International Journal of Computer Applications (0975 – 8887)
Volume 151 – No.2, October 2016

Service Discovery in Mobile Ad-Hoc Networks of Android OS Devices

Eugene K. Kamau
School of Computing and Informatics
University of Nairobi

Andrew M. Kahonge
School of Computing and Informatics,
University of Nairobi

ABSTRACT

Smartphone ad-hoc networks have been deployed for communication where infrastructure based communications are not available (or desirable). Android OS ad-hoc networks can be employed in development of collaborative applications such as entertainment systems and mobile gaming. In addition, they facilitate ubiquitous computing in areas such as home applications where wireless sensors and actuators are embedded in consumer electronics.

This paper presents an implementation of Service Discovery middleware for Android Mobile ad-hoc networks. The implementation overlays a Mobile Ad-hoc Network implementation. It further describes a routing implementation that is based on the Ad-hoc On-Demand Distance Vector protocol, with modifications to reduce control message overhead. 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 is also presented. The implementation was tested by simulating the routing implementation. An Android based prototype was developed and tested on a range of devices.

General Terms

Android OS Mobile ad-hoc Networking, Smartphone Mobile ad hoc Networks, MANET

Keywords

Mobile ad hoc networking(MANET), Service Discovery in MANETS, Smartphones, Android OS

1. INTRODUCTION

The Android OS has proven to be a good platform for driving ubiquitous computing. Multi-hop networks facilitate communication between devices where infrastructure based connections are not available or desirable.

Mobile ad-hoc networks are created to facilitate the exchange of data and sharing of resources and service and therefore the importance of service discovery cannot be overstated.

Research in Android OS based devices ad hoc networking, has mainly focused on addressing routing issues. There are currently few notable attempts to address service discovery even in major 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. As the interest in developing ad hoc networks on the smartphones and more so Android OS continues to rise, there is need to address the enabling technologies for these MANET platforms.

This paper presents a design and implementation of a Service Discovery middleware that runs on the Android OS. The middleware implements a message routing, service description, advertising and matching components. Service descriptions are based on concepts. Matching is also done using service concepts. Also discussed is an ontology for describing services on android based devices and a simple algorithm for matching services.

The routing implementation developed is based on the Ad hoc on-demand Distance Vector (AODV) routing protocol with enhancements for node neighbor knowledge, so as to reduce control message traffic and therefore power and bandwidth consumption.

The paper is organized as follows: An initial discussion on previous studies in implementing service discovery in Smart phone MANETs, and how this implementation complements the work. The subsequent sections focus on the implementation. A discussion on the implementation of the routing layer, a discussion on semantic service discovery then follows. It concludes with a discussion of the test results.

2. RELATED WORK

2.1 Mobile ad-hoc Network Implementations

There are several notable implementations of mobile ad-hoc networks for android devices. Ad-hoc networks in Android OS devices may be implemented either at application level or at the kernel level.

The Serval project [1] is an android based mobile telephony platform. The Serval project implements the BATMAN protocol, a Linux routing protocol.

MAINGATE [2] is a mobile ad hoc network gateway developed and tested for the USA military.

SPAN 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 [3].

Zhuang and Basket discuss an implementation for android MANET that does not require kernel modification. The MANET middle ware is implemented at application level [4].

2.2 Semantic Service Discovery

The importance of semantic service descriptions is that Service data can be captured along with specifications of its properties and capabilities, the interface for its execution, preconditions and effects of its use [5]. Helal S et al propose a service discovery implementation known as Konark[6], where services are described in an ontology.

2.3 Efficient Message routing

Aitha et al propose a modification to the AODV protocol where the implementation uses a weighted roughest model to control the route request per packet [7].

3. METHODOLOGY

The implementation is a standard mobile ad hoc network implementation that provides delivery and routing of data between nodes, subject to the spontaneous introduction and removal of any node at any time, on Android OS.

Each component of the implementation 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

3.1 System Design Overview

The service discovery functionality on top of an underlying MANET network layer.

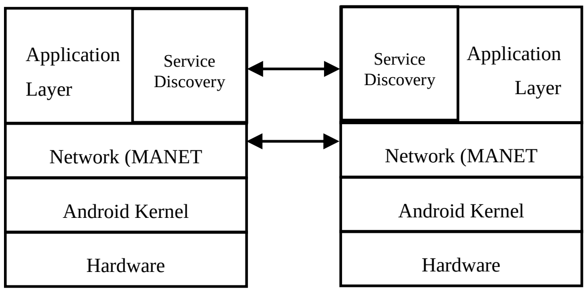

Figure 1 MANET Layer Hierarchy

An overview of the implementation is presented in the figure below.

