JYVÄSKYLÄ STUDIES IN COMPUTING
151

Niko Kotilainen

Methods and Applications for Peer-to-Peer Networking

JYVÄSKYLÄN YLIOPISTO

JYVÄSKYLÄ STUDIES IN COMPUTING 151

Niko Kotilainen

Methods and Applications for Peer-to-Peer Networking

Esitetään Jyväskylän yliopiston informaatioteknologian tiedekunnan suostumuksella
julkisesti tarkastettavaksi yliopiston Agora-rakennuksen salissa AgAud 2
joulukuun 21. päivänä 2011 kello 12.

Academic dissertation to be publicly discussed, by permission of
the Faculty of Information Technology of the University of Jyväskylä,
in building Agora, auditorium AgAud 2, on December 21, 2011 at 12 o'clock noon.

UNIVERSITY OF JYVÄSKYLÄ

JYVÄSKYLÄ 2011

Methods and Applications for Peer-to-Peer Networking

JYVÄSKYLÄ STUDIES IN COMPUTING 151

Niko Kotilainen

Methods and Applications for Peer-to-Peer Networking

UNIVERSITY OF JYVÄSKYLÄ

JYVÄSKYLÄ 2011

Editors
Timo Männikkö
Department of Mathematical Information Technology, University of Jyväskylä
Pekka Olsbo, Ville Korkiakangas
Publishing Unit, University Library of Jyväskylä

URN:ISBN:978-951-39-4604-3
ISBN 978-951-39-4604-3 (PDF)

ISBN 978-951-39-4603-6 (nid.)
ISSN 1456-5390

Jyväskylä University Printing House, Jyväskylä 2011

ABSTRACT

Kotilainen, Niko
Methods and Applications for Peer-to-Peer Networking
Jyväskylä: University of Jyväskylä, 2011, 46 p.(+included articles)
(Jyväskylä Studies in Computing
ISSN 1456-5390; 151)
ISBN 978-951-39-4603-6 (nid.)
ISBN 978-951-39-4604-3 (PDF)
Finnish summary
Diss.

Due to the exponential growth of the internet during the last 15 years, peer-to-peer networks have been utilized in several new application categories ranging from internet telephony networks to currency systems. The peer-to-peer architecture has also garnered a great amount of research interest, contributing to the rapid advancement of the field. This dissertation explores peer-to-peer networks from a number of angles and presents a wide array of research results. The results include resource discovery algorithms for peer-to-peer networks, a novel routing algorithm for mobile encounter networks, an indoors location-sensing system, middleware prototypes, application prototypes and ideas, and tools for peer-to-peer networks research. The results provide various stepping stones on the way towards new kinds of communication applications and methods, utilizing devices that can interact and communicate more efficiently and dynamically both in local and global environments.

Keywords: Peer-to-peer, Mobile peer-to-peer, Social networks, Location sensing, Mobility-assisted routing, Resource discovery, Genetic algorithms

Author	Niko Kotilainen Department of Mathematical Information Technology University of Jyväskylä Finland
Supervisors	Dr. Jani Kurhinen Department of Mathematical Information Technology University of Jyväskylä Finland
	Prof. Tapani Ristaniemi Department of Mathematical Information Technology University of Jyväskylä Finland
Reviewers	Prof. George C. Polyzos Mobile Multimedia Laboratory Department of Informatics/Computer Science Athens University of Economics and Business Greece
	Dr. Ernesto Tarantino Institute of High Performance Computing and Networking Napoli, Italy
Opponent	Adj. Prof. Sergey Balandin Department of Communications Engineering Tampere University of Technology Finland

ACKNOWLEDGEMENTS

I would like to express my greatest gratitude to everyone who has contributed in making this dissertation a reality. Out of the many people who have left their mark in this work in a way or another, Mikko Vapa and Jani Kurhinen deserve special thanks. Without their support and guidance the dissertation you are currently reading simply would not exist.

Additional thanks go to all the co-authors of articles included in this work, especially Maria Papadopouli, Annemari Auvinen, Ferrante Neri and Matthieu Weber. I'd also like to thank professor Tapani Ristaniemi for his role in supervising the work and to professor Pekka Neittaanmäki for taking the time from his busy schedule to help speed up the writing process.

ACRONYMS

BFS	Breadth First Search
C/S	Client/Server
CPU	Central Processing Unit
DFS	Depth First Search
DTN	Delay Tolerant Network
FM	Frequency Modulation
GSM	Global System for Mobile Communications
GPS	Global Positioning System
IP	Internet Protocol
JVM	Java Virtual Machine
MANET	Mobile Ad-hoc Network
MEN	Mobile Encounter Network
MP2P	Mobile Peer-to-Peer
P2P	Peer-to-Peer
RMI	Remote Method Invocation
TCP	Transmission Control Protocol
TTL	Time To Live
UDP	User Datagram Protocol
WiFi	Trademark for one popular WLAN technology
WLAN	Wireless Local Area Network
XML	Extensible Markup Language

LIST OF FIGURES

LIST OF TABLES

CONTENTS

LIST OF INCLUDED ARTICLES

PI Niko Kotilainen, Lito Kriara, Konstantinos Vandikas, Konstantinos Mastorakis and Maria Papadopouli. Location-Based Media Sharing in a MP2P Network. In *ACM SIGMOBILE Mobile Computing and Communications Review*, volume 12, issue 1, pages 62-64, 2008.

PII Pedro Tiago, Niko Kotilainen, Heikki Kokkinen, Jukka Nurminen and Mikko Vapa. Mobile Search – Social Network Search Using Mobile Devices. In *Proceedings of the 5th IEEE Consumer Communications and Networking Conference*, pages 1201-1205, 2008.

PIII Andrei Papliatseyeu, Niko Kotilainen, Oscar Mayora and Venet Osmani. FINDR: Low Cost Indoor Positioning Using FM Radio. In *Mobile Wireless Middleware, Operating Systems, and Applications*, volume 7 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pages 15-26, 2009.

PIV Niko Kotilainen and Maria Papadopouli. You've Got Photos! The design and evaluation of a location-based media-sharing application. In *Proceedings of the 4th International Mobile Multimedia Communications Conference*, 2008.

PV Niko Kotilainen and Jani Kurhinen. A Genetic-Neural Approach to Mobility-Assisted Routing in a Mobile Encounter Network. In *Proceedings of the 5th International Conference on Information Technology and Applications*, 2008.

PVI Mikko Vapa, Niko Kotilainen, Annemari Auvinen, Heikki Kainulainen and Jarkko Vuori. Resource Discovery in P2P Networks Using Evolutionary Neural Networks. In *Proceedings of the 2004 International Conference on Advances in Intelligent Systems - Theory and Applications*, 2004.

PVII Ferrante Neri, Niko Kotilainen and Mikko Vapa. An Adaptive Global-Local Memetic Algorithm to Discover Resources in P2P Networks. In *Applications of Evolutionary Computing*, volume 4448 of *Lectures Notes in Computer Science*, pages 61-70, 2007.

PVIII Ferrante Neri, Niko Kotilainen and Mikko Vapa. A Memetic-Neural Approach to Discover Resources in P2P Networks. In *Recent Advances in Evolutionary Computation for Combinatorial Optimization*, volume 153 of *Studies in Computational Intelligence*, pages 113-129, 2008.

PIX Niko Kotilainen, Matthieu Weber, Mikko Vapa and Jarkko Vuori. Mobile Chedar – A Peer-to-Peer Middleware for Mobile Devices. In *Proceedings of the Third IEEE International Conference on Pervasive Computing and Communications Workshops*, pages 86-90, 2005.

PX Niko Kotilainen, Mikko Vapa, Matthieu Weber, Joni Töyrylä and Jarkko Vuori. P2PDisCo – Java Distributed Computing for Workstations Using Chedar Peer-to-Peer Middleware. In *Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium*, 2005.

PXI Niko Kotilainen, Mikko Vapa, Teemu Keltanen, Annemari Auvinen and Jarkko Vuori. P2PRealm – Peer-to-Peer Network Simulator. In *Proceedings of the 11th International Workshop on Computer-Aided Modeling, Analysis and Design of Communication Links and Networks*, 2006.

PXII Niko Kotilainen, Mikko Vapa, Annemari Auvinen, Matthieu Weber and Jarkko Vuori. P2PStudio – Monitoring, Controlling and Visualization Tool for Peer-to-Peer Networks Research. In *Proceedings of the ACM international workshop on Performance monitoring, measurement, and evaluation of heterogeneous wireless and wired networks*, pages 9-12, 2006.

1 INTRODUCTION

The majority of current network services are based on the client/server (C/S) architecture, where a dedicated server computer processes the requests of numerous clients. The C/S architecture brings with it some benefits, for example a simple, clear structure and ease of management. The architecture does have its downsides too, and it is not the most suitable architecture for all network services.

Another architecture that has lately garnered a lot of attention from both the academic researchers and the industry, is the peer-to-peer (P2P) architecture. In peer-to-peer systems, all the nodes, also called peers, can act both as servers and as clients, both providing and consuming resources in the system. An example of a peer-to-peer system is the Skype telephony platform where calls are being routed through other nodes running the Skype software, instead of using servers operated by the company operating the network. The use of the P2P architecture leads to clear benefits for the network, which include robustness, high scalability, high availability and low initial investment costs. The P2P architecture has its problems too, as using the P2P architecture instead of the C/S architecture makes systems more complex to design and operate. Locating resources in a P2P network is especially a difficult problem due to the decentralized nature of the network.

Lately, social networking applications where users can communicate with their friends, have become hugely popular all around the world. This brings a new topic for P2P research, as the actual social relationships of people can also be used to generate the topology of the P2P network. For certain social networking applications, this would greatly boost the efficiency of locating resources from the network, and would solve the problem of joining the P2P network and the problem of being able to trust other peers in the network.

1.1 Problem Formulation

This dissertation aims to answer the following questions:

- How to discover the required amount of resources from a peer-to-peer network with a minimal usage of network capacity?
- How to route messages in a mobile encounter network while minimizing the time the messages spend en route, and minimizing the required device resources used?
- How mobile devices can sense their location indoors in a cost-effective way?
- Which kind of tools and platforms do real-life peer-to-peer applications require?

The results provide some steps on the way towards new kinds of services, utilizing devices that can interact and communicate more efficiently and dynamically both in local and global environments.

1.2 Author Contribution

The author has participated in the writing process of all the articles included in this dissertation. A short description of the articles and the author's contribution follows.

Article **PI** introduces a mobile P2P media sharing application, where short range wireless connections can be used to share location based content. The system utilizes the 7DS[42] middleware and the CLS[18] location sensing system. For this article, the author designed and implemented the prototype.

Article **PII** introduces a mobile P2P search platform for highly dynamic content within a social network. The system utilized 3G connections and web servers running on mobile devices to facilitate the search functionality. The author participated in the development and evaluation of the prototype.

Article **PIII** introduces and evaluates a location-sensing system based on measuring the signal strength received from low-power FM radio transmitters. The system was found to be as precise as WLAN-based systems, with much lower costs and smaller power consumption. The author designed and implemented the prototype and participated in running the measurements and designing the algorithms.

Article **PIV** evaluates the media sharing application first presented in Article **PI**. The author conducted the measurements and evaluated the results.

Article **PV** introduces a new mobility assisted routing algorithm for mobile encounter networks. The algorithm employs neural networks trained with genetic algorithms to make the routing decisions. The author devised the algorithm based on earlier research done on the resource discovery problem in wired P2P networks.

Article **PVI** introduces a resource discovery algorithm for P2P networks. The algorithm employs neural networks as the core of the resource discovery algorithm. The author designed and implemented the tools used, performed the simulations, and participated in designing the algorithm.

Article **PVII** enhances the algorithm presented in **PVI** by introducing a novel memetic algorithm to the training process. The algorithms are compared, and the new algorithm out performs the older one presented in **PVI**. The author implemented the algorithms, ran the measurements, and participated in algorithm design.

Because of **PVII**'s good reviews, the authors were invited to write an extended version of the paper as a book chapter. This chapter is included as Article **PVIII**, which evaluates further the algorithm presented in **PVII**. The author ran the measurements and participated in the evaluation related to the research.

Article **PIX** presents a mobile P2P middleware prototype. The mobile peers can also act as members in the non-mobile Chedar-network [5]. The author designed and implemented the prototype.

Article **PX** presents a P2P distributed computing platform prototype built on top of the Chedar [5] P2P middleware. The platform was tested and evaluated using the network simulator presented in Article **PXI**. The author designed and implemented the prototype, and did the evaluation.

Article **PXI** presents and evaluates a P2P network simulator prototype. The simulator was heavily optimized for simulating neural network training scenarios, and was used in the development of NeuroSearch presented in Article **PV**. The author designed and implemented the prototype, and participated in the evaluation process.

Article **PXII** presents a P2P network management tool for research use. The tool has been designed to be used with the Chedar [5] network, but can also easily be adapted for use with other networks. The author designed and implemented the prototype.

The dissertation is structured as follows. Chapter 2 presents networking technologies relevant to this dissertation. Chapter 3 explores the challenges inherent in peer-to-peer and mobile peer-to-peer networks. In Chapter 4, relevant research tools are presented. Chapter 5 discusses the contributions of the author, and finally, the dissertation is concluded in Chapter 6.

2 NETWORKING TECHNOLOGIES

2.1 Related network architectures

Several network architectures are currently used in data communication networks, the most widely used being the Client/Server [41] architecture (C/S). In the C/S architecture the network nodes have clearly described roles as either a client or a server. The World Wide Web and the underlying HTTP protocol are typical examples of technologies employing the C/S architecture, in which the browser application acts as a client and connects to web servers to fetch web pages requested by the user. Clients are always the active parties in establishing a connection to the server to request or send information, the servers just wait for connections and requests. Designing C/S systems is relatively simple and designs become clean with clearly determined roles. Administering C/S systems is also quite easy, as information stored in the system is located centrally on the server. C/S designs do have some drawbacks too because of the server-centricity of the design. The server becomes a single point of failure in the system, and the costs involved in building a robust C/S system can easily offset the benefits received from the simpler design and easier maintenance. According to measurements presented by Kondo [28], the average desktop computer CPU utilization is less than 15 percent, proving that client resources in C/S systems are commonly underutilized. Especially when building large scale systems, where a single server is not enough to handle the load, it would be beneficial to consider other network architectures too.

According to Hayes [21], the locus of computation is currently shifting from personal computers and workstations to services provided over the internet from multiple distant data centers distributing the computation load between each other. Cloud computing was coined as a term for these services, and has drawn considerable amounts of interest from the industry and academia alike. Several systems using the cloud architecture are already on the market, for example Google App Engine and Amazon Web Services. In cloud systems, the service providers own large amounts of servers, commonly distributed between different data centers. Commoditized resources on these servers are then sold to third parties, who might not even know where and how many servers are being used to run their processes. The aim of cloud computing providers is to make computing a utility, comparable with electricity, where customers can purchase just the required amount for their processes, and transparently scale their computing resources up and down with demand [17].

The cloud computing architecture is still very centralized, with the cloud servers owned and administered by a single entity. When combining cloud computing technologies with client/server architecture, it is possible to fix some of the problems of the C/S architecture; cloud systems generally do not suffer from a single bottleneck or a single point of failure in the system for example. Cloud architecture still suffers from several of the drawbacks of C/S networks, such as having high initial investment costs and being susceptible to monitoring and interference either from governmental, commercial or other entities [55].

2.2 Peer-to-Peer Networks

Peer-to-peer (P2P) systems differ from the traditional C/S and cloud computing systems in that all the nodes can act both as servers and as clients. In [47], Schollmeier defines P2P networks as distributed systems, where the participants share a part of their resources with other peers in the network, and other participants access these resources directly, without passing intermediary entities. These resources can be for example processing power, storage capacity or network bandwidth. Figure 1 presents the logical topologies of peer-to-peer and client/server architectures.

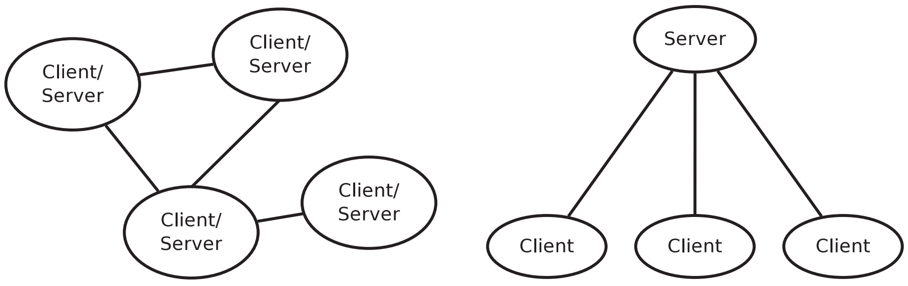


FIGURE 1 Peer-to-peer and client/server architectures

2.2.1 History of peer-to-peer networks

From the late 1960's to the 1970's the Defense Advanced Research Projects Agency of the United States Department of Defence, or DARPA, established a research project to develop a new kind of computer network. The design goals of the system were to develop a military network capable of functioning even if a large number of the nodes or communication links of the network were lost either due to an attack on the infrastructure or for other reasons.

The resulting network was named ARPANET, which was the basis for the current internet, and to some extent it still defines the nature of the internet. In the 90’s the rapid growth of the internet and the arrival of the World Wide Web moved the internet towards a client/server architecture, where the network has a small number of dedicated servers and a large number of clients using services from the servers. These clients become second-class citizens of the internet, as they were usually connecting to the internet through slow and firewalled modem connections, and only being assigned a temporary, dynamic IP address. This created a very clear distinction between the servers and the clients. This trend has reverted a little since the turn of the millennium because of peer-to-peer networks and faster internet connections with static IP addresses emerging. The first P2P networks at the turn of the millennium were used mostly for sharing copyright protected files, like music, but since then a lot of legitimate uses for P2P networks have also emerged, for example the Skype Voice-over-IP telephony application and the Bitcoin P2P currency system. Even the DARPA is again using P2P networks on the battlefield, thus closing the circle [4].

2.2.2 Hybrid Peer-to-Peer Networks

P2P networks come in two main flavors, hybrid and pure P2P networks [47]. Hybrid P2P architecture, as the name implies, is a hybrid of both client/server and pure peer-to-peer models, trying to combine the best parts of both models. In hybrid P2P networks the network is managed by a server which holds a database of resources held by the network nodes. The network nodes connect to the server and report their resources to the database. The nodes query the server for resources, and the server gives a reply containing information about nodes holding the queried resource. After receiving the reply, a node can then establish direct connection to the node(s) holding the resource and request the resource.

The best known hybrid P2P network was the original Napster network, but as the majority of files shared over the network were copyrighted material, and Napster operated without permission from the copyright holders, Napster was quickly sued for facilitating copyright infringement and later the network was shut down by US authorities. This case made the drawbacks of the hybrid model very clear, as it was very easy for the authorities to shut down the network by unplugging the servers organizing the search. It could be argued that in the Napster case there was a moral justification for shutting down the network, but in several cases, for example military networks or networks used by dissidents in countries with no free speech rights, there might be a third party trying to actively shut down the network without such justification. This has been the case in China, and lately in Iran, where the authorities have been actively trying to shut down P2P communication networks used by political pro-democracy activists.

2.2.3 Pure Peer-to-Peer Networks

Pure peer-to-peer networks on the other hand drop the server altogether and run completely on nodes with equal rights and responsibilities. The nodes of the network do not have predetermined roles such as servers or clients, but are in an equal position to other nodes in the network and can take different roles based on the requirements of the network. The nodes of the network, called peers, are usually connected to a few other peers, most commonly using TCP connections. The connections form an overlay network topology on top of the physical network connecting the nodes [40]. Nodes of the network can forward messages between other nodes, and can make resources available to other nodes of the network. The resources can be for example files, computing capacity, bandwidth, storage space, location data, etc. If a node is looking to use a resource from the network, it can act as a client and send a request to its neighbor nodes, which can then again act as routers and forward the request further, or act as servers and send a reply to the requesting client.

Oram [40] lists several benefits of pure P2P networks, some of which hold true also for hybrid networks. The most prominent one being resiliency to attacks and network failures. Due to the distributed nature of the network, there is no single point of failure, and the tasks of failed nodes can be delegated to other nodes of the network. P2P networks are also inherently scalable. As more peers join the network, the new peers provide more computing capacity and bandwidth to the network without the need to install more servers. This also lowers the hardware costs associated with setting up the service, as no expensive servers or datacenter space are needed.

As opposed to traditional client/server network architectures, P2P networks do not have a single point of authority, and while this makes P2P networks robust, it is also the source of many problems in using these networks. As no party in the network has a global view of the network topology, or of the location of resources in the network, discovering resources and routing messages in the network becomes problematic. Commonly in pure peer-to-peer networks the nodes only have knowledge of their neighbor nodes, i.e. the nodes that they have established connections to. To find a resource the node has to send a query to its neighbors, which then forward the query to their neighbors according to the resource discovery algorithm the node is utilizing.

As there is no central authority in the network, it is complicated for the network nodes to find out whether the information they are receiving from other nodes of the network is trustworthy. If trust issues are not taken care of, rogue nodes in the network can intentionally reply to resource requests with corrupted data or otherwise send misleading and erroneous control messages hampering the functionality of the network. Rogue nodes in the network can be battled with several techniques, for example a *web of trust* design, where a user designates his friends to be trusted, who then again designate their friends as being trusted. This, combined with the removal of ousted rogue nodes and the nodes on the path of trust to the rogue node, makes the network very difficult for rogue nodes to infiltrate. This technique does require the network users to actually know each other, which usually is not the case in P2P networks. To eliminate this requirement, a large number of architectures which automatically rate the nodes for their trustability have been suggested [35].

Joining pure P2P networks also presents challenges. When a node is outside the network, it has no knowledge of other nodes of the network, and thus an external source of node names is required. Research for solving the joining problem has been carried out in our research group [56].

Significant research effort has been invested into solving these problems, and several routing and resource discovery algorithms have been suggested in literature. These algorithms are discussed in more detail in Chapter 3.

2.3 Mobile Ad-Hoc Networks

The majority of current wireless networks use static, stationary access points, where mobile clients connect to gain access to the rest of the network. Examples of these infrastructure-based networks include GSM and infrastructure-mode Wi-Fi. In a way these networks can be classified as client/server (C/S) networks, where mobile clients connect to stationary servers for connectivity to the rest of the network. The infrastructure-based networks share several advantages with client/server networks, such as ease of design and operation, and simple and reliable authentication and authorization methods. Just like wired C/S networks, infrastructure based wireless networks have also some drawbacks:

- The infrastructure might not be available everywhere, or the access point could not be contacted due to some objects attenuating the signal on the way. If the devices cannot connect to the access points, no information can be dissipated in the network.
- The investment cost for the infrastructure is usually very high.
- The network has low scalability. If a large amount of devices attempt to use the services of a single access point, the access point can get overloaded and fail to serve the clients.
- The networks also have low robustness, as the access points are a single point of failure.

Some of these drawbacks can be solved with proper network design, for example selecting the locations of the access points so that the mobile client device can reach a minimum of two access points in any location. This though requires even higher initial infrastructure costs, as the access points have to be placed more densely than would be otherwise required.

These drawbacks make them unsuitable for some applications. As an example, if one is to go hiking in the wilderness, one cannot expect GSM phones to be able to connect to base stations, and thus no calls can be made even to nearby fellow hikers. Now, if GSM phones could directly connect to each other to transmit the call, the phones could effectively bypass the whole infrastructure-part of the network, and if the phones cannot reach each other directly, the call can be routed through a third party, possibly another hiker situated between the call parties. Taking the example one step further, the phones could form a network where calls are routed through several devices in range of each other. This would make calls possible between callers who are far away from each other. In this example, the phones would have created a mobile ad-hoc network (MANET).

Mobile ad-hoc networks are increasingly being used as an architecture for wireless networks in scenarios like the one presented above. MANETs are self configuring infrastructure-less networks, where the nodes connect to each other using a wireless connection [2]. The connection technologies can be for example Wireless Local Area Networks (WLAN) such as WiFi, Wireless Personal Area Networks (WPAN) such as Bluetooth, or even low-bandwidth and ultra-low power technologies designed specifically for sensor network applications, such as Zigbee, IEEE 802.15.4 or Bluetooth Low Energy [32].

Benefits of MANETs can be clearly seen with the example presented before, where the network is able to function anywhere where the devices are able to connect to each other, and as the amount of devices increases, the bandwidth and the processing capacity of the network increases linearly all the way until the capacity of the radio spectrum is reached. Also the cost of building the network is low, as no infrastructure installations are required.

Although forming the network is very easy for the MANET nodes - the nodes just connect to whomever they can reach over the radio interface, certain nodes of the network can only connect to each other if they are within radio range of each other. Routing information between far-away nodes becomes time- and resource-consuming, as messages need to be routed through several nodes between them. Also the usable routes are constantly changing, as the nodes of the networks can be moving, and thus connections are lost and created within the network all the time. Several protocols and algorithms have been suggested for routing messages in MANETs, which fall into two categories: proactive and reactive routing. Proactive algorithms aim to maintain a routing table for the network by keeping track of the changes in network topology, even if no messages are being sent, thus ensuring quick message delivery. Reactive algorithms on the other hand only find a route to a destination when a message is available for sending, this operating way minimizes routing overhead and the resources used for control traffic [37].

2.4 Mobile Encounter Networks

Delay tolerant networks (DTNs) are a subclass of mobile ad-hoc networks, lacking continuous connectivity. In DTNs the network nodes are distributed so sparsely, or their wireless range is so small, that the network commonly becomes partitioned into unconnected sub-networks. Mobile encounter networks take this even further, generating a network, where the nodes are rarely in connection with other nodes for longer periods at once. As the nodes are mobile, they frequently come within range of other nodes in the network, and when this happens they initiate a connection and exchange data between the nodes, thus diffusing information in the network [14, 24].

Information diffusion and routing in mobile encounter networks is very similar to the spread of epidemic diseases, where infected individuals coming into contact with others infect the other people too. The spread of disease has been widely researched, and this research has been used as a basis for research into information diffusion in all kinds of delay tolerant networks [12, 6]. Routing in mobile encounter networks is further discussed in the section 3.2.

Due to the high delays in message delivery, mobile encounter networks are not suitable for all networking applications, but there are a lot of applications where the delays in diffusing the data do not hamper the functionality of the application. For example web caching, commodity (i.e. gasoline or groceries) price tracking, message delivery, and sensor network data delivery.

Due to the relatively young age of the research field, there are several terms in use with relatively small differences. The terms *pocket switched network* [23] and *human contact network* [46] are very similar with the term *mobile encounter network*. The former two just place more emphasis on the human mobility and human contact aspects of the network, whereas mobile encounter networks can be for example created by swarms of mobile autonomous unmanned vehicles. Also instead of a *delay tolerant network*, several researchers have recently used the term *disruption tolerant network*.

2.5 Mobile Peer-to-Peer Networks

Mobile peer-to-peer (MP2P) networks function on the application layer of the OSI-model, and thus are independent of the underlying network architecture [20]. The networks are commonly classified into two categories, infrastructure-based and infrastructure-less networks, according to the features of their underlying network infrastructure. Infrastructure-based networks utilize the internet or other infrastructure-based networks, and infrastructure-less networks utilize wireless ad-hoc networks. This dissertation will mainly discuss infrastructure-less networks.

