Service Discovery Strategies in Ubiquitous Communication Environments

Siva Sivavakeesar, Oscar F. Gonzalez, and George Pavlou, University of Surrey, UK

ABSTRACT

Past communication and computing trends are inadequate to deal with today's complex, distributed, and diverse network environments. With the recent development of small-size tetherless communication/computing devices and the increasing diversity in their capabilities, autonomic ubiquitous communication environments are emerging. In such environments, the execution of a complex task does not necessarily make use of prconfigured devices or networks, but requires instead the selection of suitable computing elements on-the-fly, based on the task requirements and device characteristics (i.e., *networking-on-demand* or *task-centric* networking). Research is currently being carried out in various dimensions in this emerging domain, including, among other approaches for context awareness, service discovery and self-management. In this article we concentrate on proposing a distributed mechanism for discovering suitable service elements on-the-fly.

INTRODUCTION

In future ubiquitous communication environments, heterogeneous devices will be able to get together and form a network spontaneously on-demand. This is the "anyone, anywhere, anytime" paradigm of intelligent overlay community establishment that uses mobile ad hoc networking (MANET) as the basic underlying technology. Each element that becomes part of such an on-demand network does not necessarily belong to a standalone MANET; instead, it may be a fixed or mobile element that may also belong to other networks at the same time (e.g., Ethernet, WLAN, GSM, 3G/4G, etc.), but it becomes part of a new network in order to accomplish a particular task. This type of ubiquitous communication environment is one where applications and services are not deployed onto a preexisting network, but instead the network itself grows out of the applications and services the users want. This approach enables users to view the network in the manner most appropriate to them and their requirements. An expected result of this "autonomic" approach is the ease with which larger, more complex services can be composed from smaller ones. Relevant mechanisms can thus be rapidly deployed in such an on-demand network, enabling new applications in diverse areas such as smart homes and offices, distributed robotics, sensor networks, smart battlefield, disaster relief, and so forth. This requires mechanisms for distributed service and resource discovery, self/context awareness, self-organization, and self-management in general. This article, however, deals with the issue of discovering suitable devices that are going to become part of the on-demand service-centric network.

In this process, a distributed application is represented as a *task* which is composed of multiple smaller and less-complex *subtasks* that need to be performed on various, potentially heterogeneous, computing elements [1]. For any service discovery mechanism to work, the dependencies induced by logical patterns of data flow among different elements need to be known in advance. Hence, it is advantageous to represent such dependencies of a given application as an abstract task graph (TG). Given that in a ubiquitous environment there may exist several devices that can participate in the execution of subtasks, the service discovery mechanism should be intelligent and robust enough to identify the most suitable devices/elements, depending on the current context. However, in this preliminary work we address the discovery of all the relevant devices/elements, even if some of these may not necessarily be the most optimal for the current context.

Given that relevant devices/elements may be mobile, a distributed service discovery mechanism is preferable to a centralized one for reliability and robustness. Service discovery mechanisms can be classified according to one

0163-6804/06/$20.00 © 2006 IEEE

key property, in a similar fashion to MANET routing protocols. This property characterizes the way in which clients acquire server details. Three different policies can be adopted by any distributed service discovery mechanism: *proactive*, *reactive*, and *hybrid*. The proactive approach relies on periodic dissemination of service information in order to maintain consistent, accurate information regarding the services provided throughout the ubiquitous environment. The reactive approach, on the other hand, attempts to make the service discovery process more efficient by passing on-demand information regarding the required service(s) to the requesting devices. Since both schemes have their pros and cons, a hybrid mechanism tries to strike a right balance between these two extreme approaches.

In this article we propose and experimentally evaluate two different service discovery strategies. The first reactive strategy relies on flooding, and despite the fact the adoption of promiscuous listening reduces the flooding effect, its scope is limited to small-scale ubiquitous environments. On the other hand, the second strategy uses unicasting as opposed to arbitrary flooding and as such it is highly suitable for large-scale environments. We evaluated both approaches thoroughly using simulations. We also implemented and tested the first one in our experimental testbed, using as case studies an online gaming application and a cyber-foraging application. In the case of online gaming, the purpose is to identify suitable partners in order to play a given game in the ubiquitous environment — in this respect the users provide a "gaming service." We chose a simple Tic-Tac-Toe game and, in this case, the service discovery mechanism identifies interested players; Fig. 1 depicts a screen shot taken in the middle of the game. The purpose of the second application is to enable a "thin" client to perform a complex calculation in a "thick" node and to download the results (i.e., "cyber-foraging").

