Service Discovery in Pervasive Computing Environments

Michael S. Thompson

Final Examination submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Computer Engineering

Dr. Scott F. Midkiff, Chair
Dr. Luiz A. DaSilva
Dr. Ing-Ray Chen
Dr. Thomas L. Martin
Dr. William T. Baumann
Dr. George E. Morgan

July 20, 2006
Blacksburg, Virginia

Keywords: Service Discovery, Pervasive Computing, Ubiquitous Computing, Service Location, Resource Discovery, Resource Location

Service Discovery in Pervasive Computing Environments

Michael S. Thompson

(ABSTRACT)

Service discovery is a driving force in realizing pervasive computing. It provides a way for users and services to locate and interact with other services in a pervasive computing environment. Unfortunately, current service discovery solutions do not capture the effects of the human or physical world and do not deal well with diverse device populations; both of which are characteristics of pervasive computing environments.

This research concentrates on the examination and fulfillment of the goals of two of the four components of service discovery, service description and dissemination. It begins with a review of and commentary on current service discovery solutions. Following this review, is the formulation of the problem statement, including a full explanation of the problems mentioned above. The problem formulation is followed by an explanation of the process followed to design and build solutions to these problems. These solutions include the Pervasive Service Description Language (PSDL), the Pervasive Service Query Language (PSQL), and the Multi-Assurance Delivery Protocol (MADEP). Prototype implementations of the components are used to validate feasibility and evaluate performance. Experimental results are presented and analyzed. This work concludes with a discussion of overall conclusions, directions for future work, and a list of contributions.

Acknowledgments

First and foremost I would like to thank my family. You have been more supportive over the course of my life than I could ever imagine. I have no clue where I would be without you. I hope you are as proud of me as I am thankful of you.

Next, I would like to thank my advisor, Dr. Midkiff, and my committee. I have enjoyed the wonderful chance to work with you all over the years and I have learned a lot professionally and personally. Dr. Midkiff, you have been a mentor, role model, and friend through the insanity of the past 4 years. Thank you many times over for your patience and help!

In all of this I cannot forget my friends, the folks I have been around on a day-to-day basis. First off, Josh and Woody, you guys have been in the trenches with me for most of my time at Tech and I truly value your friendship. Thanks for all of the crazy nights, the cheap beer, the expensive beer, the extravagant food, the advice, and the support. I have a learned a lot from you guys. We seem to have done a good job keeping each other away from the brink. Next is Kim Baker who has made the past year of my life much more enjoyable than I could have imagined. Kim, you have taught me an amazing number of things about myself, love, relationships, and women. I have thoroughly enjoyed the past year or so and I'm interested to see where the future takes us. Next up would be my list of roommates, Ben Muuuuuuzal, Wes (big daddy wes), Josh (tall Josh, that is), Brad (the monkey), and, of course, the I-don't-know-how-in-the-world-I-got-such-a-hot-roommate Carrie (Brillo). To the folks in the networking area (both in Blacksburg and NoVa) you have served as great inspirations and friends. I've learned a lot about networking, but also an unbelievable amount about worldly

perspectives on everything from food to politics to religion. Thanks for broadening my horizons! Finally, there are a number of other folks I can't forget and would like to thank for just making my life more fun and interesting, Areej Saleh, Dave Lehn, the Joe "wild man" Adams, Juan Suris, Jason Hanson, Sarah Hicks, Kris Hall, Ingrid Burbey, Ryan Thomas, Vivek (I can't spell your last name), Unghee Lee, Haisung Wu, Dave Raymond, and Grant Jacoby. Strangely enough, I'd like to thank Gina Desiderio Edmison for being my nemesis for these years and teaching me a lot about perspective.

I probably would not be writing this today without the help of Jerry Darnell. He put me in front of a computer, showed me how to talk to it, and ignited a spark that has become one hell of a forest fire. Thanks again, Jerry D.

In all of this, I can not forget the National Science Foundation for funding 3+ years of my work and life with the IREAN program through an NSF IGERT grant (award DGE-9987586). Also, I would like to thank Microsoft Research for their funding of UbiShare and my trips to the Mobile and Embedded Developers conference for the past 3 years.

Hopefully that is everyone, but I am sure I overlooked a few. I wouldn't be who and where I am today without the the people that I know. Thanks to everyone!

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Today, computers and technology are weaving their way into our everyday lives. This ubiquity of technology is like nothing seen before. If electronics continue on the current trend of becoming smaller, faster, and easier to fuel, this technological ubiquity will continue integrating itself into the devices and world around us. The idea of embedding computation into the environment around us is called pervasive or ubiquitous computing [1]. A key component to making these enhanced devices more useful is control. Aiding in the control of these devices is communication. Just as with people, having numerous entities working individually will produce results. But, having the same number of entities sharing information and working cooperatively will likely produce better results in a more efficient manner. This is a key component of the pervasive computing movement.

1.1 Pervasive Computing

Pervasive computing came to life in the late 1980s and early 1990s as the brainchild of Mark Weiser of Xerox Palo Alto Research Center (PARC). In [1], Weiser talks of computing and technology offering the computational services we have come to enjoy in addition to

new services, all while being integrated into the world around us. This goal of distraction-free technology services is what the pervasive computing community strives for in research and development. While the actual definition and scope of pervasive computing are highly debated and continually expanding, the goal throughout the community remains constant, building unobtrusive technology and services to aid in tasks.

A key component of pervasive computing is communication. Communication is important because it provides a medium not only for direct control but, also, for information dissemination and acquisition. Collaboration between devices in pervasive computing environments means distribution of responsibilities, better understanding of the environment through greater context awareness, and combined efforts in accomplishing a task. Service discovery provides the means of location and interaction between devices, services, and users.

1.2 Service Discovery

A *service* is an interface to an application or device, by which a client may access that application or device. Services present in pervasive computing environments may vary from a display device, such as a network accessible wall display, to a stock quote retrieval service that keeps you up to date on your favorite stock values. While these services are diverse in kind, they share a common requirement. They both need to have some way for a user to locate and interact with them.

Locating and interacting with services is the job of *service discovery*. Service discovery can be broken down into the tasks of description, dissemination, selection, and interaction. Each component has a unique set of responsibilities in the service discovery process. Overall, the goal of service discovery is to provide the best service options, of the many that may be available, to the entity desiring the services and aid in communication with those services.

1.3 Problem Statement

In general, service discovery in pervasive computing environments is a difficult task. With more and more devices moving into the pervasive computing category, a large population of devices may be present in a given environment at any time. In addition, connectivity, especially global connectivity, removes the sense of location from the physical world. A device on the other side of the world can be accessed in the same manner as a device on the other side of the room. For these reasons, the number of devices that are accessible to a user is immense. It is the job of service discovery to discern viable choices out of the masses for use by a client.

Within the larger problem of service discovery, each component has its own set of problems. The description component must balance providing enough useful information with the cost of storing and processing information. Dissemination must efficiently distribute information, manage node resources, and deliver information in an acceptable amount of time while conserving resources. Selection must take the list of acceptable service options and determine which is the best option or options for completing the given task. Finally, interaction must provide a means of communication between the client and the services.

Service discovery is not a new problem. There have been numerous solutions for service discovery proposed by academic and industrial researchers, as described in Chapter 2. Pervasive computing environments pose new problems and requirements versus traditional environments. Current solutions lack awareness of characteristics of pervasive environments, such as diverse device architectures, physical location, and large service populations. As a result, nodes and networks may be overburdened and provide suboptimal performance.

1.4 Approach

The first step in this work is to define the environment and problem. This is done by devising pervasive computing usage scenarios. Further analyzing the scenarios, performance metrics are identified based on these scenarios. Using these metrics, a literature search and qualitative comparison of current solutions are conducted. Ideas from these solutions combined with new ideas are used to build new solutions. These solutions are evaluated and compared to existing solutions through simulation and experimentation.

1.5 Contributions

The major results of this research include a description and dissemination scheme designed to fulfill the requirements of service discovery in pervasive computing environments. These requirements include the ability to accommodate attributes of the real world, such as time and space, and awareness of the heterogeneity of devices used in pervasive computing environments. In route to these solutions a service discovery framework was designed to aid in the characterization and categorization of service discovery solutions. Also, a study of the IEEE 802.11b wireless local area network standard in ad hoc mode multicast and unicast performance showed that IEEE 802.11b multicast frames have higher packet loss and higher throughput than unicast frames due to MAC layer acknowledgements. This differs from wired networks, specifically IEEE 802.3 Ethernet, since acknowledgements are not used and performance for both frame types is the same.

1.6 Outline

Chapter 1 provides a foundation for this work by defining pervasive computing, service discovery, and the problems being addressed. The currently available solutions related to

this topic are reviewed and evaluated in Chapter 2. In Chapter 3, the problem domain is explored and the solution methodology is introduced. Chapter 4 presents the operations of these solutions. Implementation, evaluation, and results are broken into two chapters, one for service description, Chapter 5, and one for service dissemination, Chapter 6. A summary, conclusions, and final details are contained in Chapter 7. The Appendix contains information and figures regarding applications that were developed for this research.

Chapter 2

Previous Work

Analyzing, evaluating, and categorizing service discovery solutions, as a whole, is a difficult task. Solutions are complex due to the large number of responsibilities and diverse design approaches. The following framework serves as a standardized way of decomposing service discovery solutions and allows components of different solutions to be compared. This framework is used to categorize previous work in the area. With each component of the framework, the various solutions are compared. The summary of this chapter includes a comparison table of the solutions and characteristics.

2.1 Service Discovery Framework

The framework presented here is a way of decomposing the service discovery process into atomic components. This decomposition serves two purposes. The first purpose is to provide a way of categorizing and comparing service discovery solutions. The second is to provide a design philosophy for building new solutions.

For the purpose of comparison, the framework further divides the components into smaller attributes. Each attribute is required by a service discovery solution, but may be designed

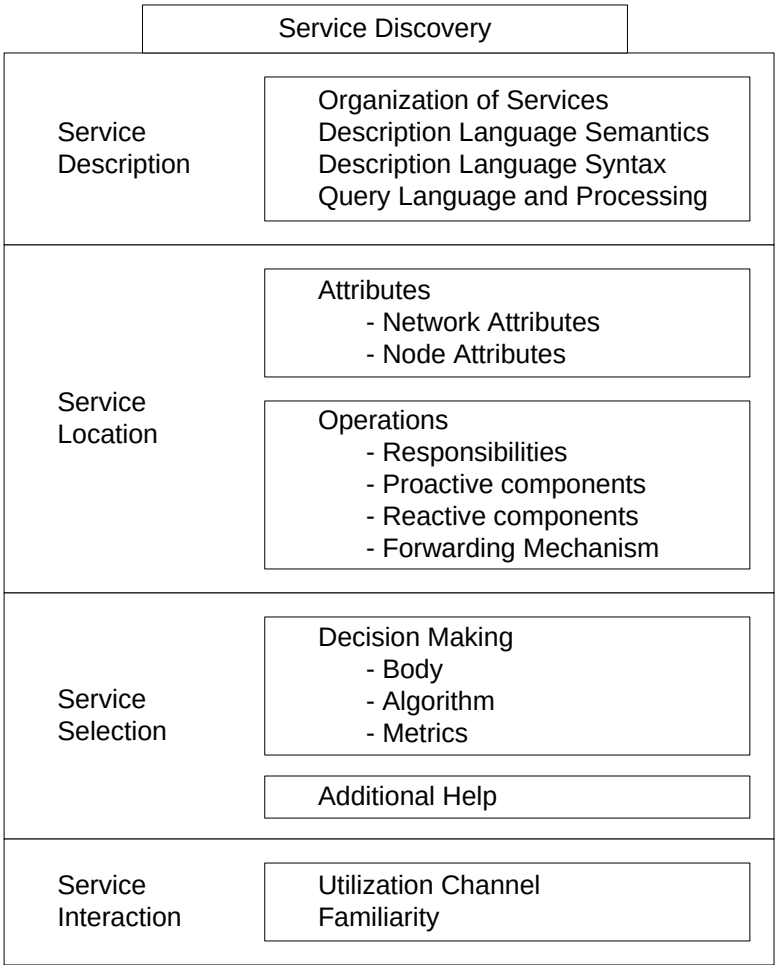

Figure 2.1: A visual representation of the framework.

differently. By mapping service discovery solutions onto the framework it is simple to compare the operations of solutions that may be different.

For the purpose of initial design, the framework urges the designer to create interchangeable components. With this design philosophy components can be easily replaced by other components that may be more suited for a specific use scenario. For example, in moving from a stationary to a mobile environment, description and selection components could be reused while the dissemination and interaction components would need to be changed to accommodate mobility.

2.1.1 Service Description

Service description is the foundation of service discovery. Contained within the description component is any information that deals with the building or processing of information about a service. Descriptions contain the information about a service that is available to external entities. Service description also directly affects the selection component by dictating what information is available for decision making. Description includes the organization of services, semantics and content of the descriptions, location queries, and query processing.

Service description is the foundation of service discovery as it deals with the service's information. This includes the information about the service and the mechanisms that process this information, the query language and matching engine.

A service description contains all of the information about a service that is available to the outside world. The information about a service includes the description content, the organization of services, and the semantics of the description. The description content is all of the information that populates the description, including identifiers, attributes, and methods. In many cases, it is useful to have a way to relate services to each other. A relational tree or trees depicting which services are related to which other services and the degree of relation can be used. The semantics of the description component specify the language used for presenting the information in a description.

The query language and matching engine describe a service from the client's point of view. This is where a client builds a description of the service it desires and the matching engine assesses which services match this request. This portion of the description component can be broken down into referencing information in a description, specifying expressions and operations or functionality that are available for building expressions. Information within a description may be organized in different ways depending on the solution. The query language is responsible for providing facilities to uniquely specify any group of information. The query language also provides a way to specify comparison expressions. Finally, the query language specifies the operations, such as equality, greater than, and less than, that

may be used in evaluating description information. The matching engine is responsible for implementing the functionality specified by the query language.

2.1.2 Service Dissemination

The service dissemination component is responsible for the interaction between entities on the network. It includes all network and node characteristics, required for operation. This component can be rather complex as a large number of solutions address this component by itself. This component includes node requirements, network requirements, solution architecture, information transference, application-level forwarding, and proxy functionality.

Network and node requirements specify the capabilities of nodes that wish to participate in the discovery process. This includes communication ability, mobility, network topology, and node resources. These requirements are meant to assure that participating nodes can fulfill roles that may be placed upon them.

The solution architecture specifies where information is stored and processed in the participating network. This includes descriptions and queries. Related to the architecture is the scheme for information transfer. This specifies how information gets from one node to another in the network. These two parts are further explained later.

2.1.3 Service Selection

After information about a service has been assembled, processed, and disseminated, it is possible that multiple, matching, service choices exist. The selection component is responsible for deciding which service or services will be used. The key idea to selection is decision making. Decisions may be made strictly based on information in the descriptions or may include other contextual and specified information. Few service discovery solutions acknowledge the importance or difficulties of the selection component.

2.1.4 Service Interaction

Finally, once a service has been selected, the requesting node must interact with the service. Service interaction dictates how this is accomplished. The method of interaction determines how much information is available about a service, familiarity with the service, and interaction requirements.

I have chosen to include service interaction in the service discovery process because knowing about the interaction between a client and service adds another dimension of information for the description and selection components. Having access through the discovery solution, also offers flexibility in service interaction by allowing clients to employ different techniques when accessing a service.

2.1.5 Security

Although important, security is not required for service discovery to function. Certain application domains require security mechanisms be present, but solutions do not require security mechanisms to function. Because security is an addition and not required for basic functionality, it is not addressed by the framework.

Additionally, security in service discovery adds difficulty in design. The primary difficulties in meeting security demands are deciding where security measures should be present in the framework and what mechanisms are to be used to provide these services. Due to the difficulties in providing a usable and robust security solution, most protocols do not include security mechanisms. For these same reasons, security will not be addressed by my work. It should be noted that security can be built into any and all components of this framework, if so desired.

2.2 Previous Work

A majority of the surveys on service discovery solutions [2, 3, 4, 5] are brief, high-level comparisons of some of the industry solutions. These reviews exhibit a lack of attention to all components of service discovery. However there has been one recent survey by Zhu, Mutka, and Ni [6] that takes an approach similar to that of Section 2.1. In this paper, the authors do a similar decomposition of the service discovery process and examine both academic and industrial service discovery solutions. While this work is similar, the author spends time discussing the security concerns of the topic. The author also views discovery from the aspect that a user will be in direct contact with the security discovery protocol. In comparison, I have the view that service discovery will be abstracted away from the user and be used as a tool to build larger services for the user using smaller services found in the environment. For example, in a smart home scenario Zhu would use a Personal Digital Assistant (PDA) to locate and operate a light fixture over the network. I am likely to have the house, the computing infrastructure, sense that I am in the room and turn on the light. This difference in perspective affects the description component of discovery more than the other components. Finally, Zhu presents valid and interesting ideas about the nature of pervasive computing service discovery and requirements for its operation.

A majority of this chapter uses the framework summarized in Section 2.1 to group solutions corresponding to common properties and behaviors. Even though this works concentrates only on the description and dissemination components, all four components are included in this review for the sake of completeness and possibility of future work. The remaining portion of this chapter includes previous work in the area of IEEE 802.11b performance evaluation. This area of work is relevant to the dissemination solution.

2.2.1 Service Description

Service description is composed of organization, content, and semantics. Organization allows services to be categorized and related to each other. Description content is information about a service available to external entities. The semantics of a description describe how information is organized within the description.

Categorization of Service Description

Within service description, solutions are grouped with solutions of similar complexity and characteristics. Description schemes can be divided into three types, basic, intermediate, and advanced.

Basic service descriptions include:

- Universal Plug and Play (UPnP) [7, 8],
- Bonjour [9, 10, 11],
- Quality of Service (QoS) [12], and
- DEAPspace [13, 14, 15].

Basic description schemes include a minimum of information. A service type and instance name are common among almost all of the schemes in this group. All of the members of this group exhibit no service organization. Due to the small amount of content available, these schemes do not need a structure for descriptions, though UPnP does employ the eXtensible Markup Language (XML) [16] to code its descriptions.

Intermediate service descriptions include:

- Salutation [17],
- the Service Location Protocol v2 (SLPv2) [18, 19],
- Jini [20, 21],

- Konark [22, 23], and
- Ninja [24].

Schemes in the intermediate category provide more information about a service or device than basic schemes. These schemes include more content information about a service or a service hierarchy. They still do not include a majority of the functionality necessary for efficient description in pervasive computing environments. Jini is unique in that it uses the Java language platform [25] and uses a different description type than the text-based descriptions of other protocols evaluated.

Advanced service descriptions include:

- Universal Description Discovery Integration (UDDI) [26] and
- Intentional Name System (INS) [27].

Solutions in this category have some essence of all elements of description. They have informative descriptions, extensible content, a structured description language and service organization. UDDI descriptions are verbose and are tailored towards human consumable resources and services.

The Web Services Description Language (WSDL) [28] was not included in the above categories because it does not provide the desired characteristics of description solutions. WSDL, present in UPnP and UDDI, is used for describing the interaction components of a service, including methods and external variables. Although, there is a documentation field available in WSDL [28], it is a single field with no further definition. We fully support the use of WSDL *within* descriptions to describe the methods available, as in UPnP and UDDI.

Query Language and Matching Engine

The query engine is responsible for deciding if a description matches a query; it is the core comparison entity. The query language describes how an entity requests a service. As with

service descriptions, query languages and matching engines are categorized by the amount of functionality they offer. The categories are basic, intermediate, and advanced.

Basic query language and matching engines include:

- UPnP [7, 8],
- Bonjour [9, 10, 11],
- Konark [22, 23],
- QoS [12], and
- DEAPspace [13, 14, 15].

Schemes that employ basic language and matching engine approaches include only the minimum functionality required to locate services. Basic schemes allow for exact matching based on a single attribute-value pair; this is typically the service type or name. Basic schemes employ a simple query language including only the minimum functionality or may not have any type of language at all.

Intermediate query language and matching engines include:

- Jini [20, 21],
- INS [27],
- Salutation [17],
- SLPv2 [18, 19], and
- Ninja [24].

Intermediate schemes extend the ideas of the basic schemes, but add more flexibility. Instead of matching on a single attribute, intermediate schemes, can match based on several characteristics of a service. Some of these schemes include additional comparison operations such as greater than and less than. Since more information can be included in a query, the query language tends to be more evolved and robust.

Advanced query language and matching engines include:

- UDDI [26] and
- XQuery [29, 30].

Schemes in the advanced category exhibit the most flexibility of the three categories. These schemes include all of the functionality of the previous two categories and include manipulation of results. Both include the ability to manipulate data prior to processing, such as sorting. Manipulation of results, such as limiting the number of results separates these schemes from the rest.

Though UDDI is strictly confined to its application space, XQuery is a generic solution that offers complete control. XQuery's flexibility comes from its development for querying in any hierarchical environment, specifically XML. One large drawback to XQuery is its verbosity as implemented in the XQueryX language [31]. XQuery also offers more functionality than is necessary for, even advanced, service query scenarios.

2.2.2 Service Dissemination

The dissemination component is comprised of numerous parts. Reviewing each part individually is beyond the scope of this review as the solutions offer a variety of attributes and operations. Instead, only the operation part of dissemination is addressed. Within the operation part are the solution architecture, information transfer, and forwarding or proxy functionality.

Architecture

The service discovery architecture includes the storage of descriptions and responsibilities of nodes in the network. To categorize the solutions used, three categories are necessary: direct, indirect, and centralized. Solutions may exist in multiple categories as many are flexible with respect to node responsibilities.

Direct solutions are completely distributed solutions where clients and service nodes communicate directly for all phases of discovery. These solutions are highly immune to collapses in the discovery process, function well in mobile ad-hoc networks (MANETs), but do not, necessarily, scale well. Solutions that exhibit a direct architecture are:

- UPnP [7, 8],
- Bonjour [9, 10, 11],
- SLPv2 [18, 19],
- Konark [22, 23],
- Jini [20, 21], and
- DEAPspace [13, 14, 15].

Centralized solutions occupy the opposite end of the spectrum by having a single or select group of nodes (intermediaries) that facilitate discovery. Clients contact service intermediary nodes which handle storage of the descriptions and communicate with clients and services. Intermediaries may participate in one or all phases of service discovery. Clients must communicate with the intermediaries for queries and services must register with the intermediaries. These solutions usually scale well as they, often, build a hierarchy of intermediaries. Each is responsible for a subset of nodes or other intermediaries. A drawback to this architecture is that it does not work well in transient environments, such as MANETs. As entities move, the hierarchy must be rebuilt meaning more information must be transmitted and processed by participating entities. Examples of this type of architecture are:

- Ninja [24],
- INS [27],
- UDDI [26], and
- Centaurus 2 [32].

