition to MANET constraints consider the following:

- Individual host behavior is independent of other hosts.
- There is no limit on the host speed
- There are no constraints on node movement
- High probability of frequent, temporary network partitions

2.9.3 Message Loss

Message loss is an important consideration in the operation of ad hoc network protocols.

Assume p represents per hop packet transmission delay and d is the maximum ad hoc network diameter. A message is sent at a time t0 and flooded throughout the diameter. Every node that receives the packet broadcasts it exactly once.to all immediate neighbors.

Let t1<=t0+dp

Assuming that the ad-hoc network stays continuously connected between t0 and t1, there exists at lest one node that has not received that packet.

There are several mechanisms for mitigation of message loss in transmission. Our implementation will come up with suitable mechanisms for recovery from message loss.

2.9.4 Flooding mechanisms

Flooding mechanisms can be categorized based on mechanisms they use. We discuss the classification of flooding strategies based on their complexity

Simple flooding

Simple flooding starts with a source node broadcasting a packet to all its neighbors. Each of those neighbors in turn rebroadcasts the packet exactly one-time time until all reachable network nodes receive the packet. Simple flooding is associated with message overheads as nodes flood the network with messages.

Probability Based Flooding Methods

Android devices are typically resource constrained. It is therefore necessary to develop mechanisms that are not resource intensive. Simple flooding owing to the message overhead generated is therefore not a viable option for our implementation. We discuss probability based broadcast techniques

Probability based flooding

Tseng. Y, in Williams B and Camp C. (2002) explain that in dense networks multiple nodes share similar transmission coverages. Thus randomly having some nodes not rebroadcast saves node and network resources without harming delivery effectiveness. A major setback with this approach though is that in sparse networks delivery effectiveness would be affected as nodes would not receive all the broadcast packets unless the probability scheme is very high.

Counter based scheme

There is an inverse relationship between the number of times a packet is received at a node being able to reach additional area on a rebroadcast Ni et al. This is how the counter based scheme works. Upon reception of a previously unsent messages, the receiving node initiates a counter with a value of one and sets a random assessment delay(RAD). During the RAD, the counter is

incremented by one for each redundant message received. If the counter is less than a threshold value when the RAD expires, the packet is rebroadcast. Otherwise it is simply dropped.

Area based methods

A node using an area based method can evaluate additional coverage area based on all received redundant transmissions. Area based methods only consider the average area of a transmission. They don't consider whether nodes exist within that area.

Distance based scheme

Here a node compares the distance between itself and each neighbor node that has previously broadcast a given packet. Upon the reception of a previously unseen packet, a RAD is initiated and redundant packets are cached. When the RAD expires all sources node locations are examined to see if any node is closer than a threshold distance value. If true, the node does not rebroadcast.

Location based schemes

With these approach nodes use GPS location mechanisms (William B) and apply a suitable selection algorithm.

Neighbor knowledge methods

These mechanisms entail nodes having some knowledge of surrounding nodes. A major weakness of this approach would be a requirement for resources in order to maintain the data.

Flooding with pruning

Each node has a list of it's 1 hop neighbors. Nodes also include a list of known neighbors in every message header. When a node receives a message, it compares the neighbor list with its own and only rebroadcasts the message if there is an additional node.

Scalable broadcast Algorithm

It requires that all nodes have knowledge of their neighbors within a two hop radius (Peng, W). Messages also have the identity of the source node. This enables nodes to know whether they would reach additional nodes by rebroadcasting.

Node B receives a broadcast data packet from Node A. Since Node A is a neighbor, Node B knows all of its neighbors, common to Node A, that have also received Node A's transmission of the broadcast packet. If Node B has additional neighbors not reached by Node A's broadcast, Node B schedules the packet for delivery with a RAD. If Node B receives a redundant broadcast packet from another neighbor, Node B again determines if it can reach any new nodes by rebroadcasting. This process continues until either the RAD expires and the packet is sent, or the packet is dropped.

Dominant Pruning

Dominant Pruning uses 2-hop neighbor knowledge, Rebroadcasting nodes proactively choose rebroadcasting nodes from its 1-hop neighbors. Only those chosen nodes are allowed to rebroadcast. Nodes inform neighbors to rebroadcast by including their address as part of a list in each broadcast packet header.

When a node receives a broadcast packet it checks the header to see if its address is part of the list. If so, it uses an algorithm to determine which subset of neighbors should rebroadcast the

packet, given knowledge of which neighbors have already been covered by the sender's broadcast.

Multipoint Relaying

Broadcasting nodes are selected by the upstream sender. Chosen nodes are called multipoint relays and they are the only nodes allowed to rebroadcast a message received. Since a node knows the network topology, it can select 1 hop neighbors as multi point reals that most efficiently reach all nodes with the two hop neighborhood.

The MRP algorithm

- Find all 2-hop neighbors that can only be reached by one 1-hop neighbor. Assign those 1-hop neighbors. as MPRs.
- Determine the resultant cover set (i.e., the set of 2-hop neighbors that will receive the packet from the current MPR set).
- From the remaining 1-hop neighbors not yet in the MPR set, find the one that would cover the most 2- hop neighbors not in the cover set.
- Repeat from step 2 until all 2-hop neighbors are covered.

Lightweight and Efficient Network-Wide Broadcast (Surcec, J and Marsic, I.)

The Lightweight and Efficient Network-Wide Broadcast (LENWB) protocol relies on 2-hop neighbor knowledge obtained from "Hello" packets. In LENWB, each node decides to rebroadcast based on knowledge of which of its other one and two hop neighbors are expected to rebroadcast. The information required for that decision is knowledge of which neighbors have received a packet from the common source node and which neighbors have a higher priority for rebroadcasting. The priority is proportional to a node's number of neighbors; the higher the node's degree the higher the priority. Since a node relies on its higher priority neighbors to rebroadcast, it can proactively compute if all of its lower priority neighbors will receive those rebroadcasts; if not, the node rebroadcasts. Surcec J and Marsic I observe that LENWB could be adapted by on demand routing protocols to minimize control packet overhead.

Ad-hoc broadcast protocol

It utilizes an approach similar to multipoint relaying in that only designated nodes rebroadcast. Only nodes designated as a Broadcast relay gateway within a broadcast header are achieved to rebroadcast the service. Information on destinations is contained in the broadcast header.

Ad-hoc On Demand Distance Vector Routing

The *Ad-Hoc On-Demand Distance Vector* routing protocol is described in RFC 3561. This is an on demand or reactive routing protocol for mobile ad-hoc networks. Nodes discover routes as and when necessary. The message routing algorithm we shall develop will be based on AODV. Awernuch, B. and Mishra ,A. summarize the characteristics of AODV as:

- It is an on demand protocol
- Routes are maintained only as long as they are necessary
- Every node maintains a monotonically increasing sequence number. This number increases every time the node notices a change in the neighborhood topology.

AODV Operation

AODV consists of two phases (Nand, P. and Sharma, S.,2010):

i. Route Discovery
ii. Route maintenance

A major weakness of AODV is in HELLO control message overhead. Nodes generate control messages in the process of route discovery and maintenance. This is a big challenge especially in non trivial networks.

Several approaches have been proposed to reduce the HELLO message overhead. These include:

a. Signal stability adaptive routing

Dube, R. et al propose a routing scheme where routing will be selective based on MANET node strength. Only RREQ received from strong nodes are forwarded.

b. Route Discovery using encounter ages

Dubois-Ferriere et al. proposed an algorithm based on node encounter ages. Nodes keep a record of their most recent encounters times with other nodes. Instead of searching the destination, a search is done instead for the most recent node that encountered the destination node.

c. The Polarized Gossip Protocol for Path Discover in MANETS

Proposed by Beraldi . Packets are forwarded based on the proximity of a node to the destination. The probability of a node forwarding a packet is determined by the relative distance between the source node and the destination and between the forwarding node and the destination.

d. Neighbor assisted route discovery

We discuss a neighbor knowledge approach proposed by (Kumar and Kumar,2010). The objective is to optimize flooding by avowing rebroadcasts. This is done by defining an uncovered neighbors set on each node prior to rebroadcasting a message. This is a probability based approach where rebroadcast probability is based on the knowledge of neighbors that have not received a message. The authors call the set of uncovered neighbors the uncovered neighbors set.

We observe though that this approach has got some weakness. Messages are exchanged prior to every single rebroadcast by a node. There would be a significant increase in traffic. Besides, most of the data exchanged could be repetitive. A less traffic intensive approach would be for neighbors to regularly exchange topology information, and reuse it for a specific validity period as opposed to getting node data for every single message received. We acknowledge however that this approach maintains the reactive behavior of AODV.

2.9.5 Broadcast Optimization

Network broadcast mechanisms determine the efficiency of a Mobile Ad-hoc Networks. MANET nodes are typically resource constrained and with limited bandwidth. It is therefore necessary for broadcast mechanisms to among others:

- Be non processor intensive
- They should minimize bandwidth use

A network broadcast procedure should minimize the number of transmissions needed to broadcast a packet. The number of transmissions required by a good algorithm to broadcast a message should be less than those required by brute force flooding. (Surces and Marsic ,2001). They argue that a good broadcast mechanism should maximize the number of non transmitting nodes. They go ahead to propose a metric for calculating the efficiency gain of a network broadcast algorithm as:

Let **E** be efficiency of the algorithm

Let $\mathbf{N_B}$ be Number of transmissions required by brute force flooding

Let $\mathbf{N_{NWB}}$ be average number of transmissions required by the network broadcast algorithm

Then $\mathbf{E = N_B / N_{NWB}}$

Our service discovery implementation will be optimized so as to be non-obstructive in the android devices where it will run.

2.9.6 Conclusion on Service Advertisement mechanisms

S. Ni, et al showed that Counter-Based schemes out performed Probabilistic schemes in most conditions. They also demonstrated that the Location-Based scheme was shown to be more robust than the Distance-Based scheme. The choice of mechanism is usually advised by the unique characteristics of an implementation.

An ecosystem of Android OS nodes would consist of devices of diverse characteristics and resource capabilities. This would lend itself to implementations that would support both all node participation and dominant node implementations as well.

2.10 Semantic Service Matching

Castano, S. et al describe a p2p ontology model, HELIOS, that organizes ontology knowledge in terms of concepts, properties, semantic relations and location relations. They define an ontology that is a 4-tuple of the form PO = (C, P, SR,LR), where:

- C is a set of concepts of PO.
- P is a set of concept properties.
- SR = {same-as, kind-of, part-of, contains, associates} is a set of semantic relations between content concepts.
- LR is a set of location relations between content concepts and network concepts.

Borrowing from the Artemis project (Islab.di.unimi.it) they classify semantic relations between content concepts into 5 relations:

a. *Same_as* – The same as relation is defined between two concepts c and c'. which are considered semantically equivalent i.e. they denote the same real world entity or they have the same meaning.
b. *Kind of* –the relations defined between two concepts c and c' as kind of states that the concept c is a specialization of the concept c'.
c. *Part of* The part of relation defined between two concepts c and c' states that the concept c represents a component of the concept c'.

d. *Contains* states that the concept c contains concept c'.
e. Associates. The associates relation defined between two concepts c and c'

These relations are illustrated in the figure below

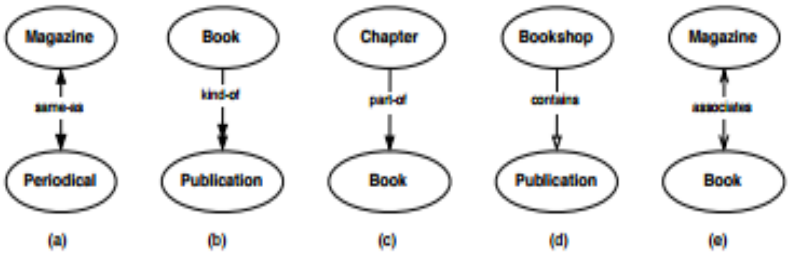

Figure 5 Castano,S. et al Examples of sematic relations in an ontology

H-Match is a semantic matching algorithm proposed by Castano, S. et al. The H-match algorithm considers both the linguistic features of concepts as well as the semantic relations among concepts. The meaning of concepts is not established according to a given definition, but depends on the network of relations holding among terms (i.e., terminological relationships) and among concepts (i.e., semantic relations), respectively (Castano, S. et al). The meaning of a concept depends on it's name as well as on its properties and semantic relations with other concepts in the ontology. They further explain that the h-match algorithm explicitly considers the context of each concept given by the set of its properties and of its adjacent (i.e., concepts which have a semantic relation with the considered concept), allowing a deep evaluation of semantic affinity between ontology concepts.

Linguistic interpretation in h-match is based on WordNet(Miller,G). Here relations are represented by the terminological relationships SYM (Synonym-of), BT/NT (Broader/Narrower Terms) and (RT Related terms).

They define weights associated with each relationship. Based on this, concepts can be evaluated for the closeness in their relationship. The higher their weight, the closer related the concepts are

	Relation	Weight
Linguistic interpretation	SYN	1.0
	BT/NT	0.8
	RT	0.5
Context interpretation	property	1.0
	same-as	1.0
	kind-of	0.8
	part-of	0.7
	contains	0.5
	associates	0.3

Figure 6 Castano,S. et al Weights associated with terminological and semantic relations

The H-Match algorithm provides a simple yet very effective method of matching concepts. It also lends itself to easy adoption irrespective of the ontology description format used. This makes the algorithm ideal for our work.

2.11 Conceptual Model

In this section a model for the system's implementation is discussed. The basis of our middle-ware implementation is a standard mobile ad hoc network implementation; it provides delivery and routing of data between nodes, subject to the spontaneous introduction and removal of any node at any time. We then developed a Service Discovery layer for this implementation.

