Available online at www.sciencedirect.com
SciVerse ScienceDirect
Procedia Computer Science 10 (2012) 976 – 983
Procedia Computer Science

The 2nd International Symposium on Frontiers in Ambient and Mobile Systems (FAMS)

A Peer-to-Peer Architecture for Remote Service Discovery[☆]

Stefan D. Bruda[a], Farzad Salehi[a], Yasir Malik[b], Bessam Abdulrazak[b]

[a]*Bishop's University, Sherbrooke, QC J1M 1Z7, Canada*
[b]*Université de Sherbrooke, Sherbrooke, QC J1K 2R1, Canada*

Abstract

We propose a new service discovery architecture for enabling typical (local) service discovery mechanisms (without the ability of remote service discovery) to discover services remotely. Our architecture does not depend on any particular local discovery protocol and is realized in a fully distributed (namely, peer-to-peer) manner.

Keywords: service discovery, peer to peer, pervasive computing

1. Introduction

In 1988 Mark Weiser gave birth to the vision of anytime, anywhere computing or "ubiquitous computing." *Ubiquitous computing is the method of enhancing computer use by making many computers available throughout the physical environment, but making them effectively invisible to the user* [1]. Ubiquitous computing is also known as "pervasive computing" (which we use throughout this paper) or "ambient intelligence." Computing anytime, anywhere, and in any device means a massive presence of computing devices in the physical world. At the same time, people should be able to access information and computation in a user-centric way i.e., user interaction with such a system must be natural and comfortable. Pervasive computing is thus a migration from desktop computing to computing integrated into everyday objects.

Pervasive computing offers an environment saturated with sensors, actuators, cameras, and all other sorts of computing devices; all these devices should work together and satisfy users' needs with minimal user intervention. Service discovery protocols are one tool that help accomplishing this, as they allow different applications, services, electronic and computing devices to recognize each other and work with each other with no human intervention. Many service discovery protocols have been designed, the dominant ones including Microsoft Universal Plug and Play (UPnP) [2], Bluetooth Service Discovery Protocol [3], Apple's Bonjour, and Sun's Jini technology [4].

All the available service discovery protocols are designed for home or enterprise environments [5]; the pervasive computing environment is however far more heterogeneous and sophisticated than any home or

[☆]This research was supported by the Natural Sciences and Engineering Research Council of Canada. Part of this work was also supported by Bishop's University.

Email addresses: `stefan@bruda.ca` (Stefan D. Bruda), `salehi@cs.ubishops.ca` (Farzad Salehi), `yasir.malik@usherbrooke.ca` (Yasir Malik), `bessam.abdulrazak@usherbrooke.ca` (Bessam Abdulrazak) enterprise one. Furthermore, most service discovery protocols are designed to work only in a local area network (LAN) [6]. Indeed, many services in a pervasive computing environment are physically-oriented (examples include video projectors and coffee machines), meaning that they are useful for the users in the same physical environment and not for distant users. Still, many other services are not physically-oriented and they can be accessed and have to be accessed by users physically far away from them (such as accessing the digital data in someone's home, remotely monitoring sensors, actuators and cameras present in a place for security or health care purposes, and so on). Computing anywhere is also the very definition of the concept of pervasive computing. While it is not possible to provide all services anywhere, remote access to any services (from anywhere) makes sense. For this purpose service discovery protocols must be able to discover services remotely in order to be able to work in a pervasive computing environment. A combination of existing technologies and services enables some level of remote access, but seamless discovery and control of remote services is currently not possible [6, 7, 8].

The objective of this work is to enable different local service discovery networks (such as UPnP networks) to discover services in other similar networks. We lay the basis of such remote service discovery by proposing a suitable architecture, where each local network is enhanced by a function called service mirror builder. A service mirror builder presents local services as remote services to other networks, and also builds mirrors of remote services in its local network. The process of finding a remote service is done with the help of the distributed peer-to-peer search protocol such as Gnutella.