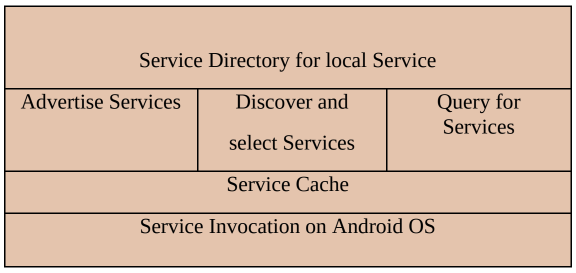
Service Directory for local Service
Advertise Services
Discover and select Services
Query for Services
Service Cache
Service Invocation on Android OS

Figure 2 Service Discovery Modules

3.2 Service Description Implementation

The main consideration in the Service Description development was to develop a service description that has a reasonable degree of heterogeneity and sufficient data to allow nodes to support mapping across the multiple ontologies. The service description is a hierarchical layout of service concepts, where the description will be more specialized lower the hierarchy and generic at the top.

3.2.1 Service tree

Services are advertised and discovered at any level of a service tree. The highest nodes serve as parents to more specialized services further down the tree [6]. The tree 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 consumption.
- 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.
- Facilitate probability matching

In Figure 3 below, 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.

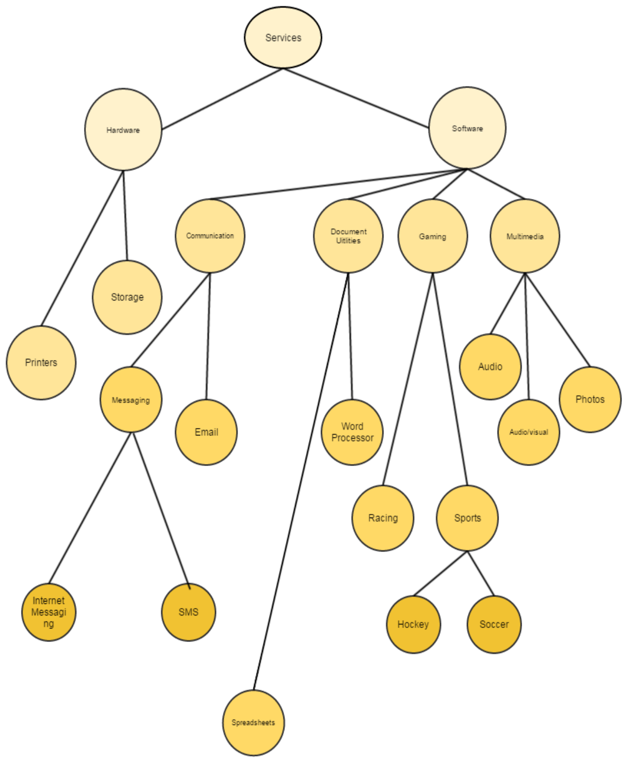

Figure 3 Android Service Tree

3.2.2 Service Description Document

The Service Description is based on the OWL-S standard [8]. Key elements and functions of the service description are the service profile, which consists of functional and non-functional requirements of the service: inputs, preconditions, expectations and outputs and service functions which contains the service composition and processing logic such as process flows.

3.3 Service Advertisement Implementation

The desired properties of the service advertisements were minimal message size, minimal communication overhead, support for the diverse capabilities of Android devices and a minimal memory footprint.

3.3.1 Service advertisement message

The service advertisement message contains information about services and is broadcast across the network from source nodes. Service advertisements follow a passive push, in that nodes broadcast service advertisement messages in the network without prompting.

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 and device version. The hop count field is incremented at each node as the message is forwarded and it 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. 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.

Upon receiving service advertisement messages nodes add them to a cache. 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 may then start the service invocation process.

