Sheikh I. Ahamed, Mohammad Zulkernine, and Suresh Anamanamuri, "A Dependable Device Discovery Approach for Pervasive Computing Middleware," *Proceedings of the International Conference on Availability, Reliability and Security (AReS), IEEE CS Press*, Vienna, Austria, April 2006, pp. 66-73.

A Dependable Device Discovery Approach for Pervasive Computing Middleware

Sheikh I Ahamed[1], Mohammad Zulkernine[2], and Suresh Anamanamuri[3]
[1, 3]*Dept. of Math., Stat. & CS, Marquette University, Milwaukee, WI 53201-1881, USA*
[2]*School of Computing, Queen's University, Kingston, Ontario, Canada K7L 3N6*
{iq@mscs.mu.edu, mzulker@cs.queensu.ca, and sanamana@mscs.mu.edu}

Abstract

Distributed applications and middleware services targeted for mobile devices must use device discovery service to provide any kind of service to other devices. Device discovery algorithms developed for wired networks are not suitable for mobile ad-hoc networks of pervasive computing environments. This research proposes a dependable device discovery mechanism for the middleware of the applications consisting of rapidly reconfiguring mobile devices. Our approach offers a comprehensive solution to potential problems that can arise in highly adaptive mobile ad-hoc networks of pervasive computing environments. The approach is robust enough to accommodate the device limitations and rapid changes in the resource strengths of each device in the network. We present three new device discovery algorithms in this paper: a window based broadcasting algorithm, a connectivity based dynamic algorithm, and a policy-based scalable algorithm. The algorithms vary in complexity and efficiency depending upon the pervasive computing applications. We identify the desirable dependability related characteristics of device discovery services and present how our algorithms realize those characteristics. Experimental results are presented to compare and contrast the algorithms.

Keywords: Device discovery, Middleware services, Pervasive computing, Dependability, and Availability.

1. Introduction

With the increase of hand-held devices for pervasive computing environments, the use of short-range ad hoc networks has become widely prevalent. The wireless ad-hoc networks are established and maintained by the self organizing behavior of the participating devices without any central administration or fixed infrastructure [1]. Bluetooth, IrDA, and 802.11(b) are some of the wireless communication technologies that are being used to establish a short-range ad-hoc network of devices that are autonomous, heterogeneous, and resource-constrained. Mobile devices in an ad-hoc network are battery-powered and they have smaller physical storage and limited processing capabilities.

Device Discovery (DD) is a service using which a device can detect all its neighbors and make its presence known to each neighbor in the network. DD is the core service in any middleware designed for pervasive computing since every application targeted for pervasive computing requires the knowledge of the devices present in the network. Device discovery enables the distributed applications to manage the communication between devices based on their availability in the network. Since pervasive computing applications rely heavily on the DD, it should be reliable, fault-tolerant, adaptive, and optimized for resource consumption. Reliability and fault tolerance for DD imply that a device that joins the ad-hoc network should be able to obtain the list of devices present in the network under all circumstances. DD should adapt itself to the changes in the ad-hoc network because of devices failing, leaving, or joining at any time. All of the above tasks need to be attained with optimum use of resources like power and memory.

Research on middleware for ad-hoc networking have concentrated on services such as routing and mobility, while the focus on device discovery research has been intensified only recently. The presence of different communication technologies posed difficulties in establishing a standard DD mechanism. Network initialization and DD in ad-hoc networks in which devices do not have MAC or unique address have been addressed in [1]. DD in Bluetooth where each device simultaneously listens on multiple frequencies has been dealt with in [6, 7, 8]. *However, all the previous related research have been carried out at the MAC layer, which are not useful as a middleware service.* While the DD issue has been handled by most of the work during the initialization of the network, not much have been discussed about the adaptability of the service to constant changes in ad-hoc networks. The issue of adaptability of devices and DD was addressed partially in [10]. It is well accepted that DD should be resource-optimized, but earlier research have not focused on this crucial aspect of DD. To the best of our knowledge, there is no work on implementing a middleware service that provides

adaptive, reliable, fault-tolerant, and resource-optimized DD.

Our objective is to develop dependable DD as a middleware service that would be used by other services and higher-layer applications. We present three algorithms in this paper to design DD service: a window based broadcasting algorithm, a connectivity based dynamic algorithm, and a policy-based scalable algorithm. The algorithms are dependable, *i.e.*, reliable, fault-tolerant, resource optimized, and adaptable to network changes due to the failure or removal of a device of the network. While supporting these traditional requirements, we also ensure that due importance is given to the issues such as reduced network traffic and battery consumption.

