Multicast Gateway for Service Location in Heterogeneous Ad Hoc Communication

Qi He, Dan Muntz
Mobile and Media Systems Laboratory
HP Laboratories Palo Alto
HPL-2002-233
August 19th , 2002*

E-mail: qhe@cc.gatech.edu, dmuntz@mail.com

service location, service discovery, multicast, ad hoc network, appliance aggregation

Service discovery is essential to enable users to effectively search for dynamically available services. Multicast is used extensively by many service discovery protocols, but it is particularly expensive in ad hoc networks. It is also often difficult to interface different multicast protocols used in different types of networks. In this work, we design a service discovery protocol for heterogeneous ad hoc communications that combines the solutions to these two issues. At the center of the protocol are: 1) clustering of devices according to the physical communication media each device supports; 2) gateways that connect physical clusters and at the same time perform scoping, tunneling and informed forwarding of multicast service discovery traffic. The protocol demonstrates the use of scoping and tunneling to improve multicast performance at the session layer and for heterogeneous ad hoc networks. The use of physical communication-based clustering at the service location protocol is an example of cross-layer optimization. Experiments with our Linux implementation of the protocol demonstrate the effectiveness of the protocol to improve the latency of service discovery and reduce bandwidth consumption.

* Internal Accession Date Only
Approved for External Publication
© Copyright Hewlett-Packard Company 2002

Multicast Gateway for Service Location in Heterogeneous Ad Hoc Communication

Qi He, Dan Muntz
qhe@cc.gatech.edu, dmuntz@mail.com

Abstract

Service discovery is essential to enable users to effectively search for dynamically available services. Multicast is used extensively by many service discovery protocols, but it is particularly expensive in ad hoc networks. It is also often difficult to interface different multicast protocols used in different types of networks. In this work, we design a service discovery protocol for heterogeneous ad hoc communications that combines the solutions to these two issues. At the center of the protocol are: 1) clustering of devices according to the physical communication media each device supports; 2) gateways that connect physical clusters and at the same time perform scoping, tunneling and informed forwarding of multicast service discovery traffic. The protocol demonstrates the use of scoping and tunneling to improve multicast performance at the session layer and for heterogeneous ad hoc networks. The use of physical communication-based clustering at the service location protocol is an example of cross-layer optimization. Experiments with our Linux implementation of the protocol demonstrate the effectiveness of the protocol to improve the latency of service discovery and reduce bandwidth consumption.

Keywords: service location, service discovery, multicast, ad hoc network, appliance aggregation

1 Introduction

With the increasing usage of network-enabled mobile devices, we are more likely to be in a changing and unfamiliar environment. This, along with the increasing number of services available in the network, makes it essential to discover services dynamically. Many service discovery protocols, such as the IETF Service Location Protocol (SLP[2]) and Bluetooth SDP[7], use multicast extensively. Although most of these protocols have considerations for lower-layer protocol cost, service discovery has been primarily investigated as a session layer issue, independently of lower-layer protocols. For heterogeneous ad hoc communication, one of the enabling technolgies for mobile computing, there are two specific issues that are not well addressed by these protocols:

- multicast routing between different physical communication media is often not available due to the difficulties involved in interfacing different multicast protocols used by each communication media
- ad hoc multicast is particularly expensive

Ad hoc multicast routing is expensive in bandwidth and power since flooding is usually involved for every packet forwarding, as we will describe in section 2.3. People have thus considered some form of node clustering indispensible for ad hoc multicast routing[4]. Many research efforts[4, 3] on multicast routing protocol for a specific type of ad hoc network have focused on an efficient dynamic clustering algorithm. Scoping of flooded multicast traffic could also be helpful. However, unlike the Internet, where multicast group membership and forwarding state can be aggregated based on networks/sub-networks and thus the scope of multicast traffic determined, an ad hoc network has a flat routing structure and consists of devices with unconstrained mobility, which makes it difficult to scope flooded multicast traffic.

In this work, we are interested in reducing multicast cost of a service discovery protocol through scoping and tunneling at a higher protocol layer. In particular, we observe that for the heterogenous ad hoc networking context we are targeting, there is a relatively stable grouping where devices are clustered according to the physical communication medium each device uses. Devices of two different clusters can communicate with each other only via one or more gateway devices that span clusters (supporting multiple physical communication media). We propose to exploit this clustering to improve the performance of service discovery in ad hoc networks. The basic idea is to scope service discovery traffic to local physical clusters, or to unicast tunnel service discovery traffic to destination clusters, both according to a small amount of

state maintained on gateways. Since the clusters connected by a gateway may use incompatible multicast routing protocols, a gateway also serves to connect the two clusters for the service discovery protocol.