2.6 Social Networking Applications in Mobile P2P Networks

In his book [29], Kopomaa discusses the effects of a mobile way of life on the urban environment. Kopomaa describes mobile devices as nomadic objects, which enable their users to move freely while carrying on with their activities, making whatever space they are in a temporary home or office replacement. These new forms of communication have a profound impact on our everyday lives, as spontaneous and real-time communication with no restrictions on place and time enables an all new social order, with all new social practices.

As the computing power, bandwidth, and storage space available to mobile devices are advancing at great speed, new kinds of applications are made possible. Current mobile devices already provide a very capable platform for social communication applications, and combining the ideas of MP2P and social networking make it possible to develop powerful communication applications. Social applications are a good fit for P2P networks, because the overlay network can be generated from the social connections of the participating people. As people would already know their neighboring nodes in the P2P network, there would be no problem in finding nodes to connect to, and also the problems caused by the lack of trust between nodes would be greatly reduced. For certain applications, for example search in social networks (see Article **PII**) this greatly boosts the efficiency of locating resources from the network [52, 51].

As proposed by Allen [3], the social aspects of P2P networks can also be used to benefit the underlying processes of the P2P network. In his proposal the network provides a higher reputation, and a higher payout to the cooperating peers, thus incentivizing unselfish behavior.

3 CHALLENGES IN PEER-TO-PEER NETWORKING

Peer-to-peer and mobile peer-to-peer architectures introduce new challenges that do not exist in the traditional client/server or infrastructure-based architectures. This chapter introduces the challenges relevant to this dissertation.

3.1 Resource Discovery

Locating resources in a pure peer-to-peer network is problematic, as there is no central authority that would keep a database of all the resources available in the network. As there is no central server, the peers have to query their neighbors, who in turn forward the query to their neighbors and so on. As the network can have millions of peers, it is not feasible bandwidth-wise to forward the query to all the peers in the network. Thus, the peers need a resource discovery algorithm to make decisions on where to forward queries and where not to. Several algorithms have been proposed to solve the resource discovery problem in different settings, the most prominent of which are described below.

- **Breadth-first-search (BFS)** [34], where the query is simply sent to all the neighbors of the node, who then again send it to all their neighbors and so on. The algorithm is technically very simple, and it only requires one configuration parameter, Time to Live (TTL). The TTL is required to keep the algorithm from flooding the whole network, as it limits the number of times a message will be forwarded. Lv et al. [33] find that BFS is not efficient nor scalable, and in particular on Gnutella and power-law graphs the effects of flooding are disastrous: the number of messages increases drastically when TTL is increased.

- **Highest Degree Search (HDS)**, which is proposed by Adamic et al. [1] and Kim et al. [26], is a search strategy that utilizes the topological properties of a power-law network. The search strategy first proceeds towards the highest-degree node, i.e. the node that has the highest number of neighbors, and then gradually moves to nodes of a lesser degree. The algorithm locates resources efficiently if they can be found from the core of the network, but the performance decreases when the central nodes are revisited in search for lower degree nodes. The HDS algorithm also causes congestion in the central nodes of the network, as all of the queries from the whole network are forwarded towards the central nodes.

- **Random walk**, which resembles HDS in that the query is only forwarded to one neighbor at a time. The random walk algorithm differs from the HDS algorithm in that the neighbor to receive the query is selected at random, thus lowering the chances of visiting the same nodes multiple times. Just like with HDS, while random walkers increase the number of hops and thus latency, they decrease the total traffic because the search proceeds in a depth-first manner [33].

There are certain limitations in all approaches described above. First, each of these algorithms uses some control parameters (for example time-to-live, the number of walkers or the proportion of neighbors to forward the query) that can be used to tune the algorithm. For a search algorithm, the number of control parameters should be kept at a minimum to allow zero configurability when applied to a real environment. Second, while some of these approaches have mechanisms to adapt to the environment, they do not utilize the entire potential of the environment because they rely only on one strategy. In general, one strategy alone cannot be efficient in all scenarios, and therefore an efficient algorithm should be able to utilize many strategies depending on the current scenario.

Figure 2 presents a query in a P2P network using the Breadth First Search (BFS) algorithm. In the beginning Peer 1, which is querying for resource R, sends the query to its neighbors, Peer 2 and Peer 3. These two send the query to all their neighbors except Peer 1, where the query was received from. When peers 2 and 3 receive the query from each other, they disregard the received message because they have processed the query before. When Peer 4 receives the query, it notices that it holds the queried resource and sends a reply to the query. Peer 2 forwards the reply to Peer 1 and thus Peer 1 gains the knowledge of where to find the resource and can then use the resource. As it is evident, the algorithm efficiency is not optimal, as the algorithm sends more messages than those which would be required to find the resource.

3.2 Routing in Mobile Encounter Networks

To bring multi-hop data transmission into mobile encounter networks, the mobility of the nodes has to be used to deliver messages between nodes that do not have a direct communication route between them. This poses a difficult problem, because there is no global knowledge of the network state, or of future locations and encounters of the peers. In fact there is no real-time knowledge of

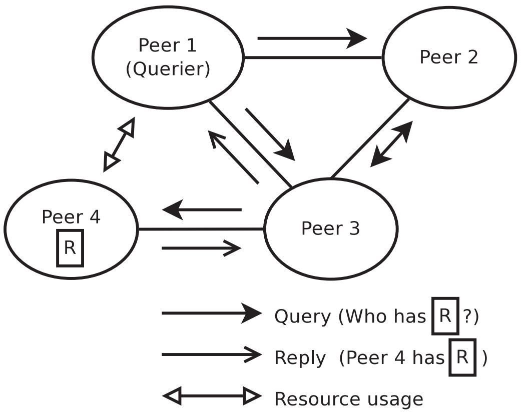


FIGURE 2 Query in a pure peer-to-peer network

what is happening outside the radio range of the device, as the devices of the network only exchange information during encounters with other devices. Further problems are caused by the difficulty in predicting the movements of the mobile devices, though a prediction of the mobility can be acquired by using statistical methods to infer the information from the current and historical location and heading information of the device.

For mobile devices it is usually very important to conserve power, thus the efficiency of the routing algorithms is essential in mobile encounter networks [8]. Some algorithms proposed for routing are as follows.

Epidemic routing was first introduced by Vahdat and Becker [10]. As the name implies, the algorithm emulates the spread of epidemic diseases to route information. Available messages are passed to all encountered nodes in the hope that one of the nodes is able to deliver it to a target location. It is a very powerful method and always gives the smallest delay possible if the network system handles the data flow properly. However, unlimited forwarding of messages makes the algorithm waste a lot of network resources.

Spray and wait [48] and later **Spray and focus** [49] were proposed by Spyropoulos et al. The algorithms are good examples of methods designed to limit the flooding of the network experienced when using the epidemic routing algorithm. Spray and Wait exploits different types of counters to control the number of message copies in the network. Spray and Focus has evolved from Spray and Wait, and combines copying and forwarding. These schemes, however, do not qualitatively distinguish distinct nodes while passing message copies. Instead, they employ numerous randomly selected nodes as message carriers. Even though they are more efficient than the pure epidemic diffusion, they still waste substantial amounts of device memory, battery power and network bandwidth while passing data to inappropriate network nodes.

During the last year, socially aware algorithms have been proposed for routing in mobile encounter networks by Bulut and Szymanski [7], Mei et al. [38] and Klein et al. [27]. Based on the observation that individuals with similar interests tend to meet more often, they hypothesize that inclusion of knowledge of the social structure improves the efficiency of routing, and the simulation results support the hypothesis.

3.3 Location Sensing

Location sensing capabilities are important for many mobile applications, especially those operating in a mobile P2P fashion [11]. In outdoor settings GPS usually provides good enough accuracy with modern receiver technologies, but indoor localization is still an area to be improved. There are plenty of approaches for implementing indoor location sensing, some of which are presented in the following list:

- **Proximity-based methods** sense nearby transmitters, and use their IDs to make an estimation of a position. This is the simplest of the categories, but also the least precise. Examples of proximity-based systems include recording pre-placed RFID tags in range and comparing this information to a known map of the RFID tags [25].
- **Signal strength -based methods** record the signal strengths received from transmitters with known locations and approximate their distance from the receiver using this data. For example CLS [18], which uses WiFi and/or Bluetooth signal strengths, and FINDR **PIII**,[45], which uses low range FM radio signals. Varshavsky [53] presents a system for using GSM signal strength measurements for indoor location sensing, and Stuntebeck et al use radio frequencies injected into the power lines running in the walls of buildings for location sensing [50].
- **Time-based methods** measure the time a signal travels from the transmitter to the receiver, and triangulate a location based on this measurement. Examples of time-based location sensing are GPS, and measuring the time of ultrasound pulses [54].

4 RESEARCH TOOLS

When doing the research discussed in this dissertation, several different research tools were evaluated. Due to the unique requirements of the research problems this dissertation covers, most of these tools did not have the required capabilities.

4.1 Mobile Peer-to-Peer middleware

Several middleware enabling MP2P functionality for applications running on top of the middleware have been proposed. The middleware offers an application programming interface for application development, each with their distinctive functionalities. This section presents the most prominent ones.

7DS (7 Degrees of Separation) [42, 43] is an architecture and a set of protocols enabling resource sharing among peers that are not necessarily connected to the internet. The 7DS prototype has been written in Java 2 Standard Edition. 7DS works only on IP networks and uses UDP multicast to query other peers, so the peers have to be on the same broadcast group.

LightPeers is a light-weight mobile peer-to-peer framework proposed by Christensen et al. [10]. It has been designed for mobile computers supporting ad hoc groups for learning, gaming, and playing by allowing peers in the field to produce, share, and present digital material in a session. The LightPeers framework utilizes UDP connections for communication between the nodes.

Proem [30] is a mobile middleware providing a solution for developing and deploying applications for mobile ad hoc networks. In Proem, middleware is responsible for presence and discovery services as well as being an identity, data space and community manager. Proem has been designed for mobile peers in ad hoc networks whereas in Mobile Chedar peers with fixed P2P network connections are also supported. The current prototype of Proem uses Wireless Local Area Network (WLAN) for communication and has been implemented using Java.

XMIDDLE [36] is a reflective middleware enabling transparent sharing of

XML documents between mobile peers, because the data structure consists of XML trees, modifications to the branches of XML tree are fine-grain for example compared to modification of files. XMIDDLE solves the problem of simultaneous edits by several peers by allowing the user to resolve the update conflicts. The current XMIDDLE prototype is based on WiFi and has been implemented using Java.

MOBY [22] is a service network enabling access to services on wide area networks. The framework is built using Jini and Jini Technology Surrogate Architecture Specification. In MOBY, resources are registered to Jini Lookup Service, which is located in the local area network. MOBY's P2P network is based on the super-peer architecture, i.e. the network is divided into domains by Mnode super-peers. Like 7DS and LightPeers, MOBY uses UDP for communication between Mnodes. Resource discovery in MOBY is done using the expanding ring search algorithm [33] between Mnodes. Overall, MOBY is designed more like a fixed overlay, because the links between Mnodes are preconfigured compared to the autonomous overlay approach used in the other middleware discussed above.

4.2 Distributed computing middleware

When choosing a distributed computing platform to distribute the P2PRealm simulator, there were several options for distributing Java-based applications available.

One class of software is formed by programming language independent distributed computing tools that support Java. An example of such software is Globus Toolkit [16], in which Java Commodity Grid kit provides an interface for accessing Globus services using Java programs. Globus contains mechanisms for code mobility which poses security risks, because the downloaded code needs to come from a trusted source or otherwise guaranteed not to be malicious. Globus also employs centralized indexes for resource discovery instead of the more flexible and robust P2P approach.

Programming language dependent class of Java distributed computing platforms can be divided into two: Java extensions and Java libraries. Java extensions such as JavaParty [44] provide special distribution mechanisms requiring changes to the Java compiler and/or Java Virtual Machine (JVM). This increases the difficulty of distributing an application. Java libraries provide special class libraries for the distribution without the need for modifications to the Java compiler or the JVM, and therefore the Java libraries are easier to deploy. An example of such library is JavaSymphony [13]. Like Globus, JavaSymphony employs a centralized index of available resources.

Some implementations of Java distributed computing that use peer-to-peer network for locating the resources do exist. In such design the resource index has been decentralized and peers cooperatively route resource queries among each other. An example of such a system is GT-P2PRMI [9], which allows Remote Method Invocation (RMI) lookups to be performed through an extended version of RMIRegistry called P2PRMIRegistry. P2PRMIRegistry is used to form the overlay network, for binding and publishing the remote methods and for looking up the published remote methods.

4.3 Network Simulators

Network simulators can be classified into two groups, packet-level simulators and message-level simulators. Packet-level simulators include real-life protocol headers in the simulation, which makes the simulation more accurate, but also slows down the simulation. A comparison of current simulators is presented in Table 1.

When training the NeuroSearch resource discovery algorithm, a simulator was practically the only choice to do the training. As described in publication **PXI**, there are various network simulators available that can be used in studying P2P networks. However, these simulators are not primarily designed for speed, and none of them contains functionalities required for training neural networks. As described in section 5.4.4, even the P2PRealm network simulator, which has been heavily optimized for speed, is not fast enough for training neural networks, and so the speed of the simulator was crucial. The requirements for training of the NeuroSearch algorithm are too unique and too computationally demanding to be run on top of a general P2P network simulator, thus it was justifiable to implement our own simulator.

TABLE 1 Characteristics of the current unstructured P2P network simulators

Simulator	Level of Detail	Parallel	Scalability	Overlay with Routers	Programming language
NS-2	Packets	Yes	Very low	Yes	C++
NS-3	Packets	Yes	Very low	Yes	C++ / Python
OPNET Modeler	Packets	Yes	High	Yes	C and C++
QualNet	Packets	Yes	n/a	Yes	C++
PLP2P	Packets	Yes	Medium	-	C++
QueryCycle	Messages	No	n/a	Yes	Java
3LS	Messages	No	Very low	Yes	Java
PeerSim	Messages	No	Very high	Yes	Java
NeuroGrid	Messages	No	High	No	Java
GPS	Messages	No	n/a	No	Java
P2PRealm	Messages	Yes	Medium	No	Java and C

5 CONTRIBUTIONS

This chapter presents briefly the contributions described in the included articles.

5.1 Social Mobile Peer-to-Peer

A social multimedia sharing application called PhotoJournal [**PI**,**PIV**] was developed in cooperation with the University of Crete and the FORTH institute from Greece. The application runs on the 7DS [42] mobile P2P middleware, enabling users to share location-tagged photos with other users of the application using a map-based user interface. The users' current location is determined using CLS [18] and/or GPS information, whichever happens to be available. This location information is also used to tag photos added to the system with location information.

A client/server version of the PhotoJournal was also developed, to enable comparisons between the two versions. The delay the application experiences while querying other nodes within the social network was measured as part of the research. In the measurements, the MP2P approach was compared with a more traditional client/server based approach where mobile devices accessed a centralized server disseminating information over a WiFi or a 3G link. The measurements show that in this case, the MP2P approach proved to be significantly faster than the infrastructure-based approach. Depending on network and device qualities, the median delay varied between 282ms and 1.9s.

Independent from the PhotoJournal project, novel search methods that could take advantage of the social network inherent to the mobile P2P architecture were investigated in Article **PII**. In this article, a prototype allowing the user to search for relevant information in a highly dynamic social network environment composed of mobile devices is presented. This kind of search functionality complements traditional web search engines, as the social network can be searched in real-time. The system also provides users fine-grained control of the privacy of information available for other people conducting searches. This differentiates the system from traditional search engines, as with them the privacy controls are very coarse – either the information can be found on Google, or not. The system is also completely decentralized, bringing all the basic benefits of P2P networks, which have already been described in Chapter 2. Since the research was published, both Google and Facebook have commercialized some ideas presented in the article.

5.2 Algorithms for Resource Discovery and Routing

A novel resource discovery algorithm for P2P networks, called NeuroSearch, was presented in Article **PVI**. The algorithm used a neural network model first presented in [15], where Fogel used evolving neural networks to create an artificial intelligence player for the game of checkers. The main difference between NeuroSearch and other algorithms, some of which are described in Section 3.1, is that NeuroSearch has been artificially developed by employing a neural network to make the forwarding decisions that control the behavior of the algorithm. This enables NeuroSearch to adapt to the current network scenario, unlike other resource discovery algorithms, which only employ one strategy to make forwarding decisions.

When a node using NeuroSearch receives a resource query, it first checks whether it has an instance of the queried resource, and if it does, it reports the resource to the querier. After this, the node inputs its neighbors' information and information from the query one by one to a neural network, which computes a decision whether the query should be forwarded to the neighbor in question. This process is depicted in Figure 3.

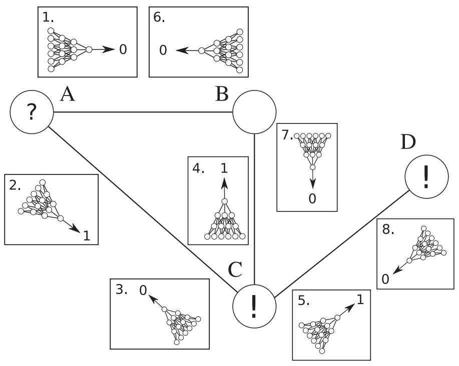


FIGURE 3 NeuroSearch query forwarding

To train the neural networks, our system uses an evolutionary method, where different neural networks compete against each other. In the beginning of training, 30 neural networks are randomly generated, tested, and compared to each other. The 15 worst performing networks are replaced with offspring of the 15 best performing networks. The offspring is created from each of the best performing networks by making random changes to the parents. This test-compare-replacement procedure is done thousands of times and so the neural networks gradually develop the ability to make good decisions for resource discovery. Finally, the best individual from the neural networks is selected to be the newly created NeuroSearch resource discovery algorithm.

Training a neural network is a very computationally demanding task. For example, when training a population of 30 neural networks for 100 000 generations, evaluating a neural network is done 3 million times. And when a single evaluation can consist of 100 queries to the P2P network and a single query can require hundreds of neural network based decisions, the training requires large amounts of computing power.

The NeuroSearch algorithm was further refined by the addition of memetic characteristics to the training model in Articles **PVII** and **PVIII**. As defined by Moscato et al in [39], memetic algorithms combine evolutionary algorithms with separate local search algorithms to create algorithms capable of finding good solutions more efficiently than evolutionary algorithms alone.

All of these algorithms were also compared to two traditional resource discovery algorithms, BFS and DFS, and with one traditional genetic algorithm(GA) employing crossover and mutation to train the neural network. The results show that AGLMA, the most complex of the algorithms, took longer to train the neural network, but eventually it generated the best algorithm.

Table 2 shows the optimization results. The final fitness $\hat{F}^b$ obtained by the most successful experiment over 30 sample runs, the related number of query packets P used in the query and the number of found resource instances R during the query are given. In addition the average best fitness at the end of the experiments $< \hat{F} >$, the final fitness of the least successful experiment $\hat{F}^w$ and the related standard deviation are shown. Since the BFS follows a deterministic logic, thus only one fitness value is shown. On the contrary, the DFS under study employs a stochastic structure and thus the same statistical analysis as that of GA, NS, ANS and AGLMA over 30 experiments has been carried out.

TABLE 2 Comparison of different NeuroSearch versions with other resource discovery algorithms

	AGLMA	ANS	NS	GA	BFS	DFS
P	350	355	372	497	819	514
R	81	81	81	85	81	81
$\hat{F}^b$	3700	3695	3678	3366	3231	3536
$< \hat{F} >$	3654	3647	3582	2705	–	3363
$\hat{F}^w$	3506	3504	3502	0	–	3056
std	36.98	36.47	37.71	1068	–	107.9

A similar approach was also taken to generate an algorithm for solving the mobility assisted routing problem in Article **PV**, where an algorithm called NeuroRouter was proposed. As with NeuroSearch, the algorithm employs a trained neural network to make decisions, though in this case the decisions are about routing in a mobile encounter network. When encountering a node, information about the encountered node with information about the message being processed are fed to a neural network, the output of which defines whether the message is forwarded to the encountered node. This makes the algorithm flexible and enables it to employ different strategies for different situations. The algorithm could also be further improved with the addition of social-aware features to the input of the neural networks.

5.3 Location sensing

FINDR, a location sensing system primarily for indoor use, is presented in Article **PIII**. FINDR is based on measuring signal strengths from stationary transmitters, but instead of measuring WLAN or Bluetooth signals the system measures the strength of FM radio signals transmitted from cheap MP3 players with built in FM radio transmitters. In the evaluation of the system, we used 3 FM radios transmitting on different frequencies, and a Nokia N800 device to measure the received signal strength. The system employs the fingerprinting method, where the signal strengths are measured within certain intervals in the map space. When using the system, measured signal strengths are then compared against the signal strengths in the fingerprint database to find out the current location.

When compared to WLAN base stations being used in other similar location sensing systems, the FM radio transmitters are much cheaper to procure and are much smaller in size. They also consume a minimal amount of power, which enables them to even run on batteries, if required. Any computing device that can measure the strength of an incoming FM radio signal can be used as a FINDR client device, and because the system does not require any specialized hardware, it is also easily deployable.

The FINDR system evaluation shows a median accuracy of 1.3m, and an accuracy of 4.5m at 95% confidence level, thus comparing favourably with similar WLAN-based systems.

The FINDR system was developed in cooperation with the Create-Net institute from Trento, Italy.

5.4 Tools

It was not always possible to find a tool fulfilling the requirements set by the research method and the research problem, and thus several tools were developed in the research process.

5.4.1 Chedar

Although the Chedar application falls outside of the scope of this dissertation, a short introduction to it is required as several of the tools described in the following chapters depend on Chedar.

Chedar (CHEap Distributed ARchitecture, [5]) is a pure peer-to-peer middleware designed for peer-to-peer applications. The goal of the Chedar software is to provide a convenient API for peer-to-peer application developers. Chedar peers use TCP connections to communicate with each other. Chedar can be used to distribute different kinds of resources to other nodes in the Chedar network. Distributed resources can be for example files, CPU time or storage space. Every node stores information about the resources it provides in XML format. When the node receives a query about a resource it checks whether it has the resource by checking the resource XML database via an XPath query.

5.4.2 Mobile Chedar – A P2P Middleware

Mobile phones have very limited capacities of memory and processing power, and also applications running on phones should conserve battery power as much as possible. On the other hand mobile applications are becoming more complex all the time, and users demand more and better functionality from mobile applications. Mobile Chedar, a system that allows mobile devices to access resources from workstations in the Chedar fixed-network P2P system, is presented in publication **PIX**, and an application using Mobile Chedar is presented in publication [31].

Mobile Chedar has been implemented using the Java 2 Micro Edition (J2ME) programming language and it uses Bluetooth [19] as a communication technology. Mobile Chedar is an extension to the the Chedar fixed-network P2P middleware, and it provides mechanisms for data streaming between Mobile Chedar nodes and between the Mobile Chedar and Chedar networks.

Let us look at one case where this could be useful. Mobile users might want to view high quality videos on their phones, but the phone's processor does not have enough power to decode such a stream and delivering the stream uncompressed over a 3G network would not work because of bandwidth limitations. One solution to the problem would be to use Mobile Chedar to get a computer from the Chedar network to fetch the video stream from the internet, re-encode the video to better fit the mobile device, and then send it to the requesting mobile phone using the free and relatively fast Bluetooth network.

5.4.3 P2PRealm – A P2P Network Simulator

Originally the NeuroSearch training was done directly on the Chedar P2P network, running on computers located at the computer laboratories of the university. As was mentioned in Chapter 5.3, this turned out to be a very time consuming process, as all the query messages used in the training needed to be physically sent between computers participating in the P2P network.

Due to the prohibitive slowness of the original NeuroSearch training approach, there was a need to find a simulator to train NeuroSearch, and as no off-the-shelf solutions were suitable, the decision was made to develop the P2PRealm network simulator. P2PRealm (Article **PXI**) is a peer-to-peer network simulator that has been designed to be as fast as possible due to the high CPU time requirements of NeuroSearch training. An average neural network training run takes about a week on an ordinary PC using P2PRealm.

Using building blocks from Chedar and P2PStudio, the first version of the P2PRealm simulator was developed very quickly, in about one working day. The same neural network training scenario was run on the early prototype as in the Chedar system and the results were very promising: running the neural network training in the simulator was about three magnitudes, or a thousand times faster than on the Chedar system. Nowadays the simulator has grown to include six classic resource discovery algorithms, and also a k-Steiner tree based algorithm for finding the upper bounds of resource discovery algorithm performance.

The simulator source code has also been heavily optimized to gain more efficiency. Several bottlenecks were identified by profiling the simulator and these bottlenecks neutralized using various software optimization techniques.

5.4.4 P2PDisCo – P2P Distributed Computing Middleware

Even with the simulator approach presented in the previous section, NeuroSearch training was not quick enough to run our simulations, because there are dozens of parameters to set in the training process, thousands of training sessions are required to find out an optimal set of parameters for the training process. As a single training simulation took about a week, it was apparent that thoroughly testing the effect of different parameters in different scenarios was prohibitively time-consuming. The next step to speed up NeuroSearch training was to distribute the P2PRealm simulator to run in parallel on several computers. This was achieved by implementing the P2PDisCo middleware.

P2PDisCo (publication **PX**) is a distributed computing middleware running on top of the Chedar [5] P2P network middleware. It can be used to distribute any Java application, as long as the application only uses files for input and output, with minor modifications. P2PDisCo redirects the I/O operations of distributed applications to access memory buffers controlled by P2PDisCo instead of accessing local files. Thus the distributed application does not see any difference between running locally and running on top of P2PDisCo, and P2PDisCo runs with minimal impact and requirements to the underlying computer system because no disk space is required for the I/O files. This was a requirement when the NeuroSearch training was run on hundreds of desktop PC computers located in the computer labs of the Agora building at the University of Jyväskylä. As the workstations' processors would otherwise be idle for most of the time, P2PDisCo enabled this idle resource to be used for computation.

When the master node is started, it connects to the Chedar network and starts a query looking for idle computing resources, and the peers that are ready for computing answer the query. The master node selects which of the located nodes will be used for computing and distributes tasks to these nodes which start the computation. During the computation, results are sent back to the master peer when the memory buffer reaches 256 kB, thus the P2PDisCo does not require large amounts of memory on the computing peers. Also, this ensures that if the computing node is reset, the computation results are still saved to the point of the last update.

5.4.5 P2PStudio

In order to test and monitor the Chedar network there was a need for a tool that enables the researchers to remotely control and monitor each Chedar network node in a centralized way. To enable Chedar nodes to be monitored and controlled, the Chedar client also has the capability to be connected for monitoring and control. P2PStudio (publication **PXII**) is a monitoring, controlling and visualization tool that uses this functionality to connect to Chedar nodes to collect data from them and to control their behavior. P2PStudio is comprised of two distinct applications: the user interface and the server. The server application connects to Chedar nodes to control them and to collect data, such as event, topology and resource information. The user interface communicates with the server and interfaces with the user.

