A Protocol for Robust and Efficient Service Discovery in Large, Highly Mobile Radio Networks

Magnus Skjegstad
University of Oslo
Oslo, Norway
E-mail: moskjegs@ifi.uio.no

Frank T. Johnsen, Trude Hafsøe, Ketil Lund
Norwegian Defence Research Establishment (FFI)
Kjeller, Norway
E-mail: {frank-trethan.johnsen, trude.hafsoe, ketil.lund}@ffi.no

***Abstract*—Existing service discovery mechanisms for ad hoc networks are often designed with one specific network type in mind. Solutions capable of handling highly mobile nodes usually have high bandwidth requirements, particularly as the number of nodes increases. The bandwidth requirement can be reduced by locally caching state information, but this increases the risk of nodes having outdated state information when mobility is high. Some solutions avoid these two issues by tightly coupling service discovery with the routing mechanism itself. However, this requires that nodes are homogeneous on the network layer.**

We propose a solution that leverages the special properties inherent in broadcast-based radio networks. In such networks, every node within transmission range will hear a transmission, be it unicast or broadcast. Each node therefore aggregates relevant service information and broadcasts it at regular intervals. Unnecessary transmissions are suppressed by efficiently synchronizing local state information.

In this paper, we describe the Mist-protocol, a robust and efficient adaptive service discovery protocol, that supports large, highly mobile networks consisting of heterogeneous nodes. We test the protocol in large scale simulations in both static and mobile environments. Finally, we show that it is feasible to actually implement the design by providing a proof-of-concept prototype, which has been evaluated in a small scale experiment.

I. Introduction

The *NATO Network Enabled Capability (NNEC) feasibility study* has identified *Web services* as a key enabler for NNEC [1]. Currently, Web services technology is in widespread use in civil systems. The technology is designed for use in fixed infrastructure networks, and can also function in ad hoc networks in an office environment. However, challenges arise when we attempt to utilize the technology in military radio networks. Such networks are significantly different from enterprise networks, in that they rely on specialized military communications hardware rather than civil commercial products. In general, we can divide military networks in three categories, depending on the hardware being used [2]:

1) Strategic networks.
2) Tactical deployed networks.
3) Tactical mobile networks.

One of the main challenges related to the usage of Web services in and across such heterogeneous military networks is the ability to discover available services. The large variations in capabilities in these networks mean that successfully deploying Web services requires solutions that can take these variations into account. Each network type needs a service discovery mechanism that is specifically designed for the properties of that network, and pervasive discovery across network boundaries can then be achieved through the use of service discovery gateways [3].

Mian et al. [4] classify networks based on size and mobility, and give recommendations for what type of service discovery solutions that fit each network type. From their classification we can see that there exist solutions that work in networks of different sizes *provided that mobility is moderate or low*. For high mobility networks, the available solutions are only suitable for small networks. A discovery solution supporting nodes ranging from low to high mobility in large radio networks remains an unsolved issue that we address in this paper with our novel Mist-protocol.

II. Related work

In previous research, we have designed a service discovery mechanism, «Service Advertisements in MANETs» (SAM), that can be employed in small, highly dynamic mobile ad hoc networks (MANETs) [5]. Also, we have developed and experimented with a solution called Search+ for service discovery across large deployed tactical networks [6]. These mechanisms are intended for use in networks where the standardized Web services discovery mechanisms (e.g., WS-Discovery [7]) do not function well, such as military tactical networks.

The protocol by Sailhan et al. [8] supports discovery of Web services in MANETs. It is based on a backbone of cooperating directories that are deployed in a multi-hop ad hoc network. The protocol has been implemented as a Java prototype, which can operate over a one-hop or multi-hop ad hoc network. The prototype is built on IEEE 802.11b, and uses «Web Service Definition Language» [9] (WSDL) for service descriptions and SOAP [10] for service invocation. Since the solution is based on a backbone of registries, it can be categorized as a peer-to-peer (P2P) discovery mechanism with overlay support, which is unsuitable for highly mobile networks.

«Service-oriented Peer-to-Peer Architecture» (SP2A) is a lightweight service oriented framework for P2P based resource sharing in grid environments, proposed by Amoretti et al. [11]. It relies on JXTA ([11], references therein) as the underlying P2P overlay which is based on a semi-structured protocol. This makes it unsuitable for networks with high mobility.

pService [12] is a P2P-based service discovery mechanism based on a structured overlay (Chord). This mechanism does not support mobility.

Suri et al. [13], [14] propose a Gnutella-like P2P network for resource and service discovery in MANETs. Their approach uses flooding to disseminate service information in a proprietary format. To our knowledge, the protocol has not been tested in networks with more than a limited number of nodes.