1.1. Motivation

Pervasive computing means between others spatial heterogeneity: some places offer all the needed services and others only have a few services to offer. A combination of remote and local services is sometimes needed to satisfy the user's needs. The following scenarios motivate our quest for remote service discovery.

One example of pervasive computing environment is a *connected (smart) home*, which is a dwelling incorporating a communications network that connects key devices (sensors and actuators, electrical appliances) and allows them to be remotely controlled, monitored or accessed [7]. To realize a smart home we thus need to have a mechanism to access its services remotely. In addition, most of us desire seamless storage, access and consumption of digital content from and to any compatible digital device in a home or smart home; ideally, users should be able to access their residential services from anywhere using any type of terminal [8]. Overall use cases for remote service discovery therefore include lighting, residential climate control, home theater, audio entertainment systems, domestic security, domestic health care systems, etc.

Vendors need to connect to their devices for various purposes such as to update their software or perform routine checks (*remote support*). Security and health care companies in particular need to be in contact with their customers and their products continuously. The information from sensors, actuators and cameras can be monitored by such companies, which can then take action in case of any threat, but also control devices to be more efficient and usable. The vendors can also advertise features and offer upgrades to their devices (*continuing close presence*).

Massively Multiplayer Games (MMGs) are traditionally supported by a client-server architecture, but such a centralized architecture lacks flexibility and can put communication and computation stress on the servers [9]. To overcome these problems inherent to centralized solutions, peer-to-peer networks are emerging as a promising architecture for MMGs [9]. Running MMGs with the help of remote service discovery and without any centralized coordinator is perhaps the best use cases to motivate our research contribution.

2. Preliminaries

2.1. Service Discovery

Most service discovery protocols have similar architectures [2, 3, 4]. We present in what follows the general functionality of these services, noting that some of them may have a reduced set of features (we choose however to include the maximal set of such features in order to cover all the services).

The architecture of most service discovery protocols involve three concepts: *device* (contains one or more services), *service* (performs actions and shows its state via state variables), and *control point* (a system that discovers and then controls services and devices). The protocols operate over an IP network.

The functioning of these protocols then involves four steps: *addressing* (all participants obtain an IP address via DHCP or Auto-IP), *discovery* (control points discover the capabilities of the devices on the network), *description* (control points find out how to invoke devices or services), and *control and eventing* (control points request actions and receive responses; they also receive status messages). Some protocols (such as UPnP) can also offer mechanisms for the presentation of services in a browser or similar.

Discovery in particular can happen in two ways. First, new devices connected to a network multicast discovery messages advertising their embedded devices and services (*discovery-advertisement*). Interested control points listen to these advertisements and then connect and control the originating devices (or only some of their services). Secondly, a new control point in the network multicasts a discovery message, searching for available devices and services (*discovery-search*). All devices in the network must listen to these messages and respond to them whenever any of their services or devices match criteria in the discovery messages.

2.2. *Peer-to-peer Networks*

A distributed network architecture may be called peer to peer (P2P) whenever the participants share a part of their own hardware resources (processing power, storage capacity, network link capacity, printers, etc.) with each other. These shared resources are necessary to provide the service and content offered by the network (e.g. file sharing or shared workspace for collaboration). Furthermore they are accessible by other peers directly, without passing through intermediate entities. The participants in such a network are thus resource (service and content) providers and at the same time resource (service and content) requesters (the "servent" concept) [10]. Peer-to-peer file sharing is a particular example of peer-to-peer network. Each peer in a P2P file sharing network is implemented by a client which uses some distributed search protocol to find other peers as well as the files that are being shared by them. Different protocols for distributed search are being used by P2P file sharing programs, the most prominent being BitTorrent [11] and Gnutella [12].

The functionality of a P2P network can be summarized as follows: The first time a servent wants to join such a network, its client software must *bootstrap* and thus find at least one other servent (node or peer) in the network. The participants in the network then use a distributed algorithm to discover (part of) the network topology (such as "ping" and "pong" messages or consolidating "super-servents").