Figure 1. *Peer-to-peer online gaming with Tic-Tac-Toe.*

Related Work

Related Work on Service Discovery Protocols

Various service discovery mechanisms have been proposed for fixed IP networks; the main ones are IETF's Service Location Protocol (SLP), Sun Microsystems' Jini, IBM's Salutation Protocol, and Microsoft's Universal Plug and Play (UPnP) [3]. With the exception of UPnP, the rest follow a centralized proactive approach, whereby devices/elements periodically update their service information with one or more predefined central directories. In SLP a centralized service repository is termed directory agent (DA) and it is used by service agents (SAs) to register their services and then by user agents (UAs) to locate the required services. Although adopting a similar strategy, Jini provides additional functionality by aiming to turn the network into a flexible and easily administered tool. Jini Services register their attributes with at least one lookup service (the repository) which is later contacted by Jini clients. However, lookup services do not just store service information, but rather save a piece of software that contains all the "intelligence" to execute the required code locally or remotely. Such pieces of software are known as proxies and they are the key to the dynamic use of drivers at runtime in Jini. Salutation is an open standard that not only provides a service discovery protocol but also a session management protocol. In this architecture, the centralized repository is called a salutation manager.

Using a centralized service repository, however, raises the possibility of a single point of failure, which makes relevant mechanisms unattractive to wireless ubiquitous environments. On the other hand, UPnP can be implemented with or without a lookup server (central directory). When it is implemented in a fully distributed manner, each service provider element needs to disseminate an "advertisement" message upon joining a group and a "bye" message upon leaving a group. Although the sheer amount of control traffic incurred in this process can be acceptable in fixed IP networks, this will burden the bandwidth-constrained wireless links in ubiquitous environments. There have also been related works in the domain of mobile ad hoc networks. Since MANETs are distributed in nature, relevant mechanisms are potentially suitable for ubiquitous environments and it is worth exploring their suitability. The work in [4] presents a distributed service discovery approach that relies on the creation of a *virtual backbone* for registering and requesting services on a dynamic network topology. The virtual backbone is formed by a subset of nodes termed service broker nodes (SBNs) (i.e., a dominating set). Each service provider element advertises its service information to the virtual backbone through its dominating node, and in this way the virtual backbone tries to maintain accurate information pertaining to the services provided by the network as a whole. However, it is not clear how adaptable the virtual backbone is to node mobility; if backbone creation does not take mobility into account, the associated control traffic may saturate the network. A similar virtual backbone

Service discovery mechanism	Network type*	Multicast	Broadcast	Approach	Backbone formation	Clustering
Service location protocol	Fixed	Yes	No	Centralized	Not applicable	Not applicable
Jini	Fixed	Yes	No	Centralized	Not applicable	Not applicable
IBM's Salutation Protocol	Fixed	No	No	Centralized	Not applicable	Not applicable
Universal plug and play	Fixed	Yes	No	Centralized	Not applicable	Not applicable
SBN-based virtual backbone [4]	Mobile	No	No	Distributed	Yes	No
Mediator-based virtual backbone [5]	Mobile	Yes	No	Distributed	Yes	Yes
Konark [6]	Mobile	Yes	No	Distributed	No	No
Ad hoc naming service [7]	Mobile	Yes	No	Distributed	No	No
Fully distributed** (ours)	Mobile	No	Yes	Distributed	No	No
Home-zone-based** (ours)	Mobile	No	No	Distributed	No	Yes

* Network type for which the protocol was designed.
** Our service discovery approaches

Table 1. *Comparison of service discovery mechanisms.*

creation mechanism has been employed in [5], but it differs from the previous work in the fact that a node can become a dominating one (mediator) only if it is a service provider. This approach does not always ensure a connected backbone graph, and hence communication among unconnected sets of the virtual backbone poses a big challenge. Related works in [6, 7] employ a proactive strategy, requiring each service provider element/node to advertise its service(s) throughout the network. A similar mechanism employing reactive strategy with an expanding ring search is specified in [1]. Although these resemble our distributed service discovery mechanism which is targeted to small-scale ubiquitous environments, these approaches will not scale simply for large environments. Table 1 presents a summary of the service discovery mechanisms considered in this section along with our own strategies to be discussed below.

It should be finally mentioned that while there exists research work on service discovery mechanisms both in fixed IP networks and MANETs, there has been no attempt to apply it to the domain of ubiquitous environments in the manner proposed here.

The Proposed Service Discovery Mechanisms

A Middleware-Based Fully Distributed Approach

Our approach is closely related to that followed by Jini — but our aim is to make it more distributed. As stated above, Jini has a number of undesirable features: its centralized approach makes it vulnerable to a single point of failure and its lookup service architecture is too resource demanding. A ubiquitous environment may be comprised of both "thick" and "thin" devices. Therefore, we prefer to use a lightweight distributed service discovery framework to cater for the capabilities of "thin" nodes, while trying to offer as many of the desirable features of Jini as possible.

