Service*: Distributed Service Advertisement for Multi-Service, Multi-Hop MANET Environments

A. Nedos, K. Singh, S. Clarke

Distributed Systems Group
Department of Computer Science
Trinity College Dublin
Ireland
{anedos,singhk,sclarke}@cs.tcd.ie

Contact details of the first author:

A. Nedos
Distributed Systems Group
Department of Computer Science
Trinity College Dublin
Ireland

Tel.: +353 1 608 2354
Fax.: +353 1 677 2204
Email: anedos@cs.tcd.ie

Abstract
We consider the problem of service advertisement when multiple services are available in a MANET. We present *Service**, a novel distributed service advertisement protocol that uses a push-based mechanism to optimally place services in a dynamically selected subset of available nodes, called brokers. Such a dynamic arrangement of broker nodes is shown to have good scalability characteristics as the number of services and nodes increase and enables discovery queries to be satisfied with high probability using only single-hop broadcasts. The approach reduces message overhead by eliminating network-wide packet dissemination while maintaining a high service discovery ratio. The protocol is distributed in that it does not require the use of any centralised infrastructure. Instead, nodes initiating an advertising session use local rules to select a minimal subset of their 1-hop neighbours and delegate to them the task of acting as service brokers. We describe the protocol and present simulation results that exhibit a high discovery ratio with only a fractional overhead as compared to two other advertising protocols that utilise a pure and an optimised flooding approach.

Keywords
MANETs, Advertisement Protocols, Service Discovery, Localised Algorithms, Service Oriented Applications

Service*: Distributed Service Advertisement for Multi-Service, Multi-Hop MANET Environments

Andronikos Nedos, Kulpreet Singh, Siobhán Clarke
Distributed Systems Group
Trinity College Dublin
Ireland
Email: {anedos,singhk,sclarke}@cs.tcd.ie

***Abstract*— We consider the problem of service advertisement when multiple services are available in a MANET. We present *Service**, a novel distributed service advertisement protocol that uses a push-based mechanism to optimally place services in a dynamically selected subset of available nodes, called brokers. Such a dynamic arrangement of broker nodes is shown to have good scalability characteristics as the number of services and nodes increase and enables discovery queries to be satisfied with high probability using only single-hop broadcasts. The approach reduces message overhead by eliminating network-wide packet dissemination while maintaining a high service discovery ratio. The protocol is distributed in that it does not require the use of any centralised infrastructure. Instead, nodes initiating an advertising session use local rules to select a minimal subset of their 1-hop neighbours and delegate to them the task of acting as service brokers. We describe the protocol and present simulation results that exhibit a high discovery ratio with only a fractional overhead as compared to two other advertising protocols that utilise a pure and an optimised flooding approach.**

***Index Terms*— MANETs, Advertisement Protocols, Service Discovery, Localised Algorithms, Service Oriented Applications**

I. Introduction

Service-oriented applications emphasise the principles of modular design coupled with on-demand discovery and usage of network services. Such applications are normally composed from a set of components that can be either service producers or consumers. Developing these network-centric, service-oriented applications in MANETs is an intrinsically difficult task mainly due to mobility and lack of global knowledge that contribute to a highly unpredictable network environment.

Some of the problems associated with building service-oriented applications are neither new nor specific to MANETs. Web Services [1], the most popular manifestation of service-oriented applications, facilitate application interoperability with a modular and platform-agnostic design. Though the focus of Web Services is on modularity and XML-based specifications, we believe that its basic tenet of *advertise*, *discover* and *use* is a good model for enabling applications to share functionality in ad hoc networks. So far, work on supporting service oriented architectures in mobile ad hoc networks has either been placed at a very high layer, overlooking and abstracting the dynamic nature of the network domain or concentrated very closely at the protocols of the routing layer trading generality for efficiency.