Bloom filters [15] have been used for information reconciliation in earlier works, in particular [16] and [17]. These mechanisms focus on parallel delivery of content streams in overlay networks and are optimized for collaborative distribution of large data elements. Mist is different from these protocols in that it is tailored for distribution of singular small data elements (advertisements) in mobile radio networks.

III. The Mist-protocol

The Mist-protocol is a generic mechanism for informed delivery of singular data elements. In this paper, we focus on its application as a Web service discovery mechanism. In practice, it could be used to disseminate any type of data that can be represented as a single network message.

Mist's core functionality is a subscription-based distribution mechanism for advertisements. The advertisements can be seen as small units of information, categorized according to interest. Based on the interest, expressed as *topics*, nodes can ask their neighbors to only send advertisements they are interested in.

In this section we give a general overview of the distribution mechanism before we proceed to describe the two message types used in Mist; the *Subscription* message and the *Advertisement* message. Finally, we describe the algorithms used to process and distribute these messages.

A. Message distribution

Messages in Mist are distributed using a single-hop broadcast mechanism provided by the network layer. Each Mist-enabled node always uses broadcast to distribute information, even when there is a single recipient. This is based on the observations that; a) in radio networks, unicast is based on broadcast messages, and b) even if there is a single recipient, other nodes within radio range may have an interest in the same information. As unicast and broadcast essentially have the same cost in wireless networks, Mist is based purely on broadcast.

Multi-hop dissemination is handled by the algorithm itself, enabling Mist to function in networks without a working routing mechanism. Each node periodically consults the list of topics that its neighbors are interested in, selects which advertisements that should be distributed and broadcasts them to all nodes within radio range. Nodes also notify their neighbors about which advertisements they have received to reduce the amount of broadcast messages. The length of the update intervals are based on the radio range and movement speed of each node and is further described in section III-D.

To support sparsely connected networks, our algorithm uses a store-carry-forward mechanism. This means that we take advantage of node mobility to increase the dispersion of advertisements in the network. When a node physically moves from one position to another, it automatically relays relevant information from its previous location. This enables the use of carrier-nodes that can be used to carry information between networks without direct radio contact. Similarly, when a node loses contact with the network, the remaining neighbors caches information on its behalf for a given time period or until contact is re-established. Nodes may themselves adjust how long neighbors should store information on their behalf by specifying longer or shorter timeouts in their Subscription message, as described in the next section.

B. The Subscription message

Each node running Mist periodically sends a Subscription message, containing information about which topics it is interested in and for how long. Each topic is a unique text string that categorizes information the node is interested in. In a Web services environment where nodes are interested in services, a topic is typically the namespace (or namespaces) a service belongs to. Topics may also be based on well-known unique identifiers (e.g. UUIDs or GUIDs). When a node sends a Subscription message to its neighbors, it must include the set of topics it is interested in. This set may include wildcards (e.g. all topics that start with "no.ffi") and regular expressions.

Each topic in the Subscription message is associated with a time-to-live (TTL) value. This value determines how far, in hops, the topic request should be propagated. When a node receives a subscription for a new topic, the topic is added to a local set of topics and the TTL value is decreased. If the TTL becomes 0, the topic is discarded. The remaining topics with TTL > 0 are included in future subscriptions sent to neighboring nodes. The TTL is handled at the application level by Mist, and must not be confused with the network level TTL field in the IP header.

As an example, Node A is interested in Topic 1 and Node B is interested in Topic 2. Node A adds Topic 1 to its Subscription message with TTL=3. The Subscription message is broadcast on the network medium and received by Node B. Node B adds Topic 1 to its local set of topics, but with TTL=2 (decreased by 1). The topics Node B is interested in are now Topic 1 (with TTL=2) and Topic 2 (with TTL=3). Node B will in turn send its own Subscription message with both topics included. We have shown in our earlier work that this mechanism is effective, even with low TTL-values [6]. It should be noted that the TTL-value follows the topic request and not the Subscription message itself.

The Subscription message includes an acknowledgment field, which contains the unique identifiers of advertisements it has received. The sender only needs to include advertisements that matches the topics it subscribes to, since these are the only advertisements that neighboring nodes will broadcast. The topics a node is interested in includes subscriptions

from nearby neighbors with a decreased TTL, as previously described.

The acknowledgment field allows neighboring nodes to avoid retransmitting advertisements to subscribers. Nodes that move to a new area of the network are able to simultaneously notify neighbors of their presence and inform the new neighbors about which information it has already received. Since state can be transferred in a single Subscription message the protocol is very robust to physical network disruptions like partitioning. It is also able to quickly adapt to changes in the topology. To limit the size of this field it is represented as a Bloom filter. A Bloom filter with a reasonable rate of false positives (0.00819) requires only 10 bits per stored element. Bloom filters and the acknowledgment mechanism are further discussed in Section III-F.

The full content of the Subscription message is shown in Table I.

