TECHNICAL RESEARCH REPORT

Network Layer Support for Service Discovery in Mobile Ad Hoc Networks

by Ulas C. Kozat, Leandros Tassiulas

TR 2003-1

Network Layer Support for Service Discovery in Mobile Ad Hoc Networks

Ulaş C. Kozat and Leandros Tassiulas
Department of Electrical and Computer Engineering
and Institute for Systems Research
University of Maryland, College Park, MD 20742, USA
Email: {kozat, leandros}@isr.umd.edu

***Abstract*— Service discovery is an integral part of the ad hoc networking to achieve stand-alone and self-configurable communication networks. In this paper, we discuss possible service discovery architectures along with the required network support for their implementation, and we propose a distributed service discovery architecture which relies on a virtual backbone for locating and registering available services within a dynamic network topology. Our proposal consists of two independent components: (i) formation of a virtual backbone and (ii) distribution of service registrations, requests, and replies. The first component creates a mesh structure from a subset of a given network graph that includes the nodes acting as service brokers and a subset of paths (which we refer as *virtual links*) connecting them. Service broker nodes (SBNs) constitute a dominating set, i.e. all the nodes in the network are either in this set or only one-hop away from at least one member of the set. The second component establishes sub-trees rooted at service requesting nodes and registering servers for efficient dissemination of the service discovery probing messages. Extensive simulation results are provided for comparison of performance measures ,i.e. latency, success rate, and control message overhead, when different architectures and network support mechanisms are utilized in service discovery.**

I. Introduction

Flexibility and minimum user intervention are essential for future communication networks which are to be easily deployed and reconfigured automatically when extended with new hardware and/or software capabilities. Service discovery, which allows devices to automatically discover network services with their attributes and advertise their own capabilities to the rest of the network, is a major component for such self-configurable networks. Today, there exist several different (yet overlapping) industrial consortiums or organizations to standardize different service discovery protocols (e.g. *Service Location Protocol (SLP)* of IETF, Sun's *Jini*, Microsoft's *Universal Plug and Play (UPnP)*, IBM's *Salutation*, and Bluetooth's *Service Discovery Protocol (SDP)*) [1], [2], [3], [4], [5] .

Mobile ad hoc networks (MANETs) are the ultimate scenarios where the nodes sharing a common stack of lower layer technologies are able to form a temporary communication network in order to facilitate instant networking needs. MANET nodes may have very little or no knowledge at all about the identities and capabilities of each other. There can be a high degree of variety in terms of the capabilities of each individual device (e.g. support of multiple physical interfaces, processing power, printing capability, multi-media libraries, etc.) and such a heterogeneity makes it even more attractive to establish an ad hoc network. In addition, MANETs are characterized by their highly dynamic, multi-hop, and infrastructure-less nature. The dynamic nature arises from the facts that (i) nodes are free to move, (ii) the adverse channel conditions of the wireless medium (e.g. multi-path fading, shadowing, interference, collisions, etc.) may be in effect, (iii) node failures may occur because of the limited energy of the battery-charged devices, and (iv) nodes may frequently join or leave the network at will. Each node is assumed to have the routing functionality, so when no direct link exists between any two devices, they can still communicate via the intermediate nodes. No existing infrastructure is assumed, because it is desirable to have a stand-alone and self-configurable network even in cases where there occur partitions in the network.

Except for Bluetooth's SDP, none of the proposals for service discovery directly targets an all wireless network. Their applicability and performance in MANET scenarios mainly depend on their assumptions about the available network support. For example, if any fixed infrastructure is assumed in the implementation, it would be against the nature of the ad hoc networking.

Almost all of the service discovery protocols include the client-server paradigm as a mode of operation. In this paradigm, service requesting nodes (clients) send out service request messages and servers listen to such messages at a predetermined network interface and port. If the requested service is supported, then a reply message is sent back to the client. The alternative scheme involves service brokers (or directory agents) which reside between clients and servers as a logical entity. Clients direct their requests to well-known service brokers whereas servers register their services with these brokers. In return, service brokers send back the service reply messages to the clients and registration acknowledgments to the servers. We will refer to each of these models as *directory-less* and *directory* architectures respectively.

It can be argued that the directory-less architecture is more suited to a MANET scenario [6], because there is no need for any infrastructure. Furthermore, network support for broadcasting, multicasting, or anycasting suffices to implement this client-server model. It is also clear that using service broker nodes necessitate either assigning this functionality statically

to certain nodes in the network or assigning it dynamically to a subset of nodes in the network depending on the current network topology. The first choice is not possible under the given constraints of MANETs. Since the latter choice presents an extra load to the network for selecting the server brokers while dynamically adapting to topology changes as well as informing the rest of the network about the identities of these *service broker nodes (SBNs)*, it can be seen as an expensive alternative and be discarded.

On the other hand, there are two very important motivation points that support the use of a directory system. The first motivation comes directly from the benefits of using SBNs: 1) Scalability is achieved when network size becomes larger. 2) Response time for locating services reduces. 3) Servers are not flooded with service requests when there is a high demand for certain type of services (e.g. inter-networking). 4) SBNs can apply simple load balancing techniques before sending back a reply message. This further reduces the load on individual servers and improves the service performance.

The second motivation comes from the recent research efforts in utilization of virtual backbones or clusters for improved efficiency as well as quality in MANET routing protocols (unicasting, multicasting, or broadcasting) and in the realization of distributed database systems again mainly used for serving routing protocols [7], [8], [9], [10]. It is highly desirable to have a tighter relation between different layers of the communication to reduce the redundancies associated with repeating similar tasks in different layers which results with increased control message overhead. Provided that a virtual backbone formation and maintenance mechanisms exist below the network layer, upper layer protocols, i.e. routing and service discovery algorithms, can exploit this backbone together. Then the overhead associated with dynamically selecting the SBNs is justified with the overall efficiency provided that SBNs are co-located with backbone nodes.

In this paper, our goal is to show that directory architecture is not only feasible, but also a good candidate for service discovery in MANETs. For that purpose, we provide our own solution to realize directory architecture and compare its performance against directory-less architectures that utilize multicasting and anycasting support from network layer.

Our solution involves two phases: (i) BackBone Management (BBM) phase: a dynamic virtual backbone is formed below the routing layer such that each node in the network is either a part of the backbone or one hop away from at least one of the backbone nodes. Each backbone node knows how to reach the other backbone nodes within 3-hop neighborhood after this phase is completed. SBNs are co-located with the backbone nodes and each non-backbone node is associated with at least one SBN. (ii) Distributed Service Discovery (DSD) phase: Service request and registration messages are used to form subtrees rooted at clients and the servers which allow efficient dissemination of the subsequent request and registration messages. Reverse paths in these subtrees are used for reply messages.

The rest of the paper is organized as follows. In section-II, an overview of possible network support options and desired features of a service discovery protocol are provided. Section-III gives the network model and the notation used. In section-IV, we present a detailed explanation of the BBM algorithm and the DSD algorithm. Performance measures, simulation framework and simulation results are introduced in section-V. Finally, in section-VI, we summarize and conclude this paper.

II. Overview

Service discovery protocols can be evaluated under a few different contexts. For us, the distinguishing characteristic of these protocols will be their applicability to a MANET environment.