The Service* protocol is based on the premise that applications for ad hoc networks are composed from a number of interdependent services. A service is defined in the broadest sense as a piece of modular, reusable code that has a light-weight interface which is advertisable. Such a decoupling of functionality amongst failure-prone, mobile network nodes, makes timely and efficient discovery of services a challenging yet salient task. This paper proposes the use of a localised, push-based advertising scheme, that is built around the concept of selecting and maintaining a dynamically changing subset of available nodes as service brokers. Brokers are nodes that are either actual service providers or were delegated the task of responding to discovery queries by some other node that in turn can be either a provider or a broker. The cost of maintaining a changing subset of nodes as brokers is amortised by the following advantages:

- Broker selection is localised. The changing set of brokers is maintained in a distributed manner by executing a local protocol and not by the formation of a virtual backbone. This avoids maintenance overhead associated with backbone mechanisms, such as multicast, while also having the property of being less dependent on mobility patterns.
- Ease of discovery. Discovery can be reduced to the task of a single broadcast. Simulations indicate a high probability that some neighbouring node will contain the wanted service.
- Service aggregation. The protocol, through its local interactions, provides the property of eventually aggregating all available services circulating in a multi-service network.

Since the protocol does not require the use of any multicast facilities or additions to the routing layer, it can be implemented at the application layer as the first tier for providing a service middleware platform in MANETs.

Section II gives a description of related work. In Section III we present the design of the Service* protocol which we evaluate in section IV. We conclude in section V.

II. Related Work

The concept of service discovery and service advertisement is well established in distributed systems [2], since networked entities need to locate available remote resources in order to

use them. The basic properties of service-oriented applications, such as on-demand operation and loosely-coupled application dependencies, make the service oriented model attractive for building network-centric mobile applications. Conversely, supporting a service-oriented model in MANETs adds some distinctive requirements that demand a different approach to those taken by current platforms. Some research in MANET services ([3], [4]) has helped qualify some of the problems between different environments. Work on service discovery in mobile ad hoc networks focuses on using decentralised architectures to overcome the limitations of traditional discovery mechanisms, such as SLP [5], Jini [6], and UPnP [7], that rely on fixed infrastructure. Research in service architectures for mobile environments can be classified into service protocols, used for service discovery or advertising and service platforms which focus on building middleware to support service oriented programming models.

A. Service Protocols

Research in the area of protocols to support service architectures has produced interesting results with some protocols exploiting location to achieve distributed discovery, while others rely on the formation of a virtual backbone of nodes which can subsequently act as service brokers. Protocols that use location for service discovery are the Grid Location Service (GLS) [8], the Distributed Location Management (DLM) protocol [9] and Rendezvous Regions (RR) [10]. The Grid Location System is a routing protocol that enables location dissemination and performs geographic forwarding of packets, while DLM and RR use hashing to map services to geographic regions and geographic forwarding to direct discovery requests to specific regions. Such protocols make strict assumptions about the knowledge of a geographic grid that must be shared by all operating nodes in addition to having very basic discovery query semantics due to hashing (e.g., difficult to search for services using attribute–value pairs or regular expressions).

The protocol described in [11], provides a distributed service discovery mechanism by forming a virtual backbone from selected mobile nodes. The nodes in the backbone subsequently act as service brokers, permitting service registration and thus facilitate service discovery. In the same paper, the authors propose applying the graph-theoretical property of dominating sets to build a virtual backbone in ad hoc networks. They show encouraging results when comparing such a mechanism against anycast and multicast protocols. However, it requires a complex network layer infrastructure and trades efficiency for generality by specifying the protocol at the routing layer.

B. Service Platforms

Konark [12], GSD [13], and DEAPspace [14] are projects designed to provide a service discovery platform. Konark is a service discovery mechanism that is based on a hierarchical classification of services. Its architecture supports both discovery and advertising, but is limited to locally scoped multicast. GSD uses caching and semantic matching of services to efficiently discover services in a distributed manner. GSD uses a predefined ontology to group similar services and forwards service requests by exploiting this grouping. In contrast, Service* does not rely on a predefined ontology and makes no assumptions on the nature of the service interface. DEAPspace has been designed for pervasive environments and supports service discovery, but has a more narrow scope as is directed towards personal area networks.

III. Service* Architecture and Design

A. System Model

Service* has minimal dependencies on the MAC and network layers. Without loss of generality, we rely on any 802.11 compatible MAC protocol. From the network layer, we require that every node exposes its 1-hop neighbours with some degree of accuracy as well as basic unicast and 1-hop broadcast facilities.