3.4 Service Directory

Messages are stored in an SQLite database and a Bloom Filter [9] based memory cache. The memory cache optimizes the performance of the memory for records that are frequently accessed. Records will be initially persisted in the memory cache before being pushed to the database. The structure is illustrated below

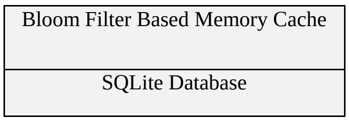

Figure 4 Service Directory Structure

Nodes cache service advertisements received for a specified period of time. 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 a service named music player
Check cache
        Is service name or concept in cache?
        If true read advertisement from database
Else
        Broadcast discover message in the network.
```

3.5 Broadcast Mechanism Implementation

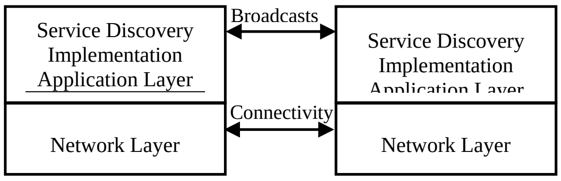
Service Discovery Implementation Application Layer
Broadcasts
Service Discovery Implementation Application Layer
Network Layer
Connectivity
Network Layer

Figure 5 Network broadcast overview

3.5.1 *Format of a broadcast message*

- Broadcast id- This is the unique identifier for each broadcast.
- Hop count Indicates the number of nodes a message will be allowed to transverse. The count is decremented at every node as the message is propagated across the network.
- Time to live - This specifies the validity of a message.
- Time generated- The time when the broadcast was generated.
- Source address – The address of the last node that sent the message.
- Destination Address (optional field) – The address of the node where the message is headed. It will be populated for direct message routing and will be set empty for broadcasts.
- Payload- The service specific messages such as service advertisements, service discovery messages and request replies.

It is necessary to reduce the longevity of broadcast messages as well as the range in order to avoid stale messages, avoid continuous broadcasts which would cause a broadcast storm and consume devices resources.

Broadcast Algorithm

A broadcast algorithm that extends Ad-hoc on Demand Distance Vector (AODV) routing protocol to support neighbor aware routing was implemented. The purpose of introducing neighbor aware routing was to reduce control packets traffic.

3.6 Broadcast Algorithm Implementation

The broadcast algorithm employs selective forwarding of Route Request (RREQ) control messages. This reduces the number of nodes required to forward a Route Request Message.

HELLO messages are exchanged by 1 hop neighbors every time a node joins the network and whenever a node moves. The framework uses the Android OS Wi-Fi p2p API to determine a node's neighbors.

3.6.1 *HELLO message*

The HELLO message sent 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. They also use it to determine 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.

3.6.2 *RREQ propagation*

In stable networks implementing AODV, Route Request messages are the highest proportion of messages exchanged.

Using the 2-hop neighbor information, nodes 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 forward messages.

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

3.6.3 Pseudo code for message propagation

A typical message transmission goes through the below stages:

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.

In highly dense networks, nodes would have a large number of neighbors. Neighbor discovery using the algorithm and management of neighbor information would stretch processing and memory resources on devices. It is proposed to limit the size of neighbor knowledge records that a node can have. The implication of this is that the neighbor discovery will be effective only up to a limit determined by the device's resource capabilities.

3.7 Semantic processing of messages

Semantic message processing revolves around service concepts that are derived from the Service tree. They comprise the tree nodes and service properties. Service matching will use the services semantic description.

Consider an ad-hoc network N={n_1,n_2 …n_m} as a set of mobiles nodes where $n_i \in N$ of size m. 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.

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

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

3.7.1 Service Advertisement processing

The idea is to cache a message's concepts if the broadcast has not been encountered before. Assume a node k receives a message m originating from p. Message contains concepts Cn={c0,c1…..cn}.