Our approach works in a similar fashion to the reactive routing protocol for mobile ad hoc networks (MANETs). Each node in our scheme has its own lightweight lookup server (i.e., service repository) consisting of entries that are subject to a timeout mechanism. An entry in our scheme is not proxy based as in Jini, because a proxy-based approach demands high resource usage. Instead, an entry is a description of a service and its attributes using a descriptive language defined using the extensible markup language (XML). The lookup server uses a tree-based service registry in order to logically group services by category and to locate the required services within each category with minimal search time. Accordingly, a tree has a number of levels that represent service classification. With this, as we move down the tree from the root to the leaves, services become more specific. This tree-based approach enables each node to maintain the known services in a systematic fashion, and hence helps to minimize the latency involved in the service discovery process. For example, a printer service can be identified by the following logical path name: "<path> RootService:HardwareService:PrintService: LaserPrint</path>," under which we can identify the preferred printer of our choice depending on its attributes (e.g., color, 300 dpi, 35 ppm).

In this scheme, whenever a node needs a particular service and does not know the devices that provide it, it will initiate a service lookup

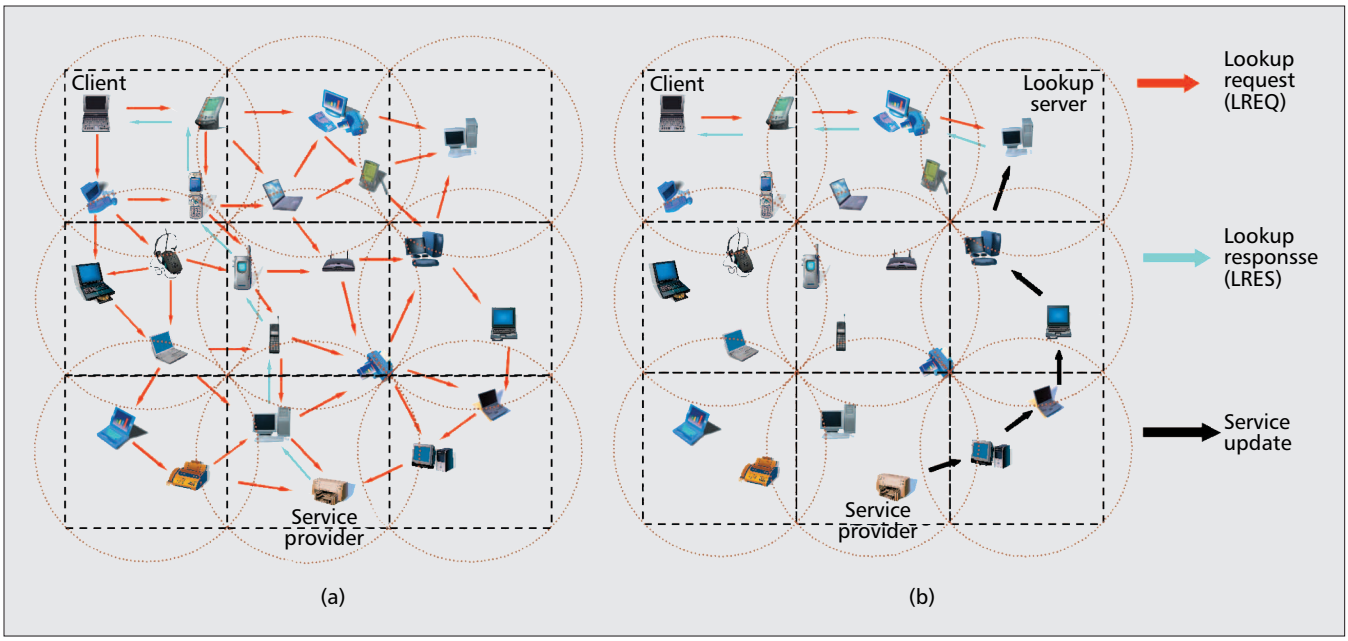

Figure 2. *Working mechanism of the proposed service discovery strategies: a) the middleware-based fully distributed approach; b) the home-zone-based service discovery mechanism.*

(discovery) process, as illustrated in Fig. 2a. In this process, the client node looking for the service disseminates a lookup request (LREQ) message at the middleware level with the following information used to describe the service being looked up {{*service*},{*attribute1, attribute2,* …}, where *service* can be represented by a logical path name as described above and the second part consists of a set of attributes used to locate the desired service provider node. As in ad hoc on-demand distance-vector (AODV) routing, the dissemination of such an LREQ message is subject to an expanding ring search algorithm. Accordingly, at the initial attempt the scope of the search is limited to a relatively small area in the vicinity of the client node in order to avoid networkwide flooding. Depending on the outcome of the initial search, the scope can gradually be increased, with the worst case being networkwide flooding if the provider node is at a "far edge" of the network. Any intermediate node having fresh enough information regarding the devices that provide the service being looked up may also respond with a lookup response (LRES) message being passed back to the LREQ source. Note that all this functionality takes place at the middleware level.