Service* also assumes that the service interface is lightweight so that both storing in resource-constrained devices and transmitting in bandwidth-constrained channels is not expensive. In the simulated protocol version we assume that multiple service interfaces can fit in a single packet.

Throughout the paper we use the following terms and definitions:

- *Service Interface—SI*: A generic interface to a service can contain a name, a set of attribute-value pairs and a set of callable methods or a Uniform Resource Identifier (URI). The service interface is the datum that is advertised. For the purposes of this paper we assume that the interface simply consists of a name, an expiration timestamp and an incarnation number. The incarnation number is updated every time the service provider initiates an advertisement. We use the service incarnation number to characterise the latency between advertisement and discovery.
- *Service Provider—SP*: Any node that implements a service interface. Service providers are responsible for periodically initiating advertisement sessions.
- *Service Broker—SB*: Any node that is not a provider, but has registered a set of service interfaces in its repository. Broker nodes are delegated the same task as *SP* nodes (i.e., initiating advertising sessions) but are selected and deselected dynamically based on local rules.
- *Initiator*: A node that is either an *SP* or an *SB* and has initiated an advertising session.
- *k-Neighbourhood—*N_i^k: The set of nodes that is within k–hops from node i.
- *Registry—*R_i: The set of service interfaces registered at node i.

B. Overview of the Distributed Advertising Protocol

The goal of Service* is to facilitate distributed advertising and eventual aggregation of services. To this aim, providers and brokers independently invoke advertising sessions. Each session consists of a 3-round exchange of service and node

Protocol 1 Advertising Session

1: $SrvcReq_I$ #Service Request packet at Initiator node I
2: $SrvcRply_j$ #Service Reply packet at node j
3: $SrvcBrokerSelect_I$ #Broker Select packet at Initiator node I
4: I broadcasts $SrvcReq_I$ to N_I^1
5: **for all** $j \in N_I^1$ such that j received $SrvcReq_I$ **do**
6: $SrvcRply_j \Leftarrow N_j^1$ #Append 1-Neighbourhood to SrvcRply packet
7: **if** j is *SP* or j is *SB* **then**
8: $SrvcRply_j \Leftarrow R_j$ #Append list of SI to SrvcReply packet
9: **end if**
10: j unicasts $SrvcRply_j$ to I
11: **end for**
12: **for all** $SrvcRply_j$ received in I **do**
13: $I \Leftarrow N_j^1$ #Aggregate j's 1-Neighbourhood in I
14: $I \Leftarrow R_j$ if j is *SP* or *SB* #Aggregate j's list of SI in I
15: **end for**
16: I runs the *Broker Selection* Algorithm (Algorithm 2) to identify Brokers
17: $SrvcBrokerSelect_I \Leftarrow$ *List of Brokers* #Append list of Brokers to SrvcBrokerSelect packet
18: $SrvcBrokerSelect_I \Leftarrow R_I$ #Append list of SI to SrvcBrokerSelect packet
19: I broadcasts $SrvcBrokerSelect_I$ to N_I^1

Algorithm 2 Broker Selection Algorithm

1: D_j #Degree of a node $j \in N_I^1$ is the size of N_j^1, excluding all the members of N_I^1 and excluding node I
2: $BSet_I = \{\}$ #broker set variable at Initiator node I
3: **for all** $j \in N_I^1$ such that I has *NOT* received packet $SrvcRply_j$ **do**
4: Remove node j from N_I^1
5: **end for**
6: **for all** $j \in N_I^1$ **do**
7: Calculate D_j
8: **if** j is *SP* **then**
9: $BSet_I \Leftarrow j$
10: **end if**
11: **end for**
12: Add to $BSet_I$ those nodes in N_I^1 that are the *only* nodes to provide coverage to only a single node in N_I^2
13: Remove nodes from N_I^2 that are now covered by nodes in $BSet_I$
14: **while** $k \exists N_I^2$ such that k is not covered by any node in $BSet_I$ **do**
15: **for all** $j \in N_I^1$ **do**
16: $BSet_I \Leftarrow j$ such that j covers most nodes in N_I^2
17: Remove nodes from N_I^2 that are now covered by nodes in $BSet_I$
18: **end for**
19: **end while**

information between the initiator and its 1-neighbourhood. Protocol 1 describes the core steps undertaken when a session is initiated. Line 4 shows the initial phase of the protocol, where a node broadcasts a service request packet. Lines 5—11 outline the actions of nodes receiving such a broadcast. Each receiving node appends its own list of 1-neighbourhood to a reply packet. Nodes that are service providers or were delegated the task of acting as brokers also append the list of service interfaces stored in their registry. Consequently, nodes send their replies back to the initiator. After the initiator receives the replies, it runs the *broker selection* algorithm listed in Algorithm 2. We discuss the issues arising with the request–reply nature of the protocol at the end of this section.