TABLE I
SUBSCRIPTION MESSAGE

Field	*Description*
sourceNodeId	A unique identifier of the subscribing node.
topics	A list of topics the subscribing node is interested in. Each topic is associated with a time-to-live value that describes the distance to the closest node that requested this topic.
subscriptionTimeout	A timeout value in seconds. When the timeout expires the subscription is terminated (unless it is renewed).
position	The sender's geographical position (optional).
acknowledgment	A Bloom filter containing the identifiers of the already received advertisement.

After receiving Subscription messages, the node has an overview of which advertisements neighboring nodes are interested in. Based on this overview, the node broadcasts Advertisement messages periodically.

C. The Advertisement message

Each node creates a set of advertisements containing the information it wants to make available. This information may for instance be a set of service descriptions (e.g. WSDLs). A single Advertisement message may include information about one or several services.

The contents of the Advertisement message is shown in Table II.

An Advertisement message is uniquely identified by the set (sourceNodeId, version). All Advertisement messages must include this set. The source node identifier may be the node's IP address, but this is not a requirement. As Mist uses network broadcast, it does not rely on knowing the IP addresses of the participating nodes. It is however important that the same node identifier is not shared between multiple nodes.

A version number is included to enable nodes to send updates to existing advertisements. Upon receiving two Advertisement messages with identical identifiers, the receiver should discard the message with the lowest version number.

TABLE II
ADVERTISEMENT MESSAGE

Field	*Description*
sourceNodeId	A unique identifier (e.g. IP address) of the node that created the Advertisement.
version	A version number that is incremented when the Advertisement is changed and needs to be redistributed.
topics	A list of topics the Advertisement covers.
timeout	Life-time in seconds.
position	Geographical location of node that sent the advertisement (optional).
data	Additional data elements, e.g. full service descriptions as WSDLs or QoS information. This field is implementation-specific.

The Advertisement message also includes a list of topics covered by the advertisement. While the topics in the Subscription message may contain wildcards, the topics in the advertisements must be exact. Note that TTL-values are not included here, as these are only applicable to Subscription messages.

The timeout field is used to describe how long the information stored in the advertisement is relevant. After this timeout has expired, the advertisement should be dropped by all nodes. To avoid the need for clock synchronization between participating nodes, this timeout is based on the number of seconds the receiving node has stored the advertisement. When the Advertisement message is forwarded between nodes, the holding-time is subtracted from the timeout in the forwarded copy. The timeout does not account for transfer and processing times and should be considered an approximated value.

As Mist works independently of the routing mechanism, an advertisement could have been received from a node that is unreachable through the network layer. The optional position field enables recipients to move towards areas where full connectivity on the network layer may be available. E.g., when Mist is used for service discovery, moving into the same area as the node that sent a service description may be required to be able to invoke the service.

Finally, the Advertisement message includes the data-element. The data-element contains the information element the creator wishes to distribute, e.g. service descriptions. For Web services, it may include full service descriptions (e.g. WSDLs) in networks with high bandwidth or simple keyword-based Bloom filter descriptions in disadvantaged grids. An implementation using Mist to distribute WSDLs efficiently is discussed in Section V.

D. Algorithms

1) Sending a new packet: A network packet in Mist contains an optional Subscription message and zero or more Advertisements messages. At regular intervals, each node consults its internal data structures for new information to broadcast. Each node considers if it should send a new Subscription message and if there are Advertisements that need to be sent to the neighbors.

A Subscription message is added to the packet if one or more of the following criteria are true:

- The topics we are interested in have changed. This can be triggered by the node itself or by neighbors changing their topics.
- A new acknowledgment needs to be sent. This happens when new or updated advertisements have been received.
- A Subscription message has been received from a new node. New neighbors should quickly be notified that we are present.
- If no messages have been sent for a specified interval a Subscription-message should be sent to notify neighbors that we are still alive.
- If we have moved a distance exceeding our radio range. Neighbors in the new area must be notified of our presence.

Advertisement-messages are added to the packet if we have one or more advertisements that one or more *valid* subscribing nodes are interested in.

Algorithm 1 getValidSubscriptions()

```
validSubscriptions ← ∅
for all s ∈ Subscriptions do
  age ← currentTime() − s.lastUpdated
  timeout ← s.message.subscriptionTimeout
  if s.message.position ≠ ∅ then
    distance ← calculateDistance(s.message.position)
  else
    distance = 0
  end if
  if age < timeout and s.retries < maxRetries and
  distance < radioRange then
    validSubscriptions = validSubscriptions + s
  end if
end for
return validSubscriptions
```

The algorithm that determines if a subscription is valid is shown in Algorithm 1. A subscription is valid if a) it has not timed out, b) it is sent from a node within our radio range and c) the retry counter is lower than the maximum number of retries. The retry counter is increased by one each time we send advertisements to this node and reset to 0 each time we receive a response (e.g. a Subscription-message with updated acknowledgments). This limits the number of times we send information to nodes that we have lost contact with.