The device discovery service module that we designed will be incorporated as a core service in our custom middleware MARKS [18] for mobile ad-hoc networks as shown in Figure 1. Applications and middleware services use the device discovery module that contains three different algorithms, to obtain device lists using our provided API without any extra overhead. Figure 2 shows the components of our device discovery service module along with the three device discovery algorithms. Each algorithm differs from the other in aspects such as resource-optimization and the time taken for device discovery. It might be critical for applications designed for ephemeral wireless network with very short time for collaboration, to discover devices quickly and with reasonable resource-optimization. While applications designed for networks that exist for long times, would require comprehensive resource optimization. Some networks might be stable, and applications in that network would like to use an algorithm that does a better resource optimization in a scenario where no device is joining or leaving the network for a long period of time. Since application layer has better knowledge about its requirements, it would use the algorithm that best satisfies its objectives.

Section 2 discusses some important issues to be considered while designing a DD service. Section 3 describes the related work. In Section 4, we describe three novel DD algorithms to be used in a middleware service. Section 5 provides experimental evaluation of our approach through analysis of real-time data. In Section 6, we conclude our presentation with a few words on future work.

2. Characteristics of Dependable DD Service

An ad-hoc network with mobile devices in Pervasive computing should have a robust device discovery service with the following characteristics C1-C5.

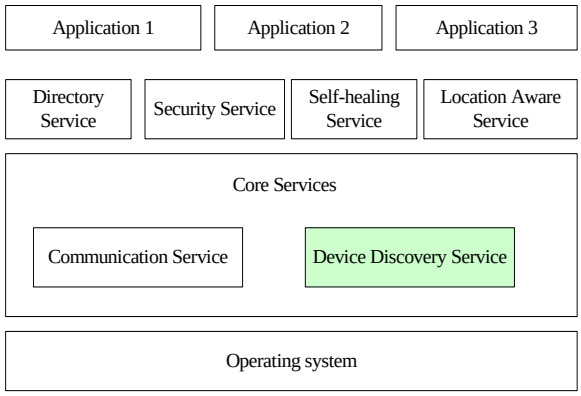

Figure 1. Middleware hierarchy for pervasive computing [18]

C1. Adaptability: The devices should self-configure themselves since devices are autonomous and will join and leave network at their wish. Devices leaving the network must realize that their list of discovered devices is redundant and other devices in the network should update their lists by removing the devices that left the network. The frequency with which individual devices update their lists determines the accuracy of the DD service.

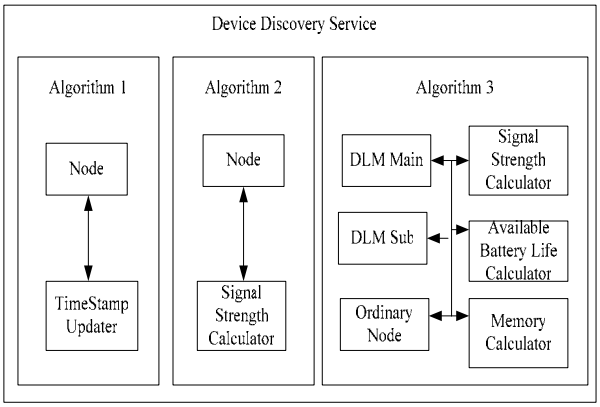

Figure 2. Device discovery service module

C2. Reliability: The DD service needs to be reliable, which means it should continue to work and maintain accuracy even when devices in the network fail or leave the network. The handling of this issue requires decisions on the scenarios in which DD will cause all the devices to re-initialize the entire network or only a few devices affected.

C3. Fault-Tolerance: Devices that fail sometimes give rise to critical errors in the expected functioning of a

DD service. The DD service needs to have efficient mechanism for replication of information and transparent hand-off responsibilities among devices when faults occur in the network.

C4. Bandwidth-Optimization: It is vital to maintain low traffic in the network for efficient device discovery. Devices need to create, send, receive, and process the packets in the network. This implies more packets require more computation and power consumption. Hence, DD algorithms should have efficient methods to achieve all the above characteristics with less bandwidth.