During broker selection, the initiator node decides the optimal placement of its service interfaces in a subset of its 1-hop neighbouring nodes. The subset is selected so that every node in the initiator's 2-neighbourhood has access to the aggregated services via a single broadcast. The selected subset constitute the new broker nodes and is subsequently broadcasted along with the aggregated services (line 19 of protocol 1). Any node that receives this final broadcast, checks its own address against the set of included addresses in the $SrvcBrokerSelect$ packet. A positive match from a standard node results in such a node registering the services found in the same packet and thus becoming a broker. Any node that is already a broker or a provider and is included in the list of new broker nodes updates its repository accordingly. A broker node not selected as an *SB* node, becomes a standard node again (i.e., clears its repository and stops advertising).

The broker selection algorithm runs at the initiator node and is similar[1] to the heuristic to compute Multi Point Relays (MPR) as discussed in section 8.3.1 of [15], but with some important differences. Firstly, it does not result in the formation of a dominating set between the selected MPRs as does the algorithm in [15]. Secondly, the actual selection algorithm is changed to cater for the concept of *service coverage* rather than that of *wireless coverage*. Although the two concepts are similar (e.g., in MANETs you can't have single hop service coverage unless you have wireless coverage) there are important differences to fit the domain of service discovery. One such difference is the fact that providers are prioritised over other nodes during the selection round (line 8 of algorithm 2). This results in selecting broker nodes only after removing a subset of nodes from the initiator's 2-neighbourhood since they are covered by existing providers (line 13 of algorithm 2). Such a rule is aimed at delegating advertising responsibilities to nodes that are already providers, avoiding in this way the unnecessary expansion of broker nodes.

More importantly, while the process of selecting MPRs is a continuous and reactive process (i.e., execution of the algorithm takes place every time the 1-neighbourhood changes) which is carried out by all participating nodes in the network, Service* executes the *broker selection* algorithm after discrete time-out periods. Since service discovery does not share the same strict requirements as communication (i.e., maintaining up to date routes, etc.), we chose to relax reactive behaviour for the benefit of reducing overhead. Doing so allows node communication to realise the maximum channel throughput.

After broadcasting the $SrvcBrokerSelect_I$ packet, the initiator node can decide to deregister all services from its repository, thus becoming a standard node. This avoids an infinitely expanding broker set. Deregistration is subject to cer-

[1]Lines 12—19 of algorithm 2 describe an abbreviated version of the heuristic in section 8.3.1 of the OLSR IETF RFC.

Algorithm 3 Deregistration Algorithm

Require: N_I^1 contains the newly selected brokers
1: $BSet_I = \{\}$ #aux variable holding broker node ids at I
2: $SSet_I = \{\}$ #aux variable holding standard node ids at I
3: **for all** $j \in N_I^1$ **do**
4: **if** j is *SP* or j is *SB* **then**
5: $BSet_I \Leftarrow j$
6: **else**
7: $SSet_I \Leftarrow j$
8: **end if**
9: **end for**
10: **for all** $x \in BSet_I$ **do**
11: **for all** $y \in SSet_I$ such that $SSet_I \cap N_x^1 \neq \emptyset$ **do**
12: Remove y from $SSet_I$
13: **end for**
14: **end for**
15: **if** $SSet_I$ is empty and I is *SB* **then**
16: Deregister
17: **end if**

tain service coverage conditions and is decided by algorithm 3. In short, an initiator node deregisters if it is a broker node and all providers and brokers in its 1-neighbourhood cover every other node in its 1-neighbourhood. It is easy to see that the initiator node is then a redundant broker node, since by virtue of the broker selection algorithm we cover all 2-neighbourhood nodes while a positive result by algorithm 3 guarantees service coverage for all 1-neighbourhood nodes. Thus, any discovery broadcast by any node in the initiator's 2-neighbourhood can be given a positive response without the initiator acting as a broker.