Given that the service discovery process partly employs flooding, and this together with node-mobility may induce loops, all LREQ and LRES messages include a broadcast id and a sequence number. Using the broadcast id along with the source address and the sequence number, intermediate nodes can easily identify a "fresh" service discovery process from a "stale" one. However, flooding in an environment employing the IEEE 802.11 distributed coordination function (DCF) as the underlying MAC will not work due to the potential presence of hidden and exposed terminals [8]. Although flooding in certain situations is inevitable, our strategy makes an attempt to limit its usage. For this purpose, we exploit the promiscuous listening functionality, whereby a node overhearing a LRES transmission makes a corresponding entry in its own lookup server. As mentioned above, such entries are subject to a timeout mechanism.

If, in the future, any new client node initiates a service lookup for a particular service and an intermediate node happens to have an entry learned through promiscuous listening, it may respond. The typical objective of employing such promiscuous listening functionality is to minimize unnecessary superfluous flooding which would otherwise take place.

Since this fully distributed service discovery strategy employs flooding, it can operate well in small networks but incurs heavy control traffic in large-scale networks. As such, we propose a second service discovery mechanism tailored to large-scale ubiquitous environments.

A Home-Zone-Based Service Discovery Mechanism

The basic idea of this scheme is to dynamically assign lookup server functionality to certain nodes located in a certain predefined location (*virtual cluster*) for a particular service. This strategy enables:

- The nodes providing a certain service S1 to always update the details regarding the service being provided, their address, location, and so forth with the lookup servers chosen specifically for S1, irrespective of the service provider locations, or whether they are moving or not
- Any client requiring the service S1 can straightaway contact the lookup servers and get the providers' details

This strategy improves scalability in a number of ways. Firstly, it minimizes superfluous flooding; secondly, it prevents the control messages from traversing unnecessary parts of the network; and

In order for the hash function to be able to map a service to a particular virtual cluster, it is necessary for the service to have an identifier. We propose the creation of a standard publicly-known ordered list. Such a list may not be static, but could be maintained dynamically.

thirdly, it minimizes the latency involved in the service discovery process. As will be inferred below, this strategy follows a hybrid approach, trying to blend the best of the proactive and reactive approaches.

It has been shown that routing protocols that use approximate location information scale better than topology-based routing protocols [8]. This motivates us to use geographic location information in our service discovery mechanism for scalability reasons. For this purpose, we make use of the virtual-cluster principles originally proposed in [8]. The idea is that a geographical area is divided into equal regions of circular shape in a systematic way, so that each mobile node can determine the circle it resides in if location information of the node is available. For this purpose we assume that

- Each node has a clear picture of the locations of the virtual-cluster in its memory at the time of bootstrap [8].
- Each virtual-cluster has a unique identifier, termed VID.
- Nodes get their location information through either GPS or a GPS-free positioning system.
- Each service provided in the network has a unique identifier.
- There exists a universal hash-function that maps every service to a specific virtual-cluster based on the service identifier.

Nodes in that particular cluster act as lookup servers (repositories) for that service and cease to do so when they move away. The lookup server functionality of a node will vary depending on its location, for example, a node in cluster A may function as a lookup server for the service S1; while if it moves to cluster D, it may function as a server for service S4.

A geographically oriented clustering algorithm and protocol are necessary for new nodes in the virtual-cluster to acquire service details. Our algorithm does not necessitate a leader or cluster head (CH) election [8]. The purpose is to maintain consistent group membership within a particular virtual-cluster with less overhead. The detailed description of the clustering mechanism in virtual clusters is, however, beyond the scope of this article. In this scheme, each node has its own service repository as described above. The lookup server functionality for a given service S1 is assigned to nodes located in a virtual-cluster that has a unique identifier. Accordingly, every node residing in the selected virtual cluster is responsible for maintaining the given service details in its service repository, and in this respect a virtual cluster becomes a home zone for a particular service. Since this mechanism requires all nodes to store service information about some other service providers, it can be classified as an all-for-some approach, which can scale well [8].