C5. Power-Optimization: All the features mentioned above must be attained with least possible power consumption. Battery is the most critical resource on mobile devices and greater network traffic implies greater processing done by both the senders and the receivers. Battery power can also be reduced by investigating means to maintain accurate discovered device lists, when devices participate intermittently.

3. Related Work

There are a number of protocols and design architectures that provide the DD in mobile ad-hoc networks [1, 5, 6, 7, 10]. These protocols are designed to work with diverse communication technologies such as Jini, Bluetooth, and 802.11. However, all the existing protocols aim at discovering devices at the MAC and network layer without providing application interface for middleware services to access the list of discovered devices. Hence, those are not useful for pervasive computing application developers.

Discussion about power and bandwidth-optimized (C4 & C5) DD in networks with devices that may not even have unique ID is presented in [1]. This work also focuses on collision avoidance and recovery using randomized distributed algorithms. The process of electing a leader that performs the device discovery by detecting new devices joining the ad-hoc network is also explained in detail. However, [1] does not satisfy adaptability (C1) since the protocol does not update the discovered devices when existing devices leave the network. As a result, list of discovered devices will be inaccurate in mobile ad-hoc networks where devices join and leave network at will. Also the protocol suggested in that paper, does not focus on fault-tolerance (C3) when the leader fails. This presents a single point failure of the proposed device discovery protocol. Hence, the approach presented in the paper is not suitable for dynamically self-configuring mobile ad- hoc networks since it does not achieve important characteristics C1 and C3. Our algorithms handle all the above mentioned issues in an efficient way.

SLP, UPnP [15], Jini [16], and OSGi [17] are different technologies that perform resource discovery in ad-hoc networks, with emphasis on accuracy of discovered resources. However, the technologies perform poorly with respect to C4 and C5 in network**.** Although, each one of these technologies agree that efficient management of the resources of the portable devices is crucial, resource optimization is not given due importance. Our approach ensures that the devices in the network adapt to the changes taking place with optimum use of their resources.

The DD in networks that use Bluetooth as communication technology has been discussed in [6, 7, 8]. The challenges in Bluetooth multiply because the frequency hopping system by which the devices listen on different frequencies and being physically close to each other will not result in discovery of two devices. [6] achieves device discovery using inquiry and paging mechanism and [8] enhances the inquiry mechanism. [7] discusses about device discovery in multi-hop ad-hoc networks. Bluetooth specification inherently supports a master and slave concept that result in a leader serving the network. The protocols discussed above do not implement characteristics such as fault-tolerance (C3)**,** bandwidth-optimization (C4)**,** and power-optimization (C5). [10] addresses the issue of adaptability (C1) of devices in DD service. However, this work does not implement required DD characteristics C3, C4, and C5.

4. Device Discovery Algorithms

We have used the mobile ad hoc network (MANET) mode of 802.11(b) protocol in our ad-hoc wireless network. Each device has a unique name by which it is identified in the network. We have used the IP Address of the device to identify it in the network. The discovered devices are stored in a list called DeviceList.

4.1. Algorithm 1: Window based broadcasting approach

In this algorithm, a simple and basic device discovery method is implemented where each mobile device in the network would broadcast a beacon to all the devices in the network at regular intervals of time. In this algorithm, the devices in the network continuously broadcast data within the window of four seconds. Any window size could be used but four seconds is a reasonable assumption since it takes one second for a device to respond. Note that Node, Personal Digital Assistant (PDA) and Device are used

interchangeably in this paper. The different steps in the execution of Algorithm 1 are as follows:

Step1: A device joining a network broadcasts its information every two seconds. Other devices in the network hear about this device and add that device to their DeviceList.

Step2: Other devices already in the network also broadcast their information after every two seconds. The new dev ice listens to these broadcasts and updates its DeviceList with the devices from which it received signals.

Step3: When a device leaves a network, other devices in the network will not pick up the signals it broadcasts.

Step4: Each device updates its list by reading DeviceList at regular intervals and a device that did not send a signal for more than four seconds would be removed from the list.

Step5: Since the devices in the network continuously broadcast data within the window of four seconds, the previous step removes only the devices that left the network.

4.2. Characteristics of Algorithm 1

This algorithm achieves characteristics C1, C2, and C3 of DD service. When a few devices fail or leave the network, the beacons they send will not be received by the other devices in this network. Therefore, all the other devices in this network will remove those devices from their DeviceList after two consecutive intervals. This results in the algorithm being adaptive (C1) since it transparently reconfigures the DeviceList on each device, whenever topology changes. Algorithm 1 is reliable (C2) since every device maintains an accurate list of devices in the network at all times. Each device is independent and is responsible for maintaining its own DeviceList because of which the algorithm is fault-tolerant (C3) and continues to function accurately even when any device fails in the network.