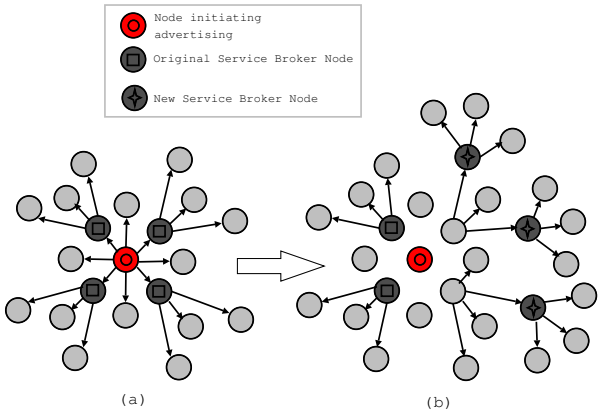

Fig. 1. Broker Registration and Deregistration in Service*

Figure 1 displays the broker selection concept graphically. In part (a), a provider, after executing the Service* protocol, has selected four broker nodes which now contain a set of service interfaces in their repositories. Part (b) illustrates a potential network state, after a time interval has passed and some nodes have initiated a new advertising round. New brokers have formed thereby expanding the service coverage, while the nodes that initiated the advertising sessions have reasoned that there is sufficient coverage for all their neighbouring nodes and have ceased acting as brokers.

Parameters	
Network Area	1000 x 1000 m.
# of Nodes	40/60/80
Node Tx Range	150 m.
Simulation Time	400 sec.
# of Trials	10
Mobility Model	Random Waypoint
Min-Max Speed	Uniform Distr. 1–12 m/sec
Advertise Timeout	Uniform Distr. 1.6875–2.8125 sec.

TABLE I
SIMULATION DETAILS

The aim of such a dynamic approach of registration and deregistration of brokers is to provide a mechanism where service advertisement does not require the use of network-wide transmission of messages but achieves the desired property of single hop discovery through local interactions. Achieving global behaviour through local interactions has important ramifications. When service dissemination is performed by single-hop broadcasts, the overall control packet overhead is reduced. Additionally, aggregation and redistribution of service interfaces provides the property of eventual network-wide availability of all provided services. However, contrary to approaches that use multicast or anycast for service discovery, this approach does not guarantee 100% service discovery even when a service exists in an accessible node. Evaluation of the Service* protocol though, shows that very high discoverability (i.e., over 89%) is possible with only a fraction of the cost of advertising through flooding. Moreover, the protocol scales gracefully and increases the ratio of successful discoveries as the number of providers and node density is increased.

Certain protocol states in the Service* protocol require termination conditions that rely on up to date knowledge of the initiator's 1-neighbourhood. Such a condition is displayed in line 12 of Protocol 1. In this waiting state, the initiator has sent the service request packet and is waiting for a reply packet from its surrounding nodes. How the set of 1-hop neighbours is provided to each node is an orthogonal issue, but an accurate and complete view of 1-neighbourhood can result in a more optimal broker placement. It also enables the initiator node to take more responsive actions (i.e. no timeouts when waiting for replies) and avoid spurious reply packets from nodes not appearing in the initiator's 1-neighbourhood. However, such an accurate view of a node's neighbours is not always available. A rigid approach to providing both an accurate and complete view would require a membership service like the one developed in [16], which would demand an extra protocol adding to channel saturation. Currently, we take a best effort approach and rely on the 1-neighbourhood view given by routing protocols such as AODV and OLSR. Because of inherent view inaccuracies, we also rely on the use of timeouts for terminating waiting states.