To that end, we first need to devise a way for the service discovery protocol to use the clustered topology. We have each gateway device join the service discovery multicast group of adjacent clusters so that forwarding between clusters can be performed. Secondly, we need to consider how we can benefit from using clustering in the resource discovery protocol. Our approach is for gateway devices to maintain a small amount of information about the service types available in each cluster in the whole network. With this information, 1) certain service discovery queries can be satisfied by local gateways; 2) other queries can be tunnelled directly to relevant clusters rather than be multicast throughout the whole network.

The rest of the paper is organized as follows. Section 2 details the application and networking context we are targeting and discusses the performance implications of a service discovery protocol that uses multicast extensively. Section 3 gives an overview of the proposed service discovery protocol. Section 4 describes the details of the protocol. Sections 5 and 6 present the implementation and simulation-based evaluation. Section 7 discusses some future work. The paper is concluded in section 8.

2. Overview of the Problem

2.1. Networking and Application Context

Our proposed service discovery protocol are designed for network environments that have the following characteristics:

- devices use different communication media, wired or wireless, such as 802.11, Bluetooth, IrDA
- network infrastructure or servers are not assumed to exist
- devices peer with each other in an ad hoc way, i.e., there is no prior knowledge of the services available on each other, or server support to lookup each other
- the range of communication is relatively small, such that devices using the same physical ad hoc communication medium can reach each other through multi-hop routing, i.e., they are in the same cluster
- devices have different resource availability, and many are expected to be resource constrained

With the fourth characteristic above, we constrain our current work to situations where devices that have the same physical communication medium form only one multi-hop cluster. Although the ideas in our work apply to situations where this condition does not hold, we need to address additional problems in those cases, such as device mobility betweeen two clusters of the same communication medium.

Applications running within such a context are becoming more and more pervasive. Some examples include mobile device users who participate in an interactive lecture or conference, personnel coordinating in a disaster relief effort, and soldiers sharing information on the battlefield[9]. In particular, we identify appliance aggregation as a promising application, which has become a topic of interest recently. Through sharing of resources, appliance aggregation can both make individual devices more usable and enable many distributed applications among the devices. Appliances come with different networking technologies. While there have been lots of research efforts in ad hoc routing for each wireless communication medium, cross media communication has received much less attention. It is one of our goals in this paper to show how cross media communication could be considered along with higher layer protocols.

2.2. Service Discovery and Location

Service discovery and location protocols are designed to simplify or eliminate the configuration needs for users to access the services available in the network. Through a service discovery protocol, a client can:

- browse for available services
- search for (specific) services

. They have been studied for different networking and application contexts, in particular, for Internet/WWW and for mobile/ad hoc users. In the latter category, existing work can be categorized as follows.

Server-based Approaches Many research projects adopt a server-based approach, where clients are statically or dynamically configured with a service discovery or location server on which the client can register and lookup services. The BARWAN[5] project at UC Berkeley automatically reconfigures a device of common local services as the device enters a new location. Those services include common services such as DNS, SMTP, as welll as the resource discovery server. Other work in this category includes INS (Intentional Naming Service)[11] at MIT, Centaurus[10] at UMBC, etc.

Server-less Approaches While server-based approaches eases client implementation, it does not necessarily fit a pervasive computing environment we are targeting, such

as the appliance aggregation application. In ad hoc communcations, fixed infrasture, or a server, is unlikely to be present. High device mobility makes it hard for centralized servers to aid service locating, thus the decoupled service discovery and service locating may make the overall system more expensive. Devices that provide services to each other could join and leave without notice, so that it is more difficult for the central servers to provide correct information. For such ad hoc and dynamic environment, serverless service discovery protocols have been favored.

In server-less protocols, a client usually uses multicast to send queries with service specifications, and matching services reply to the queries. Two examples in this category are SDP[7], the service discovery protocol of Bluetooth and SLP (Service Location Protocol)[2] developed in IETF. In SDP, a service type is represented by a Universally Unique IDentification(UUID). A service in SLP is represented by a string composed of a service type part and a URL part with optional scope and attributes specification.