4.3. Algorithm 2: Connectivity based dynamic approach

In Algorithm 1, all the devices will require to send beacon packets at regular intervals of time. This would increase the power usage because of processing done to create, send, and receive these packets. Algorithm 2 is an improvement over Algorithm 1 by eliminating the repeated broadcast of beacon packets. In Algorithm 2, the devices joining the network will send beacons only in the beginning. The only communication that can take place from hereon is when a device is leaving the network. To implement this mechanism, a device uses MAC layer to compute its signal strength with respect to the entire wireless network. If the signal strength decreases beyond a critical value the device would understand that it is going beyond the acceptable proximity range for the network and would send a "LeavingNetwork" beacon to all the devices in the network. As always the beacon would consist of IP Address. When the remaining devices receive this beacon, they remove the entry containing that IP Address from their respective DeviceList. Hence, the algorithm is called connectivity based dynamic algorithm.

Every device broadcasts a beacon containing IP Address and hostname during network initialization or the first time when it enters the network. After the initial broadcast the device will keep listening to the incoming data and will store the IP Address and hostnames of devices that sent "JoiningNetwork" beacons. The device understands the difference between two different signals through a particular bit called message type. If the device is joining network it is added to the DeviceList. Every device in the network thus detects the new device. Next step lies in informing the new device about the devices present in the network. Hence, each device in the network would now send a beacon containing their IP Address and hostname to the new device. However, if the "JoiningNetwork" bit is not set, it implies that the new device should not reply to these beacons. The new device would then read the messages it received and would update its DeviceList with the devices which sent the messages. If a device is leaving the network, it sends a "LeavingNetwork" beacon based on signal strength for sudden leaving that results in the device's IP Address being removed from the rest of the device's DeviceList lists.

The different steps in the execution of Algorithm 2 are as follows**:**

Step1: A device joining the network broadcasts its information.

Step2: Other devices present in the network store the new device in their list of discovered devices and send their information to the new device.

Step3: Each device monitors its signal strength with respect to the network. If the signal strength is less than a critical value, it means that the device is leaving the network.

Step4: Before leaving the network, a device will broadcast to the network that it is leaving the network.

Step5: Other devices in the network will update their list of devices by removing this device from their DeviceLists.

4.4. Characteristics of Algorithm 2

Algorithm 2 is a great improvement in terms of network traffic and battery power consumption in comparison with Algorithm 1. The network traffic grows linearly with the number of devices in the network. When there are 'm' devices in the network and 'n' devices joining the network, there will be 'm+n' number of signals sent and when 'n' devices leave the network there will only be 'n' signals. There will be no network traffic related to DD when the network is in equilibrium (*i.e.*, no new devices are joining the network and no existing device is leaving it) and it is here that this algorithm scores high when compared to the previous algorithm.

This algorithm is adaptive (C1) so that when devices leave or fail, the DeviceList of other devices are updated accurately. This device discovery service is highly reliable (C2) since every device will have its own DeviceList and will not depend on other devices. Accuracy of this approach depends on the signal strength calculation. This algorithm also supports fault-tolerance (C3) since there is no single point failure and the service will continue to function even when a few devices fail or leave the network. Bandwidth-optimization (C4) is achieved by decreasing the network traffic in the network. There is burst traffic only when new devices enter the network and even in that case, this algorithm has less network traffic. The amount of processing that the devices need to do is also highly reduced since the number of packets created, sent, and received is less, resulting in reduced power consumption.

Listening for packets continuously involve scanning the network for signals every second. Power consumption as a whole in the network could be saved if devices are able to sleep and stop listening, yet not losing the accuracy described in the previous paragraph. In Algorithm 1 and Algorithm 2, the devices are required to listen continuously even when the network is in equilibrium. In Algorithm 1, if a device sleeps, it will not be able to send or receive the beacons sent by other devices and as a result may risk being undiscovered by other devices in the network. With Algorithm 2, devices can sleep when the network is in equilibrium, but there is no way that the devices can know when a network is in equilibrium. Hence, if a device sleeps even for a small time, it will miss the signals sent by devices joining and leaving the network and this will render the DeviceList inaccurate. Thus, Algorithm 1 and 2 prevent devices from sleeping.