The final type is a combination of direct and centralized architectures and called an indirect architecture. Indirect solutions include some characteristics of direct solutions, as all nodes may be clients, service nodes or both, but are not dedicated intermediaries. They also include some characteristics of centralized solutions, as clients and services may communicate with nodes other than the origin, service node. These schemes may build hierarchies where select nodes have more responsibilities than other nodes. These responsibilities may change to different nodes over time and include fewer or more nodes as the network changes. Solutions that demonstrate indirect characteristics include:

- SLPv2 [18, 19],
- Salutation [17],
- Ninja [24],
- Konark [22, 23],
- Jini [20, 21],
- QoS [12], and
- DSDP [33].

Information Transfer

The next part of dissemination is the actual transfer of information from node to node within a network. The two methods of transference are proactive and reactive.

The proactive method of transfer refers to the proactive transfer of description advertisements in a direct architecture. This means that services continually send out advertisements describing their services, typically at regular time intervals. By design, indirect and centralized architectures use proactive service registration between the service node and intermediary nodes or caching nodes. It would be possible to build indirect and centralized architectures that have reactive methods of registration, but none have been found. Solutions that exhibit proactive transfer are:

- UPnP [7, 8],
- Konark [22, 23], and
- DEAPspace [13, 14, 15].

DEAPspace is the only solution that is exclusively proactive. The rest of the solutions exhibit both reactive and proactive characteristics.

The reactive method of transfer is shared by all architectures. Nodes only exchange information when it is desired and do not exchange information at any other time. In direct architectures, this typically involves sending a multicast query message to nodes. Indirect and centralized architectures include reactive messages between client nodes and the caching or intermediary nodes. Again, indirect and centralized architectures typically utilize a proactive exchange between the service nodes and cache or intermediary nodes. Solutions that exhibit reactive transfer mechanisms are:

- UPnP [7, 8],
- Bonjour [9, 10, 11],
- SLPv2 [18, 19],
- Salutation [17],
- DSDP [33],
- QoS [12],
- Konark [22, 23],
- INS [27],
- UDDI [26], and
- Centaurus 2 [32].

Forwarding and Proxy Services

The final part of dissemination is the forwarding and proxy component. Within a network it is possible that multiple network layers exist and divide clients and services due to lower layer protocol incompatibilities. If a node has the ability to converse using multiple protocols, it may act as a forwarding or proxy node for nodes on the different networks. This is different from a node's network layer forwarding responsibilities in ad-hoc networks as the retransmission occurs within the service discovery protocol. In such a case, a node must offer interaction proxy support to facilitate communication between the client and the service node where they use different lower layer protocols. The only protocol that demonstrates this type of functionality is Salutation. Work has also been done to facilitate communication between different service discovery protocols [34, 35]. Forwarding may also occur in the service discovery solution across the same protocol. An example situation is a mobile ad-hoc network where nodes make forwarding decisions based on information within the packet. This cross-layer approach removes the need for additional measures, such as an ad-hoc multicast routing protocol.

2.2.3 Service Selection

Service selection is responsible for choosing a service from a list of services that is returned from the service query. The assumption is made that at any point in time there might be multiple services available that offer similar or identical services. Selection is often an overlooked component of service discovery, especially for partial solutions such as [33, 24, 32]. Solutions that do account for selection place the decision making responsibility on the service, the client application, an intermediary, or the user.

Decisions by the Service

Since services are on the opposite end from the client, it is difficult for a service to force itself upon the client. None of the reviewed solutions employ this type of selection. This type is included to cover all possible decision making points in the service discovery process.

Decisions by the Client Application

A majority of the solutions select services on the client-end. After receiving a list of a feasible services, the client application chooses a service based on service information and additional information provided by the user, operating system, and other bodies. Using this type of decision making process, though, the client must receive a full or partial list of available services. Solutions that use this type of selection mechanism include:

- UPnP [7, 8],
- Bonjour [9, 10, 11],
- SLPv2 [18, 19],
- Salutation [17],
- QoS [12],
- Jini [20, 21],
- Konark [22, 23],
- INS [27], and
- UDDI [26].

Decisions by an Intermediary

An intermediary is any entity that is neither the client nor the service. This type of decision making is typically associated with solutions that fall into the indirect and centralized

dissemination category. It is possible to incorporate this type of decision making into other situations. For example, a situation where a resource-poor client needs to make complex decisions. The client could outsource the decision to a more suitable entity to make the decision. Solutions that use this type of selection are:

- QoS [12] and
- INS [27].

Decisions by the User

With a user making decisions, numerous computing and analysis steps are avoided within the application. People are typically better at making decisions than applications in scenarios of interpretation or inference. This type of decision making is simpler on the application designer and requires fewer application resources. The main drawbacks to this approach are that it requires each client to have a user attached and for the user to be interrupted and burdened by decision making. The only scheme that explicitly advocates this type of selection is DEAPspace. It is feasible to place the selection in the hands of the user in any of the client-selection schemes above, as long as a user is present.

2.2.4 Service Interaction

Once a service has been located and selected, the client is ready to use the chosen service. The job of interfacing with the service is all that remains. Service interaction takes place in two ways, in-band and out-of-band.

In-band

In-band communication does not require a separate client application. Of the reviewed solutions, most that fall into this category use the Simple Object Access Protocol (SOAP) [36]

and WSDL. SOAP allows for the specification of method invocation without worrying about the end architectures. WSDL allows a service to describe itself to the client in such a way that the client may build an interface without any prior knowledge of the service. These ideas fall into the in-band category because all interaction takes place within a single application-layer protocol. The solutions that use this type of interaction are:

- UPnP (SOAP) [7, 8],
- Salutation (three different modes) [17],
- Konark (SOAP) [22, 23], and
- UDDI (SOAP) [26].

Out-of-band

Out-of-band solutions require the use of an additional client application. These solutions use service discovery as a means to find connection information for a service. This information is passed to an external client and interaction takes place through it. This is similar to the way that the Domain Naming System (DNS) [37] is used by numerous current applications. Solutions that employ the out-of-band communication model are:

- INS [27],
- SLPv2 [18, 19],
- Bonjour [9, 10, 11], and
- Salutation [17].

Hybrid

Similar to the other components, the interaction component has some hybrid approaches that incorporate both in- and out-of-band communication characteristics. The two solutions that fall into this category are:

- Jini [20, 21] and
- Ninja [24].

The main reason for their unique characteristics is their reliance on Java. The portability of Java bytecode allows code segments to be downloaded from other nodes and run, while still not requiring an additional external application.

2.3 IEEE 802.11b Performance

IEEE 802.11b (or some other IEEE 802.11 standard) is a likely communications mechanism for near- to medium-term deployment of pervasive computing devices due to its current ubiquity. Taking into account both the positive and negative attributes of IEEE 802.11b will allow for a solution that can avoid the pitfalls and take advantage of the useful attributes of this technology. A handful of experimental data about IEEE 802.11b performance exists [38, 39, 40, 41]. In all of this work, except [39], performance is defined as data throughput or channel utilization. In [39], Zeng, et al. address frame loss rate, but do so in coordination with the signal-to-interference and noise ratio (SINR). These works are based on infrastructure networks and use unicast data transmission for testing. Service discovery scenarios differ from these experimental scenarios because service discovery is a low-bandwidth application and frequently uses multicast in addition to unicast delivery. While none of the previously presented data are directly relevant to this work, they do suggest that there is a degradation of IEEE 802.11b performance in both throughput and packet loss as interference increases. Here, increasing interference means that the interference source moves closer to the receiver,

as in the work by Heusse, et al. [40]. Instead of throughput, packet loss is of more interest to service discovery, as it determines the reliability of discovery. Furthermore, this work makes use of IEEE 802.11's ad hoc mode instead of infrastructure mode, which is used by all of the reviewed works. These three characteristics of multicast use, ad hoc mode, and interest in packet loss metrics mean that the current performance data is not particularly relevant to this work.

2.4 Summary

In this chapter, a scheme for breaking down service discovery solutions was introduced and used to categorize current service discovery solutions. Using this scheme, a categorization of current service discovery work was conducted and presented. Table 2.1 contains a summary of the solutions and the categories into which they fall. Finally, a review of work pertaining to the performance of IEEE 802.11b in ad hoc mode was presented. The next chapter details the problem statement and methodology of this work.

Table 2.1: Summary of Related Work

Solution	References	Description	Query Language	Architecture	Transfer	Selection	Interaction
UPnP	[7, 8]	Basic	Basic	Direct	Proactive and Reactive	Client	In-band (WSDL/SOAP)
Bonjour	[9, 10, 11]	Basic	Basic	Direct	Reactive	Client	OOB
SLPv2	[18, 19]	Intermediate	Intermediate	Direct and Centralized	Reactive	Client	OOB
Salutation	[17]	Intermediate	Intermediate	Centralized	Reactive	Client	Both (3 modes)
DSDP	[33]	-	-	Indirect	Reactive	-	-
XQuery	[29, 30]	N/A	Advanced	N/A	N/A	N/A	N/A
Ninja	[24]	Intermediate	Intermediate	Centralized	-	-	Hybrid (RMI)
QoS	[12]	Basic	Basic	Indirect	Reactive	Client and Intermediary	-
Jini	[20, 21]	Intermediate	Intermediate	Direct and Indirect	Reactive	Client	Hybrid (RMI)
Konark	[22, 23]	Intermediate	Intermediate	Direct and Indirect	Proactive and Reactive	Client	In-band (SOAP)
INS	[27]	Advanced	Intermediate	Centralized	Reactive	Client and Intermediary	OOB
UDDI	[26]	Advanced	Advanced	Centralized	Reactive	Client	In-band (WSDL/SOAP)
Centaurus2	[32]	-	-	Centralized	Reactive	-	-
DEAPspace	[13, 14, 15]	Basic	Basic	Direct	Proactive	User	-

N/A : This component is not applicable to this solution.
- : This component is not addressed in the reference.
OOB : Out-of-band * : Remote Method Invocation

Chapter 3

Problem Statement and Methodology

This chapter describes the problems involved with service discovery in pervasive computing environments with emphasis on problems that are addressed in this research. It begins with a generalized problem statement, followed by detailed problem descriptions as they pertain to each component of service discovery. An explanation of the methodology used in addressing these problems precedes the validation steps of the current and devised solutions.

3.1 Problem Definition

The problem of service discovery in pervasive computing environments reduces to two areas, computers in a human-centric world and resource consciousness in heterogeneous environments. Both of these areas arise in the transition from a traditional computing environment to a pervasive computing environment. Some of these ideas are discussed by Friday et al. [42] and Zhu, Mutka, and Ni [6].

3.1.1 Computers in a Human-Centric World

As computing moves into the world around us, it must adapt to fit the interaction methods required by pervasive computing. Just as device interface methods must change to support the lack of a keyboard and mouse on every device, a device's concept of the world must change to resemble the physical world of the human users.

Space and time are examples of physical world characteristics of which computers have little knowledge. In the virtual world of computers, distance is based on the network and the number of hops between two nodes. For example, I have a personal digital assistant (PDA) and a desktop system on my desk. The PDA is attached to the wireless network and the desktop system is attached to the wired network. While they are physically less two feet apart, virtually they are four hops apart.

The growing ubiquity of network access and connectivity allows nodes almost anywhere on the globe to communicate. A node can connect to a node across the room or across the continent in about the same amount of time, at least from a human perspective. Additionally, information is typically always available. This is contrary to human-provided services which close for numerous reasons, such as the time of day, day of the week, and adverse weather.

While communication can circumvent the globe in a fraction of a second, humans cannot. Physics and current technology severely limit people from travelling near the speed of data. Because of this, location is an important factor in determining what services are useful to users. Physically-oriented services, such as a projector or light control, are location dependent. Finally, as pervasive computing environments can incorporate physical services, like a projector and light control, services must be aware of where they are in respect to users and other services.

3.1.2 Heterogenous Networks and Resource Consciousness

As pervasive computing moves into our working and living environments, services and devices will take dramatically different forms. One example of a pervasive computing application is the area of environmental controls, such as light and temperature. These services are simple in function and require minimal technology to implement. Consequently, the hardware needed to build these services can be resource-limited, only needing to support the processing, storage, and communication abilities of light and temperature control.

On the other end of the spectrum are larger and more intelligent devices such as storage servers and desktop computers. These more complex devices require more resources to fulfill their primary roles. As a result, service discovery processes have more resources at their disposal.

In addition to the diverse types of devices the numbers of different devices in a pervasive computing environment will vary. It is likely that besides being a large number of devices in an environment, there will be a larger number of smaller devices than large devices. For example, in a normal home the number of light switches far outnumbers the number of personal computers and electronics. Figure 3.1 presents a pyramid view of how device population sizes may vary based on the device type.

The diversity of nodes involved in pervasive computing environments requires service discovery solutions to accommodate the constraints of resource-poor devices, as well as exploit the comparatively infinite resources of more powerful devices. This must be done directly by controlling the amount of work directly imposed on a device, such as avoiding the responsibility of registering descriptions and answering queries. Additionally, care must be taken to avoid side-effect resource consumption. For example, a node must process all packets that arrive on its network interfaces. If there are numerous, high-frequency advertisements being sent out on a network, nodes must process each one, consuming time and energy.

The remainder of this section discusses the above problems with respect to each component

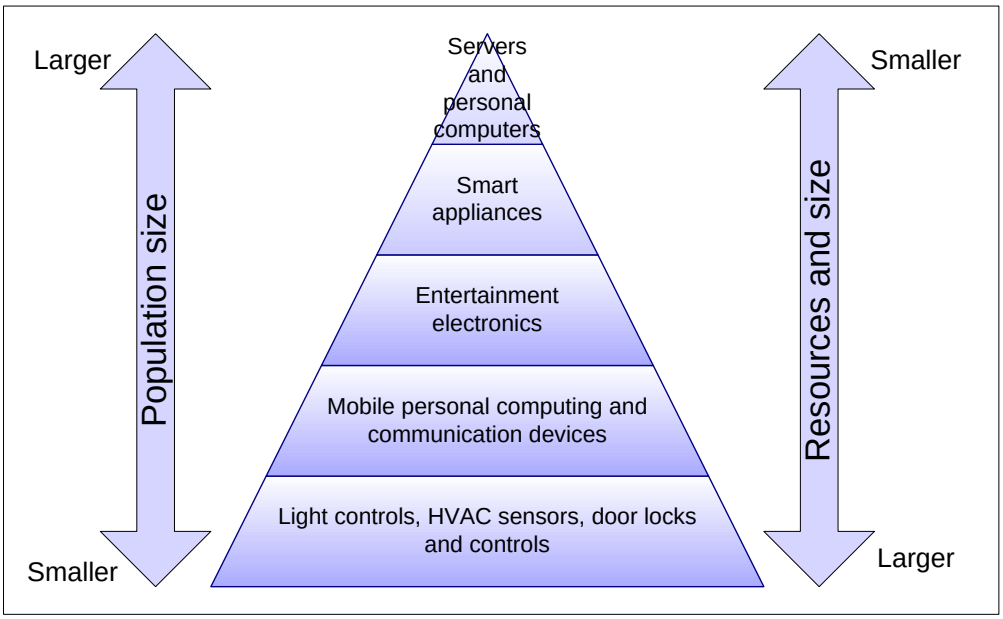

Figure 3.1: Device size versus population.

of service discovery.

3.1.3 Problems in Service Description

Service description problems fall into a subset of the previously addressed problems. These problems are knowledge of the physical world and balancing accuracy with resource constraints.

Descriptions of services contain information about a service and are the only source of information about that service. Consequently, they must contain as much information as possible about the service. Currently, no service discovery solutions incorporate enough information about the physical world to be useful in pervasive computing. This could be remedied by adding more information about the device's physical world characteristics. For example location, times of availability, and current state are common physical world attributes. Each device has a set of physical world attributes that are relevant to the device and its function.

Before a node involved in service discovery can process physical world information, the

concept of the physical world must be present in the node. A node must be conscious of what physical characteristics are and be able to interpret them. For example, a node might need to discern if another node is within a certain distance of it. This understanding of the physical world is assumed and beyond the scope of my work.

Having infinite resources to store all conceivable information about a node is desirable, but not feasible. The heterogeneity of pervasive computing nodes does not lend itself to infinite resources. To stay within the constraints of device resources, the balancing of location accuracy and resource constraints is required. Enough information must be present to ensure that a device is well represented to other nodes. For example, in a smart home setting there may be numerous light controls. Distinguishing one light control from another is important, so enough information must be present to assure that all controls have a portion of unique identifying information that is useful. Too much information may exceed a device's storage capabilities or require too much processing.

3.1.4 Problems in Service Dissemination

Service dissemination is responsible for distributing information to nodes on the network. As described in Chapter 2, there are choices in the type of architecture used for dissemination. In addition to the architecture, there are other goals to fulfill. These include resource considerations, reaching the intended devices, and dealing with the environment.

The concern for heterogeneous devices is also present in dissemination. Energy, storage, and processing capabilities are the resources of concern in dissemination. As network communication requires energy for both transmission and reception, it is desirable to keep network communication to a minimum. The dissemination architecture specifies where description information is stored. Some nodes may be unable to store the desired information, due to limited resources. The processing of network packets at lower layers and descriptions at higher layers consumes time as well as energy.

Outside of device resources, the reliability of communication is a concern for applications in pervasive computing. Locating a network access point in a building, most likely, has little regard for which particular access point is obtained, as most will fulfill the needs of the client. Certain situations require slightly different location abilities. Locating and activating a light control in a dark room is a high priority. Without locating the light, it is not only inconvenient, but difficult and possibly dangerous to navigate the room. This example depicts the need for assured location of a single service. In a case where door locks are controlled remotely, the location and activation of *all* external locks in a domicile is required when the inhabitant leaves for vacation. This scenario requires the assured location of *all* matching services.

Finally, dealing with a dynamic environment is required in pervasive computing. Wireless link reliability and mobility are two concerns in pervasive computing. It is expected that packets will be lost during transmission and nodes can join and leave the network at any time. These characteristics make it difficult to maintain knowledge of the network and node status over time.

3.1.5 Problems in Service Selection

The main problem associated with selecting a service is designing decision mechanisms. The amount of information that is applicable to a device is greater than in traditional computing, adding more variables to the decision maker. In addition to information about the device itself, user context and user preferences must be considered when selecting a service. This context could include environmental conditions, node mobility, availability of other services, device resources, and network conditions. All of this information means that decision mechanisms must be complex and take into account any information the service consumer deems important.

3.1.6 Problems in Service Interaction

Service interaction in pervasive computing environments versus traditional computing environments experiences some of the same problems encountered by service dissemination. The main problem is unreliable networking. Mobile nodes pose two main problems. The first problem is they are "bad" clients, because they can move away from a service at any moment. Second, and along the same lines, they are "bad" service providers for the same reason. Because of this, interactions involving mobile nodes should be designed to accommodate and compensate for these characteristics as much as possible. Some of these issues may be solved at lower layers with technologies like Mobile IP [43, 44]. Other issues may require solutions in the service's operation.

3.1.7 Scoping the Problem

This section presents the scope of my work. This is done before proceeding into the methodology section as this section provides the network and node attributes and assumptions of my work. Examples for an item are contained in parenthesis.

I assume the following network attributes.

- Ad-hoc wireless network (wireless physical layer and IEEE 802.11b medium access control layer).
- Addressing - local, global, and link-local multicast (Internet Protocol version 4 and 6 [45], IPv6 was only available for the latter portion of this work due to the lack of support in version 1 of the .NET Compact Framework). Moving to IPv6 is possible, but has not been completely implemented.
- Routing - most multi-hop routing protocols are admissible.
- Transport - best effort (UDP) and stream delivery (TCP). A reliable datagram solution would be useful, but is not included in this work.

- Network size - up to 14 nodes were used for development and testing.

I assume the following node attributes.

- All nodes have one or more network interfaces and share atleast one network type.
- All nodes have minimal storage, processing, and communication resources to facilitate their role.
- Node types - resource-poor sensors to server systems. PDAs were used for development and testing.
- Any node can be a client, a service, or both.

This work addresses only the description and dissemination components of service discovery. The following sections and chapters concentrate on these two components.

3.2 Methodology

This section details the steps taken in undertaking this work. I began by formulating use scenarios where service discovery would be necessary in pervasive computing. Analysis of these scenarios produced a list of requirements. Taking these requirements into account, the problem domain was established by comparing what current solutions offer to what is needed. The requirements, found through the usage scenarios, were used to formulate ideas for a solution. For comparison, a list of metrics was needed and, thus, proposed. Finally, implementations of the different schemes were constructed. Experiments were conducted and data were collected using the proposed metrics and used to compare the different schemes. The remainder of this section details these steps.

3.2.1 Usage Scenarios

Pervasive computing is rather immature in the area of available services and discovery mechanisms. This immaturity stems from a "chicken and egg" problem. The lack of services is a result of the lack of discovery mechanisms and vice versa. Service designers have little knowledge about what types of discovery mechanisms will be available when designing services. Additionally, discovery designers have little knowledge about what discovery attributes will be needed for discovery. In an attempt to address this problem, I have devised several discovery scenarios that stress certain discovery attributes. These scenarios are divided into two categories:

- Location by service attributes, physical world properties, and methods (multiple matches) and
- Location of different populations (one, some, or all services within a scope).

Location of services by information about the service is a straight-forward concept. Information about a service including attributes, physical-world properties, and methods offered by the service can be used for location.

Additionally, the proposed solution is meant to accommodate changes and future revisions gracefully. This is done by building components in the compartmentalized manner set out by the framework I introduced previously. Following this component-based approach, a specific component can be rebuilt and swapped with an existing component to fit the needs of the application.

- *Service Attributes:* Service attributes are characteristics of a service such as the service type, device type, model information, service specific capabilities, and state information. Capabilities of a service pertain to what a service can do, such as color and duplex printing for a print service. Examples of state information are service utilization and whether the service is in working order. Locating services by attributes is the most common location practice.

- *Physical-world Properties:* A service's physical-world properties are an extension of its attributes. These characteristics are different because they are not included in traditional computing. These properties include a service's physical location and times of availability. For example, someone in an office building can locate all of the services available on their current floor of the building.

- *Methods:* Locating services by the methods they offer is not often done. This is useful when a certain functionality is of interest, but knowing all about a service is not. For example, if the radio is on and the phone starts to ring, a user might not hear the phone over the radio. Also, upon answering the phone, the user must turn down the radio. If the phone could locate all services with a volume control or mute functionality, the phone could turn down the radio and then ring so the user could promptly and easily hear and answer the phone.