```
for m.all concepts do
    if m.time_to_live >0
        $V_0^c = V_k \cup \{c\}$ *
        m.cache $\leftarrow$ p.address
    endif
end for
```

* Add node address to the cache against each concept encountered

3.7.2 Service Matching

Concepts match if their respective names are syntactically equal and each of their properties match in scope and name. The matching mechanism applies a weighting mechanism to find the similarity of concepts to one another where an exact match cannot be found.

From the Service tree implementation of Service representation, one can see that services will be more closely related further down the tree. At the very top of the tree the categories are highly generic so there would be no relationship.

Table 1 Concept matching

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

Representing 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 $C_1C_2 \Leftrightarrow C_2C_1$.

3.7.3 Service Discovery Implementation

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

Service discovery follow 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.

3.7.4 Format of a service discovery reply message

If a service discovery request is successfully matched, a reply message will be generated, comprising the matching score, the weighted score generated by the matching mechanism. Other fields are the original broadcast id, the service host address, the source address which 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 the scope of this work.

Additionally, a service discovery reply message will also contain a destination address, a time to live field that determines how long the response should be maintained in a cache and the node distance in terms of hops.

3.8 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 will rely on the MANET's underlying network addressing layer. A typical service invocation would be a http based file transfer service to a node hosting a service as shown on Figure 6.

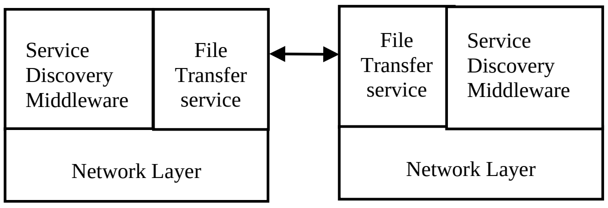

Figure 6 File transfer service for service invocation from Node A to Node B

4. GENERAL PROCEDURE

Neighbor Knowledge Management is implemented using Android Wi-Fi Direct framework, available since 4.0 (API level 14). It is done by an Android intent service that executes at a fixed interval, sending out hello packets from each node. The information is then persisted in the device. 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. The service tree path is implemented as a character separated list of service concepts. Concepts represent nodes in the service tree. An example of a service path would be: Services, Software, entertainment, audio, mp3. A background task will execute service advertisements at fixed intervals of time.

4.1 Process flow for advertisements

A background periodically runs to read the broadcasts from the directory and broadcast advertisements. Service advertisements are created by reading service descriptions to extract required fields. Service Advertisements received are cached. This is illustrated in Figure 7.

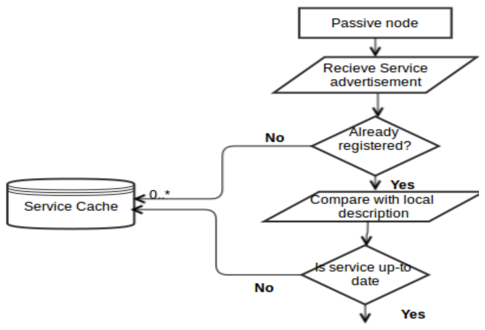

Figure 7 Process flow for Service Advertisement

4.2 Service Tree traversing

Locating a concept theoretically involves executing a depth first search on the service tree.

4.2.1 Example of Service Tree Traversing

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

a. The path has a cardinality of 5.

b. If a successful match is returned for a search of concept 'mp3' then the match score is 5x3=15.

c. If a successful match is returned for a search of concept 'audio' then the match score is 5x2=10.

d. If a successful match is returned for a search of concept 'entertainment' then the match score is 5x1=5.

e. If no specific concept matches such that the lowest concept is at the generic software concept, the score is 5x0=0.

The highest point on the service tree above which no further matches can be done is defined.

4.3 Overview of Message Flow

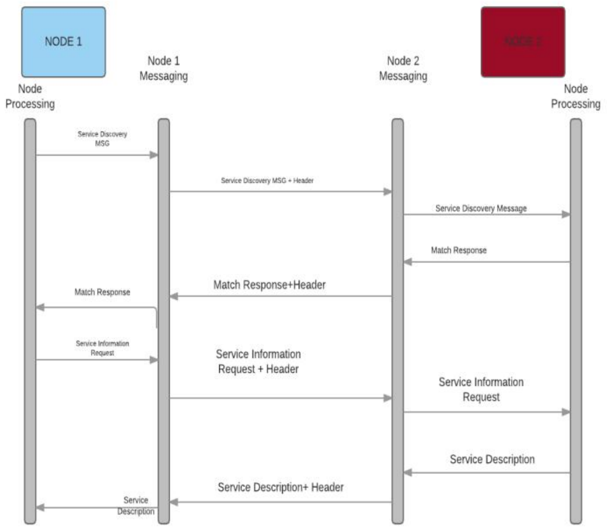

Figure 8 Message flow in the implementation

5. RESULTS

5.1 Routing Performance