After a list of valid subscriptions has been gathered, each known advertisement is tested to see if it matches one of the subscriptions, as shown in Algorithm 2. A node is considered interested in the advertisement if it a) matches one or more topics the node is interested in and b) the advertisement has not been previously acknowledged by the node.

When a match is found, the advertisement is added to a list of candidate advertisements. Candidate advertisements are then removed randomly until the number of elements

Algorithm 2 getAdsToSend()

```
adsToSend ← ∅
for all ad ∈ Advertisements do
  for all subscription ∈ getValidSubscriptions() do
    if ad.topics ∈ subscription.topics then
      if ad ∉ s.acknowledgments then
        adsToSend = adsToSend + ad
      end if
    end if
  end for
end for
while count(adsToSend) > maxAdsPerPacket do
  removeRandomAd(adsToSend)
end while
return adsToSend
```

are less than a predefined value. Combined with the fact that messages are only sent at regular intervals, this allows the algorithm to limit the maximum bandwidth required to disseminate advertisements.

The messages that are ready to be sent are finally combined into a single packet and broadcast to immediate neighbors in the network. At this point the retry counter is increased for each neighbor that is interested in one of the advertisements.

2) Receiving a new packet: When a new packet is received, the following actions are performed:

- Register that the node that sent the packet is still alive. Set send retry counter to 0.
- If a Subscription-message is present, update or add acknowledgments and topics in the subscriptions database for the sending node.
- If one or more Advertisements are present, update or add them to the local repository.

E. Setting the beacon interval

Each node must periodically broadcast a beacon to notify its neighbors that they are still present or that they have moved to a new area. The beacon in Mist is a Subscribe-message that is sent either when a) 60 seconds have passed since the last message was sent or b) the node has moved more than its radio range. When a node is moving, the speed is calculated and the beacon update interval is adjusted accordingly. The beacon interval of a moving node can be calculated using Equation 1, where BI is the beacon interval, r is the radio range in meters and s is the speed in m/s. PT is the (worst-case) processing time in seconds required to process the beacon and send a reply. The resulting beacon interval is intended to be high enough for a mobile node to be able to notify neighbors of its presence as well as receiving at least one response before moving to a new area.

$$BI = \left(\frac{r}{s}\right) - PT \tag{1}$$

Assuming a processing time of 2 seconds, a mobile node moving at 1 m/s with radio range of 25 meters, would be

sending a Subscription message every 23rd second.

F. Reducing false positives

As our distribution mechanism is based on using Bloom filters for acknowledgments, there is always a small probability that we falsely acknowledge an advertisement before it has been transferred. The probability of this happening corresponds to the probability of a false positive in the Bloom filter. This probability p is determined by the number of bits in the filter m relative to the number of inserted elements n and the number of hash functions used k, as shown in Equation 2.

$$P_{FP} = \left(1 - e^{-k\frac{n}{m}}\right)^k \quad (2)$$

As an example, using the parameters $m/n = 10$ and $k = 7$, we can achieve a reasonably low p of 0.00819. In other words, when the number of bits in the Bloom filter is 10 times larger than the number of inserted elements, we must use 7 distinct hash functions on each inserted element to maintain a 0.00819 probability of false positives. This results in less than 1 falsely acknowledged message per 100 messages sent, which can be considered reasonable. Additionally, since each node is listening to traffic sent to other neighbors, the chance of receiving a message even when it has been falsely acknowledged is high.

IV. Evaluation

We have evaluated Mist in simulations using ShoX [18], an event-driven MANET simulator written in Java. Since we focus on Web services, we have used SOAP-over-UDP [19] (without retry messages) for delivery of Subscription- and Advertisement-messages.

For our simulations we used ShoX' built-in 802.11b support with radio range set to 25 meters and interference set to 50 meters. When interference occurs, affected packets are dropped. The mobility model is random waypoint with fixed speed and no pause time. We tested our algorithm with 250 nodes in an area of varying size. The smallest area was 156 x 157 meters, which equals an average degree of 20 - i.e. each node had 20 reachable neighbors on average. The largest area we tested was 700 x 700 meters, equaling an average degree of 1 node. All the configurations we tested are shown in Table III. The simulations end after 120 seconds. The algorithm is configured to send at most one packet per second and at most 10 advertisements in each packet. Each advertisement has a randomly generated data element of 200 bytes, except for degree 10 where we have included a 2500 byte element to see how this affects the bandwidth usage. For the bandwidth experiments we have also tested 50 and 100 advertisements per packet.