The P2PStudio user interface component can also be used as a standalone application without a connection to the server when visualizing network topologies and graphs generated by the P2PRealm (Article **PXI**) network simulator.

6 CONCLUSION

The history of P2P networks dates back to the 1960's, when the P2P architecture was the basis for some of the earliest large-scale computer networks. During the following decades the architecture received only limited attention, as computing at the time was mostly dominated by mainframes and client/server systems. Only during the last 15 years have P2P networks had a revival due to the advent of the internet. Peer-to-peer networks are an exciting field to do research in, as the state of the art is moving forward at such a great pace. This dissertation suggests advancements on a wide range of research subjects, from algorithms to middleware prototypes and from application ideas to research tools.

Due to the distributed nature of the P2P architecture, finding available resources in the network is a difficult problem. The resource discovery should be done with minimal computing power and network bandwidth usage to prevent network congestion. NeuroSearch, a resource discovery algorithm presented in Article **PVI**, provides one solution to this problem. NeuroSearch employed trained neural networks to make the routing decisions, and has been evaluated to perform better than reference algorithms presented in the relevant literature. Memetic algorithms were later added to the NeuroSearch training process, as described in Articles **PVII** and **PVIII**, which further improved the efficiency of the algorithm.

This dissertation also discussed mobile peer-to-peer networks, both from an algorithmic point of view and from an applications point of view. Due to the mobility of the nodes, mobile peer-to-peer networks often gain and lose connections all the time. When the network becomes sparse enough that the devices can only maintain connections to other devices during short encounters, the network is called a mobile encounter network. Routing in mobile encounter networks is discussed in Article **PV**, which aimed to solve the routing problem using similar ideas that were used to tackle the resource discovery problem in the NeuroSearch articles.

Location-sensing is an important requirement for many mobile applications, whether they utilize P2P networking or not. Location-sensing was explored in Article **PIII**, which introduced and evaluated a location-sensing system based on measuring the signal strength received from low-power FM radio transmitters. The system was found to be as precise as WLAN-based systems, with much lower costs and smaller power consumption.

Articles **PI**, **PII** and **PIV** presented and evaluated two mobile peer-to-peer application prototypes. PhotoJournal was a location-based media sharing application utilizing low range wireless networks, which harnessed the WiFi ad-hoc mode to enable the users to share content with other nearby users. The Mobile Search application on the other hand utilized standard web protocols and servers running on mobile devices to enable users to make real-time searches within their social networks.

The dissertation also presented several research tools used in the development of the NeuroSearch resource discovery algorithm. A system where the Chedar P2P network was running the tests controlled by the P2PStudio tool, described in **PXII**, was originally used in training the neural networks of NeuroSearch. This approach was discovered to be too slow – a single neural network training in a P2P network with a hundred nodes would have taken almost a year. The next step was to implement a network simulator that could be used to speed up the training of NeuroSearch. Thus P2PRealm, described in **PXI**, was born. The simulator is unique in the sense that it has been designed for training neural networks and as such it is very optimized for speed. The simulator speeded up the training about 1000-fold, but even this was not fast enough for us. To speed up the training even further, P2PDisCo, described in **PX**, was developed to distribute the computing needed in training NeuroSearch to hundreds of PCs. P2PDisCo is a general distributed computing platform running on top of the Chedar middleware.

A mobile extension to the Chedar network called Mobile Chedar was also presented in Article **PIX**. Mobile Chedar was able to utilize resources from the fixed Chedar network and to provide mechanisms for data streaming within the network.

Even after the advances described in this dissertation, several research questions remain open. As the progress in the field of P2P networks is very intensive, new research directions are guaranteed to appear. It follows that P2P networks are bound to be an interesting research field for the foreseeable future.

YHTEENVETO (FINNISH SUMMARY)

Suurin osa nykyisin käytössä olevista verkkopalveluista on suunniteltu käyttäen asiakas/palvelin-mallia, jossa palvelinlaite käsittelee lukuisien asiakaslaitteiden pyyntöjä. Asiakas/palvelin-mallia käytetään sen selkeän rakenteen ja helpon hallittavuuden vuoksi. Asiakas/palvelin-malli ei kuitenkaan sovellu kaikkiin käyttötilanteisiin mallin heikkouksien vuoksi. Mallin mukaan toteutetut järjestelmät skaalautuvat usein huonosti suuriin käyttäjämääriin ja vaativat kalliita alkuinvestointeja. Järjestelmät ovat myös alttiita niiden vakautta uhkaaville tekijöille järjestelmän keskitetyn luonteen takia.

Näitä ongelmia voidaan ratkoa käyttämällä vertaisverkkomallia verkkopalveluiden suunnittelussa. Vertaisverkko on tietoverkko, jossa verkon kaikki toimijat ovat tasa-arvoisessa asemassa ja voivat toimia sekä palvelin- että asiakaslaitteina eli voivat sekä tarjota että kuluttaa verkon palveluita. Vertaisverkkojen hajautetun luonteen ansiosta vertaisverkkojärjestelmät eivät yleensä kärsi näistä ongelmista. Eduistaan huolimatta vertaisverkot eivät sovi kaikkiin käyttötarkoituksiin, sillä niiden hajautettu luonne luo erityyppisiä ongelmia. Esimerkiksi resurssien löytäminen vertaisverkosta on vaikeaa, koska verkossa ei ole keskitettyä rekisteriä resursseista, vaan ne sijaitsevat hajallaan verkossa.

Tässä väitöskirjassa, jonka otsikko on "Menetelmiä ja sovelluksia vertaisverkkokäyttöön", tarjotaan ratkaisuja vertaisverkkojen ongelmiin ja uusia menetelmiä vertaisverkkojen toiminnan tehostamiseen. Tavoitteena on siis laajentaa perinteistä vertaisverkkojen käyttöaluetta. Tämän lisäksi väitöskirja esittelee vertaisverkkoja hyödyntäviä sovellusprototyyppejä sekä sovellusideoita. Väitöskirjassa esitellään myös useita tutkimuksen avuksi kehitettyjä työkaluohjelmistoja. Väitöskirjassa esiteltyjä tuloksia voi hyödyntää uudentyyppisten verkkopalveluiden ja palvelualustojen kehityksessä.

REFERENCES

[1] L. Adamic, R. Lukose, and B. Huberman. Local Search in Unstructured Networks. In S. Bornholdt and H. Schuster, editors, *Handbook of Graphs and Networks:From the Genome to the Internet*. Wiley-VCH, Berlin, 2000.

[2] I. Akyildiz and X. Wang. *Wireless Mesh Networks*. John Wiley & Sons, Inc., 2009.

[3] S. M. Allen, G. Colombo, and R. M. Whitaker. Cooperation through self-similar social networks. *ACM Trans. Auton. Adapt. Syst.*, 5:4:1–4:29, February 2010.

[4] Z. Anwar, W. Yurcik, and R. H. Campbell. A survey and comparison of peer-to-peer group communication systems suitable for network-centric warfare. *SPIE Security and Defense Conference, Program on Communications and Networking Technologies and Systems*, 2005.

[5] A. Auvinen, M. Vapa, M. Weber, N. Kotilainen, and J. Vuori. Chedar: peer-to-peer middleware. In *Parallel and Distributed Processing Symposium, 2006. IPDPS 2006. 20th International*, page 7 pp., april 2006.

[6] N. Bailey. *The Mathematical Theory of Infectious Diseases*. Hafner, 1975.

[7] E. Bulut and B. Szymanski. Friendship based routing in delay tolerant mobile social networks. In *GLOBECOM 2010, 2010 IEEE Global Telecommunications Conference*, pages 1 –5, dec. 2010.

[8] S. Buruhanudeen, M. Othman, and B. Ali. Existing manet routing protocols and metrics used towards the efficiency and reliability- an overview. In *Telecommunications and Malaysia International Conference on Communications, 2007. ICT-MICC 2007. IEEE International Conference on*, pages 231 –236, may 2007.

[9] T. Chang and M. Ahamad. GT-P2PRMI: Improving middleware performance using peer-to-peer service replication. *Future Trends of Distributed Computing Systems, IEEE International Workshop*, pages 172–177, 2004.

[10] B. G. Christensen. Lightpeers: A lightweight mobile p2p platform. In *Pervasive Computing and Communications Workshops, 2007. PerCom Workshops '07. Fifth Annual IEEE International Conference on*, pages 132 –136, march 2007.

[11] W. Dargie and C. Poellabauer. *Fundamentals of Wireless Sensor Networks: Theory and Practice*. Wireless Communications and Mobile Computing. John Wiley & Sons, 2010.

[12] P. T. Eugster, R. Guerraoui, A.-M. Kermarrec, and L. Massoulieacute;. Epidemic information dissemination in distributed systems. *Computer*, 37:60–67, 2004.

[13] T. Fahringer. JavaSymphony: a system for development of locality-oriented distributed and parallel Java applications. In *Cluster Computing, 2000. Proceedings. IEEE International Conference on*, pages 145 –152, 2000.

[14] K. Fall. A delay-tolerant network architecture for challenged internets. In *SIGCOMM '03: Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 27–34, New York, NY, USA, 2003. ACM.

[15] D. B. Fogel. Evolving a checkers player without relying on human experience. *Intelligence*, 11(2):20–27, 2000.

[16] I. Foster and C. Kesselman. Globus: A metacomputing infrastructure toolkit. *The International Journal of Supercomputer Applications and High Performance Computing*, 11:115–128, 1997.

[17] I. Foster, Y. Zhao, I. Raicu, and S. Lu. Cloud computing and grid computing 360-degree compared. In *Grid Computing Environments Workshop, 2008. GCE '08*, pages 1 –10, nov. 2008.

[18] C. Fretzagias and M. Papadopouli. Cooperative location-sensing for wireless networks. In *Pervasive Computing and Communications, 2004. PerCom 2004. Proceedings of the Second IEEE Annual Conference on*, pages 121–131, March 2004.

[19] B. S. I. Group. Bluetooth core specification v1.2. March 2004.

[20] I. Gruber, R. Schollmeier, and W. Kellerer. Performance evaluation of the mobile peer-to-peer service. *Cluster Computing and the Grid, IEEE International Symposium on*, 0:363–371, 2004.

[21] B. Hayes. Cloud computing. *Commun. ACM*, 51:9–11, July 2008.

[22] T. Horozov, A. Grama, V. Vasudevan, and S. Landis. Moby — a mobile peer-to-peer service and data network. In *Proceedings of International Conference on Parallel Processing*, pages 437–444, 2002.

[23] P. Hui, A. Chaintreau, J. Scott, R. Gass, J. Crowcroft, and C. Diot. Pocket switched networks and human mobility in conference environments. In *WDTN '05: Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*, pages 244–251, New York, NY, USA, 2005. ACM.

[24] S. Jain, K. Fall, and R. Patra. Routing in a delay tolerant network. In *SIGCOMM '04: Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 145–158, New York, NY, USA, 2004. ACM.

[25] G. Y. Jin, X. Y. Lu, and M. S. Park. An indoor localization mechanism using active rfid tag. *Sensor Networks, Ubiquitous, and Trustworthy Computing, International Conference on*, 1:40–43, 2006.

[26] B. J. Kim, C. N. Yoon, S. K. Han, and H. Jeong. Path finding strategies in scale-free networks. *Physical Review E*, 65, 2002.

[27] B. Klein and H. Hlavacs. A socially aware caching mechanism for encounter networks. *Telecommunication Systems*, pages 1–8, 2011.

[28] D. Kondo, G. Fedak, F. Cappello, A. A. Chien, and H. Casanova. Characterizing resource availability in enterprise desktop grids. *Future Gener. Comput. Syst.*, 23(7):888–903, 2007.

[29] T. Kopomaa. *The city in your pocket: Birth of the mobile information society*. Gaudeamus, 2000.

[30] G. Kortuem. Proem: a middleware platform for mobile peer-to-peer computing. *Mobile Computing and Communications Review*, 6:62–64, 2002.

[31] J. Kurhinen, M. Vapa, M. Weber, N. Kotilainen, and J. Vuori. Short range wireless p2p for co-operative learning. In *Proceedings of the 3rd International Conference on Emerging Telecommunications Technologies and Applications*. University of South Bohemia, 2005.

[32] J. Kuula. Langattoman anturiverkon hyödyntäminen parsinavetan ilmanvaihdon ohjauksessa. Master's thesis, University of Jyväskylä, 2011.

[33] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker. Search and replication in unstructured peer-to-peer networks. In *Proceedings of the 16th International Conference on Supercomputing*. ACM Press, 2002.

[34] N. A. Lynch. *Distributed Algorithms*. Morgan Kaufmann publishers, 1997.

[35] S. Marti and H. Garcia-Molina. Taxonomy of trust: Categorizing P2P reputation systems. *Computer Networks*, 50(4):472 – 484, 2006. <ce:title>Management in Peer-to-Peer Systems</ce:title>.

[36] C. Mascolo, L. Capra, S. Zachariadis, and W. Emmerich. XMIDDLE: A data-sharing middleware for mobile computing. *International Journal on Wireless Personal Communications, Kluwer Academic Publisher*, 21:77–103, 2002.

[37] C. Mbarushimana and A. Shahrabi. Comparative study of reactive and proactive routing protocols performance in mobile ad hoc networks. In *Advanced Information Networking and Applications Workshops, 2007, AINAW '07. 21st International Conference on*, volume 2, pages 679 –684, may 2007.

[38] A. Mei, G. Morabito, P. Santi, and J. Stefa. Social-aware stateless forwarding in pocket switched networks. In *INFOCOM, 2011 Proceedings IEEE*, pages 251 –255, april 2011.

[39] P. Moscato, R. Berretta, and C. Cotta. *Memetic Algorithms*. John Wiley & Sons, Inc., 2010.

[40] A. Oram, editor. *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*. O'Reilly & Associates, Inc., 2001.

[41] R. Orfali, J. Edwards, and D. Harkey. *Essential Client/Server Survival Guide*. John Wiley & Sons, Inc., New York, NY, USA, 1994.

[42] M. Papadopouli and H. Schulzrinne. Design and implementation of a peer-to-peer data dissemination and prefetching tool for mobile users. In *Proceedings of the First New York Metro Area Networking Workshop*, 2002.

[43] M. Papadopouli and H. Shulzrinne. *Peer-to-Peer Computing for Mobile Networks: Information Discovery and Dissemination*. Springer, 2009.

[44] M. Phillipsen and M. Zenger. JavaParty: Transparent remote objects in java. *Concurrency and Computation: Practice and Experience*, 9:1225–1242, 1997.

[45] A. Popleteev. *Indoor positioning using FM radio signals*. PhD thesis, University of Trento, 2011.

[46] N. Sastry, D. Manjunath, K. Sollins, and J. Crowcroft. Data delivery properties of human contact networks. *Mobile Computing, IEEE Transactions on*, 10(6):868 –880, june 2011.

[47] R. Schollmeier. A definition of peer-to-peer networking for the classification of peer-to-peer architectures and applications. In *Peer-to-Peer Computing, 2001. Proceedings. First International Conference on*, pages 101 –102, aug 2001.

[48] T. Spyropoulos, K. Psounis, and C. S. Raghavendra. Spray and wait: an efficient routing scheme for intermittently connected mobile networks. In *Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*, WDTN '05, pages 252–259, New York, NY, USA, 2005. ACM.

[49] T. Spyropoulos, K. Psounis, and C. S. Raghavendra. Spray and focus: Efficient mobility-assisted routing for heterogeneous and correlated mobility. In *Pervasive Computing and Communications Workshops, 2007. PerCom Workshops '07. Fifth Annual IEEE International Conference on*, pages 79 –85, march 2007.

[50] E. P. Stuntebeck, S. N. Patel, T. Robertson, M. S. Reynolds, and G. D. Abowd. Wideband powerline positioning for indoor localization. In *UbiComp '08: Proceedings of the 10th international conference on Ubiquitous computing*, pages 94–103, New York, NY, USA, 2008. ACM.

[51] F. S. Tsai, W. Han, J. Xu, and H. C. Chua. Design and development of a mobile peer-to-peer social networking application. *Expert Syst. Appl.*, 36(8):11077–11087, 2009.

[52] Y. Upadrashta, J. Vassileva, and W. Grassmann. Social networks in peer-to-peer systems. In *HICSS '05: Proceedings of the Proceedings of the 38th Annual*

Hawaii International Conference on System Sciences, page 200.3, Washington, DC, USA, 2005. IEEE Computer Society.

[53] A. Varshavsky, E. de Lara, J. Hightower, A. LaMarca, and V. Otsason. GSM indoor localization. *Pervasive Mob. Comput.*, 3(6):698–720, 2007.

[54] A. Varshavsky, E. de Lara, J. Hightower, A. LaMarca, and V. Otsason. ATLINTIDA: A robust indoor ultrasound location system: Design and evaluation. *Advances in Soft Computing*, 51:180–190, 2009.

[55] C. Wang, Q. Wang, K. Ren, and W. Lou. Privacy-preserving public auditing for data storage security in cloud computing. In *INFOCOM, 2010 Proceedings IEEE*, pages 1 –9, march 2010.

[56] M. Weber, J. Vuori, and M. Vapa. Advertising peer-to-peer networks over the internet. *Radiotekhnika*, 133:162–170, 2003.

ORIGINAL PAPERS

PI

LOCATION-BASED MEDIA SHARING IN A MP2P NETWORK

by

Niko Kotilainen, Lito Kriara, Konstantinos Vandikas, Konstantinos Mastorakis
and Maria Papadopouli 2008

In ACM SIGMOBILE Mobile Computing and Communications Review,
volume 12, issue 1, pages 62-64

Demonstration Abstract: Location-based Media Sharing in a MP2P Network [*]

Niko Kotilainen[b] **Lito Kriara**[a] **Konstantinos Vandikas**[a] **Konstantinos Mastorakis**[a]
Maria Papadopouli[a]

[a]Department of Computer Science, University of Crete, Greece &
Institute of Computer Science (ICS), Foundation for Research and Technology - Hellas (FORTH)
[b]Department of Mathematical Information Technology, University of Jyväskylä, Finland

In both academia and industry, peer-to-peer (p2p) applications have attracted great attention. This paper introduces and implemented a novel location-based multimedia application, the Multimedia Traveling Journal application (PhotoJournal) that employs the p2p paradigm and enables location-based content sharing among mobile users.

I. Introduction

The Web and Internet have been catalysts for the creation of collaborative applications and tools. On-line collaboration has been enriched with new applications and tools for sharing and experimenting with multimedia data in a synchronous or asynchronous manner, such as YouTube and Flickr. These technologies have allowed the formation of new types of social networks, interactions, and online communities. We anticipate that in the near future mobile devices that have the processing, communication and geolocating capabilities will enable seamless integration of services combining media sharing and geographical tagging.

The Multimedia Traveling Journal application (PhotoJournal) applies the peer-to-peer (p2p) paradigm to share location-based content among mobile devices. It also enables users to build interactive multimedia journals that associate multimedia objects such as pictures, video, or hypertext, with locations on maps. The PhotoJournal is supported by a middleware with two main components, namely a positioning and an information discovery system. The underlying positioning technologies are GPS and Cooperative Location-sensing System (CLS) [1, 2]. 7DS [3] enables information discovery and sharing in a p2p manner. Section II focuses on PhotoJournal and briefly introduces CLS and 7DS, while Section III summarizes our main conclusions and future work plans.

[*]This work was supported by the Greek General Secretariat for Research and Technology under Programs Regional of Crete, Crete-Wise KP-18 (ΚΠΣ 00126) and 05NON-EU-238, and the European Commission (MIRG-CT-2005-029186). Niko Kotilainen participated in this project while visiting the University of Crete and FORTH. Contact person: Maria Papadopouli (mgp@ics.forth.gr).

II. Architecture of PhotoJournal

In general when an application requests a data object, 7DS first checks its cache, and if the data is not available, it tries to acquire it from the Internet. If the local web client fails to connect to the Internet, the local 7DS instance multicasts a query about that object in the wireless LAN. Figure 1 summarizes the main components of the location-based media sharing system, namely the PhotoJournal application, 7DS and CLS.

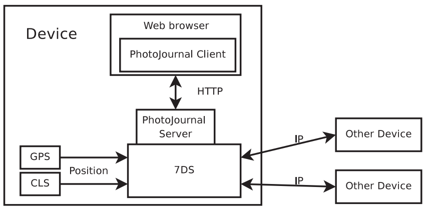


Figure 1: The architecture of a location-based media sharing system.

Through 7DS, PhotoJournal allows peers to share files associated with certain locations. The multimedia files and maps are stored in the cache of the local 7DS instance. A user can add multimedia objects to a certain point of the map by clicking on the map and browsing the image files corresponding to that location. Moreover, the user can add, modify, or delete comments on a certain multimedia file, and rate its content. A multimedia file can be set public or private, while only public files are shared with other peers.

The PhotoJournal searches other 7DS peers for multimedia files associated with a given area marked on the map by the user. It forms a 7DS query and mul-

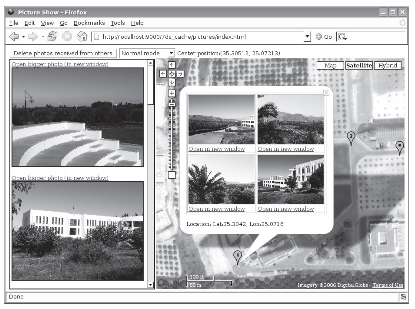
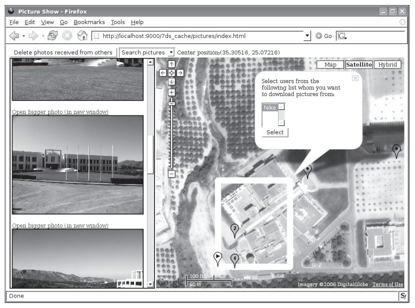

Picture Show - Firefox
File Edit View Go Bookmarks Tools Help
http://localhost:9000/7ds_cache/pictures/index.html
Go
Delete photos received from others
Normal mode
Center position:(35.30512, 25.07213)
Open bigger photo (in new window)
Map
Satellite
Hybrid
Open in new window
Location: Lat:35.3042, Lon:25.0716
Done

Figure 2: PhotoJournal can superimpose multimedia objects at their locations on a map. A marker indicates the number of files associated with that location.


Picture Show - Firefox
File Edit View Go Bookmarks Tools Help
http://localhost:9000/7ds_cache/pictures/index.html
Go
Delete photos received from others
Search pictures
Center position:(35.30516, 25.07216)
Open bigger photo (in new window)
Map
Satellite
Hybrid
Select
Done

Select users from the following list whom you want to download pictures from:

Figure 3: A user can mark the area for which multimedia objects are requested.

ticasts it to other 7DS peers. Furthermore, it maintains and displays a list of neighboring 7DS peers, updating it periodically. A user may then select the 7DS peers from which the application retrieves the files associated with the predefined area, stores them in the local cache and displays them on the map (as illustrated in Figures 2 and 3). Areas on the map associated with multimedia files can be distinguished by a marker that also indicates the number of the available relevant files. Moreover, the queries are formed using location-based or rate-related criteria. The response of a peer includes the multimedia files and reviews.

As shown on Figure 3, a web browser is the frontend of PhotoJournal. It consists of a map frame on the right and a photo bar on the left side of the window. Its backend runs on 7DS. It receives all queries from the frontend through 7DS's proxy server, and supports the typical 7DS functionality by adding or deleting photos, querying photos from 7DS neighbors or handing out photos from the local cache. 7DS can also cache map files, enabling the application to work without an Internet connection.

If the device has a built-in camera, users can take photos and videoclips and upload this information along with position traces (produced by GPS or CLS) to the PhotoJournal which can automatically associate the multimedia files with the user's current position. The PhotoJournal can automatically superimpose the uploaded content on an appropriate map by matching the timestamp of the content of the multimedia files with the timestamp of the GPS/CLS trace. Furthermore it locates them on the map, and updates its local 7DS cache.

CLS is a novel location sensing system that employs the p2p paradigm and a probabilistic framework to estimate the position of wireless-enabled devices in an iterative manner. CLS can incorporate signal-strength maps of the environment to improve the position estimates. Such maps have been built using measurements that were acquired from access points and peers during a training phase. Periodically, CLS can refine its positioning estimations by incorporating newly received information from other devices.

At run-time, the local CLS instance acquires signal-strength measurements from peers, constructs a run-time signature, and compares it with the ones that have been generated during the training phase. For the comparison, it employs confidence interval-based and percentiles-based criteria.

CLS adopts a grid-based representation of the physical space; each cell of the grid corresponds to a physical position of the physical space. The cell size reflects the spatial granularity/scale. Each cell of the grid is associated with a value that indicates the likelihood that the node is in that cell. These values are computed iteratively using one of the following two approaches, namely a voting algorithm and a particle filter based model.

In the voting process, a local CLS instance casts votes on cells of the grid based on measurements received from peers. A signature associates each cell of the grid with a vector of statistical information of the RSSI values that were recorded from messages received from those peers during the training and runtime phases. The algorithm assigns a weight at each cell depending on the similarity of the training and runtime signatures. The cell with the highest weight is the one that CLS reports as the user's position.

The CLS particle-filter based approach can be formulated in probabilistic terms as the problem of determining the probability of a node being at a certain location given a sequence of signal strengths. Assuming first-order Markov dynamics, the above problem can be expressed using the network graph depicted in Figure 4, where x_k is the node location (system state) at time instant $k = 1, \ldots, T$. x_k cannot be observed directly (it is "hidden"). Yet, for each location x_k, a measurement vector y_k (signal strength) is available that depends on the hidden variable according to a known observation function.

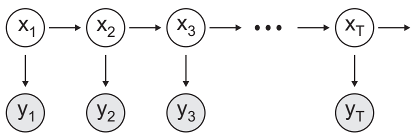


Figure 4: Clear circles indicate hidden state variables, grayed circles indicate observations, horizontal arrows indicate state transition functions and vertical arrows indicate observation functions.

Due to the Markov assumption, each node location, given its immediately previous location, is conditionally independent of all earlier locations, that is $P(x_k|x_0, x_1, \ldots, x_{k-1}) = P(x_k|x_{k-1})$. Similarly, the observation at the k-th time instant, given the current state, is conditionally independent of all other states $P(y_k|x_0, x_1, \ldots, x_k) = P(y_k|x_k)$.

Based on the this model, location-sensing can be formulated as the problem of computing the location x_k of a node at time k, given the sequence of observations $y_1, y_2, ...y_k$, up to time k, that is, determining the *a posteriori* distribution $P(x_k|y_1, y_2, \ldots, y_k)$, using particle filters. To generate and maintain the particles, we utilize the Sampling/Importance Resampling (SIR) algorithm introduced by Rubin [4]. According to SIR, instead of sampling the true *a posteriori* distribution (which is not possible because this distribution is not available in closed form), particles are drawn from the so-called proposal distribution $\pi(x_k|y_1, y_2, \ldots, y_k)$. To compensate for this difference, each particle $s^{(L)}$ is also assigned a weight $w^{(L)}$, which is computed, according to the Importance Sampling Principle: $w_t^{(L)} = \frac{P(x_k|y_1,y_2,\ldots,y_k)}{\pi(x_k|y_1,y_2,\ldots,y_k)}$

The performance of CLS was evaluated in Telecommunication and Networks Laboratory at FORTH, an area of $7x12m^2$ and a median location error of 1.8m was reported [2].