Implementations for pervasive computing follow a cross-layered approach. Cross layered approach implies that the implementation breaks away from the traditional OSI model. None the less, the implementation is layered.

2.11.1 Cross Layer Implementation

Cross-Layer Design may be defined as the breaking of OSI hierarchical layers or the violation of reference architecture. It includes merging of layers, creation of new interfaces, or providing additional interdependencies between any two layers. In the cross layer approach each layer can share information with any other layer unlike in the OSI model where each layer has predefined functionality and can use only the services provided by the layer below it.

Applications will consume service provided in the service discovery layer which will in turn invoke services running on the application layer. This cross layer implementation is illustrated in Figure 7 below.

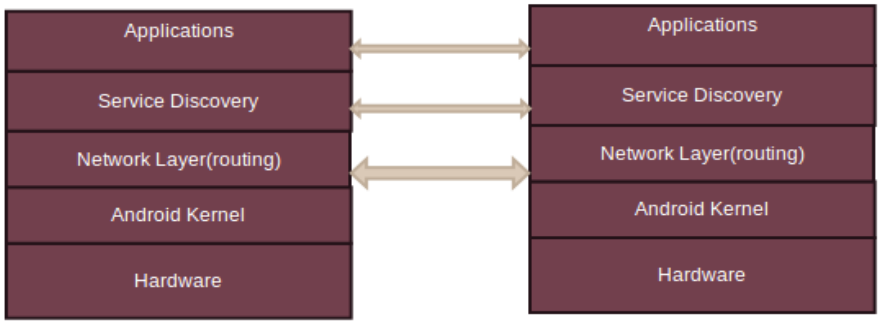

Figure 7 Cross Layer Implementation

2.11.1.1 Schematic Overview of Modules

As mentioned earlier in this document, the components of a service discovery implementation are:

a. Service Description
b. Service Registration Routing
c. Service Discovery
d. Mobility support

The interaction of these components is illustrated in Figure 98 below. Nodes interaction in the network is illustrated in Figure 9.

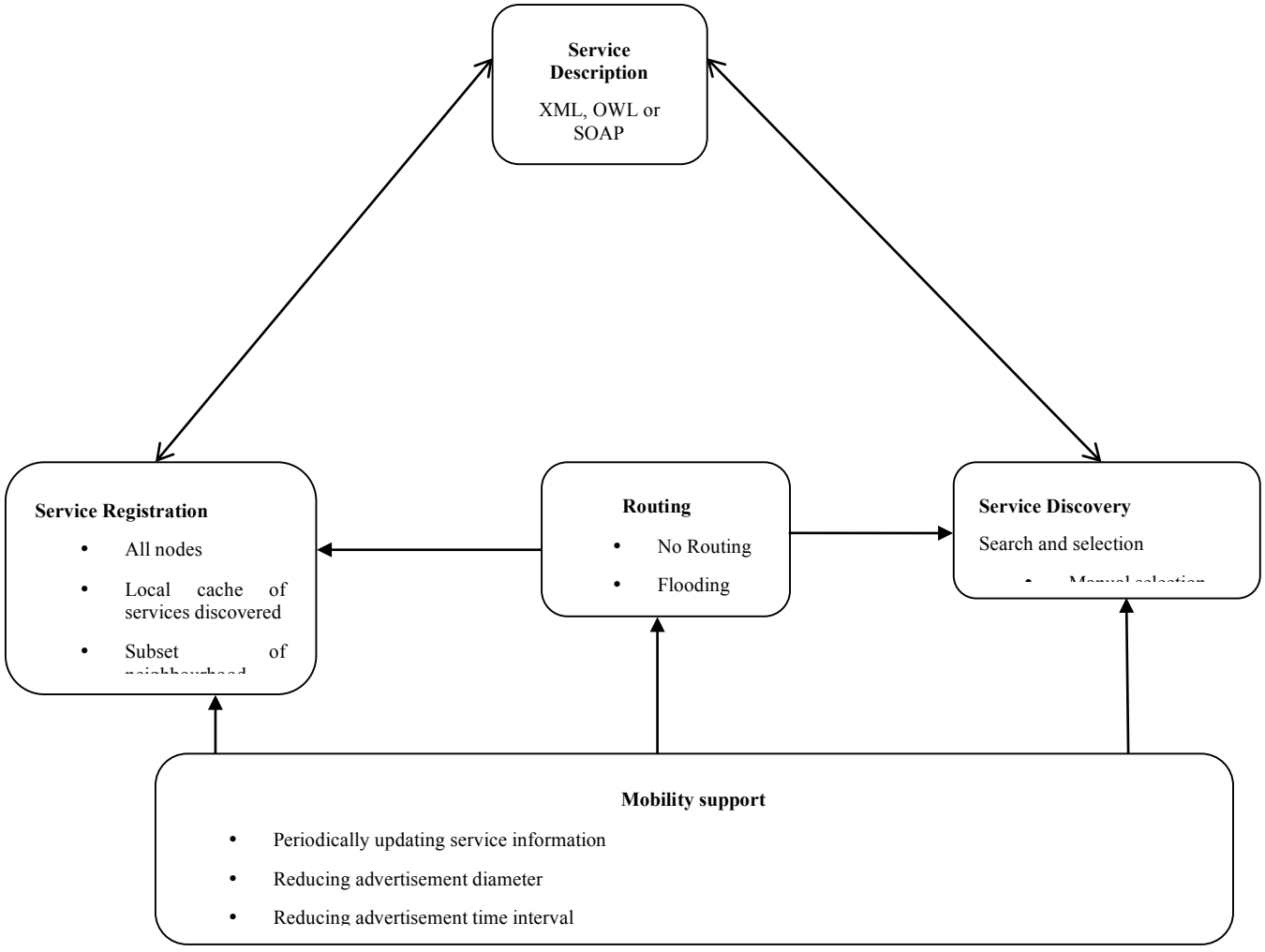

Figure 8 Interaction of Components

2.11.1.2 Conceptual view of node interaction

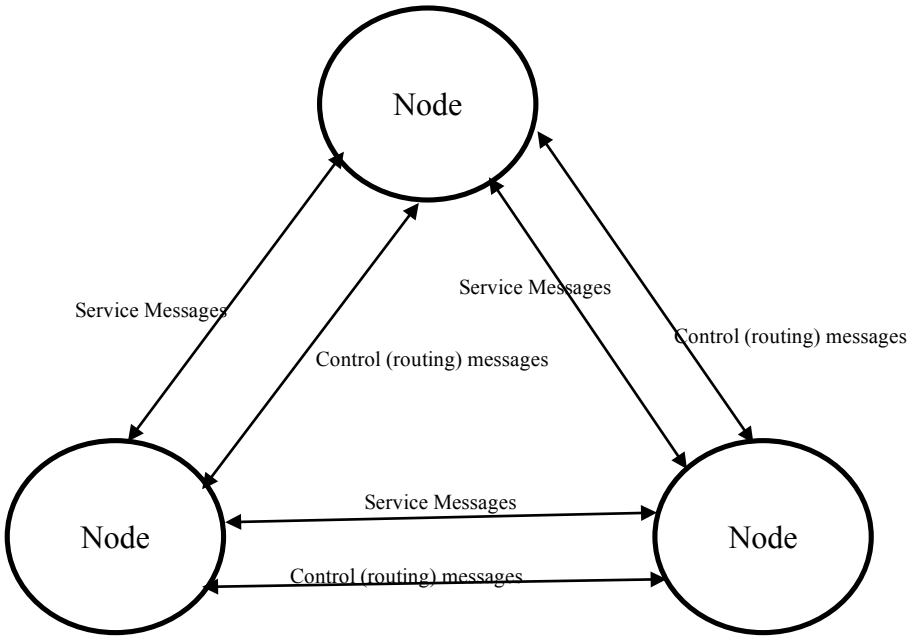

Figure 9 Conceptual View of Node interaction

CHAPTER 3

RESEARCH METHODOLOGY

3.1 Introduction

In this section we will discuss the procedures that we applied in developing our solution. Research methodology is a systematic way to solve a problem (Rajasekar et al, 2011). The procedures by which researchers go about their work of describing, explaining and predicting phenomena are called research methodology.

3.2 Research Design

Research design is defined as the clearly defined structures within which the study is implemented (Burns and Grove 2001:223). The research design method for this project is Exploratory research and Application Development.

6.3.1 Exploratory research

Exploratory research is used to develop a better understanding (Hair et al, 2003). Exploratory research is applied to find out what is happening, to seek new insight, to ask questions and to access phenomenon in a new light. The approach for this research was initially be a review of work in the field of Service discovery in Mobile Ad hoc networks. This guided the approach we took in implementing a framework for service discovery for an Android OS based MANET.

6.3.1 Application Development

The implementation is a standard mobile ad hoc network implementation; it provides delivery and routing of data between nodes, subject to the spontaneous introduction and removal of any node at any time, on Android OS. In addition, it also provides Service Discovery capability. The work flow for the implementation of the project was: -

- An initial ad hoc network middle-ware that runs on Android. This was the basis of our implementation of a Service Discovery middle-ware and provided basic routing and connectivity options.
- The Service Discovery middle-ware. The project explored several protocols and approaches for the implementation.

The middle-ware provides Service Discovery functionality to other applications via API calls. The scope of the project was limited to exploring the best approaches for Service discovery middle-ware on Android. We addressed other issues related to the ad hoc network only to facilitate that objective.

6.3.1 Information Sources

There has been a considerable amount of research in the past in the field of mobile ad hoc networking. There are studies covering specific areas of Mobile ad hoc networks. Most of the current academic research is in developing the algorithms and ideas to solve the various

challenges of mobile ad hoc networks. Protocols and theories of Mobile Ad hoc Networking have also been tested. Our research dwelt on protocols and algorithms for implementing android ad hoc networks.

3.3 System Analysis and design

Our service discovery functionality is implemented as an overlay, on top of an underlying MANET network layer. It is implemented at the application layer.

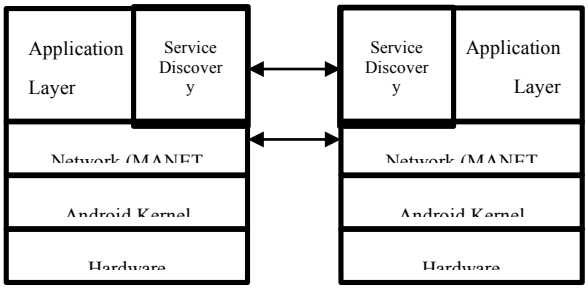

Figure 10 MANET layer Hierarchy

As seen in Figure 10 , Service Discovery functionality was implemented at the application layer. Our implementation relied on the underlying MANET routing implementation for addressing and node identification. The implementation of routing for the Android Mobile Ad-hoc Network is outside the scope of this work and will therefore not be discussed in this document.

Our Service discovery implementation is modeled by the Figure 11 below.

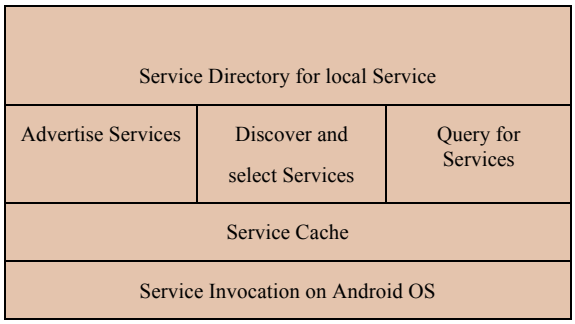

Figure 11 Service Discovery Modules

3.4 Android Service Description Implementation

3.4.1 Considerations

The operating system targeted for our proposed implementation was Android. All assumptions of a MANET environment made earlier were considered.

OWL-S offers sufficient semantics to describe services. However, the unique characteristics of MANET operation limit OWL-S applicability in such networks.

- Nodes are continuously moving hence there is a challenge in maintaining a common ontology reference across all nodes.
- Our implementation targets devices that will be heterogeneous in many respects such as different android OS versions, different hardware capabilities.

This calls for a Service Description implementation that will allow for a reasonable degree of heterogeneity but still have sufficient data to allow nodes to support mapping across the multiple ontologies.

We propose a hierarchical description of services (Figure 3), where the service description will be more specialized lower the hierarchy and generic at the top. Services can then be advertised and discovered at any level of the tree.

3.4.2 Service tree

The highest nodes serve as parents to more specialized services further down the tree. In the illustration below Figure 12, the topmost parent is class service. Its children are services and hardware. Hardware includes printers and storage. Service has children communication, document utilities, gaming and multimedia.

Multimedia for instance would have audio applications category. This could further be sub-classed on the basis of audio format that the services can handle.

It is worth noting at this point that parameters such as functional and non-functional requirements of the services will not be brought out in this hierarchy. These will be handled in the section that will follows. The approach achieves the following objectives: -

- Faster service matching. The matching algorithms only have to identify a parent node then navigate down the tree to the service required. This leads to lower resource and power utilization.
- Facilitate service advertisements. Advertisements can be done at any level of the tree. This facilitates the discovery of services especially in cases where there are heterogeneous nodes.

Example Scenario: Find a VLC music service to play a music track

The service request would contain both keywords audio and VLC. Supposing the network does not have a VLC service, then the service searching algorithm would then resolve the request by referring to the audio node and returning other similar services based on the request's other parameters. If a suitable match is still not found, then the algorithm would move further up and try the multimedia node. Without such a service description the request would have been dismissed.