A. Static environments

Although Mist is created for mobile environments, a simulation of stationary nodes provides a good impression of what can be expected of the mechanism. A well connected static network can be seen as a best case scenario, as mobility complicates the communication pattern. For sparsely connected networks, however, the mechanism relies on mobility as is shown in the next section.

TABLE III
Evaluated configurations

Area (m)	*Deg.*	*Nodes*	*Speeds (m/s)*	*Msg./packet*	*Ad. size*
157x156	20	250	0, 1, 3, 5	10	200
181x181	15	250	0, 1, 3, 5	10	200
221x221	10	250	0, 1, 3, 5	10, 50, 100	200, 2500
313x313	5	250	0, 1, 3, 5	10	200
443x443	2.5	250	1, 3, 5	10	200
700x700	1	250	1, 3, 5	10	200

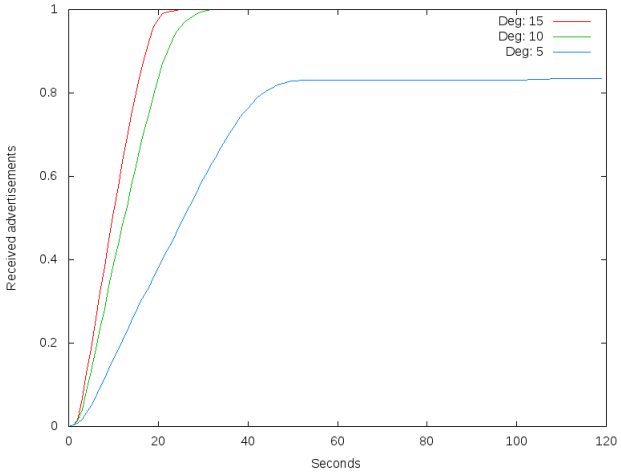

Fig. 1. Global search results for 250 randomly positioned stationary nodes. The success rate represents total received vs. available advertisements. We can see that the success rate reaches 100% after less than 30 seconds for well-connected networks (degree >= 10).

Figure 1 shows the ratio of advertisements received from other nodes over time. As each node publishes exactly one advertisement, 250 received advertisements equals a ratio of 1.

As we can see, the results vary depending on the average degree of the network. In well connected environments (degree >= 10), each node has a full overview of the network after 20 to 40 seconds. For lower degrees, the received advertisement ratio never reaches 1. This is expected, as lower degrees tend to produce isolated clusters in the topology. We have not included degrees lower than 5 in the graph shown here, as they show the same tendency.

To verify that the mechanism works well within isolated areas of the network on lower degrees, we ran a second simulation with a modified success criteria. In this simulation, the received advertisement ratio is 1 when a node has received all advertisements from reachable nodes. A node is considered reachable if it is possible to create one or more paths in the network that eventually reaches the node, e.g. nodes that would be reachable with an optimal routing algorithm. The results are shown in Figure 2. As we can see, Mist provides full advertisement distribution within the isolated clusters.

Figure 3 shows how the average SOAP message size varies with the maximum number of advertisements per message and

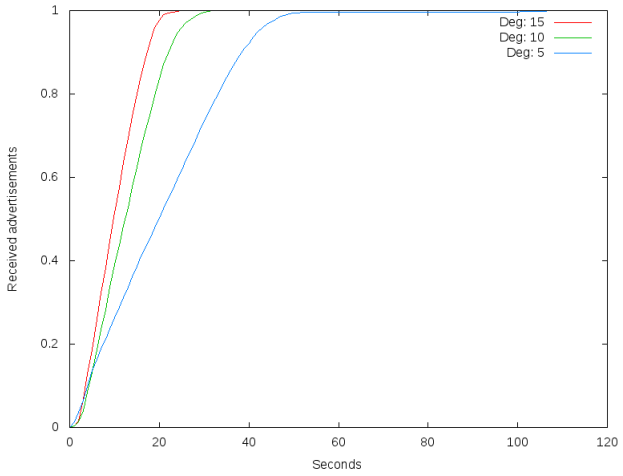

Fig. 2. Local search results for 250 randomly positioned nodes in a static environment. When nodes only consider advertisements from nodes that are reachable through one or more hops, the success rate reaches 100% also for lower degrees.