The first condition is the infrastructure-less operation and it has a direct impact on the architectural choice of the service discovery protocols. Directory-less architecture does not have any directory agents (DAs), hence no infrastructure associated with them is needed. But directory architectures are not free of the infrastructure burden in a straight-forward manner. In directory architectures, first issue is about the nodes that can carry out the functions of DAs. If DAs are very specialized nodes having requirements of bridging between different communication media (e.g. *salutation manager (SLM)* in Salutation protocol [4]) or containing the objects to access the services (e.g. *lookup server* in JINI [5]), then these functions can be impossible or very costly to relocate to another node in the network without any user intervention. Therefore these DAs constitute an infrastructure contrary to MANET requirements. On the other hand if DAs simply contain records of services registered dynamically by the servers, then any node with enough resources (e.g. battery power, memory, processing power, etc.) can assume the responsibility of being a DA which is essential for infrastructure-less operation. Thus simplicity of DA functions is desired in MANETs. Then we need mechanisms to dynamically (s)elect the nodes that DAs will reside in and make their locations known to other nodes as the topology changes. But these mechanisms should be same in principle with forming clusters or virtual backbone in MANETs at the network level except for the tight coupling with service discovery in application layer [8], [11], [7], [10], [12], [9], [13]. We will exploit this observation in designing a directory-based service discovery architecture.

The second condition is the provisioning of efficient yet satisfactory network layer support for service discovery in a dynamic topology. When there are multiple servers (or DAs) and no direct link exists between clients and servers, this network support must be in the form of broadcasting, multicasting, or anycasting. Even for medium-size networks, broadcasting can have excessive control message overhead which is quite important in shared wireless channels and unintended nodes have to receive, process, and re-transmit these packets which wastes the network resources. Therefore we will not consider broadcasting in this work as a viable alternative.

In multicasting, server and DA nodes (depending on which architecture is being used) are assigned well-defined multicast addresses, so they can be reached by all client nodes. The formation and maintenance of multicast groups in terms of client, servers, and DAs can incur significant cost in network operations. Thus, even directory-less systems ,which are preferred because of their light-weight, may in fact turn out to be heavy-weight in the overall cost when multicasting is used. On the other hand, anycasting can provide a simpler framework, because in principle it can be simulated by any unicast routing protocol [14] which is clearly less demanding than multicasting. But in anycasting there is no differentiation among the attributes of the services. Service requests will be eventually received by only one of the servers which may not satisfy client request[1]. Clearly, assigning a different anycast address for each service type and attribute combination is not a feasible approach. Hence anycasting has a limited scope as compared to the cases where service attributes may exhibit a high variety and clients have preferences[2].

We have already pointed out the similarity between virtual backbone or cluster formation and implementing a directory architecture. Second condition basically imposes a control message distribution support on top of this virtual backbone with a close interaction with service discovery agents in the application layer. In the following sections, we unravel our algorithms to satisfy both conditions ,i.e. lack of infrastructure and efficient dissemination of service discovery control messages, in a directory architecture.

III. Network Model and Notation

In this section, we present the network model and the notation that is used throughout the paper.

A. Network Model

We assume that all the nodes in the network have an omni-directional antenna and have the same transmission power. All links are bi-directional, i.e. if node A can hear node B, then node B also can hear node A. Nodes share the same communication channel (e.g. same frequency band, same spreading code or frequency hopping pattern) to transmit and receive packets. Hence no node is allowed to transmit and receive at the same time. No particular assumption is made on medium access control (MAC) and access scheme can be random, reservation based, or any variant of both. Without loss of generality, partitioning in the network is not allowed, because each partition can be treated as an independent network.

B. Notation and Definitions

We use a color convention in determining the roles of each node in the network. SBNs are represented by *black* color. If a node is not part of the virtual backbone and it has at least one SBN (i.e. black) neighbor, then it is called to be *associated* with the virtual backbone and it is represented by *green* color. When a node is neither a SBN nor an associated node, then it is represented by *white* color. The rest of the notation and the definitions is as follows.

- N: Set of all the nodes in the network.
- $N_i^{(d)}$: Set of nodes that are at most *d* hops away from node i excluding node i itself.
- W: Set of white nodes in N.
- c_i: Color of node $i \in N$, which can be black, green, or white.
- VAP_i: Virtual Access Point (VAP) of green node i. This node is used by node i as its access point to the backbone. If node i is a server, it always registers its service with the DA residing on node VAP_i.
- d_i**,** dw_i: Degree information for node i, i.e. total number of neighbors (or *degree*) and total number of white neighbors (or *effective degree*) of node i in the given network topology.
- $NLFF_i$: Normalized link failure frequency of node i. This parameter represents the total number of link losses for node i in a fixed time window normalized by d_i at the end of the observation window.
- *nlff_th*: System threshold that sets the preferred level of normalized link losses for any backbone node.
- ID_i: Network identifier for node i.
- T_w**,** T_l**,** T_s**,** T_h: Waiting time, long time, short time, and hello beacon periods which are ordered as $T_w > T_l > T_s > T_h$.

We will use the terms SBN, VAP, and backbone nodes interchangeably throughout the paper.

IV. A Directory Architecture Solution for Service Discovery

Our network level solution to support a directory architecture consists of two parts. The first part, BBM phase, selects a subset of the network nodes to form a relatively stable *dominating set*, discovers the paths between dominating nodes and adapts to the topology changes by adding or removing network nodes into this dominating set. The formation algorithm for the dominating set is very similar to the *backbone selection phase* used in VDBP [7]. But, the way we incorporate the effect of link failures and we interconnect the VAP nodes are quite different. In VDBP, the node with minimum *NLFF* selects itself as a backbone node. Instead, we only eliminate the nodes with *NLFF* values higher than a given threshold and use the degree (or effective degree) as the selection criterion for the remaining nodes. Note that $NLFF_i$ is simply the proportion of the link losses at node i. Relying on a threshold value eliminates the extreme cases as seen in VDBP, i.e. a node i with no link losses but very few neighbors would be selected instead of a node j with very high degree but a few link losses. Unlike most of the other backbone or clustering algorithms, BBM utilizes only a 1-hop local broadcast control message (*Hello Beacon*) for forming the backbone, creating virtual links

[1]One may also consider sending application data directly to the anycast address reserved for a service type rather than first discovering the server location. But then, subsequent data packets will possibly be routed to different servers and still the server attributes are not distinguished.

[2]A client may request a color printer in a close location with low number of jobs queued. If all the printers are assigned a single anycast address, the user preference will be ignored in the service query.

between backbone nodes, and maintaining the backbone. Hello beacons are also light-weight, because they do not carry all the neighborhood information of the transmitting node.

After the first part is successfully carried out, we have a virtual backbone that constitutes a mesh structure with the backbone nodes and the virtual links connecting them. The second part is used to efficiently distribute the request and registration messages from the service discovery agents to the DAs (i.e. backbone nodes). These messages assist in forming multicast trees rooted at client and server nodes on top of the backbone mesh.

The detailed descriptions of both parts are provided in the subsequent sub-sections.

A. BackBone Management (BBM) Phase

The goal of the BBM algorithm is to obtain a small size (not necessarily minimum size) and relatively stable backbone. The algorithm is highly distributed and based on local decisions which makes it fast to react back to the changes in the network topology. BBM algorithm can be described in three components: (i) initial selection of backbone nodes, (ii) mesh formation by finding the paths between backbone nodes, (iii) and maintenance against topology changes. All the components rely on the periodically broadcasted *hello beacons* which bear the following information about the transmitting node i: $\{ID_i$, d_i, dw_i, $NLFF_i$, c_i, $VAP_i, flags, routing\ information\}$. Each node creates a *neighborhood information table (NIT)* and a *routing table* using the information carried by these beacons.