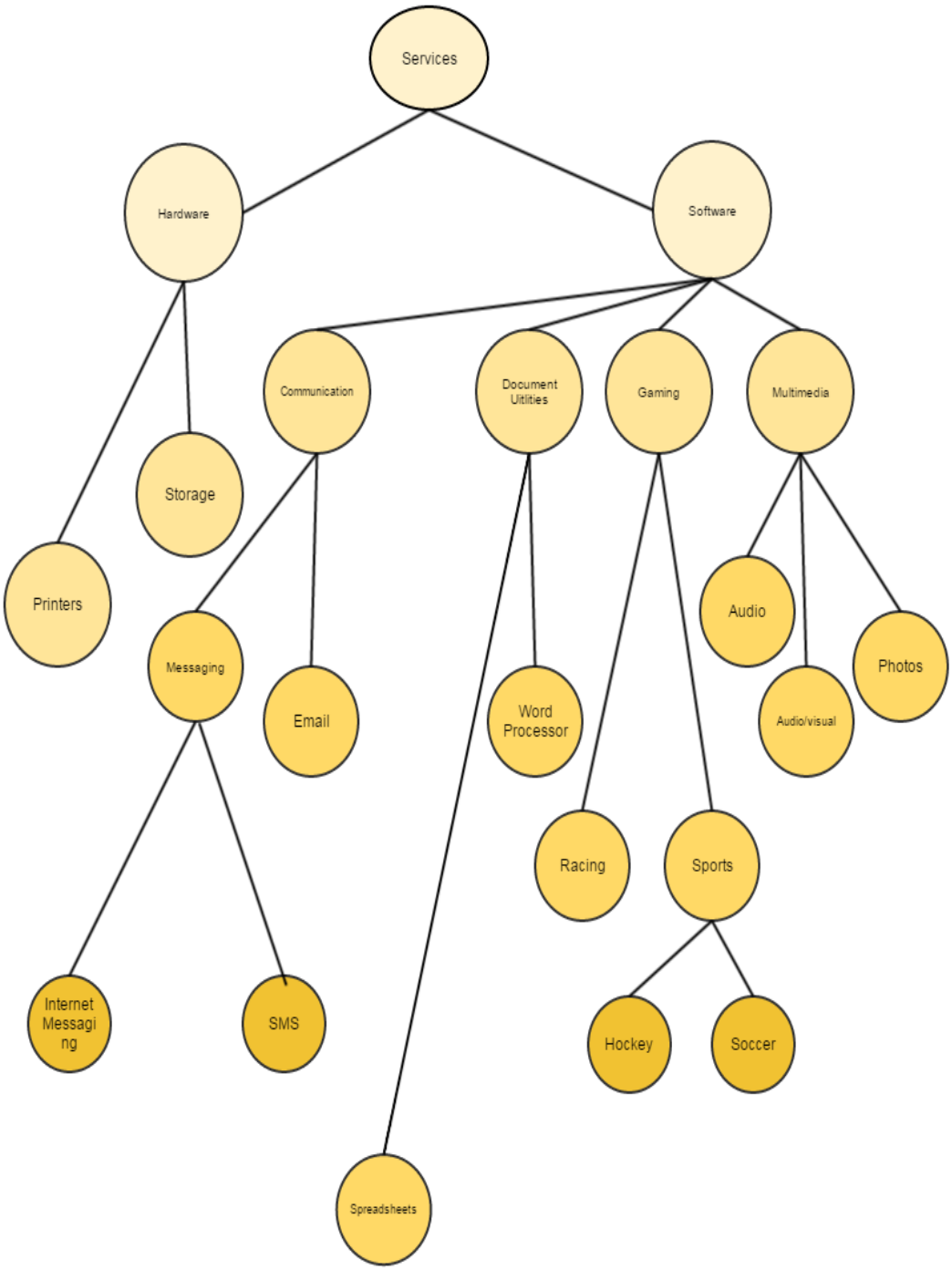

Figure 12 Service Tree

3.4.3 Service Description Document

Components of the document

The Service Description is loosely based on OWL-S. Key elements and functions of the service description are:

The service profile

This is the equivalent of the OWL-S service profile. It contains the input, outputs, preconditions and expectations of the service. These are functional and non-functional requirements of the service

- Inputs refer to the parameters that are required to be supplied to the service.
- Preconditions include authentication details, service compatibility (android OS versions), and service validity period.
- Expectations-This link in OWL-S is the expected state of a service after it has executed.
- Outputs

Service functions

This is an analogy of the OWL-S profile model. It contains the service composition and processing logic such as process flows. Like OWL-S described services, services will either be atomic, simple or composite. Atomic services exist on their own whereas composite services are an aggregate of services.

```
<service>
<profile>
<!-- The service profile-->
<version></version>
<serviceName></serviceName>
<servicePath></servicePath>
<description></description>
<contactInformation>
   <actor>
      <name></name>
      <deviceType></deviceType>
   </actor>
   <node-address></node-address>
</contactInformation>
<hasInput name="userName" dataType="string"/>
<hasInput name="date" dataType="date"/>
<hasPrecondition authentication="false"/>
<hasOutput name=" " dataType="string"/>
<hasResult value=" " />
</profile>
<operations>
   <operation>
      <name></name>
      <description></description>
      <parameter>
         <name></name>
         <description></description>
         <dataType></dataType>
      </parameter>
       <parameter>
         <name></name>
         <description></description>
         <dataType></dataType>
      </parameter>
       <returnParameter>
         <name></name>
         <description></description>
         <dataType></dataType>
      </returnParameter>
   </operation>
</operations>
</service>
```

Figure 13 Service Description

3.4.4 Service Advertisement Implementation

Service advertisement is the mechanism by which a node notifies other nodes in a Mobile Ad-hoc network of its services. Service advertisement is a core component of our work. An android OS based ad hoc system would consist of devices with varying capabilities. Devices would include mobile phones, smart watches and domestic embedded household devices. This would introduce variations such as:

- Different memory capacity
- Variable processing power
- Communication range
- Multiple operating system versions

Our implementation of service advertisement took into consideration this heterogeneity of the network. The desired properties of the implementation were:

- Minimal message size
- Minimal communication overhead
- Support for the diverse capabilities of Android devices
- The implementation should minimize on the use of memory resources

We have separated the implementation into components based on functionality. Each component presents a unique implementation consideration. The components are

i. Service advertisement message implementation
ii. Implementation of message broadcast mechanisms
iii. Management of service message caching on nodes
iv. Service delivery and invocation
v. Management of security in the implementation
vi. Node participation management
vii. Node addressing and identification mechanisms

3.4.4.1 Service advertisement message

The service advertisement message contains information about services that will be broadcast across the network from source nodes. Service advertisement and discovery will follow either of two approaches. Passive push will entail a server node broadcasting service advertisement messages in the network.

A service advertisement contains:

- The service name
- Address of the host node
- The path of the service on the service tree and a keyword
- Time to live header
- Hop count
- Originating time
- Advertisement id
- Device version

The hop count field is incremented at each node as the message is forwarded. The hop count describes the distance covered by a message.

Time to live sets the validity period of a message. The message is expunged from the nodes after the TTL has expired.

The originating time describes the time the service advertisement was generated.

The unique id identifies an advertisement message and is used to disambiguate duplicate advertisements. It is generated as a UUID appended to a node id.

Device version specifies the minimum android OS version

To advertise, a node either sends a keyword or a path from the service tree. If both are included, the keyword takes preference.

```
<service_advertisement>
   <profile>
      <keyword/>
      <path/>
      <device_os_version/>
      <service_id/>
      <node_address/>
   </profile>
   <advertisement_data>
      <time_to_live/>
      <time_created/>
      <hop_count/>
   </advertisement_data>
</service_advertisement>
```

Upon receiving service advertisement messages nodes cache then. To consume a service, nodes first search in their local cache. If successfully matched, the node then requests for the service description from the host node, by sending a service discovery message. The node can then start the service invocation process.

6.3.1 Service Directory

Messages will be stored in an SQLite a database and a memory cache. The memory cache will optimize the performance of the memory for records that are in frequently accessed. Records will be initially persisted in memory before being pushed to the database. Incoming records will be added to the cache before being moved to the SQLite database. The structure is illustrated below

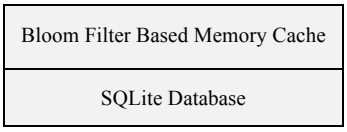

Figure 14 Message Memory

Nodes will cache service advertisements received for a specified period of time. A high frequency of messages in an Android network would result in draining device power besides clogging the network. Caching service advertisements in nodes will reduce the frequency of service discovery requests being propagated across the entire network. Node caches can handle service requests without having to propagate them across the network.

If (duplicate advertisement) {

Update the advertisements time created parameter (therefore its lifetime)

Discard message

} else{

Extract service information

Add the information to a cache

}

A typical usage scenario would be:

Node requires service named music player

Check cache

Is service description in cache?

If true read advertisement

Else

Broadcast discover message in the network.

Android devices typically have a low memory. In networks with a large number of nodes, caches would grow large, consuming a lot of the limited memory resources. There is need to design the caching mechanism in such a way that minimal resources will be occupied.

Before a search is done in the database for a message advertisement, the cache will first be searched to ensure the record exists. In order to optimize search performance, we shall use bloom filters to guarantee the certainty of a search before it is done.

Bloom filter

A Bloom filter is a simple space-efficient randomized data structure for representing a set in order to support membership queries. The Bloom filter is a way of using hash transforms to determine set member- ship. Bloom filters are efficient in memory usage as opposed to a store everything approach. Our implementation uses bloom filters to minimize the memory footprint

of node caches and optimize search. A search for services will first be executed on a cache implemented with a bloom filter. The project implements a Java based bloom filter. When an application searches for a service, a query will first be executed on the bloom filter to verify whether the service is present. If the service is present, the search then extends to the SQLite database otherwise a false result is returned.

Pseudo code for a bloom filter(Tarkoma,S et al)

A Bloom filter is an array of m bits for representing a set

$S = \{x_1, x_2, \ldots, x_n\}$ of n elements. Initially all the bits in the filter are set to zero. The key idea is to use k hash functions, $h_i(x)$, $1 \leq i \leq k$ to map items $x \in S$ to random numbers uniform in the range 1, . . .m . The hash functions are assumed to be uniform. An element x of set S is inserted into the filter by setting the bits $h_i(x)$ to one for $1 \leq i \leq k$. Conversely, $y \in S$ if the bits $h_i(y)$ are set, and guaranteed not to be a member if any bit $h_i(y)$ is not set.

Bloom filter Insertion

Data*: x is the object key to insert into the Bloom filter.*

Function*: insert(x)*

for *j : 1 . . . k* ***do***

/ Loop all hash functions k */*

i ← $h_j(x)$;

if *B_i == 0* ***then***

/ Bloom filter had zero bit at position i */*

B_i ← 1;

end

end

Bloom filter query

Data: x is the object key for which membership is tested.

Function: ismember(x) returns true or false to the membership test

```
m ← 1;
j ← 1;
        while m == 1 and j _ k do
                i ← hj(x);
                if Bi == 0 then
                        m ← 0;
                end
j ← j + 1;
end
return m;
```

Nodes also specify a fixed memory size allocation for service caches. When the cache gets full, a lowest lifetime policy is applied to remove advertisements from the cache

3.5 Broadcast Mechanism

3.5.1 Introduction

We discuss the forwarding of messages from node to node across the network.

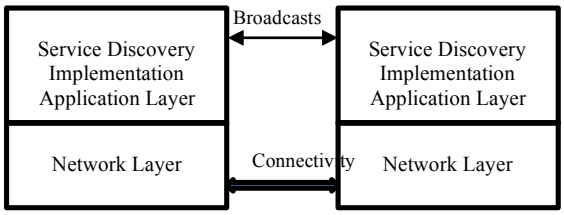

Figure 15 Service broadcast layer

Service broadcasts are done at the application layer. Operations at this layer are abstracted from the underlying network layer. Broadcast messages relay service advertisement and discovery messages.

Service specific messages initiated by nodes will piggyback on broadcast messages. Service messages are appended to the broadcast messages for relaying across the network. Service advertisement messages will be broadcast across the nodes. If a node is interested in a service, it requests a service description from the host node, which will be relayed back using a reverse routing mechanism discussed in this document.

3.5.2 Format of a broadcast message

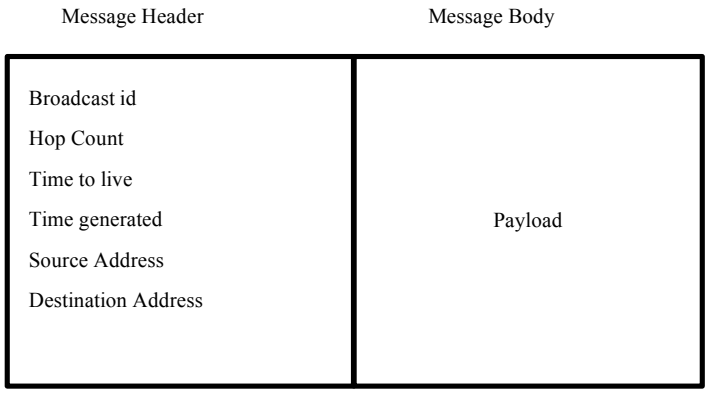

Figure 16 **Message Structure**

A broadcast message consists of the following fields:

- Broadcast id- This is the unique identifier for each broadcast. The broadcast id consists of the node id appended with an incremental id that will be unique at the node where it will be generated.
- Hop count This field will indicate the number of nodes a message will be allowed to transverse. It is a node specific value. The hop count will be decremented at every node as the message is propagated across the network.
- Time to live(ttl) - This specifies the validity of a message. The time to live parameter will have a direct bearing on the broadcast longevity. This is not to be confused with the service advertisement's time to live parameter. It determines whether a broadcast can continue to be propagated. On the other hand, the service description *ttl* determines when the service should be expired from the cache.
- Time generated- It is the time when the broadcast was generated.
- Source address – This is the address of the last node that sent the message. It will be updated at every node as the message moves along.
- Destination Address (optional field) – The destination address will be the address of the node where the message is headed. It is populated for direct message routing and will be set at broadcast time in broadcast mode.
- Content- This will be the service specific messages such as service advertisements, service discovery messages and request replies.