III. Conclusions

This implementation of PhotoJournal has adopted a p2p architecture. We are implementing a more centralized approach for thin devices (e.g. smart-phones) in which a client acquires and sends the multimedia files to a webserver. The hybrid architecture facilitates both the centralized and p2p approaches, enabling devices to acquire the data either from a web-server or another peer. We are in the process of deploying a testbed in an aquarium and perform user studies. Furthermore, we will evaluate the delay and scalability of these architectures via empirical-based measurements and simulations.

Privacy plays a critical role in the adoption of mobile peer-to-peer computing applications. 7DS-like systems facilitate sharing among devices in different types of environments. However, depending on the application and usage peers may have different privacy requirements. Currently, 7DS offers a crude distinction between private and non-private objects and a finer description of the privacy requirements is required.

Mobile p2p computing is a relatively new technology, not yet proven in the research community and industry. A fruitful approach would be to develop a general infrastructure for mobile peer-to-peer applications, build some robust applications, and extract a toolkit that other new applications could use. Our group sets the directions for exploring this technology further.

References

[1] C. Fretzagias and M. Papadopouli, "Cooperative location-sensing for wireless networks", *Second IEEE International conference on Pervasive Computing and Communications 2004*, Orlando, Florida, March 14-17, 2004.

[2] K. Vandikas, L. Kriara, T. Papakonstantinou, A. Katranidou, H. Baltzakis and M. Papadopouli, "Empirical-based analysis of a cooperative location-sensing system", *First International Conference on Autonomic Computing and Communication Systems (ACM Autonomics'07)*, Rome, Italy, October 28-30, 2007.

[3] M. Papadopouli and H. Schulzrinne, "Effects of power conservation, wireless coverage and cooperation on data dissemination among mobile devices", *ACM SIGMOBILE Symposium on Mobile Ad Hoc Networking & Computing (MobiHoc) 2001*, October 4-5, 2001, Long Beach, California.

[4] D.B. Rubin, "Using the SIR Algorithm to Simulate Posterior Distributions.", *Bayesian Statistics. Oxford University Press. vol.3, p.395-402*, 1988.

PII

MOBILE SEARCH – SOCIAL NETWORK SEARCH USING MOBILE DEVICES

by

Pedro Tiago, Niko Kotilainen, Heikki Kokkinen, Jukka Nurminen and Mikko Vapa 2008

In Proceedings of the 5th IEEE Consumer Communications and Networking Conference, pages 1201-1205

Mobile Search – Social Network Search Using Mobile Devices

Pedro Tiago
FCT, Universidade Nova de Lisboa
Caparica, PORTUGAL
ptiago@gmail.com

Niko Kotilainen
University of Jyväskylä
Jyväskylä, FINLAND
niko.kotilainen@jyu.fi

Mikko Vapa
University of Jyväskylä
Jyväskylä, FINLAND
mikko.vapa@jyu.fi

Heikki Kokkinen
Nokia Research Center
Helsinki, FINLAND
heikki.kokkinen@nokia.com

Jukka K. Nurminen
Nokia Research Center
Helsinki, FINLAND
jukka.k.nurminen@nokia.com

***Abstract*- During the last years progress in web search engines has been made to the point that relevant information can be reached easily most of the times. However very little empirical research has been carried to study web search in highly dynamic social network environments composed of mobile devices. The aim of this work was therefore to investigate novel approaches that took advantage of the social network environment inherent to mobile peer-to-peer paradigm. The work focused mainly on the development of a prototype for Mobile Search concept. The prototype was built on top of Drupal content site management system. This study suggests that the methods presented can be a complement to traditional web search engines.**

I. Introduction

Mobile phones' computational power has been improving approaching the capabilities of general purpose computers. Nowadays it is possible to host a web site on a mobile device. It is also expected that the number of mobile web sites will outnumber the static web servers [17].

Mobile phones possess an extra set of concerns that are not present in normal web servers (e.g. Personalization; Interactivity; Location and context dependence; Dynamicity) [17]. Those concerns can be further expanded by taking into consideration the social network formed by the contacts in the address book. This fact introduces paradigm shifts in relation to the Peer-to-Peer web search paradigm and the traditional centralized search approach.

Recently, there has been a growing interest in how to explore the mobile phone capabilities in the web search context and how to merge them with existing phone functionalities [15,17]. However the research has tended to focus on centralized approaches or Peer-to-Peer web search, rather than on the Peer-to-Peer web search in the social network context. The purpose of this article is to present different strategies that take advantage of the described type of an environment and extend the current web search mechanisms giving the end user new possibilities of exploring information.

In the future it will be common to have a web server running in mobiles devices. This represents a shift in normal web servers' webware. The biggest change is the possibility of users to freely manage their own content without being restricted by third parties. There is a need to categorize content in different ways in order to create new forms of navigation and search.

The content in mobile phones can be divided in two distinct logical groups: dynamic and static. Dynamic content usually is unique and generated by the mobile phone sensors. Static content on the other hand is not context dependent and is generated by the user. Both types of content can be easily replicated. Usually dynamic content can be easily characterized by tags, although static content can be categorized in a similar way. Content is distributed to overlapping data islands. Each user may belong to several data islands simultaneously because each user is connected to users who belong to different interest groups (even unknowingly) [5]. The connections are created based on the address book contacts forming presumably a power law graph [5]. It's assumed that the nearest neighbors of a node have higher probability to own relevant content to that node. In the information searching context it is important to have an ability to search through relevant data and take advantage of the overall network topology.

The article is structured as follows. The motivation behind the need for Mobile Search is presented in section II. Section III continues with the core concerns and major differences between this type of search and traditional centralized web search. Subsequently in section IV a brief description of the prototype is given and the related work within the topic is reviewed in section V. Finally, section VI describes the future work and section VII concludes the paper.

II. Mobile Search

This section describes a system for Mobile Search. The system is based on pure Peer-to-Peer architecture and it offers scalability, efficiency, resilience to failures and privacy at a higher degree than current centralized solutions. [4]

To take advantage of the portrayed scenario a new set of concepts were introduced. One is how to navigate through the data in a social network. Social network's connections are determined from an address book of a mobile device. Users search one graph level of their social network at a time usually starting from their neighbors. However, users may also start a query anywhere in the social network. Every time a user issues a search query the mobile device forwards it to all its neighbors. The neighbors answer back by returning a result set and a list of their neighbors. If the user who issued the query is not satisfied by the results he can always ask new results from the next level neighbors as long as there are non-visited nodes in the network. This concept was named *manual multi-hopping*. In manual multi-hopping the user needs to select which of the non-visited nodes will be used for querying the next level. Manual multi-hopping can be extended to *automatic multi-hopping* if an algorithm is used to sort which of the non-visited nodes to query further thus avoiding the need for user decision. One example of such algorithm is only to forward a query to neighbors of the nodes that previously returned results to that query. Automatically sorting the non-visited nodes leads to tradeoff between search accuracy and easiness of searching suggesting that both manual and automatic multi-hopping should be available for the user.

Another way of navigating is by searching neighbor content tags and getting the result set composed by the content links with the tags and the list of next level neighbors. Tags work as links between content categorized similarly. At each hop the user gets the list of contents tagged in a similar way by nodes in its neighborhood.

The Mobile Search system can be divided to two logical parts: local web search engine and meta crawling. Local web search engine is a search service, which manages the search index of a mobile device. Meta crawling term refers to a search service, which uses other local web search engines for getting the results and then combines different result sets into one. The part responsible for the meta crawler gets it's results from direct neighbors. The way the results are presented can always be changed thus the mobile device bears the load of processing the returned references. Any specific method to sort out the references in certain order can be employed. For example more relevance can be given to results from a certain source so they appear first in the result list. There is also the possibility to merge different types of mobile phone data with different type of content. For example user A may search for user B's meetings and after getting the results he may merge the results with his own agenda and display the meeting locations on a map.

The local web search engine gives a user the power to tailor the search results to his/her own needs. The search index can be updated every time the content changes. The user may allow certain information to be only searched by a specific group of users or to influence certain query results in a certain context. This feature allows users to create groups of trust. They can decide which information source is more relevant to them in different contexts. Also the level of privacy and who to trust is determined by each node following the motto: "I only display what I want to who I want".

III. Comparison

It may be pointed out that centralized solutions have a single point of failure, load balance and trust issues and may censor certain entities [11]. Although nowadays they have grown incredibly robust. One main advantage of Mobile Search is the total independence of the nodes. The system can operate without any central server and system load is fully distributed. Each node is responsible for processing the queries and search requests.

For example Google presents in its back end a highly scalable architecture [3] but it cannot address the premise that our friends are more likely to have interesting results to us and may not even be connected or linked to our content [8]. In this scenario the hyperlink concept is expanded by the network of connections formed by the mobile phone's address book. These types of links enable the blend of several groups of interest along the network. In several situations the link web structure of documents doesn't portray possible relations between people [10].

The search space indexed by centralized solutions is limited because central servers have limited crawling capacity. Index of a centralized solution can thus be characterized as one large result set. Also, crawling cannot easily find content without external references. In contrast, decentralized social network search consists of multiple small result sets, does not have indexing limitations and does not need external links to point out the content. Non-referenced content can be found by finding a neighbor of the owner of non-referenced content. Thus decentralized search potentially provides more results than centralized solutions when user continues navigating the social network further. However, queries executed in immediate surroundings of the querying node usually result in fewer and more accurate results than centralized solutions.

Web search engines do not allow tailoring results to individual needs. For example user A only wants to display a specific result list to a certain query from user B. Centralized solutions provide an efficient way of finding popular content but lack the ability to find more personal/social proximity content [8]. This situation is evident in a corporate setting where many documents are not available to the outside world. Other type of personal/social proximity content that is not indexed by web search engines is mobile phone data. One example is searching for a phone number or meeting information that is available in one of our neighbors. This capability avoids the use of third entities (e.g. number services, central servers) and enhances the information availability. In the other hand Mobile Search due to the topic oriented network nature is not suited to find popular content. Conversely, it's a powerful mechanism in restricted topic set environment [8].

One major issue of Mobile Search in relation to the centralized approach is the quality of the results returned. Different sites may have different criteria to classify and rank information. This poses a problem how to merge the different results sets returned for a query [12]. In the other hand, this can highly increase the quality of the results in some scenarios. For example in a work context user A can give more weight to Document X in searches made by users from the workgroup because that document is more relevant to them.

Other issue is the high number of neighbors and free riding. Those factors are a risk to network traffic. They can be overcome first by limiting the search query to a pre-selected group of users and second by only returning back neighbors who have a higher probability of having meaningful content.

Centralized solutions update their index when content is crawled whereas in Mobile Search the owner can index the searchable content whenever he/she desires. This leads to up-to-date result sets without any increase in network traffic. And as long as the user sets the permissions for different content, other users authorized to view that content can find it without knowing the exact location. With centralized solutions everyone has to trust a single entity allowing possibilities for censorship or pressure from external entities.

Concern	Centralized solutions	Mobile Search
Load	*centralized/single point of failure*	*highly distributed*
Trust	*censorship/pressure from external entities*	*highly distributed*
Search space	*billions (single set)*	*hundreds to billions (multiple different sets)*
Index update	*days to months*	*every second*
Content type	*popular*	*personal/social proximity*

IV. Drupal Prototype

Drupal was used as a test platform for Mobile Search. Drupal is an open-source content management system. It allows managing and publishing several types of content. The meta crawler described in section II was built as a weakly coupled component on top of Drupal local web search engine. This component allows automatic multi-hopping and result interleaving.

The current implementation is single threaded because Mobile Apache doesn't support multiple threads [15,16]. Drupal tac_lite module and Drupal module were also used as fundamental elements in the prototype. These modules allow setting content access rules and to process user authentication in distributed fashion without any central servers.

An extra component that allows to do queries to local mobile phone content such as location, address book and meeting data was implemented. This feature was built as a proof of concept. However, the prototype is also able to gather search results from unmodified Drupal web sites.

One drawback during the elaboration of the prototype is related to the single-threaded nature of the meta crawler. This can have a negative impact on response time because site crawling is done in a serial way. A multi-thread implementation would speed up the system considerably.

V. Related Work

The concept of Peer-to-Peer web search has been harnessed before in the literature. Different approaches [2,8,13,14,20] have been tried before. Although these studies tended to focus on Peer-to-Peer web search, rather less attention has been paid to how to take advantage in this scenario of mobile sites' concerns and integration in the social network context.

Mislove et al. [8] studied how to integrate social network search with web search in order to complement search results. Also, how content publishing and locating influence the overall searching experience in the web perspective and in the social network context is discussed. Supported by the experiment made with PeerSpective prototype, [8] points out flaws in the traditional hyperlinked search like the difficulty of web search engines to index content not well linked to the general web or that is not publicly available. Similar to Mobile Search, [8] presents the idea that social networks, due to data islands formed by user communities, can lead to more timely and efficient searching experience.

Like in our work, [8] gives special importance to social network links but leaves as an open topic how the underlying social network links are formed. In Mobile Search social network structure is automatically defined by the mobile phone address book contacts and can be enhanced by linking content neighbor tags every time a search is performed. Ultimately, the Mobile Search presents the possibility of creating a virtual multi-level content social network. The mechanisms described in [8] could also be adapted and incorporated into Mobile Search.

Bawa et al. [2] introduce YouSearch, which allows searching dynamically changing content from personal web servers. YouSearch differs from Mobile Search approach by having a centralized server (registrar) for storing bloom filters of indexed keywords. This introduces a need to update bloom filters periodically to accommodate changes in content of the peers. Mobile Search is designed for mobile devices with a limited battery and therefore periodically occurring updates needs to be avoided. According to the calculations in [2] one registrar could serve approximately 10000 peers with a 1,5 Mbps network connection. In Mobile Search such an entity is not needed, because all functionalities are decentralized. YouSearch uses caching for storing search results on a querying peer to avoid re-executing a similar query later. This is a feature which could also be applied in Mobile Search.

Finally, YouSearch does not take into account social network connections and therefore searching needs to be explicitly directed to different groups or to specific registrar. This reduces the flexibility of searching.

Zhou et al. [20] states that the evaluation of resources by human users is more important for search quality than the traditional machine based approach. They present a novel page ranking algorithm - Peer-Rank. In this paper a simpler version to rank remote results is presented. First of all, in the problem context described in this study it's assumed that the content on the mobile phone can be divided in two sub-types: dynamic/unique (photos taken with mobile phone camera) and static/common (music files). It will be rare to have different sites returning the same content. Secondly, it's also supposed that the majority of the content will be dynamic/unique due to the nature of mobile phone. Furthermore, each mobile site can employ its own human/machine based methods to rank results. With these details in mind two ways of ranking the results are proposed: Explicit (Tagging content) and Implicit (Machine based methods).

Galanx [14] focuses on query forwarding in Peer-to-Peer web search context. Traditionally Peer-to-Peer web search studies try to "emulate" the behavior of centralized solutions. Those approaches are completely orthogonal to the one presented in this paper. One of the main concepts derived from the social network environment is the ability to navigate through neighbor sites and explore them like in a common social network site where users are able to follow friends' links and explore them. In this case links are created based on the search results. If users are not satisfied with the results they can always jump to the next set of nodes and continue searching. In the Galanx case, like in a centralized web search, only a set of results is provided and the users are unable to explore the network by themselves. The sites are presented as fully separated entities, although they can have hyperlinks between them allowing partial network navigation.

The query forwarding mechanism of Mobile Search can be described as a directed breadth first search with manual iterative deepening. The algorithm is similar to the one described in [19] and [6] with the exception of using manual iterative deepening. A search is only continued if the user is not satisfied with the results.

Other major source of inspiration was the social network tagging system. Similarly the same principle was applied to the system with minor modifications. Users are able to tag content freely. Some predefined tags related with mobile phone concerns will be always available (e.g., photo location). Generally user tags have only a local significance in the network [9]. The predefined tags try to create general tags present all over the network enhancing the navigation. Each time a user in a site can search for neighbor tags and navigate through them like in the normal web search presented in this report.

VI. Future Work

The concept of Mobile Search can be easily expanded and integrated as an extension to existing systems.

Query forwarding algorithms should be considered in order to minimize several problems like free riding [1] though in a different setting than previous studies. Algorithms like Ant search [18], K-Random walk, Expanding Ring and hybrid approaches should be considered.

Other way of extending the Mobile Search functionalities is by creating different ways of accessing the same content. Information could be accessed by a search result or by different entry point. An entry point is a link to a specific

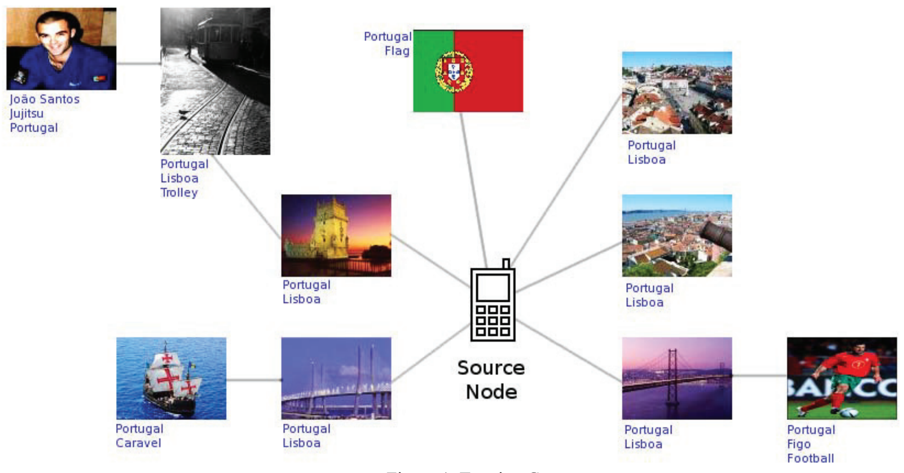


Figure 1. Tagging Concept

content. Tags are an example of creation of different entry points. A different way of creating an entry point is by merging different types of data.

Figure 1 shows an example of the tagging concept. The mobile phone represents the source node who issued a query searching for the tag Portugal. The figure represents the results returned by the neighbor nodes in different network levels (each image corresponds to a neighbor who returned a result).

For example if the source node issued the query Portugal it would obtain six results. If then the user chooses to navigate by the tag Lisboa he would get one result (the trolley image). If instead the user chooses the keyword Portugal he would get three results (the trolley, the caravel and Figo).

Mobile Search enables the creation of multi-social network fusion. With the Mobile Search the user doesn't need to know exactly where the different entry points are. The returned results will allow exploring vicinities following the links of the different tags or by asking for new results. The same user may present in its own site several data related to it's own interests. Certain data may only be available to a specific group of users. The data also may be presented in different ways for different groups. These features could be particularly valuable in an enterprise setting. One example would be a fully distributed enterprise portal [10] using the technology described in this paper.

Other feature worth exploring is adaptive ranking. Historical behavior of users who conducted similar searches or may have a similar role in an organization may be used to boost document rating. This concept may be expanded if more data is available by creating a profile to generate suggestions for documents based on user context and role in that particular social network [10].

All those features can be tweaked at different granularity for different group of users that access the system. For example a user may only generate profiles of work mates in order to make suggestions.

Other topic of interest is the usability of search results, and new paradigms of displaying different types of information and user interaction. Current Web2.0 may not be fully suitable for mobile device paradigm of interaction. This could also be an excellent opportunity to use a query language applied to this type of systems for example an adaptation of webSQL [7]. This would likely create a bigger interoperability and homogenization in this type of systems with easier deployment of new functionalities.

VII. Conclusions

Mobile Search complements traditional web search engines. It gives the user means to explore the neighbors' contents by traveling to the friends network topology. It covers a multitude of environments not covered by the centralized solutions.

One of the main advantages in relation to current centralized social network sites is the possibility to manage the site without interference from an external entity. Currently in a normal social network site a user can only display or use modules made available by a third entity. Due to this characteristic it is possible to merge different network sites that cover different topics and create a social network "melting pot". Each user can have what type of content he/she wishes in the site and display different content for different users.

This type of system is better suited for mobile devices due to the "always on" characteristic [18]. Content can be always updated on spot.

Mobile Search has an enormous potential to evolve and become a major tool in knowledge management technology. Adaptive Ranking, Role-based Recommendations, Locating Experts and Communities [10] can be taken to extreme. To sum up Mobile Search can be used to enhance the ability to search for critical information.

References

[1] E. Adar and B. Huberman. Free riding on gnutella. Technical report, Xerox PARC, 2000.

[2] M. Bawa, R. J. Bayardo Jr., S. Rajagopalan, and E. J. Shekita. Make it Fresh, Make it Quick – Searching a Network of Personal Webservers. Proceedings of the 12th International Conference on World Wide Web, ACM Press, pages 577-586, 2003.

[3] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. Computer Networks and ISDN Systems, 30(1–7):107–117, 1998.

[4] D. Choon-Hoong, S. Nutanong, and R. Buyya, Peer-to-Peer Networks for Content Sharing, Peer-to-Peer Computing: Evolution of a Disruptive Technology, pages 28-65, Idea Group Inc., Hershey, PA, USA, 2005.

[5] E. F. Churchill and C. A. Halverson. Social networks and social networking. IEEE Internet Computing, 9(5):14–19, 2005.

[6] X. Li and J. Wu. Searching techniques in peer-to-peer networks. Handbook of Theoretical and Algorithmic Aspects of Ad Hoc, Sensor, and Peer-to-Peer Networks. Auerbach Publications, pages 613 – 642, 2006.

[7] A. O. Mendelzon, G. A. Mihaila, and T. Milo. Querying the world wide web. Int. J. on Digital Libraries, 1(1):54–67, 1997.

[8] A. Mislove, K. P. Gummadi, and P. Druschel. Exploiting social networks for internet search. In Proceedings of the 5th Workshop on Hot Topics in Networks, Irvine, CA, 2006.

[9] J. C. Paolillo and S. Penumarthy. The social structure of tagging internet video on del.icio.us. In Proceedings of the 40th Annual Hawaii International Conference on System Sciences, page 85, Washington, DC, USA, 2007. IEEE Computer Society.

[10] P. Raghavan. Social networks from the web to enterprise. IEEE Internet Computing, 6(1), 2002.

[11] P. Reynolds and A. Vahdat. Efficient peer-to-peer keyword searching. In Proceedings of International Middleware Conference, pages 21–40, June 2003.

[12] E. Selberg and O. Etzioni. The metacrawler architecture for resource aggregation on the web. IEEE Expert, 12(1):11-14, 1997.

[13] H. Tomiyasu, T. Maekawa, T. Hara, and S. Nishio. Social network applications using cellular phones with e-mail function. In Proceedings of 21st International Conference on Data Engineering. IEEE Computer Society, 2005.

[14] Y. Wang, L. Galanis, and D. J. DeWitt. Galanx: An efficient peer-to-peer search engine. Technical report, University of Wisconsin - Madison, 2003.

[15] J. Wikman. Mobile web server - eurooscon presentation, 2006.

[16] J. Wikman and Ferenc Dosa. Providing http access to web servers running on mobile phones. Technical report, Nokia Research Center, May 2006.

[17] J. Wikman, Ferenc Dosa, and Mikko Tarkiainen. Personal website on a mobile phone. Technical report, Nokia Research Center, 2006.

[18] C.-J. Wu, K.-H. Yang, and J.-M. Ho. Antsearch: An ant search algorithm in unstructured peer-to-peer networks. In Proceedings of the 11th IEEE Symposium on Computers and Communications, pages 429–434, Washington, DC, USA, 2006. IEEE Computer Society.

[19] B. Yang and H. Garcia-Molina. Improving search in peer-to-peer networks. In Proceedings of 22nd International Conference on Distributed Computing Systems, Vienna, Austria, 2002.

[20] J. Zhou, K. Li, and L. Tang. Towards a fully distributed p2p web search engine. In Distributed Computing Systems, 2004. FTDCS 2004. Proceedings. 10th IEEE International Workshop on Future Trends of, pages 332–338. IEEE Computer Society, 2004.

PIII

FINDR: LOW COST INDOOR POSITIONING USING FM RADIO

by

Andrei Papliatseyeu, Niko Kotilainen, Oscar Mayora and Venet Osmani 2009

In Mobile Wireless Middleware, Operating Systems, and Applications, volume 7 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pages 15-26

FINDR: Low-Cost Indoor Positioning Using FM Radio

Andrei Papliatseyeu[1], Niko Kotilainen[2], Oscar Mayora[3], and Venet Osmani[3]

[1] University of Trento, Via Sommarive 14, Povo (TN), 38050, Italy
papliats@disi.unitn.it
[2] University of Jyväskylä, P.O. Box 35, 40014 Jyväskylä, Finland
niko.kotilainen@jyu.fi
[3] Create-Net, Via alla Cascata 56/D, Povo (TN), 38050, Italy
oscar.mayora@create-net.org, venet.osmani@create-net.org

Abstract. This paper presents an indoor positioning system based on FM radio. The system is built upon commercially available, short-range FM transmitters. The features of the FM radio which make it distinct from other localisation technologies are discussed. Despite the low cost and off-the-shelf components, the performance of the FM positioning is comparable to that of other positioning technologies (such as Wi-Fi). From our experiments, the median accuracy of the system is around 1.3 m and in 95% of cases the error is below 4.5 m.

Keywords: Indoors positioning, FM radio, location awareness.

1 Introduction

Location awareness is an important requirement for many modern applications, spanning from mobile maps and geotagging to Internet of Things and healthcare. The Global Positioning System (GPS) is most widely used for location sensing, but it is limited to outdoors-only applications. A body of research has addressed indoor positioning using different technologies, like ultrasound and infrared beacons, Wi-Fi and GSM networks, or other types of radios [1]. Most of these systems are limited in terms of expensive/custom hardware, laborious deployment or low accuracy.

Our paper explores the applicability of short-range FM radio transmitters for indoor positioning. We have installed our FINDR (FM INDooR) positioning system in our lab and this paper presents performance evaluation results of the system as well as an overview of particular properties of FM radio with respect to localisation.

FM has a number of advantages over other positioning technologies, like Wi-Fi. Firstly, although Wi-Fi infrastructure is readily available in office buildings, the installation of a localisation system in domestic environment requires additional hardware. In this case, FM is a cheaper alternative to the deployment of multiple Wi-Fi access points per apartment. FM transmitters are cheaply available from many consumer electronics shops; the client device can be represented by a PDA or a cellphone with an embedded FM receiver. Secondly, FM radio can be safely used in sensitive environments, e.g. hospitals, whereas GSM, Wi-Fi or Bluetooth devices must be switched off there. Finally, FM is very power-effective: an average FM receiver consumes about 15 mW, compared to almost 300 mW of Wi-Fi (in receiving mode) [2, 3].

The paper is organized as follows. Section 2 provides and overview of the related work. Section 3 then introduces our approach and our experimental testbed. Section 4 presents results pertaining to performance evaluation of FINDR, while Section 5 describes the possible application scenarios of the system. Finally, Section 6 draws the conclusions and outlines the future work.