Initially, e.g. when first powered on, every node is assigned white color. Before deciding on their role in the network, white nodes collect hello messages and built up their own neighborhood information table (NIT) for time period T_w. After the waiting period is over, each node $k \in W$, which has $NLFF_k < nlff\_th$, joins the virtual backbone and becomes black if $dw_k > dw_l$ for each $l \in (N_k^{(1)} \bigcap W)$ such that $NLFF_l < nlff\_th$. Ties are broken by giving strict priority to lower ID node. Checking the normalized link loss threshold helps to avoid the nodes with a lot of link losses relative to their total number of links becoming backbone nodes. If no node in $\{k\} \bigcup N_k^{(1)}$ has a link loss rate lower than the threshold, then node k decides as if its nlff_th is set to ∞. Effective degree information is checked to force undecided subnets in the network to continue the process. If a node is still white and does not have the best effective degree among its white neighbors, it extends its waiting period to receive more hello messages. This extra waiting time must be in the order of hello beacon interval T_h. At any point in the waiting time period, if a white node k receives a hello message from a black node l, VAP_k becomes ID_l and c_k becomes *green*. If white node is left as the only white node in its neighborhood at the end of the waiting time, then this node must select a green node as its VAP node by giving strict priority to nodes that first satisfy the nlff_th requirement and secondly have a higher degree. When any green node i receives a hello beacon from node j with $VAP_j = ID_i$, node i must become a backbone node and turn into black.

Following lemmas show that all nodes decide in a finite amount of time, we end up with a dominating set, and each VAP node has other VAP nodes within 3-hop distance if the network radius is large enough.

Lemma 1: [Time-boundedness and Correctness of BBM phase]: Initial selection part of BBM terminates in a finite amount of time and the set of black nodes constitutes a dominating set under the assumptions that the network graph is connected, network size is bounded, hello beacons are transmitted error-free and frequency of hello beacons is faster than the topology change events.

Proof: Since network size is bounded, in a finite amount of time, all the nodes in the network must be powered on. Without loss of generality, suppose that at any time t_0, there is connected subgraph G^s where any node $i \in G^s$ is white, all nodes have already waited for time T_w, and $|G^s| > 1$. If we show that at least one node in G^s decides to be black in a finite amount of time τ, then $|G^s|$ becomes a monotonically decreasing function when observed at time instants $t_0 + k \times \tau$ until $|G^s|$ becomes one. But at this moment, we have an isolated white node which selects another green node as a VAP node and it turns into green. Since all the white nodes are exhausted, we end up with a set of black nodes which forms a dominating set.

Since all nodes have finished initial waiting time of T_w, all of them are in the extended waiting time period which is in the order of T_h. Thus in time interval $[t_0, t_0 + \alpha \times T_h)$, all the nodes must check their NIT to see if they are the best node. Let's choose $\alpha = 0.5$ and suppose a topology change occurred at $t_1 \in [t_0, t_0 + 0.5 \times T_h)$. Then the assumption about the topology changes ensures that in $[t_1, t_1 + T_h)$ topology remains same. But our selection of α enforces all the nodes to check their NIT table again in $[t_1, t_1 + T_h)$. Our criterion for selecting the best node guarantees that there is a unique best node at any time instant. Because the best node remains same in $[t_1, t_1 + T_h)$, it decides to be black in that time interval. Thus, choosing $\tau = T_h$ suffices for completing the proof. ■

Lemma 2: Assuming that network graph is connected and the maximum distance in that graph (i.e. network radius) is greater than or equal to 3 hops, there exists a black node i for each black node j such that $i \neq j$ and $i \in N_j^{(3)}$ after the selection part of BBM.

Proof: We will prove the lemma by using the way of contradiction. Let's assume that there is not any black node $i \in N_j^{(3)}$. Because of the assumptions of the lemma, there is a green node k which is 2 hops away from node j. Since there is no black node in 3 hops of node j, green node k can not have black neighbors either. This is a contradiction to the definition of a green node. Hence, the lemma follows. ■

Lemma-1 and lemma-2 provide the necessary framework for completing the backbone formation. If each VAP node discovers the paths to other VAP nodes within 3 hops (i.e. *VAP neighbors*), then we obtain a mesh structure which will later be utilized in distributing control messages between VAP nodes.

Hello beacons convey enough information for finding paths

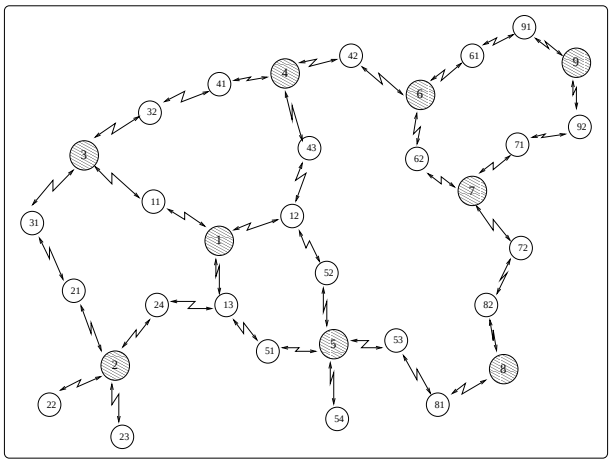

Fig. 1. An instance of virtual backbone formation.

between VAP neighbors. We will refer these paths as *virtual links*. A virtual link can be 2-hop or 3-hop long, i.e. there may be one or two green nodes between VAP neighbors respectively. Both situations are outlined in fig.1. In the figure, dashed nodes are our black nodes, and non-dashed smaller size nodes represent the green nodes. Black nodes are identified by single digit numbers, whereas green nodes are identified by two digit numbers with most significant digit indicating their VAP node. For instance of a 2-hop virtual link, we see green node 11 between black nodes 1 and 3. Node 3 sees in the hello beacons of node 11 that node 11 has a different VAP node than itself. Since node 11 also sees that node 3 is black and it is not its VAP node, it includes this routing information in its Hello beacons. Thus both black nodes 1 and 3 have the information that they can reach to each other via node 11 and they update their routing tables. When there exist 2 green nodes along the virtual link, the situation is slightly different. For example, we have two green nodes between black nodes 1 and 2. Nodes 13 and 24 recognize from each other's hello beacons that they have different VAP nodes. Therefore, node 13 caches node 24 as next hop for node 2 and node 24 caches node 13 as next hop for node 1. They also include this routing information as an extension in their Hello beacons, so that nodes 1 and 2 will know that nodes 13 and 24 are next hop nodes respectively towards each other. Hence, green nodes play the major role in discovering the virtual links between VAP neighbors. To reduce the size of hello beacons, routing extensions that carry no new information are avoided for a time-out period.