Protocol evaluation results discussed in an earlier section have demonstrated that counter based broadcast schemes are more effective than probabilistic schemes in simulated environments. This informs our decision to develop a counter based approach for node broadcasts. In addition, a probabilistic scheme would require computation on nodes thus introducing a bottleneck in processor power when there are multiple broadcasts.

The longevity of broadcast messages will be limited based on a time to live parameter. The range covered by the messages will be limited by a hop count parameter. The importance of this is to avoid stale messages and avoid continuous broadcasts which would cause a broadcast storm. There is also a need to conserve devices resources. Most Android OS based devices are battery powered.

The setting for which our implementation will be designed for is as follows.

- The network will consist of Android OS devices such as smartphones, smart watches and so on. These are typically resource constrained in terms of memory, processor capability, power and bandwidth.
- Devices will be highly mobile and connections will be formed spontaneously
- All nodes will co-operate in a broadcast
- There may be stationery nodes with higher capabilities than other nodes in the network

Key considerations in designing our broadcast mechanism are:

- The need to minimize the number of broadcast messages. This can be done by maximizing the number of non-transmitting nodes. This rules out the use of brute force flooding. We expect that in an Android OS network, there would at be nodes that serve as routing nodes for other nodes.
- It is important for nodes to cooperate in broadcasts and not be selfish as to refuse to use their resources to propagate messages. An incentive mechanism however is out of the scope of this work.
- Devices are resource and bandwidth constrained.
- The devices might not have sufficient storage to be able to maintain connection information.

3.5.3 Broadcast Algorithm

We have devised a broadcast algorithm that extends Ad-hoc on Demand Distance Vector (AODV) routing protocol to support neighbor aware routing. The purpose of introducing neighbor aware routing is to reduce control packets traffic by minimizing the transmitting nodes. We discuss the implementation of AODV and the modifications done.

3.5.3.1 Neighbor Discovery Implementation

We describe a protocol where nodes have knowledge of their neighbors. Our approach to implementing neighbor knowledge is informed by neighbor knowledge implementations by (Surcec and Marsic,2010) and (Kumar and Kumar,2010).

This information is used to optimize re-broadcasts. Nodes obtain 2-hop neighbor knowledge from ‘HELLO’ packets. Each node decides to rebroadcast based on knowledge of which of it’s other 1-hop and 2-hop are expected to rebroadcast.

Supposing node B receive a broadcast message from node A. Since node A is a neighbor, node B knows all its neighbors, common to node A that have also received node A’s transmission of the broadcast message. If node B has additional neighbors not reached by Node A’s broadcast, node B schedules the message for delivery, if it also satisfies the rule discussed below.

A decision to broadcast also depends on which node has a higher priority for rebroadcasting. The priority is proportional to a nodes number of neighbors(cardinality). The higher the nodes cardinality, the higher the priority.

Nodes periodically advertise their presence by sending out HELLO packets to their neighbors. This makes the neighbor discovery process inapplicable to AODV which is an on demand protocol.

We have adapted this neighbor knowledge protocol for ADOV, with the objective being to exchange HELLO messages, while still maintaining the reactive behavior of AODV.

3.5.3.2 AODV implementation

A node wishing to communicate with another node first looks for a path to the destination in the routing table. Each routing table entry consists of the following fields:

- The destination node address
- Next hop address
- Destination node sequence number
- Hop count.

If a route exists, the node simply forwards the message to the destination. Otherwise, it saves the message in a message queue, and then it initiates a route request, RREQ, to determine a route. The RREQ message is the broadcast in the network. The RREQ consists of:

- The source address
- The source node's current sequence number. This number is incremented for every message generated at the node
- Destination node address
- Destination sequence number
- The broadcast ID. This is incremented each time a node issues a RREQ message. The broadcast ID and source address uniquely identify a broadcast.

Sequence numbers serve to identify the freshness of a message. Each time a node sends out any type of message, it increases its own sequence number. Each node records the sequence number of all other nodes it talks to. A higher sequence number signifies a fresher route.

If a node has a valid route to the destination or if the RREQ reaches the destination node, the node initiates a Route Reply (RREP) message. The node also creates a reverse route entry in its routing table. The routing table contains:

- The address of the source node
- Number of hops to the source node
- A time to live value to set the lifetime for the message. If the message expires before a response is received, the node rebroadcasts an RREQ, this time with a longer TTL.
- The next hop

Route maintenance is done to maintain a route. When an intermediate node moves, a route Error (RRER) message is sent. Nodes receiving the RRER update their routing table to expunge the route entry. When a source node receives an RRER it will initiate a new route discovery. HELLO messages are also sent out to maintain routes.

3.5.3.3 Our Broadcast Algorithm Implementation

AODV suffers from control message overhead. This can be particularly noticeable in large MANET networks. It is possible to reduce the number of control messages forwarded by nodes by enhancing HELLO packets. This is done by introducing neighbor knowledge HELLO

packets. Selective forwarding of RREQ then follows and this reduces the number of nodes required to forward a Route Request Message.

HELLO messages will be exchanged by 1 hop neighbors every time a node joins the network and whenever a node moves. We will use the Android OS networking capability to obtain this information. The framework uses the Android Wi-Fi p2p API to determine a node's neighbors.

HELLO message

The HELLO message sent will contain:

i. The node address (of the source node)
ii. The node address of each neighbor of the node
iii. The degree of each neighbor of the node

The receiving node then uses this information to:

- Assess node transmission priority relative to other nodes. The higher the number of neighbors a node has, the higher it's priority.
- The 2-hop topology of nodes.

A node w satisfies the priority condition with respect to the node v if one of the following is true:

- deg(w) > deg(v). Degree of w is greater than v degree's
- deg(w) = deg(v) and w is encountered after v in the network path

deg-Degree implies the number of neighbors that a packet has.

RREQ propagation

In stable networks implementing AODV, Route Request messages are the highest proportion of messages exchanged. Route error messages, RERR, in highly mobile node networks are however not negligible as well.

Using the 2-hop neighbor information, nodes can probabilistically re-broadcast messages. When a node A and B receives an RREQ from node X, it uses its topology information to establish other neighbors that might have also received the message from X. If all all of B's neighbors are contained in neighbor A, and A has a higher degree than B (more neighbors, then B will not rebroadcast the message, otherwise it forwards the RREQ. Effectively, not all nodes, can forward messages.

Let $N(a)=\{n_1, n_2,\ldots n_k\}$ be the set of neighbors of node A

Let $N(b)=\{n_1,n_2,\ldots n_k\}$ be the set of neighbors of node B

Nodes A and B are neighbors and they both receive an RREQ broadcast from node X

If $N(a) \supseteq N(b)$

//process

if $|N(a)| > |N(b)|$

A rebroadcasts the message as it has a higher cardinality

Else if $|N(a)| > |N(b)|$

B forwards message as it has a higher cardinality

Else if |N(a)| = |N(b)|

Node with highest node id rebroadcasts the message

3.5.4 Scalability Challenge

In highly dense networks, nodes would have a large number of neighbors. Neighbor discovery using our algorithm and management of neighbor information would strain processing and storage resources on the Android OS devices from excessive HELLO packets. We will illustrate the effect of message frequency on processing power in Figure 24. We propose to limit the size of neighbor knowledge records that a node can have. This cache's size, along with other caches will be determine by configurations in the implementation. The implication of this is that the neighbor discovery will be effective only up to a limit determined by the device's capabilities.

As an improvement, we propose the implementation of a scoring mechanism, by which we could determine reliable nodes and thereby form an overlay network. Nodes that are not well scored will have a lower priority and could be left out in building the neighbor cache. The criteria for scoring could be the broadcast range of devices, message delivery success history of the node or device preferences. This is however out of scope of this work.

3.5.5 Pseudo code for message propagation

A typical message transmission. Each step is numbered for identification in the discussion that will follow.

1. *Node A transmits a message in the network with node id nwb_id*
2. *Node B receives the message.*
3. *If the received message represents the first copy of message with nwb_id*
4. *Write nwb_id and ID into a broadcast message*
5. *Otherwise STOP*
6. *If all neighbors of A=neighbors of B*
7. *STOP*
8. *ELSE*
9. *If hop_count > 0*
10. *Process received message*
11. *Decrement hop_count by 1*
12. *Send Broadcast*
13. *ELSE*
14. *STOP*
15. END

Step 3 is executed by applying a bloom filter query using the broadcast id

Step 10 involves the application of logic for caching and service matching mechanisms. Service matching is discussed in this paper.

NB

The time to live parameter determines the validity of messages. Message are expired and not processed after they exceed the time to live period.

3.5.6 Reverse Routing

Reverse routing in AODV will be used for responses.

3.6 Semantic processing of messages

We discuss management of broadcast messages using semantic techniques. Services are described using a semantic mechanism. In this section we will discuss: -

- Handling of advertisement messages
- Service matching
- Service discovery

Semantic message processing will revolve around service concepts. Concepts are derived from the Service tree. They comprise the tree nodes and service properties. Service matching will use the services semantic description. We discuss service matching using the implemented ontology. We consider an ad-hoc network $N=\{n_1,n_2 \ldots n_m\}$ as a set of mobiles nodes where $n_i \in N$ of size m. Each node describes its services using the semantic syntax discussed earlier.

Assume the sum of concepts on node o is represented by V_0^c . This is a sum of the node's concepts and those acquired from other nodes. *g* represents the maximum number of concepts a node can host.

$V_0^c=\{C_i,\ldots,C_n\}$

$V_0^c = \{ C_{kl} \mid C_{kl} \in V_k^o, k \in N \text{ and } 1 \le l \le g\}$

3.6.1 Service Advertisement processing

We describe the handling of service advertisement messages with an algorithm. The idea is to cache a message's concepts if the broadcast has not been encountered before. All rules described before apply. Assume a node k receives a message m originating from p. Message contains concepts $Cn=\{c_0,c_1\ldots..c_n\}$.

for *m.all concepts* ***do***

if *m.time_to_live >0*

$V_0^c = V_k \cup \{c\}$ *

m.cache←p.address

endif

end for

* Add node address to the cache against each concept encountered

3.6.2 Service Matching

Scarcity of computational resources will require a lightweight matching implementation.

Two concepts will be said to match if their respective names are syntactically correct and each of their properties match in scope and name. Our matching mechanism will apply a weighting mechanism with the idea being to find the closeness of concepts to one another where an exact match cannot be found.

From our Service Tree implementation of Service representation, one can see that services will be more closely related further down the tree. It is possible that the major differentiator further down the tree would be the inputs and outputs. At the very top of the tree the categories are highly generic so there would be no relationship.

Table 1 Concept matching weighting

	Service Tree Level	Weight
Concept Relationship	1	0
	2	0
	3	1
	4	2
	5	3

Let us represent weighted score after concept matching as W_s , the inputs matching is represented by I, output matching is represented by O. then the matching success S of the matching process will be the binary AND of each of the three tuples.

$$S = (W_s > 0) \wedge I \wedge O$$

Concept matching is symmetric $C_1C_2 \Leftrightarrow C_2C_1$.

3.6.3 Service Discovery Implementation

This is an active pull mechanism. The requesting node, a service client, will broadcast a service discovery request.

Clients will use a service discovery process to learn of new services in the network. The client first creates a message containing either a keyword or a service path from the service tree. A service discovery message is an expression of interest in consuming a service.

<service_discovery>

<profile>

<keywor/>

<path/d>

<device_os_version/>

<node_address/>

</profile>

<advertisement_data>

<time_to_live/>

<time_created/>

<hop_count/>

</advertisement_data>

</service_ discovery>

When a node matches a service discovery request, it will send a service description.

Service discovery will follow several stages:

- Node A interested in a service will broadcast a service discovery message in the network.
- Nodes that receive the message will apply a matching algorithm and return successful matches which will also have a weighted result.
- Node A will send a direct message to the node for which the highest score was returned, requesting for a service description.

Format of a service discovery reply message

If a service discovery request is successfully matched, a reply message will be generated, comprising the following fields:

- Original broadcast id. This is the broadcast id of the service discovery message.
- Matching score. This is the weighted score generated by the matching mechanism.
- Service host Address: The address of the node hosting the service requested
- Source address - The address of the node generating the response. This could be used to identify reliable and malicious nodes. If a source node consistently returns reliable responses, then it would be more trustworthy and could be relied on at the requesting node. On the other hand, persistently inaccurate nodes would be flagged as such. This is however out of scope for our work.
- Destination Address- This is the address obtained from the service discovery message
- Time to live – This field determines how long the response should be maintained in a cache.
- Node distance- This indicates the number of hops between the requesting and reply node. It is incremented at each node. The requesting node could use it to calculate the appropriate distance to the node hosting the service

3.7 Service Invocation

Once a node has identified a service.it will send a service request message to the host none. The host node responds by sending a service description of the requested service. Service invocation will rely on the MANET's underlying network addressing layer.

We assume a server client model for service invocation as shown in Figure 9. A typical service invocation would be a http based file transfer service to a node hosting a service.

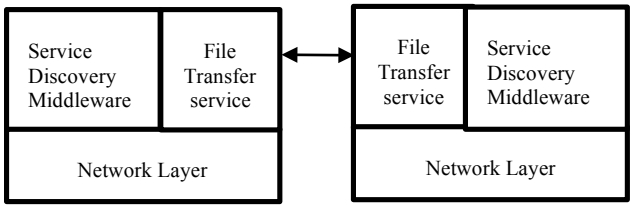

Figure 17 Micro HTTP server for service invocation from Node A to Node B

CHAPTER 4

PROTOTYPE IMPLEMENTATION

4.1 Introduction

In this section we discuss the implementation of the Android prototype. The implementation consists of 6 major modules:

- Service description management
- Neighbor knowledge management
- Message routing
- Matching mechanism
- Service advertisement mechanisms
- Service invocation

Each node has the above capabilities as summarized in Figure 18.