In traditional network environments, service discovery is, typically, used to locate a single instance of a service, such as a printer. Pervasive computing environments are different and may require the location of single, multiple, or all instances of a services.

- *Single Service Instance:* This is the typical scenario for service discovery in traditional networks. In this case, only a single service instance is required to fulfill the needs of the client. In the case of an internet gateway, for example, a user does not need multiple gateways to access the internet.

- *Multiple Service Instances:* In this scenario, a client desires to locate more than one instance of a service, but not necessarily *all* of the service instances. An example of this scenario is the case of environmental sensors. Assuming that numerous sensors are present in a room or building, some, but not all, of the sensors are used to obtain environmental information. One sensor could be an outlier and, therefore, present inaccurate data to base decisions. On the contrary, locating all of the sensors may incur additional unneeded overhead.

- *All Service Instances:* There are certain scenarios where *all* matching services *must* be located. An example of this is environmental controls. Section 3.2.1 mentions the locking of all external doors upon departure. The same is true for turning off lights upon departure. If any single service is not found, a door could remain unlocked or a light on.

3.2.2 Problem Formulation

After establishing use scenarios, documented in Section 3.2.1, the solutions from Chapter 2 were applied to the scenarios and the shortcomings of these approaches were noted. From the list of shortcomings, a generalized problem statement was developed; this was presented in Section 3.1. The following section explores design ideas and validation mechanisms.

3.2.3 Design Ideas and Validation

This section describes goals, design ideas, and validation techniques for each component of service discovery. Full descriptions of the solutions are presented in Chapter 4. Chapters 5 and 6 detail the testing and validation of these solutions.

Service Description

The purpose of the service description component is to deal with information about the service. In building a description solution the goals and ideas presented previously in Section 3.1 were used to produce mechanisms that satisfy these goals. The driving interest was to integrate ideas from the information retrieval domain into the service discovery domain. The Pervasive Service Description Language (PSDL) and the Pervasive Service Query Language (PSQL) were the result of this work and are further described in Chapter 5.

The purpose of this solution pair is not to strictly define the information that must be in-

cluded in service descriptions or explicitly define how it is to be processed, but to build a mechanism that can handle a variety of information and processing demands. This generalized approach means that as services and usage scenarios evolve over time, the information and information processing routines can accommodate the changes in how services are described and searched for.

Service Dissemination

The goal of service dissemination is to move service discovery information from one node to other nodes in the network. By nature, this component is concerned with node diversity and network behavior in a pervasive computing environment. One goal of this work is to build a dissemination component that is aware of the differences in performance and resources of different types of nodes and respond accordingly to exploit larger nodes and protect smaller nodes. A second goal of this work is to deal with instability of mobile ad hoc networks and build a solution that handles this gracefully. In addition to accomplishing both of these goals, it is important to have an efficient solution with low operation costs and overhead.

To build this solution, ideas from ad hoc routing protocols, cross-layer optimization, and current service discovery solutions are utilized. The combination of these ideas is meant to ensure that the solution is efficient on both the node-system level and the network-system level. These ideas include the following.

- *Proactive and reactive delivery mechanisms* - What are the advantages of delivering descriptions ahead of time versus on-demand? Are both delivery mechanisms feasible and for what set of boundaries?
- *Application-level forwarding* - Ad hoc multicast protocols must maintain a multicast tree or mesh. Is application-level forwarding less resource intensive?
- *Caching* - What are the resource requirements versus the metric response? What information should be cached?

3.2.4 Solution Validation

Validation of these ideas is meant to assess the feasibility of the approaches as well as the general performance of the approaches given the metrics defined below.

Service Description Solution Validation

Since the major concerns of this work are resource consciousness and accuracy of location, the metrics must support them. These metrics are the number of responses, the size of the collection of responses, and the time taken to generate the responses. These three metrics relate directly to resource consciousness, as they represent processing, storage, and network resource levels required by involved nodes. Having knowledge of the resource cost of specific operations will yield a discovery solution that can be optimized for the constraints of a given device.

The accuracy of description matching can be interpreted two ways. The first way is whether a description correctly matches the query it is said to match. This is based upon the query language and whether the information in the description correctly evaluates given the specified query evaluations. Testing for this type of accuracy is easy. For example, asserting that service that responds to a query for a printing service is, in fact, a printing service.

The second interpretation is more objective and deals with whether the query language can effectively represent what the client wishes to find. Additionally, does the description language accommodate information that the client desires to use in its query? This type of accuracy is difficult to represent with quantitative metrics. Through the use of the scenarios presented in Section 3.2.1, I have tried to accommodate a wide variety of query situations. This includes useful query operations and description information. The best way to evaluate this type of accuracy is to conduct tests where the technology is implemented around users and they use it on a daily basis. After a period of use, the users are surveyed and assess the ability of the technology to find a desired service. This study must be sure to avoid

questions that pertain to the services themselves, but concentrates on the location ability. This is also highly dependent on the application implementation to offer an effective way for users to interact with a device and have the application capture what the user is interested in finding.

Service Dissemination Solution Validation

The purpose of service dissemination is to distribute information to nodes in an efficient manner. Evaluation metrics of dissemination schemes include: location time, location probability, and overhead.

While location time is a valid concern in some scenarios, it will not be included in the measurements taken. In the stated environments, location time depends on a number of variables including the number of hops between the client node and the service node, the processing ability of the intermediary devices, the communication link speed, and the condition of the network. Actual usage scenarios will have diverse configurations of these variables, creating different results. I do not feel that our configurations will provide valid timing results that are applicable to other scenarios. For these reasons, the location time will not be measured.

Location probability and overhead should be related. As mechanisms to support a higher location probability are used, the overhead should increase. For situations where a lower probability of location is acceptable, the overhead should be less. With these measurements, the desired reward and corresponding risk may be used to choose the appropriate level of location assurance.

3.3 Summary

This chapter presented the general problem of service discovery in pervasive computing environments. It also defined the scope of what is addressed by this work. Following the problem

definition, the methodology of how the solution is derived and validated was presented. The next chapter details the solutions designed to address these problems.

Chapter 4

Service Description and Dissemination

4.1 Overview

This chapter presents the service description and dissemination schemes I designed. It is divided into three sections, the Pervasive Service Description Language (PSDL), the Pervasive Service Query Language (PSQL), and the Multi-Assurance Delivery Protocol (MADEP).

The PSDL and PSQL, combined, are an effective description component for service discovery in pervasive computing. At the node level they offer an efficient way to store, reference, and process large service descriptions. At the network level they provide mechanisms for reducing the amount of information transferred between nodes in the network and processed by end nodes. These characteristics enable PSDL and PSQL to support more complex descriptions, required by pervasive computing services, while working to keep network usage and processing low for middle and end nodes.

The MADEP is an application layer forwarding scheme for content delivery in ad hoc networks with varying levels of delivery assurance. Various levels of assurance correspond to

the various types of services found in a pervasive computing environment. It provides the basic level of assurance found in other distributed service discovery solutions and adds higher levels of assurance for services that may be of more importance. This is useful in reference to physical services as they may control security mechanisms.

4.2 The Pervasive Service Description Language (PSDL)

The PSDL is a language for building service descriptions. As previously seen in Table 2.1, most service discovery solutions use a simple description language that lacks the flexibility and structure to accommodate the amount of information needed for service discovery in pervasive computing environments. The PSDL is a feasible description language option as it includes the three parts of a description language, organization, semantics, and content.

The PSDL addresses many of the problems associated with service discovery in pervasive computing environments. It is meant to efficiently organize information about a service, allow for a variety and large amount of information about a service, provide relational mechanisms for services, and be content agnostic.

4.2.1 Organization

Organization in the PSDL is implemented with multiple relationship trees, representing a set of characteristics. Non-leaf nodes on the tree represent a level of specialization. Leaf nodes represent services. Nodes that share a parent, at some level, are said to be related. Relationships become weaker as the number of parents between the two nodes increases. All nodes in a tree are related on, at least, a high level. Nodes that share the same immediate parent are strongly related. Services may be and are encouraged to be present in as many trees as possible. Tree memberships are enumerated in the service descriptions. This way, additional trees may be created and used within an organization.

4.2.2 Content

The PSDL does not define specific content for service descriptions. Instead it leaves content specification up to the service developers and user. The PSDL provides a way to present and organize content information. Below is a list of information that is relevant to pervasive computing environment services.

- Device information (make, model, software, resources)
- Service information (type, attributes)
- Physical information (location, accessible times)
- Methods provided (e.g. as specified using WSDL)
- Quality of service information (optional)
- Security information (optional)

To ensure interoperability and consistency across service vendors, standardization of service description information is required. The World Wide Web Consortium (W3C) [46] is viable choice for a standardization body for this information. Additionally, an open forum for standard development is advisable.

4.2.3 Semantics

The final part of the PSDL is the arrangement of information in a description. The PSDL uses a hierarchical syntax consisting of three description elements, attribute-value (AV) pairs [27], subtemplates, and templates.

Attribute-value pairs consist of an attribute name and a corresponding value. The attribute name is a string. AV values can be any type, given that the format is defined and platform independent. For example, `service-type:light-control` is an AV pair separated with a colon (':'). The attribute name is `service-type` and the value is `light-control`.

Subtemplates consist of a version, AV pairs, and may have embedded subtemplates. Subtemplates are generic groupings that are context agnostic. Physical location is an example of information that could be modeled with a subtemplate; it is a collection of information, but it has an ambiguous meaning outside of a specific service. There are numerous ways of denoting location, latitude-longitude coordinates, a street address, or a building name and room number. Each notation could be stored in a 'location' subtemplate with a unique version number. For example, the location subtemplate version 0.1 could be a street address format and version 0.2 could be the latitude-longitude coordinates.

Subtemplates offer advantages over other methods because they allow a collection of information to be referenced as a single entity. Using the location example from above, a subtemplate could be compared to a query to find services within a certain distance of a point or area. Also, since subtemplates are standardized entities, clients can search for subtemplate information without knowing anything about the rest of the service. For example, a user could locate all devices built by the XYZ company between 1996 and 1998, for recall purposes. For this example, the manufacturer information is contained in a subtemplate.

Templates are collections of subtemplates and AV pairs, but are a full description and have a context. An example template is a template for a printing service. Templates are not intended to be embedded inside of other templates. Information within templates and subtemplates has different levels of inclusion, *required*, *optional*, and *additional*. Required information *must* be present in every instance of the template or subtemplate. Optional information is defined in the standard, is understood by all devices that implement the standard, but is *not required* to be present in the description element. Finally, *additional* information is not defined in the standard but may be added by the designer or user. *Additional* information should be ignored by any party that is unfamiliar with it. The purpose of additional information is to allow extensions to be made to standardized descriptions for application-specific purposes.

4.2.4 Implementation

XML is a suitable choice to implement PSDL. It is hierarchically based, well structured, and includes mechanisms for defining content types. XML Schema [47] allows portions of the language to be content-type defined and validated to avoid bad value types prior to being processed. XML Schema also supports inclusion levels, ensuring that certain defined information is present.

4.3 The Pervasive Service Query Language (PSQL)

As there are two parts to service description, the PSQL describes the second, the description processing portion. As previously seen in Table 2.1, most of the reviewed query languages are either too simple or not directly applicable to service discovery in pervasive computing environments. The PSQL remedies this by providing a flexible language that is much less verbose and more specialized than XQuery [29, 30] and XQueryX [31].

The PSQL is responsible for describing a service from the client's point of view. The functionality of the PSQL falls into three areas, pre-processing operations, expression evaluation, and post-processing operations.[1] Pre-processing functionality modifies information in a description before it is evaluated. Changing a string to all lower-case or all capital letters before comparing it to another string are examples of pre-processing operations. Evaluation contains the core of the matching engine where comparisons are made. The simplest example of an evaluation is comparing a service's type to the requested type. Finally, post-processing operations modify information after it has been evaluated, but before it is presented to the client. Sorting the list of matching descriptions based on service utilization is an example of a post-processing operation.

The goal of the PSQL is to counteract the large amount of information that may be present

[1] The PSQL was designed to be used with a description language like the PSDL. The assumption is made that these two are bundled.

in pervasive computing service descriptions. Transmitting, receiving, storing, and processing this information may be taxing on small nodes. The PSQL strives to minimize the amount of information that nodes need to deal with during the discovery process. This is accomplished by providing mechanisms for various levels of query granularity, explicit declaration of content specification, and pre-processing of description content. Strict queries allow a querying node to accurately or loosely define the desired service node to minimize or maximum the number of responses. The fewer responses a node receives the fewer the resources required for response processing and selection. Content specification and content processing allows a node to explicitly define what information it is interested in, avoiding unneeded transmission and processing of undesired information, saving resources of not only the querying node, but all other nodes involved in the querying process. All of these attributes make the PSQL useful in pervasive computing environments because it provides protection mechanisms for smaller nodes by controlling what information is transmitted, stored, and processed by nodes in the network.

4.3.1 Pre-Processing Operations

There are only a few pre-processing operations for the PSQL at this time:

- converting a string to all lower-case,
- converting a string to all capital letters, and
- normalizing spaces.

4.3.2 Expression Evaluation

There are numerous comparison operations in the PSQL. They are broken into number, date and time, and string operations. Number comparisons include, equal to, not equal to,

greater than, and less than. Date operations include all of the number comparisons. Instead of defining a list of string comparisons, regular expressions [48] prove to be an excellent solution that are well defined and widely used. Regular expressions are a powerful and flexible method for pattern matching and offer a large amount of the needed functionality. Complex expressions allow queries to accurately express the type of service desired minimizing or maximizing the number of responses.

4.3.3 Post-Processing Operations

A new feature that the PSQL adds to service discovery is post-processing operations. The purpose of post-processing operations is to further define the information that is sent to the client. Explicit inclusion or exclusion of description information, response ordering, limiting the number of responses, and returning only unique responses encompass post-processing operations at this time. Given the potential for large descriptions and the possibility of different content interests when locating a service, transferring the entire description from one node to another is often wasteful, as the client may only be interested in a small amount of the description. Therefore, the PSQL allows a client to explicitly declare the information in which it is interested or not interested and only receive desired information.

Response ordering, limiting, and sorting are related. Sorting responses based on a specific value or values allows a client to process the *best* responses first. In the event that multiple responses are available, such as in a centralized discovery model [20, 19, 17], response limiting allows the client to specify a maximum number of responses to be returned. This practice reduces network load and client storage and processing requirements, while still providing multiple responses. This is applicable in distributed, multi-hop environments as well, as nodes between a client and responders can mediate the number of responses that are passed through them, offering a loose control mechanism.

The application of post-processing operations is meant to provide a level of optimization for querying and service devices. These optimizations reflect constraints on a node due to

resource availability, including processing power, storage capacity, energy, and transmission costs. These constraints define what operations may or may not be applied.

4.3.4 Semantics and Syntax

The goals of the PSQL are to be able to reference any part of a description and create an extensible operation and expression syntax, while having a small code and data footprint. A smaller footprint means faster processing and transmission of queries. Seeing the implementation of XQuery in XQueryX [31] made me wary of overly verbose solutions.

The PSQL queries are composed of pre-processing operations, an expression, post-processing operations, and a return specification sections. Each section may have a number of elements beneath it. The pre-processing section specifies the operation to execute and the parts of the description upon which to act.

The expression section includes one or more expressions and an operation to be performed on the overall result. Valid operations for this are *and*, *or*, and *exclusive or*. Each individual expression is composed of an operation and parameters. A single operation may have one or more parameters, depending on the operation. The result of each operation is a boolean value.

The post-processing section includes any post-processing operation except for results specification. Each operation section includes the description elements on which to perform the operation.

The final section of a query is the specification of the results. The results section has a mode of either *include* or *exclude*. In exclusion mode, all description elements specified in the results section will not be returned. In inclusion mode, only the elements specified in the results section will be returned.

As with the PSDL, XML is also a suitable choice for implementing the PSQL. Figure 4.1 shows a pseudo-code example of a PSQL query. Information within curly braces enumerates

```
<query>
   <preproc>
      <op type=toCaps>Attribute Name</op>
      <op type=toLower>Attribute Name</op>
   </preproc>
   <expression operation={and,or,xor}>
      <op type={and,or,xor}>
         <param type=string>”constant value”</param>
         <param type=string>Attribute Name</param>
      </op>
      <scope type=subtemplate name=”Subtemplate Name”>
         <op type={and,or,xor}>
            <param type=number>Attribute Name</param>
         </op>
      </scope>
   </expression>
   <postproc>
      <op type=sort></op>
      <op type=distinct></op>
      <op type=limit></op>
   </postproc>
   <return type={include,exclude}>
      <attrib>Attribute Name</attrib>
      <sub>Subtemplate Name</sub>
      <method>Method Name</method>
   </return>
</query>
```

Figure 4.1: Composition of a PSQL query.

the possible values of the corresponding parameter. There is an additional element in the figure that is not mentioned above, the `<scope>` element. This element is used to specify where in the description to look for information. For example, if the make and model information are contained in a subtemplate and the date field inside is required, the operation, or entire expression section, is contained within a `<scope>` element, specifying the subtemplate name.

4.4 The Multi-Assurance Delivery Protocol (MADEP)

The Multi-Assurance Delivery Protocol is a major part of my research. The MADEP is a distributed, reactive dissemination scheme that provides multiple levels of delivery assurance and is optimized for use with IEEE 802.11. It is designed to be used in dynamic peer-to-peer style networks, such as mobile ad hoc networks. In addition to enabling various levels of delivery assurance, the use of application-level forwarding in the MADEP enables fine-grained forwarding decision making.

Application-level forwarding is useful in pervasive computing environments because it allows forwarding nodes to make forwarding decisions based on the content of the message and not just the destination. In the case of assurance, the desired assurance level dictates the forwarding mechanisms used to forward the message. This can be extended to include not only how the message is forwarded, but to whom it is forwarded. Essentially, finer-grained application-specific routing decisions are made at this level than at the network level, the typical approach. Pervasive computing benefits from this approach because nodes are heterogeneous and whether they desire the current message may be based on more information than the typical routing information of source and destination information.

The MADEP addresses two main problems. The first is how to obtain and maintain information about the surrounding environment, including node presence and characteristics. The second is knowledge of the status of data delivery. The MADEP takes separate approaches

in addressing these concerns.

For the neighbor tracking problem, only first and second hop neighbors are of interest. To counteract the instability of nodes due to mobility, user preference, or any number of reasons, constant updating by neighbor nodes is needed. Since there is a chance that multicast packets will be lost, an on-demand approach has a probability of missing nodes in a group. Because of this, a hybrid approach, including proactive updates and reactive queries, is used.

The proactive portion includes the use of periodic neighbor advertisements containing information about the node, but likely not its services. These announcements are valid for a specified amount of time. If no additional announcements are received within that time, the node is assumed to be unreachable. The two time frames that must be chosen are the *expiration time* and *advertisement time*. Both depend heavily on the transitivity of the environment, the probability of packet loss, and user preference.

The expiration time depends highly on the movement of the node and surrounding nodes. If the expiration time is too long, stale node information will be kept, causing additional expenditure of resources in attempting to reach an unreachable node. An expiration time that is too short may render a node unreachable when it is, in fact, still present.

The advertisement time should be chosen such that it assumes a certain number of packets will be lost. To keep all neighbors current, a node must send this number plus at least one packet to ensure that at least one packet reaches all neighbors within the expiration time, keeping the node current. For a higher probability of current information, a node may want to assure that more than one packet is received by neighboring nodes.

The limitations of this approach are in the warm-up period. The warm-up period is the time from when a node joins a network until all of the neighbor nodes have knowledge of this node. In the worst case, a node will wait for the length of its advertisement time before advertising itself. If a query is issued from another node in this time period, it is likely that the new node may miss the query because none of the neighbors know of its existence yet. Solutions for this are forcing a faster warm-up time by advertising the node when it

joins the new network. In the case that a node has just been turned on, or the discovery application has just been started, the new node can start advertising immediately with a shorter inter-advertisement time for a set number of advertisements. After it has informed its neighbors of its existence, it can revert to the prescribed advertisement time. In the case that a node has moved, knowledge of this movement could be used to start the fast warm-up operation. This knowledge could be obtained directly by a location tracking mechanism, such as the Global Positioning System (GPS), or some other contextual knowledge.

The cool-down period in which a node has become unreachable is of less interest as it is acceptable, in most cases, to fail to connect to a node because it is not there. This is valid up to a point, as resources are consumed storing invalid information and attempting to communicate with unreachable nodes. To expedite this process nodes could advertise their departure from the network area. This is only suggested when a node is removing itself from the discovery scope completely, and not just moving from one area to another.

4.4.1 The Neighbor Information Base (NIB)

To solve the problem of tracking neighbors in a network, an application to periodically advertise the node and specified information about the node to other nodes in the network, is needed. To solve this problem, the Network Information Base (NIB) was created.

The NIB offers multiple solutions at once. On a node, there may be multiple applications that wish to advertise information to other applications on other nodes. One purpose of the NIB is to provide a centralized space for these applications to put information and have it advertised for them. Applications also query the NIB to extract information in their scope of interest, advertised from other nodes. For example, a node is running the OLSR routing protocol, our discovery scheme, and the Simplified Multicast Forwarding protocol (SMF) [49]. The two routing protocols could put their advertisement in the NIB and it would be sent out to the rest of the NIBs. The discovery protocol's information would be included, as well. This saves the overhead of all three protocols advertising independently.

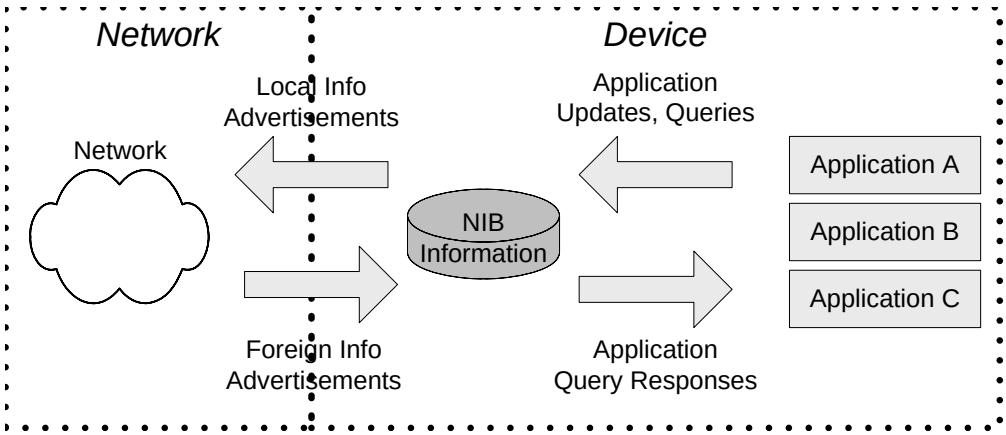

Figure 4.2: NIB architecture.