The maintenance of the dominating set feature of the backbone is a very important task against frequent topology changes. If green nodes do not receive hello messages for a time period from their VAP nodes, they choose another neighbor as their new VAP node by giving strict priority to black nodes and then green nodes that conform to the mobility threshold and highest degree criterion. Thus no node is left without a VAP node. On the other hand, a black node can migrate to a location where none of the green nodes have this node as their VAP node. Therefore, when no hello message indicating itself as a VAP node is received for a time period T_l, a black node must turn into a green node and follow the same actions that a green node left without a VAP node takes. Because of mobility, black nodes can be grouped together in the same locality. To resolve such cases, if a black node i has other black nodes in its NIT, it transmits its hello message with a flag indicating it will change its color to green. When green neighbors which has node i as their VAP node receive this hello message, they compute the best black neighbor from their own NIT. If the best black neighbor is not node i, they simply assign the best black neighbor as their new VAP node. Otherwise they set the flag in their own hello messages indicating i as the best node. As long as black node i receives hello messages from its green neighbors indicating i as the best VAP node, node i remains black. If no such messages are received for a time period T_s, then node i turns into green and leave the backbone.

B. Distributed Service Discovery (DSD) Phase

Now, we have the virtual backbone and DAs are co-located with the VAP nodes. But we still need mechanisms to let servers register their services with one or more DAs and clients request the services. This is done in the following fashion. When a server located on node i wants to register its service, it has to register with the DA located on VAP_i assuming node i is a non-backbone node. VAP_i is referred as *source VAP node*. If the node i is already a black node, then the service is registered with the DA on the same node and node i itself becomes the source VAP node. Server may register with more DAs (even maybe with all DAs). Then we need a multicast or broadcast mechanism to distribute the registration messages to other DAs located on other VAP nodes. Any time the VAP node of a server changes, it must renew its registration with the DA operating on the new VAP node. Also, the server should be able to keep the scope of its registration messages local by bounding the number of black nodes the registration messages could traverse. Similar arguments are true for the service request. When a client on node j requests for a service, node j forwards the request to VAP_j provided node j is not already a black node and VAP_j passes the request to co-located DA. If node j is black, then the request is passed to the DA on node j. In case DAs do not have any fresh registration for the service, the service should be requested from other DAs again by multicasting or broadcasting.

Wireless bandwidth is scarce because of the shared medium and wireless channel impairments. Although backbone itself helps to reduce the overhead in disseminating broadcast or multicast messages by using simple mechanisms like flooding the backbone, it is not sufficient when we consider the increasing frequency of multicast events and the topologies where backbone with virtual links exhibits lots of loops and high average degree. To make things simple, scalable and efficient, we propose a source based multicast tree algorithm which is triggered by service discovery request and registration messages sent to the backbone management layer by clients and servers.

In our algorithm, every backbone node keeps a forwarding list among their VAP neighbors for each multicast tree uniquely identified by the source VAP node. As initial condition, forwarding lists include all of the VAP neighbors except for the source VAP node. Multicast messages contain the following fields: {*source node, source VAP node, sequence number, last-hop VAP node, next-hop VAP node, next hop node, time-to-live (TTL), options, payload data*}. *Source node* indicates the client or the server which initiated the request or registration process. Each multicast message is uniquely identified by the 2-tuple {*source VAP node, sequence number*}. Multicast messages flow from last-hop VAP node to next-hop VAP node. Last hop VAP node and the green nodes along the virtual link compute the *next hop node* from their routing tables using *next-hop VAP node* as the destination point. Note that these routing tables are generated and updated by BBM phase. When *next-hop VAP node* receives the message, it prunes the *last-hop VAP node* from the forwarding list of the particular tree. If the message is received for the first time the replicas are sent to each node in the forwarding list. When a duplicate multicast message is received from a pruned VAP node to which the same message has not been sent, an explicit PRUNE message must be sent to that VAP node to force it to prune the same link. This algorithm guarantees a multicast tree after a convergence time given that topology changes slower than the convergence time. In the worst case situation, multicasting to backbone nodes will be same as flooding among the backbone nodes. Options field basically defines the type of the encapsulated payload data, e.g. service registration, service request, etc. Depending on the options field, a VAP node waits for a feedback from upper layer protocols, where the payload data is handled, in order to stop or proceed with the forwarding operation. TTL field can be used to further limit the depth of forwarding for a particular multicast message (i.e. the information is explicitly kept local).

Forwarding list members basically can be referred as child nodes, and the VAP node from which a multicast message is received for the first time can be referred as parent node. When a VAP node loses its parent node, it sends an UNPRUNE message to its VAP neighbors. Child nodes upon receiving an UNPRUNE message from their parent, generates their own UNPRUNE message and send it to other VAP neighbors except for their parent node. All VAP nodes which receive an UNPRUNE message add the sender to the forwarding list of the particular multicast tree as specified in the UNPRUNE message.

To give more insight about the algorithm, we provide two examples in fig.2 and fig.3. These figures essentially show the same topology as in fig.1 except for the fact that the links and green nodes between VAP nodes are replaced by bidirectional dashed lines as virtual links. Suppose server on node 12 wants to register its service to all VAP nodes (nodes 1 to 9 in the figure). Node 12 sends the registration message to its VAP node 1. Then node 1 initiates multicasting process by first unicasting the copies of the registration message to its VAP neighbors. Nodes 2, 3, 4, and 5 receive a multicast message

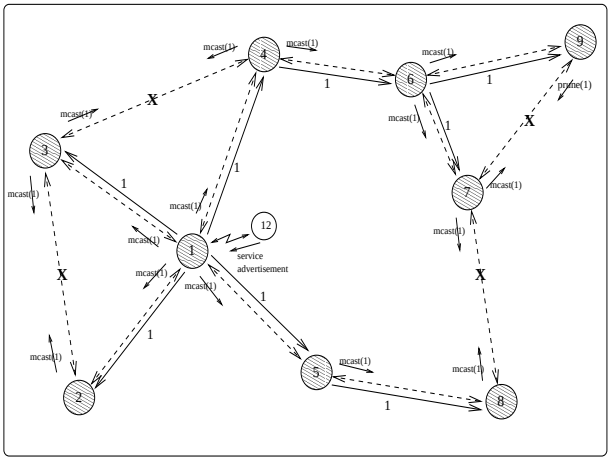

Fig. 2. Source based multicast tree formation on virtual backbone.

originated by node 1 for the first time. So they forward the copies of the message to their own VAP neighbors except for the node from which that message is received and the originator of the message. In the figure, node 2 will receive the same multicast message from node 3 as it received from node 1. Thus node 2 knows at that moment that node 3 has already received the same multicast message. As a result node 2 stops forwarding multicast messages originated from node 1 to node 3. Similarly node 3 will see duplicates via node 2 and 4, so it stops forwarding multicast messages originated from node 1 to these nodes. A different case will happen at node 9. Node 9 first receives the multicast message via 6 and then via 7 at almost the same time. The message from node 7 is duplicate, and since node 7 is pruned from the forwarding list of node 9, node 9 explicitly sends a PRUNE message to node 7. Hence node 7 stops forwarding multicast messages originated from node 1. Explicit PRUNE mechanism is also used when duplicates are received persistently from a pruned node. This can happen in cases where multicast or PRUNE messages are lost. At the end a multicast tree is formed on top of the virtual backbone. The source VAP node, i.e. node 1, becomes the root of this multicast tree. In fig.2, directed solid lines labeled with node 1 represents this tree. Dashed lines with a cross represent the pruned links. If node 13 also wants to initiate a service registration or request, since it has the same VAP node as 12, it uses the same multicast tree represented by source VAP node 1. Note that the multicast tree can be logically viewed as two separate trees rooted at nodes 12 and 13.