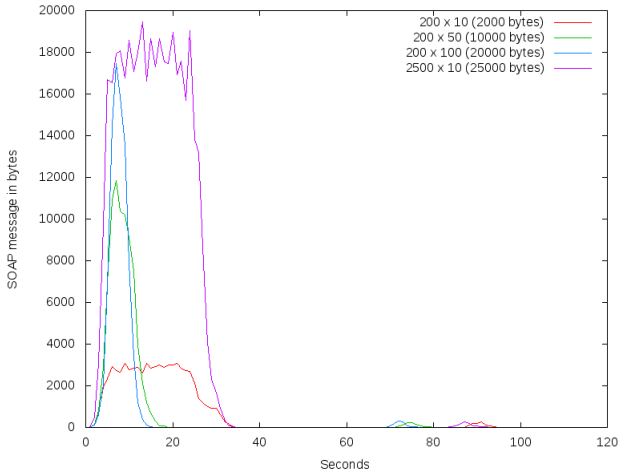

Fig. 3. Average SOAP message size in bytes for average degree 10 with no mobility. Maximum number of advertisement messages per packet is set to 10, 50 or 100 for messages of 200 bytes and 10 for messages of 2500 bytes. We can see that the bandwidth consumption is relative to these values. After 60 seconds without traffic a beacon message is sent, which leads to the small increase in SOAP messages at around 80 seconds.

the advertisement size in a static network with degree 10. 10 advertisements of 200 bytes per packet (200 x 10) has the lowest bandwidth requirements at around 3000 bytes/second in the initial period. The extra overhead is caused by the included Subscription message as well as by the SOAP message format.

We can see that after about 30 seconds, messages no longer need to be sent until the beacon message is transmitted 60 seconds later. The initial traffic ends when the algorithm reaches full advertisement distribution, as seen in Figure 1. By increasing the advertisement size, the bandwidth consumption also increases.

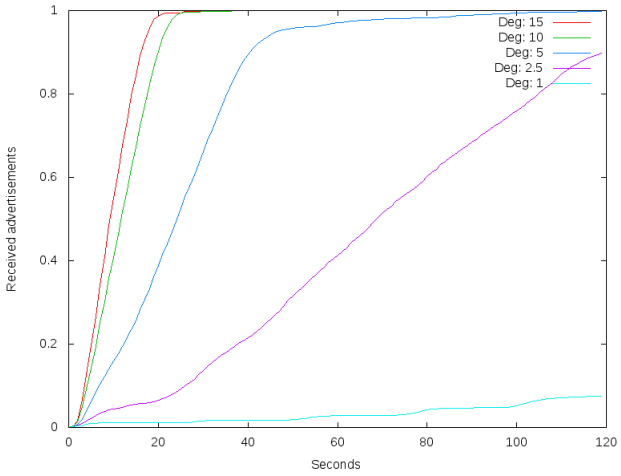

Fig. 4. Ratio of advertisements delivered when the nodes are moving at 1 m/s. Due to the mobility, Mist is now able to distribute the advertisements to all nodes with average degree 5.

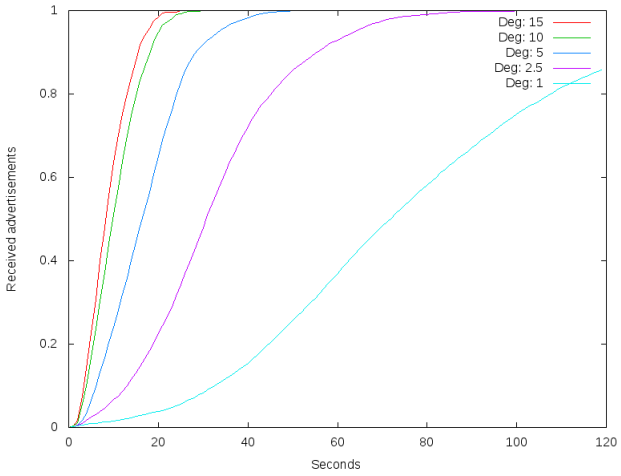

Fig. 5. Ratio of advertisements delivered when the nodes are moving at 5 m/s. The increased mobility allows Mist to distribute advertisements to networks with average degree as low as 2.5 and 1.

B. *Mobile environments*

Due to the store-carry-forward mechanism incorporated in the Mist protocol, nodes can carry data with them when they move between network partitions. This has the effect of allowing full advertisement distribution even in scarcely connected networks. The time it takes for the advertisements to be fully distributed depends on how fast the nodes are moving and the average degree of the network.

Figure 4 shows the average advertisement distribution over time when the nodes are moving at 1 m/s. This corresponds to normal walking speed. As we can see, the movement enables better advertisement distribution at lower degrees. With 5 neighbors on average, Mist is able to reach 100% advertisement distribution within 100 seconds.