Another advantage of the NIB is the possibility of cross-layer optimization. Applications can peek at other application information in the NIB, such as the routing topology from a routing protocol. This could be used to change the application's operation or to optimize the routing protocol. There are also implications for quality of service applications in this scenario.

Our discovery schemes uses the NIB to track one- and two-hop neighbors in the network. Additionally, information, such as node capabilities, are included in the NIB information. The NIB and NIB client library were implemented in C# for Windows Mobile 2003 and 5.0 devices.

Figure 4.2 shows the logical composition of a single NIB. Figure 4.3 shows a NIB, discovery application, services and applications and how they communicate.

4.4.2 Varied Assurance of Query Delivery

The second requirement of assuring location is assuring that nodes receive the message. Our dissemination scheme has three levels of delivery assurance, low, medium, and high assurance. The more assurance that a level provides, the greater the overhead associated with the level. Table 4.1 lists the notation and meaning used in the following sections.

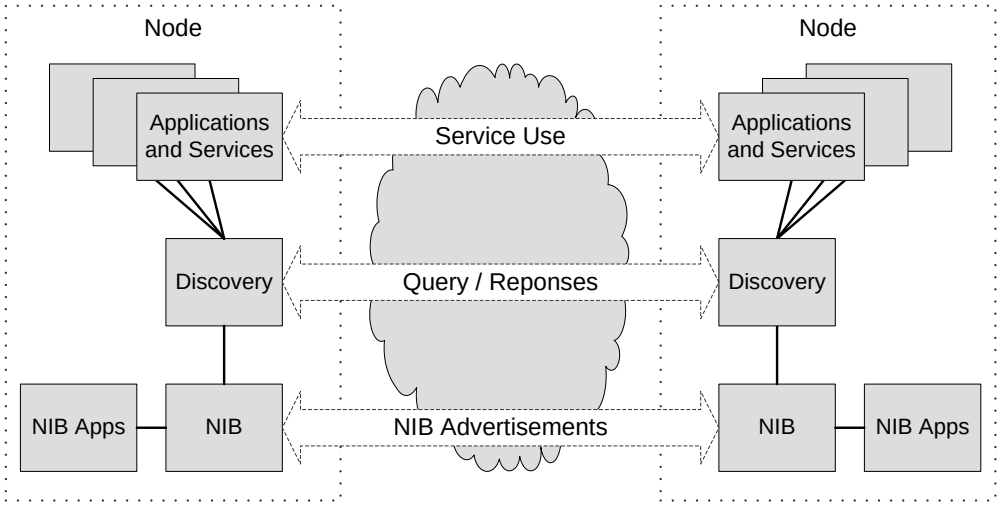

Figure 4.3: Multiple NIBs.

Table 4.1: Notation Glossary

N	Number of nodes in the network
N_C	Average connectivity of nodes in the network
R_1	The number of retransmissions by the data-link layer. (IEEE 802.11b in this case)
R_2	The number of retransmissions by the transport layer. (TCP in this case)
S_0	The size of a query frame
S_1	The size of a data-link layer acknowledgement frame
S_2	The size of a transport layer acknowledgement frame

Note on Assurance Levels

The use of three service levels was chosen based on implementation details and the ability to design three separate levels. Addition or subtraction of more assurance levels is possible and could be based on the integration of other components such as quality of service. The three levels described here are meant as an example and proof of concept.

Low Assurance

The *low* assurance scheme requires the least amount of overhead and has the simplest operation. For this level, the query is sent out using multicast. In a non-forwarding, single-hop network, there will be only one transmission made for each query. In the case of a multi-hop network, any sort of forwarding scheme to deliver the query to all nodes is acceptable. For simplicity, a simple flooding algorithm was used. The number of transmissions in a successful, multi-hop, flooding scenario is N.

The information in the NIB is not required for this level of assurance to be utilized since all traffic is multicast.

Medium Assurance

Medium level assurance provides a higher level of assurance than the low level, but requires more overhead. When a query is sent out in this scheme, a unicast, best-effort datagram message is sent to each of a node's participating, one-hop neighbors. In a successful, single-hop, non-forwarding scenario, the application layer may transmit the data $N - 1$ times.

Equation 4.1 is an approximation of the number of transmissions that will be made in a multi-hop scenario using a flooding, forwarding scheme, where there are N nodes in the network. The average node connectivity, or the number of neighbors that a node can contact,

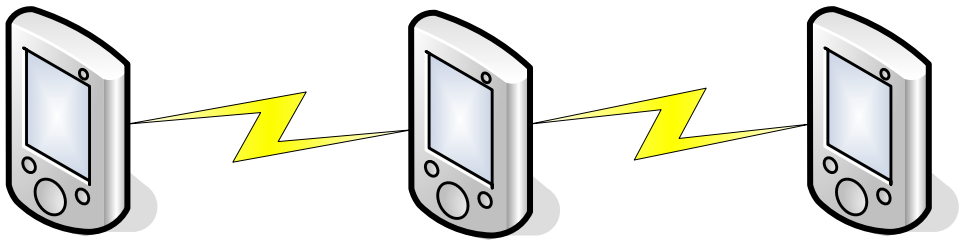

Figure 4.4: Best case topology for flooding.

is represented as N_C.

$$P = N_C + (N_C - 1)(N - 1) \tag{4.1}$$

Figure 4.4 depicts the topology yielding the lowest bound, a chain, for a multi-hop network. In this case, each outer node has a connectivity of one, while each internal node has a connectivity of two. This is the lowest value for N_C, $N_C = \frac{2(N-2)+2}{N}$. Using Equation 4.1, the number of transmissions for this topology is $P = (N-1)$. This serves as the lower bound of the number of transmissions in a multi-hop network.

Figure 4.5 depicts the highest connectivity scenario of a single-hop network. In this topology, each node can contact all of the other nodes in the network, meaning $N_C = N - 1$. Substituting this value into Equation 4.1 results in $P = N^2 - 2N + 1$. This serves as the upper bound of the number of transmissions in a multi-hop network.

High Assurance

For *high* assurance scenarios a certain level of confidence should be available that if the node is present the message will be delivered. To accomplish this, the *medium* assurance scheme was extended by using a reliable transport protocol instead of best effort. The number of transmissions in this scenario is the same as that of the *medium* assurance scenario, previously discussed.

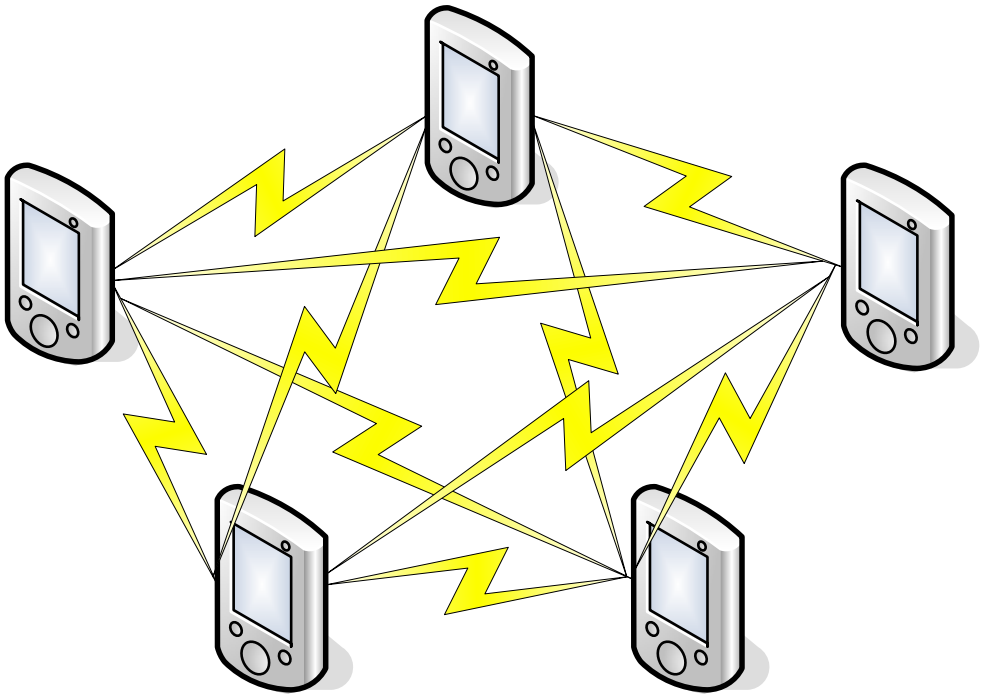

Figure 4.5: Worst case topology for flooding.

4.4.3 Choosing an Assurance Level

With the choice of three assurance levels, the user of the discovery mechanism must weigh the benefits of higher assurance with the cost of higher overhead. In the case of virtual services and multiple service instances, it is acceptable, in most cases, to use the low assurance level. For high priority services and specific queries, the high assurance level is recommended. The medium assurance level can be used instead of the high assurance level in some cases, as the packet dropped rate is still fairly low, as will be seen in Section 6.2.

4.5 Summary

In this chapter, the solutions to the problems from the previous chapter were presented. The PSDL and PSQL address the concerns of service description. The MADEP addresses the problems of service description dissemination. The next chapter details the implementation,

testing, and analysis of the the PSDL and PSQL. The MADEP is further described in Chatper 6.

Chapter 5

Service Description: Implementation, Testing, and Results

5.1 Query Simulator

5.1.1 Overview

The goal of the PSDL and PSQL is to optimize the performance and resource consumption of the query process and results. The metrics used to evaluate the effectiveness of the measures employed to meet this goal are time to complete a query, the size of the result set, and the number of results in the result set. These metrics represent the resources required of the processing nodes, intermediate transport nodes, and the querying node. Also, it is important to note that these results are not specific to a certain type of dissemination scheme and may be used in the design of different description schemes. The duties performed and resources used must be present in a description scheme, the dissemination scheme determines on which nodes they take place.

Simulation was used to collect data on the performance and resource consumption of current

and new query processing mechanisms used for service discovery. This simulator mimics the processes undertaken by query processing nodes. Because a major goal of this work is to support both small and large devices, a small device, a PDA, was used as the test platform. It is assumed that a large device, having considerably more resources, would yield much better results and be less of a concern.

The simulator starts by generating a list of service descriptions and a list of queries. Descriptions are composed of a user-specified number of string and number attributes, allowing the user to model the length and content types of a real description. Random values, within a specified range, are assigned to the string and number attributes. Queries are constructed in much of the same way, using a user-specified number of matches and match type, string or numeric. Once the lists of queries and descriptions are created the queries are evaluated against the list of descriptions using each of the five query types. These five query types will be defined in the following section. For each query type the aforementioned metrics are collected and logged. Later, these logs are processed and reports are generated.

While this work is usable in conventional service discovery, it is of more interest for pervasive computing environments because of the increase in the amount of information in service descriptions. The goal of this work is to keep processing and transmission to a minimum, as much as possible, to reduce the load on smaller nodes, and increase general system performance.

Again, it is stressed that these results are admissible to different types of dissemination schemes. However, a reactive, distributed, direct scheme is the dissemination architecture of choice for this work. The following analysis will reflect this, as well.

5.1.2 Experimental Setup

The experimental setup consists of the query simulator running on a Dell Axim X30 (624 MHz XScale processor, 64 MB memory). A desktop computer was used for development

and testing.

Parameters for the simulator include:

- A random seed for description creation,
- Description generation parameters (number of services, number of integer and string attributes, attribute ranges),
- Pre-processing option (convert strings to lower-case, convert strings to upper-case),
- Evaluation options (number of string and integer evaluations, evaluation operation), and
- Post-processing options (distinct results, sorting, limiting return population, limiting returned description information).

The simulator operates by generating a list of descriptions. Each description contains the specified number of integer and string attributes. The size of the description is meant to reflect the size of pervasive computing service descriptions and is larger than typical service descriptions. The contents of the attributes are randomly generated within specified parameters. Once the list has been generated, a query is also randomly generated. A query consists of the specified number and type of evaluations. The query values are randomly generated within the specified range of the description attributes. The query is evaluated against the list of descriptions and a list of responses is created. This operation is timed and used as one of the metrics. Each query is implemented and executed against the service list using a set of query types. The query types are as follows.

- Basic: Basic queries mimic the simplest of the query abilities. A basic query can only evaluate a single expression at a time. If a query includes a combination logic step, each element of the equation becomes a single query and is executed individually. The resulting responses must then be processed at the querying node to create the final

result set. For example, if the query is looking for `(attribute A AND attribute B AND attribute C)`, three separate queries would result searching for each attribute individually. This is the query type available in UPnP, Bonjour, the QoS protocol, and DEAPspace.

- Intermediate: Intermediate queries are more powerful than basic queries as they can execute complex combination logic statements in a single query. The previous example of `(attribute A AND attribute B AND attribute C)`, can be searched for in single pass instead of three. This type of query is present in SLPv2, Salutation, Ninja, Jini, INS, and Konark.

- Advanced with population limiting: This type of query is an extension of the intermediate variety, but provides a limit on the number of responses to a query. Population limiting is useful because it specifies a maximum number of responses avoiding a large flood of responses in the case of an overly broad query. This type of query is easy to implement in a centralized dissemination architecture, but harder in a distributed architecture. This type of query is available in UDDI.

- Advanced with information limiting: This type of query is an extension of the intermediate query type, but adds the ability to specify what information from the description is desired. In the case of large descriptions, it is quite possible that only a small amount of information will be desired by a client. This query type allows the client to directly specify what information it is or is not interested in receiving, thus reducing the amount of information it has to process and the amount of information that is transmitted between nodes in the network. This query type is not directly supported by any existing service discovery protocol.

- Advanced with population and information limiting: This final type of query includes the attributes of both population limiting and information limiting. It allows for further reduction of responses and their size. This query type, also, is not available in any of the previously evaluated service discovery protocols.

Basic or intermediate query types are used by a majority of the service discovery protocols reviewed for this research. This makes them a basis for comparison with types that include additional processing capabilities. The simulation compares the performance of these existing methods to the performance of new methods proposed in this research intended to reduce the amount of information that must be processed in the service discovery process.

Screen captures of the simulator may be found in the Appendix.

5.1.3 Results

The performance of the five query variations was simulated for each of the three metrics, time, response list population size, and response list data size. The variations are basic, intermediate, advanced with population limiting (pop-limit), advanced with information specification (info-limit), and advanced with both population and information limiting (info-pop-limit). The population limiting scheme set a maximum number of returns for the search result. When this number was reached, the search returned and did not process any other descriptions. The information limiting scheme specified a subset of the description to be returned for each description. This reduces the size of the returned information for each result, but costs extra resources to load and process each result. The scheme that included both population and information limiting, executed the population limiting portion first and, therefore, only operated on the remaining results.

All simulations use the number of services as the independent variable. The number of services ranges from 20 to 400 in increments of 20. Service lists consist of 25 string values and 25 integer values, with a range of three values for each. This models an environment with a small number of a service types, but a large number of each type of service, such as a small to large pervasive computing space like an Aware Home or office. Services in these environments may include light controls, environmental sensors, printers, projects, and wall displays. The size of the descriptions is meant to include model information, manufacturer information, location, utilization, and device specific information. A printer, for example,

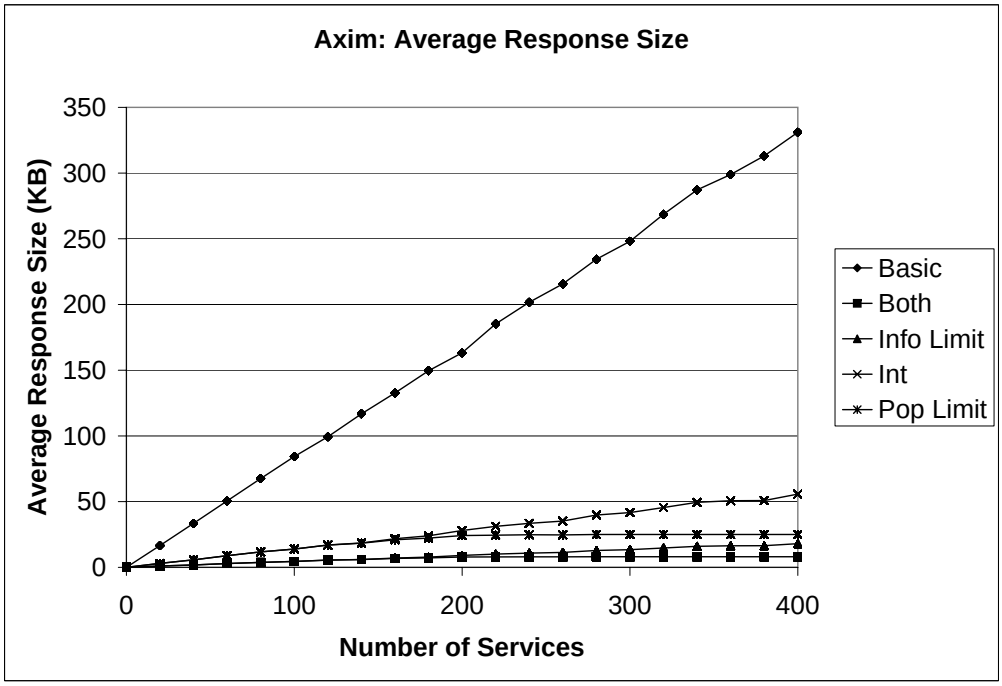

Figure 5.1: Response size versus the number of services.

may include the ink levels, color printing ability, duplexing ability, the amount of paper available, and collating ability. Each simulation was run 50 times with a different random seed each time. For both schemes that include population limiting, a maximum of 20 services was used. For schemes that include information limiting, a maximum of 20 values was used.

The data size of the response list versus the service population is shown in Figure 5.1. The intermediate and population limiting schemes are equal until the advanced scheme reaches the maximum number of responses in its response list and then remains constant. The size of our service descriptions is fairly invariant, so the size of the response list for the population limiting scheme stays fairly constant. The information limiting scheme provides the second to lowest response list data size. This comes at the cost of additional time, as is seen in Figure 5.3. Finally, the combination of population and information limiting provides the smallest total size. The size of the response list is important because it is the amount of information that a client will have to process upon the reception of responses. The goal of information is to return only the information that the client requests to evaluate the

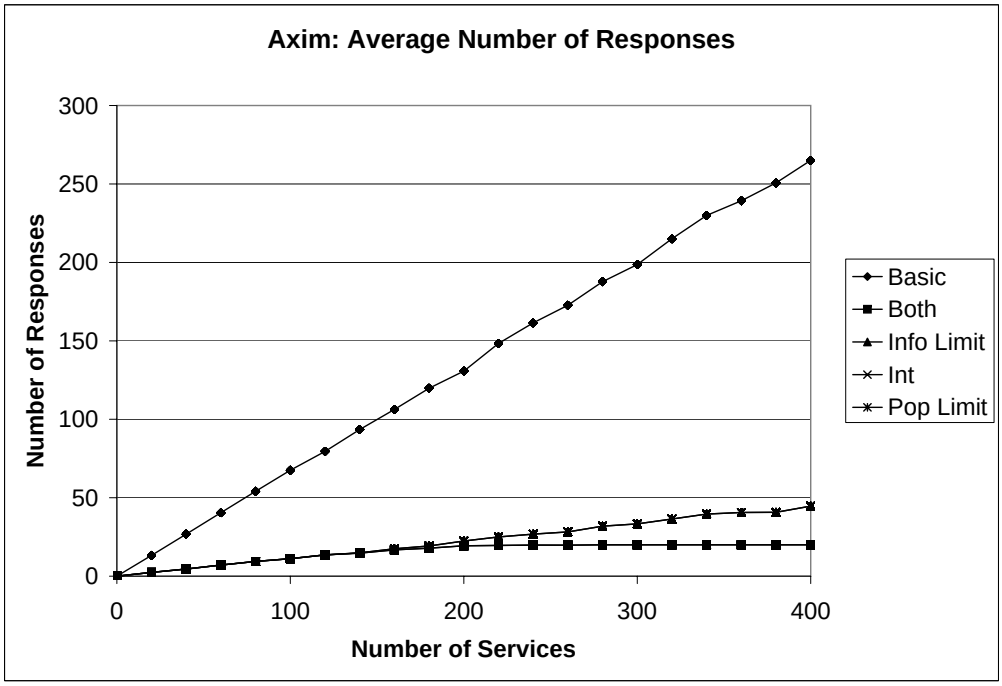

Figure 5.2: Number of responses versus the number of services.

description against the query. All information in addition to the required information that is sent to the client wastes the resources of all nodes involved.

Figure 5.2 depicts the response list population size as the number of services increases. The intermediate and information limiting schemes continue to increase as the number of services increases. The other two advanced schemes take advantage of response population limiting and do not exceed 20 responses. This type of behavior is useful in situations where a large number of matching services are present and it avoids high response processing overhead by a client.

Finally, the processing time of evaluating these queries on the Dell Axim X30 is shown in Figure 5.3. The main goal of this result is to demonstrate the cost of the different schemes at the devices rather than for the network. I make the loose assumption that greater completion time directly relates to greater resource requirements, including, at a minimum, energy and processor cycles.

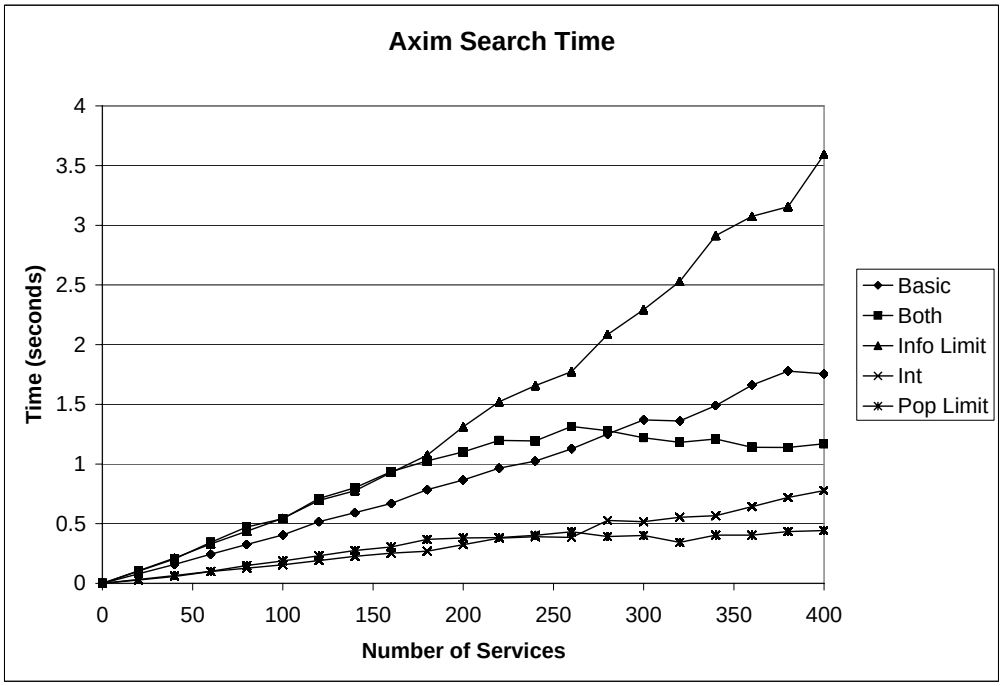

Figure 5.3: Response time versus the number of services for the Dell Axim X30.