4.5. Algorithm 3: Policy-based scalable approach

In this algorithm, each device will have certain role to play in the network. A device can either be a Device List Maintainer (DLM), a DLM Substitute (DLMSub), or an ordinary device. DLM is a central device with the highest resource availability that is elected initially through a process called as DLM resolution. DLM keeps track of the list of devices in the network. After a DLM is elected, it broadcasts a signal informing the other devices in the network, that it is the DLM and thus causing the other devices to break out of their DLM resolution process and assume the role of the ordinary devices. DLM uses the information received from different devices in the network to elect DLMSub, which is a device with the highest resource availability from the remaining devices. The role of DLMSub is to check the state of DLM and make sure that DLM is functional. If DLM

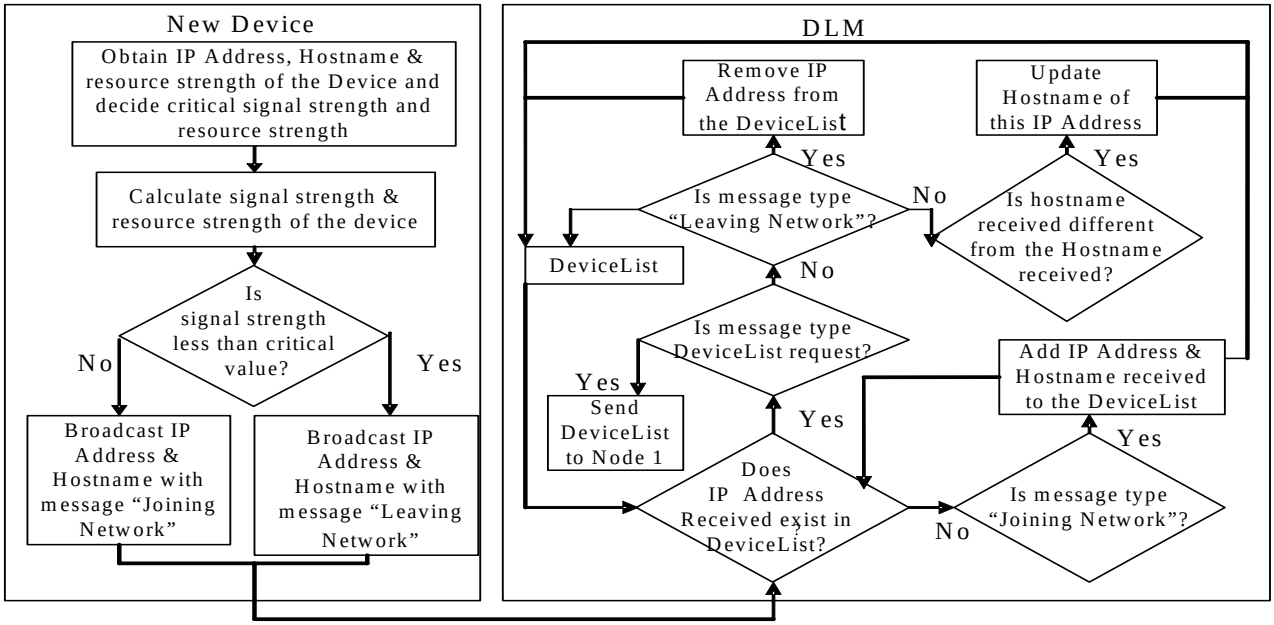

Figure 3. Joining of a new device in algorithm 3

crashes then DLMSub will have to take up the task of maintaining the DeviceLists. After DLM and DLMSub are elected all the other devices in the network are termed as ordinary devices. The ordinary devices in the network will store the IP Address of the DLM and request DeviceList from DLM when required. The only responsibility of these devices is to respond to the messages from DLM and DLMSub and facilitate the DLM resolution process. They monitor their signal strength continuously and would inform the DLM when they are leaving the network. These devices will query the DLM for the device list when required. If the query of a device is not served after more than two requests, then it will assume that it needs to participate in DLM resolution and broadcast its information to all the devices in the network.