Figure 5 shows how the advertisements are distributed when the nodes are moving at 5 m/s. This speed corresponds

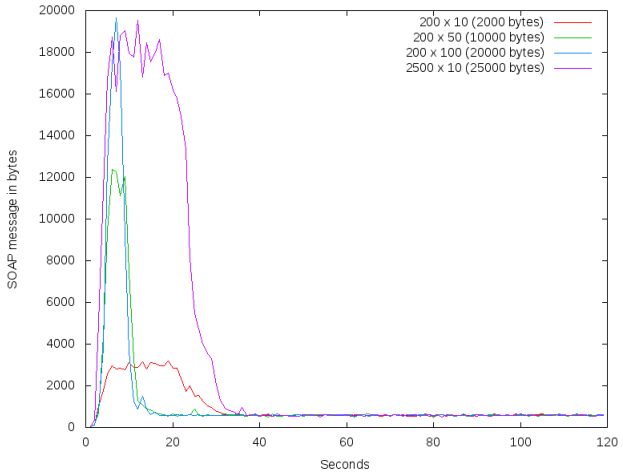

Fig. 6. Average SOAP message size in bytes for average degree 10 with nodes moving at 5 m/s.

to 18 km/h. We can see that the increased mobility helps advertisement distribution and that an average degree of 2.5 gives 100% distribution after 100 seconds. An average degree of 1 gives an 85% distribution after 120 seconds, which we consider acceptable.

Figure 6 shows the average SOAP message size for varying packet sizes at degree 10 with 5 m/s movement. We can see that the results are similar to the results for static networks shown in Figure 3. The movement triggers periodic Subscribe-messages that produce a steady traffic of around 500 bytes/second. The Subscribe-messages are sent with a rate determined by Equation 1, described in Section III.

Using Equation 1, having a node radio range of 25 meters moving at 5 m/s would require a $BI = 5 - PT$. In our simulation we have used a PT of 2 seconds, resulting in a $BI = 3$. Updating every 3 seconds is bandwidth intensive, as we can see in Figure 6. To reduce the bandwidth consumption the speed would have to be lowered or the radio range increased, as described in Section III-E. A radio range of 150 meters and movement of 5 m/s would result in a beacon interval of 28 seconds, which would bring the bandwidth use much closer to the static results. Similarly, a radio range of 1000 meters and movement at 20 m/s results in a beacon every 50 seconds.

V. Practical Experiment

Through the simulations discussed in Section IV, we have shown that Mist is a resource efficient and scalable protocol for information delivery, suitable for use in large mobile radio networks.

We have implemented a service discovery mechanism using the Mist-protocol, called Mist-SD. In Mist-SD, services are described by a simple service description coupled with a hash-value that represents the WSDL. The WSDLs are then distributed as separate advertisements. By separating service descriptions from WSDLs, nodes may choose not to subscribe to full WSDL-files when they are not needed. WSDLs may also be pre-distributed to save bandwidth. The default message format is SOAP-over-UDP (with 0 retries for transmission), but the implementation also supports a binary format for bandwidth constrained environments. The implementation is written in Java.

We have used the implementation for a simple proof-of-concept experiment. The experiment shows that the Mist protocol is feasible to implement and extend, and that it functions as expected on actual hardware. For the experiments we used five laptops running Windows XP SP3, using built-in 802.11b WiFi cards configured in ad hoc mode (i.e., using only P2P connections, no central base station required). We did not configure a routing protocol, meaning that no network level multi-hop communication was possible (i.e., all multi-hop forwarding was performed at the application level by our Mist protocol). This was done to show the independence of Mist from the routing layer, as well as proving that it could function as the simulations promised by utilizing only node mobility and broadcast range for information dissemination.

The experiment consisted of two deployed clusters (immobile units) and a mobile unit that would go in and out of reach of the two clusters. The clusters were out of radio range of each other, and could not communicate directly. This setup allowed us to test all aspects of Mist, i.e. the subscriptions for topics as well as the timeout of advertisements. First, we configured subscriptions for all nodes. Then, we deployed some services in the clusters: The first cluster consisted of two laptops where one offered a couple of services and the other offered none. Since the two machines in this cluster were in radio range of each other, both machines were aware of the services offered in that area. The second cluster, deployed in a separate location with no link to the first cluster, also consisted of two laptops, here each laptop offered one service. The mobile node started out of radio range of both clusters, offering one service. Moving around, first via the first cluster, then via the second, the node was able to discover and store-carry-forward service information from one cluster to the next. The advertisement timeouts ensured that outdated information was removed from each node's service cache after the timeout period had expired.

The experiment was successful, thus showing the proper functioning and implementational feasibility of the Mist protocol design. Our prototype implementation is currently being packaged for release as an open source project, and will become available for download from "http://mist-sd.googlecode.com/".

VI. Conclusion

The Mist protocol is a novel, generic information distribution protocol for large, highly mobile radio networks. We have designed and implemented the protocol and evaluated it in a service discovery scenario using SOAP-over-UDP. The protocol is very robust and resilient to disruptions and partitioning. It has limited bandwidth requirements, making it suitable for tactical networks. The protocol is fully functional without a working routing mechanism and is implemented

on the application layer, enabling it to function on any unit capable of radio broadcast without modifications to lower layers. The protocol supports nodes with varying radio range. Optional positioning information allows service consumers to move geographically closer to a discovered service to improve connectivity.