Fig.3 shows a general scenario when TTL field is set to 2 hops for multicasting service registrations from node 32. TTL field includes the link from node 32 to its VAP node 3. The solid one directional links show the multicast tree and the numbers on them indicate the root of the tree. Suppose this service is only provided by node 32 and node 91 wants to utilize it. Then node 91 sends a service request message to its VAP node 9 and request is multicasted until it reaches nodes 1 and 4 which already have the information. Therefore,

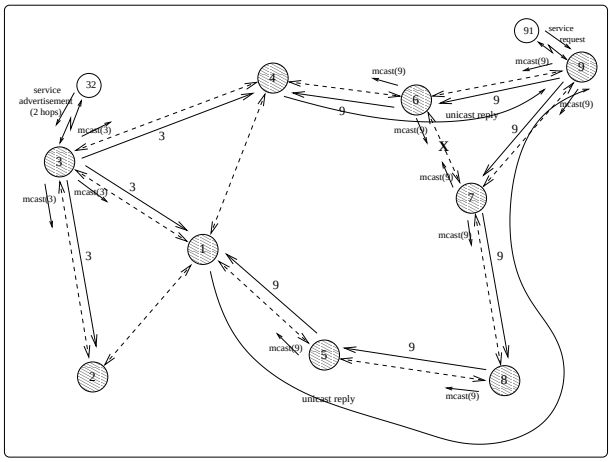

Fig. 3. General view of resource discovery.

nodes 1 and 4 stops forwarding the request message and using reverse link information, they reply back to node 9 and node 9 replies back to node 91. Here, it is important to note the interaction between service discovery agent and multicasting in forwarding decisions. A service request is propagated further unless query is resolved.

V. Simulation Environment

In this section, we first describe the performance metrics that are used to evaluate different architectural and network support choices. Then the simulation framework and the simulation results are presented as two separate sub-sections.

A. Performance Metrics

Three performance criteria are considered in our simulations. The first performance metric is the total mean *control message overhead* of each service discovery mechanism which measures the load of the algorithms on network resources in terms of the number of packets. The second performance metric is the mean *hit ratio* of these mechanisms. In the generic service discovery algorithm used in our simulations, a client does not repeat the request until it receives a successful reply. This is simply because we want to see how many original requests are successfully replied, and we label these requests as successful attempts. Hit ratio is simply the ratio of the total number of successful attempts to the total number of requests. When hit ratio and control message overhead combined together, it reflects the efficiency of each approach. Our last performance metric is the *average time delay* between the time any successful request is sent from a client and the time corresponding reply is received by the same client. This metric measures the promptness of the service discovery and it is particularly important when we have real time applications waiting for timely response for each service query.

B. Simulation Model

We simulated four different service discovery mechanisms using ns-2 with CMU wireless extensions [15]. One of these mechanisms is our proposal for the directory architecture, and other three mechanisms are based on directory-less architectures. As network support, we considered multicasting and anycasting as two major contenders for the directory-less architecture.

There exists a rich literature on multicasting for mobile ad hoc networks [16], [17], [18], [19], [20], [21], [22]. In our scenarios, we considered multiple senders (i.e. clients) and multiple receivers (i.e. servers). We did not include the case, where servers advertise their services and clients learn about the services passively. Instead only the request-reply and registration mechanisms are considered which cover the majority of the applications. Since clients issue the service requests at will, the multicast protocol should be sender based rather than receiver based. We also want to have a multicast protocol which does not depend on any particular unicast routing protocol which restricts its use. When these choices are considered together, *on demand multicast routing protocol (ODMRP)* [16] fits quite well to the features that we seek for. We implemented ODMRP in ns-2 without using its mobility adaptive part which requires GPS receivers on mobile devices.

We implemented anycasting by modifying two very popular ad hoc unicast routing protocols already implemented in ns-2, namely DSR [23] and AODV [20], by defining a virtual node that only server nodes have routing entries as suggested in [14]. We will refer these two modified algorithms as *anycast-DSR* and *anycast-AODV* respectively. The choice of these protocols also stems from the fact that they are reactive algorithms and quite efficient in terms of control message overhead. Both of them support similar mechanisms like flooding route request messages and obtaining replies, and the main difference is in the way the routing entries are created, cached, and maintained.

For our proposal, BBM and DSD are implemented below the routing and above the link layer with direct interfaces with the service discovery protocol again in ns-2. The TTL field for service registration messages are kept fixed at 1, so each servers registers with only one DA. We will refer the overall proposal as *distributed service discovery protocol (DSDP)* in simulation results.

We use the distributed coordination function (DCF) of IEEE 802.11 as the underlying MAC protocol. DCF in IEEE 802.11 is a random access scheme and belongs to CSMA/CA family. The radio interface is based on Lucent's WaveLan technology with 250 meters of nominal propagation range and 2 Mbps of nominal bit rate. Radios use omni-directional antennas and we assume a two-ray ground propagation model. The network size is fixed to 50 nodes. Both square (1000mx1000m) and rectangular (1500mx300m) topologies are considered, but since the relative results are very similar to each other, we only provide the results for the rectangular topology. We use random-way point model as the mobility model. In this model, nodes select a random destination point and a speed value from a uniform distribution $U[0, V_{max}]$ after a pause time P. When they reach the destination, they repeat the same process. A higher V_{max} or a lower P corresponds to a higher mobility level. Five different mobility patterns are used

for each $\{V_{max}, P\}$ pair. In the experiments, where V_{max} is varied, P value is kept at 0 sec.,i.e. nodes are always in motion. When we vary the P value, we had fixed the V_{max} value at 20 m/sec.

Besides mobility, other important parameters to play with are the number of clients and the number of the servers. Number of clients is selected as 10, 20, and 30, whereas number of servers is varied between 1, 3, and 5. For each mobility pattern, again five different random set of clients and servers are used. Thus, each point in the simulation plots are averaged over 25 random scenarios. Each server periodically registers its service every 10 seconds and whenever its VAP node changes for the directory architecture. Without any loss of generality, we assumed that there exists only one service class which is offered by all the servers in the network. Clients, on the other hand, send their requests such that the inter-arrival time is a random process $\zeta = T_0 + \tau$ where T_0 is deterministic time set to 6 seconds and τ is exponential random variable with mean 2 seconds.

C. Simulation Results

In the first set of experiments, we try to capture the effect of the number of servers when the number of users is kept constant. Control message overheads of on-demand anycast protocols are found to be very sensitive to the number of servers[3]. ODMRP tends to have more overhead while other protocols have a lower overhead as number of servers increase. This is not an unusual outcome considering the main mechanisms of these approaches. In multicasting, all the servers receive the requests, and then all of them have to reply back. But in anycasting only one of the servers receives the message regardless of the number of servers. Since it is highly likely that the closer server replies back and the average shortest distance between clients and servers gets smaller with more servers in the network, the control message overhead decreases. Higher number of servers also decrease the depth of the query trees in DSDP, hence we have a slight reduction in control overhead.

End to end delays for successful service discovery improve with the increasing number of servers. This result is expected again due to the fact that the average distance between users and servers decrease as the number of servers increases. Anycast-DSR and ODMRP show more rapid improvements, however anycast-AODV and DSDP offer consistently lower delays. Delay performance of ODMRP reaches to that of anycast-AODV and DSDP, but anycast-DSR can not compete in terms of delay.