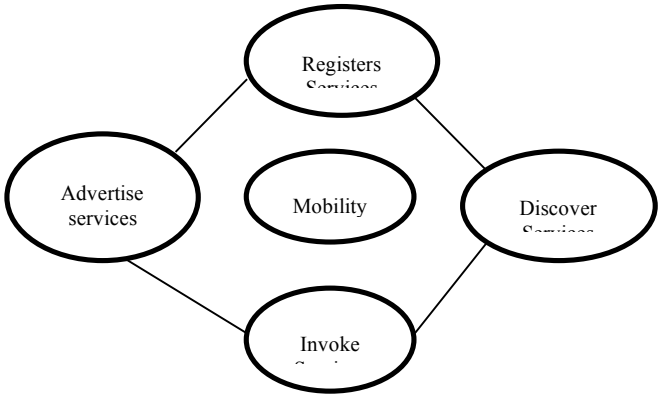

Figure 18 Interaction of components

4.2 Neighbor Knowledge Management

This is implemented using Android Wi-Fi direct framework, available since 4.0 (API level 14). All nodes send regular hello messages to their neighbor's that contain:

- The sending node's mac address
- The destination address
- A time stamp

- An incremental sequence number
- A list containing information on the node's neighbors.

We implemented an Android intent service that executes at a fixed interval, sending out hello packets from each node. The information sent on immediate nodes is incremental and is based of the most recent information received from each of its neighbors.

The messages received are saved in a data base table. Only one message is persisted for each node. This functionality requires that the WIFI service on the devices be enabled.

Refer to class *com.manetsd.hello.HelloMessage*

4.3 Service Description

Every node implements a database based directory. A description of each services hosted on the node is saved in the database. The database entities are related as illustrated in Figure 19.

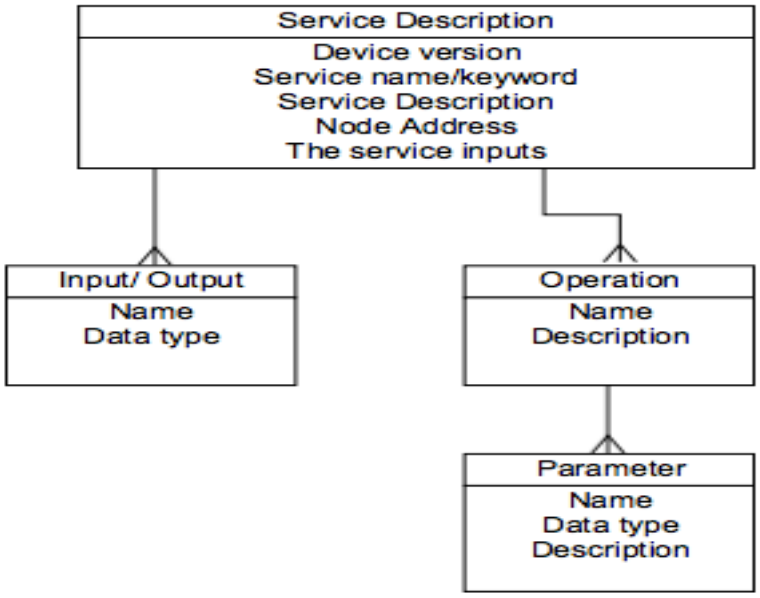

Figure 19 Service Description

Nodes define a service description for all their shared services.

A back ground service runs at regular intervals, to generate service advertisements from the supplied descriptions. This is an android intent service, that is started by an android Pending Intent. Refer to class *com.manetsd.advertisement.AdvertisementService.*

4.4 Service tree path

The service tree path is implemented as a character separated list of service concepts. Concepts represent nodes in the service tree. The concepts are more generic further up the tree. An example of a service path would be: Services, Software, entertainment, audio, mp3. The application has a mechanism to construct a service tree from the string above.

4.5 Service Advertisement

Service advertisements will be generated from the service descriptions stored on each node. A background task will execute at fixed intervals. Figure 20 illustrates a service advertisement flow.

4.5.1 Process flow for advertisements

Read all service descriptions D(desc)

for desc → D(desc)

if (desc is new)

Generate Advert

end if

end for

Service advertisements are created by reading service descriptions to extract required fields, which are the service name, its service tree path, the service description and the service's host node address. Service Advertisements received are cached.

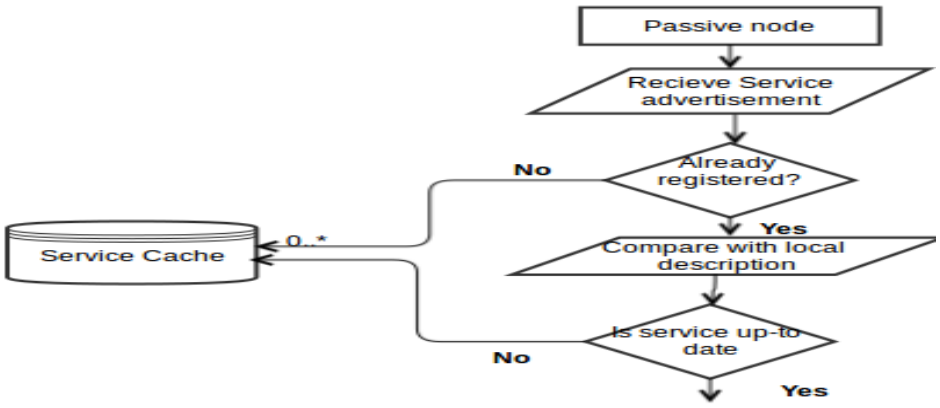

Figure 20 Service advertisement management

The implementation for the service description is contained in package *com.manetsd.advertisement.*

4.6 Message Routing

The application uses the Ad-hoc On Demand Distance Routing Vector algorithm with modifications for neighbor aware routing. Service Messages generated at the nodes are routed using the protocol. Nodes first deduce information on their peers before forwarding packets. Our implementation of AODV is contained in package *com.manetsd.routing.aodv.Aodv.* When a node receives a route request, RREQ, message if first establishes whether it is eligible to forward the message. In order to do this, it uses the peer information collected in the earlier functions.

A single class serves as the façade for incoming and outgoing messages.

4.6.1 Service Messages

Messages are in XML format. They are generated by converting Java domain classes to XML. A Service message consists of two parts:

- A message header
- A message payload

Message Header	Message Payload

Service messages are generated by individual modules. They are then modified centrally to append message header information which consists of:

- Origin addresses
- Destination addresses
- Message validity data
- Message payload type

The message header information facilitates the routing of messages at nodes to the intended modules. The message payload is the actual message content being sent. We identified 5 message payload types:

a. Service Advertisements

Sent by nodes to advertise services that they host

b. Service Information Requests

Sent to a specific node to request information on a specified service. The node responds with a service description of the mentioned service.

c. Service Discovery Request

Broadcast across the network by a node to search for services that meet some specified criteria. Nodes respond with a service discovery response message.

d. Service Discovery Response

This message is a response generated with respect to a service discovery request message. A service discovery response message has a score field, a value that is assigned after a matching mechanism has been applied at the node. Typically, multiple Service Discovery response messages would be sent to a node. The node will then use the score field to identify a service, with the highest value representing the best match. See classes *com.manetsd.xml.matching.Match* and *com.manetsd.matching.ServiceMatching.*

e. Service description

A service description message is sent in response to Service Information Request. This is the service description discussed earlier in this section.

f. *Service Matching*

When a node sends a request for a service, a service matching exercise is done on each receiving that receives the node. Services are matched either by keyword or by path. Keyword matching entails a simple match between the keyword provided and a service name. Path matching involves navigating the service tree for an appropriate match based on an algorithm we have developed.

Searching for a service on a node is involves two stages:

- Searching from the cache (Implemented using Bloom filters).
- Searching from the database

4.7 Matching Mechanisms

4.7.1 Overview

Match by Service Name

- Check whether the keyword is present in the cache.
- If present, search for the service from the Service Description table
- Search for the service from the Service Advertisement table

Match by Path

The service tree path is implemented as a comma separated list of service concepts. The application generates a service tree from the path string. The concepts serve as nodes of the service tree. Steps executed in matching by path are:

1. Read all service tree paths from local service descriptions and service advertisement cache.
2. Extract concepts from each of the paths.
3. Generate a tree from the concepts
4. Search for the least significant concept from the service tree
5. If present, return a positive match.
6. If not present, decrement the matching score, then repeat step 4 with the next least significant string
7. Repeat 4 to 6 until a match if found.

4.7.2 Service Tree traversing

We have developed a tree data structure representing concepts in a service tree. To locate a concept, we execute a depth first search on the tree.

DFS(graph G, Vertex v)

// Recursive algorithm

for *all edges e in G.incidentEdges(v)* ***do***

if *edge e is unexplored* ***then***

w = G.opposite(v, e)

if *vertex w is unexplored* ***then***

label e as discovery edge

recursively call DFS(G, w)

else

label e as a back edge

See com.manetsd.matching.servicetree.DepthFirstSearch

Example

Services,Software,entertainment,audio,mp3-----→ most significant concept

- The path has a cardinality of 5.
- If a successful match is returned for 'mp3' then the match score is 10.
- If a successful match is returned for 'audio' then the match score is 8.

The application defines the highest point on the service tree above which no further matches can be done. This is because service descriptions are generic further up the tree. Figure 21 illustrates this.

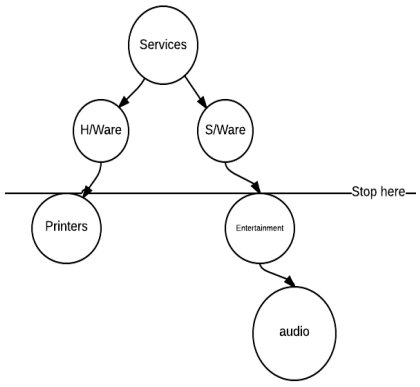

Figure 21 Service Tree Matching

6.3.1 4.7.3 Service Match response

A matching response is returned for successful matches. The match response message contains the service tree path, node address and a score. This message is sent back to the requesting node. The requesting node upon receiving match responses selects the best match from the score and sends a request service info message to the node address of the selected match.

4.8 Message Flow

Figure 22 summarizes message flow in the framework as discussed in previous sections.

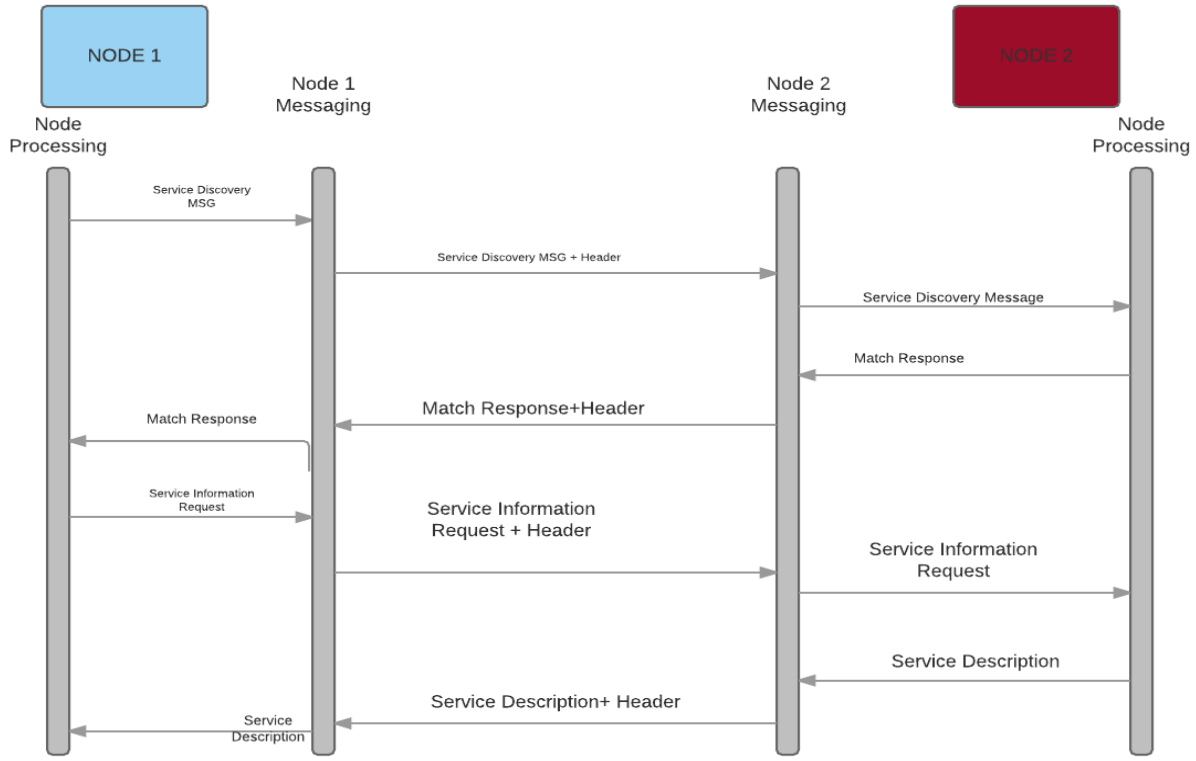

Figure 22 Information flow in the network

CHAPTER 5

EVALUATION AND RESULTS

5.1 Introduction

We evaluated the routing implementation using a simulation. The prototype was also evaluated by varying network characteristics and measuring specific parameters.