The information limiting scheme, while demonstrating mediocre characteristics in other areas, requires the greatest amount of time to complete the algorithm. The scheme employing both information and population limiting yields the second highest time until the specified population size is reached. Once this happens, the time required stays fairly constant. Intertwined with the information and population limiting scheme is the basic scheme. The response time for the basic scheme continues to increase after the information and population limiting scheme reach a steady state. The intermediate and population limiting schemes perform about the same until the population limiting scheme reaches its maximum population size. Following this, the response time for the intermediate scheme continues to increase while the response time for the population limiting scheme remains constant.

5.1.4 Analysis

The appropriate combination of techniques for a given deployment environment depends on many factors, such as the number of services, the number of services per device, the location of devices, etc. As a result of this, a single solution set is not possible. This section describes the applicability of these techniques and how they may be applied to optimize performance and resource use.

By all metrics, the basic scheme does a poor job. While it is not the slowest of the approaches it is the most resource heavy. In a distributed architecture, transferring the large amount of response data, versus the smaller amounts of other schemes, would be detrimental to the network's overall performance and consume device energy to send and receive, as well as, process. Since the basic scheme offers no logic operations, the client node would have to process this data locally, taking more time and resources. Since service descriptions are likely to be large and the focus of the PSQL is to use complex queries to limit the number of matches as much as possible, the excess resources required by this scheme for query execution and result storage, make it less desirable for pervasive computing environments.

The intermediate query type fairs well in pure time performance and ties with the information limiting scheme in the number or responses. Unfortunately, this query type comes in second to last in the size of total responses. While this type outperforms the basic query type in all categories, it might have problems in the case of a large number of responses or very large descriptions. The major attribute of this query type is that it is reasonably fast, compared to the information processing types and the basic type. In a scenario where speed is important while storage, transfer, and processing costs are low at the query processing node, the query node and all points in between, this is an acceptable query type.

The population limiting query type has the ability to bound the number of responses in a centralized architecture. However, in a distributed environment, such as the one on which this research concentrates, it is less feasible as no single node processes all of the descriptions. It would be possible to have nodes restrict the number of responses they forward. This would

not provide a strict bound, but it may reduce the number of responses a querying node receives, but all passing queries must be processed to determine if they should be forwarded or not.

One large problem with population limiting is how to choose a metric on which to base limits. In these tests, the descriptions were filtered based on the position in the list, meaning descriptions at the bottom have a lower probability of being searched. Using another metric, such as relevance, would pose a new list of problems as descriptions would have to be processed to assess relevance before filtering. This would cost additional processing resources and take more time. The complexities of choosing and implementing a population limiting scheme are many and its use should be carefully considered.

Queries including information limiting characteristics strive to reduce the amount of data in a single description. Based on the results, it is acceptable to say this is the most processing intensive query type. Given a distributed architecture where a node is only responsible for its services and descriptions, this type of query may be feasible since a node will, likely, have a small number of services. It is assumed that the more services a node offers, the more resources the node has, as it takes more resources to complete more tasks. In this case, since a node will only be searching a small number of descriptions, it is likely that it will be able to handle the small amount of additional processing. Figure 5.4 shows a magnified view of Figure 5.3 showing that the time required for a small number of services, less than 20, it takes less than 100 milliseconds to search and generate a response. Given this small amount of time, it is quite feasible for a small device to process and generate responses in a low volume scenario. Because of this, the information limiting scheme offers much to the service discovery process by reducing the amount of information that querying nodes have to process and intermediate nodes have to forward.

By using both information limiting and population limiting, a system can further reduce the amount of information processed and transmitted. Since this query type includes population limiting, it inherits the same aforementioned problems and should be used carefully.

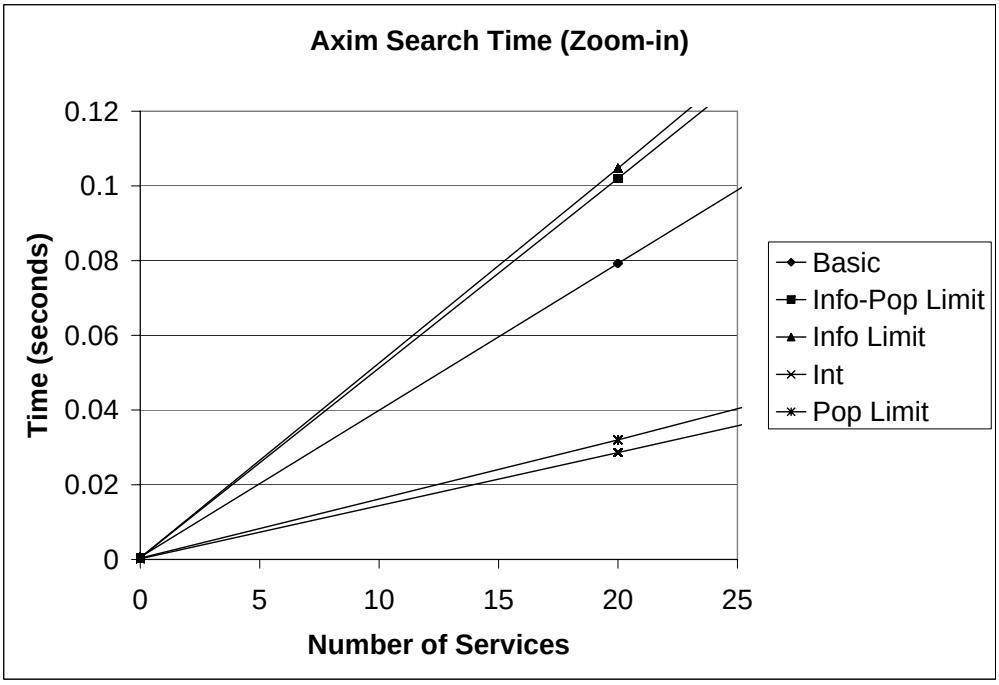

Figure 5.4: Response time versus the number of services for the Dell Axim X30 (zoomed-in).

Given these results and analysis, for the applications scenarios considered in this research, information limiting is feasible and useful. Population limiting could be used, as well, in some simple form that incurs as little additional processing as possible. This approach provides a reduction of the amount of information that a querying node will have to process during the selection stage and that intermediate nodes will have to forward as it travels from service nodes to the querying node.

5.2 Summary

This chapter presented the performance data for the addition of post-processing mechanisms and the impact on resource consumption and performance. These results show that these techniques, when applied in the correct manner for a given application, can greatly reduce the amount of data that is sent and processed by the querying node. This is important because there may be numerous nodes between the querying and responding nodes that

have to forward this information, as well as, the querying node must process all information in the next step of the discovery process, while most of this information may not be relevant to the decision process. The next chapter describes the testing and validation of the MADEP for dissemination.

Chapter 6

Service Dissemination: Implementation, Testing, and Results

6.1 Overview

The NIB and discovery applications were written in the C# language, using the .NET Compact Framework version 2.0. The three devices listed in Table 6.1 were used for testing. All devices include an integrated IEEE 802.11b interface.

6.1.1 Network Information Base Implementation

Overview

The NIB has two parts. The first part handles local communication between the NIB and applications on the device. The other half of the NIB communicates with the rest of the network, sending advertisements and processing the advertisements of other nodes on the network. Both parts of the NIB utilize the central data store for storing and retrieving information.

Table 6.1: Test Devices Used

Model	CPU	Memory	OS
Dell Axim X51	XScale 520MHz	64MB RAM 128MB ROM	Windows Mobile 5.0
Dell Axim X30	XScale 624MHz	64MB RAM 64MB ROM	Windows Mobile 2003 Second Edition
HP iPAQ h4150	XScale 400MHz	64MB RAM 32MB ROM	Windows Mobile 2003 Second Edition

The central data store is a table containing a number of columns. Our design employs four columns in the table: a node identifier (the node's network address, in this case), the number of hops to the node, a timestamp of when the node was first added to the table, and a timestamp of the most recent update of the node. Additional columns, including other node specifics, could be added to the table. In addition to the standard columns, there is a column for each local application that has added information to the NIB, such as a routing protocol or our discovery protocol.

Local Application Updates

Local applications connect to the local NIB application through a C# library, the NibLib, and update, add, or remove their information. Local applications may not modify the information of other nodes in the table.

Local Application Queries

Applications may query information from the NIB by protocol name or node identifier. A maximum hop parameter is included to ensure only information of interest is provided to the application. Specifying the node identifier returns all information for all instances matching

the identification string with less than the specified hop count. Specifying a protocol returns all node entries that have a non-null value for information corresponding to the specified protocol. Nodes must also be below the specified hop count. Finally, the entire table may be requested by querying for an identification or protocol with a wildcard. The hop count parameter limits nodes by hop count here, as well.

Network Advertisements

Advertisements are sent containing all of the information in the table for nodes that are less than a specified hop count away. Two was chosen as the maximum hop distance, so each node advertises itself and its one-hop neighbors. Upon reception, a node adds unknown nodes and updates known nodes in its table. Advertisements are sent out at a specified interval, determined using the previously mentioned considerations.

Local Storage Maintenance

To make sure that the NIB has recent information, maintenance must be performed. This includes removing nodes that have not been updated in a specified amount of time. This time is also determined based on the mobility of the network. The greater the network mobility, the more often nodes will leave the vicinity, and the more often the list of known nodes should be checked for expired nodes.

Implementation Specifics

For our implementation, all queries, query responses, and advertisements were coded using the eXtensible Markup Language (XML). An expiration time of 121 seconds was used. For an advertisement interval, a uniform distribution process with a maximum value of 45 seconds and a minimum value of 15 seconds was used. The expiration value is meant to model a pervasive computing environment where devices and services are stationary or attached to

users. Given the movement speed of people, these values are meant to model the time it takes for a person to move well out of the radio range of a stationary device or service. The interval time was chosen to assure that at least two advertisements were sent during an expiration time interval. This value performed adequately during the tests. All communication was done over IP version 4 and two UDP sockets, one for local communication and one for advertisements. Scoped local-multicast was used for the advertisements.

Additional Comment

In the case of small devices, it could be useful for an application to act as a proxy application for small devices by advertising for the small node and allowing the small node to query its NIB.

6.1.2 Discovery Implementation

The test discovery application is broken down into the following parts and responsibilities.

- User Interface (UI) Thread: responsible for interacting with the user. The user starts and stops the other application threads with the UI thread.
- Test Manager Thread: manages the test runs, starting and stopping the other threads at set time intervals.
- Generator Thread: generates queries and passes them to the socket threads for transmission.
- Maintenance Thread: maintains the known query list by removing expired packets.
- TCP Socket Threads: send and receive traffic over TCP. Each child socket has a send and receive thread. The send thread handles connecting and sending datagrams. The

receive thread receives the datagram from the network and executes the query processing methods. There is also a single accept thread that handles incoming connections on the parent socket.

- UDP Socket Threads: Winsock asynchronous calls are used for UDP sends and receives.

Implementing Assurance

To implement the different assurance levels, the transport protocol and addressing type were varied. Known characteristics and performance knowledge of the IEEE 802.11b MAC-layer in ad hoc mode from [50] were also used in the design. These characteristics include the difference in packet loss probability between unicast and multicast frames. Unlike multicast frames, unicast frames include a MAC-layer acknowledgement and retransmission scheme. This retransmission function incurs extra overhead, but provides a higher probability of delivery at low volumes. The difference in delivery probability is used to build separate levels of assurance. As with the specifics of the assurance levels used in this work, the implementation of these levels is presented as an example and proof-of-concept.

Overhead

This section introduces a method of approximating the amount of data required for each assurance level for the flooding scheme used in this work. This calculation is composed of three parts. The first part is the number of message exchanges that are made in order to disseminate a message to all nodes. This value is dependent on the type of forwarding scheme. Since flooding is used here, this will be the number of node-to-node paths in a network. This is given in Equation 4.1. The second part is the size of the data and acknowledgement frames transmitted between nodes. The final part is the number and type of frames that are exchanged between two nodes as a message is transmitted along a single path. This portion of the equation will vary as levels of acknowledgement are added. Equation 6.1 presents a

generalized equation for the amount of data given a specified number of acknowledgement levels, x. Level 0 is the application layer and does not use acknowledgments.

$$T = P * \Big(\Big(\sum_{i=0}^{x} S_i \Big) \Big(\prod_{j=1}^{x} (R_j + 1) \Big) + \Big(\sum_{k=1}^{x-1} S_k \Big) \Big(\prod_{l=2}^{x-k} (R_l + 1) \Big) \Big) \tag{6.1}$$

The low level of assurance was implemented using UDP with multicast addressing. As the IEEE 802.11 MAC uses a best-effort model for multicast, no additional lower-layer overhead is incurred. Using Equation 6.1, $x = 0$ as no levels of acknowledgement are included, reducing the equation to $T = P * (S_0)$. Since messages are resent once by each node, P is N; $T = (N)(S_0)$.

The low level of assurance, since it is employed by all of the distributed architecture schemes, provides a point for comparison of current solutions versus work presented here. The results for this level of assurance may be assumed to represent the performance of all of the distributed architecture schemes, such as UPnP, Konark, and DEAPspace.

The medium assurance level was implemented using UDP and unicast addressing. The list of neighbors is obtained from the NIB and used to send individual packets to all one-hop neighbors. Since the IEEE 802.11 MAC employs acknowledgements for unicast frames, each packet sent incurs at least one additional frame transmission. While an acknowledgement is much smaller than the original packet, it does add more traffic to the medium. If an acknowledgement is not received by a sending node in a specified amount of time, the original packet is retransmitted, increasing the network traffic even more. This will continue until an acknowledgment is received or the maximum number of retransmissions is reached. For this level of assurance there is one level of acknowledgements, therefore $x = 1$. This reduces Equation 6.1 to $T = (N_C + (N_C - 1)(N - 1))((S_0 + S_1)(R_1 + 1))$. In the lower and upper bound network topology scenarios the P portion of the equation will vary from $P = N - 1$ to $P = N^2 - 2N + 1$. The packet loss during the transmission of a message from one node to another will dictate the number of retransmissions. If the communication is successful on the

first attempt, the number of retransmissions will be 0, thus $R_1 = 0$. In a poor environment where packet loss is great, the maximum number of retransmissions will be needed, $R_1 > 0$ and equal the maximum number of retransmissions for the MAC layer in this case.

The high assurance level was implemented using TCP and unicast addressing. Like the medium level, the list of one-hop neighbors is obtained from the NIB and an individual message is sent to the node using TCP. In addition to the IEEE 802.11 MAC's frame acknowledgement overhead, TCP also uses acknowledgments. We also assume the connection is already established, the connection stays open after the transmission, and TCP does not include any other operation besides acknowledging the reception of a message. These additional operations are specific to TCP and may not apply to other implementations of these schemes. For the high assurance level $x = 2$ in Equation 6.1 reducing the equation to $T = (N_C + (N_C - 1)(N - 1))\Big(((S_0 + S_1 + S_2)(R_1 + 1)(R_2 + 1)) + S_1\Big)$. In this equation the IEEE 802.11 is layer 1 and TCP is layer 2. Like the medium assurance level, the value of P is dependent on the network topology and varies the same as the medium assurance level. In a scenario with no packet loss, the value of T is close to the value of T for the medium assurance level, with the addition of the additional upper layer acknowledgement, TCP in this case. If network conditions are poor, the total amount of data required will be much larger the medium assurance level due to the additional acknowledgement scheme of TCP.

TCP Woes

A number of problems were encountered in building the datagram protocol using TCP. To simulate datagram behavior, only send and receive methods were made available to the query generator. TCP sessions had to be built and destroyed "on the fly." Instability of nodes caused problems, as well, mainly through connect and send actions. Winsock asynchronous calls were used at first, but had problems with networks of eight or more nodes. The Winsock subsystem has a limited number of threads to service these calls. Operations blocked these threads leaving other operations queued to be processed. Additionally, new calls that were

Figure 6.1: Test setup for the close scenario.

made while these calls were blocked were queued and were serviced in their arriving order. At bad times, there were over 130 asynchronous calls waiting to be handled. To bound these operations, dedicated threads were used instead, giving each child socket a send and receive thread. There is also a queue for sending datagrams. This proved to be an effective way of handling the connect and send blocking issue. If a connection or send failed, meaning a node was no longer available, all datagrams going to the node were discarded. This avoids other send and connect calls from blocking when it is likely that they will fail, as well. Another possibility that was left untried was using the select() method and handling multiple sockets with a single thread. This method is not preferred, as discussed in "The Windows Sockets Lame List"[51].

6.1.3 Test Configurations

Two test configurations were used. The first was a close scenario where all nodes were in close radio range of each other, shown in Figure 6.1. This configuration models pervasive computing environments where devices are close to each other. These types of networks are

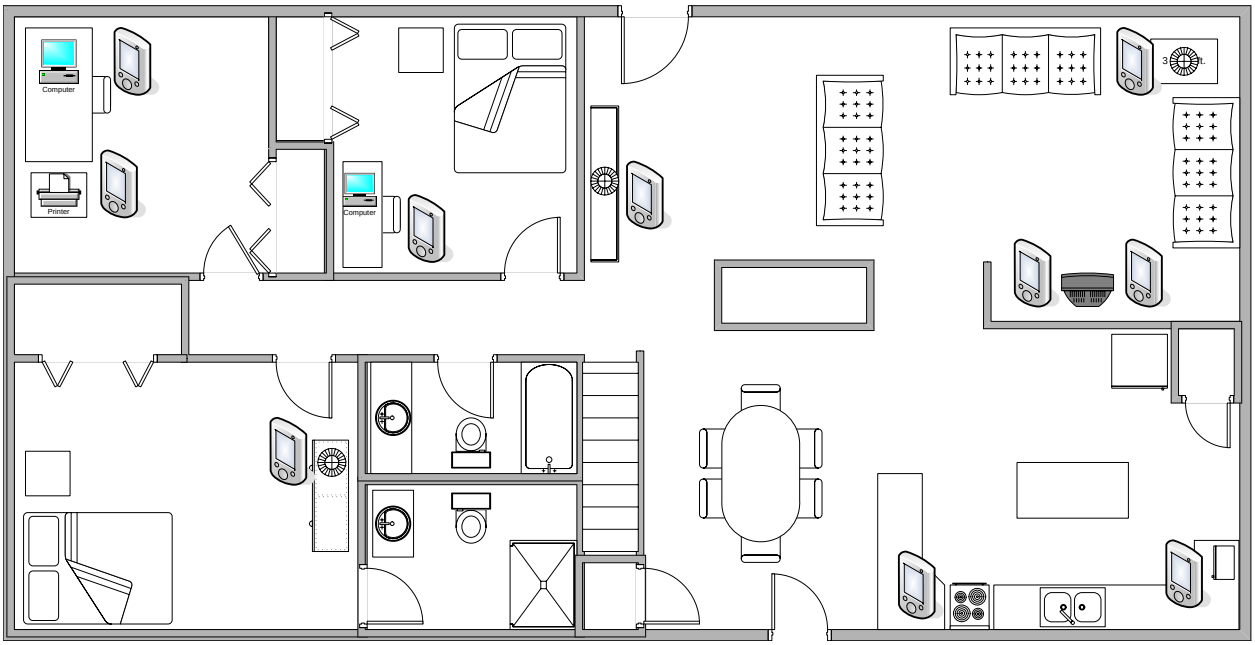

Figure 6.2: Test setup for the Aware home configuration.

called personal area networks and may appear in scenarios of medical monitoring or where a user carries a number of computing devices. Tests were conducted in this scenario for cases with and without a forwarding scheme and with and without interference. Testing without a forwarding scheme demonstrates the scheme's single-hop performance. This could be used to interpolate the scheme's performance when a forwarding scheme is applied, as it demonstrates the scheme's ability at each hop of the forwarding process. It can also indicate a worst-case scenario where there are no other nodes to forward the information, as forwarding schemes can add redundancy and, therefore, a higher probability of delivery.

The second scenario was meant to model another type of pervasive computing environment where nodes are situated to model an "Aware home"[52] with connected appliances, lights, and computing devices. The house layout and position of the devices are shown in Figure 6.2. Nodes were placed in the following locations:

- Master bedroom: floor lamp

- Office: desktop computer and printer
- Second bedroom: desktop computer
- Living area: two table lamps, television, and stereo
- Kitchen: toaster oven and stove

This scenario is a feasible configuration for an Aware home, as all of these devices could benefit from the integration of electronics and communication. While it may not be likely that these devices initiate discovery queries of their own, as they do in this test scenario, it is possible that a user is near one of these devices and the query is forwarded by the device to the other devices. Given this assumption, the current configuration mimics a real scenario reasonably accurately. It is also possible that the user may use the device to initiate the query directly from the device in the case of a device with visual output and user input ability. For example, the user could use the television to check the status of the oven.

Forwarding

A simple flooding algorithm was used as the forwarding scheme for this work. It is possible to build any type of forwarding scheme and implement it based on this work. This was not explored because it is not central to this research. An area for future investigation is the performance evaluation of different forwarding schemes in different scenarios. Differences in forwarding schemes affect the overhead of the system as certain schemes will have lower overhead in certain scenarios than other schemes. Broadcast flooding provides an upper bound of performance as it has the greatest overhead and, likely, the highest probability of delivery. A random forwarding delay from 50 to 250 ms was included to avoid the broadcast storm problem [53].