The expected result of selective message forwarding is to reduce the control message overhead. Route Request (RREQ) control messages were used to access the performance of the routing implementation. The efficiency of this approach over plain Ad-hoc On-demand Distance Vector(AODV) can be assessed using an approach for comparing routing protocols that compares control message packets count [10], 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.

A simulation was setup to establish routing efficiency. Results obtained from the simulation are shown in the chart of Figure 10. Fixed parameters were

- 1 message every 5 seconds
- Time To Live 5 seconds
- Simulated hello loss of 2%

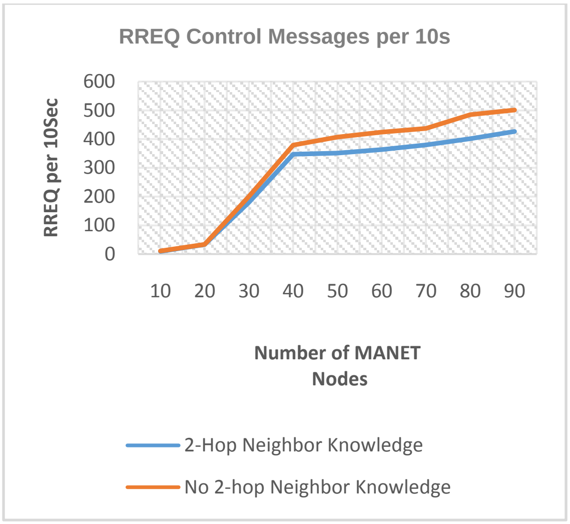

Figure 9 Chart showing simulation results

5.2 Deductions on routing

As the number of nodes increases, nodes consume more processing power in computing a routing path. There is therefore a tradeoff between routing performance and processing power.

5.3 Android Prototype Evaluation

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:

a. **Service discovery latency** This is the time needed to resolve a service request to a valid service binding and contacting the service host successfully.
b. **Service availability** The ratio of successful service bindings to the total number of requests obtained.

Messages exchanged between the nodes consist of network connectivity messages (acknowledgments, hello messages) and service discovery messages discussed in a previous section. The count of messages transmitted over a period of time was used measure the performance of the implementation.

5.3.1 Experiment 1

The hops between the devices were then varied and the step repeated. 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.

Messages are lost when the device is overwhelmed to the point that it is not able to process and forward all messages it receives. Better performance of the framework is therefore achieved at lower message rates. The message response rate would also be affected by the number of network hops.

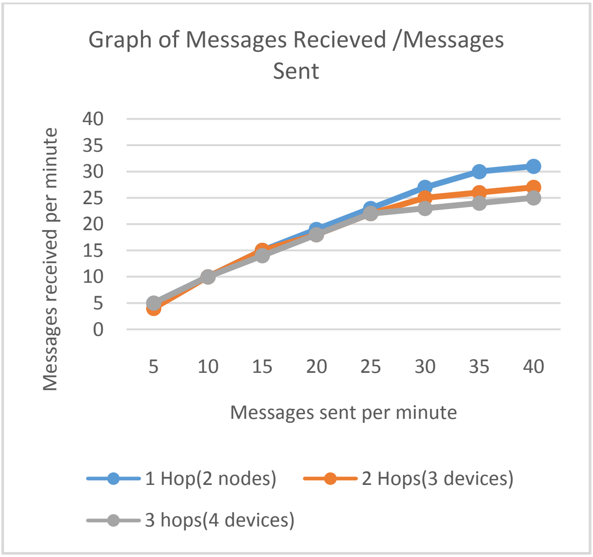

Figure 10 Messages success rate

5.3.2 Experiment 2 Power Consumption

Power consumption of the application on an actual device increased 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[11], an android power monitoring application. This application was used to monitor the android implementation's power consumption over a duration of five minutes. The rate of messages was then gradually increased. The power output recorded is a summation of application CPU processing power and the power consumed by the display.

The important observation here (in Figure 12) is the overall increase in power consumption as the messaging frequency increases, hence the need to keep message frequency as low as possible.

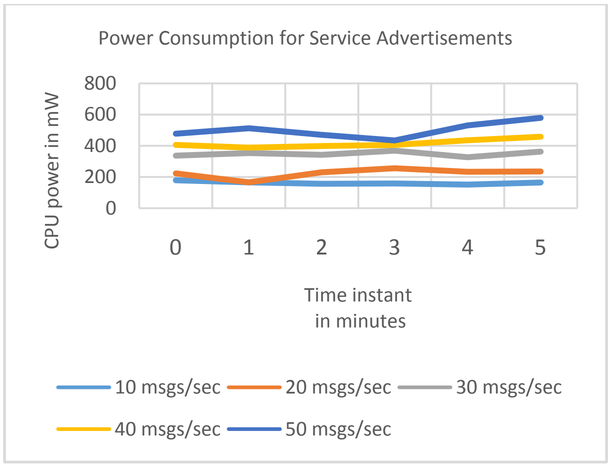

Figure 11 Power consumption

5.3.3 *Experiment 3 Service Matching*

Interaction between the service discovery middleware and services

Audio Player Service

The aim of this experiment 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

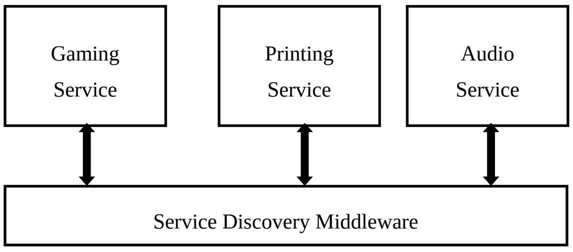

Figure 12 Interaction between the implementation and services

A service request was sent out from the requesting device. Devices that received the request responded with a service match response. The requestor then selected the highest scored match. Different service profiles were created for each of the test devices.

Table 2 Sample Service Profiles

Description	**Device 1**	**Device 2**	**Device 3**
Service Name	Jet audio	vlc	mediaone
Service Path	services,software, entertainment,,mediaplayer, audio,mp3,jetaudio	services,software,entertainment,, mediaplayer,audio,mp3,vlc	services,software, entertainment, ,mediaplayer,audio,mp3, mediaone
Description		My HIFI	AndroidWearable
Actor Name	EugenePhone		
Node Address	To be set by device	To be set by device	To be set by device
Inputs	Audio file	Audio file	Audio file

A service request was broadcast, to locate VLC mp3 player in the network. The responses received are tabulated below:

Table 3 Results from matching

Device	Matching Score	Device address
Device 1	30	**EugenePhone**
Device 2	42	**MY HIFI**
Device 3	30	**AndroidWearable**

5.4 Discussion

The hop count field 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.