2 Related Work

2.1 Wireless Positioning Techniques

In the last decade, a large body of research has been dedicated to the development of location-aware systems. Indoors positioning systems rely on several types of sensors: ultrasound [4, 5], infrared (IR) [5, 6], digital compass [4], RFID [7], and various kinds of radio: Wi-Fi [8, 9], GSM [10], Bluetooth [11], domestic powerline [12, 13], and others [14, 15]. Such systems usually rely on one or a number of the following criteria: user proximity to some fixed beacons, time of signal propagation, and received signal strength. In the sections that follow we briefly describe each of these approaches to localisation.

Proximity-Based. Given an environment with a number of beacons with known positions, the algorithm assumes that the user's position is that of the nearest beacon. Due to its simplicity, the method is widely adapted by the systems using custom radio beacons [15], as well as Bluetooth [16], IR [5] and GSM base stations [17, 18]. Unfortunately, the accuracy of such systems is low and depends on the density and the number of installed beacons.

Time-Based. Time-based methods use information about signal propagation time between the mobile device and beacons with known positions, in order to estimate the position of the mobile user. The most prominent example of this class of methods is GPS. Using the signals from a set of GPS satellites, a basic GPS receiver is able to compute its position with the accuracy of about 8 m [19, p. 22]. However, GPS has long start-up times (up to a few minutes) and does not work indoors and in dense urban areas, which limits GPS's applicability for ubiquitous location-based services. Ultrasonic localisation systems, like Cricket [4], also rely on the travel time of an ultrasound pulse. While providing a good accuracy, time-based systems usually require custom hardware and expensive installation.

Signal Strength-Based. There are two general positioning approaches that use Received Signal Strength Indication (RSSI), namely propagation modelling and fingerprinting. The first approach attempts to build a model of the signal propagation in the space in order to identify the distance between the user and beacons. The fingerprinting approach, in turn, relies on a database associating RSSI measurements with corresponding coordinates and then uses statistics and machine learning algorithms in order to recognize user position among those learned during the training phase. RSSI-based methods are the most powerful, as they can provide a rather high accuracy with a few beacons.

One of the pioneering projects in RSSI-based positioning was RADAR [20]. The authors applied both propagation modelling and fingerprinting within a Wi-Fi network, and, with some enhancements, the system error was as low as 2 m [8]. With more advanced probabilistic methods, the median error of a Wi-Fi based system can reach 1.2–1.45 m [9, 21]. RSSI fingerprinting has also been successfully applied for indoor localisation using GSM base stations. In [10], the authors employed so-called wide fingerprints, which included RSSIs of up to 35 GSM channels, and thus managed to achieve a Wi-Fi-like median positioning accuracy. However, the topology of a GSM network can be changed at any time by the network operator, thus requiring system recalibration. [12] proposed a more reliable approach for indoors positioning. In their system, two beacons were injecting high-frequency signals into domestic powerline. These signals could then be detected by a specialised receiver and associated with the user's location. An extended, wideband version of the system achieved a 90% accurate room recognition [13]. Despite the easy installation, the system requires specialised hardware with limited availability.

To the best of our knowledge, there is only one work dedicated to positioning with FM radio. [14] described their experiments on using prototype hand watch with an embedded FM radio, to localise using commercial FM broadcasting stations. The authors applied a Bayesian classifier to distinguish six areas of Seattle, based on RSSI ranking of the local FM stations. In the best case, the recognition accuracy was 82%. Although the paper does not provide any information about error distances, the system accuracy can be estimated as hundreds of meters to kilometers, which renders it impracticable for indoor environments. Our system, instead, is based on readily available hardware and is particularly suitable for indoor use.

3 FM Positioning

3.1 Our Approach

The FINDR positioning system employs a set of short-range FM transmitters as wireless beacons and a programmable radio on the client device. Most of the beacon-based positioning technologies have two general requirements: measuring of user to beacon relative position and the ability to distinguish different beacons. In the next two sections we identify possible solutions how FM radio can address these requirements.

Relative Position-Dependent Features. The relative position of the user with regard to a beacon can be characterised by angle between directed antennas, signal propagation time and RSSI. For the FM positioning, we have identified three features that can be used as a measure of distance between the beacons and the user.

The first feature is RSSI, defined as the amplitude of the received radio-frequency signal. Most of the current FM receivers employ RSSI value internally, to enable auto-tuning capability.

When RSSI is not available, one can use the signal-to-noise ratio (SNR) of the demodulated signal. In this case, the beacon is set to transmit a known periodic signal (for example, a sine wave of 1kHz) and the receiver performs a fast Fourier transform (FFT) of the demodulated signal, calculating the intensities of different frequency bands. Then, the intensity of the band of interest is divided by the average intensity of the all bands, thus representing signal-to-noise ratio. A similar method was applied by [12] to an amplitude-modulated (AM) signal. However, our experiments show that SNR of an FM signal is almost a step function, which considerably limits applicability of this approach to FM-based positioning (see Section 4.1).

There is also another feature that depends on the signal quality and, consequently, on the distance between the transmitter and the receiver, namely, stereo channels separation. In good reception conditions the stereo channels are well separated, providing best sound quality. However, as the radio signal deteriorates, the receiver's circuitry will start to reduce the audio bandwidth and thus decrease channel separation in order to filter out the noise [22]. Ultimately, this results in a plain mono signal.

Distinguishing Beacons. For a beacon-based positioning system it is crucial to distinguish current beacon from the others. The beacons can be identified either by their carrier frequencies or by the signals they transmit (e.g. coordinates, ID, name, etc).

Unfortunately, due to the properties of FM, it is impossible to use the same frequency for all beacons. Due to the so-called "capture effect", when a number of stations transmit on the same (or close by) frequency, the signal from the strongest one will dominate the others, while the weaker signals get attenuated [23]. Therefore, in our experiments we had to tune each transmitter to a different frequency and switch between them at the receiver side. Despite this, no special network planning is required for larger-scale deployments to avoid beacons interference, as any distant interfering beacons will not be observed due to the capture effect.

3.2 Experimental Setup

The FINDR was evaluated with empirical measurements in the Multimedia, Interaction and Smart Environments (MISE) lab of Create-Net [24]. The room dimensions were 12 x 6 m, and the room contained ordinary office furnishing.

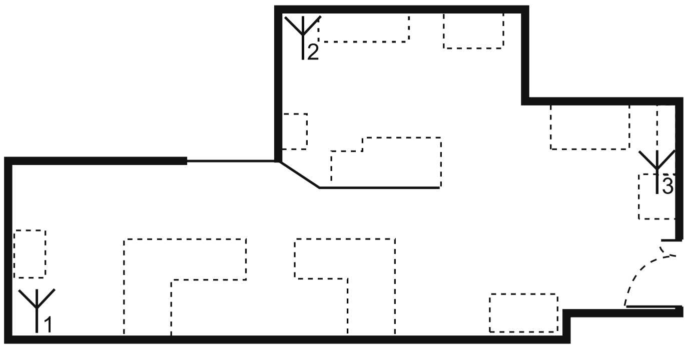


Fig. 1. Floorplan of the measurement area. The antennas mark the positions of the three transmitters and the dashed lines mark room furniture.

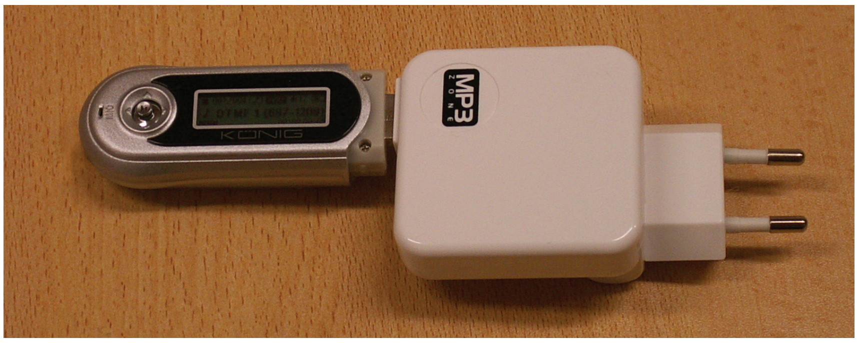


Fig. 2. MP3 player with an embedded FM transmitter, connected to power adapter. The antenna is not connected.

Figure 1 presents the layout of the room. A grid of 1 x 1 m cells was created for testing, and measurements were carried out in all accessible points of the grid (totally 46 points).

The receiving device used in the tests was a Nokia N800 Internet Tablet. The N800 is an based on an ARM processor and features a built-in FM receiver. The N800 is running an open, Linux-based operating system, so developing low-level custom applications for the device is relatively easy. The prototype locating software was programmed in Python and used the PyFMRadio-library to tune the FM-receiver to each of the transmitter's frequency one after another and read the signal strength from the FM-receiver hardware. The signal strength was reported on a 16-step scale (normalized to range 0. . . 1) and was measured 300 times in a row for each frequency, with about 0.01 second between the measurements. The standard N800 headset was used as an antenna.

The transmitter used was a König mp3 player, which features a built-in FM-transmitter (Figure 2) [25]. To increase the range of the transmitters, a 1.8-meter audio cable was connected to the player's audio output to act as an antenna. Initially, the whole FM band was scanned and manually checked for frequencies with little interference from local FM-radio stations. The transmitters were then tuned to these frequencies. To avoid the effect of battery degradation, the transmitters were powered by USB power adapters.

4 Results

4.1 RSSI Dependency on Distance

In order to estimate the feasibility of the FM positioning, we first carried out a test to see which of the features discussed in Section 3.1 are more suitable for positioning. Stereo channel separation method has not been implemented yet and will be addressed in the future work.

The RSSI dependence on the distance from the transmitter is presented in Figure 3. To avoid any interference from the testbed's furniture, this test was performed outdoors. The graph is relatively smooth and monotone starting from 0.5 m, and proves RSSI to be a good feature for positioning. Eventual plateau-looking areas can be explained by the limited number of RSSI levels recognized by our receiver.

Figure 4 corresponds to the indoors measurements and shows the RSSI from each of three transmitters while the user was moving from Transmitter 1 to Transmitter 3 (as of floorplan in Figure 1). The dependencies are not very smooth, which is caused by the distortions from the furniture and multipath propagation. Nevertheless, the general trends are clearly observable.

For the RSSI$_{SNR}$ method, the transmitter was set to broadcast a continuous dual tone multi-frequency (DTMF) signal for digit "1" (1209 Hz and 697 Hz).

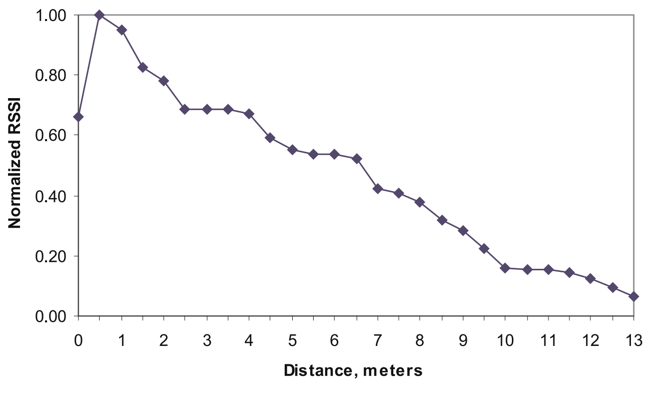


Fig. 3. RSSI dependence on distance

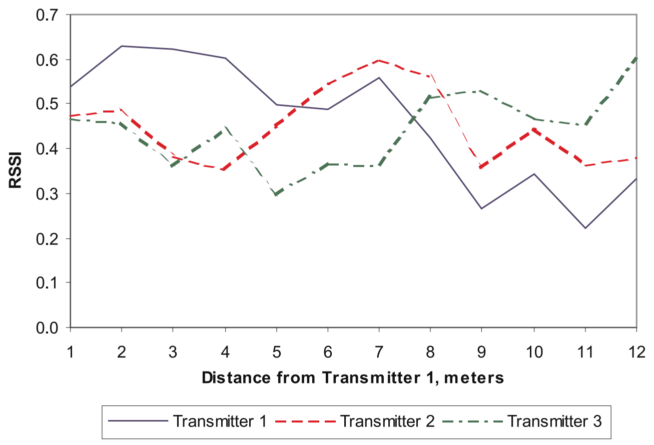


Fig. 4. RSSI variation while moving from Transmitter 1 to Transmitter 3, with Transmitter 2 placed between them

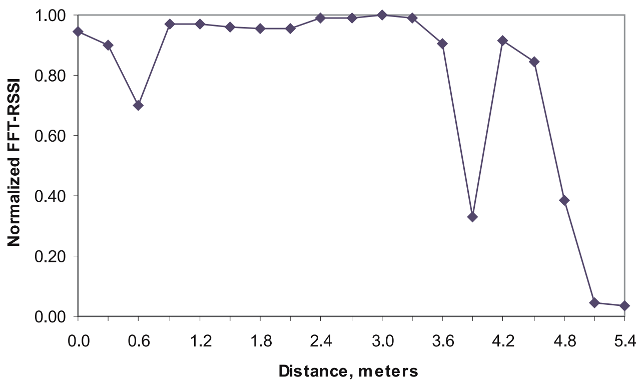


Fig. 5. RSSI$_{SNR}$ dependence on distance

At the receiver side, the audio signal from an FM radio was sampled by a laptop sound card at 8 kHz sampling frequency and transformed to the frequency domain using 1024-band FFT. For each point, 32 spectra were recorded and then averaged. RSSI$_{SNR}$ was then calculated as follows:

$$RSSI_{SNR} = \frac{band_{697Hz} + band_{1209Hz}}{mean(all\_bands)}$$

The experiment discovered no clear dependency of RSSI$_{SNR}$ from the distance to the transmitter (see Figure 5). In range from 0.5 m to 3.6 m the mean RSSI$_{SNR}$ value barely changed, between 3.6 m and 4.5 m it became unstable, and then rapidly degraded to the noise level. Such a behaviour can be explained by the capture effect, which improves the post-detection SNR for non-linear modulations (such as FM) when the pre-detection SNR is above a certain threshold, "capture threshold"; below this threshold the SNR drops dramatically [26]. In our case, the capture effect is complemented by the receiver noise-reduction circuitry which automatically mutes the audio output if the received signal is too weak [2].

Thus, RSSI$_{SNR}$ dependency on the distance is almost a step function due to intrinsic properties of FM. Therefore, we did not consider RSSI$_{SNR}$ for further experiments.

4.2 2D Positioning

To estimate the FINDR accuracy in two-dimensional positioning, we have used fingerprinting approach with two evaluation methods: leave-one-out validation and an independent test set. In leave-one-out method, we sequentially selected one of the RSSI measurements and excluded all the measurements related to the same coordinates from the training set. The selected measurement was then used as test data. It should be noted however, that leave-one-out evaluation tends to

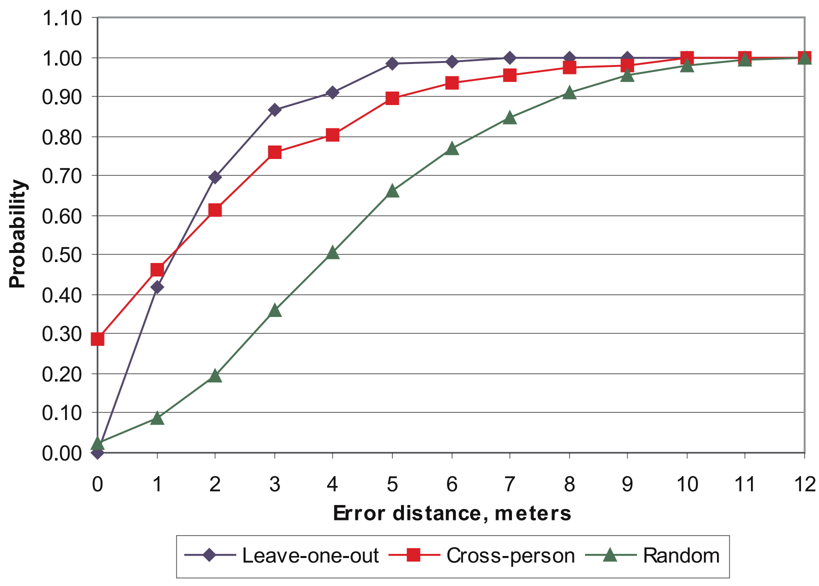


Fig. 6. Error distributions for two-dimensional positioning

worsen the actual positioning accuracy, as the classifier is unable to recognize the class it has not been trained on (that is, the error distance is always greater than zero) [20]. Besides that, in order to estimate the real-world system accuracy, we have tested the FINDR on an independent data set collected by another person.

For classification, a k-nearest neighbour (kNN) method was used [27]. The kNN classifier evaluates the distance from the test point to all the training points, and selects the labels (classes) of the k nearest training points. From these k labels, the prevailing one is returned as the classification result. For our task, we employed the Euclidean distance measure. The optimal value of k ($k = 9$) was selected by leave-one-out validation and then reused for cross-person evaluation.

The error distance distributions for both approaches are shown in Figure 6. The baseline performance is represented by a random classifier. The median accuracy for the leave-one-out evaluation method is 1.3 m, falling to about 4.5 m at 95% confidence level. For the independent test set, 29% of places are recognized correctly. The median accuracy is 1.3 m. Despite the long tail, caused by distant outliers, in 95% of the cases the positioning error stays below 6.8 m.

4.3 RSSI Stability over Time

For a fingerprinting-based system, it is very important that the values measured during calibration phase do not drift over time. Otherwise, the system accuracy may diminish significantly, and the system will require recalibration. It has been demonstrated, that many current fingerprinting-based systems are affected by the signal stability problems [13, 28].

In order to estimate the stability of the FM signal strength in FINDR, we placed a transmitter 4 meters apart from the receiver and left it recording the RSSI over the weekend. However, in four hours the device ran out of memory and only 1.7 million samples have been recorded. Their mean value was 0.57975 and the variance was 0.00097.

The RSSI distribution in Figure 7 proves the FM RSSI to be rather stable. The two peaks are different by one quantization step only. There are about 4000 outliers, which constitute only about 30 seconds of the whole 4-hour dataset. Note that the measurements have been done by a receiver that distinguishes only 16 RSSI levels; a more advanced receiver could improve both the distribution detail and the positioning accuracy.

5 Application Scenarios

The need for finding one's position has sprung up a number of technologies that fulfil this purpose with varying degrees of success. While outdoor positioning is a relatively mature technology (i.e. GPS), the indoor localisation has been proven an interesting research challenge. The interest in indoor positioning has been fuelled by the potential it offers in creating novel applications that can span across diverse domains. Applications ranging from locating lost keys within home, up

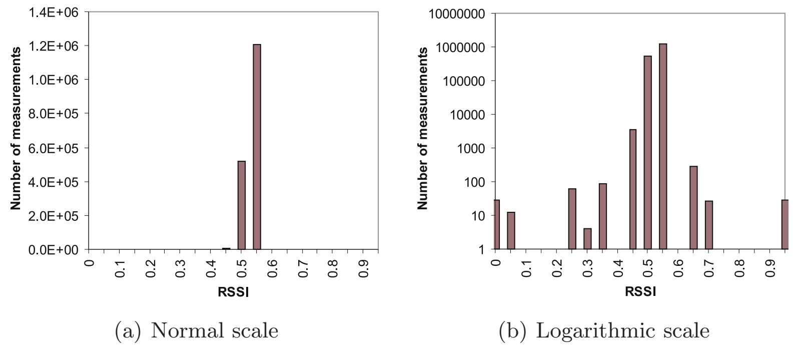


Fig. 7. RSSI distribution for 4-hour long measurements

to detecting mobility patterns of elderly that aid disease diagnosis, are made possible by utilising technologies that offer relatively precise location information, while considering the cost benefits. FM localisation method, described thus far, is such technology that can give rise to a number of interesting applications.

Applications that make use of localisation can be found in the realm of social sciences, amongst other domains. Localisation can be utilised to infer mobility patterns of users. A study, described in [29], tracked location of 100.000 mobile phones. Analysis of the data revealed that users have predictable mobility behaviour patterns, which authors were able to infer by analysing only half of the data collected. However, this study was limited since location data was based on GSM localisation, thus had a low granularity, typical of a GSM cell tower range.

FM localisation will allow analysis of data that has much higher localisation granularity, by simply utilising a mobile phone with built-in FM receiver. This information then can be used, not only to infer mobility patterns, but by using the concept of group location, the social network of a user can also be deduced. In other words FM localisation method will allow inference of human relationships, for example colleagues that spent time in the same office, through analysis of sub-room mobility patterns.

Naturally, localisation technology is applicable to a number of other domains, including health care, where it can be used to aid elderly locate misplaced objects (such as their mobile phone), or even deliver location dependent reminders - locking the front door when entering the house for instance. These applications can be enabled by a low-cost, sub room location solution, which FM positioning is able to provide.

6 Conclusion

This paper presented the FINDR, an indoor positioning system based on FM radio technology. The system is a low-cost solution that does not require any specialised hardware, thus is easily deployable. FM transmitters, used as beacons, are easily available in the most of electronics shops. Virtually any cellphone or PDA, with an embedded FM tuner can be used as a client device. FM receiver is by an order of magnitude more power-efficient than Wi-Fi. The preliminary results of the system evaluation show a median accuracy of about 1.3 m and 4.5 m at 95% confidence level that is favourably comparable to other state-of-the-art positioning systems.

In the future we plan to conduct a more comprehensive evaluation of FINDR using probabilistic classifiers and perform a same-environment comparison with other positioning systems. These results will be applied to a number of previously described application domains.

References

[1] Hightower, J., Borriello, G.: Location systems for ubiquitous computing. Computer 34(8), 57–66 (2001)

[2] TDA7088 Datasheet (1996)

[3] Broadcomm BCM4326 Single-Chip IEEE 802.11b/g MAC/baseband/radio Datasheet (2006)

[4] Priyantha, N.B., Miu, A.K.L., Balakrishnan, H., Teller, S.: The cricket compass for context-aware mobile applications. In: Proceedings of the 7th Annual International Conference on Mobile Computing and Networking, pp. 1–14. ACM Press, New York (2001)

[5] Fox, D., Hightower, J., Liao, L., Schulz, D., Borriello, G.: Bayesian Filtering for Location Estimation. IEEE Pervasive Computing (2003)

[6] Liao, L., Fox, D., Hightower, J., Kautz, H., Schulz, D.: Voronoi Tracking: Location Estimation Using Sparse and Noisy Sensor Data. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (2003)

[7] Hightower, J., Borriello, G.: Particle Filters for Location Estimation in Ubiquitous Computing: A Case Study. In: Davies, N., Mynatt, E.D., Siio, I. (eds.) UbiComp 2004. LNCS, vol. 3205, pp. 88–106. Springer, Heidelberg (2004)

[8] Bahl, P., Padmanabhan, V.N., Balachandran, A.: Enhancements to the RADAR User Location and Tracking System. Technical Report MSR-TR-2000-12, Microsoft Research, Redmond, WA, USA (2000)

[9] Youssef, M.A., Agrawala, A., Shankar, A.U.: WLAN location determination via clustering and probability distributions. In: Proceedings of the 1st IEEE International Conference on Pervasive Computing and Communications (PerCom), pp. 143–150 (2003)

[10] Varshavsky, A., de Lara, E., Hightower, J., LaMarca, A., Otsason, V.: GSM indoor localization. Pervasive and Mobile Computing 3(6), 698–720 (2007)

[11] Kotanen, A., Hännikäinen, M., Leppäkoski, H., Hämäläinen, T.D.: Experiments on local positioning with Bluetooth. In: Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC 2003), pp. 297–303 (2003)

[12] Patel, S.N., Truong, K.N., Abowd, G.D.: PowerLine Positioning: A Practical Sub-Room-Level Indoor Location System for Domestic Use. In: Dourish, P., Friday, A. (eds.) UbiComp 2006. LNCS, vol. 4206, pp. 441–458. Springer, Heidelberg (2006)

[13] Stuntebeck, E.P., Patel, S.N., Robertson, T., Reynolds, M.S., Abowd, G.D.: Wideband powerline positioning for indoor localization. In: Proceedings of Ubicomp 2008 (2008)

[14] Krumm, J., Cermak, G., Horvitz, E.: RightSPOT: A Novel Sense of Location for a Smart Personal Object. In: Dey, A.K., Schmidt, A., McCarthy, J.F. (eds.) UbiComp 2003. LNCS, vol. 2864, pp. 36–43. Springer, Heidelberg (2003)

[15] Bulusu, N., Heidemann, J., Estrin, D.: GPS-less low-cost outdoor localization for very small devices. Personal Communications, IEEE 7(5), 28–34 (2000)

[16] Hallberg, J., Nilsson, M., Synnes, K.: Positioning with Bluetooth. In: Proceedings of 10th International Conference on Telecommunications (ICT 2003), 23 February–1 March 2003, vol. 2, pp. 954–958 (2003) doi:10.1109/ICTEL.2003.1191568

[17] LaMarca, A., Chawathe, Y., Consolvo, S., Hightower, J., Smith, I., Scott, J., Sohn, T., Howard, J., Hughes, J., Potter, F., Tabert, J., Poweldge, P., Borriello, G., Schilit, B.: Place Lab: Device Positioning Using Radio Beacons in the Wild. In: Gellersen, H.-W., Want, R., Schmidt, A. (eds.) PERVASIVE 2005. LNCS, vol. 3468, pp. 116–133. Springer, Heidelberg (2005)

[18] Laasonen, K., Raento, M., Toivonen, H.: Adaptive On-Device Location Recognition. In: Ferscha, A., Mattern, F. (eds.) PERVASIVE 2004. LNCS, vol. 3001, pp. 287–304. Springer, Heidelberg (2004)

[19] US Department of Defence. GPS SPS Performance Standard. (September 2008), `http://www.navcen.uscg.gov/gps/geninfo/2008SPSPerformanceStandardFINAL.pdf`

[20] Bahl, P., Padmanabhan, V.N.: RADAR: an in-building RF-based user location and tracking system. In: Proceedings of 9th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2000), March 26-30, 2000, vol. 2, pp. 775–784 (2000) doi:10.1109/INFCOM.2000.832252

[21] Roos, T., Myllymäki, P., Tirri, H., Misikangas, P., Sievänen, J.: A Probabilistic Approach to WLAN User Location Estimation. International Journal of Wireless Information Networks 9(3), 155–164 (2002)

[22] Honjo, K., Simpson Jr., D.L.: Stereo FM receiver, noise control circuit therefor. US Patent 5432854 (July 1995)

[23] Leentvaar, K., Flint, J.H.: The capture effect in FM receivers. IEEE Transactions on Communications 24, 531–539 (1976)

[24] `http://www.create-net.org/mise`

[25] `http://www.koniggaming.com/mp3withfmtransmitter.html` (as accessed on 2009.02.19)

[26] JPL's Wireless Communication Reference Website. Analog Transmission over Fading Channels, `http://www.wirelesscommunication.nl/reference/chaptr05/analog/analog.htm` (as accessed on 2008.11.20)

[27] Mitchell, T.: Machine Learning. McGraw-Hill, New York (1997)

[28] Kaemarungsi, K.: Distribution of WLAN Received Signal Strength Indication for Indoor Location Determination. In: Proceedings of the 1st International Symposium on Wireless Pervasive Computing, pp. 1–6 (2006)

[29] Smith, K.: Mobile phones demystify commuter rat race. Nature News, June 4 (2008)

PIV

YOU'VE GOT PHOTOS! THE DESIGN AND EVALUATION OF A LOCATION-BASED MEDIA-SHARING APPLICATION

by

Niko Kotilainen and Maria Papadopouli 2008

In Proceedings of the 4th International Mobile Multimedia Communications Conference

You've Got Photos! The design and evaluation of a location-based media-sharing application

Niko Kotilainen
Department of Mathematical Information Technology
University of Jyväskylä
Finland

Maria Papadopouli[*]
Department of Computer Science
University of Crete &
Institute of Computer Science
Foundation for Research & Technology-Hellas

ABSTRACT

PhotoJournal is a novel location-based media sharing application that enables users to build interactive journals that associate multimedia files with locations on maps and share this information with other users. Its underlying information discovery and sharing mechanism is 7DS that runs in either pure peer-to-peer or centralized server-to-client mode, depending on the availability of a server and/or an infrastructure. 7DS-enabled devices act as miniature caches, sharing information with each other. When access to an information server (e.g., web server) is not available, the local 7DS instance running on the device enables the device to search and access information from other peers in proximity. We have implemented the prototype and evaluated the delay to access the data using three testbeds. Two of these testbeds employ a centralized (server-to-client) architecture, while the third one applies the peer-to-peer paradigm. Depending on the underlying network technology and device capabilities, this delay varies. The results encourage us to perform additional empirical-based studies under increased traffic load conditions and initiate a user-study in the premises of a museum and a research park.