When a client wants to search for a file (or as we will see in Section 3 for a service), it sends a *query message* to all its directly connected neighbour servents (except the one which delivered this query). The neighbour servents forward the query to their neighbours and so on. If a servent receives a query and finds a match in its directory, it will respond with a *query-hit message* containing enough information for the retrieval of the data matching the corresponding query.

3. A Distributed Architecture for Remote Service Discovery

Service discovery protocols should be able to step out of their local domain in order to find services and in turn serve the users' needs. Additionally, a control point may reside in a pervasive computing environment with heterogeneous protocols and networks; even if some otherwise available services in the local domain could not be accessed because of heterogeneity in protocols, networks, ontologies, and so on, the controller may still be able to access services within its capabilities but far from its physical location. We are proposing a new architecture that accomplishes remote service discovery in a fully distributed manner.

3.1. *The Local Network*

All the participating local networks include devices, services, and control points are connected with each other locally through some service discovery protocol (just discovery protocol henceforth). We introduce in each local network a special function called *service mirror builder*, which is a device from the point of view of the discovery protocol (containing a control point and a varying number of services). Beside the service mirror builder the network contains a number of (local) devices, services, and control points. The network is an IP based network with all of these devices connected normally through the discovery protocol (that is, addressing is established as prescribed by the protocol).

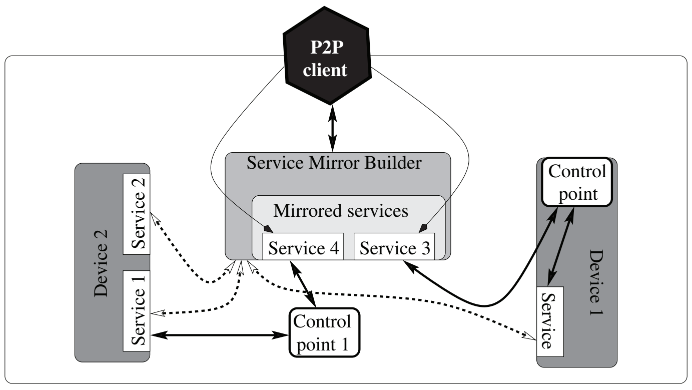

Fig. 1. Local network structure

Discovery-advertising and discovery-search then happens in the local network again as prescribed by the local protocol. The service mirror builder must be aware of all the available services in the local network, so it will not ignore any multicast message. It will also advertise its services (that are all remotely discovered as we will see later) as they become available. During any kind of discovery-search process (that is, whenever a control point becomes interested in a new service) the interested control point multicasts a discovery message, thus searching for available services and devices in the network. All the devices in the network must listen to these messages and respond whenever any of their services match the criteria specified by the respective message. The service mirror builder also listens to all these messages and for each such a message it checks whether the requested service is in the list of available local services. If it is, then the service mirror builder drops the message; otherwise, it proceeds to discovering the respective service remotely.

In addition, each local network runs a specialized P2P client software (just P2P client henceforth), that share local services to the outside world and find services requested by their service mirror builder. The P2P clients establish a P2P network between them according to the respective protocol. When the service mirror builder receives a request for a service (which is not locally available) through the discovery-search mechanism, it tries to request it from a remote network. Once such a service is found, a mirror of that service is made available in the local network. In such a case the addition is made available to the local network via the discovery-advertising mechanism.

After the discovery step (which makes the control points aware of the available services), the control points must know how to use these available services (description). Advertising messages circulated during discovery contain URLs from which the control points can retrieve the description of the respective devices. Once a control point has the device or service description it can invoke actions on that service and get result values in return. Invoking an action is a particular instance of Remote Procedure Call and is done once again according to the local discovery protocol. The major focus of this research contribution however is service discovery so we will not elaborate further on service control, eventing and presentation.

Refer to Figure 1 for a closer look at a possible local network, which includes four components: two devices, one control point, and one service mirror builder. The service mirror builder typically resides on the smart environment gateway (such as a connected home gateway).