2.3. Multicast in Ad Hoc Networks

Since serverless service discovery and location protocols are important to ad hoc networking applications and they use multicast heavily, multicast performance in ad hoc networks becomes an interesting issue. Multicast in ad hoc networks has some distinctive characteristics which make it particularly expensive:

- Some form of flooding is usually used because keeping multicast forwarding state is not desirable or feasible[8].
- At the MAC level, multicast packet receiving/sending consumes more power than unicast packet receiving/sending[6].

The unconstrained mobility of devices within a totally flat topology makes it hard to keep track of multicast group membership in ad hoc networks. In addition, it is infeasible to maintain much routing state on the often resource-constrained peer devices. Therefore, ad hoc multicast protocol usually uses flooding or some variation of flooding, instead of keeping multicast forwarding state. With such a protocol, all nodes forward a multicast packet to all neighbouring nodes unless the same message has been received already. We can see that without any knowledge of receivers' position in the network, lots of bandwidth could be wasted to reach receivers. This is different from Internet multicast, where forwarding state can be aggregated and maintained according to receivers' domain or IP address, by which multicast traffic is only directed to those parts of the network where there are receivers. While some form of flooding is also used for establishing unicast routing state in ad hoc networks, flooding for multicast is often per packet forwarding, and hence is much more frequent and pervasive.

2.4. Summary

We concern our work with an efficient serverless service discovery protocol for heterogeneous ad hoc communication. Such a protocol should have low bandwidth/energy cost and low service discovery latency. We have identified that ad hoc multicast can adversely affect the performance of a service discovery protocol, but we do not intend to improve a particular multicast routing protocol, nor are we interested in changing the interfaces provided by the service discovery protocol such that the users use multicast differently. Instead, we try to reduce the cost through session layer scoping and tunneling of multicast traffic.

3. Overview of the System

In this paper, we propose a cluster-based service location protocol that provides interfaces compatible with those of SLP. In this section, we will describe key ideas of the proposed protocol: clustering and informed forwarding, through which the protocol reduces SLP cost in heterogeneous ad hoc communication. To do that, we first have to introduce the main components and service primitives of SLP.

3.1. SLP

SLP[2] is defined to be distributed protocol but with the option to have centralized Directory Agents (DA). SLP defines three main protocol components:

- User Agents(UA) perform service discovery on behalf of the clients
- Service Agents(SA) represent and advertise services
- Directory Agents(DA) collect service advertisement and respond to service discovery queries

Both SAs and DAs join the SLP multicast group. SLP client (UA or SA) can use a statically configured DA or use multicast to dynamically discover DAs. When there is no DA, all registration or service discovery query is multicast to the SA group and those SAs that represent matching services will respond to the query.

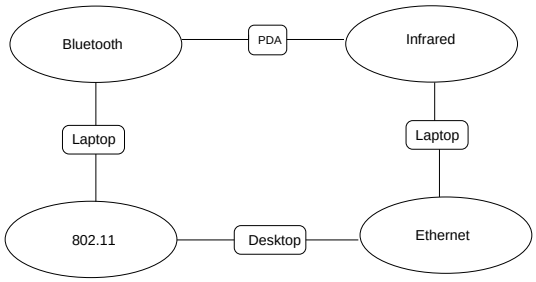

Figure 1. A cluster-based view

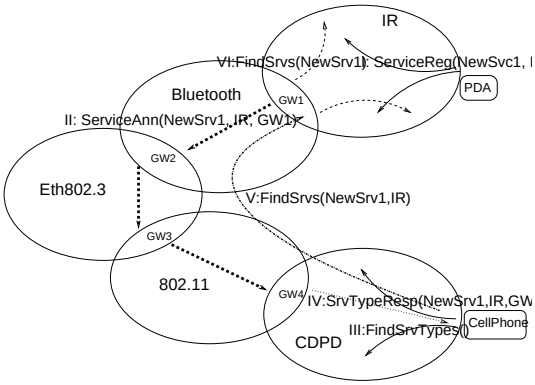

Figure 2. Protocol Operation

SLP Service Primitives Following is a list of major service primitives provided by SLP along with the protocol behavior when there are no DAs[1].

1. *register/deregister*: a registration/deregistration is sent to the SLP group. The registration is cached by local SLP daemon that represents the service.

2. *findsrvtypes*: a multicast query is sent to the SLP group and every SLP daemon should respond with the service types provided by the services it represents.

3. *findsrvs*: a multicast query is sent to the SLP group and those SLP daemon that represents a service matching the specification in the query should respond with the details of that service.