Nodes	*SP*	*SP + SB*	*SB* as % of all nodes
40	4	14.3	25.67%
	8	17.1	22.69%
	16	22.5	16.35%
60	6	22.0	26.73%
	12	26.3	23.78%
	24	34.6	17.69%
80	8	29.6	26.97%
	16	35.3	24.13%
	32	46.6	18.22%

TABLE II
AVERAGE BROKER NUMBERS

IV. EVALUATION

We evaluated the Service* protocol against other push-based advertising schemes as opposed to discovery protocols based on multicast or other backbone formation mechanisms. Evaluation against the latter type would necessitate comparing dissimilar mechanisms and would require specific infrastructural requirements from the routing layer. Instead, we have evaluated the Service* protocol against a pure flooding and an optimised flooding advertising mechanism. We implemented the Service* protocol in the ns-2 simulator and used the set of parameters in table I for all experiments. All simulations were run over 10 different mobility scenarios with different experiments executed by varying the total number of nodes and the percentage of nodes acting as providers. Each mobility scenario was further simulated 10 times, such that the different parameter permutations are averaged over 100 simulations. Nodes selected as providers are randomly chosen in each run and start advertising after 10 seconds of simulation time.

The evaluation criteria we used are discoverability and overhead relative to the number of total and provider nodes. With these criteria we try to capture how the protocol scales when more services are added to an ever increasing network size. We define discoverability as the ratio of discovery queries with positive replies over the total number of discovery queries. Overhead is simply the number of control packets transmitted during the advertising phase, excluding discovery.

Discoverability is measured by selecting a random node every 0.5 seconds and issuing a discovery query for a randomly selected service. Discovery starts after 30 seconds of simulation time. A discovery query in this case entails searching the node's cache in addition to broadcasting a single hop query to neighbouring nodes. In a successful discovery, the node either has the requested service in its cache, resulting in a cache hit or receives a positive reply from a neighbouring node. For the flooding and optimised flooding approaches we assume discoverability of 100%, since services would be pushed into every available node. Discoverability results for the Service* protocol are shown in figure 2. We run experiments varying the number of provider nodes between 10%, 20% and 40% of available nodes. The actual number of provider nodes appears at the top of every bar.

Service* exhibits high discoverability, over 89%, in every scenario. The cache hit ratio varies from about 35% to 55%.

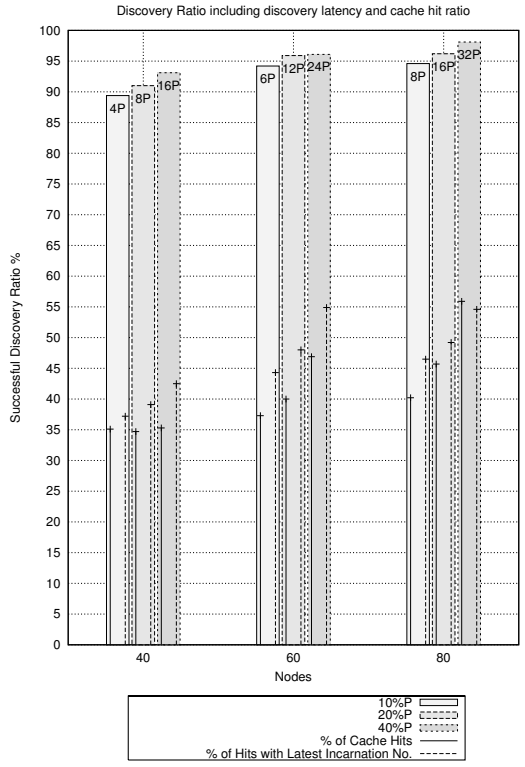

Fig. 2. Successful discovery ratio

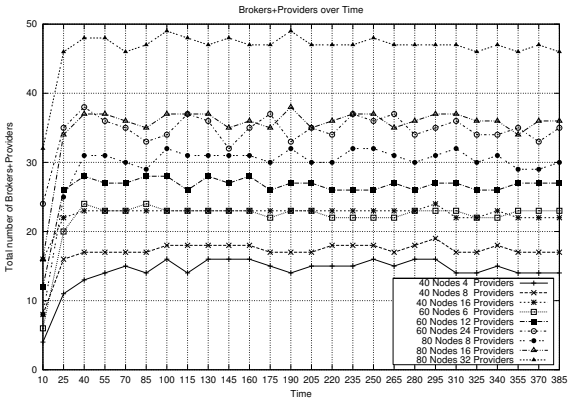

Fig. 3. Broker fluctuation over time

This variation depends mainly on the node density and the absolute number of provider and broker nodes. The cache hit ratio helps characterise the degree to which cached service interfaces aid the discovery procedure. While a high cache hit ratio reduces discovery overhead, it can also result in stale service interfaces when provider nodes disconnect or fail. We plan to further investigate the effects of cache utilisation on the Service* protocol.