The hit ratio also improves with number of servers. Anycast-AODV performs inferior compared to other protocols. ODMRP has consistently the best hit ratio. DSDP outperforms anycast-DSR more significantly as mobility level and/or number of servers increase. Note that all performance metrics become worse as mobility in the network increases, because the link failures occur more often.

[3]In the plots, arrow directions indicate the increasing number of servers or users.

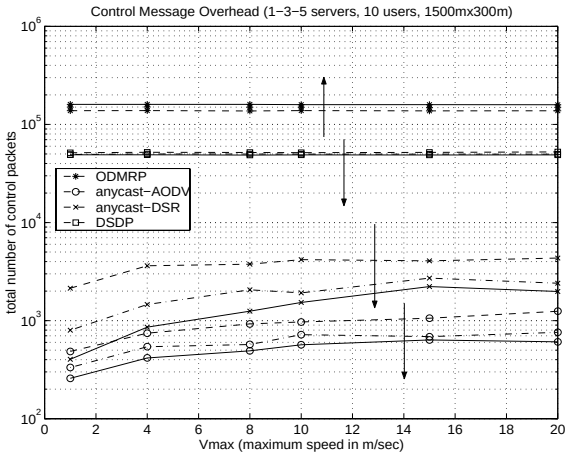

Fig. 4. Control message overhead for 10-user case.

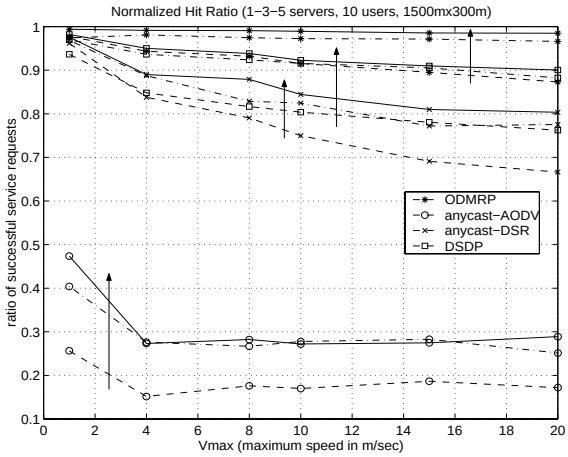

Fig. 5. Ratio of successful requests for 10-user case.

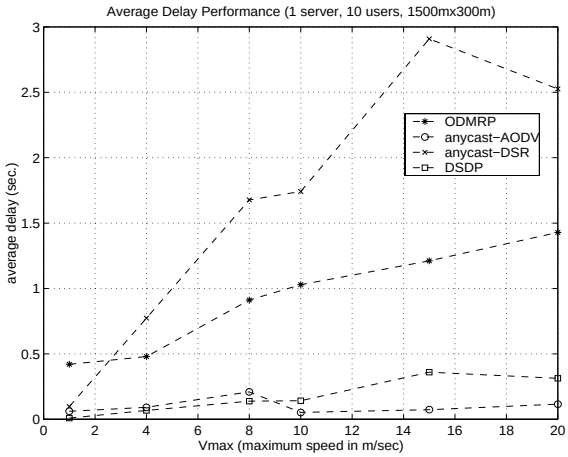

Fig. 6. Average delay comparison for 1-server/10-user case.

The second set of simulations address the question how increased load in terms of number of users affects the performance of the protocols. The results are plotted in figures 7, 9, 10, 11, 12, 13, 14, and 15. The number of servers are kept

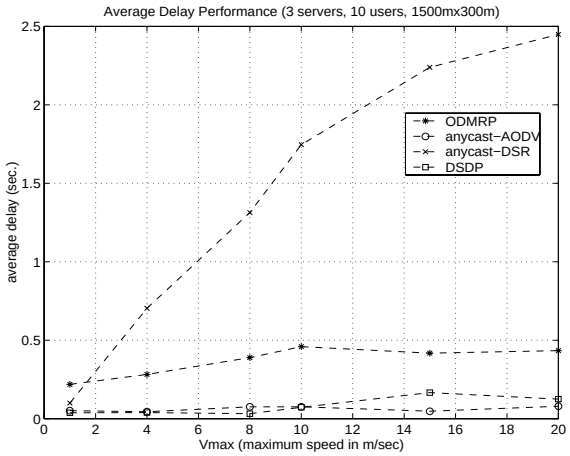

Fig. 7. Average delay comparison for 3-server/10-user case.

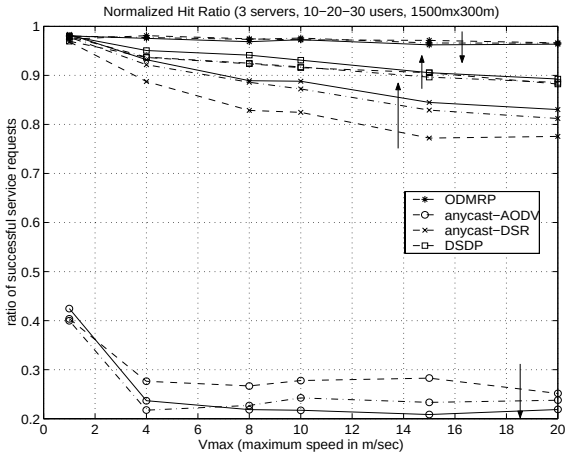

Fig. 10. Ratio of successful requests for 3-server case.

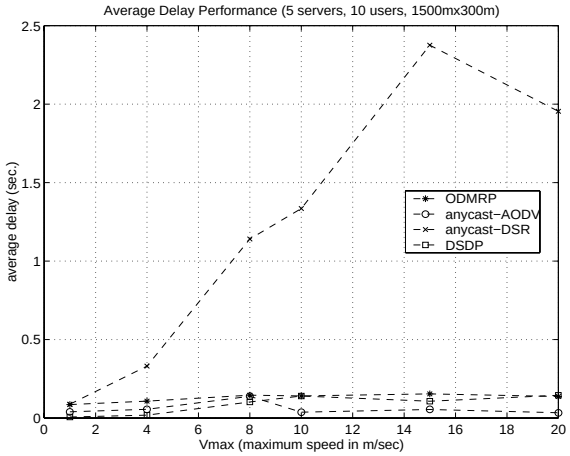

Fig. 8. Average delay comparison for 5-server/10-user case.

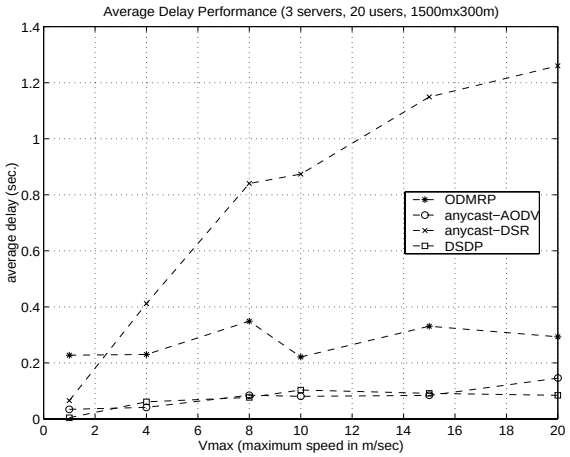

Fig. 11. Average delay comparison for 3-server/20-user case.