Creating Interference

To fully demonstrate how the different assurance levels perform in poor conditions, such conditions had to be created. The first attempt added traffic generators to the same ad hoc network. Two generator nodes sent 100-KB/s multicast streams into the network. Multicast streams were chosen to avoid the inclusion of MAC-layer acknowledgements. This ensured higher traffic throughput. This proved to have little effect. The problem with this approach was that the interfering nodes had to contend with the MAC protocol which was designed to deal with nodes that want to dominate the medium. This approach tried to force packet loss through medium contention and was not effective.

A better approach was to create radio frequency-level (RF) interference. A second IEEE 802.11b network was created and set to the same frequency channel as the test network. In this case the second network was an infrastructure network. The traffic generation nodes were placed on the "interference" network and generated 100-KB/s multicast streams. Two nodes were used in the single-hop scenario and four were used for the multi-hop scenario. This proved to produce the desired effect and packets were lost on the test network.

Reducing Transmission Range

To create a multi-hop network in the amount of space available for testing, the transmission range of the nodes needed to be reduced. Initial tests showed that nodes placed at opposite ends of the area could still contact each other with little packet loss. Aluminum cooking pots were used first, but worked too well and secluded the node completely. Disposable, aluminum loaf baking pans covered with a few layers of aluminum foil proved to provide enough signal attenuation to create a multi-hop configuration. Figure 6.3 shows one of the Axims in a "tin."

Figure 6.3: Left: An Axim in the tin. Right: with the lid on.

Test Runs

Each data set consisted of ten five-minute tests with a network of ten nodes. Each run was padded with a two-minute lead time and identical lag time. The lead and lag times counteract the lack of synchronization of the nodes in the network. Nodes were started by hand, so some nodes started generating packets before others. During both lead and lag times, a node was listening but not generating any packets. Queries were generated by each node at a uniform rate ranging from 15 to 45 seconds between queries. Query generation was kept low to avoid congestion in the network caused by queries. Service discovery, currently and in the near future, is a low bandwidth application. It is hard to contrive scenarios where applications would need to discovery new resources frequently. Using the current discovery examples of UPnP devices, Microsoft's Server Message Block protocol, and domain name system queries, discovery is expected to be infrequent and low volume. These characteristics were observed through a packet capture of my traffic during normal daily activity, including surfing the World Wide Web, reading and sending email, and printing documents. Starting new discovery sessions every 15 to 45 seconds is more often than most scenarios will encounter.

6.2 Results and Analysis

To analyze the effectiveness of using multiple assurance levels, the following metrics were used.

- Missed packet percentage: The percentage of sent (unique) packets that were missed by one or more nodes.
- Nodes per missed packet: The number of nodes that missed a packet. This metric only considers packets that were missed by at least one node.

6.2.1 Single-Hop Configuration

In the "close" configuration, four different test scenarios were used. They were:

1. No forwarding with no interference,
2. No forwarding with interference,
3. Flooding with no interference, and
4. Flooding with interference.

Interference consisted of two nodes generating 100-KB/s streams. One node sent 1000-byte packets and the other sent 100-byte packets. The values were chosen to model interference sources with both large and small interference times. As mentioned previously, these nodes were located on a network occupying the same channel as the test network.

Figure 6.4 shows the average percentage of missed packets for each assurance level and for the average of all four scenarios. The average is included to show general performance of the three assurance levels. The scenarios without forwarding are included to represent the performance of the scheme on a one-try basis. Figure 6.5 shows the standard deviation of the percentages of missed packets for the resulting values.

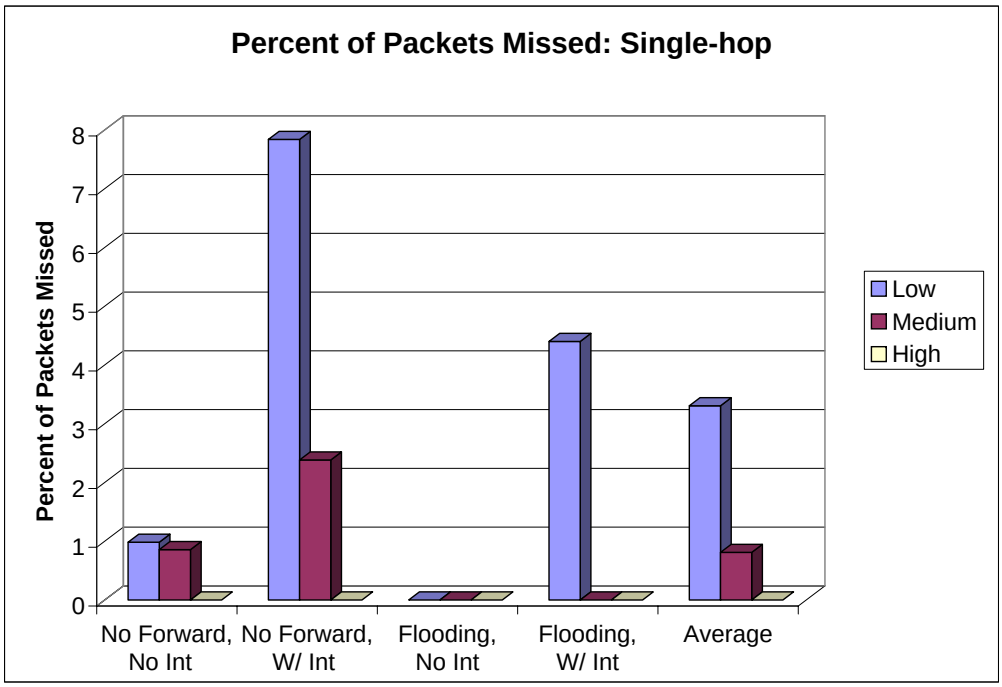

Figure 6.4: Close configuration: packet miss percentage.

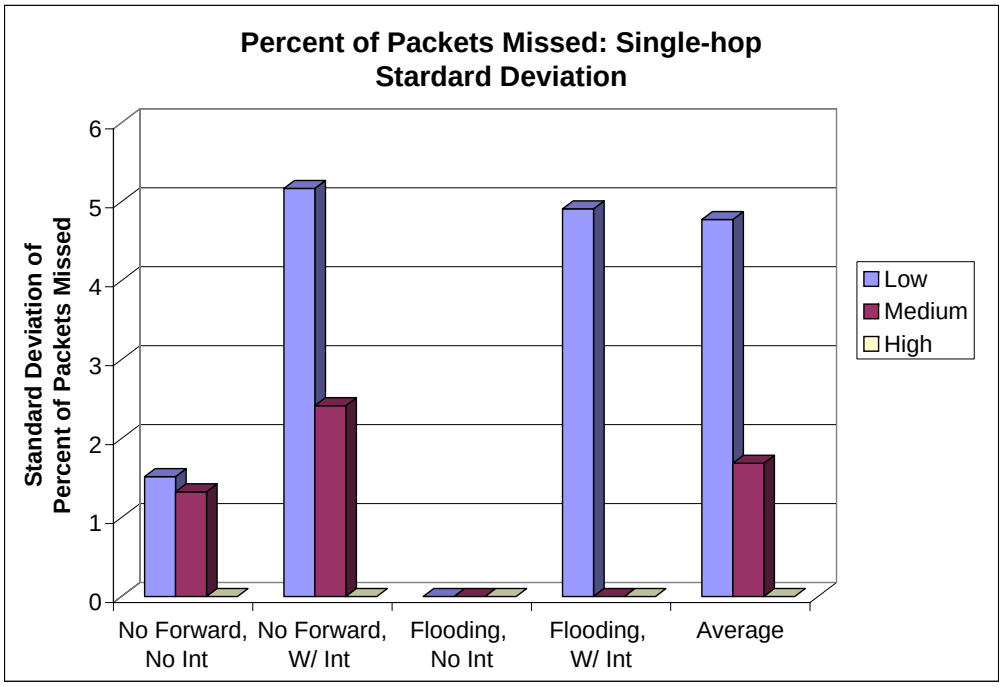

Figure 6.5: Close configuration: standard deviation of packet miss percentage.

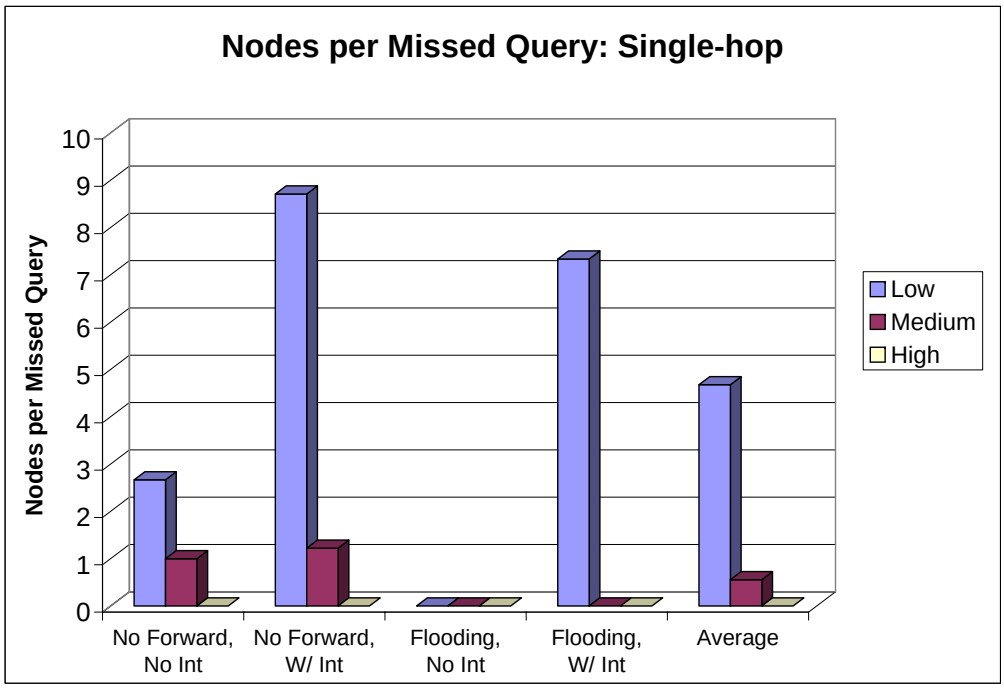

Figure 6.6: Close configuration: nodes per missed packet.

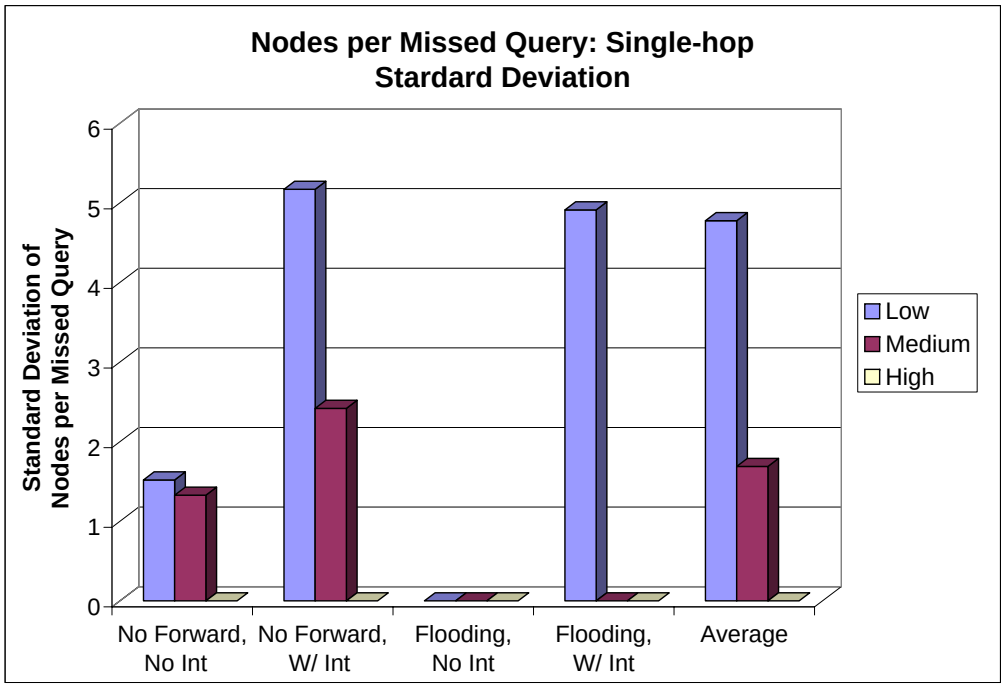

Figure 6.7: Close configuration: standard deviation of nodes per missed packet.

Figure 6.6 shows the average number of nodes that missed each "missed packet" as well as the average for all four scenarios. Again, the average was included to provide a general performance level for each assurance level. Figure 6.7 shows the standard deviation of the number of nodes per missed query.

Observations

In the four tested scenarios, the lowest assurance scheme had the highest percentage of lost packets and the largest number of nodes per missed packet. The medium assurance scheme had a lower likelihood of losing packets and a much lower node per missed packet ratio. The high assurance scheme lost no packets in any of the tests. The addition of forwarding reduced the percentage of lost packets in all cases and reduced the average number of nodes that missed a missed packet. The interference increased the percentage of lost packets and the number of nodes that missed a missed packet. These results agree with our initial hypothesis.

The two scenarios without forwarding show the effectiveness of the IEEE 802.11 acknowledgements and multiple unicast sends. The addition of forwarding in the close proximity scenario shows the effect of the redundancy added by flooding, increasing the delivery probability to 100%. Regarding delivery assurance, the three schemes provide varying levels of assurance, as desired.

Upon observation of the raw data, it should be noted that there were only three instances where *all* nodes missed a packet. All three were in the scenario with interference and no forwarding. This scenario has the highest percentage of missed packets and number of nodes per missed packet, as well. More instances where all nodes missed a packet were expected in both scenarios with no forwarding.

Analysis

As desired, the three assurance levels provide three distinct levels of delivery probability and nodes per missed packet. The low assurance level has the lowest probability of delivery and the largest number of nodes that miss a missed message. The low assurance level provides a baseline for comparison as it is the scheme used by all of the current discovery protocols. The medium assurance level experiences a higher probability of delivery, with fewer nodes affected by a missed message. Finally, the high level of assurance performs the best with the highest probability of delivery and the lowest number of nodes per missed message. These results demonstrate that this approach is feasible and works as desired.

The cost of this approach can be seen by including the approximations of Section 6.1.2 with these results. In the scenario where forwarding is not used, the number of paths is one, $P = 1$. In these two scenarios, there is still a recognizable difference in the performance of the three assurance levels. The overhead incurred by this scenario is much lower than with a forwarding scheme due to to lack of redundancy. Since the performance is still acceptable without forwarding, the forwarding scheme does not contribute much except more traffic to the medium. This shows the importance of choosing a forwarding scheme that can complement the application scenario. While different types of forwarding schemes are not addressed in this research, the application of different schemes can dramatically change the overhead and performance of the MADEP.

The standard deviation indicates the reliability of the average value. The higher the deviation the less reliable the average is because the data varies more. The opposite is true for the reverse and the smaller the deviation the less the data varies from the average. The comparatively high standard deviation of the percent of packets missed using the low assurance level means that it is harder to accurately rely on the average to predict whether a packet will be received by all nodes or not. The medium assurance level has a lower standard deviation, but is higher than the high assurance level, meaning the average is more reliable than the low assurance level, but not as reliable as the high assurance level.

With regard to the number of nodes per missed packet, the relatively high standard deviation in both low assurance and interference scenarios shows that the interference caused minimal problems sometimes and only a small number of nodes missed a packet, while also causing much more drastic effects and causing a large number of nodes to miss a packet other times. This data demonstrates how much of an impact interference may have on a network and how unreliable current discovery mechanisms can be when it is present. Conversely, the lower standard deviation in the medium and high assurance scenarios shows that these schemes, while possibly lossy, give a more consistent delivery probability when interference is present than the low assurance scheme.

The low assurance level, with the largest percentage of lost packets and largest standard deviation, proves to have the poorest delivery performance, outright, and it also has the least reliable average making delivery assumption difficult. The medium assurance level has a lower percentage of lost packets and lower standard deviation, meaning it is more reliable and exhibits better performance, but it does not perform as well as the high assurance level scheme. The high assurance level scheme offers a high probability of delivery with little variation in performance.

Actual Traffic

While it is infeasible to gather exact overhead data for the experiments, due to the lack of available tools, Observer [54] a network analyzer was used to gather approximate upper bound values for the variables in Equation 6.1 for the single-hop configuration. Table 6.2 shows the approximate overhead for all three assurance levels in a single-hop configuration with 10 nodes. The lower-bound refers to the successful reception of the query by the receiver on the first attempt, meaning that $R_1 = 0$ and $R_2 = 0$. The upper-bound refers to the successful reception of the query by the receiver on the final attempt by both the MAC and TCP layers, meaning that $R_1 = 3$ and $R_2 = 4$[1]. For all calculations, the data

[1]Since Observer must be run on a separate node it is possible that all traffic was not received and there may have been retransmission attempts that were unseen. The evaluation of these values was performed

Table 6.2: Overhead for MADEP in bytes for the single-hop network ($N_C = 9$).

	Lower-bound $R_1 = 0, R_2 = 0$	Upper-bound $R_1 = 3, R_2 = 4$
Low Assurance $x = 0$	2,925	2,925
Medium Assurance $x = 1, S_1 = 14bytes$	20,800	86,784
High Assurance $x = 2, S_1 = 14bytes, S_2 = 54bytes$	25,152	503,040

frame size was 325 bytes, the MAC acknowledgement frame was 14 bytes, and the TCP acknowledgement frame was 54 bytes. This was the size of the query used for testing. The MAC and TCP acknowledgement frame sizes were observed from a packet trace.

Table 6.2 shows the overhead for the three assurance levels. When compared with Figures 6.4 and 6.6, the reason for the difference in reliability is evident. While the low assurance level has a much smaller overhead, it has the highest packet miss percentage and the highest nodes per missed packet. The high assurance level transmits more data to deliver a query to all nodes, but all nodes will receive the query. In essence, the overhead of Table 6.2 represents the cost of increasing the reliability of delivery.

6.2.2 Multi-Hop Configuration

Four tests were also conducted in the multi-hop or "Aware home"configuration described previously. The following test scenarios were used:

1. 6 nodes in tins, 4 "naked" nodes without interference,
2. 6 nodes in tins, 4 "naked" nodes with interference,
3. 10 nodes in tins without interference, and
4. 10 nodes in tins with interference.

multiple times to obtain approximate values. Additionally, these values are implementation specific and may vary for different platforms.

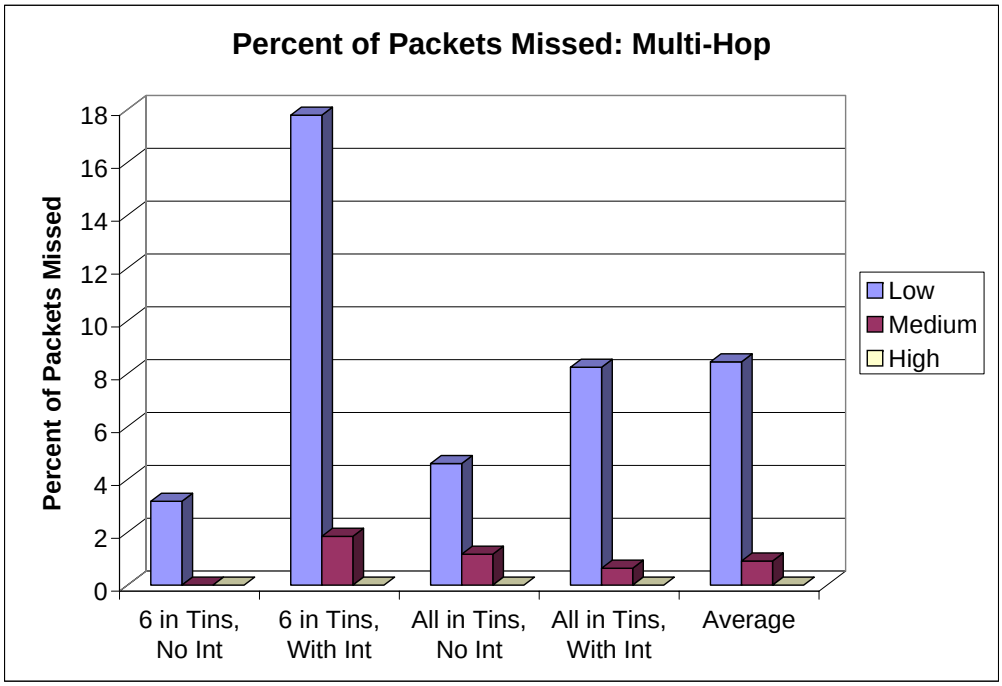

Figure 6.8: Aware home configuration: packet miss percentage.

The tin configurations, one with six nodes in tins and the other with all nodes in tins, were used to model situations where nodes may have different transmission powers and where they have the same transmission power, respectively.

Interference in this scenario consisted of four nodes placed among the ten test nodes, each generating 100-KB/s streams. Two nodes generated 1000-byte packets and two nodes generated 100-byte packets. These nodes were located on a network with an overlapping channel to the test network. Figures 6.8 and 6.9 show the data gathered in these tests. Figure 6.10 shows the standard deviation of the percentage of missed packets. Figure 6.11 shows the standard deviation of the number of nodes per missed packet.

Observations

As with the single-hop scenarios, the three assurance levels demonstrated similar performance characteristics. The low level had the greatest loss percentage and the highest number of

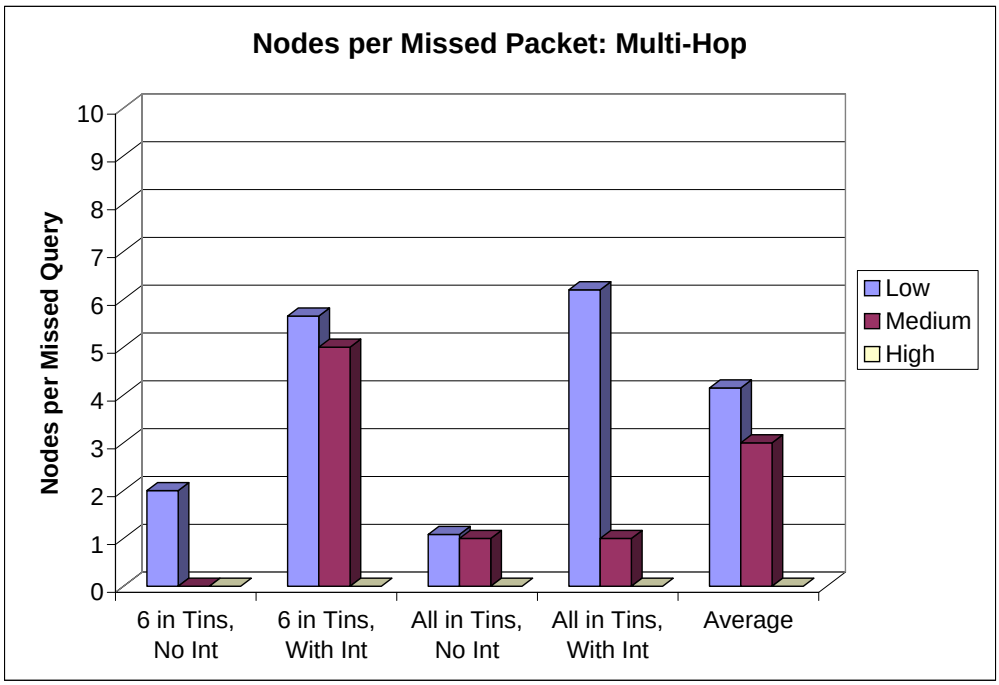

Figure 6.9: Aware home configuration: nodes per missed packet.