The static mapping of our hash function, as given by Eq. 1, is to facilitate simplicity and distributed operation, and it does not depend on the node density (a typical hash function that we used in our simulations is provided below).

$$hf(ServiceIdentifier) \rightarrow virtual\text{-}cluster \quad (1)$$

In this hybrid approach, each service provider needs to first identify the corresponding lookup server (i.e., virtual-cluster) for each service it provides by applying the hash function of Eq. 1 and then update the service details with nodes in that cluster, as illustrated in Fig. 2b. In our scheme, the service update mechanism is mobility driven as well as time driven. In the service discovery mechanism, service updates, and lookup requests (LREQs) and responses (LRESs) can be unicast based on the virtual-cluster identifier (VID), locations, and addresses of nodes. This strategy prevents these control messages from traversing unnecessary parts of the network in the same way as adopted in [8]. Whenever a node in a virtual cluster receives a service update meant for that cluster, it can stop unicasting that packet any further. Instead, it updates its own service repository and informs other nodes within the same cluster through a periodic *HELLO* packet, which includes the entries of the service repository maintained by that node, so that other cluster nodes can update their repositories.

Whenever a client node is interested in a particular service, it needs first to use the hash function of Eq. 1 to determine the virtual cluster associated with a service being looked up and then to contact the nodes' (any node functions as a lookup server residing in a cluster associated with the given service) lookup server(s) in that cluster. The lookup response can be initiated by lookup servers (i.e., nodes in the identified cluster), the service provider itself, or any intermediate node as long as it contains "fresh" information about the service being requested.

In the worst case, when a querying node has not received any response within a prespecified time period after having tried for a prespecified number of times, it will start gradually flooding its request in the network — in the same fashion as that adopted in our fully distributed approach. This may happen, for example, due to the fact that a given service's home-zone virtual cluster is currently empty.

Our service discovery mechanism is location-based using service repositories in home-zone virtual clusters, but not necessarily the underlying routing protocols. Any service provided by any node in the ubiquitous environment should have a consistent service identifier on a networkwide basis. In other words, in order for the hash function to be able to map a service to a particular virtual cluster, it is necessary for the service to have an identifier. We propose the creation of a standard publicly-known ordered list. Such a list may not be static, but could be maintained dynamically, with newly appearing services appended at the end of it. New services added to the list need to be known to all nodes in the network and, therefore, broadcasting may become inevitable in such situations. Since having popular services in the initial list reduces the likelihood of many new services being added, it is important to have a well structured and populated "standard" list at bootstrap. The identifier of a service is the rank of it as maintained in the list and with this the chances for two services to have the same identifier is minimized.

EVALUATION

TESTBED EXPERIMENTATION

The objective of our experimental testbed was to test and evaluate our fully distributed approach as described earlier in a real scenario.

Our cyber-foraging application (at the client side) made use of the Remote Method Invocation (RMI) tool, while our middleware was implemented on top of the JAdhoc packet version 0.21 — which is an implementation of the AODV routing protocol in Java [9]. JAdhoc employs tools such as tcpdump and libpcap and their objective is to allow an application to capture and process packets at the link layer. This functionality is very important to our service discovery mechanism, which works as a user space daemon, since it allows promiscuous listening. The ad hoc testbed supported the IEEE 802.11b/Bluetooth wireless standards for the required wireless communication among nodes. In our experiments, the service the client looked up was remote execution of a piece of Java code and retrieval of the results — our cyber-foraging case study, which was subsequently invoked.

The experimental testbed consisted of three laptops (named Zeus, Ares, and Poseidon) placed in such a way to have a random topology. The factorial server ran on one laptop (Ares) while the client was on another (Zeus) and Poseidon passively observed the medium. The factorial server first registered its service attributes with the lookup server running on the same machine (Ares). When the client initiated the service discovery process, the server responded. Zeus and Poseidon made an entry for this service in their local repositories. We measured the latency involved in the service discovery process. The service discovery latency was defined as the time involved from the time an LREQ is generated up to the time at which an RMI object is successfully instantiated, thus, confirming that the server has been contacted. The average latency obtained was of 1.86 ms with a standard deviation not exceeding 0.1 ms, which confirmed the relatively good performance of the developed service discovery middleware, although values varied considerably depending on the environment, the channel condition and its interference level, and so forth.

In order to demonstrate the benefit of exploiting the promiscuous listening functionality, we performed the second experiment. In this process, we disabled the lookup servers of Ares and Zeus, while keeping the entries of Poseidon intact. Zeus initiated the service discovery process for the second time and now Poseidon responded. The latency involved in this experiment was roughly equal to that of the previous case, because of the fact that all the three laptops were within close range of each other. It would have been much less, if Ares were placed far away from Zeus in a bigger network.

In order to test our module with "thin" nodes, we replaced the client Zeus with a PDA, named Venus, which has a 400 MHz Intel XScale processor and possesses a 48 MB ROM and a 128 MB RAM. The tests were carried out as described before with Venus as the client. In this case, the average latency was measured at 33 ms with a standard deviation of 21.7 ms. This difference is partly due to the fact that the PDA needs longer processing time to perform instructions associated with the opening and closing of sockets as well as sending the data. This is inevitable because of its reduced capabilities.