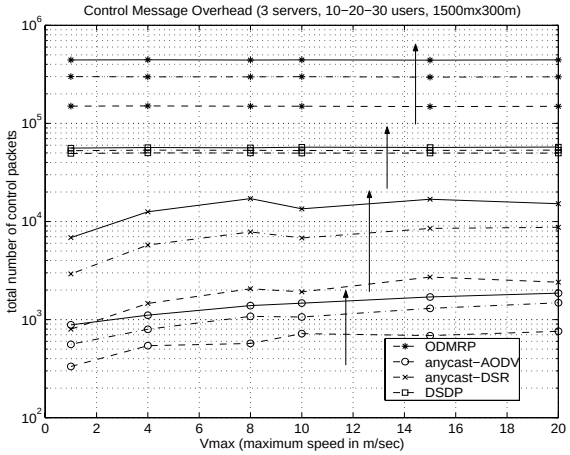

Fig. 9. Control message overhead for 3-server case.

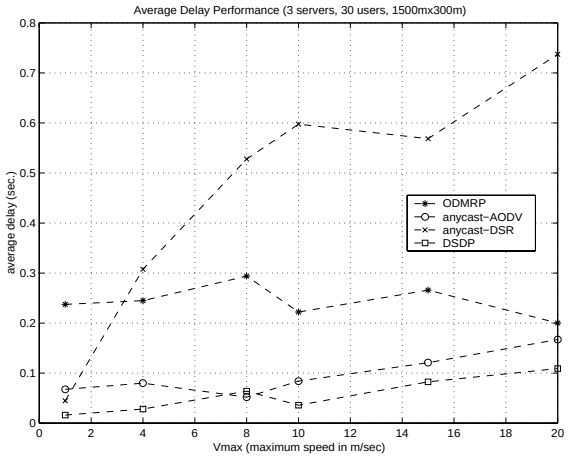

Fig. 12. Average delay comparison for 3-server/30-user case.

as three and number of users varied between 10 and 30.

The overhead of each protocol increases with the number of users with DSDP being the least sensitive one. Although ODMRP is a heavy-weight protocol, it is quite sensitive to the increased number of users rather than the servers. This result is a natural consequence of the facts that ODMRP is a sender-based multicast scheme and senders broadcast periodic refresh messages to maintain the multicast group.

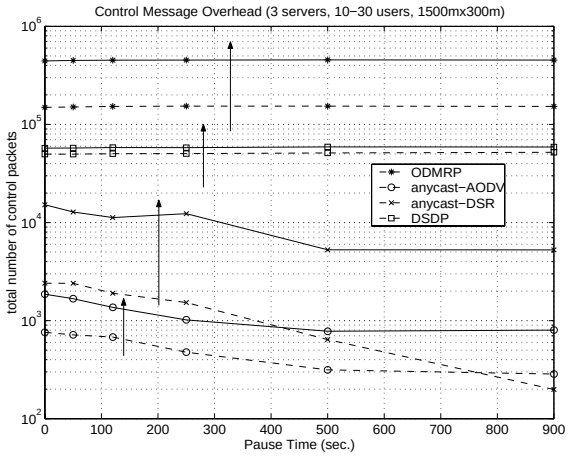

Fig. 13. Control message overhead under variable pause time.

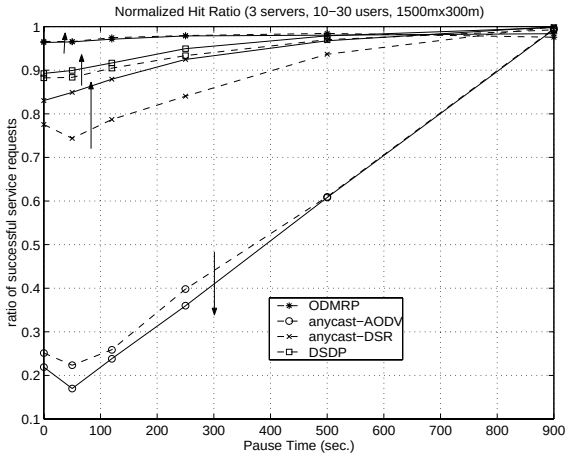

Fig. 14. Ratio of successful requests under variable pause time.

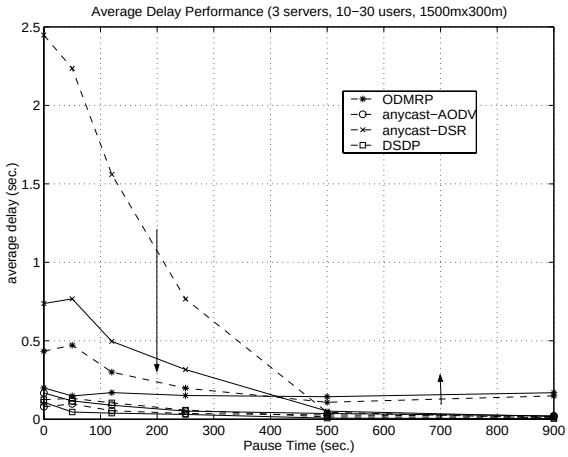

Fig. 15. Average delay under variable pause time.

The delay of each protocol tends to decrease as number of users increase. At first sight, this can be regarded as a counter-intuitive result. Because an increase in number of users create a higher load on the network which may result in congestion and packet collisions, hence deteriorating delay values. But increasing number of users can have the effect of reducing the average distance from the servers, therefore it can improve the end-to-end delays. It also helps to on-demand anycast algorithms to discover the routes in advance when users share more common links. For networks operating below its capacity, these positive effects suffice to make delay values better.

ODMRP and DSDP does not show much response to the change in number of users in terms of successful service requests. Anycast-DSR shows significant improvements at higher mobility cases whereas anycast-AODV suffers from further performance loss.

To summarize all the results: 1) Under various mobility scenarios, number of users and servers, the relative performances of the protocols remain same in general for a fixed number of nodes and persistent service requests. 2) In terms of overhead, ODMRP is the most heavy-weight protocol and on-demand anycast protocols are the most light-weight. An increase in number of users almost linearly effect the message overhead of ODMRP, because it is a sender-driven multicast protocol. On the other hand, DSDP is not as sensitive as other protocols against mobility, number of servers or number of clients, since the bulk of the message overhead for DSDP is generated by periodically transmitted hello beacons. In our simulations, each node transmits a hello beacon every 1 second. Considering that we run simulations with 50 nodes and for 900 seconds, the overall cost of these beacons amounts to 45,000 packets, i.e. 75% to 94% of the overall overhead. 3) In terms of hit ratio anycast-AODV performs poorly except for almost stationary scenarios and therefore can not be a good candidate in general even though the delay and overhead performances are much better than ODMRP and anycast-DSR. ODMRP has consistently the best hit ratio performance as much as 18% above the next best protocol DSDP at very mobile environments. Anycast-DSR catches ODMRP and DSDP for low mobility cases, but the performance difference goes up to more than 10% with second best DSDP. 4) Delay performance of DSDP is consistently better than other choices. This is due to the fact that lower number of nodes are involved in search queries and because registration of services makes the average distance in number of hops shorter. 5) Although we have not presented any results in the paper, we also tested the performance of DSDP for TTL values higher than 1 hop. Depending on the number of servers and receivers as well as the frequency of service requests and registration, higher TTL value may have more positive or negative impact on overhead. In high mobility conditions, since servers have to re-register their services when they change their VAP nodes, higher TTL value turns out to be costlier. But it certainly increases the promptness in service discovery and the hit ratio.