5.2 Network Performance

5.2.1 Routing

Nodes acquire 2 hop neighbor knowledge. The HELLO packets are modified to include information on node neighbors.

2-hop Neighbor knowledge is used for the selective forwarding of routing control messages. The expected result is a reduction in control message traffic. We will use Route Request (RREQ) control messages to access the performance of the routing implementation.

The volume of RREQ requests in a MANET implementing AODV is a factor of:

- MANET size It increases with the number of nodes. It is affected by the size of the routing table in nodes
- Validity period of RREP. The longer route table entries are retained the fewer the RREQ requests that will be required to replenish the table. This however introduces the possibility of stale data in routing tables.
- The MANET topology.

The results in Table 2 below were obtained from a simulation. Fixed parameters were

a. 1 message every 5 seconds
b. Time to Live 5 seconds
c. Simulated hello loss of 2%

Table 2 RREQ control messages count

Number of Nodes	RREQ Count with 2-hop neighbor Knowledge	RREQ Count without 2-hop neighbor Knowledge
10	9	11
20	33	34
30	180	201
40	347	379
50	351	407
60	363	424
70	379	437
80	401	485
90	426	501

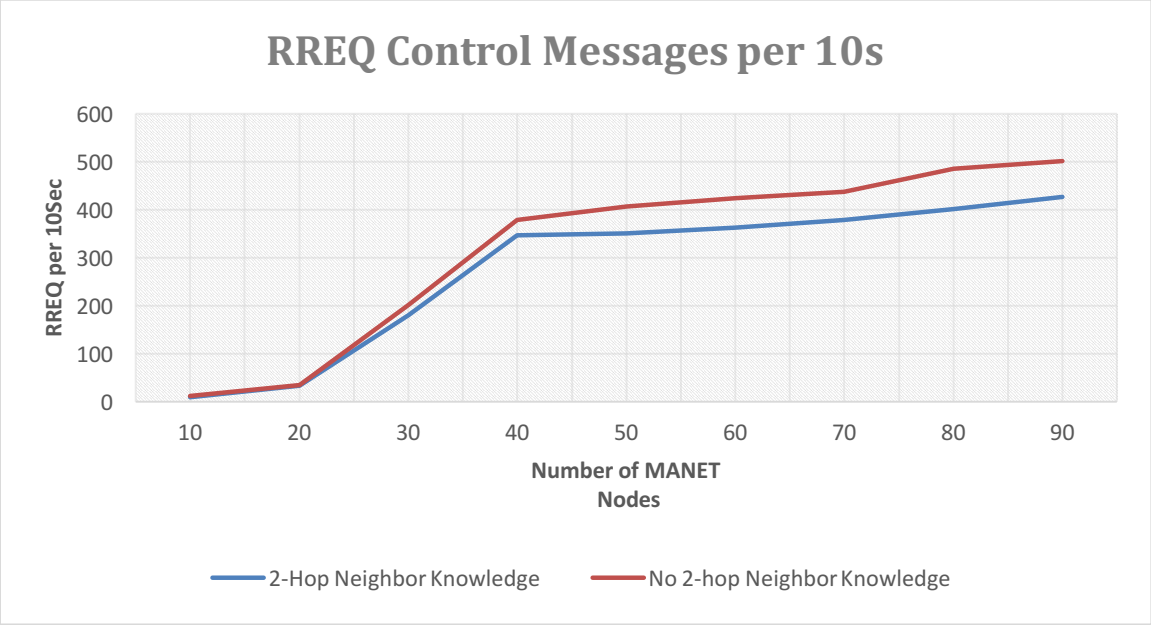

Figure 23 Comparison of RREQ count in AODV and AODV neighbor aware MANET

Deductions

The efficiency, which is the gradient on Figure 23 of the simulation results, is more pronounced as the number of nodes increases. Our comparison used the number of RREQ control messages exchanged. As can be seen, the number of RREQ messages increases as the number of nodes in the network increases.

It is worth noting that in an actual there is an increase in power consumption as the number of messages is increased as more CPU power would be required to process the additional messages. This can be observed from Figure 24 in this document.

5.3 Android Prototype

The Android prototype was evaluated on multiple devices. Comparisons were drawn on performance of the devices selected based on:

- The android OS version
- Device processor speed
- Device memory

Parameters of interest were:

- Service discovery latency
  This is the time needed to resolve a service request to a valid service binding and contacting the service host successfully.
- Service availability
  We define this as the ratio of successful service bindings to the total number of requests obtained.

5.3.1 Experiments

Messages are a key component of our implementation. The various types of messages exchanged between the nodes consist of network connectivity messages (acknowledgments, hello messages) and service discovery messages shown in section 4.8. We will use the count of messages transmitted over a period of time to measure the performance of the application in our experiments.

Experiment 1

We deployed the application on multiple devices.

i. Random service requests were generated. All messages exchanged by the devices were logged into a database.
ii. The hops between the devices were then varied and the above step repeated.

Results were obtained from the database logs generated.

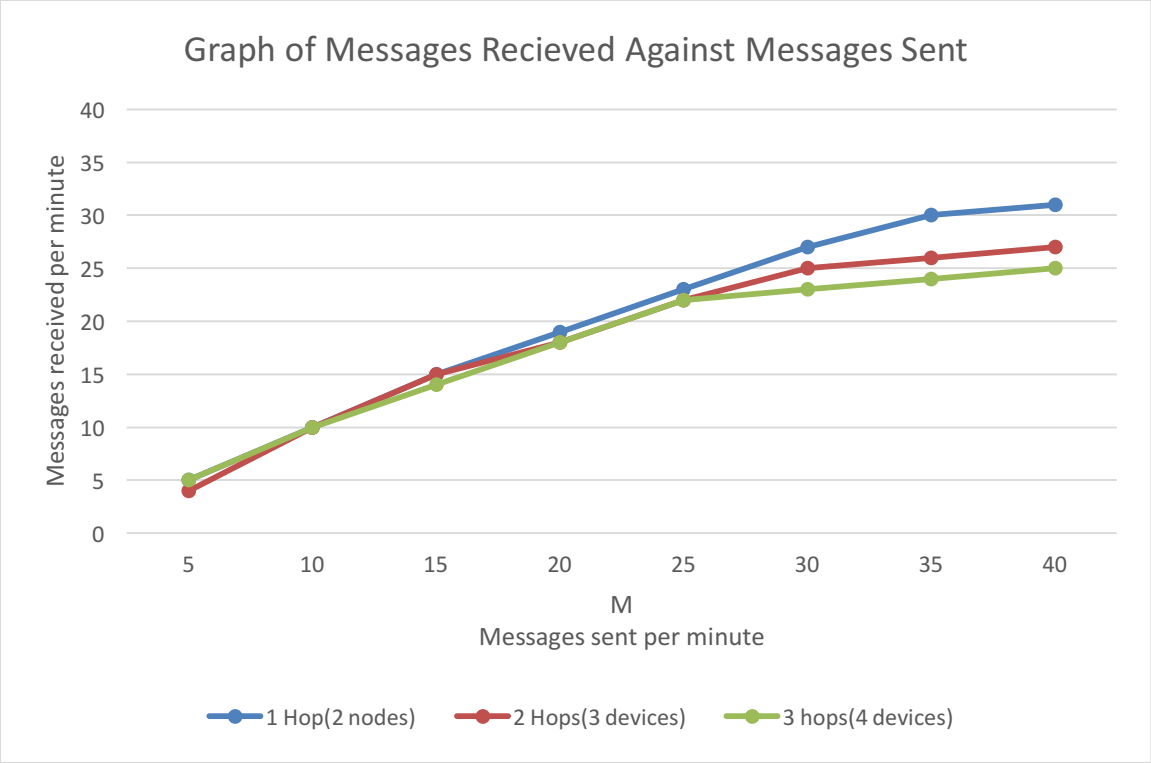

Figure 24 Chart of messages received against messages sent

It can be seen that some service messages are dropped, hence lost, at the nodes. The rate of message loss is not consistent. The pattern however is such that there is a higher loss of messages for higher rates of message transmission.

Experiment 2 Power Consumption for service advertisements (units mWA)

We observed the power consumption of the application on an actual device as the number of Service Advertisement messages was increased.

(Zhang et al, 2010) describe a technique for estimating android device power consumption. Based on this, they developed PowerTutor, an android power monitoring application. We shall use this application to monitor the apps power consumption over a duration of five minutes. The rate of messages was then gradually increased using a java timer. The power output recorded is a summation of application CPU processing power and the power consumed by the display. The results are tabulated in Table 3 and illustrated in Figure 25 .

Table 3 Power Consumption for Service Advertisements

Count/Time	0	60	120	180	240	300
10	179	166	156	158	151	169
20	223	165	230	256	233	235
30	337	354	342	369	327	363
40	406	388	392	406	435	458
50	477	512	470	434	531	579

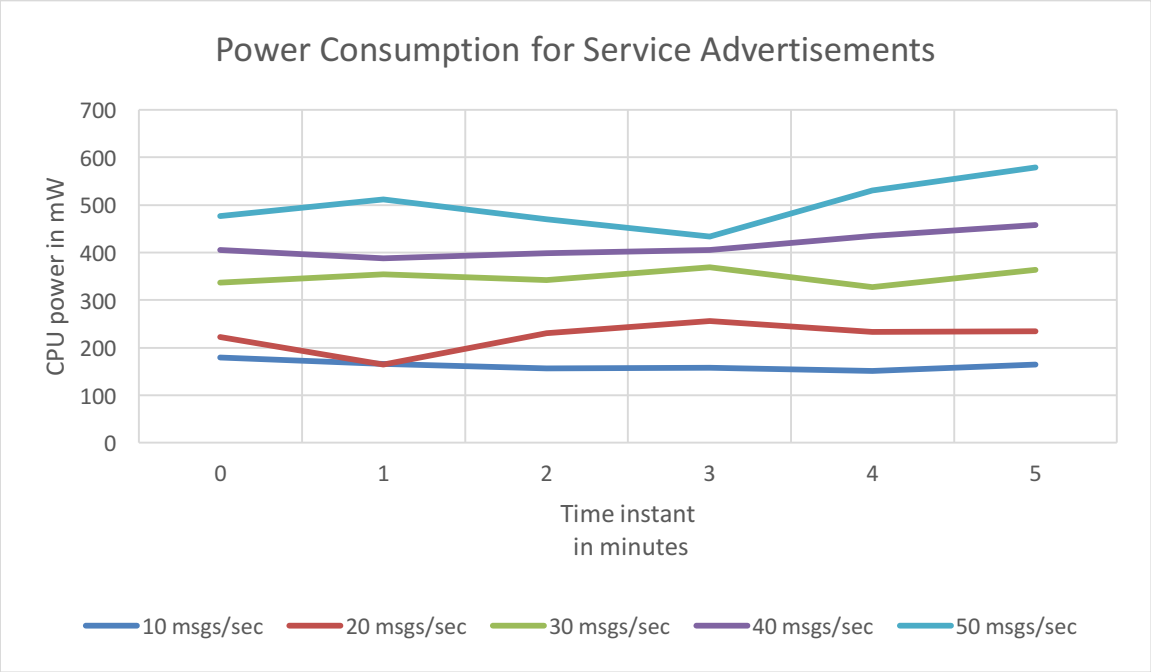

Figure 25 Chart of Power Consumption for Service Advertisements

The important observation here is the overall increase in power consumption as the messaging frequency increases.

Experiment 3 Matching Scenario

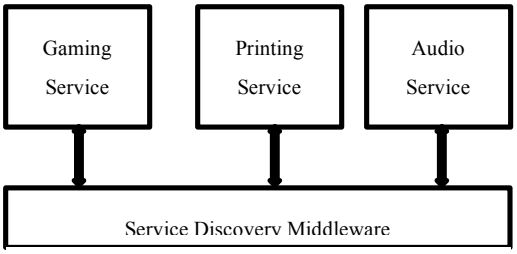

Figure 26 Interaction between services and SD middleware

We discuss an application scenario tested. Figure 26 shows the interaction of services with the service discovery middleware.

Audio player service

The aim of this service is to locate a service that can play an mp3 clip using the VLC media player. The invocation mechanism was not implemented. The service is a network application in a setting where there are several Android Devices.

We started by sending out a service request from the requesting device. Devices that received the request respond with a service match response. The requestor then selects the highest scored match.

We tested this scenario on 3 devices. Different service profiles were created for each of the devices. The service descriptions we composed for the devices are shown below.