In our example suppose that Device 1 has not introduced its service to other control points except the service mirror builder, and its control point has discovered a mirror of a remote service (Service 3). Device 2 has two embedded services (Service 1 and Service 2) which are similarly not known to the others. Then Device 2 must inform all the available control points in the network about its services; it does so by multicasting a message and thus advertising Services 1 and 2 (discovery-advertisement). The multicast message will be received by the service mirror builder and also by Device 1. The control point in Device 1 is not interested in (or not capable to control) either Service 1 or Service 2 and so it ignores this message. The service mirror builder uses this information for remote service discovery, which will be discussed later. The service mirror builder is (obviously) interested in Service 1 and Service 2. It then sends a message to Device

1 to retrieve the description of the two services.

In Figure 1 the mirrors of the remote services are shown in the service mirror builder box (Service 3 and Service 4). The service mirror builder is thus a service-discovery device that offer services but is also a control point (the latter feature is not specified explicitly in the figure).

Control point 1 is then added to the network, has its own IP address, but has not discovered any services to control yet. In such a case the newly added control point multicasts a discovery message, to which the devices respond whenever the criteria specified by the respective message match. The service mirror builder listens to all these messages and for each such a message it checks whether the requested service is in the list of available local services and proceeds to remote service discovery if necessary (as outlined above).

In our example, Service 1 in Device 2 is matched with the request of Control point 1. Therefore Device 2 unicasts a response message to Control point 1. Now that Control point 1 has discovered one of its needed services, it will ask for a description. Once the description is received, Control point 1 can control Service 1 in Device 2. Assume now that Control point 1 multicasts a discovery search message requesting a service which is not locally available (say, Service 4). The service mirror builder will recognize that this service is not locally available, and so it sends a query for that service to the local P2P client. The P2P client will then propagate that query to the P2P network, as we will show in the next section.

The overall relationship between control points and the controlled services as described above is summarized graphically in Figure 1. Solid lines incident to the control points (Control point controls Service and Service 3, Control point 1 controls Service 1 and Service 4). The service mirror builder controls all the locally available services (dashed arrows), but for the sole purpose of making them available to other networks upon request. Service 3 and Service 4 are mirrored from different networks (using the connecting P2P network).

3.2. P2P Remote Service Discovery

Servents in a P2P network can share any type of resources [13]. In our design servents are sharing their local services with remote servents. The local networks being put together via the P2P network are shown graphically in Figure 2.

As soon as the local networks and the P2P network are established (according to the existing protocols), remote service discovery can begin. Such an event happens whenever a control point requests a service which is not locally available. The service mirror builder then activates and tries to remotely discover it.

Each service mirror builder has a cached description of all of the available local services. When a control point requests a service, the service mirror builder checks in its local service directory and if there is such a service just ignores the query (since the control point can locally discover and control that service). However, if that service is not in the local directory, the service mirror builder proceeds to discover it remotely, as follows: The service mirror builder sends a request for the service to the P2P client. The P2P client issues in turn a query message asking for the requested service and sends this query to the network according to the P2P protocol. When receiving a query, each P2P client will send the included service request to the local service mirror builder, which in turn will check the availability of the requested service in its local network. Should the service be locally available, the service mirror builder will communicate this to the P2P client, which will in turn respond with a query-hit message to the original requester. The nodes that issue a query-hit also send a service description and other information back to the node that issues the query. This information will be delivered to the service mirror builder of that node, which then creates a mirror of the service in the local network. From the point of view of the control points in the local network the service looks just likes a local one and can be controlled in the usual way.

The P2P network functionality remains unchanged from the original protocol, except for the query and query-hit messages (since the original such messages are used for requesting for and responding with shared files). The query messages can easily encode requests for services instead of requests for files and the such; indeed, many P2P specifications allow extension frameworks, which can be trivially used. The payload of the a typical query-hit message can then be modified for the purpose of remote service discovery. To prevent increased complexity and extra work to define a new specification, we recommend however that these fields be left unused and extension blocks (featured by most P2P protocols) be used for remote service discovery instead.