Notice that these primitives are sufficient for the basic services users expect of a service discovery protocol, although there are a few other SLP primitives, such as *findattrs*, *findscopes*, so we focus on the optimized protocol behavior with these operations.

3.2. Clustering

If we can scope multicast traffic within a part of the network or tunnel multicast packets only to those parts of the network that have receivers, we can greatly reduce the multicast-related cost of a protocol in ad hoc networks. However, to do that in a service discovery protocol, we have to maintain the semantics of the protocol. E.g., while a multicast discovery packet can be scoped, we want the originating client to still be able to get information of the whole network. We observe that scoping and tunneling based on network routing is most desirable. If we know that messages will be bound according to certain routing structure, it is much easier to have a strategy to store service information and maintain the semantics of the service resource discovery protocol. This is analogous to some reliable multicast protocols where certain multicast routers are designated to maintain a copy of recently received packets so that local recovery can be provided to downstream group members.

An ad hoc network typically does not have a fixed routing structure, as the intermediate nodes connecting a pair of nodes can be any nodes and keep changing. However, for heterogeneous ad hoc networks, we find another view of the system as shown in Figure 1. In terms of communication, devices are grouped according to the physical communication media they are equipped with. Devices using different communication media can talk to each other only via those gateway devices that have multiple media. Compared to the MAC layer clustering of some ad hoc networks, which is often based on physical proximity and has to adapt to node mobility to dynamically form clusters, this clustering is much more stable since it is unlikely that a device will change the communication media it uses dynamically. This motivates the idea of scoping and tunneling based on this cluster structure and maintaining resource discovery semantics using the gateways.

Many ad hoc multicast routing protocols use clustering based on physical proximity so that flooding is performed not by every node, but by every cluster. Our work is orthogonal to these protocols and can always be used with them. The scoping in our protocol is also different from that of SLP in that an SLP scope is defined to restrict unauthorized accesses from outside a group. SLP also provides the option for applications to specify the maximum hops (TTL) a multicast packet traverse for performance considerations similar to ours. It should be noted that our protocol does not interfere with this feature.

3.3. Gateways

To utilize the cluster structure of the physical network in the service location protocol without integrated layer

[1]Since the protocol behavior without DA is not fully specified in rfc2246, we have to sketch it out based on some references.

processing, the gateway devices in Figure 1 join the SLP multicast group and become the multicast gateways defined in our service discovery protocol. Gateway is one of the three components defined in our protocol. The other two components are UA and SA, both of which perform essentially the same functionaltiy as defined in SLP and run on every node that needs to use service discovery or provide services. A gateway node runs as an SLP daemon that performs the multicast scoping, tunneling and forwarding of SLP traffic, as we will describe shortly.

Notice that in case that the network layer forwarding of multicast traffic between different physical media clusters is already implemented and enabled on a gateway device, we need to map the global SLP group address to a multicast address that is local to and unique to each of the adjacent clusters. In the rest of the description, we assume network layer multicast forwarding is not available, while network layer unicast forwarding is.

A gateway can be configured with or learn the type of communication medium, as well as the IP address of each interface. A node can specify whether it can and wants to act as a gateway, although if zero configuration is strictly required, a node can dynamically find out whether it is a gateway based on whether it has more than active nework interfaces.

3.4. Scoping, Informed Forwarding and Tunneling

To maintain the semantics of the SLP primitives, gateways perform informed scoping, forwarding and tunneling based on certain information they maintain. Since gateways in our protocol are not dedicated servers and are often resource constrained themselves, it becomes a distinguishing characteristic between gateways and the SLP DAs that the amount of state maintained by gateways is much smaller.

We first observe that browsing for services constitutes a heavy usage of a service discovery protocol and that much less information is needed to satisfy this query than queries for the details of particular services. In our protocol, gateways only remember the service types available in each cluster along with the gateways to each cluster. The update of this information is triggered by *register/deregister* invocations. With this information available on every gateway, the *findsrvtypes* query can be satisfied locally and need not be forwarded beyond the local cluster. We call this scoping. The *findsrvs* query can be unicast tunnelled to the gateways of those destination clusters where there are services matching the service type specified in the query and need to be multicast only in those clusters. We call this informed forwarding and tunneling. Whenever a *register* message results in a new service type in a cluster or a *deregister* message results in the disappearance of a service type from a cluster, local cluster gateways should send the service type information to all the other gateways[2], also indicating the cluster where the service type exists and the entry gateway to that cluster.