Table 4 Sample service profiles

Device 1	***Device 2***	***Device 3***
<service> *<profile>* *<version>1</version>* *<serviceName>**jetaudio**</serviceName>* *<servicePath>**services,software,entertainment,,mediaplayer,audio,mp3,jetaudio**</servicePath>* *<description>**Music player**</description>* *<contactInformation>* *<actor>* *<name>**EugenePhone**</name>* *<deviceType>**Phone**</deviceType>* *</actor>* *<node-address></node-address>* *</contactInformation>* *<hasInput name="**track**" dataType="stream"/>* *</profile>* *<operations>* *<operation>* *<name>**stream**</name>*	*<service>* *<profile>* *<version>1</version>* *<serviceName>**vlc**</serviceName>* *<servicePath>**services,software,entertainment,,mediaplayer,audio,mp3,vlc**</servicePath>* *<description>**Music player**</description>* *<contactInformation>* *<actor>* *<name> **My HIFI** </name>* *<deviceType>**Phone**</deviceType>* *</actor>* *<node-address></node-address>* *</contactInformation>* *<hasInput name="**track**" dataType="stream"/>* *</profile>* *<operations>* *<operation>*	*<service>* *<profile>* *<version>1</version>* *<serviceName>**mediaone**</serviceName>* *<servicePath>**services,software,entertainment,,mediaplayer,audio,mp3,mediaone** </servicePath>* *<description>**Music player**</description>* *<contactInformation>* *<actor>* *name> **AndroidWearable** </name>* *<deviceType>**Phone**</deviceType>* *</actor>* *<node-address></node-address>* *</contactInformation>* *<hasInput name="**track**" dataType="stream"/>* *</profile>* *<operations>*

<description>**Play mp3 clip**</description> <parameter> <name></name> <description></description> <dataType></dataType> </parameter> <parameter> <name>**Stream**</name> </operation> </operations> </service>	<name>**stream**</name> <description>**Play mp3 clip**</description> <parameter> <name></name> <description></description> <dataType></dataType> </parameter> <parameter> <name>**Stream**</name> </operation> </operations> </service>	<operation> <name>**stream**</name> <description>**Play mp3 clip**</description> <parameter> <name></name> description></description> <dataType></dataType> </parameter> <parameter> <name>**Stream**</name> </operation> </operations> </service>

A service request was broadcast, to locate VLC player in the network. The response received is shown in Table 4 below:

Table 5 Matching Response

Device	Matching Score	Device address
Device 1	3	**EugenePhone**
Device 2	4	**MY HIFI**
Device 3	3	**AndroidWearable**

CHAPTER 6

DISCUSSION

6.1 Introduction

Our discussion on the framework will cover these key areas:

- Service description and advertisement
- Service Discoverability
- Network performance

6.2 Service description and advertisement

The service description was developed using a standard specification, OWL_S and XML. This makes it interoperable across devices.

6.3 Service Discoverability

Service Discoverability is the likelihood of finding a service in the MANET when it is present (Chakraborty, D,2006). In our implementation, service request messages have a hop count field, that determines the distance a service request can cover. The further a service request can go, then the higher the likely hood of discovering a service. Service discoverability is therefore directly proportional to the advertisement diameter and inversely proportional to the request hop count.

6.4 Network performance

The efficiency, which is the gradient on Figure 23, is more pronounced as the number of nodes increases. Thus introducing 2 hop neighbor knowledge in our AODV implementation has the net effect of improving efficiency.

The efficiency of this approach over plain Ad-hoc On-demand Distance Vector(AODV) can be assessed using an approach for comparing routing protocols, suggested by (Surces J and Marsic I ,2001). Applied to our implementation, efficiency e can be computed as:

$e=C_{NK}/C$

where e is the efficiency, C_{NK} is Route Request (RREQ) count with 2 hop neighbor knowledge present and N is RREQ count with 2 hop neighbor knowledge disabled.

Table 6 Routing Efficiency

Number of Nodes	Efficiency
10	1.22
20	1.03

30	1.12
40	1.10
50	1.16
60	1.17
70	1.15
80	1.21
90	1.18

The number of control messages transmitted in a network increases as the number of nodes increases. Efficiency of the middleware, which is the gradient on Figure 23, is more pronounced as the number of nodes increases. Introducing 2 hop neighbor knowledge in our AODV implementation has the net effect of improving efficiency. As the number of nodes increases, nodes consume more processing power in computing a routing path. There is a tradeoff between routing performance and processing power.

In Android devices, messages are lost when the device is overwhelmed to the point that it is not able to process and forward all messages it receives. The optimal point is a factor of the device's memory capacity and device processor. Better performance of the framework is achieved at lower message rates.

It follows therefore that the message response rate would also be affected by the number of inter-node hops. Node range is a characteristic of device WIFI capabilities. Nodes periodically advertise their services and other nodes cache these advertisements on receiving them. Caching makes service information available across the networks, thus reducing the need for service request messages to travel across multiple nodes.

6.5 Benchmarking

(Ranganathan et al, n.d.) proposed a benchmark for the assessment and evaluation of pervasive computing environments, that is applicable to our work. An evaluation of our framework based on the metrics suggested follows.

a) ***Precision and Recall***

This takes into consideration the percentage of correct hits among those returned as well as the percentage of all possible hits available. Applied to our implementation, this would essentially be a measure of the interaction of several aspects namely: service advertisement effectiveness, service request and matching mechanisms. Our implementation includes a shared ontology across the network. This facilitates a common understanding in the environment and consistence in interactions between nodes.

b) Context-Sensitivity

The context of participating devices is important in pervasive computing environments. Service messages contain a description of the originating node. Service descriptions contain information on their host nodes. Services are therefore discovered on the basis of their relevance to their situations. When searching for a music player for instance, one can tell whether the device located is a smartphone or a HIFI system. Our Service discovery implementation is therefore context aware. Context aware search enable devices to locate the most appropriate and relevant services.

c) Semantics

This they explain, is the extent to which service queries are semantic based as opposed to pure syntax. We implement service matching using concepts. This facilitates automation, and enhances search. We designed our implementation to use concept based search in addition to keyword search. Services are identified as a series of service concepts that group services based on a shared ontology. Service advertisements involve broadcasting concepts across the network.

d) Scalability

Does the protocol scale for large scale networks (networks with many nodes) and networks that have a lot of services? Scalability is evaluated with regards to the number of nodes, services and message traffic. Scalability is determined both by the device capabilities and the service discovery implementation. We have demonstrated from the tests that there is a progressive loss of messages as the number of hops is increased hence the implementation is scalable only up-to some point after which efficiency decreases. This is largely a factor of the android devices WIFI and memory capability.

CHAPTER 6

CONCLUSION

6.1 Overview

The performance of the implementation is discussed in chapter 5 of this document. We accessed the performance of the routing mechanism and effectiveness of the matching technique. Section 2.9 discusses the merits of semantic service discovery implementations.

This research has accomplished the following objectives:

- We have developed a service registration syntax for managing services in Android OS based MANETs. We also described a tree based approach for grouping services.
- We have developed functionality for routing service advertisement and service discovery messages that is bandwidth and power efficient.
- The research has also demonstrated an implementation for service invocation for service discovery platforms in Android OS MANETs

We discuss our findings in the next section

6.2 Service Discovery Implementation for Android OS MANETS

In this section, we shall briefly mention our findings on developing Service Discovery implementation for Android OS. We shall also discuss briefly the merits of each approach, and its impact on overall performance.

6.2.1 Service Description and Service Matching

Services are described using XML. In developing our ontology, we avoided re-inventing the wheel and reused components of OWL-S. Our service description contains elements that describes a services profile, its operations and dependencies.

By employing semantic techniques, it is possible to select the most relevant and appropriate services in an Android MANET. We also defined a novel way of grouping shared Android Services, a service tree. Services are the named and matched using concepts. Semantic service description facilitates automatic service discovery. Additionally, in Android OS network, nodes are highly heterogeneous in many respects, such as OS versions, device memory and processor capabilities. Services description should therefore be implemented in such a way that discovery and matching takes into consideration the context of each node. We have developed a scoring mechanism for service matching responses. Match result scores enable devices to locate the most relevant and appropriate services. Using a concept based search, the most appropriate matching response can be located even where an exact match is unavailable. For instance, a concept search for HP Laser Jet printer may fail to locate the printer as it is not available, but would return a list of services listed under the printer concept and the user would then consume the most appropriate alternative.

Our implementation took in to consideration device heterogeneity. Devices running the Android OS are highly heterogeneous in many respects, such as location, OS versions, device capabilities and so on. Heterogeneity should be taken into consideration when implementing a service discovery mechanism for ad hoc networks.

We have described Services on a service tree, using service concepts. Service advertisement and matching are the carried out at the service tree level. This provides a consistent identification of services across all nodes in a network. The implementation does not have a significant memory footprint and will therefore run efficiently in the background, without consuming a lot of processing power and bandwidth.

6.2.2 Inter-Node Communication /Messaging

Figure 22 demonstrates the flow of messages in the application. Messages are forwarded from node to node both actively (on a need to need basis) and passively, such as service advertisements. Node to node messages form the basis of the implementation. Devices rely on service advertisements to learn about services in the network.

Service discovery implementations may apply either of two strategies of internode communication: active or passive mode. Active mode enables node to correctly locate available services at the instant when they are needed, especially in highly dynamic MANETs. In passive mode, nodes would regularly advertise their services to their neighboring nodes which would then cache them. As observed in Figure 24, there is a progressive increase in message loss as messaging frequency increases. Additionally, nodes consume power as they send messages. As demonstrated in Figure 25. A hybrid model is an effective strategy to mitigate the challenges of either approach. Service advertisement messages are temporarily cached on devices. When a device needs to use a service, it first searches its cache before sending a service request in the network. When a device receives a service request, it checks its service cache and only propagates it if it does not find a match. This reduces message traffic.

Optimizing broadcasts to reduce message traffic is key in Android OS networks. Williams B and Camp C. (2002) compared several broadcast mechanisms. They discuss the merits of various approaches in broadcasts. C. Cho and D. Lee,(2002) also compared various multicast protocols. We agree with the findings of these studies that the best strategy for developing a routing mechanism is one that has contextual knowledge such as neighbor knowledge and uses the same when making routing decisions. We implemented a routing mechanism that features 2-hop neighbor knowledge.

As mentioned earlier, android devices are resource constrained. From the prototype, we observed that there was a progressive increase in memory use over time. Effective implementations for mobile ad-hoc MANET are required to have a low memory footprint. As a result, it is important to implement architectures that are not fully directory based. In our implementation, we implement a hybrid architecture. Messages are temporarily cached and expired once a set of conditions have been met. These are

- Expiry based on a time to live *ttl* parameter
- Upon arrival of a similar message as uniquely identified by a message id field.

We mentioned earlier the challenges of Mobile Ad-hoc networks as among others power and bandwidth availability. In any broadcast mechanisms, there is a tradeoff between reliability and message overhead. Increasing reliability in order to guarantee delivery generally means re-

sending a greater number of messages. We have demonstrated before in Figure 25 that there is an overall increase in power consumption as the messaging frequency increases. To mitigate this, we developed a routing mechanism that is optimized to reduce network traffic and at the same time conserve bandwidth. The implementation uses 2-hop neighbor knowledge to make routing decisions. The tests have demonstrated the improved efficiency.

6.3.1 Service Invocation

Once a node has identified a service, it will send a service request message to the host node. The host node responds by sending a service description of the requested service. Service invocation relies on the MANET's underlying network addressing layer. Figure 27 Illustrates this concept.

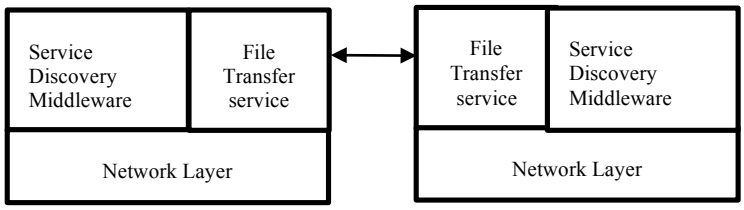

Figure 27 Service Invocation

The invocation mechanism we have developed requires files to be transferred across devices. The intermittency of the internode connections means that packets cannot be reliably streamed. Transferring large files such as audio files has the overall effect of affecting the performance as the file has to be completed delivered before it can be processed. Unreliable connectivity can be solved by implementing a message acknowledgement mechanism.

6.4 Further Work

There is room for improvement in the implementation of MANETs for Android Devices. As mentioned earlier in this document, the Android OS does not support the ad-hoc networking mode out of the box. Android MANETs implemented at application level are not stable.

Nodes acquire information about their 2-hop neighbors and use this knowledge for selective message forwarding. In dense networks, devices would consume a lot of processing power. As an improvement, we propose the implementation of a node scoring mechanism, by which we could determine reliable nodes and thereby form an overlay network. Nodes that are not well scored will have a lower priority and could be left out in building the neighbor cache. The criteria for scoring could be the broadcast range of devices, message delivery success history of the node or device preferences.

It is necessary to consider that device users may want to control the extent to which their devices participate in ad-hoc networks. Power is usually limited in wireless devices such as MANET nodes. Security is also an important consideration. It is therefore necessary to give users the opportunity to determine whether they can participate in a MANET.

Security concerns are paramount in the implementation of MANETs. Since our framework handles inter-node communication, it will need to incorporate security features. Security aspects

in MANETS include network availability, authorization and key management, data integrity and non-repudiation (Rai and Singh, 2010).

There is also need to implement incentive management in the framework so as to encourage node co-operation in the MANET. In an Android MANET network, a user may choose not co-operate such as by switching off their WIFI connection, for various reasons, such as power management. This leads to a loss of network connectivity. Various techniques have been proposed to encourage node cooperation Al-Karaki and Kamal (2007). In virtual payment schemes a node is compensated by some form of payment for every message that it forwards. The payment is then redeemed for preferential treatment on the network. In reputation based schemes a nodes reputation is established from its handling of messages and the reputation could be used in selective node forwarding of messages. Implementation of an incentive mechanism is however out of the scope of this work.

We observed that when multi-hop networking is implemented at Kernel level in Android devices, the networks formed are more resilient as compared to implementation at the application layer. The Android OS does not support the multi hop networking mode out of the box. In order to implement stable multi-hop networking at the Kernel level, one is required to build an Operating System patch as Android OS does not natively support ad hoc networking. This usually requires rooting of the Android device, an operation that could permanently damage the device.

7 REFERENCES

Abou El Saoud, M., Kunz, T. and Mahmoud, S. (2006). BENCHManet: An Evaluation Framework for Service Discovery Protocols in MANET. 3, pp.860--865.

Agarwal, S., Handschuh , S. and Staab, S. (2004). Annotation, composition and invocation of Semantic Web Services. Web Semantics, 2.