Continuing with figure 2, the dotted line in every bar represents discovery latency, i.e., the property of discovery to obtain the service interface with the most up to date incarnation number. In a dynamic environment with frequent

node failures and network partitions, it is important that most recent advertisements propagate quickly throughout the network. In Service*, this is even more challenging because of the nonreactive nature of the protocol. In addition, discovery latency is also dependent on the frequency of advertisement and network coverage characteristics making such measurement a complex issue. Despite that, in some cases over 50 percent of successful discovery queries correspond to the most recent service incarnation.

Figure 3 shows that after an initial period during which service brokers propagate through the network, the overall number of providers and brokers eventually reach dynamic stability. While nodes become brokers and brokers deregister, the total number remains stable. The average number of *SP* and *SB* nodes throughout all the simulations is presented in column 3 of table II. More importantly, column 4 indicates that the protocol converges to a ratio of broker nodes that is independent of available nodes and mobility patterns. Instead, it depends on the percentage of provider nodes. As the percentage of provider nodes increases, we see the percentage of broker nodes dropping. This explains the good scalability characteristics of Service*.

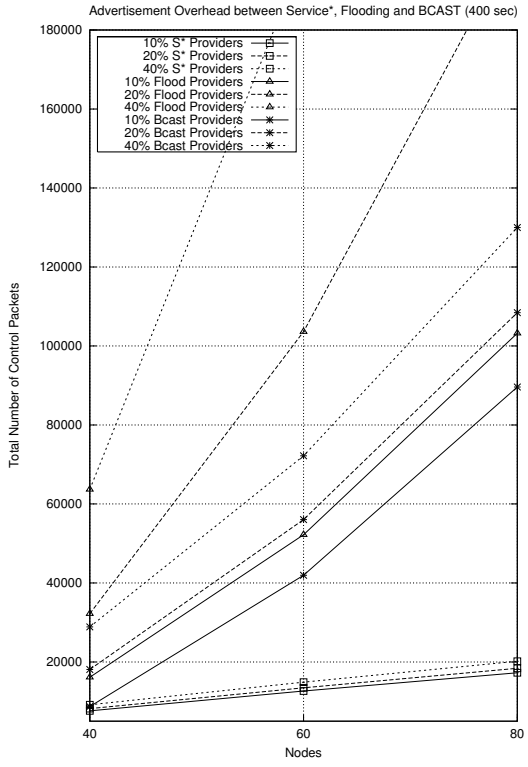

Fig. 4. Control message overhead between Service*, Flood and Bcast

Figure 4 compares the number of packets generated between Service* and two other advertising mechanisms, namely Flood and Bcast [17]. All three mechanisms use the same simulation parameters and are averaged over the different simulation executions. Flood is a pure flooding advertising mechanism, implemented as a network-wide broadcast from provider nodes to every other node in the network. As expected this is the most inefficient scheme, with the number of advertising control packets quickly saturating the majority of available channel capacity. Bcast is an aggressive broadcast optimisation scheme first introduced in [18]. We use the ns-2 implementation of this protocol as described in [17]. While both of the advertising flooding schemes do not require extra discovery broadcasts as service interfaces get stored in each node's registry, they do however incur a significant cost in the network, demonstrating poor scalability as the number of nodes and providers increases.

V. Conclusion and Future Work

In this paper, we presented Service*, a distributed, push-based service advertisement protocol for ad hoc networks. We provided the rationale behind distributed service advertisement as being the first step towards a service architecture for MANETs. We implemented and evaluated Service* against two other advertising mechanisms based on flooding. Results indicate that Service* offers over 89% discoverability with a single-hop discovery broadcast, while demonstrating excellent scalability properties as the number of nodes and the number of service providers in the network increases. As future work, we want to continue evaluation of Service*, specifically in the areas of discovery latency and the handling of a large number of services. We also plan to move towards specifying a lightweight service architecture for mobile ad hoc networks that relies on the properties of the protocol presented in this paper.

References

[1] W. S. A. W. Group, "Web services architecture," February 2004, http://www.w3.org/TR/ws-arch/.