Figure 3 shows communication between a new device in the network and the DLM in Algorithm 3. A device that enters a new network starts in DLM resolution mode. If a DLM is present in the network, it would send a signal to the new device, informing that it is DLM, and terminating the DLM resolution mode of the new device. The new device stores DLM and would request the DeviceList from DLM when required. There should be a policy that defines the frequency with which each device requests the device lists from DLM. When a device is about to leave the network it would inform the DLM about it. DLM updates the list by removing the device from the list of discovered devices. Also, when there are large numbers of devices in the network, it is feasible to create more than one DLM to reduce the burden on a single device. Hence, it is scalable.

The different steps in the execution of Algorithm 3 are as follows:

Step1: Every device broadcasts its IP address, hostname, and resource availability to elect a DLM.

Step2: Devices with the highest and the second highest resource availability are elected as DLM and DLMSub respectively. Rest of the devices (ordinary devices) request DeviceList from DLM as required.

Step3: A new device starts in DLM resolution mode. If a DLM is already elected, it terminates the DLM resolution process of the new device. The new device then requests DeviceList from the DLM.

Step4: When a device is about to leave the network (using signal strength) it would inform the DLM about it. DLM updates the list by removing the device from the list of discovered devices.

4.6. Characteristics of Algorithm 3

In Algorithm 2, each device maintains its own copy of DeviceList. Although this eliminates a single point of failure, memory used through out the network for DD can be reduced if a single device can maintain the list of devices and serve the requests for DeviceList from other devices in the network. This is precisely what the third algorithm achieves, while preserving the advantages of the first and the second algorithms. There is a single copy of DeviceList in the network at any point of time and all the other devices request the DLM for the DeviceList. A single DeviceList for the entire network will also result in conformity and accuracy since all the devices will have the same copy of DeviceList at a given instant of time.

Also, the previous algorithms required the devices to be active to have accurate DeviceLists. Here the devices can sleep for small amounts of time and yet have accurate DeviceLists from DLM. The device after waking up will poll the DLM to make sure that the DLM did not change. If DLM changes when the device was asleep, the device will not get any reply from the remote device. The device would then broadcast a query to which the DLM of the network will respond.

However, having a single leader brings many complexities inherent to the mobile ad-hoc networks mentioned earlier. We need to ensure that the device discovery process does not degrade when the DLM leaves the network or simply fails at any point of time. When DLM leaves the network the algorithm ensures that the devices in the network adapt (C1) themselves using the “LeavingNetwork” beacon sent by DLM or DLMSub. However, ordinary devices will be unaware if DLM or DLMSub fail unexpectedly. Unexpected failure of DLM is termed as fault and faults are likely to happen in mobile ad-hoc network. Transparent fault-recovery (C1) mechanism is implemented by the algorithm, by having the DLMSub recognize and handle the failure of DLM when the latter does not answer a probe beacon. In a similar way, DLM would recognize DLMSub failure when it does not receive probe requests for a certain time. Algorithm 3 is also optimized for performance using less network traffic and battery-power.

Since each device will have to query the DLM to obtain the latest DeviceList, this algorithm requires extra time to serve the higher-level applications. This algorithm performs very well in comparison to the other two algorithms when the ad-hoc network is being formed for a long time rather than for only a few minutes. This algorithm requires a lot of processing before DLM resolution stage but very little after a DLM is elected. The time and processing invested in electing a DLM is significant for mobile ad-hoc networks, and this investment would be justified only when the network is operational for longer spans of time. There should be a policy that defines the frequency with which each device requests the device lists from DLM. Also, when there are huge numbers of

devices in the network, it is feasible to create more than one DLM to reduce the burden on a single device.

5. Experimental Evaluation

We have implemented the algorithms described above on Dell Axim 50v PocketPCs using C# as programming language on .Net Compact Framework platform. We have calculated signal strength using the modules developed by OpenNetCF.org. We have used 802.11(b) protocol for communication between devices since it has a larger area of coverage. We have experimented with the limited devices at our disposal and collected real time data for each of the three algorithms. In this section, we analyze the data collected and will try to deduce some interesting conjectures.

We have measured the amount of traffic caused by each algorithm in a wireless network consisting of 3 PDAs. The three PDAs are present in the network from the beginning and never leave the network. Every device discovery algorithm detected the three devices in the network. Figure 4 shows comparisons of performance of the three algorithms with respect to the cumulative number of bytes that passed between devices during device discovery at different points in time. Time is calculated in seconds and data in KBs. There is significant traffic in the network when Algorithm 1 is used for device discovery since individual devices continue to send data packets at regular intervals. With Algorithm 2, there are no additional data packets being sent into the network after initial broadcast. Hence, the total number of bytes remained same. With Algorithm 3, the number of bytes used for device discovery stabilizes once DLM and DLMSub are elected. DLM and DLMSub election requires back and forth communication between devices and hence higher number of bytes in the network. However, in a completely dynamic ad-hoc network, where lot of devices join and leave network continuously, Algorithm 3 performs better than Algorithm 1 and Algorithm 2.