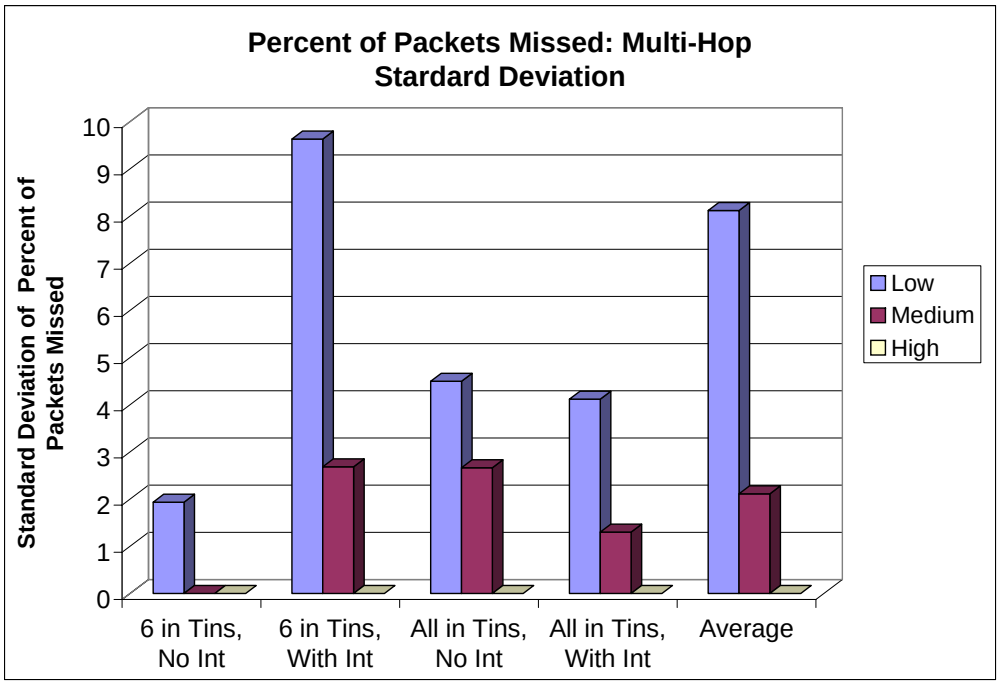

Figure 6.10: Aware home configuration: standard deviation of packet miss percentage.

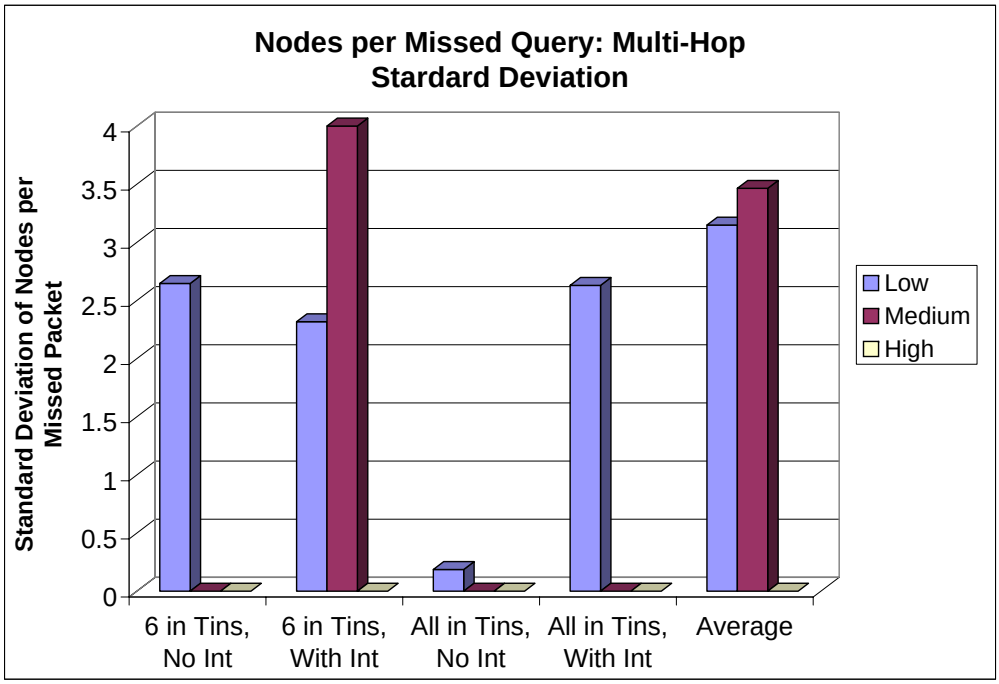

Figure 6.11: Aware home configuration: standard deviation of nodes per missed packet.

nodes per missed packet. The low level still serves as a point of comparison as it is the scheme employed by all of the available service discovery protocols. The medium level had fewer losses and fewer nodes per lost packet. The high level of assurance had no lost packets.

From the figures, the most pronounced difference is the large number of missed packets and high nodes per missed packet level in the heterogeneous tin configuration with interference for the medium assurance level. This is a result of a large number of packets missed by all nodes in the network. Since the medium assurance level uses multiple, individual transmissions, the probability of all nodes missing a transmission decreases as the number of nodes increases. The homogeneous scenario with interference also exhibits a high number of nodes per missed packet level for the low assurance level.

Upon observation of the raw data, it should be noted that in the heterogeneous scenario with interference, nodes in tins incur higher reception miss rates and sent a majority of the packets that were missed by all nodes in the network. In the scenario where all nodes are in tins, the miss reception and sending rates are more evenly distributed among all nodes.

Analysis

This test scenario shows the performance of the MADEP in a multi-hop environment where forwarding is required, unlike the previous test scenario. Like the previous test, the results show a distinct difference between the three assurance levels in both low and high interference environments. While both the heterogeneous and homogeneous configuration show similar traits the heterogeneous configuration has a relatively low level packet loss rate and medium nodes per missed packet for the interference case. The first hypothesis of why nodes in this scenario were missing packets was the existence of asymmetric transmission links. Upon initial inspection of the data, this seemed untrue as there were no one-way transmission scenarios. Upon further inspection of the raw data, it was found that a large number of the low assurance level misses were total misses and none of the nodes received the packet. Additionally, a majority of the missed packets in both the low and medium assurance levels were sent by nodes in tins. This leads to the belief that the interfering nodes were simply overpowering the tin nodes. All interference nodes were not in tins, therefore they should have more transmit power, as viewed at the receiver, than those inhibited by the tins. I have an interest in further pursuing configurations with heterogenous node configurations as this is a likely scenario for pervasive computing as less powerful nodes may have a lower transmit power than larger devices.

In the case of the low assurance level, the standard deviation of the percent of packets missed is high, meaning that assuming a packet will be delivered based on the average is not likely. As the standard deviation shrinks, in the case of the medium and high assurance levels, the likelihood of delivery closer to the average is more likely, in the case of medium level and very likely, in the case of the high level.

The standard deviation of the number of nodes per missed packet does not follow the same trend for all scenarios. The heterogeneous scenario with interference differs from the other scenarios in that the medium assurance level has a higher standard deviation than the the low assurance level. This was caused by multiple instances of a large number of nodes missing

a packet, as mentioned in Section 6.2.2. Again, the cause for this is likely the heterogeneous transmission power caused by the tins.

Similar to the single-hop scenarios, the low assurance level has the lowest probability of disseminating information throughout the network. Since all of the current service discovery protocols use this style of delivery, they can provide little assurance that queries are populated to all of the node in the network. The medium and high assurance level schemes, available with the MADEP, prove to have higher probabilities of delivery giving the application a selection of assurance levels to choose from based on its needs.

The multi-hop performance is different from the single-hop performance because of the spike in the standard deviation in the heterogeneous scenario with interference. Overall, the low assurance level scheme provides the lowest likelihood of packet delivery, but it has a lower standard deviation across all of the scenarios than the medium assurance level scheme. This spike in the standard deviation in the outlying case causes the medium assurance scheme to be less consistent than the low assurance scheme, but even with the inconsistency, the medium assurance scheme is likely to provide a higher delivery percentage and less nodes per missed packet than the low assurance scheme. Overall, the high assurance scenario still has the lowest loss percentage and the lowest standard deviation.

6.2.3 Scalability

As pervasive computing environments may have a large number of nodes, scalability is a concern. The scalability of the MADEP is controlled, mostly, by the forwarding scheme. Simple network-layer forwarding, while effective, may not have enough information to make intelligent forwarding decisions. Continuing the use of application level forwarding, more informed decisions can be made to reduce overhead. Forwarding decisions may be based on the following considerations.

- Location - is this message too far away from the node to be useful?

- Mobility - will nodes involved move before this transaction is over?
- User or system preference - has this message travelled far enough?
- Quality of service - can this transaction take place with a high enough level of service quality?

Decisions based on these and similar values may help to localize traffic for a subset of communication, thereby reducing the load on the overall system. An optimization must be made to ensure a satisfactory delivery probability, as reducing redundancy may affect the delivery rate.

6.3 Summary

This chapter discussed the use of multiple assurance levels in service discovery for the delivery of queries in reactive discovery protocols. It discussed the need for such levels, as well as, a general design of them and a specific implementation. The need for up-to-date information about neighbors and how the NIB accomplishes this were presented. Using assurance levels and the NIB, multiple scenarios of delivery were tested and data was gathered on delivery probability and the severity of information delivery failure. This data was analyzed and presented. The next chapter concludes this work with a summary, a list of contributions, and closing remarks.

Chapter 7

Conclusion

This chapter summarizes the research, enumerates contributions of this work, and presents paths for future research.

7.1 Summary

In this dissertation, I have discussed the need for service discovery in pervasive computing environments and the desired operation. The inability of current service discovery schemes to effectively deal with a human-centric operating environment and diverse devices were discussed next. I chose to concentrate on the description and dissemination components of service discovery and proposed the Pervasive Service Description Language, the Pervasive Service Query Language, and the Multi-Assurance Delivery Protocol. The product of these three ideas is the majority of a service discovery solution that accommodates and processes the large amount of description information present in pervasive computing service descriptions, attempts to reduce the amount of information that must be transmitted and processed by nodes in the network, and provides multiple levels of delivery assurance for queries in a reactive, distributed discovery architecture. These traits allow this solution to provide more

reliable discovery that can accommodate the additional requirements of a pervasive computing environment. These ideas were constructed and tested against current solutions. The results of these tests indicated that the ideas presented in this research reduce the amount of data in service discovery responses through more accurate service queries and provide higher levels of delivery reliability than currently available service discovery schemes. These advantages come at the cost of additional query processing at the service nodes and greater network overhead.

7.2 Contributions

The initial contribution of this research was the service discovery framework. Using the framework, existing service discovery protocols were analyzed and found to be insufficient for pervasive computing environments. With the framework and use examples, requirements for service discovery in pervasive computing environments were enumerated. With these requirements, the PSDL and PSQL were designed to fulfill the needs of the service description component. Following that, the MADEP was designed to fulfill the service dissemination component of service discovery. The NIB was designed during the building of the MADEP to provide up-to-date information about neighboring nodes in the network. Additionally, a study of the performance of unicast and multicast delivery probability in IEEE 802.11b ad hoc networks was conducted during the design of the MADEP to provide information about a likely deployment scenario for the MADEP. Together these pieces combine to form a portion of a service discovery solution for pervasive computing environments.

7.2.1 Service Discovery Framework

The service discovery framework was a direct result of my beginning work in service discovery. It was designed to decompose service discovery schemes into easily comparable, atomic, components. It is also useful in designing new service discovery solutions. The atomic

decomposition allows for componentized design and allows for interchanging components based on the system need. This was the first known framework to break down service discovery into comparable components.

7.2.2 Description and Query Language

The PSDL and PSQL are languages that attempt to integrate the power of modern query languages into service discovery. This combination concentrates less on the content of the descriptions and more on how to organize and use the content. The desire to incorporate more powerful content manipulation has not been seen in previous service discovery schemes. The main results on the PSDL and PSQL were reported in [55]. The PSQL is the first language to bring advanced search techniques into the service discovery domain. The PSDL is the first formalized description language for service discovery, in addition to incorporating the concept of templates and subtemplates for organization of information.

7.2.3 Multi-Assurance Delivery Protocol

The Multi-Assurance Delivery Protocol (MADEP) provides various levels of location assurance within a pervasive computing environment. These levels allow a service consumer to decide what level of service location assurance is desired based on the type of service and overhead concerns. Previous service discovery schemes make no distinction between queries and often offer suboptimal information delivery in a MANET setting. The MADEP is the first distributed service discovery approach to include the concept of multiple levels of delivery assurance. These levels provide equal or better delivery probability than previous schemes.

7.2.4 Network Information Base

The Neighbor Information Base came about as a need for information about network neighbors was encountered. It is a decentralized advertisement system between nodes in an ad hoc network. Any application may add information to the NIB and have it advertised to other NIBs. Applications on other nodes can retrieve this information and use it. In this work, the main use of the NIB is to track one- and two-hop network neighbors involved in service discovery for the higher levels of delivery assurance. The NIB is the first known application agnostic, neighbor knowledge maintenance application.

7.2.5 Ad Hoc IEEE 802.11b Throughput Study

In designing the the MADEP protocol, knowledge of the underlying MAC-layer performance was needed. A study of IEEE 802.11b in ad hoc mode performance was conducted. This study concentrated on the delivery probability of unicast and multicast frames in high and low traffic scenarios. This information was used to optimize the performance of MADEP in IEEE 802.11b ad hoc networks. This is the first known study to present data on the performance of multicast in IEEE 802.11b ad hoc networks. Results of this study were published in [50].

7.2.6 UbiShare

UbiShare is a pervasive computing collaboration project that resulted from a Microsoft Embedded Systems Request for Proposals. The idea behind UbiShare is to create a service-oriented environment that is used for communicating with users and devices in a network. Though only partially finished, two undergraduate students implemented pieces of it for independent study credit. The main contribution of UbiShare was a conceptual test arena for a generalized version of my discovery work. A poster on UbiShare was presented at UbiComp 2004 [56].

7.3 Potential Future Work

The next step for this work is to make the source code for the NIB and the MADEP available to the community. To accomplish, the source code needs to be cleaned up and proper accompanying documentation created.

The main direction of future research is to build a complete service discovery library or application. This would include the PSDL, the PSQL, the NIB, and the MADEP along with the addition of a selection and interaction component. One way to accomplish this would be to use currently available ideas for these two additional components. By using the user for service selection, this component could be bypassed completely. A suitable interaction component could make use of the Simple Object Access Protocol (SOAP) [36], much in the same way that UPnP and web services use SOAP.

Within the components that I have described, there is room for additional research as well. In the area of service description a study of service templates and what information should be included is one path. Another path is further defining the functionality of the PSDL and PSQL.

For information dissemination, the choice of forwarding algorithm and development of new forwarding algorithms is one area of potential future research. Another is to implement and test the heterogeneous considerations discussed previously. Also, the integration of a reliable datagram protocol instead of TCP is highly encouraged.

Bibliography

[1] M. Weiser, "The Computer of the 21st Century," *Scientific American*, vol. 265, no. 3, pp. 66–75, Sept. 1991.

[2] S. Helal, "Standards for Service Discovery and Delivery," *IEEE Pervasive Computing*, vol. 1, no. 3, pp. 95–100, 2002.

[3] C. Lee and S. Helal, "Protocols for Service Discovery in Dynamic and Mobile Networks," *International Journal of Computer Resesarch*, vol. 11, no. 1, pp. 1–12, 2002.

[4] G. G. R. III, "Service Advertisement and Discovery: Enabling Universal Device Cooperation," *IEEE Internet Computing*, vol. 4, no. 5, pp. 18–26, Sept. 2000.

[5] C. Bettstetter and C. Renner, "A Comparison of Service Discovery Protocols and Implementation of the Service Location Protocol," in *Proc. 6th EUNICE Open European Summer School: Innovative Internet Applications*, Sept. 2000. [Online]. Available: http://wwwtgs.cs.utwente.nl/Docs/eunice/summerschool/papers/paper5-1.pdf.

[6] F. Zhu, M. W. Mutka, and L. M. Ni, "Service Discovery in Pervasive Computing Environments," *IEEE Pervasive Computing*, vol. 4, no. 4, pp. 81–90, Dec. 2005.

[7] (2003) UPnP Device Architecture 1.0. UPnP Forum. [Online]. Available: http://www.upnp.org/resources/documents.asp.

[8] M. Jeronimo and J. Weast, *UPnP Design by Example: A Software Developer's Guide to Universal Plug and Play.* Intel Press, 2003.

[9] Apple Developer Connection. (2006) Bonjour Homepage. Apple Computer. [Online]. Available: http://developer.apple.com/networking/bonjour/.

[10] S. Cheshire and M. Krockmal. (2005, June) Multicast DNS. IETF Internet Draft. [Online]. Available: http://files.multicastdns.org/draft-cheshire-dnsext-multicastdns.txt.

[11] S. Cheshire and M. Krochmal. (2005, June) DNS-Based Service Discovery. IETF Internet Draft. [Online]. Available: http://files.dns-sd.org/draft-cheshire-dnsext-dns-sd.txt.

[12] J. Liu, Q. Zhang, B. Li, W. Zhu, and J. Zhang, "A Unified Framework for Resource Discovery and QoS-Aware Provider Selection in Ad Hoc Networks," *ACM Mobile Computing and Communications Review*, vol. 6, no. 1, pp. 13–21, Jan. 2002.

[13] M. Nidd, "Service Discovery in DEAPspace," *IEEE Personal Communications*, vol. 8, no. 4, pp. 39–45, Aug. 2001.

[14] M. Nidd, "Timeliness of Service Discovery in DEAPspace," in *Proc. IEEE Int'l Workshops on Parallel Processing*, Aug. 2000, pp. 73–80.

[15] R. Hermann, D. Husemann, M. Moser, M. Nidd, C. Rohner, and A. Schade, "DEAPspace: Transient Ad-Hoc Networking of Pervasive Devices," in *First Annual Workshop on Mobile and Ad-Hoc Networking and Comptuing*, Aug. 2000, pp. 133–134.

[16] A. Malhotra and M. Maloney, *XML Schema Requirements*, World Wide Web Consortium (W3C) Std., Feb. 1999. [Online]. Available: http://www.w3.org/TR/NOTE-xml-schema-req.

[17] (1999, June) Salutation Architecture Specification V2.0C, Parts 1-3. Salutation Consortium. [Online]. Available: http://www.salutation.org/specordr.htm.

[18] E. Guttman, C. Perkins, and J. Kempf. (1999, June) RFC 2609: Service Templates and Service Schemes. IETF. [Online]. Available: http://www.ietf.org/rfc/rfc2609.txt.

[19] E. Guttman, C. Perkins, and J. Veizades. (1999, June) RFC 2608: Service Location Protocol, Version 2. IETF. [Online]. Available: http://www.ietf.org/rfc/rfc2608.txt.

[20] K. Arnold, B. O'Sullivan, R. W. Sheifler, J. Waldo, and A. Wollrath, *The Jini Specification*. Sun Microsystems, Inc., 1999.

[21] S. I. Kumaran, *Jini Technology*. Prentice Hall PTR, 2002.

[22] C. Lee, A. Helal, N. Desai, V. Verma, and B. Arslan, "Konark: A System and Protocols for Device Independent, Peer-to-Peer Discovery and Delivery of Mobile Services," *IEEE Transations on Systems, Man and Cybernetics*, vol. 33, no. 6, pp. 682–696, Nov. 2003.

[23] S. Helal, N. Desai, V. Verma, and C. Lee, "Konark: A Service Discovery and Delivery Protocol for Ad-Hoc Networks," in *Proc. IEEE Wireless Communications and Networking Conference*, 2003, pp. 2107–2113.

[24] S. E. Czerwinski, B. Y. Zhao, T. D. Hodes, A. D. Joseph, and R. H. Katz, "An Architecture for a Secure Service Discovery Service," in *Mobile Computing and Networking*, 1999, pp. 24–35. [Online]. Available: http://www.cs.berkeley.edu/~ravenben/xset/sds-mobicom.pdf.

[25] Java Technology. Sun Microsystems, Inc. [Online]. Available: http://java.sun.com/.

[26] T. Bellwood, L. Clement, and C. von Riegen, *UDDI Version 3.0.1*, OASIS Std., Oct. 2003. [Online]. Available: http://uddi.org/pubs/uddi-v3.0.1-20031014.htm.

[27] W. Adjie-Winoto, E. Schwartz, H. Balakrishnan, and J. Lilley, "The Design and Implementation of an Intentional Naming System," in *Symposium on Operating Systems Principles*, 1999, pp. 186–201. [Online]. Available: http://citeseer.ist.psu.edu/adjie-winoto99design.html.

[28] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana. (2001, Mar.) Web Service Description Lanauge (WSDL). World Wide Web Consortium (W3C) recommendation. [Online]. Available: http://www.w3.org/TR/wsdl.

[29] S. Boag, D. Chamberlin, M. F. Fernndez, D. Florescu, J. Robie, and J. Simon, *XQuery 1.0: An XML Query Language*, World Wide Web Consortium (W3C) Std., July 2004. [Online]. Available: http://www.w3.org/TR/xquery/.

[30] M. Brundage, *XQuery: The XML Query Language*. Addison-Wesley, 2004.

[31] A. Malhotra, J. Melton, J. Robie, and M. Rys, *XML Syntax for XQuery 1.0 (XQueryX)*, World Wide Web Consortium (W3C) Std., Dec. 2003. [Online]. Available: http://www.w3.org/TR/2003/WD-xqueryx-20031219/.