[2] S. J. Mullender and P. M. B. Vitányi, "Distributed match-making for processes in computer networks," in *Proceedings of the 4th annual ACM Symposium on Principles of Distributed Computing*. ACM Press, 1985, pp. 261–271.

[3] R. Sent, R. Handorean, and G.-C. Roman, "Service oriented computing imperatives in ad hoc wireless settings," Washington University, Department of Computer Science, St. Louis, Missouri, Tech. Rep. WU-CSE-2004-05, 2004.

[4] R. Handorean, R. Sen, G. Hackmann, and G.-C. Roman, "A component deployment mechanism supporting service oriented computing in ad hoc networks," Washington University, Department of Computer Science, St. Louis, Missouri, Tech. Rep. WUCSE-04-02, 2004.

[5] E. Guttman, C. Perkins, J.Veizades, and M.Day, *Service Location Protocol, Version 2*, IETF, 1999, http://www.rfceditor.org/rfc/rfc2608.txt.

[6] K. Arnold, R. Scheifler, J. Waldo, B. O'Sullivan, A. Wollrath, B. O'Sullivan, and A. Wollrath, *Jini Specification*. Addison-Wesley Longman Publishing Co., Inc., 1999.

[7] M. Corporation, "Universal plug and play: Background," 1999.

[8] J. Li, J. Jannotti, D. S. J. De Couto, D. R. Karger, and R. Morris, "A scalable location service for geographic ad hoc routing," in *Proceedings of the 6th annual international conference on Mobile computing and networking*. ACM Press, 2000, pp. 120–130.

[9] Y. Xue, B. Li, and K. Nahrstedt, "A scalable location management scheme in mobile ad-hoc networks," in *Proceedings of the 26th Annual IEEE Conference on Local Computer Networks*. IEEE Computer Society, 2001, pp. 102–112.

[10] K. Seada and A. Helmy, "Rendezvous regions: A scalable architecture for service location and data-centric storage in large-scale wireless networks," in *18th International Parallel and Distributed Processing Symposium (IPDPS'04) - Workshop 12*, IEEE. IEEE Computer Societyr, April 2004, p. 218.

[11] U. C. Kozat and L. Tassiulas, "Network layer support for service discovery in mobile ad hoc networks," in *IEEE INFOCOM*, vol. 22, March 2003, pp. 1965 – 1975.

[12] S. Helal, N. Desai, V. Verma, and C. Lee, "Konark – a service discovery and delivery protocol for ad-hoc networks," in *Proceedings of the Third IEEE Conference on Wireless Communication Networks (WCNC)*. IEEE Computer Society, March 2002.

[13] D. Chakraborty, A. Joshi, T. Finin, and Y. Yesha, "Gsd:a novel group based service discovery protocol for manets," in *4th IEEE Conference on Mobile and Wireless Communications Networks (MWCN 2002)*. IEEE Computer Society, September 2002. [Online]. Available: gunther.smeal.psu.edu/13044.html

[14] R. Hermann, D. Husemann, M. Moser, M. Nidd, C. Rohner, and A. Schade, "Deapspace: Transient ad-doc networking of pervasive devices," in *Proceedings of the 1st ACM international symposium on Mobile ad hoc networking & computing*. IEEE Computer Society, 2000, pp. 133–134.

[15] T. Clausen, P. Jacquet, A. Laouiti, P. Minet, P. Muhlethaler, A. Qayyum, and L. Viennot, "Optimized Link State Routing Protocol, Internet Draft (draft-ietf-manet-olsr-06.txt)," September 1 2001, work in Progress.

[16] L. Briesemeister, "Group membership and communication in highly mobile ad hoc networks," Ph.D. dissertation, School of Electrical Engineering and Computer Science, Technical University of Berlin, Germany, Nov. 2001.

[17] T. Kunz, "Multicasting in mobile ad-hoc networks: achieving high packet delivery ratios," in *CASCON '03: Proceedings of the 2003 conference of the Centre for Advanced Studies on Collaborative research*. IBM Press, 2003, pp. 156–170.

[18] W. Lou and J. Wu, "On reducing broadcast redundancy in ad hoc wireless networks," *IEEE Transactions on Mobile Computing*, vol. 1, no. 2, April-June 2002.