1. INTRODUCTION

New applications and tools for sharing and experimenting with multimedia data in a synchronous or asynchronous manner, such as Flickr, YouTube, Me.dium, MySpace, facebook, and JumpCut, have enriched on-line collaboration, allowing the formation of new types of social networks, interactions, and online communities. Furthermore, the market of location-based services grows rapidly. In the near future, mobile devices that have the processing, communication, and geolocating capabilities will enable seamless integration of services combining media sharing and geographical tagging.

PhotoJournal applies the peer-to-peer (p2p) paradigm to facilitate the access and sharing of location-based multimedia content among mobile devices. It also enables users to build interactive multimedia "journals" that associate multimedia objects, such as pictures, video, or hypertext, with locations on maps. Multimedia files can be "superimposed" on certain locations of maps and users may manage, review, update or delete them. Users may query for content regarding an area of a map and update their journal. Peers that run an instance of the PhotoJournal application on a device may respond to such queries and share their multimedia content.

PhotoJournal is supported by graphical user interfaces (GUIs) to access, search, share, and manage the multimedia content and a middleware with two main components, namely a positioning and an information discovery and sharing system. The underlying positioning technologies are GPS and the Cooperative Location-sensing System (CLS) [9, 22], while 7DS [20] enables information discovery and sharing. When access to an information server (e.g., web server) is not available (e.g., a device experiences intermittent connectivity to the Internet), the 7DS instance running on that device (e.g., peer) enables the peer to search and access information from other peers in the wireless LAN. 7DS can instantiate both the server-to-client and peer-to-peer paradigms and provide complementary access through peers, when an infrastructure—or connectivity to an infrastructure—is not available. 7DS assumes that, in the face of disconnections, users can trade the data consistency and currency over data availability.

We implemented PhotoJournal and evaluated its performance under both peer-to-peer and infrastructure-based architectures. The delay that a user experiences to access the requested data from the time the device is in the range of another cooperative device with relevant data (i.e., *dataholder*) is measured. Depending on the underlying network technology (e.g., 3G or IEEE802.11), architecture, and device capabilities, the median delay varies from 282 ms (in a p2p architecture, running on a PC in a IEEE802.11 single-hop network) to 1.9 s (running the application on a smartphone and accessing the web server via a 3G network). However, the frequency that a device is close to a dataholder depends on several parameters, such as popularity of the data, den-

[*]This research was supported by the Greek General Secretariat for Research and Technology under Regional of Crete, Crete-Wise KP-18 (ΚΠΣ 00126) and 05NON-EU-238.

sity of peers, mobility pattern, and transmission power.

Section 2 presents the related work, while Section 3 focuses on the PhotoJournal architecture and introduces its main components. The performance of the PhotoJournal is analyzed in Section 4. Finally, Section 5 reflects on mobile peer-to-peer computing and Section 6 summarizes our main conclusions and future work plans.

2. RELATED WORK

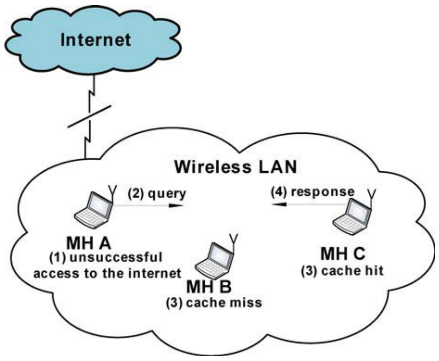


Figure 1: Example of information sharing using 7DS. The arrows show the message exchange for the 7DS communication. The light-shaded area denotes the wireless LAN, the darker-shaded area the Internet, and the thunderbolt-like shape the WAN connection that is not currently available.

The anticipation of a growing number of users that form "on-line" communities to gossip, share information and resources via their wireless-enabled devices inspired the design of 7DS. 7DS may relay, search for and disseminate information in a self-organizing manner, without the need for an infrastructure. 7DS-enabled devices can interact either in a p2p or server-to-client manner. These different modes of operation allow 7DS to instantiate different mobile information access schemes when possible, and provide complementary access through peers, when an infrastructure is not available. Figure 1 illustrates an example of 7DS peer-to-peer use. Mobile host A (MH A) tries to access a data object. The local 7DS instance running on host A detects an unsuccessful attempt to connect to the Internet and tries to retrieve the data from peers that are within its wireless range. Both hosts B and C (MH B and MH C, respectively) are within the range of host A and receive the query. Unlike host B, host C has a copy of the data in its cache and responds to host A's query.

Applications interact with 7DS employing pairs of attributes to describe the data that they are willing to share with other application instances running on peers. For each application, 7DS maintains an index of the local cache that is populated with data that can be shared. This data may have been acquired from other peers or servers [18, 19, 20, 21].

MOBY [11] proposes a service-oriented network architecture, in which each peer interacts both with the available infrastructure and its neighbours. It provides a method to integrate available services in handheld mobile devices. MOBY's architecture is based on Mnode super-peers, which allow mobile devices to access and locate available services, as opposed to the 7DS platform, where there is no need for external storage. Although super-peers are mainly responsible for service management, interaction among Mnodes is encouraged in order to reduce load on peers acting as gateways. Horozov *et al.* in [11] discuss security challenges by integrating secure service registration capabilities in the available architecture.

Mobile chedar [15] is a middleware extension to Chedar [4], providing resource sharing and distribution in mobile p2p systems, in a completely decentralized fashion. The proposed API performs topology management by selecting connections that aim to establish a scalable and fault-tolerant network. Communication among peers in both Chedar and mobile Chedar is Gnutella-like, in which queries are sent to neighboring peers and direct connections are created among them. P2P systems that adopt the Chedar API are not evaluated in terms of performance, and no analysis is provided concerning the impact of malicious users on system security.

LightPeers [7] is a lightweight mobile p2p platform, developed to support users utilizing a variety of mobile devices with limited capabilities. Communication among peers is established by broadcasting discovery messages and multicasting queries to nodes of the same group. LightPeers was implemented to ease the exchange of information and services among peers and support interactive applications. This architecture can be used in ad-hoc networks that facilitate delay tolerant messaging.

Proem is a Java oriented middleware platform for developing and deploying applications for mobile ad-hoc networks. The Proem middleware consists of three parts: an application runtime environment, a set of middleware services, and a protocol stack for communication. In Proem, each application is managed by the peerlet engine, which is responsible for dynamically adding and removing peerlets from the system. The set of middleware functions is designed to allow distributed applications to share resources and exchange event information, and declare and discover new services. The protocol stack defines the syntax and semantics required to enable communication between peers.

This paper considers that in the wireless range of a querier, there is a cooperative device with the relevant data (in the p2p architecture) or a predefined web server that can be accessed via the wireless Internet (in the infrastructure architecture), respectively. The frequency that a device is in the range of dataholders has a dominant impact on the total delay that a user will experience, i.e., the total time elapsed from the formation of the query until the local device receives relevant data. However, this delay depends on several parameters, such as popularity of the data, density of peers, their mobility pattern and transmission power. An evaluation of the impact of these parameters on data dissemination assuming random-walk based mobility patterns can be found in [18, 19, 20, 21]. In general, for different usage and application characteristics, the likelihood that users in proximity would be interested in similar data varies.

Although currently 7DS uses single-hop multicast, a routing protocol could facilitate the communication among peers. Mobile peer-to-peer computing applications often create sparse and intermittently-connected networks, referred to as delay tolerant networks (DTNs). Traditional routing protocols for ad-hoc networks do not perform well in DTNs due to their unstable paths. Important parameters in the design of such routing protocols are the co-residency time between peers, time that a peer is out of the range of other relay nodes, information servers, or access points (APs), relaying, querying, and cooperation policies, information locality, and buffer management. Several studies on routing protocols in DTNs have appeared, evaluating the impact of buffer management, relaying policies, and placement of relay nodes (e.g., [8, 5, 17, 25]) or of the knowledge about device location, peer movement and connectivity patterns on the routing protocol (e.g., [16, 13, 23, 24]). Depending on the communication patterns between peers, different network topologies can be formed, affecting dramatically the speed that the information is disseminated. Analyzing how fast data spread in scale-free networks has been the focus of recent studies (e.g., [14]).

3. PHOTOJOURNAL ARCHITECTURE

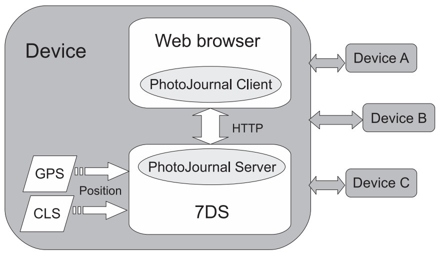


Figure 2: PhotoJournal architecture: The application is supported by an underlying information discovery and sharing mechanisms (7DS) and positioning system (e.g., CLS, GPS).

PhotoJournal allows the creation of interactive location-based multimedia journals and enables users to discover and share their content with each other. A local PhotoJournal instance may automatically superimpose local multimedia content to the appropriate areas of maps and enable a user to specify location-based queries for certain areas of a map. It is supported by 7DS, its underlying information sharing and discovery mechanism, and a positioning system, GPS and CLS or outdoor and indoor environments, respectively. PhotoJournal runs as an application with a web browser-based user frontend.

3.1 CLS

CLS applies the peer-to-peer paradigm by enabling devices to gather positioning information from other neighboring peers, estimate their distance from their peers based on signal-strength measurements, and position themselves accordingly [9]. CLS creates a *signal-strength signature map* of the physical space during a training phase and compares it with analogous run-time measurements employing various statistical-based criteria. Iteratively, it can refine its positioning estimates by incorporating newly received information from other devices.

CLS and GPS periodically record the coordinates of the current position of the device with a timestamp in the *positioning trace*. Users can upload pictures and videos with their associated timestamp. PhotoJournal can correlate the timestamp information of the multimedia content with the positioning trace and associate the multimedia files with certain areas of a map.[1]

3.2 User frontend

The user frontend of the PhotoJournal client is a web-browser based interface that communicates with the local PhotoJournal server using HTTP. The latter is responsible for the interaction of the PhotoJournal client with the 7DS module, forwarding the corresponding messages and assuring that packets are in the proper format as they are exchanged across different modules.

Using a PhotoJournal GUI, a user may superimpose multimedia content on certain locations of the map by clicking on the map and browsing the multimedia files corresponding to that location. Moreover, a user can add, modify, or delete comments about a certain multimedia file, rate its content and set its access permission. A multimedia file can be set public or private—only public files can be shared with other peers. The PhotoJournal frontend (as shown in Figure 4) runs on a web browser and consists of a map frame on the right and a photo bar on the left side of the window. Its backend runs on 7DS. It receives all queries from the frontend through 7DS's proxy server, and supports the typical 7DS functionality by adding or deleting photos, querying photos from 7DS neighbors or handing photos from the local cache. 7DS can also cache map files, enabling the application to work without an Internet connection.

PhotoJournal can automatically superimpose the uploaded content on an appropriate map by matching the timestamp of the content of the multimedia files with the timestamp of the GPS/CLS trace and associating these files with the corresponding position on the map. Furthermore, it updates its local 7DS cache and its indexing mechanism.

Figure 2 summarizes the main components of the location-based media sharing system, namely the PhotoJournal application, 7DS and CLS.

3.3 Peer discovery and information access

A user may search for multimedia content related to a certain location in the following manner: First, the user indicates the region of interest by marking the corresponding area on the displayed map (e.g., the white rectangular on the map illustrated in Figure 4). Then, the local 7DS instance will search for relevant data in its cache, on the web, and in the cache of other peers. Specifically, it will first check its local cache for multimedia files associated with that area. If the search is successful, it will display a marker with a number indicating the number of multimedia files associated with that location. In the case that no relevant data can be found, 7DS's web client attempts to acquire it from the Internet by accessing a predefined web site. Finally, if the web client fails to acquire the requested data (e.g., in the case of intermittent connectivity to the Internet or unavailability of a web server), 7DS will form a *media query* and multicast it to its peers. A media query describes the requested data

[1]We assumed that the digital camera timestamps recorded files and is synchronized with the user's device running a positioning system.

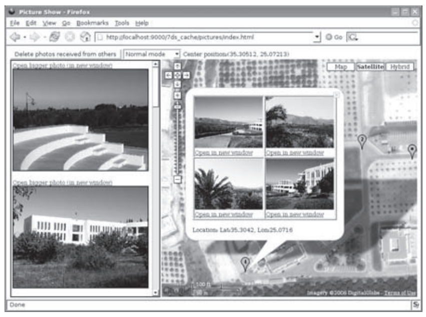

Picture Show - Firefox
File Edit View Go Bookmarks Tools Help
http://localhost:9000/7ds_cache/pictures/index.html
Delete photos received from others
Normal mode
Map Satellite Hybrid
Open in new window
Location: Lat:35.3042, Lon:25.0716
Done

Figure 3: PhotoJournal can superimpose multimedia objects at their locations on a map. A marker indicates the number of files associated with that location.

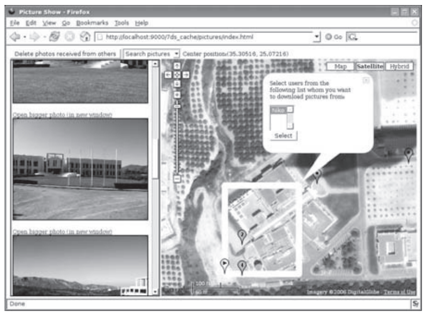

Picture Show - Firefox
File Edit View Go Bookmarks Tools Help
http://localhost:9000/7ds_cache/pictures/index.html
Delete photos received from others
Search pictures
Map Satellite Hybrid
Select users from the following list whom you want to download pictures from:
Select
Done

Figure 4: On the main GUI of the PhotoJournal a user can mark the area for which multimedia information will be requested.

in XML format and is formed using location- and rate-based criteria. Upon the reception of a media query, the local 7DS instance of a peer may search its local cache for relevant data. If a relevant data object is found in its cache, it will form and send an XML response, including a URL to the relevant multimedia file, reviews and rating information.

The PhotoJournal will display—periodically updating it—the list of peers that responded to recent queries. A user may select a certain 7DS peer from this list to retrieve the requested content. When the web client retrieves the relevant data objects from the peer, it stores them in the local cache and displays them on the map (as illustrated in Figures 3 and 4). Areas on the map associated with multimedia files can be distinguished by a marker that also indicates the number of the available relevant files. A demo of the PhotoJournal can be found in [2].

4. PERFORMANCE ANALYSIS

To evaluate the performance of the PhotoJournal, empirical-based measurements were performed using two different architectures, namely, an ad-hoc and an infrastructure-based one. The infrastructure-based approach realizes the traditional server-to-client approach: PhotoJournal clients request content from a web server. Unlike the infrastructure-based architecture, in the ad-hoc (i.e., p2p) architecture, devices access the information in a peer-to-peer manner. In the infrastructure-based architecture, we experimented with both 3G or IEEE802.11 technologies. The estimated effective downlink speed of the 3G connection is approximately 400 Kbps. The peers in the p2p architecture communicate via IEEE802.11 in the ad hoc mode.

Throughout the text, the terms "infrastructure" and "centralized" are used interchangeably for describing the architecture paradigm and testbed (similarly with the terms "ad-hoc" and "p2p").

4.1 Metrics

To evaluate the performance of the PhotoJournal application over the two different architectures, the following benchmarks are defined:

1. *Query processing delay*: the total time elapsed between the reception of a (neighbor or a media) query and the transmission of a response.

2. *Query transmission delay*: the time spent by a specific query to travel over the network, subject only to network elements and propagation delays.

3. *Query forming delay*: the time that a request spends at various levels of the protocol stack before being forwarded to the network layer.

4.2 Testbeds

The empirical-based measurements were performed on three testbeds. Two of them employed the infrastructure-based architecture, while the third one the ad-hoc based one. The client-to-server communication takes place using 3G (in the "3G infrastructure" testbed) and IEEE802.11 (in the "IEEE802.11 infrastructure" testbed). In these testbeds, the wireless client is a Nokia N80 smartphone and the web server runs Apache. In the ad-hoc testbed, wireless clients running the PhotoJournal application are part of an IEEE802.11b adhoc network, each using a PC equipped with an A-Link USB WLAN interface.

To measure these different delays, monitors are placed at certain testbed locations. The time granularity of these measurements depends on the specific platform. For example, the time-keeping clock of the smartphone has a frequency of 64 Hz (corresponding to a granularity of approximately 15 ms [10]), while the granularity of the web server monitor is of 1 μs. Table 1 illustrates the list of monitors used in our measurements along with the specific event they capture.

The main difference between the p2p and the centralized architecture is that the latter does not require a peer (neighbor) discovery phase and querying devices send their requests directly to a predefined web server via either the 3G or IEEE802.11 infrastructure.

We ran 30 experiments using PhotoJournal in each of the three testbeds. In the infrastructure-based experiments, a script initiated a sequence of queries for the same content. For each experiment in the p2p testbed, a user selected a different region of interest on the map, initiating a query for related content. The resulted delay measurements are shown in Figures 5, 6, and 7.

P2P architecture	
Time	**Event description**
T_1	local 7DS instance receives a request
T_2	local 7DS multicasts a discovery query
T_3	peer receives a query
T_4	dataholder sends a response
T_5	querier receives response
Centralized architecture	
Time	**Event description**
T_6	smartphone receives a user query
T_7	smartphone sends a query to web server
T_8	web server receives a query
T_9	web server sends response to smartphone
T_{10}	smartphone receives response

Table 1: Monitors capturing various event types in the infrastructure and p2p architectures. The term T_i indicates the time the event i was recorded at the corresponding monitor.

Delay	P2P	Infrastructure
Query forming	$T_2 - T_1$	$T_7 - T_6$
Query processing	$T_4 - T_3$	$T_9 - T_8$
Query transmitting	$T_5 - T_2 - T_4 + T_3$	$T_{10} - T_7 - T_9 + T_8$

Table 2: Different delay types as measured based on the recorded event times for the p2p- and infrastructure-based architectures.

4.3 Query forming delay

The processing power and CPU load impact the query forming delay. In the infrastructure architecture, the query forming delay is significantly larger, exhibiting also higher variability compared to the ad-hoc one (as shown in Figure 5). The PhotoJournal query in the centralized testbed is formed by a smartphone with scarce resources and low processing capabilities compared to the powerful PCs used in the p2p testbed. Furthermore, compared to IEEE802.11 LANs, 3G networks are more demanding in terms of processing power. However, even when a PhotoJournal-client runs on a powerful PC, the query forming delay remains significantly large, up to 100 ms. Such values are due to the XML document processing performed by 7DS in order to describe the requested items. Moreover, the interaction between the 7DS component and the PhotoJournal client results in frequent context switches, increasing further the measured delay.

As expected, query forming delay is larger in the case of media queries compared to neighbor queries, since in the former case the corresponding request needs to describe the area of interest whereas in the latter case, a peer discovery request is simply a "template" message (used to search for peers in the wireless range of the querier).

4.4 Query processing delay

The query processing delay is significantly lower in the centralized setting compared to the p2p one (as shown in Figure 6). This is due to the complexity that 7Ds introduces, when used in networks where no 3G or IEEE802.11 infrastructure is available. The time required for a web server to form a response is significantly lower from the one of a regular 7DS peer. Note that in the ad-hoc testbed, 7DS employs XML for describing a data item, while in the infrastructure-based architecture, a 7DS client *only* sends a simple HTTP request *directly* to the web server over the Internet. The total time elapsed between the reception of a media query and the transmission of a response is greater than 40 ms for the 70% of the queries in p2p mode (as illustrated in Figure 6).

The query processing delay for media access is larger than for peer discovery due to the intense processing required for the first query type. When a peer receives a media query, it will search the local cache for relevant media items. The variability exhibited in the ad-hoc architecture is due to the variable sized map areas.

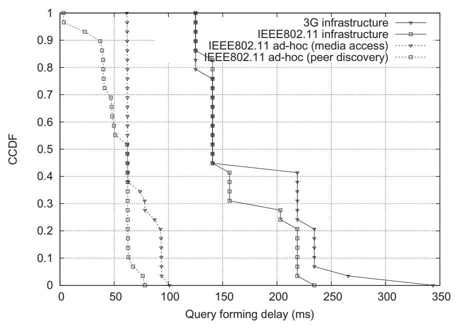


Figure 5: Complementary cumulative distribution function (CCDF) of query forming delays.

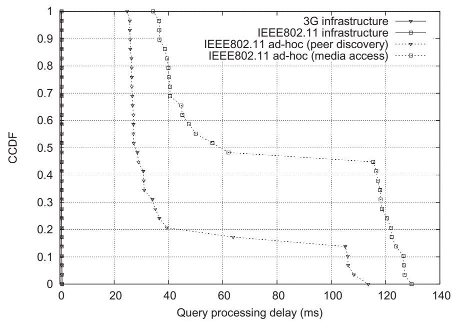


Figure 6: Complementary cumulative distribution function (CCDF) of query processing delays.

4.5 Query transmission delay

The transmission delay of a media access query is stochastically larger than the delay of a peer discovery one (as shown in Figure 7). For example, in p2p, the median media access delay is approximately 39 ms compared to a median peer discovery delay of 15 ms. A typical response size to a media access query is approximately 29 KB (compared to only a few bytes which is the response to a peer discovery one).

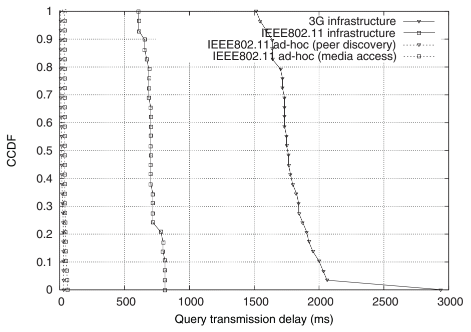


Figure 7: Complementary cumulative distribution function (CCDF) of query transmission delays.

The variability in query transmission delay is due to the various sizes of the response for different query types. Also compared to the ad-hoc network, an infrastructure results in increased transmission delays due to the aggregate traffic of other clients associated with the IEEE802.11 AP and the lower transfer rates (in 3G). The IEEE802.11 centralized approach exhibits lower transmission delays than 3G (e.g., median delay of 750 ms compared to 1800 ms). With 3G's maximum speed of 2 Mbps, it is hard to compete with the 11 Mbps transfer rate of IEEE802.11.

To summarize, the mean and median delays from the time the PhotoJournal received user's input until the reception of the relevant media files are are 1,995 ms and 1,898 ms in 3G, 876 ms and 843 ms in IEEE802.11 infrastructure, and 305 ms and 282 ms in IEEE802.11 ad-hoc, respectively. Thus, the 7DS/PhotoJournal introduces delays that are tolerable. However, several aspects of the current 7DS implementation can be improved; For instance, the code is large and complex. It can be simplified significantly using libraries included in recent Java versions. We intend to evaluate its scalability under increased traffic load conditions. Furthermore, it would be interesting to perform a user study in the premises of a museum or FORTH and collect additional feedback from users (e.g., visitors in these premises) not only about its performance but also its features and GUIs.

5. DISCUSSION ON MOBILE P2P SYSTEMS

The development of 7DS and PhotoJournal motivated us to reflect on mobile peer-to-peer computing. Critical aspects of a mobile peer-to-peer system are the incentives for cooperation and privacy requirements.

The effectiveness of mobile peer-to-peer computing systems depends on their substantial deployment, cooperation, interoperability, and scalability. Depending on the availability of a resource, a peer may dynamically adapt its cooperation strategy. The scarcity of resources enhances the tension between cooperation and competition. Given the energy constraints, the nondeterministic characteristics of the environment, and the presence of exogenous parameters that impact the resource availability, such resource allocation algorithms are non-trivial. In general, the following parameters impact the power consumption of a network interface: size and number of packets sent and received, and time the network interface is on. To reduce the power consumption, these parameters need to be kept low.

To prevent denial of service attacks, encourage cooperation, and better allocate resources, the use of micropayment-based and/or reputation-based mechanisms can be important [3, 6, 26, 12, 1]. However, these mechanisms should have a relatively low overhead, in order to not discourage the energetic participation of peers. While a relaxed protection of resources may impede the use of a peer-to-peer system, high costs or strict conditions to access the resources may dissuade their usage. The design of a mobile p2p system needs to address the balance between these two requirements.

Increasingly wireless devices collect a large amount of information that can be analyzed to reveal the personal and social context of the user. This abundance of information makes users vulnerable to intrusion of privacy threats. The identification of the position of the device and potentially, the identity of the subject using the device—which can be acquired directly or inferred using statistical analysis—are examples of such threats. Malicious users can abuse such information by spamming users with advertisements or disclosing it inappropriately. Thus, a tradeoff between enhancing the information access and disclosing private information inappropriately is exposed. The larger the availability of information, the more likely is to enhance the information access and sharing but also the higher the vulnerability in privacy threats.

As in the case of the Internet, peer-to-peer systems need to be flexible and dynamic to sustain long-term use. Privacy will play an important role in the adoption of mobile peer-to-peer computing applications. Currently, 7DS and PhotoJournal offer a crude distinction between private and non-private objects and a finer way to describe their privacy requirements is needed. However, privacy is context sensitive and depends on the social context, user activity, ownership of the device, application, and personality of the user. Depending on these parameters, the system may decide about the privacy and cooperation policies with or without any user intervention. Thus, it is important to provide mechanisms that allow a fine-level description of the privacy requirements and draw a balance between enhancing the service and protecting user privacy.