SIMULATIONS

The scalability of our service discovery approaches was assessed in terms of increasing node-count and increasing average node speed. We chose the reactive approach proposed in [1] for performance comparison purposes — this approach is termed FD_without_PL(AODV), meaning that it is a fully distributed (FD) approach that uses AODV as the underlying routing protocol and does not use promiscuous listening (PL). We performed our simulations using the GloMoSim simulation package in which we implemented both our service discovery mechanisms together with FD_without_PL(AODV) [8]. Unless explicitly stated otherwise, our simulation parameters took the following values:

- Nodes move according to the random waypoint mobility model with a constant speed chosen uniformly between zero and 10 ms^{-1}.
- The pause time took a value that is exponentially distributed with a mean of 30 s.
- The transmission range of every node was 100 m.
- The link capacity was 2 Mb/s.
- The MAC layer protocol was the DCF of IEEE 802.11.
- The total simulation time for each scenario was 300 s.

The network provided 12 different services in total. The probability that any node could arbitrarily become a server to offer one of the 12 services was 50 percent. Each node generated a service request periodically, with the period being uniformly distributed between 10 and 15 s. However, the probability that a given node becomes a client at the end of the above period was 70 percent. Our fully distributed approach was tested using both AODV (called FD_with_PL(AODV)) and a home-zone-based geo-casting (HZGC) routing protocol (called FD_with_PL(HZGC)) [8], and their results were compared against our home-zone-based service discovery mechanism, Home-Zone Service Discovery, running over the HZGC. The principle metrics of interest are network control overhead and service discovery latency. The control overhead is defined here as the extra control traffic generated as part of the service discovery process only. The hash function used in our simulations was a simple modulus function. Accordingly, if a particular region has H number of virtual clusters in total and the service of interest has an identifier ι in the ordered service list, then the corresponding lookup servers are located in the cluster VID given by ι Modulus H.

The objective of the first set of simulations is to assess the scalability of our service discovery approaches in terms of increasing node count. In order to properly see the effect of increasing network nodes on the service discovery schemes,

In the future, we will concentrate on devising a generic way to represent a distributed task/application in terms of a TG and also use contextual information in the discovery of the best possible (optimal) nodes to invoke the service.

the terrain area is also increased with an increase in the number of nodes, so that the average node density is kept constant in this set of simulations. The number of nodes in this case was varied from 20, 80, 180, 320, 500, and 720. The terrain-area size was varied such that the average node degree remained the same and, accordingly, 200×200 m^2, 400×400 m^2, 600×600 m^2, 800×800 m^2, 1000×1000 m^2, and 1200×1200 m^2 were selected for each scenario.

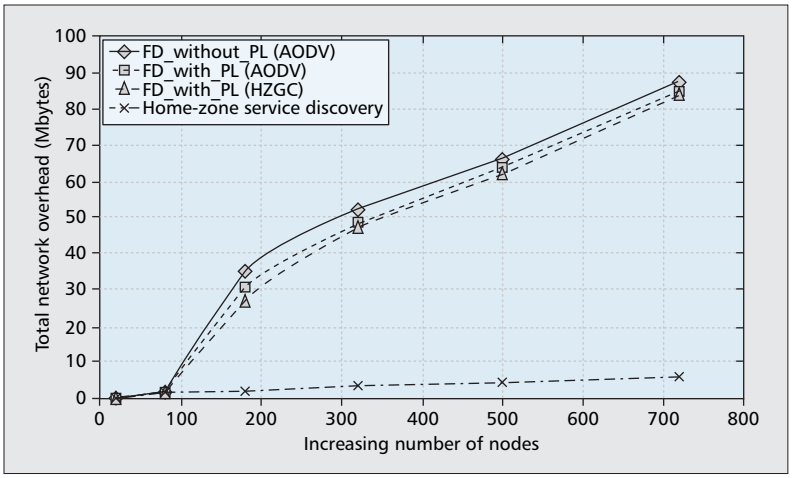

Figure 3. *Total service discovery control overhead incurred as a function of increasing node count.*