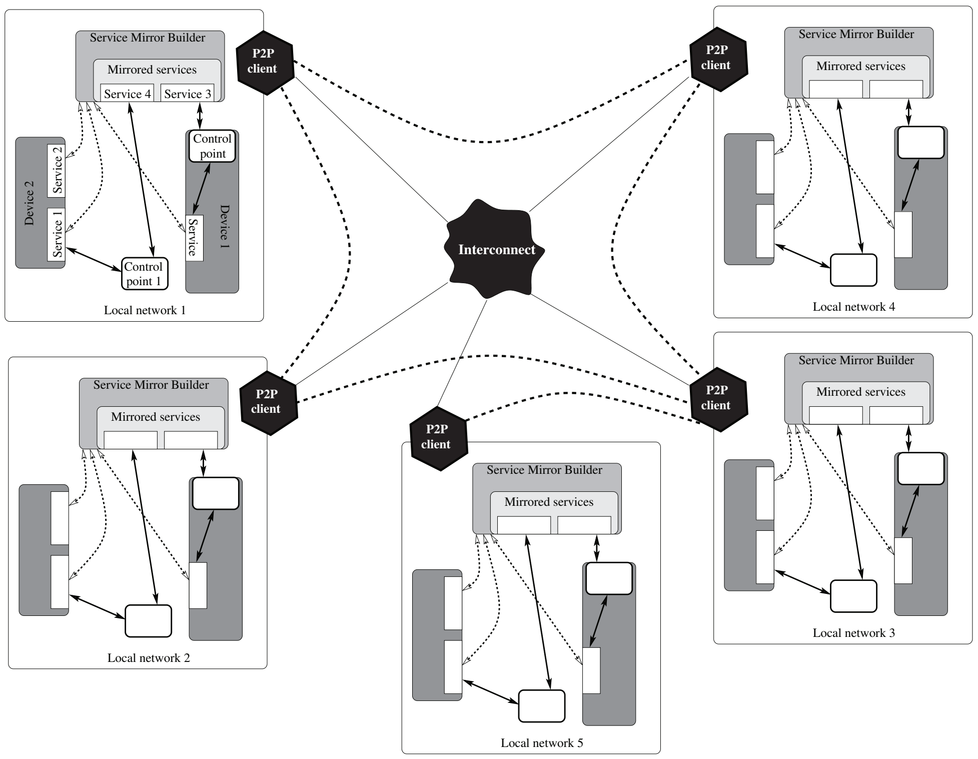

Fig. 2. A distributed architecture for remote service discovery (doted lines connecting local networks show the P2P network overlay; each local network has the structure shown in Figure 1)

3.3. P2P Protocol Example: Gnutella

Consider the well-known Gnutella P2P protocol [14] as a concrete example of P2P network usable in our architecture.

We note that the structure of a query message (which can be as large as 4kB) includes a minimum speed (bytes 0 and 1), a search criteria (byte 2) and a rest field (optional extension block). The rest field carries extra information using the Gnutella Generic Extension Protocol (GGEP), Hash/URN Gnutella Extensions (HUGE), and XML. The Gnutella Generic Extension Protocol (GGEP) in particular allows arbitrary extensions, which are particularly suitable for the encoding of remote service requests.

A Gnutella query-hit message includes the number of hits (byte 0), port and IP address (bytes 1–6), speed (bytes 7–10), and a result set; the result set (from byte 11 on) includes the file index (bytes 11–14), file size (bytes 15–18), a null terminated file name (from byte 19 on) immediately followed by a null-terminated extension block (that can contain GGEP, HUGE, and plain text metadata). Thus we recomment that the response messages from a service mirror builders be formatted using the GGEP extension and sent back to the network in the extensions field of the query-hit message.

4. Related Work

4.1. Remote Access to UPnP Devices Using the Atom Publishing Protocol

The network topology of one architecture for remote service discovery in UPnP [6] consists of at least two network segments: the home network and the remote network. These networks are connected to each other through the Internet. The architecture assumes that there is an IP tunnelling mechanism such as a Virtual Private Network (VPN) between the two network segments. The architecture introduces a new element called *UPnP Device Aggregator* which is acting as a proxy for the existing standard UPnP devices.