6. CONCLUSIONS

This work focused on PhotoJournal, a multimedia location-based application, and analyzed the delay that the application experiences from the time the request is formed until a response is received. Depending on the underlying network technology and device capabilities, this median delay varies from 282 ms to 1,9 s. In these experiments, in the wireless range of a querier, there was always a cooperative device with the relevant data (in the ad-hoc testbed) or a predefined web server that can be accessed via the wireless Internet (in the infrastructure testbeds), respectively. As mentioned earlier, the frequency that a device is in the range of dataholders has a great impact on the total delay that a user will experience, i.e., the total time elapsed from the formation of a query until the local device receives relevant data. Our earlier research analyzed the data dissemination in ad hoc wireless network, assuming random-walk based mobility models. An interesting followup study would consider heterogeneous wireless environments, supported partially by wireless infrastructures; in areas with limited or no coverage by APs, the mobile peer-to-peer computing paradigm can be used to enhance the information access. In such environments, it would be useful to evaluate various routing protocols integrated with mobile peer-to-peer systems using *more realistic* access and traffic patterns.

Only a few studies on mobile p2p systems evaluate the performance of their system with empirical-based measurements. Typically, the evolution of a technology includes the following steps: simulation-based studies of the technology, measurements in a real-life testbed and controlled experiments, and further empirical-based measurement studies in large-scale testbeds (if the technology becomes widely adopted). To assist the deployment of mobile peer-to-peer computing systems, a fruitful approach would include the development of the following components [21]:

- a general infrastructure for mobile peer-to-peer applications and a toolkit that new applications could use
- robust mobile peer-to-peer applications with friendly GUIs that can also control the distribution of data and form context- and semantic-based queries
- protocols that ensure anonymity and privacy
- mechanisms that encourage cooperation among peers in an energy-efficient manner

Mobile p2p computing opens up exciting challenges in computer science, demanding interdisciplinary research and innovative paradigms.

7. ACKNOWLEDGEMENT

The authors would like to thank Manolis Ploumidis and Elias Raftopoulos for their help in reviewing an earlier draft and providing useful feedback on the performance analysis and Alexandros Kapravelos, Despina Maggina, and Vasiliki Papavasileiou for their help in the implementation of some features of the PhotoJournal.

8. REFERENCES

[1] Ling liu. security and trust in peer-to-peer systems: Risks and countermeasures. http://www.cc.gatech.edu/~lingliu/keynotes/.

[2] Mobile Computing Activity at FORTH-ICS. http://www.ics.forth.gr/mobile/software.html/.

[3] Peer-to-peer: Harnessing the power of disruptive technologies. http://www.freehaven.net/doc/oreilly/accountability-ch16.html.

[4] A. Auvinen, M. Vapa, M. Weber, N. Kotilainen, and J. Vuori. Chedar: Peer-to-peer middleware. In *Proceedings of the 19th IEEE International Parallel & Distributed Processing Symposium (IPDPS 2006)*, Rhodes Island, Greece, July 2006.

[5] Brendan Burns, Oliver Brock, and Brian Neil Levine. Autonomous Enhancement of Disruption Tolerant Networks. In *Proc. IEEE International Conference on Robotics and Automation*, Orlando, Florida, May 2006.

[6] Levente Buttyan and Jean-Pierre Hubaux. Nuglets: a virtual currency to stimulate cooperation in self-organized mobile ad hoc networks. Technical Report DSC/2001/001, Swiss Federal Institute of Technology, Lausanne, January 2001.

[7] Bent Guldbjerg Christensen. Lightpeers: A lightweight mobile p2p platform. In *Proceedings of the Fifth IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOMW '07)*, pages 132–136, White Plains, NY, March 2007.

[8] James Davis, Andy Fagg, and Brian Neil Levine. Wearable computers as packet transport mechanisms in highly partitioned ad-hoc networks. In *Proc. International Symposium on Wearable Computers (ISWC)*, Zurich, October 2001.

[9] Charalampos Fretzagias and Maria Papadopouli. Cooperative Location Sensing for Wireless Networks. In *Second IEEE International conference on Pervasive Computing and Communications*, Orlando, Florida, March 2004.

[10] Richard Harrison. *Symbian OS C++ for Mobile Phones*. John Wiley & Sons Ltd, 2003.

[11] T. Horozov, A. Grama, V. Vasudevan, and S. Landis. Moby — a mobile peer-to-peer service and data network. In *Proceedings of International Conference on Parallel Processing*, pages 437–444, Washington, DC, USA, August 2002.

[12] Jean-Pierre Hubaux, Levente Butyan, and Srdan Capkun. The quest for security in mobile ad hoc networks. In *ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, pages 146–155, Long Beach, CA, October 2001.

[13] Sushant Jain, Kevin Fall, and Rabin Patra. Routing in a delay-tolerant network. In *ACM Symposium on Communications Architectures and Protocols (SigComm)*, Portland, OR, USA, August 2004.

[14] Jon Kleinberg. The wireless epidemic. *Nature (News and Views)*, 449:287–288, 2007.

[15] Niko Kotilainen, Matthieu Weber, Mikko Vapa, and Jarkko Vuori. Mobile Chedar - a peer-to-peer middleware for mobile devices. In *Proceedings of the Second International Workshop on Mobile Peer-to-Peer Computing (MP2P'05)*, pages 86–90, Kauai Island, Hawaii, March 2005.

[16] Qun Li and Daniela Rus. Sending messages to mobile users in disconnected ad-hoc wireless networks. In *ACM International Conference on Mobile Computing and Networking (MobiCom)*, pages 44–55, Boston, MA, USA, August 2000.

[17] Marc Liberatore, Brian Neil Levine, and Chadi Barakat. Maximizing Transfer Opportunities in Bluetooth DTNs. In *Proc. ACM Conference on Future Networking Technologies (CoNext)*, Lisboa, Portugal, December 2006.

[18] Maria Papadopouli and Henning Schulzrinne. Seven degrees of separation in mobile ad hoc networks. In *IEEE Conference on Global Communications (GLOBECOM)*, San Francisco, CA, November 2000.

[19] Maria Papadopouli and Henning Schulzrinne. Effects of power conservation, wireless coverage and cooperation on data dissemination among mobile devices. In *ACM International Symposium on Mobile Ad Hoc Networking and Computing (Mobihoc)*, Long Beach, CA, October 2001.

[20] Maria Papadopouli and Henning Schulzrinne. A performance analysis of 7DS a peer-to-peer data dissemination and prefetching tool for mobile users. In *Advances in wired and wireless communications, IEEE Sarnoff Symposium Digest*, Ewing, NJ, March 2001.

[21] Maria Papadopouli and Henning Schulzrinne. *Peer-to-Peer Computing for Mobile Networks: Information Discovery and Dissemination*. Springer (under preparation), 2008.

[22] Konstantinos Vandikas, Lito Kriara, Tonia Papakonstantinou, Anastasia Katranidou, Haris Baltzakis, and Maria Papadopouli. Empirical-based analysis of a cooperative location-sensing system. In *ACM First International Conference on Autonomic Computing and Communication Systems (Autonomics)*, Rome, Italy, October 2007.

[23] J. Yang, C.-K. Lee Y. Chen, and M. Ammar. Ferry

replacement protocols in sparse manet message ferrying systems. In *IEEE Wireless Communications and Networking (WCNC)*, New Orleans, LA, March 2005.

[24] Wenrui Zhao, Mostafa Ammar, and Ellen Zegura. A message ferrying approach for data delivery in sparse mobile ad hoc networks. In *IEEE Conference on Computer Communications (InfoCom)*, Hong Kong, March 2004.

[25] Wenrui Zhao, Yang Chen, Mostafa Ammar, Mark D. Corner, Brian Neil Levine, and Ellen Zegura. Capacity Enhancement using Throwboxes in DTNs. In *IEEE International Conference on Mobile Ad hoc and Sensor Systems*, Vancouver, Canada, October 2006.

[26] Lidong Zhou and Zygmunt J. Haas. Securing ad hoc networks. *IEEE Network*, 13(6), November 1999.

PV

A GENETIC-NEURAL APPROACH TO MOBILITY-ASSISTED ROUTING IN A MOBILE ENCOUNTER NETWORK

by

Niko Kotilainen and Jani Kurhinen 2008

In Proceedings of the 5th International Conference on Information Technology and Applications

A Genetic-Neural Approach for Mobility Assisted Routing in a Mobile Encounter Network

Niko P. Kotilainen, Jani Kurhinen

Abstract--**Mobility assisted routing (MAR) is a concept, where the mobility of a network's nodes is used to physically carry data to its destination. Traditionally, MAR algorithms have been based on few simple rules, often limiting the performance of these algorithms. In this paper, we propose an architecture in which a trained neural network is fed information about the message and the encountered peer, and which then decides whether to forward the message to the encountered peer. This algorithm, called *NeuroRouter*, is capable of utilizing the most efficient routing strategies in different environments by adapting its behavior based on environmental variables.**

***Index Terms*—Mobile encounter network, Mobile peer-to-peer, Mobility assisted routing, Neural network.**

I. INTRODUCTION

Personal digital assistants (PDA) and voice-centered mobile phones have become powerful application platforms which are used in almost all fields of modern society. In addition to supporting a wide spectrum of applications, they can be used for creating new data. For example, one can contribute to a live blog or share photographs with the world immediately after they have been captured. The created data is transmitted for the most part via cellular or wireless local area networks, but short range wireless data links are also employed. At the same time peer-to-peer communication systems such as BitTorrent and Skype have taught people to utilize this new communication paradigm in both entertainment and business. While peer-to-peer computing has clearly shown its potential on the fixed Internet, application scenarios using short range connectivity remain underdeveloped. However, the idea of harnessing millions of mobile terminals to provide all imaginable content to information consumers is intriguing.

In the past, the mobility of a network's nodes has been considered problematic with respect to data delivery in a short-range local communication system. However, as Spyropoulos et al. [8] said, "mobility can be turned into a useful ally". In fact in ad-hoc networks where connectivity is very intermittent, node mobility is often the only option to deliver messages between distant nodes of the network. In [5] we introduced the concept of a mobile encounter network (MEN), which builds on the concept cited above. In a MEN environment, data is transmitted only during node encounters. Instead of being a cause of problems, the mobility of the nodes provides a method for data delivery from one node to another. The actual mobile encounter network is the result of all the encounters and data exchange. In a communication system like MEN, a given network node is able to create short-term connections with other network nodes, i.e. the network topology can be defined as a function that is dependent on time. Due to the frequent changes in the network topology, a node may end up inside the communication range of other parties which posses desired information or desire information from the node.

In general these types of systems are called delay tolerant networks (DTNs) [1]. On the other hand, DTNs often do not rely only on direct node to node data delivery, but also benefit from multi-hop routing. Data MULEs [6], one of the first concepts to describe this kind of environment, route data using several independent mobile carriers. Our studies in [3] and [4] discuss similar network systems where data is collected from several sources to one data sink. The data is collected and transported by mobile entities already moving within the environment, and therefore the delivery does not incur additional costs. The multi-hop transmission is the most practical approach in this case; instead of giving full responsibility to one mobile entity to deliver the data packet to its target location, the data is passed to another unit that, in turn, might be able to transmit the data to the actual receiver.

In mobility assisted routing (MAR), the mobility of the nodes in the network is an important data transportation medium. Because of the continuously changing network topology, there are short term internode communication links in the network that follow certain rules based on the mobility patterns of the nodes. In this paper, we propose an architecture in which neural networks are trained to become efficient MAR algorithms.

Section II of this paper describes mobile encounter networks. Section III describes currently proposed mobility assisted routing algorithms. In section IV we present our proposal for a MAR algorithm. Section V describes the neural network training process, and section VI contains conclusions and future work plans.

II. MOBILE ENCOUNTER NETWORKS

Short-range wireless technologies, such as Bluetooth and WLAN, enable mobile devices to network with other similar devices. Information can be diffused from a member of the network to another, and the mobility of the nodes enables a sparse network to transfer information between distant nodes of

This work was supported in part by the Nokia Foundation.

N. Kotilainen and J. Kurhinen are with the Department of Mathematical Information Technology, University of Jyväskylä, P.O. Box 35, 40014 Jyväskylä, Finland (e-mail: niko.kotilainen@jyu.fi, kurhinen@mit.jyu.fi).

the network. There is no known route between the nodes; the sender of the data just forwards the data to some of the devices it encounters, which in turn forward it further, and eventually the data is very likely to reach its destination. Mobile encounter networks form a new class of mobile networks that emerge when devices encounter and exchange information. One encounter is made up of the discovery of devices, the establishment of a connection between two devices and the exchange of data. The duration of the encounters is usually short, because of the mobility of the devices, but it can also be long if the mobile devices are not moving. These single information exchanges form a MEN, resulting in the diffusion of information in the network with a delay.

MENs are very dynamic, and unlike traditional ad-hoc networks, they don't provide multi-hop communication. This lack of real-time routing limits MEN usage to applications which can tolerate some delay in communication. But for suitable applications, MENs have several benefits: they are scalable, robust, do not require network infrastructure, and can work in very sparse networks. In addition, the short-range communication medium is free.

III. CURRENT MOBILITY ASSISTED ROUTING ALGORITHMS

To bring multi-hop data transmission into mobile encounter networks, the mobility of the nodes has to be used to deliver messages between nodes that do not have a direct communication route between them. The nodes forward their messages to encountered peer nodes, with the hope that they would deliver the message to the destination, or at least would forward it further to nodes going to the right direction. This is usually called mobility assisted routing (MAR). Fig. 1. shows a simple example of message delivery using MAR.

The nodes in a mobile encounter network only know their own situation and the information they get from encountered nodes; they do not have a global view of the network status or topology. Hence, making routing decisions is problematic. Mobility assisted routing algorithms can be divided into three classes: epidemic spreading, epidemic spreading with limitations or restrictions, and targeted data delivery. The third class of MAR protocols can be described as being more intelligent than the former classes. As opposed to random spreading, targeted data delivery methods focus on selecting appropriate carrier nodes among the contacted nodes.

Epidemic routing was first introduced by Vahdat and Becker [10]. As the name implies, the algorithm works like a disease: using epidemic routing, messages are passed to all possible network nodes in the hope that some node is able to deliver it to a target location. It is a very powerful method and always gives the smallest delay possible if the network system handles the data flow properly. However, its efficacy requires vast amounts of network resources. While copying the messages to other network nodes, the epidemic algorithm wastes plenty of system's resources like storage capacity, network bandwidth and battery power.

Spyropoulos et al. has proposed Spray and Wait [7] and later Spray and Focus [9] protocols, which are good examples of methods designed to limit the problems of pure epidemic diffusion. Spray and Wait exploits different types of counters to control the number of message copies in the network. Spray and Focus has evolved from Spray and Wait, and combines copying and forwarding. These schemes, however, do not qualitatively distinguish distinct nodes while passing message copies. Instead, they employ numerous randomly selected nodes as message carriers. However, the Spray and Focus protocol does try to take advantage of potential opportunities to forward the message closer to its destination during the focusing phase.

Even though they are more efficient than the pure epidemic diffusion, they still waste substantial amounts of device memory, battery power and network bandwidth while passing data to inappropriate network nodes.

There are certain limitations in all of the algorithms described above. First, these algorithms don't take into account the qualities of the receiving nodes when making the routing

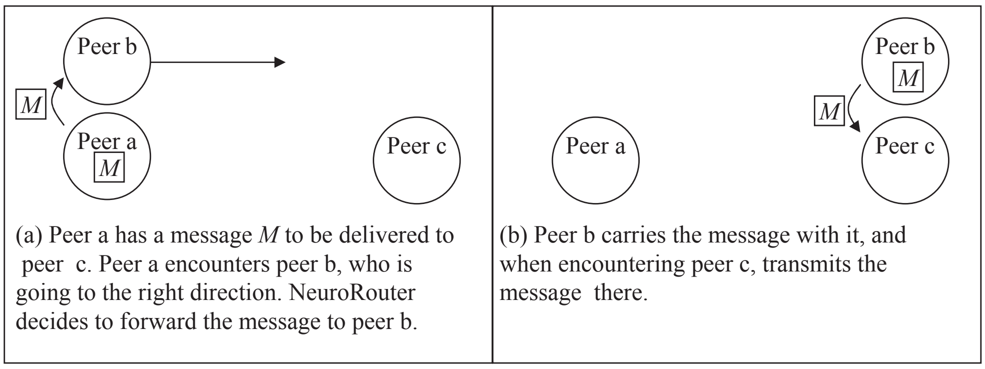

Peer b
M
Peer a
M
Peer c
(a) Peer a has a message M to be delivered to peer c. Peer a encounters peer b, who is going to the right direction. NeuroRouter decides to forward the message to peer b.
Peer a
Peer b
M
M
Peer c
(b) Peer b carries the message with it, and when encountering peer c, transmits the message there.

Fig. 1. Message delivery in a mobile encounter network using mobility assisted routing.

decisions. Second, each of these algorithms uses some control parameters (for example the number of "sprayed" packets or time-to-live) that can be used to tune the algorithm. In situations where a priori knowledge of the network environment is unavailable, a routing algorithm including configuration parameters is less than desirable. Finally, these algorithms don't adapt to the environment or to environmental changes because they rely only on one routing strategy. In general, only one strategy cannot be efficient in all scenarios. Therefore, an efficient algorithm should be able to utilize many strategies at the same time. To overcome these limitations, we propose a neural network based mobility assisted routing algorithm called NeuroRouter. NeuroRouter independently learns the correct behavior in given network conditions and uses many combinations of strategies to route packages. To our knowledge, it is the first MAR algorithm utilizing neural networks, or genetic algorithms in general.

IV. NEUROROUTER – A MOBILITY ASSISTED ROUTING ALGORITHM

When encountering peer nodes in the network, nodes have to decide whether to forward messages to the encountered peer node. This decision has a large impact on the efficiency of the network. As was discussed in section III, current MAR algorithms have limitations that affect their efficiency. Similar problems with resource discovery algorithms in static peer-to-peer (P2P) networks have been successfully solved using genetic algorithms [11]. In this paper we are proposing that the same idea be used in mobility assisted routing.

The proposed algorithm, NeuroRouter, decides to which of the encountered nodes to forward the messages held in its memory. Each time a pair of devices encounter one another, both devices input local information about their messages and the encountered node to a multi-layer perceptron neural network, of which output determines whether the message is forwarded to the encountered node. Fig. 2. illustrates this process. The neural network is a non-linear function approximator, which is organized into four layers: an input layer, two hidden layers and an output layer. The input layer contains the values of the neural network's inputs. The hidden layers do the actual work of decision making. The output layer simply provides a "Yes" answer if its inputs' sum is positive; otherwise the answer is "No". The layers are connected with weights, which determine the qualities of the neural network. In

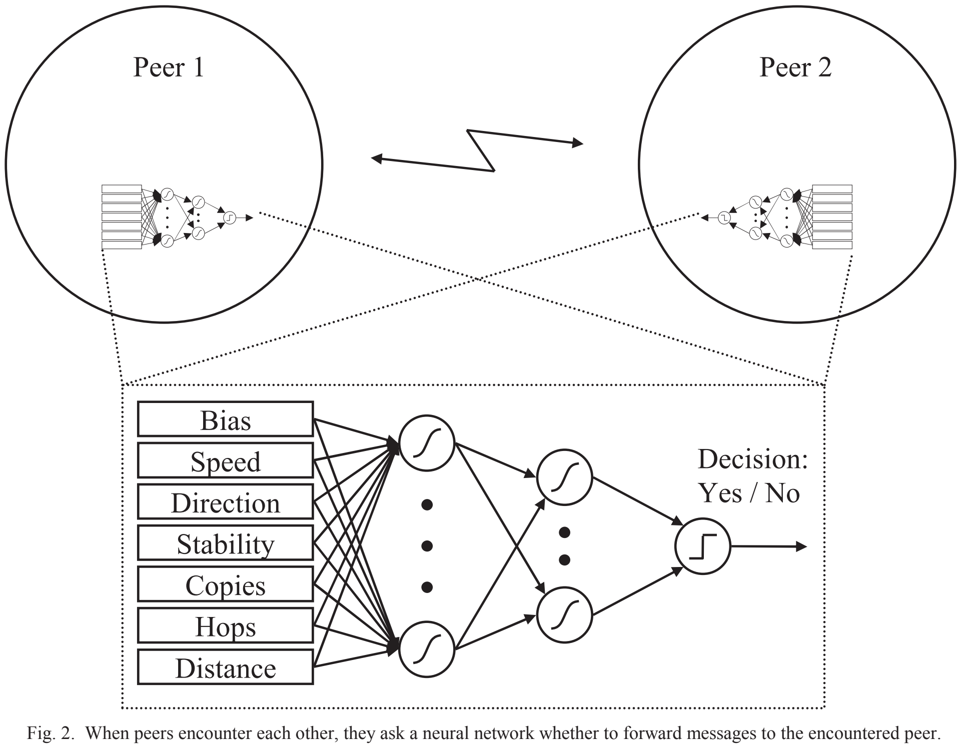


Fig. 2. When peers encounter each other, they ask a neural network whether to forward messages to the encountered peer.

Fig. 2's neural network, the arrows represent connections between layers, each connection having its own weight, and the circles represent neurons on the hidden layers and the output neuron. The first hidden layer has 16 neurons and the second one has 4 neurons. The activation function of the nodes of the hidden layer is hyperbolic tangent,

$$t(x) = \frac{2}{1+e^{-2x}} - 1,$$

where x is the sum of inputs of the neuron. The output of the neural network is calculated using neural network's weights W and inputs I with the following formula:

$$output(W,I) = \sum_{i=1}^{4} W_{3,i} t(\sum_{j=1}^{16} W_{2,j} t(\sum_{k=1}^{7} W_{1,k} I_i)).$$

The input parameters for the neural network are:

- "Bias", a constant 1.0
- "Speed", the encountered node's speed
- "Direction", the difference between the encountered node's direction and the direction to the destination
- "Stability", the stability of the encountered node's speed and direction
- "Copies", the number of copies of the message already sent.
- "Hops", the number of hops the message has taken to reach the current node from the message originator
- "Distance", the distance to the destination

V. NEURAL NETWORK TRAINING

Neural networks cannot make good decisions automatically, they have to be trained. Neural networks are trained by optimizing the weights that define the neural network's behavior until the neural network provides good results. Fig. 3. introduces the training process. Our system uses an evolutionary method to train the neural networks. In the beginning of the method, 30 neural networks are randomly generated, tested, and compared to each other. Then 15 worst performing networks are replaced with offspring of the 15 best performing networks. The offspring are created from the best performing networks by making Gaussian random changes to the parents. This test-compare-replace procedure is repeated thousands of times, and the neural networks gradually become very high-quality problem solvers. In the end the best individual from the neural networks is chosen to be the newly created MAR algorithm.

The training requires a lot of neural network evaluations. For example, training a population of 30 neural networks for 100.000 generations entails three million evaluations. As a result, the training cannot be done in a real-life network, but needs to be run in a simulator. For the training phase, we therefore need to define a mobility model of the environment. The model should reflect the parameters of the particular system, and therefore there is no one single solution that suits all. However, the random waypoint model, one of the most widely used mobility models, is a close enough approximation for training purposes.

We are currently modifying the P2PRealm [2] peer-to-peer simulator to support mobility assisted routing. After the NeuroRouter algorithm has been developed, i.e., the neural network has been trained; it can be deployed to a real-life network. After it has been deployed, the network's nodes can further improve and adapt the algorithm to their needs by using message history data as training material.

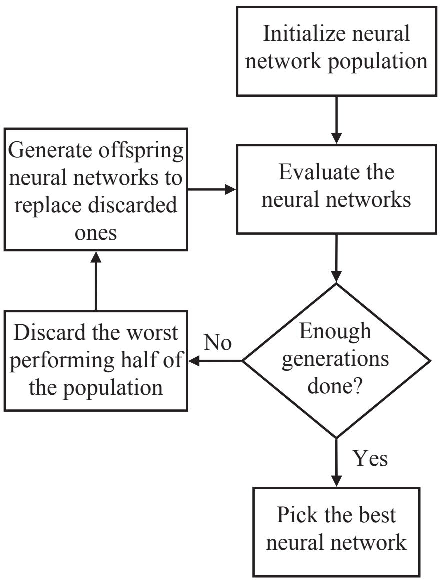

Initialize neural network population
Evaluate the neural networks
Generate offspring neural networks to replace discarded ones
Enough generations done?
No
Discard the worst performing half of the population
Yes
Pick the best neural network

Fig. 3. Neural network training procedure

VI. CONCLUSION AND FUTURE WORK

In this paper, a new mobility assisted routing algorithm called NeuroRouter has been proposed. The algorithm employs a trained neural network to make the routing decisions when peer nodes are encountered and thus can adapt to the environment and make more efficient routing choices.

We are now in the process of implementing the described system in a simulator environment using the P2PRealm [2] network simulator, so that the proposed algorithm could be compared to currently proposed MAR algorithms. We also intend to implement a testbed to evaluate the system in a real-life scenario. Future work on the subject will include using global information about the network to find an optimal solution to this problem. This solution would be the upper bound for MAR algorithms, and current MAR algorithms could be compared to this limit.

REFERENCES

[1] K. Fall, "A delay-tolerant network architecture for challenged internets," In *ACM SIGCOMM* 2003.

[2] N. Kotilainen, M. Vapa, T. Keltanen, A. Auvinen, and J. Vuori, "P2PRealm - peer-to-peer network simulator," in *11th International Workshop on Computer-Aided Modeling, Analysis and Design of Communication Links and Networks*, 2006.

[3] J. Kurhinen, "MP2P network in collecting data from sensor networks," in *11th IEEE Symposium on Computers and Communications*, 2006.

[4] J. Kurhinen and J. Janatuinen, "Geographical routing for delay tolerant encounter networks," in *12th IEEE Symposium on Computers and Communications*, 2007.

[5] J. Kurhinen, V. Korhonen, M. Vapa, and M. Weber, "Modelling mobile encounter networks," in *17th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, 2006.

[6] R. Shah, S. Roy, S. Jain, and W. Brunette, "Data MULEs: modeling a three-tier architecture for sparse sensor networks," in *First IEEE International Workshop on Sensor Network Protocols and Applications*, 2003.

[7] T. Spyropoulos, K. Psounis, and C. S. Raghavendra, "Spray and wait: Efficient routing scheme for intermittently connected mobile networks," in *ACM SIGCOMM workshop on Delay Tolerant Networking*, 2005.

[8] T. Spyropoulos, K. Psounis, and C. S. Raghavendra, "Performance analysis of mobility-assisted routing," in *7th ACM international symposium on Mobile ad hoc networking and computing*, 2006.

[9] T. Spyropoulos, K. Psounis, and C. S. Raghavendra, "Spray and focus: efficient mobility-assisted routing for heterogeneous and correlated mobility," in *Fifth annual IEEE international conference on Pervasive computing and communications workshop*, 2007.

[10] A. Vahdat and D. Becker, "Epidemic Routing for Partially-Connected Ad Hoc Networks," *Technical Report CS-200006*. Duke University, 2000.

[11] M. Vapa, N. Kotilainen, A. Auvinen, H. Kainulainen, and J. Vuori. "Resource discovery in P2P networks using evolutionary neural networks," in *International Conference on Advances in Intelligent - Systems Theory and Applications*, 2004.