VII. Future work

In this paper we have used Mist for service discovery. In this context, further work with regards to gateway functionality is required. If time and resources permit it we will evaluate the protocol in large-scale real-life experiments. We will continue our work to improve Mist with focus on mobile networks with low bandwidths. We will also consider implementing it for other scenarios where the protocol may be suitable.

References

[1] P. Bartolomasi, T. Buckman, A. Campbell, J. Grainger, J. Mahaffey, R. Marchand, O. Kruidhof, C. Shawcross, and K. Veum, "NATO network enabled capability feasibility study," Version 2.0, October 2005.

[2] F. T. Johnsen, T. Hafsøe, and M. Skjegstad, "Web services and service discovery in military networks," The 14th International Command and Control Research and Technology Symposium (ICCRTS), Washington DC, USA, 2009.

[3] F. T. Johnsen, J. Flathagen, and T. Hafsøe, "Pervasive service discovery across heterogeneous tactical networks," in *IEEE Military Communications Conference (MILCOM 2009)*, Oct. 2009, pp. 1–8.

[4] A. N. Mian, R. Baldoni, and R. Beraldi, "A survey of service discovery protocols in multihop mobile ad hoc networks," in *IEEE Pervasive computing*, January-March 2009, pp. 66–74.

[5] F. T. Johnsen, "An NFFI-based Web service discovery mechanism for tactical networks," *The Military Communications and Information Systems Conference (MCC 2009), Prague, Czech Republic*, September 2009.

[6] M. Skjegstad, U. Roedig, and F. T. Johnsen, "Search+: A resource efficient peer-to-peer service discovery mechanism," IEEE MILCOM, Boston, MA, USA, October 2009.

[7] V. Modi and D. Kemp (eds.), "Web services dynamic discovery (ws-discovery) version 1.1," http://docs.oasis-open.org/ws-dd/discovery/1.1/wsdd-discovery-1.1-spec.pdf, July 2009.

[8] F. Sailhan and V. Issarny, "Scalable service discovery for MANETS," in *Third IEEE International Conference on Pervasive Computing and Communications (PERCOM)*. IEEE Computer Society, March 2005, pp. 235–244, ISBN 0-7695-2299-8.

[9] W3C, "Web Services Description Language (WSDL) 1.1," http://www.w3.org/TR/2001/NOTE-wsdl-20010315, W3C Note 15 March 2001.

[10] ——, "SOAP Version 1.2," http://www.w3.org/TR/soap12, W3C Recommendation 27 April 2007.

[11] M. Amoretti, F. Zanichelli, and G. Conte, "SP2A: A service-oriented framework for P2P-based grids," *Proceedings of the 3rd International Workshop on Middleware for Grid Computing (MGC05), Grenoble, France*, 2005.

[12] W. Lv and J. Yu, "pservice: Peer-to-peer based web services discovery and matching," in *Systems and Networks Communications, 2007. ICSNC 2007. Second International Conference on*, aug. 2007, pp. 54 –54.

[13] N. Suri, M. Rebeschini, M. Breedy, M. Carvalho, and M. Arguedas, "Resource and service discovery in wireless ad-hoc networks with agile computing," in *Military Communications Conference, 2006. MILCOM 2006. IEEE*, oct. 2006, pp. 1 –7.

[14] N. Suri, M. Marcon, R. Quitadamo, M. Rebeschini, M. Arguedas, S. Stabellini, M. Tortonesi, and C. Stefanelli, "An adaptive and efficient peer-to-peer service-oriented architecture for manet environments with agile computing," in *Network Operations and Management Symposium Workshops, 2008. NOMS Workshops 2008. IEEE*, april 2008, pp. 364 –371.

[15] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Commun. ACM*, vol. 13, no. 7, pp. 422–426, 1970.

[16] J. Byers, J. Considine, M. Mitzenmacher, and S. Rost, "Informed content delivery across adaptive overlay networks," *Networking, IEEE/ACM Transactions on*, vol. 12, no. 5, pp. 767 – 780, oct. 2004.

[17] D. Kostic, A. Rodriguez, J. Albrecht, and A. Vahdat, "Bullet: High-bandwidth data dissemination using an overlay mesh," *Proc. ACM SOSP*, pp. 282–297, 2003.

[18] J. Lessmann, T. Heimfarth, and P. Janacik, "ShoX: An Easy to Use Simulation Platform for Wireless Networks," in *UKSIM '08: Proceedings of the Tenth International Conference on Computer Modeling and Simulation*. Washington, DC, USA: IEEE Computer Society, 2008, pp. 410–415.

[19] OASIS, "Soap-over-udp version 1.1," http://docs.oasis-open.org/ws-dd/soapoverudp/1.1/os/wsdd-soapoverudp-1.1-spec-os.html, OASIS Standard, 1 July 2009.