Enhanced UPnP Devices or *Control Points* are then UPnP devices or control points which are compatible with this remote service discovery architecture. The UPnP Device Aggregator aggregates information about the services and devices in the local network as an Atom feed, which can then be retrieved (using GET commands) by the enhanced UPnP control points in the remote network. Additionally, a UPnP Device Aggregator can receive information from remote Enhanced UPnP Devices and present them to the local control points. This information can be received by the UPnP Device Aggregator via HTTP POST.

The main shortcoming of this architecture is the need for VPN. Indeed, VPN does not scale well, requiring careful administration of IP addresses and subnetworks [8]. VPN also limits the architecture to the domains within the VPN network (limiting heterogeneity). No such limiting factors are present in our architecture, which is substantially more scalable. In addition, all remote service discovery requests are addressed to the home network, so this architecture can be considered centralized or partially centralized: there are some service coordinators (the UPnP Device Aggregators) to register and cache services [7]. By contrast, our architecture is fully distributed: no centralized coordinator is necessary. We note that many P2P network have switched to a hybrid architecture (such as Gnutella Ultrapeers [15]) for efficiency purposes, but even in this case we obtain a more distributed architecture.

4.2. Presence-Based Remote Service Discovery and Control for Ubiquitous Environments

An architecture for remote service discovery and control based on presence service (as used in instant messaging and VOIP) was also proposed [8]. A presentity can be anything that can have a presence state (be present or absent); presence information is sent to a presence service, which is a network service that records and distributes presence information. In the remote service discovery architecture based on presence service [8] there are two new functions called *service discovery gateway* and *service virtualizer*. Each service is seen as a presentity. The service discovery gateways register local services as presentities in a presence server. They can also retrieve other presentities from the presence server and present them to the service virtualizer. The service virtualizer uses this presence information to virtualize a local service in the local network. That is, a service virtualizer presents a remote service as a local one.

This architecture is partially centralized, as remote service providers and remote service requesters must first find a presence server to register or request a service. Although presence servers (as service coordinators) provide service visibility, the benefit does not come without cost and complexity [7, 16]. By contrast, our architecture is fully distributed.

4.3. Content Sharing and Transparent UPnP Interaction Between UPnP Gateways

Dynamic Overlay Topology Optimizing Content Search (DOTOCS) [17] enables flexible content searches among UPnP gateways. DOTOCS aims to establish an optimized peer-to-peer overlay network among UPnP gateways. DOTOCS uses a communication protocol between UPnP local networks described elsewhere (transparent interaction solution [18]): The communication between two connected UPnP local networks across the Internet is accomplished using the Web service technology. A local gateway encapsulates Simple Service Discovery Protocol (SSDP) messages into Simple Object Access Protocol (SOAP) messages and transmit them to another gateway over the global network. A Web service at the destination UPnP gateway extracts the SSDP message and replaces the original IP address (which is not valid in this local network) with the IP address of the gateway itself. The gateway then multicasts this discovery search message in the local UPnP network. If any device responds to that message (meaning that the device has the service demanded by the SSDP message), then the gateway encapsulates that message into another SOAP message and sends it back to the first network. This way one local UPnP network can discover remote services from a different UPnP network.

Scalability between local networks is manageable when this solution is used. However, each gateway multicasts in its local UPnP network any received discovery message (regardless whether the demanded service in that discovery message is locally available or not). This creates substantial traffic in the local network, most of it useless, which reduces scalability. Our protocol does not multicast remote requests to the local network (for indeed the service mirror builder has already discovered the locally available services), so the local UPnP network will not be loaded with spurious messages. Scalability therefore only depends on the P2P network used (and most of them are scalable to a high degree).

5. Conclusions

Service discovery plays an important role in pervasive computing. At the same time pervasive computing creates many challenges for service discovery protocols, which now need to work properly in a heterogeneous and dynamic environment. One other, related challenge is remote service discovery. We introduced a new approach to remote service discovery that is decentralized and fully distributed; we therefore believe that our approach offers better compatibility with pervasive computing environments. Additionally, our architecture is independent on the underlying protocols used for local service discovery and P2P communication. We have included a comprehensive set of featured for the local service discovery protocol in our discussion, but this is done without loss of generality: those protocols that lack some of the features can still function in our framework, which is agnostic with respect to which of the described services are actually used. The P2P architecture is also immaterial to the discussion, leaving the implementer the liberty to choose at will in this respect.