Aitha, N. and Srinadas, R. (2009). A Strategy to Reduce the Control Packet Load of AODV Using Weighted Rough Set Model for MANET. *A Strategy to Reduce the Control Packet Load of AODV Using Weighted Rough Set Model for MANET*, 18(1), pp.108-116

Al-Karaki, J. and Kamal, A. (2007). Stimulating Node Cooperation in Mobile Ad hoc Networks. *Wireless Pers Commun*, 44(2), pp.219-239.

Androidcentral.com, (2016). *home automation | Android Central*. [online] Available at: http://www.androidcentral.com/tag/home-automation [Accessed 16 Jan. 2016].

Azzedine Boukerche, Daniel Cˆ amara, Antonio A.F. Loureiro, and Carlos M.S. Figueiredo. (2009). Algorithms for Mobile Adhoc Networks. In: Azzedine Boukerche *Algorithms and protocols for wireless, mobile ad hoc networks* . Hoboken, New Jersey: John Wiley & Sons Inc. pp.1-21

Basagni, S., Conti, M., Giordano, S. and Stojmenović, I. (2004). *Mobile ad hoc networking*. 1st ed. Piscataway, NJ: IEEE Press.

C. Cho and D. Lee, "Survey of Service Discovery Architectures for Mobile Adhoc Networks," term paper, Mobile Computing, CEN 5531, Dept. Computer and Information Science and Eng.,Univ. Florida, Fall, 2005

Chakraborty, D. et al. , "DReggie: A Smart Service Discovery Technique for E-Commerce Applications," Proc. Workshop in conjunction with 20th Symp. Reliable Distributed Systems, Oct. 2001.

Chakraborty, D. et al., "GSD: A Novel Group-Based Service Discovery Protocol for Manets," Proc. 4th IEEE Conf. Mobile and Wireless Communications Networks (MWCN 02), IEEE Press,2002, pp. 140–144.

Chakraborty, D., Joshi, A., Yesha, Y. and Finin, T. (2006). Toward distributed service discovery in pervasive computing environments. *Mobile Computing, IEEE Transactions on*, 5(2), pp.97--112.

Charif, Yasmine, and Nicolas Sabouret (2006). ‘Approaches to Semantic Web Services:An Overview and Comparisons'. *Electronic Notes in Theoretical Computer Science* 146.1 pp 33-41.

Cheng, L.(2002) *Service Advertisement and Discovery in Mobile Ad hoc Networks*

CNET, (2014). *Android stays 'unbeatable' in smartphone market -- for now - CNET*. [online] Available at: http://www.cnet.com/news/android-stays-unbeatable-in-smartphone-market-for-now/ [Accessed 16 Nov. 2015].

Corson, S., and Macker, J. Mobile Ad-hocNetworking (MANET): Routing Protocol Performance Issues and Evaluation Considerations. RFC 2501, IETF, Jan. 1999.

Darpa.mil, (2013). 2013/12/12 Radio Gateway Connects U.S. and Allied Troops to a Common Mobile Network. [online] Available at: http://www.darpa.mil/NewsEvents/Releases/2013/12/12.aspx [Accessed 9 Jul. 2014].

David, M. et al, OWL-S: Semantic Markup for Web Service, Available from http://www.w3.org/Submission/OWL-S/

Econsultancy, (2014). *65% of global smartphone owners use Android OS: stats*. [online] Available at: https://econsultancy.com/blog/64376-65-of-global-smartphone-owners-use-android-os-stats#i.x1oa1q99adufyz [Accessed 15 Jul. 2014].

Fcw.com, (2012). *DARPA commissioning ad-hoc smart phone network -- FCW*. [online] Available at: http://fcw.com/Articles/2012/01/20/DARPA-SAIC-smart-phone-mobile-ad-hoc-network.aspx [Accessed 15 Jun. 2014].

Garc'ia L'opez, P., Gracia Tinedo, R. and Ban'us Alsina, J. (2010). Moving routing protocols to the user space in MANET middleware. *Journal of Network and Computer Applications*, 33(5), pp.588—6

Gardner-Stephen, P. (2011). *The Serval Project: Practical Wireless Ad-Hoc Mobile Telecommunications*. [online] developer.servalproject.org. Available at: http://developer.servalproject.org/site/docs/2011/Serval_Introduction.html [Accessed 20 Jun. 2014]

Gaudin, S. (2016). *At Google I/O, the Internet of Things gets a new OS*. [online] Computerworld. Available at: http://www.computerworld.com/article/2927408/android/at-google-io-the-internet-of-things-gets-a-new-os.html [Accessed 21 Nov. 2015]

Google Developers, (2016). *Brillo | Google Developers*. [online] Available at: https://developers.google.com/brillo/?hl=en [Accessed 21 Nov. 2015].

Goyal, P., Parmar, V. and Rishi, R. (2011). Manet: Vulnerabilities, challenges, attacks, application. *IJCEM International Journal of Computational Engineering \& Management*, 11(2011), pp.32--37.

Haas, et al. (2006). 'Gossip-Based Ad Hoc Routing'. *IEEE/ACM Transactions on Networking* 14.3 479-491

Helal, S. et al., "Konark: A Service Discovery and Delivery Protocol for Ad Hoc Networks," Wireless Comm. and Networking, vol. 3, 2003, pp. 2107–2113.

Hermann, R., Husemann, D., Moser, M., Nidd, M., Rohner, C. and Schade, A. (2001). DEAPspace – Transient ad hoc networking of pervasive devices. *Computer Networks*, 35(4), pp.411-428.

Hoebeke, J., Moerman, I., Dhoedt, B. and Demeester, P. (2004). An overview of mobile ad hoc networks: Applications and challenges. *Journal-Communications Network*, 3(3), pp.60—66.

Kirsch, A. and Mitzenmacher, M. (2008). Less hashing, same performance: Building a better Bloom filter. *Random Struct. Alg.*, 33(2), pp.187-218

Lego, K. and Sutradhar, D. (2011). Comparative Study of Adhoc Routing Protocol AODV, DSR and DSDV in Mobile Adhoc NETwork 1. *Citeseer*.

Lenders, V., May, M. and Plattner, B.Service Discovery in Mobile Ad Hoc Networks: A Field Theoretic Approach,Proc. 6th IEEE Int'l Symp. World of Wireless Mobile and Multimedia Networks (WoWMoM 05), IEEE Press, vol.1, 2005, pp. 120–130.

Mcllraith et al ,2001:. 'Semantic Web Services'. *IEEE* 46-54.

Manvi, S. and Kakkasageri, M. (2008). Multicast routing in mobile ad hoc networks by using a multiagent system. *Information Sciences*, 178(6), pp.1611—1628

Mathias, K. (2015). Semantic Web Service Description'. *CASCOM: Intelligent Service Coordination In The Semantic Web Whitestein Series In Software Agent Technologies And Autonomic Computing 2008, Pp 31-5*. Monique Calisti et al. 1st ed. 2008. 31-57. Web. 9 Aug. 2015.

Mian, A., Baldoni, R. and Beraldi, R. (2009). A survey of service discovery protocols in multihop mobile ad hoc networks. *Pervasive Computing, IEEE*, 8(1), pp.66--74.

Militaryaerospace.com, (2014). *Army demonstration of commercial cell phone technology on the battlefield relies on Raytheon technology*. [online] Available at: http://www.militaryaerospace.com/articles/print/volume-22/issue-12/news/news/army-demonstration-of-commercial-cell-phone-technology-on-the-battlefield-relies-on-raytheon-technology.html [Accessed 17 Jun. 2014].

Minhas, Q., Mahmood, H. and Malik, H. (2011). The Role of Ad Hoc Networks in Mobile Telecommunication. In:Dr J. Maicas, ed., *Recent Developments in Mobile Communications - A Multidisciplinary Approach*, 1st ed. pp.179-201.

Mitre.org, (2014). *The MITRE Corporation*. [online] Available at: http://www.mitre.org/ [Accessed 17 Jun. 2014].

Mlinarsky, Fanny. *The Challenges And Importance Of Testing Mesh Networks Prior To Deployment*. 2006. Web. 3 Feb. 2015.

Mueller, H. (2014). *android-wifi-tether - Wireless Tether for Root Users - Google Project Hosting*. [online] Code.google.com. Available at: http://code.google.com/p/android-wifi-tether/ [Accessed 26 June. 2014].

Naga Satish, G., Raghavendran, C., JoseMoses, G., Suresh Varma, P. and Murthy, 5. (2011). Service Discovery in ADHOC Networks. *International Journal of Advanced Research in Computer Science and Software Engineering*, [online] 2(11), pp.283-289. Available at: http://www.ijarcsse.com/docs/papers/11_November2012/Volume_2_issue_11_November2012/V2I11-0183.pdf [Accessed 25 Jun. 2014].

Ni, S., Tseng, Y., Chen, Y., Sheu, J.: The Broadcast Storm Problem in a Mobile Ad Hoc Networks. In: The Broadcast Storm Problem in a Mobile Ad Hoc Networks, pp. 151–162. IEEE Computer Society Press, Los Alamitos (1999)

Obaid, A., Khir, A., Mili, H. and Laforest, L. (2007). A Routing Based Service Discovery Protocol for Ad hoc Networks. p.108.

Open-mesh.org, (2014). *WikiStart - Open-mesh - Open Mesh*. [online] Available at: http://www.open-mesh.org/projects/open-mesh/wiki [Accessed 17 June. 2014].

Patterson, S. (2014). *Android phones are connecting without carrier networks*. [online] Network World. Available at: http://www.networkworld.com/article/2224025/smartphones/android-phones-are-connecting-without-carrier-networks.html [Accessed 9 Jun. 2014].

Penttinen, A. (2002). Research on ad hoc networking: Current activity and future directions. Networking Laboratory, Helsinki University of Technology, Finland. See also http://citeseer. nj. necm. com/533517. html.

Plus, G. (2013). *Samsung and HTC phones to soon support ZigBee?*. [online] Android Central. Available at: http://www.androidcentral.com/samsung-and-htc-phones-soon-support-zigbee [Accessed 14 Nov. 2015].

Qurratul-Ain Minhas, Hasan Mahmood and Hafiz Malik (2011). The Role of Ad Hoc Networks in Mobile Telecommunication, Recent Developments in Mobile Communications - A Multidisciplinary Approach, Dr Juan P. Maícas (Ed.), ISBN: 978-953-307-910-3, InTech, Available from: http://www.intechopen.com/books/recent-developments-in-mobile-communications-a-multidisciplinary-approach/the-role-of-ad-hoc-networks-in-mobile-telecommunication.

Radhamani, G., Vanitha, K. and Sud hamathi, C. (2011). Cross Layer Issues in Service Discovery on Pervasive Computin g: An Approach. *International Journal of Computer Applications*, 17(8,March).

Rai, P. and Singh, S. (2010). A Review of 'MANET's Security Aspects and Challenges'. *ICJA*, (Special Issue on "Mobile Ad-hoc Networks" MANETs).

Ranganathan, A., Al-Muhtadi, J., Biehl, J., Ziebart, B., Campbell, R. and Bailey, B. (n.d.). Towards a Pervasive Computing Benchmark. *Third IEEE International Conference on Pervasive Computing and Communications Workshops*.

Sarkar, K., Basavaraju, T. and Puttamadappa, C. (2008). *Ad hoc mobile wireless networks*. 1st ed. New York: Auerbach Publications.

Sharma, S., Dhamija, A. and Rawal, V. (2012). Current issues with Ad-hoc network \& proposed solutions. *International Journal of Applied Engineering Research*, 7(11), p.2012.

Srirama, S., Paniagua, C. and Liivi, J. (2013). Mobile Web Service Provisioning and Discovery in Android Days. pp.15—22.

Tetcos.com,. 'Netsim - Network Simulator'. N.p., 2015. Web. 29 May 2015.

Thomas, J., Robble, J. and Modly, N. (2012). Off grid communications with android -Meshing the Mobile world. pp.401—405.

Toh, C-K. Ad Hoc Mobile Wireless Networks: Protocols and Systems.Prentice Hall, 2002

Ververidis, C. and Polyzos, G. (2008). Service discovery for mobile ad hoc networks: a survey of issues and techniques. *Communications Surveys \& Tutorials, IEEE*, 10(3), pp.30—45.

Village Telco, (2008). *Mesh Potato*. [online] Available at: http://villagetelco.org/mesh-potato/ [Accessed 25 Jun. 2014].

Wireshark.org,. 'Wireshark · Go Deep.'. N.p., 2015. Web. 29 July 2015.

Wu, T. and Kuo, G. (2006). An analytical model for centralized service discovery architecture in wireless networks. pp.1—5

Xu, B. and Wolfson, O. (n.d.). *Data Management in Mobile Peer-to-Peer Networks*. 1st ed. [ebook] Available at: https://www.cs.umd.edu/class/fall2009/cmsc828r/PAPERS/Wolfson_dbisp2p04-keynote.pdf [Accessed 24 Jun. 2014].

Zakarya, M. and ur Rahman, I. (2013). A Short Overview of Service Discovery Protocols for MANETS. *VAWKUM Transaction on Computer Sciences*, 1(2).

Zhang, L., Tiwana, B., Qian, Z., Wang, Z., Dick, R., Mao, Z. and Yang, L. (2010). Accurate online power estimation and automatic battery behavior based power model generation for smartphones. *Proceedings of the eighth IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis - CODES/ISSS '10*

Zhuang, T., Baskett, P. and Shang, Y. (2013). Managing Ad Hoc Networks of Smartphones. *International Journal of Information \& Education Technology*, 3(5).

Ziyang.eecs.umich.edu, (2016). *PowerTutor*. [online] Available at: http://ziyang.eecs.umich.edu/projects/powertutor/ [Accessed 11 Jan. 2016].

3wdroid.org, (2014). *Span (Smart Phone Ad-Hoc Networks)*. [online] Available at: http://www.3wdroid.org/span [Accessed 23 Jun. 2014].