It should be noted that both the *findsrvtypes* query and the *findsrvs* query can optionally include specification of scopes and/or attributes that furthur narrow down the search results. Apparently, there is a tradeoff between the multicast traffic saving and the amount of state gateways maintain when we decide whether to store scopes and attributes along with service types. Our protocol currrently maintains service type information only.

4. Cluster-based Service Location Protocol

This section provides more details of the protocol behavior related to the *register/deregister*, *findsrvtypes* and *findsrvs* primitives.

Registering Services As a user invokes the *register* primitive and its SA sends out a service registration message (e.g., message I in Figure 2) to the SLP group, as they do in SLP, a local gateway that receives the message checks existing information and decides whether the service type is new to the local cluster. If it is, the gateway adds a record in the form of <*service type, cluster, cnt*>, where *cnt* indicates the number of services of this type in local cluster and is initialized to 1. The local gateway then composes a service *announcement* message (e.g., message II in Figure 2), which conveys the service type, service residing cluster and cluster gateway information, and sends the message to the group of gateways. Upon receiving an *announcement* message, a gateway will create a record in the form of <*service type, cluster, cluster gateway*>. If a service registration provides a service type existing in the local cluster, local gateways will increment the *cnt* of the corresponding record by 1 but will not send an announcement to other gateways. When an SA multicasts a deregistration message, a local gateway that receives the message will decrement the *cnt* of the corresponding record by 1. If *cnt* is reduced to 0 after the decrement, the gateway decides that the service type no longer exists in local cluster and it will compose a *deannouncement* message to other gateways.

Browsing for Services When a UA invokes the *findsrvtypes* primitive, a multicast query (e.g., message III in Figure 2) is sent to the SLP group and the local gateway replies (e.g., message IV in Figure 2) to the

[2]To further reduce the energy consumption of nodes receiving unwanted multicast message, a different multicast address is used by gateways only to receive those messages intended only for gateways.

initiating client with the service types information along with corresponding clusters and gateways. The cluster information might be useful for clients to choose services from specific clusters, e.g., a service in an 802.3 Ethernet cluster might be favored over a service residing in a Bluetooth cluster.

Searching for Services A UA can invoke the *findsrvs* primitive with or without cluster/gateway specifications. The user interface to *findsrvs* is changed to allow users to specify clusters/gateways. If the user does not specify this information, the UA may specify it on behalf of the user. Note that the UA entity is usually implemented as a library or a daemon that is shared by applications on a host. It can cache the results of some previous queries and use some information, clusters information for a specific service type in this case, as input to later queries.

If a local gateway receives a query with cluster/gateway information, it will directly use this information as the basis of forwarding/tunneling. Otherwise, it will lookup the gateways of all the destination clusters that have the matching service type. Once the local gateway has the cluster/gateway information for all the intended destinations, the gateway will unicast tunnel the message (e.g., message V in Figure 2) to those gateways, unless an adjacent cluster is one of those destination clusters, in which case it multicasts the query. Assuming that there is another service of type *NewSrv1* in the 802.11 cluster on Figure 2 and that the CDPD user does not specify cluster information in its *FindSrvs* query, *GW4* will multicast the query to the 802.11 cluster and the same process of deciding whether and how to tunnel will be performed on *GW3*. A gateway receiving a tunnelled *FindSrvs* query will re-multicast the message (e.g., message VI in Figure 2) in the adjacent cluster that has the matching service type.

The *FindSrvs* query also carries a new field indicating the original client that issues the query, so that matching servers can reply to it.

Gateway Dynamics Gateways are usual peer devices that may come and leave without notice. A leaving gateway does not have to do anything. If there are gateways connecting the same two clusters, then its leaving should not have interrupted the normal operation of the service discovery protocol[3]. If two clusters are partitioned after the gateway's leaving, then communication between the two clusters is stopped at the physical layer anyway. For a newly joining gateway to participate in the protocol, it should first get the service type information of the current network from an adjacent gateway.

[3]Some devices may experience longer service discovery delays.