Power consumption of devices is another parameter which we measured for our algorithms. The comparison of cumulative power consumption in the network calculated at intervals of four minutes for the three algorithms is shown in Figure 5. The power consumed is measured in milli watts. As with network traffic, power consumption in the network is also more when Algorithm 1 is used since more data packets had to be processed by each device. The power consumption of devices in the network is stable when Algorithm 2 and Algorithm 3 are used since the network is in equilibrium, and there is not much processing to do.

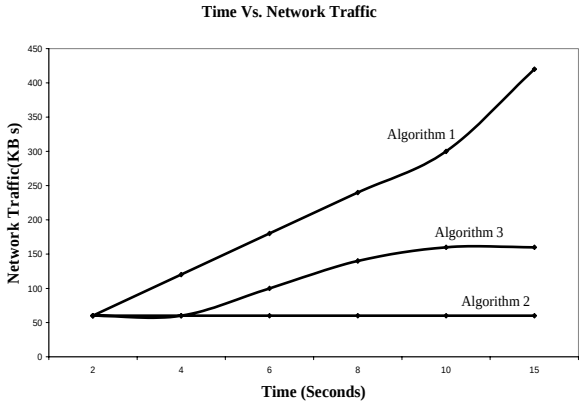

Figure 4. Performance w.r.t. cumulative network traffic

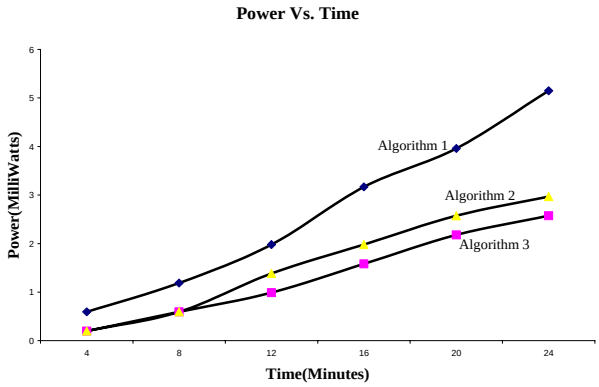

Figure 5. Performance w.r.t. cumulative power consumption

Table 1. Criteria for an application to use the algorithms

Algorithms	*Suitable applications*
Algorithm 1: Window based broadcasting	• No signal strength support from MAC layer. • Runs for extremely short period.
Algorithm 2: Connectivity based dynamic	• Participation of each device in the network all the time. • Hard real-time applications.
Algorithm 3: Policy-based scalable	• Infrequent participation from devices. • Runs for long period.

6. Conclusion and Future Work

In this paper, we have discussed the essential characteristics of device discovery service, which are very important for pervasive computing environments. We have presented three algorithms for device

discovery: a window based broadcasting algorithm (Algorithm 1), a connectivity based dynamic algorithm (Algorithm 2), and a policy-based scalable algorithm (Algorithm 3), which reside in the core service of Figure 1. These DD Algorithms operate at middleware level compared to other previously developed DD algorithms [1, 6, 7, 8] that operate at MAC layer. The DD algorithms are dependable, *i.e.*, the algorithms are resource optimized, adaptable to network changes such as a device failing or leaving the network, reliable, and fault-tolerant.

Each of the algorithms presented in this paper has some advantages over the others depending on the requirements of an application. A DD algorithm can be selected from the core service of Figure 1 by an application based on a set of criterion mentioned in Table 1. Window based broadcasting algorithm can be used when the MAC layer does not support the computation of signal strength. It can also be used by applications that run for extremely short period since it requires extremely low time for network initialization. The connectivity based dynamic algorithm is useful for applications that require the participation of each device in the network all the time. Since the device lists are stored in every device, the applications that require prompt response by each device can use this algorithm. It is useful for hard real-time applications. Policy-based scalable algorithm is useful for applications that require the participation of devices, but not at all times. It gives better power-performance, hence it is useful for applications which execute for long times. The proposed protocols will not be suitable for fast moving devices (like in vehicular ad-hoc networks) without lower network layer support.