[32] J. L. Undercoffer, F. Perich, A. Cedilnik, L. Kagal, and A. Joshi, "A Secure Infrastructure for Service Discovery and Access in Pervasive Computing," *ACM Mobile Networks and Applications*, vol. 8, no. 2, pp. 113–125, Apr. 2003.

[33] U. C. Kozat and L. Tassiulas, "Network Layer Support for Service Discovery in Mobile Ad Hoc Networks," in *Proc. IEEE Infocom*, Mar. 2003, pp. 1965–1975. [Online]. Available: http://www.ieee-infocom.org/2003/papers/48_02.PDF.

[34] J. Allard, V. Chinta, S. Gundala, and G. G. R. III, "Jini Meets UPnP: An Architecture for Jini UPnP Interoperability," in *2003 Symposium on Applications and the Internet (SAINT03)*, Jan. 2003, pp. 268–275.

[35] B. Miller, "Mapping Salutation Architecture APIs to Bluetooth Service Discovery Layer," Salutation Consortium, Tech. Rep., 1999. [Online]. Available: http://www.salutation.org/whitepaper/BtoothMapping.pdf.

[36] M. Gudgin, M. Hadley, N. Mendelsohn, J.-J. Morea, and H. F. Nielsen, *SOAP Version 1.2 Part 1: Messenging Framework*, World Wide Web Consortium (W3C) Std., June 2003.

[37] P. Miller, *TCP/IP Explained*. Digital Press, 1996.

[38] S.-C. Wang, Y.-M. Chen, T.-H. Lee, and A. Helmy, "Performance Evaluations of Hybrid IEEE 802.11b and 802.11g Wireless Networks," in *Proceedings of 24th IEEE International Performance, Computing, and Communications Conference(IPCCC2005)*, Apr. 2005, pp. 111–118.

[39] Z. Zeng, B. Allen, Z. Liu, and A. Aghvami, "Evaluation of IEEE 802.11b and Bluetooth Coexistence in an Office Environment," in *Fifth IEE International Conference on 3G Mobile Communication Technologies 2004 (3G 2004)*, 2004, pp. 54–58.

[40] M. Heusse, F. Rousseau, G. Berger-Sabbatel, and A. Duda, "Performance anomaly of 802.11b," in *Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM2003)*, 2003, pp. 836–843.

[41] J. del Prado and S. Choi, "Experimental Study on Co-existence of 802.11b with Alien Devices," in *Proceedings of the IEEE Vehicular Technology Conference (VTC2001)*, Oct. 2001, pp. 977–981.

[42] A. Friday, N. Davies, N. Wallbank, E. Catterall, and S. Pink, "Supporting Service Discovery, Querying and Interaction in Ubiquitous Computing Environments," *Wireless Networks*, vol. 10, no. 6, pp. 631–641, 2004.

[43] C. Perkins. (2002, Aug.) IP Mobility Support for IPv4. IETF. [Online]. Available: http://www.ietf.org/rfc/rfc3344.txt.

[44] D. Johnson, C. Perkins, and J. Arkko. (2004, June) Mobility Support in IPv6. IETF. [Online]. Available: http://www.ietf.org/rfc/rfc3775.txt.

[45] S. Deering and R. Hinden. (1998, Dec.) RFC 2460: Internet Protocol version 6 (IPv6) Specification. IETF. [Online]. Available: http://www.ietf.org/rfc/rfc2460.txt.

[46] World Wide Web Consortium (W3C). [Online]. Available: http://www.w3c.org.

[47] E. V. der Vlist, *XML Schema*. O'Reilly and Associates, 2002.

[48] J. Goyvaerts. (2004, Sept.) Regular Expression Tutorial. [Online]. Available: http://www.regular-expressions.info/tutorial.html.

[49] J. Macker and S. D. Team. (2006, Mar.) Simplified Multicast Forwarding for MANET: draft-ietf-manet-smf-02. IETF. [Online]. Available: http://www.ietf.org/internet-drafts/draft-ietf-manet-smf-02.txt.

[50] M. S. Thompson and S. F. Midkiff, "Experiences using IEEE 802.11b for Service Discovery," in *The Third IEEE International Workshop on Mobile Peer-to-Peer Computing (MP2P2006)*, Mar. 2006, pp. 146–150.

[51] K. Moore. The Windows Sockets Lame List. Windows Sockets Vendor Community. [Online]. Available: http://www.sockets.com/lamelist.htm.

[52] C. D. Kidd, R. Orr, G. D. Abowd, C. G. Atkeson, I. A. Essa, B. MacIntyre, E. D. Mynatt, T. Starner, and W. Newstetter, "The Aware Home: A Living Laboratory for Ubiquitous Computing Research," in *Cooperative Buildings*, 1999, pp. 191–198. [Online]. Available: http://citeseer.ist.psu.edu/kidd99aware.html.

[53] S.-Y. Ni, Y.-C. Tseng, Y.-S. Chen, and J.-P. Sheu, "The Broadcast Storm Problem in a Mobile Ad Hoc Network," in *MobiCom '99: Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking*. ACM Press, 1999, pp. 151–162.

[54] Observer. Network Instruments. [Online]. Available: http://www.networkinstruments.com/products/observer.html.

[55] M. S. Thompson and S. F. Midkiff, "Service Descriptions for Pervasive Service Discovery," in *The First International Workshop on Services and Infrastructure for the Ubiquitous and Mobile Internet (SIUMI05)*, June 2005, pp. 273–279.

[56] M. S. Thompson, W. O. Plymale, C. T. Hager, K. Henderson, S. F. Midkiff, L. A. DaSilva, N. J. Davis, and J. S. Park, "Ad-Hoc Networking Support for Pervasive Collaboration," in *Adjunct Proc. of the Sixth Annual Conference on Ubiquitous Computing (UbiComp04)*, Sept. 2004. [Online]. Available: http://ubicomp.org/ubicomp2004/adjunct/posters/.

Appendix A

Implementation Details

This chapter includes screen captures and implementation details of the different applications that have been written during the course of this research.

A.1 Query Simulator Application

This section contains screen captures of the query simulator running on the Windows Mobile 2003 desktop emulator. Each figure is of a different parameter set. The operation of the query simulator can be expressed using the pseudo-code of Figure A.1. Figures A.2 through A.7 are screen captures of the query simulator application running on the Windows Mobile emulator.

```
list = GenerateRandomServiceList ( <list parameters> ) ;
for ( 1 to runLength )
{
    query = GenerateRandomQuery ( <query parameters> ) ;
    startTime = GetCurrentTime ( ) ;
    PreProcessingOperations ( query, list ) ;
    matches[] = QueryList ( list , query ) ;
    foreach ( match in matches )
    {
        PostProcessingOperations ( match ) ;
    }
    stopTime = GetCurrentTime ( ) ;
}
```

Figure A.1: Query simulator pseudocode.

Figure A.2: Query simulator: general configuration.

Figure A.3: Query simulator: debugging output.

Figure A.4: Query simulator: pre-processing configuration.

Figure A.5: Query simulator: post-processing configuration.

Figure A.6: Query simulator: evaluation configuration.

Figure A.7: Query simulator: service generation configuration.

A.2 Traffic Generator and Receiver for IEEE 802.11b Testing

This section contains specific details involving the traffic generator and receiver written for mobile devices. This application can send and receive unicast or multicast UDP packets. Parameters of the generator include the multicast IP address, the port number, the size of the packets, the number of times to send a packet, and the inter-packet time, in milliseconds. Reception parameters include the multicast address and port number on which to listen. Screen captures of the traffic generation application are shown in Figures A.8 and A.9.

Figure A.8: Traffic generator: sending configuration.

Figure A.9: Traffic generator: receiving configuration.

A.3 Network Information Base Application

This section details the operation and components of the NIB. It is divided into two parts, the NIB application itself and the application library, the NibLib, to interact with the NIB on a host.

A.3.1 NIB Application

The NIB application is composed of 4 threads.

- Advertisement thread: handles sending the information in the NIB to other NIBs at a specified interval.
- Maintenance thread: handles removing stale entries from the NIB after they have not been updated for a specified amount of time.
- Advertisement reception thread: complementing the advertisement thread, this thread handles receiving and processing advertisements from other NIBs. This is not actually a single thread as Winsock asynchronous receive methods are used to process received datagrams.
- Client thread: handles receiving and replying to updates and queries from applications on the device. This thread interacts with the NibLib. This is also not an actual thread as it, too, uses asynchronous Winsock methods.

A.3.2 NibLib

The NibLib is an application programming interface (API) that allows an application to update and query the local NIB. The definition of NibLib is given in Figure A.10. A screen capture of the NIB application is shown in Figure A.11.

```
public class NibLib_v2
{
    public event Debugging.DebugDel DebugEvent;
        // point to pass out debugging information,
        //      see Debugging class

    public struct Proto
    {
        public string name;
        public string data;
        // the name and data fields for a single protocol entity.
        // the data can be anything will fit in a string;
        //      I suggest XML.
    }

    public class Node
    {
        public event Debugging.DebugDel DebugEvent;
        // Debug point
        public string id;
        // id of this node entry (usually the IP address)
        public int hops;
        // number of hops to 'this' node
        public DateTime updated;
        // time of last update
        public DateTime added;
        // time this 'Node' was added to the list
        public ArrayList protos;
        // arraylist of 'Proto's
        public Node();
        // Node constructor
        public string Serialize();
        // used to Serialize this class into a string
        // RETURN: a string to be advertised
        public void Deserialize(string info);
        // the reverse of Serialize(), used to deserialize
        // from an advertisement
        // string info: (should be) the output of Serialize()
    }
```

Figure A.10: NibLib version 2 API (partial).

```
    public NibLib_v2();
    // niblib constructor
    public void Init();
    // called to start the NibLib
    // a NIB connection attempt is required
    public void Stop();
    // shuts down this NibLib instance
    public void UpdateProto(string protoName, string protoData);
    // update the the desired proto field in the local NIB
    // string protoName: the name of the protocol
    // string protoData: information about the protocol
    public ArrayList QueryByProto(string protoName, int maxHops);
    // query for nodes by 'proto' name
    // string protoName: the protocol name of interest
    //       '*': wildcard, returns all protocols
    // int maxHops: maximum number of hops to search,
    //   nodes beyond this, will not be returned
    // RETURN: an ArrayList of matching 'Node' objects
    // NOTE: returns nodes with any information present
    //       for this protocol name
    public ArrayList QueryByNode(string nodeName, int maxHops);
    // query for nodes by the node 'id' field
    // string nodeName: node name, corresponding to the
    //       Node.id field
    //       '*': wildcard for all nodes; NIB dump
    // int maxHops: maximum number of hops to search
    // RETURN: ArrayList of matching 'Node' objects
}
```

Figure A.10: NibLib version 2 API (continued).

Figure A.11: NIB application: list of neighbors and number of hops.

A.4 Discovery Test Application

This section contains screenshots of the discovery test application in Figures A.12 through A.14. The application is composed of the following four threads.

- Generator thread: handles generating random queries and sending them.
- Maintenance thread: handles removing previously seen packets that have become stale (older than a specified amount of time).
- Socket thread pool: there is a single UDP thread for all UDP operations. There is a single TCP accept thread to handle incoming TCP connections. Each established or connecting socket has both a send and receive thread. The send thread is responsible for connecting, if necessary, and sending data. The receive thread receives data and executes the query processing methods.
- Control thread: handles starting and stopping all other threads based on set run and wait times.

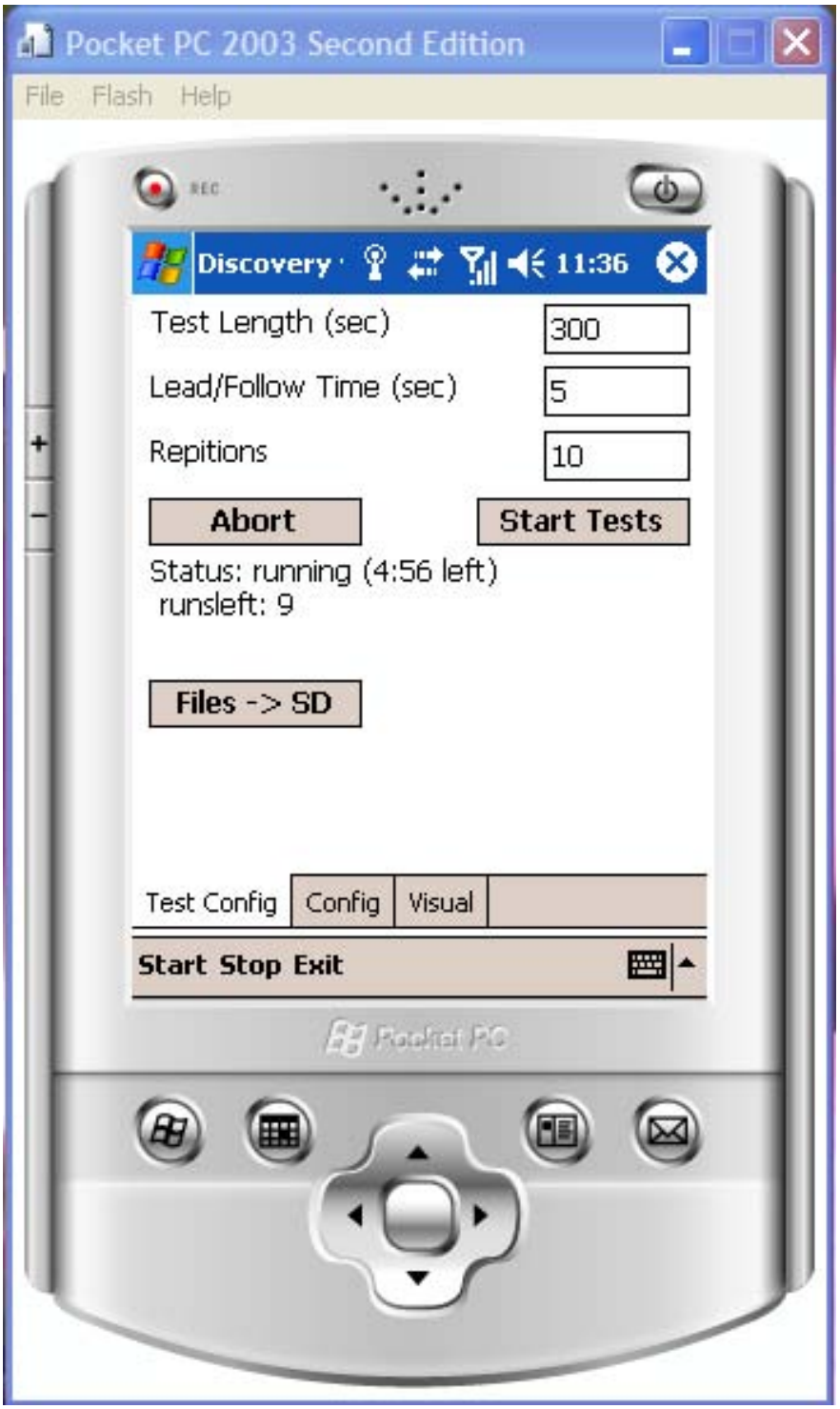

Figure A.12: Discovery application: test run configuration.

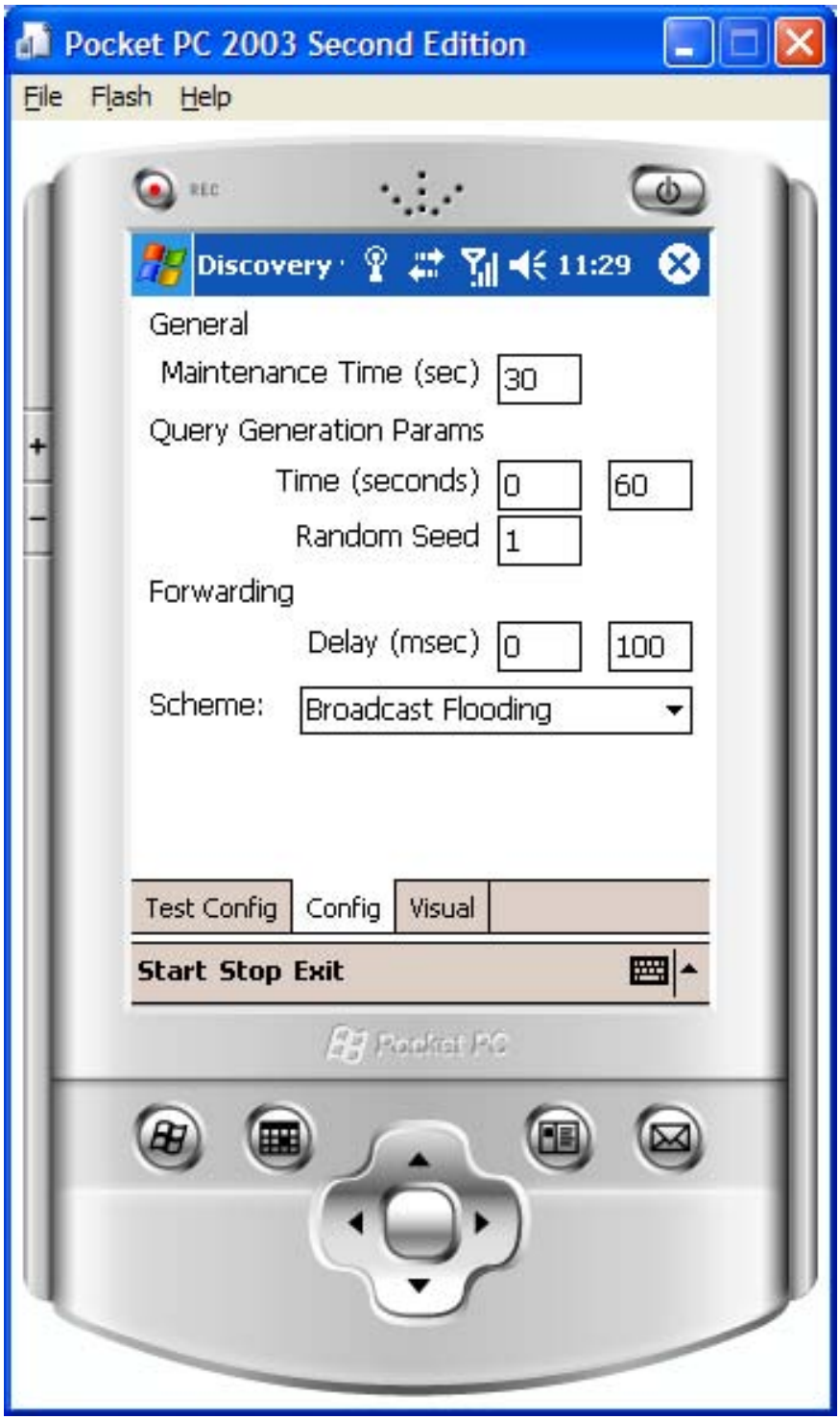

Figure A.13: Discovery application: generation and forwarding configuration.

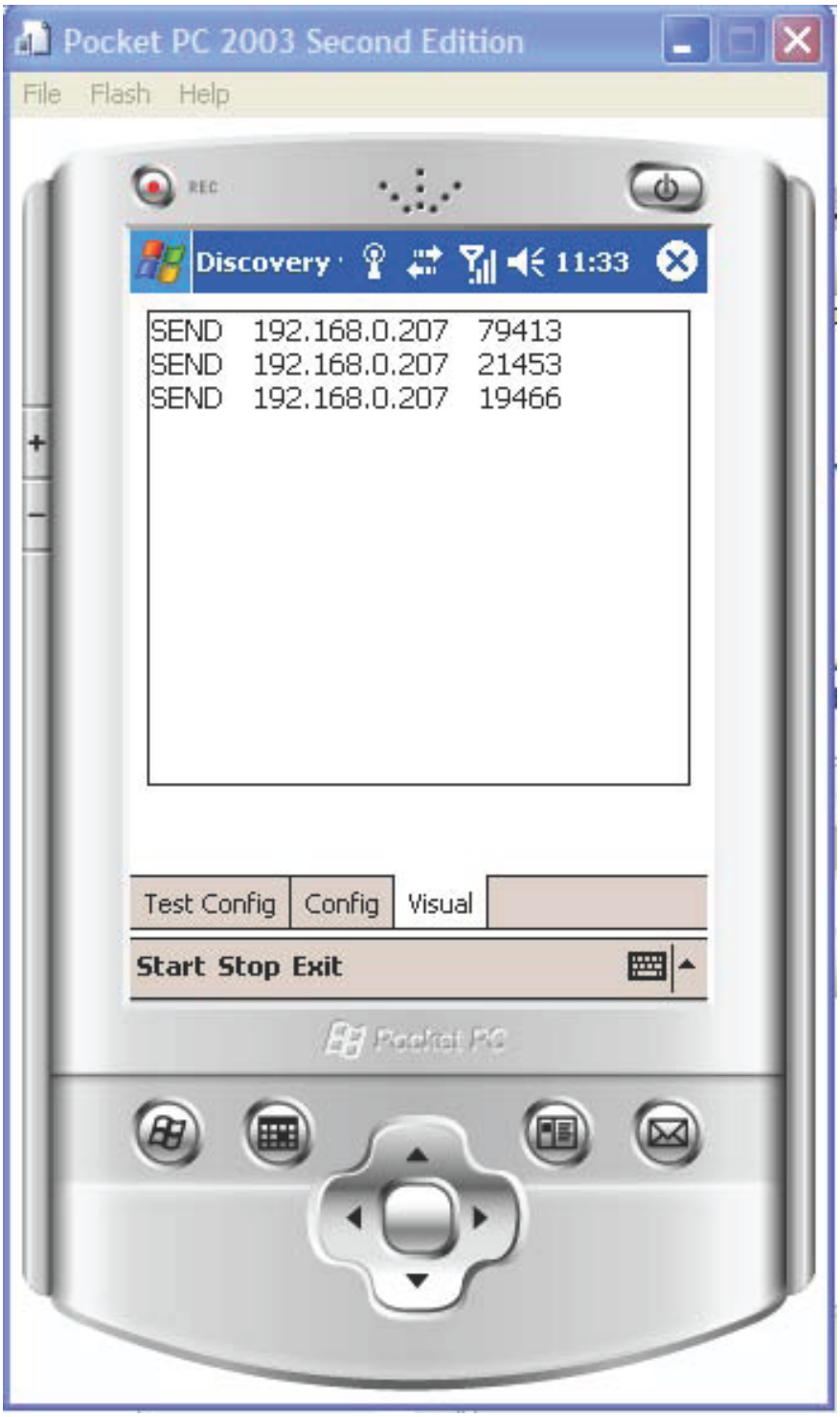

Figure A.14: Discovery application: visual status.