Known Gateway Cache Since many messages sent to the SLP group or gateway group are actually intended for local gateways or gateways of adjacent clusters, one optimization to the protocol is for regular nodes to cache local gateway addresses and for gateway nodes to cache addresses of other gateways in adjacent clusters. Although we can certainly do active discovery for that purpose, a more efficient and adaptive approach is to update the cache with the source address information as messages are received. For a regular node, local gateways are those that have ACKed a *registration/deregistration* message or replied to a *findsrvtypes* message. For gateways, adjacent gateways are those that ACK a service announcement/deannouncement message. As gateways join and leave the aggregation or as gateways move in/out of the range of local gateway, the gateway cache on each node can be updated with the reply messages sent out from the local node, although there will be a failure when a cached gateway is no longer reachable and the message should be resent with multicast.

Loop Detection Physical connections in a heterogeneous ad hoc network can form loops as in any other networks. Hence in forwarding SLP multicast packets, the protocol should avoid loops among gateways. Gateways can run a routing protocol among themselves to avoid loops, as IP routing protocols do. However, such a routing protocol will require gateways to periodically exchange information. In an ad hoc network where gateway themselves could move constantly, the overhead of keeping the routing state up-to-date among gateways will be overwhelming. A more appropriate approach is to dynamically detect loops by stamping each message with the cluster identifications as it passes each cluster/gateway. Since service discovery traffic is not very heavy as compared to (multimedia) data traffic, and the applications we are targeting are not likely to have many clusters, the bandwidth overhead of this approach is unlikely to be significant.

5. Implementation

We implemented the proposed protocol on Linux based on the *openslp*[1] code. We add about 3,000 lines of code to *openslp*. The major changes/additions are related to the *SLPDaemon* (SLP SA), which now supports gateway functionality as detailed in section 4. There are also changes to *libslp* (UA) to provide an enhanced interface to *FindSrvs* and to use gateway caches. The forwarding and tunneling process is based on the information configured for the interfaces of a gateway, the source of a query and the IP routing table of the gateway. At several places, the implementation also uses two multicast related socket options provided by the system: IP_MULTICAST_LOOP to enable/disable delivering multicast packets to receivers

on the same node as the sender, and IP_MULTICAST_IF to set the interface on which to send out a multicast packet. Appendix gives a snippet of pseudo code for forwarding and tunneling of a *FindSrvs* query.

We test the protocol by using it in AAFS (an Appliance Aggregation File System that provides a file system interface for resource sharing among devices in an ensemble) and conducted experiments in a three cluster setting. We use three Ethernet subnets (all composed of Linux machines) to emulate a heterogeneous communication environment. Multicast routing is not available across subnets while unicast routing is supported on the gateway machines (nodes connecting two subnets). Our experiments demonstrate that:

- the protocol provides an SLP gateway between clusters that multicast routing is not supported at the network layer
- the protocol has a shorter latency than SLP for the findsrvtypes query
- the protocol results in less multicast traffic than SLP
- the cost of running a gateway SLP daemon on a device is minimal
- the known gateway cache further reduces multicast traffic

6. Performance

Although we are able to prove that the protocol works, for the following reasons, we are not able to evaluate the performance of the protocol in terms of its energy and bandwidth savings through either real tests or simulations:

- ad hoc multicast protocol implementation is not currently available and it is hard to obtain a reasonably large multi-hop test environment
- ad hoc multicast protocols are not available in network simulation tools in the public domain
- it is difficult to obtain a workload trace for the service discovery protocol

However, if we assume that global flooding is used for multicast in our ad hoc networks, we can count the total number of multicast transmissions, which is often used to measure multicast efficiency[4]. With global flooding, every node that receives a new multicast packet will re-broadcast it. Given that service discovery traffic is light as compared to data traffic, and that global flooding has the best performance for light load multicast traffic[4], it is a reasonable assumption to make. Under this assumption, we use the following synthetic workload to evaluate the protocol:

- we use a real or synthetic trace of multicast group membership
- we use an ensemble environment that has three physical clusters and we assume the three clusters are never partitioned, i.e., there are always gateways connecting them. Each device belongs to one of the clusters with certain probabilities.
- when a new device (member) joins an ensemble (group)
    1. it registers n services, each of which is 1 of S predefined service types, where n is uniformly distributed in (0,N)
    2. it browses (*FindSrvTypes*) for the service types available in the ensemble
    3. it chooses m service types that it is interested in and do a search (*FindSrvs*) for each type, where m is uniformly distributed in (0,N) and is smaller than the maximum number of service types available. Notice that *FindSrvs* with the cluster-based protocol provides the option to specify a cluster. A *FindSrvs* in our experiment uses this option with a probability of 50%
    4. if the new device provides a new service type, other devices in the ensemble performs a search for the new service type with a certain probability
- when a device leaves an ensemble, it deregisters all the services it registered