The efficiency of the routing implementation is more pronounced as the number of nodes increases. It is however worth noting that as the number of nodes increases, nodes consume more processing power in computing a routing path.

5.4.1 *Benchmarking*

Ranganathan et al proposed a standard benchmark for the assessment and evaluation of pervasive computing environments, that is applicable to our work [12]. An evaluation of the 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. The shared ontology facilitates a common understanding in the environment and consistence in interactions between nodes.

b. Context-Sensitivity
Context aware search enable devices to locate the most appropriate and relevant services. This is enhanced by semantic search and matching.

c. Semantics
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
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. The tests have demonstrated 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.

6. CONCLUSION

This research has accomplished the following objectives:

- Developed a service registration syntax for managing services in Android OS based MANETs. We also described a tree based approach for grouping services.
- Developed functionality for routing service advertisement and service discovery messages that is bandwidth and power efficient.

- The research has also demonstrated a suitable implementation for service invocation for service discovery platforms in Android OS MANETs

The android prototype demonstrated the feasibility and effectiveness of a middleware for service discovery in Android OS multi-hop networks.

It also demonstrated that by employing semantic techniques, it is possible to select the most relevant and appropriate services in an Android MANET.

To mitigate power and bandwidth challenges in mobile ad-hoc networks, the framework implementation included a routing mechanism that is optimized to reduce network traffic and thus conserve bandwidth. The tests have demonstrated the improved efficiency.

The implementation does not have a significant memory footprint and will therefore run efficiently in the background, without consuming a lot of processing power.

7. FURTHER WORK

It is necessary to implement a scoring mechanism, by which nodes could determine reliable nodes and thereby form an overlay network for message propagation. The criteria for scoring could be the broadcast range of devices, message delivery success history of the node or device preferences.

Since the framework handles inter-node communication, it will need to incorporate security features. Security concerns in MANETS include network availability, authorization and key management, data integrity and non-repudiation.

There is also need to implement incentive management in the framework. In MANET networks, a user may choose not cooperate such as by exiting the network. This could lead to a loss of network connectivity. Also users may choose to selectively forward messages, in the process crippling the network. It is therefore necessary to incentivize cooperating nodes

8. ACKNOWLEDGEMENTS

The authors wish to thank their colleagues and the School of Computing and informatics at the University of Nairobi for facilitating this research.

9. REFERENCES

[1] 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]

[2] 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]

[3] 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]

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

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

[6] 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.

[7] 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

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

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

[10] Sucec, J. and Marci, I. (2014). An Efficient Distributed Network-Wide Broadcast Algorithm for Mobile Ad Hoc Networks. CAIP Center,Rutgers.

[11] 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*

[12] 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*.

[13] 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.

[14] 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].

[15] 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)