The core of our architecture is the new function in a local network called service mirror builder and its cooperation with a specialized P2P client software to discover remote services and then present them as local ones. Conversely, a service mirror builder can also control local services to serve them as remote services for other, remote service mirror builders. From the point of view of the local network the service mirror builder can control other services (whenever the respective service is offered outside the local network) but can also offer services (the mirrors of the remote services); overall, it is just a normal service discovery-enabled device.

The very definition of pervasive computing is distributed and mobile computing. In this paper we perform remote service discovery using a P2P network. The very design of such a network as a decentralized and distributed protocol moves this remote service discovery architecture one step ahead toward truly distributed computing.

References

[1] M. Weiser, Some computer science issues in ubiquitous computing, Communications of the ACM 36 (7) (1993) 75–84.

[2] Understanding Universal Plug and Play, white paper: www.upnp.org/download/UPNP_understandingUPNP.doc.

[3] Bluetooth Special Interest Group (SIG), Specification of the Bluetooth System Version 1.1, www.tscm.com/BluetoothSpec.pdf (2001).

[4] Sun Microsystems Inc., Jini Technology Core Platform Specification Version 1.2, www-csag.ucsd.edu/teaching/cse291s03/Readings/core1_2.pdf (2001).

[5] F. Zhu, M. Mutka, L. Ni, Service discovery in pervasive computing environments, Pervasive Computing 4 (4) (2005) 81–90.

[6] P. Belimpasakis, V. Stirbu, Remote access to universal plug and play (UPnP) devices utilizing the Atom publishing protocol, in: International Conference on Networking and Services, IEEE Computer Society, 2007, p. 59.

[7] W. Feng, Remote service provision for connected homes, Ph.D. thesis, De Montfort University (2010).

[8] A. Häber, Remote service discovery and control for ubiquitous service environments in next-generation networks, Ph.D. thesis, University of Agder (2010).

[9] E. Buyukkaya, M. Abdallah, R. Cavagna, VoroGame: A hybrid P2P architecture for massively multiplayer games, in: 6th IEEE Consumer Communications and Networking Conference (CCNC), IEEE, 2009, pp. 1–5.

[10] R. Schollmeier, A definition of peer-to-peer networking for the classification of peer-to-peer architectures and applications, in: 1st International Conference on Peer-to-Peer Computing, 2001, pp. 101 –102.

[11] B. Cohen, The BitTorrent Protocol Specification, www.bittorrent.org/ beps/bep_0003. html (2008).

[12] Clip2 Distributed Search Services, The Gnutella Protocol Specification Version 0.4, www.stanford.edu/class/cs244b/gnutella_protocol_0.4.pdf.

[13] D. Ilie, Gnutella Network Traffic-Measurements and Characteristics, Master's thesis, Blekinge Tekniska Högskola (2006).

[14] T. Klingberg, R. Manfredi, Gnutella 0.6, Network Working Group (2002).

[15] A. Oram, Peer-to-Peer: Harnessing the Benefits of a Disruptive Technology, O'Reilly Media, 2001.

[16] P. Engelstad, Y. Zheng, J. Tore, Service discovery and name resolution architectures for on-demand MANETs, in: 23rd International Conference on Distributed Computing Systems, IEEE Computer Society, 2003, pp. 736–742.

[17] E. Kawamoto, K. Kadowaki, T. Koita, K. Sato, Content sharing among UPnP gateways on unstructured P2P network using dynamic overlay topology optimization, in: 6th IEEE Consumer Communications and Networking Conference (CCNC), IEEE, 2009, pp. 1–5.

[18] M. Ogawa, H. Hayakawa, T. Koita, K. Sato, Transparent UPnP interactions over global network, in: Proceedings of SPIE, Vol. 6794, 2007, p. 67944P.