For the membership traces, we use the following model:

- the system has a fixed population of 450 members
- each member switches between in-ensemble state and out-of-ensemble state. The periods it stays in and out of the ensemble both follow possion distributions and the ratio of mean out-period to mean in-period is a parameter to the trace generation procedure
- the trace has a fixed number (900) of joins/leaves and it ends after the last leave

The main parameters to an experiment are: a membership trace, a distribution of members to the clusters, S and N. For each experiment, we count the multicast transmissions over the whole trace, with both the original and the cluster-based SLP protocol. The ratio (cluster-based SLP vs. SLP) is a measure of relative multicast cost of the cluster-based SLP protocol.

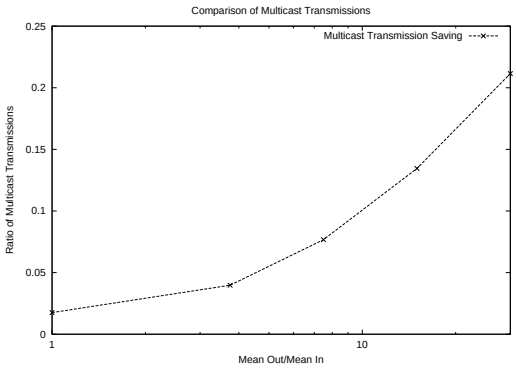

Figure 3. Perf with diff traces

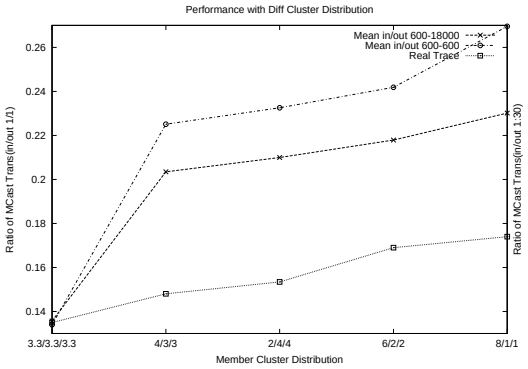

Figure 4. Perf with diff cluster distribution

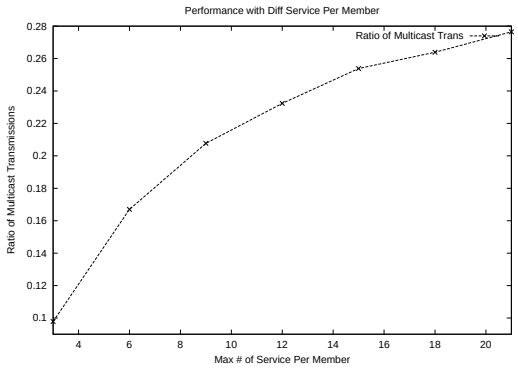

Figure 5. Perf with diff # of services per member

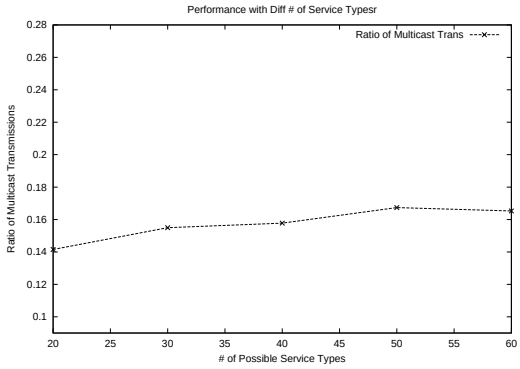

Figure 6. Perf with diff # of service types

Figure 3 shows the multicast transmission comparison between the two protocols with different synthetic traces (we use a 5:3:2 membership distribution across the three clusters in this experiment). We can see that when we decrease the mean length of stay outside the ensemble, thus increase the average group size, the saving by the cluster-based SLP increases, suggesting that the protocol scales with the ensemble size. In this case, cluster-based SLP gains performance primarily on the *FindSrvs* queries. Given certain N, S and member cluster distribution, a *FindSrvs* query has a better chance to be satisfied locally in a larger ensemble than in a smaller one. Figure 4 compares the two protocols with different membership distributions over the clusters. We can see that, for all the membership traces, the highest saving by the cluster-based SLP is achieved when membership distribution is most even. This is explained by the fact that with uniform distribution of services among all devices and without a user's perference of using local cluster services over remote services, cross cluster traffic is minimized when members are most evenly distributed over clusters.