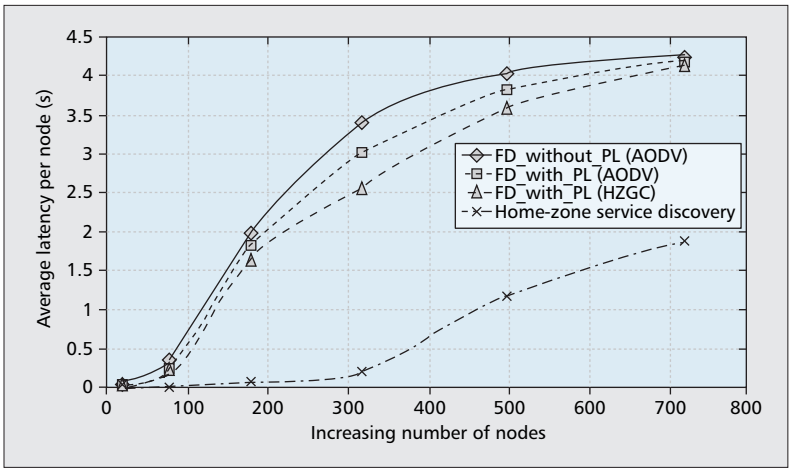

Figure 4. *Average service discovery latency as a function of increasing node count.*

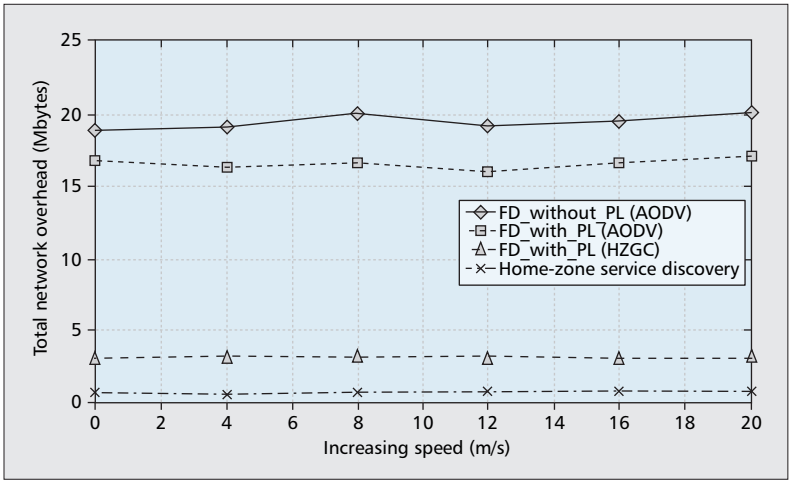

Figure 5. *Total service discovery control overhead incurred as a function of increasing average node speed.*

Figure 3 depicts the total overhead incurred as part of the service discovery mechanism as a function of increasing number of nodes for all the four schemes. The home-zone service discovery mechanism performs better than other three fully distributed approaches, namely, FD_without_PL(AODV), FD_with_PL(AODV), and FD_with_PL(HZGC). While the incurred control overhead can be minimized by enabling promiscuous listening, the actual performance is badly affected by their reliance on flooding. As a result, these fully distributed approaches have very high increase in the total overhead incurred when the number of nodes increases beyond 100. Due to the inherent working mechanism of the underlying DCF-based MAC of IEEE 802.11, when a collision occurs, the nodes try to retransmit, creating even more traffic, which saturates the network rapidly. Since our home-zone service discovery mechanism and its associated routing protocol (HZGC) do not rely heavily on flooding, their combined performance is excellent. Figure 4 shows the average latency involved for a successful service discovery as a function of increasing node count. Although Fig. 4 appears to follow the same pattern of Fig. 3, it should be noted that Fig. 4 measures the average latency involved for a successful service discovery. In this respect, it does not show the actual performance; for example, if there is no successful service discovery, the latency appears to be zero.

Finally, Figs. 5 and 6 depict the network overhead and latency as a function of node mobility. In this case, the number of nodes and the terrain area were kept at constant values that took 300 and 1000×1000 m^2, respectively. While keeping the other simulation parameters the same as the previous case, the maximum node speed was increased from 0 to 20 ms^{-1}. It is evident that our fully distributed approach performs substantially well when used with the home-zone-based location routing protocol. The same is not true when it is used with AODV, although the performance of FD_with_PL(AODV) is better than that of FD_without_PL(AODV). Our home-zone service discovery mechanism continues to have a better overall performance than the fully distributed approach in terms of the total overhead incurred and the average latency involved.

Conclusion

This article has presented and evaluated two service discovery mechanisms that can be used in future ubiquitous communication environments. As shown, the fully distributed service discovery mechanism, although preferred, does not scale well due its mere reliance on flooding. On the other hand, our second service discovery mechanism scales well and performs better when compared to the fully distributed approach. Although we assumed the availability of task graphs (TGs) in our current work, a more generic way of representing a distributed task/application in terms of TGs needs to be devised. Hence, in the future, we will concentrate on devising a generic way to

represent a distributed task/application in terms of a TG and also use contextual information in the discovery of the best possible (optimal) nodes to invoke the service. This renders itself as the initial work towards realizing autonomic self-managed ubiquitous environments.

Acknowledgment

The work presented in this article was carried out in the context of the UK Programmable Ad hoc Networks (PAN) EPSRC project — GR/S02129/01.