In future, we will simulate our device discovery service with more number of devices to collect data on the scalability of the algorithms although our Algorithm 3 is inherently scalable. We plan to use our device discovery service to build higher middleware services such as directory, routing, and location aware services along with security.

References

[1] P. Popovski, T. Kozlova, L. Gavrilovska, and R. Prasad, "Device Discovery in Short-Range Wireless Ad Hoc Networks," *IEEE network*, pp.1361-1365, 2002.

[2] O. Ratsimor, D. Chakraborty, A. Joshi, T. Finin, and Y. Yesha, "Service Discovery in Agent-Based Pervasive Computing Environments," *Mobile Networks and Applications*, vol. 9, no. 6, pp. 679-692, Dec. 2004.

[3] P. Basu and T. Little, "Wireless Ad Hoc discovery of Parking Meters," http://lcawww.epfl.ch/luo/WAMES%202004_files/wames_Parkinng%20Meters.pdf.

[4] M. Satyanarayanan, "Pervasive Computing: Vision and Challenges," Personal Comm., vol. 8, no. 4, pp. 10-17, Aug. 2001.

[5] A. Wils, F. Matthijs, Y. Berbers, T. Holvoet, and K. De Vlaminck, "Device discovery in residential gateways," *IEEE Transactions Consumer Electronics*, vol. 48, issue 3, pp. 478–483, Aug. 2002.

[6] G.V. Zaruba and V. Gupta, "Simplified Bluetooth Device Discovery – Analysis and Simulation," *Proceedings of the 37th Hawaii International Conference on System Sciences*, 2004.

[7] F. Ferraguto, G. Mambrini, A. Panconesi, and C. Petrioli, "A new approach to device discovery and scatternet formation in Bluetooth networks," *Proceedings of 18th International Parallel and Distributed Processing Symposium*, pp. 21, April 2004.

[8] G.V. Zaruba and I. Chlamtac, "Accelerating Bluetooth Inquiry for Personal Area Networks," *Proceedings of Global Telecommunications Conference*, IEEE Volume 2, pp.702 – 706, Dec. 2003.

[9] T. Pering, V. Raghunathan, and R. Want, "Exploiting radio hierarchies for power-efficient wireless device discovery and connection setup," *Proceedings of 18th International Conference on VLSI Design*, pp.774–779, Jan. 2005.

[10] K. Sohrabi, J. Gao, V. Ailawadhi, and G.J. Pottie, "Protocols for Self-Organization of a Wireless Sensor Network," *IEEE Personal Communications*, vol. 7, Issue 5, pp. 16 – 27, Oct. 2000.

[11] A. Akkaya and M. Younis, "A survey on Routing Protocols for Wireless Sensor Networks," *Proceedings of the IEEE Wireless communications and Networking Conference (WCNC)*, Orlando, FL, March 2002.

[12] K. Nakano and S. Olariu, "Random Initialization Protocols for Ad-hoc networks," *IEEE network*, vol. 15, no. 5, pp. 28-37, 2001.

[13] E. Hall, D. Vawdrey, and C. Knuston, "RF Rendez-Blue: Reducing Power and Inquiry Costs in Bluetooth-Enabled Mobile Systems," *Proc.of Eleventh Int. Conf. on Computer Communications and Networks*, pp. 640-645, Oct. 2002.

[14] I. Chlamtac, C. Petrioli, and J. Redi, "Energy conserving access protocols for identification networks," *IEEE ACM Transactions on Networking 7(1)*, pp. 51-59, 1999.

[15] Universal Plug and Play, "Understanding Universal Plug and Play: a White Paper," June 2000, http://upnp.org/resources/whitepapers.asp.

[16] Sun Microsystems, "JINI technology Architectural Overview," *http://www.sun.com/jini*

[17] P. Dobrev, D. Famolari, C. Kurzke, and B. Miller, "Device and Service Discovery in Home Networks with OSGI," *IEEE Communications Magazine*, vol. 40, issue 8, pp. 86 – 92, Aug. 2002.

[18] M. Sharmin, S. Ahmed, and S. I. Ahamed, "MARKS (Middleware Adaptability for Resource Discovery, Knowledge Usability and Self-healing) for Mobile Devices of Pervasive Computing Environments," To appear in the *Proc. of Third Int. Conf. on Information Technology : New Generations (ITNG)*, Las Vegas, USA, April 2006.