Figure 5 shows that the relative cost of the cluster-based SLP increases significantly as the average number of services provided/imported by a device increases. Given a fixed number of clusters and a certain membership trace, as the average number of services provided/imported by each device increases, there are more chances that a device will issue a query for a service that is in another cluster, thus increasing the relative cost of the cluster-based SLP. Figure 6 shows that the relative cost of the cluster-based SLP increases, although very little, as the number of possible service types increases. Increasing the number of possible service types increases the probability that a device issues a query for a service type not available in its local cluster, with a 50% probability to flood multicast throughout the network.

7. Future Work

We will futher work on solutions to larger range environments where multiple clusters of the same type of

communication medium may exist. Our evaluation has not explicitly shown the bandwidth and energy savings resulted from the less multicast traffic in ad hoc networks. Our future work will focus on a simulation environment where both the heterogeneous ad hoc network environment and an ad hoc multicast protocol are available. We also need to obtain a real workload for our simulation.

8. Conclusion and Acknowledgments

In this paper, we propose to improve the performance of a service location protocol in ad hoc networks through scoping, tunneling of multicast traffic with minimal state maintenance. The work is also an example of cross layer optimization in that the scoping and tunneling in the proposed protocol utilizes the physical layer cluster structure within the network. Our preliminary results from the experiments show that shorter service discovery latency can be obtained and bandwidth/energy cost of multicast traffic can be reduced.

We would like to thank Dejan Milojicic for some helpful discussions and comments on the draft. We would also like to thank Ira Greenberg, Alan Messer, Rajnish Kumar and Vahe Poladian for their useful input.

References

[1] http://www.openslp.org.

[2] http://www.srvloc.org.

[3] C.-C. Chiang and M. Gerla. On-Demand Multicast in Mobile Wireless Networks. In *ICNP*, 1998.

[4] C.-C. Chiang, M. Gerla, and L. Zhang. Forwarding group multicast protocol(FGMP) for multihop, mobile wireless networks. *Cluster Computing*, Dec 1998.

[5] E. B. et.al. A Network Architectuer for Heterogeneous Mobile Computing. *IEEE Personal Communications Magazine*, 1998.

[6] L. Feeney and M. Nilsson. Investigating the Energy Consumption of a Wireless Network Interface in an Ad Hoc Networking Environment. In *Infocom*, 2001.

[7] E. A. Gryazin. Service Discovery in Bluetooth.

[8] G.Tsudik and K.Obraczka. Multicast Routing Issues in Ad Hoc networks. In *International Conference on Universal Personal Communications*, 1998.

[9] J.Broch, D.A.Maltz, D.B.Johnson, Y.C.Hu, and J.Jetcheva. A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols. In *Mobicom*, 1998.

[10] L.Kagal, V.Korolev, S.Avancha, A.Joshi, T.Finin, and Y.Yesha. A highly adaptable infrasture for service discovery and management in ubiquitous computing. Technical report, 2001.

[11] W.Adjie-Winoto, E.Schwartz, H.Balakrishnan, and J.Lilley. The Design and Implementation of an Intentional Naming System. In *ACM SOSP*, 1999.

9. Appendix

```
ProcessSrvRqst(query, src)
{
  incoming_if = RouteLookup(src);
  orig_cluster = query->orig_cluster;

  clusterList = NULL;
  if (query->clusterList == NULL)
      clusterList =
         SLPDatabaseLookup(query->srvtype);
  else
      clusterList = query->clusterList;

  tunnel = 1;
  need_forward = 0;

  while (clusterList)
  {
    cluster = clusterList->cluster;
    if (cluster != orig_cluster )
    {
      if (cluster belongsto G_Clusters)
      {
        tunnel = 0;
        need_forward = 1;
      }
      else
        need_forward = 1;
   }
    clusterList = cluster->next;
  }

  if (need_forward)
  {
    if (tunnel)
    {
      for each gw in query->gwlist
      {
        unicast_to_gw(gw, query);
      }
    }
    else
    {
       for each if in G_interfaces
       {
         if (if != incoming_if)
         {
           setsockopt(G_socket,
                      IPPROTO_IP,
                      IP_MULTICAST_IF,
              &if,
                      sizeof(struct sockaddr_in));
           sendto(SLP_SERVICE_GROUP, query);
         }
      }
    }
  }
}
```