References

[1] P. Basu, "A Task-Based Approach for Modeling Distributed Applications on Mobile Ad hoc Networks," Ph.D. thesis, Boston University, Boston, MA, May 2003.
[2] P. E. Engelstad, and Y.Zheng, "Evaluation of Service Discovery Architectures for Mobile Ad Hoc Networks," *Proc. IEEE Wireless On-Demand Network Systems and Services*, Jan. 2005, pp. 2–15.
[3] Y. Yuan, and W. Arbaugh, "A Secure Service Discovery Protocol for MANETs," *Proc. IEEE Symp. Pers., Indoor and Mobile Radio Commun.*, Sep. 2003, pp. 502–06.
[4] U. C. Kozat and L. Tassiulas, "Network Layer Support for Service Discovery in Mobile Ad Hoc Networks," *Proc. IEEE INFOCOM*, vol. 22, no. 1, Mar. 2003, pp. 1965–75.
[5] H. Koubaa and E. Fleury, "A Fully Distributed Mediator Based Service Location Protocol in Ad Hoc Networks," *Proc. IEEE GLOBECOM*, Nov. 2001, pp. 2949–53.
[6] C. Lee *et al.*, "Konark: A System and Protocols for Device Independent, Peer-to-Peer Discovery and Delivery of Mobile Services," *IEEE Trans. Sys., Man, and Cybernetics*, vol. 33, no. 6, Nov. 2003, pp. 682–96.
[7] J. Jeong, J. Park, and H. Kim, "Service Discovery Based on Multicast DNS in IPv6 Mobile Ad-hoc Networks," *Proc. IEEE VTC 2003-Spring*, pp. 1763–67.
[8] S. Sivavakeesar, and G. Pavlou, "Scalable Location Services for Hierarchically Organized Mobile Ad hoc Networks," *Proc. ACM MobiHoc*, 2005, pp. 217–28.
[9] UoB-JAdhoc New Release, v. 0.2, http://www.aodv.org

Biographies

Siva Sivavakeesar (S.Sivavakeesar@surrey.ac.uk) finished his M.Sc. in communication networks and software engineering with distinction at the University of Surrey (UniS), United Kingdom, in 2001. After completing his Ph.D. at the same institution, he is currently employed as a post-doctoral research fellow at UniS, where he has been carrying out research in the areas of quality of service (QoS) support in mobile ad hoc networks (MANETs), programmable MANETs for self-alignment purposes, autonomic communication, and computing systems. For additional information, including his recent publications, see http://www.ee.surrey.ac.uk/Personal/S.Sivavakeesar/.

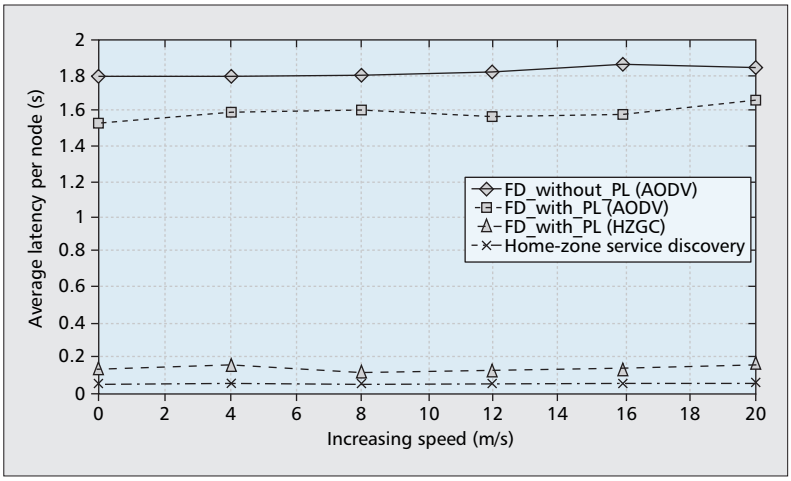

Figure 6. *Average service discovery latency involved as a function of increasing node speed.*

Oscar F. Gonzalez received a B.Eng. degree in electronic engineering from the National University of Colombia in 2002 and an M.Sc. degree in communications networks and software from the University of Surrey in 2006. He is currently pursuing a Ph.D. degree in autonomic communications at the University of Surrey. His research interests lie in the areas of detection of malicious nodes, self-protection, and trust building in ubiquitous systems.

George Pavlou is head of the Networks Research Group of the Centre for Communication Systems Research at the University of Surrey. Over the last 15 years he has been undertaking and directing research in the areas of communications with emphasis on network dimensioning, traffic engineering and management, quality of service, multimedia service control, mobile ad hoc networks, programmable networks, and communications middleware. He has managed several research projects and contributed to ISO, ITU-T, IETF, TMF, and OMG standardization activities.