VI. Conclusion

In this paper, we presented possible architectural and network support choices for service discovery in mobile ad hoc

networks. We provided our original service discovery mechanism to support directory architecture. We also implemented ODMRP, anycast-DSR, and anycast-AODV protocols along with our proposal to compare (i) directory and directory-less architectures and (ii) to compare different network support options, i.e. anycasting vs. multicasting. We used control message overhead, mean success rate and the average delay as three performance metrics for comparison purposes. We changed the parameters such as number of clients, number of users, and the mobility level for a comprehensive analysis.

It is the general idea that since no cost of selecting and maintaining DAs are involved, a directory-less architecture would be the least expensive and easiest to implement in a mobile ad hoc network. This view does not take into account the operational costs of the lower layer support required for such an implementation. Our results indicate that if the required network support is multicasting, then maintaining multicast trees can be very expensive in terms of control message overhead. Hence overall cost of directory-less architecture with multicast support requirement can in fact be more than that of the directory architectures. On the other hand, if anycasting is used as a network support, we can have a very light-weight directory-less service discovery. However this reward comes at the expense of significantly reduced performance in terms of average hit ratio. Even the level of hit ratio may drop to unacceptable levels as seen in our simulation scenarios with anycast-AODV. Anycast-DSR shows a more competitive level in terms of hit ratio, but then the mean delay values are compromised a lot even under mild mobility conditions. These problems are put aside, the main restriction for anycast support arises from the fact that it can only be utilized in a limited number of service classes. Therefore multicast support displays a more robust, reliable, and general framework for directory-less service discovery architectures.

Results also reveal that directory architecture supported by a virtual backbone structure can perform quite well under various mobility conditions in addition to its inherent advantages, e.g. resource allocation, load balancing, localization, etc. The most dominant figures are observed in the average delay performances. Our proposal consistently have the best delay values as well as achieving very competitive mean hit ratio values relative to the best values obtained by ODMRP in directory-less service discovery. Also the performance results show relatively very small sensitivity against mobility, load on the network, and number of servers. This suggests that DSDP is not only feasible, but also a very good candidate for real-time service discovery scenarios where a prompt and low jitter response is essential. Although virtual backbone approach is not as light-weight as anycasting solutions in terms of message complexity, when backbone is exploited by multiple stack of higher level protocols and light-weight hello messages are piggybacked behind data packets or other layer control beacons, the overhead of forming and maintaining the backbone can be quite justified. Thus, contrary to the general view, we demonstrate in this paper that directory architecture is a compelling solution especially for medium size MANETs.

In this paper, we have not investigated the effects of several load balancing and resource allocation techniques on the loads of the servers and on the system performance, e.g. delay and throughput, when a directory architecture is utilized. This will be our immediate future research interest. As a future work, we also want to develop a full stack of routing protocols which rely on our virtual backbone proposal.

REFERENCES

[1] E. Guttman, C. Perkins, J. Veizades, and M. Day, "Service Location Protocol, Version 2," *IETF RFC 2608*, June 1999.

[2] "Specification of the Bluetooth System," *www.bluetooth.com*, Dec. 1999.

[3] Microsoft Corporation, "Universal Plug and Play: Background, www.upnp.org/resources/UPnPbkgnd.htm," 1999.

[4] Salutation Consortium, "Salutation Architecture Specification, www.salutation.org/specordr.htm," 1999.

[5] Sun Microsystems, JINI Architecture Specification, Nov. 1999.

[6] Michel Barbeau, "Service Discovery Protocols for Ad Hoc Networking," *CASCON 2000 Workshop on ad hoc communications*, 2000.

[7] Ulaş C. Kozat, G. Kondylis, and M.K. Marina, "Virtual Dynamic Backbone for Mobile Ad hoc Networks," in *Proc. IEEE ICC-2001*, June 2001.

[8] Raghupathy Sivakumar, Prasun Sinha, and Vaduvur Bharghavan, "CEDAR: A core-extraction distributed ad hoc routing algorithm," *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 8, pp. 1454–1465, Aug. 1999.

[9] R. Ramanathan and M. Steenstrup, "Hierarchically-organized, multihop mobile wireless networks for quality-of-service support," *Mobile Networks and Applications*, vol. 3, no. 1, pp. 101–119, 1998.

[10] Ben Liang and Zygmunt J. Haas, "Virtual Backbone Generation and Maintenance in Ad Hoc Network Mobility Management," in *IEEE INFOCOM 2000*, March 2000.

[11] Raghupathy Sivakumar, Bevan Das, and Vaduvur Bharghavan, "Spine routing in ad hoc networks," *Cluster Computing 1*, pp. 237–248, 1998.

[12] C.R. Lin and M. Gerla, "Adaptive Clustering for mobile wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 15, no. 7, pp. 1265–1275, Sep. 1997.

[13] M. Chatterjee, S.K. Das, and D. Turgut, "An on-demand weighted clustering algorithm (WCA) for ad hoc networks," in *Proceedings of IEEE Globecom 2000*, Nov. 2000, pp. 1967–1701.

[14] V. Park and J. Macker, "Anycast Routing for Mobile Services," in *Proc. Conference on Information Sciences and Systems (CISS) '99*, Jan. 1999.

[15] Kevin Fall and Kannan Varadhan (Eds.), "ns notes and documentation," available from http://www.isi.edu/nsnam/ns/.

[16] S-J. Lee, W. Su, and M. Gerla, "On-Demand Multicast Routing Protocol (ODMRP) for Ad Hoc Networks," *Internet Draft, draft-ietf-manet-odmrp-02.txt*, Jan. 2000.

[17] C-C. Chiang, Mario Gerla, and L. Zhang, "Forwarding Group Multicast Protocol (FGMP) for Multihop, Mobile Wireless Networks," *Cluster Computing*, 1998.

[18] M.S. Corson and S.G. Batsell, "A Reservation-Based Multicast (RBM) Routing Protocol for Mobile Networks: Initial Route Construction phase," *ACM/Baltzer Wireless Networks*, pp. 427–450, 1995.

[19] J.J. Garcia-Luna-Aceves and E. Madruga, "The Core-Assisted Mesh Protocol," *IEEE Journal on Selected Areas in Communications*, Aug. 1999.

[20] C.E. Perkins, E.M. Royer, and S.R. Das, "Ad Hoc On-Demand Distance Vector (AODV) Routing," *Internet Draft, draft-ietf-manet-aodv-05.txt*, 2000.

[21] P. Sinha, R. Sivakumar, and V. Bharghavan, "MCEDAR: Multicast Core-Extraction Distributed Ad hoc Routing," in *IEEE Wireless Communications and Networking Conference*, Sep. 1999.

[22] C.W. Wu and Y.C. Tay, "Ad hoc Multicast Routing Protocol utilizing Increasing id-numbers," *Internet Draft, draft-ietf-manet-amris-spec-00.txt*, Nov. 1998.

[23] David B. Johnson et al., "The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks," *Internet Draft, draft-ietf-manet-dsr-05.txt*, March 2001.