[12] O. Volovikov, T. Juonoja, M. Weber, N. Kotilainen, M. Vapa, and J. Vuori, "Mobile encounter networks and their applications," in *5th IEEE Consumer Communications and Networking Conference*, 2008.

PVI

RESOURCE DISCOVERY IN P2P NETWORKS USING EVOLUTIONARY NEURAL NETWORKS

by

Mikko Vapa, Niko Kotilainen, Annemari Auvinen, Heikki Kainulainen and Jarkko Vuori 2004

In Proceedings of the 2004 International Conference on Advances in Intelligent Systems - Theory and Applications

Resource Discovery in P2P Networks Using Evolutionary Neural Networks

Mikko A. VAPA, Niko P. KOTILAINEN, Annemari K. AUVINEN, Heikki M. KAINULAINEN, and Jarkko T. VUORI

***Abstract*-- Resource discovery is an essential problem in peer-to-peer networks since there is no centralized index in which to look for information about resources. One solution for the problem is to use a search algorithm that locates resources based on the local knowledge about the network. Traditionally, the search algorithms have been based on few simple rules, which often reduces the performance from optimal. In this paper, we describe the results of a process where evolutionary neural networks are used for finding an efficient search algorithm from a class of local search algorithms. The initial test results indicate that an evolutionary optimization process can produce search algorithm candidates that are competent compared to the breadth-first search algorithm (BFS) used in Gnutella peer-to-peer network.**

***Index Terms*-- resource discovery, peer-to-peer networks, multi-layer perceptrons, genetic algorithms.**

I. Introduction

In the *resource discovery problem*, any node can possess resources and query these resources from other nodes in the network. The problem consists of graph with nodes, links and resources. Resources are identified by unique IDs and nodes may contain any number of resources. One node knows only the resources it is currently hosting. Any node in the graph can start a query, which means that some of the links are traversed based on a local decision in the graph. Whenever the query reaches the node with the queried ID, the node replies. The goal is to locate a predetermined amount of resource instances with a given ID using as few query packets as possible.

One possible solution for the resource discovery problem is the breadth-first search algorithm (BFS) [1]. In BFS a node that starts a query passes the query to all its neighbors. When the neighbors receive the query, they pass it further to all their neighbors except the one from which the query was received. Nodes cache the messages that they have received and if the query has already been received from other neighbor then query is dropped. Time-to-Live (TTL) value is used to limit the number of hops the query can take by reducing TTL value each time a query is received. When TTL decreases to zero the query is dropped. The BFS algorithm ensures that if a resource is located in the network it can be found from the network if TTL is high enough. The downside of the algorithm, however, is that it uses many query packets to find the needed resources. Thus, we propose an alternative algorithm that is more efficient in face of used query packets and evaluate it using peer-to-peer scenario with power-law distributed topology [2].

The rest of this paper is organized as follows. The next section presents the references to related work done in P2P resource discovery. Section III describes the NeuroSearch algorithm as a solution for the resource discovery problem. Section IV describes the optimization process and Section V the test case used in the study. Section VI analyzes the simulation results and in Section VII the paper is concluded.

II. Related Work

Much research has been done regarding the resource discovery problem. Adamic et al. [3] and Kim et al. [4] propose a search strategy that utilizes the topological properties of a power-law network. The search strategy first proceeds towards highest-degree node, e.g. the node that has the highest number of neighbors, and then gradually moves to lower degree ones. The algorithm locates resources efficiently if they can be found from the core of the network, but the performance decreases when the central nodes are revisited in search for lower degree nodes.

Lv et al. [5] evaluate BFS, expanding ring and random walk search mechanisms with varying topologies, including random graphs [2], power-law graphs and a snapshot of the Gnutella network obtained in October 2000. These researchers find that BFS is not scalable and in particular on Gnutella and power-law graphs the effects of flooding are disastrous: the number of messages increases drastically when TTL is increased. Expanding ring, where TTL is extended gradually for BFS, is the first aid to the problem. However, because it forwards duplicate messages to the nodes that the query has already reached, a better solution to the problem using random walkers is proposed by the researchers. A search initiates multiple walkers and forwards them based on a random selection of a neighbor. In addition to the TTL as a termination condition for the walkers, Lv et al. use checking, where the random walkers periodically check from the query originator whether the

Manuscript received September 2, 2004. This work was supported in part by the Graduate School in Electronics, Telecommunications and Automation (GETA) and Innovations in Business, Communication and Technology (InBCT) –project of Agora Center.

M. A. Vapa, A. K. Auvinen, and J. T. Vuori are with Department of Mathematical Information Technology, University of Jyväskylä, Finland (e-mail: firstname.lastname@jyu.fi).

N. P. Kotilainen is with Agora Center, University of Jyväskylä, Finland (e-mail: niko.kotilainen@jyu.fi).

H. M. Kainulainen is with WTS Networks, Jyväskylä, Finland (e-mail: heikki.kainulainen@wts.fi)

walker should be terminated or not. While random walkers increase the number of hops and thus latency, they decrease the total traffic because the search proceeds in a depth-first manner.

Kalogeraki et al. [6] consider two search algorithms for the resource discovery problem. The Modified Random BFS Search behaves like BFS, but the neighbors select only a random subset of neighbors for forwarding the query. This reduces traffic, but adjusting the correct size of the subset for various networks may be difficult. The researchers' work uses a random graph in which all the nodes have approximately similar degrees. Thus the performance of the algorithm in power-law graphs cannot be directly determined from the results. In another algorithm they present, called Intelligent Search Mechanism, the nodes keep track of recent query results provided by their neighbors. When a new query arrives, the neighbors are sorted based on the similarity of the query to earlier replies from the neighbor. Because the nodes keep track of the earlier queries, the performance of the algorithm improves as the network evolves.

Yang and Garcia-Molina [7] experimented with many types of directed search strategies based on various heuristics. These heuristics include the number of results returned, shortest average time to satisfaction, smallest average number of hops of received results, the highest number of results returned, shortest message queue, shortest latency and highest degree. Their work suggests that, to minimize the time to satisfaction measure, the best strategy is to pass the query to the neighbor that has had the shortest average time to satisfaction for last ten queries. Also, when considering the bandwidth use, the most reliable measure is the smallest average number of hops of received results for last ten queries. The heuristics used in the study are based on history data collected locally in each node.

Similar use of history data is found from the work by Tsoumakos and Roussopoulos [8]. In their proposal, called Adaptive Probabilistic Search algorithm, neighbors keep track of the success rates of earlier queries and forward random walkers probabilistically, based on the earlier success rate. The algorithm is able to adapt to different query patterns and, therefore, performs better than random walkers.

There are certain limitations in all the approaches described above. First, each of these algorithms uses some control parameters (for example time-to-live, the number of walkers or the proportion of neighbors to forward the query) that can be used to tune the algorithm. For a search algorithm, the number

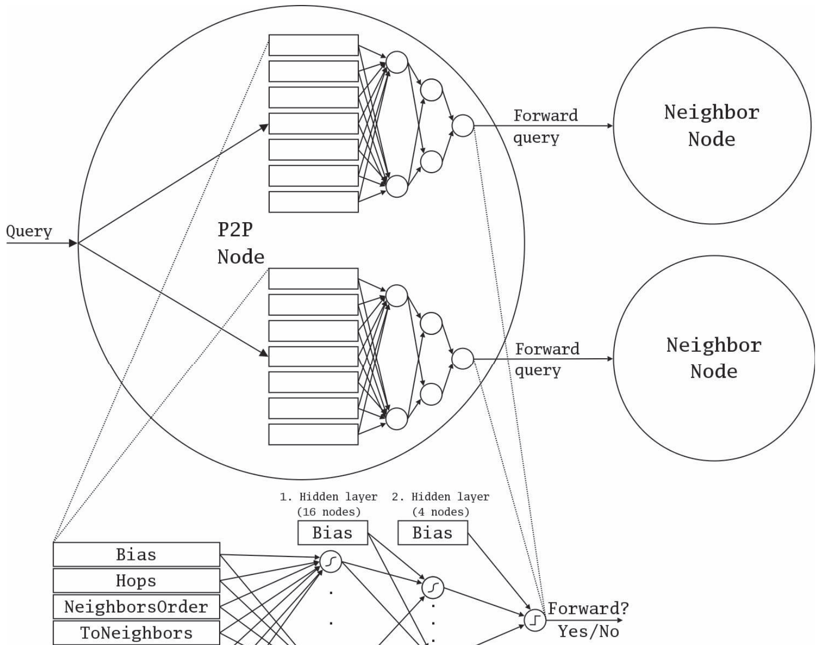


Fig. 1: Processing of NeuroSearch resource query and the NeuroSearch neural network

of control parameters should be kept to a minimal to allow zero configurability when applied to a real environment. Second, while some of these approaches have mechanisms to adapt to the environment, they do not utilize the entire potential of the environment because they rely only on one strategy (for example the similarity of the query and earlier replies, shortest average time to satisfaction for last 10 queries or the success rate of earlier queries). In general, only one strategy cannot be efficient in all scenarios and therefore an efficient algorithm should be able to utilize many strategies at the same time.

To overcome these limitations a neural network based resource discovery algorithm called NeuroSearch was designed. NeuroSearch learns by itself the correct behavior in given network conditions and uses many combinations of strategies to locate resources. To authors' knowledge this is the first time when neural networks are being applied to resource discovery problem.

III. NeuroSearch Resource Discovery Algorithm

The proposed algorithm, called as *NeuroSearch*, makes decision to whom of the node's neighbors the resource request message is forwarded based on the output neuron of three-layer perceptron neural network. The algorithm is located inside a peer node as shown in Fig. 1 and is the same for all peers in the network. NeuroSearch can be represented as a function $O : I \to \{0,1\}$, where $I \in [0,1]^7$ is a 7-dimensional input vector representing the state of a resource discovery query. The output of O defines whether in a given state query should be dropped O = 0 or forwarded to a peer O = 1 and is evaluated for each neighbor peer separately.

When a resource request arrives to the algorithm it goes through all the node's neighbors (denoted as receivers) one by one with the neural network. The input parameters for the neural network are:

- *Bias* is the bias term and has value 1.
- *Hops* is the number of hops the message has travelled.
- *NeighborsOrder* indicates in which rank this receiver is in terms of number of neighbors compared to other neighbors. The connection with highest rank has the value of 0, second rank has the value of 1 and so on.
- *ToNeighbors* is the number of the receiver's neighbors.
- *CurrentNeighbors* is the number of node's neighbors.
- *Sent* has value 1 if the message has already been forwarded to the receiver. Otherwise it has value of 0.
- *Received* has value 1 if the message has been received earlier, else it has value of 0.

Hops and *NeighborsOrder* are scaled with the function $f(x) = \frac{1}{x+1}$ and *Neighbors* and *CurrentNeighbors* with $f(x) = \frac{1}{x}$ before giving them to the neural network. Scaling is performed to ensure that all the inputs are between 0 and 1.

There are two hidden layers in the network. In the first hidden layer there are 15 nodes + bias and in the second hidden layer 3 nodes + bias. Tanh is used as an activation function in the hidden layers: $t(a) = \frac{2}{1+e^{-2a}} - 1$, where a is the weighted sum of inputs to a neuron. Activation function in the output node is the threshold function $s(a) = \begin{cases} 0, a < 0 \\ 1, a \geq 0 \end{cases}$.

Combining all together, the output O of the neural network can be calculated with the following formula:

$$O = s(1 + \sum_{k=1}^{4} w_{3k} t(1 + \sum_{j=1}^{16} w_{2j} t(\sum_{i=1}^{7} w_{1i} f(I_i)))),$$

where I_i is the value of input parameter i and w_{xy} the neural network weights on layer x in position y.

Whenever the query locates a queried resource a reply message is sent back to the neighbor, which forwarded the request to the node. When all the nodes in the query path have forwarded the reply message backward, it is finally received by the query initiator.

IV. Neural Network Optimization

The weights w_{xy} are unknown and therefore they need to be adjusted to appropriate values. For doing this we use methods of evolutionary computing [9]. The decision, which neural networks are better than the others is done by counting the query packets traversed in the test network and found resources. The fitness for the neural network is defined in two parts. Each query j is scored for the neural network h and the fitness is calculated by summing up all the scores after n queries: $fitness_h = \sum_{j=1}^{n} score_j$. The *score* is defined with the following conditions:

1. If *packets* > 300 then *score* = 0
2. If *foundResources* = 0 then *score* = $1 - \frac{1}{packets + 1}$
3. If *foundResources* < *availableResources* / 2 and *foundResources* > *0* then *score* = 50 × *foundResources* – *packets*
4. If *foundResources* ≥ *availableResources* / 2 then *score* = 50 × *availableResources* / 2 – *packets*

In the equations *availableResources* is the maximum number of resource intances that can be located in the query, *foundResources* is the number of resource instances that the neural network was able to locate for the query, and *packets* is the number of query packets the neural network used for the query. The constant value 300 was set as criterion for determining when the neural network is considered to forward the query indefinitely and the query can be stopped. Another constant value, 50, was selected to be large enough to guide the training process towards neural networks that locate more resources than other neural networks. Now a neural network could spend 49 query packets more in a query to locate one additional resource compared to other neural network, which located one resource less.

The first rule ascertains that an algorithm that eventually stops is always better than algorithm that does not. The goal of finding half of the available resource instances was set to demonstrate the algorithm's ability to balance on a predetermined quality of service level and not just on locating all resource instances or one resource instance. The second rule makes sure that if none of the resources are found then the neural network should increase the number of query packets sent to the network. The third rule states that if the number of found resources is not enough then the neural network develops only by locating more resources. Finally the last rule ensures that when half of the available resource instances are found from the network the fitness grows if neural network uses fewer query packets.

The optimization process had an initial population of 30 neural networks whose weights were randomly defined from interval [-0.2, 0.2]. Next, every neural network was tested in the peer-to-peer simulation environment and fitness value calculated. When all neural networks had been tested 15 best were chosen for mutation and used to breed the new generation of neural networks. As a result, 30 neural networks were available for testing the new generation.

Mutation was based on the Gaussian random variation and used weighted mutation parameter to improve the adaptability of the evolutionary search. The random variation function was similar to the one used by Fogel and Chellapilla in their research [10] and is given as:

$$\sigma_i^{'}(j) = \sigma_i(j)\exp(\tau N_j(0,1)),\ j = 1,\dots,N_w,$$
$$w_i^{'}(j) = w_i(j) + \sigma_i^{j}(j) N_j(0,1),\ j = 1,\dots,N_w,$$

where $N_w =$ is the total number of weights and bias terms in the neural network, $\tau = \frac{1}{\sqrt{2\sqrt{N_w}}}$, $N_j(0,1)$ is a standard Gaussian random variable resampled for every *j*, σ is the self-adaptive parameter vector for defining the step size for finding the new weight, $w_i^{'}(j)$ is the new weight value and index $1 \le i \le 185$ denotes the number of neuron enumerated over all layers.

V. Simulation Environment

As a peer-to-peer simulation environment, we used Peer-to-Peer Realm (P2PRealm) network simulator [11] that we have developed. The simulator can be used to simulate the behavior of a static peer-to-peer network and to train neural networks using Gaussian random variation. P2PRealm has been implemented using Java.

In the test case we used power-law graphs generated using the Barabási-Albert model [12]. A power-law network's neighbor distribution follows the power-curve $P(k) = \frac{1}{k^{\gamma}}$, where $\gamma = 3$ for Barabási-Albert graph. Therefore in power-law networks there exist few hubs in the network that have many neighbors as well as many nodes that have only few neighbors. A power-law graph was selected because existing P2P networks have shown to express power-law dependencies [13]. The graphs tested contained 100 nodes with the highest degree node having 25 neighbors. Small network size was selected to allow visualisation of query paths in the network. Dynamic changes e.g., node failures were not taken into account to simplify the analysis. However, the approach can be applied in dynamic scenarios also as shown in [14].

The test case data was divided into three distinct data sets as described in [15]: a training set, a generalization set and a validation set. Training set is used for training the neural network. Generalization set is used to measure how well the trained neural network performs with a new data set indicating neural network's ability to generalize. When performance starts to decrease in generalization set the training can be stopped, because the neural network adapts only to the training set if training process is continued. Validation set is used as an objective measure to verify how well the algorithm performs with arbitrarily chosen new data set and ensures that the true generalization ability of the neural network is being measured.

The training set contained two power-law topologies with both being queried *n* = 50 times per generation for each neural network. Two topologies were used to have neural networks adapt to a wider range of situations than one topology would have provided. The generalization set consisted of two power-law topologies with 50 queries. When the performance started to decrease in the generalization set the neural network having highest fitness was selected and, as a validation set, one topology with 100 queries was used to produce the final simulation results.

For each topology, resource instances were allocated based on the number of neighbors each node has. There were 25 different resources in the test case and the number of different resources in a node was the same as the number of neighbors the node had. This means that the largest hub had one instance of all resources and the lower degree nodes only some of these, randomly chosen from uniform distribution. The querying nodes and queried resources were selected also randomly from a uniform distribution for each query.

As stopping criteria for the optimization process, 100,000 generations were set. This seemed to take approximately two

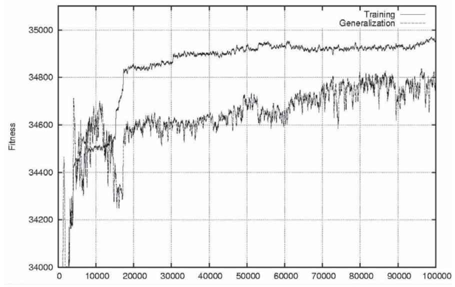


Fig. 2: Evolution of the best neural networks in each generation for training and generalization sets

weeks on our desktop PC equipped with an AMD Athlon XP 1800 processor. The evolution of the best neural network in each generation is shown in Fig. 2.

VI. Simulation Results

To evaluate the difference between BFS and NeuroSearch, we selected the best algorithm at the 85,736th generation and calculated the number of packets used and found resources for 100 different queries using validation set. The 85,736th generation was selected because between the 80,000 and 90,000 generations the neural networks had achieved steadily good results and, in particular, in the 85,736th generation, neural network had the best fitness. The results are presented in Fig. 3 and Fig. 4.

The results of Fig. 3 show that the performance of NeuroSearch regarding the number of packets is nearer to BFS with a time-to-live value 2 (BFS-2), rather than BFS with a time-to-live value 3 (BFS-3). In average NeuroSearch consumes 47.2 packets per query whereas BFS-2 consumes 30.0 and BFS-3 122.0 packets. The reason why there is some variation in the number of packets for successive BFS queries is that the number of delivered packets depends on which node is querying. If the query starts from a central node (nodes 0-10), it will produce more packets than the same query started from an edge node (nodes 90-99) because the edge query has fewer connections where BFS can spread. In case of NeuroSearch, the performance is stable and does not depend on which node is querying.

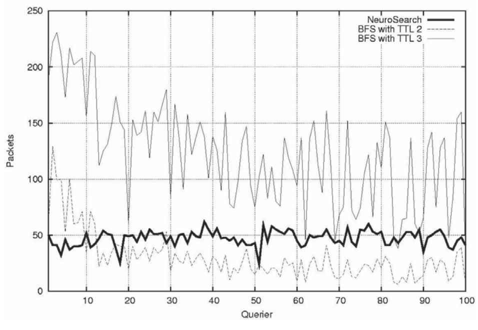


Fig. 3: Number of packets used by the algorithms

Fig. 4 shows how many resources the algorithms were able to locate. NeuroSearch's performance in terms of located resources is quite similar to BFS-2 at central nodes, but better in the edge nodes. Compared to BFS-3 NeuroSearch's performance is constantly lower, reaching the same performance level only at some edge nodes. The reason why NeuroSearch is satisfied with this level of performance is that it has already reached the goal of finding half of the available resources as defined in the fitness function and locating more resources is not needed.

By calculating the ratio between the located resources and

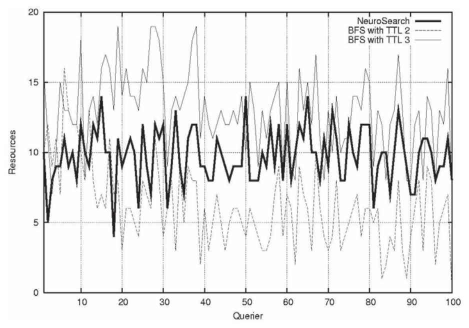


Fig. 4: Number of resources found by the algorithms

used query packets we can determine the efficiency of the algorithms. These values are shown in Table I. The results show that NeuroSearch's efficiency is at the same level as BFS-2's locating a new resource every fifth packet. BFS-3 locates a new resource approximately every ninth packet. Efficiency is easier to keep high when locating only few resources because usually those can be found from the central nodes alone. When the number of needed resources increases, query has to spread more to the edges to locate the additional resources. Therefore the efficiency of BFS-3 decreases significantly. BFS-2 and NeuroSearch achieve near similar efficiency indicating that NeuroSearch is able to sustain a good efficiency even though it needs to locate more resources than BFS-2.

TABLE I
Efficiency of the Algorithms

Algorithm	Packets	Resources	Efficiency
BFS-2	3000	619	0.2063
BFS-3	12202	1295	0.1061
NeuroSearch	4719	975	0.2066

For each query, NeuroSearch locates approximately half of the resources or more, which can be seen in Fig. 5. There are six queries in which NeuroSearch misses the target to locate half of the resources. This variation results from the difference

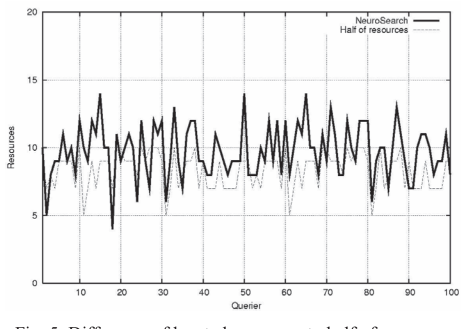


Fig. 5: Difference of located resources to half of resources

between the training set and the validation set. Nonetheless, the results indicate that the optimization process has found an algorithm that is able to locate nearly half of the resources from the network with high probability.

We analyzed the behavior of the best-evolved neural network by tracking the path used by the queries. NeuroSearch seems to prefer central nodes early in the query and uses multiple paths for doing this. After reaching central nodes or one hop later the spreading is stopped. The maximum number of hops is 5. As verification for this the behavior of a typical NeuroSearch query started from an edge node is illustrated in Fig. 5. In the figure the query travels through the connections denoted with a black line starting from node 99 with question mark (?). Nodes marked with an exclamation mark (!) contain the queried resource. In total the query uses 49 packets and locates 11 resources. Six connections are traversed from both directions, which is not shown in the figure.

VII. Conclusion

In this paper, a new resource discovery algorithm has been proposed. NeuroSearch algorithm takes into account the special characteristics of its environment and can be adjusted to different kind of P2P networks. The algorithm's performance is also stable and competitive compared to the BFS algorithm.

While NeuroSearch performs well compared to BFS it is by no means yet designed to be optimal. For example, NeuroSearch does not yet include history-based inputs even though they would significantly improve the performance. Therefore, the results obtained in [3]-[8] will be considered in forthcoming research on NeuroSearch. There are also other directions that were left out of this research. First, we are studying what improvements to the performance would be gained by varying the neural network's internal structure. Second, we are aiming to find out what are the scalability factors of NeuroSearch when the network size grows, and third we are developing an optimal resource discovery algorithm using global knowledge to be able to measure the best efficiency a resource discovery algorithm can achieve. Also, we are working on a solution to speed up the optimization process by parallelizing the evolutionary algorithm using distributed computing. This helps us to more accurately determine the performance maximum of NeuroSearch.

Acknowledgment

The authors would like to thank the co-designers of NeuroSearch Joni Töyrylä, Yevgeniy Ivanchenko, Matthieu Weber and Hermanni Hyytiälä. Also we thank Tommi Kärkkäinen for giving useful hints how to develop the algorithm further and Barbara Crawford for proofreading the article.

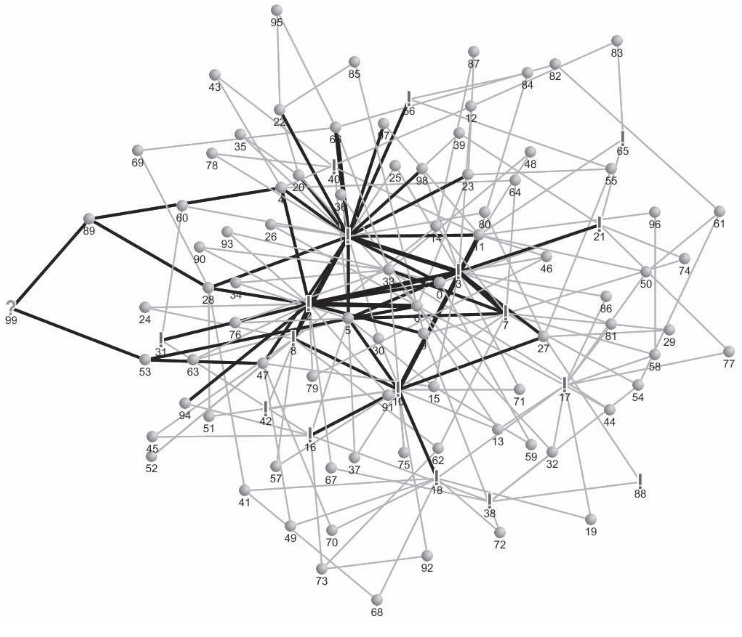


Fig. 5: Typical NeuroSearch resource query

REFERENCES

[1] N. A. Lynch, *Distributed Algorithms*, Morgan Kauffmann Publishers, 1996.

[2] A. Barabási, *Linked*, Perseus Publishing, 2002.

[3] L. A. Adamic, R. M. Lukose, and B. A. Huberman, "Local Search in Unstructured Networks", in *Handbook of Graphs and Networks: From the Genome to the Internet*, Wiley-VCH, 2003, pp. 295-317.

[4] B. J. Kim, C. N. Yoon, S. K. Han, and H. Jeong, "Path finding strategies in scale-free networks", *Physical Review E 65*, 2002.

[5] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker, "Search and Replication in Unstructured Peer-to-Peer Networks", in *Proceedings of the 16th International Conference on Supercomputing*, ACM Press, 2002, pp. 84-95.

[6] V. Kalogeraki, D. Gunopulos, and D. Zeinalipour-Yatzi, "A Local Search Mechanism for Peer-to-Peer Networks", in *Proceedings of the 11th International Conference on Information and Knowledge Management*, ACM Press, 2002, pp. 300-307.

[7] B. Yang and H. Garcia-Molina, "Improving search in peer-to-peer networks," in *Proceedings of the 22nd IEEE International Conference on Distributed Computing Systems (ICDCS'02)*, 2002.

[8] D. Tsoumakos and N. Roussopoulos, "Adaptive Probabilistic Search for Peer-to-Peer Networks", in *Proceedings of the Third IEEE International Conference on P2P Computing (P2P2003)*, IEEE Press, 2003, pp. 102-109.

[9] K. Miettinen, M. Mäkelä, and P. Neittaanmäki and J. Périaux (eds.), *Evolutionary algorithms in engineering and computer science*, John Wiley & Sons, 1999.

[10] K. Chellapilla and D. Fogel, "Evolving neural networks to play checkers without relying on expert knowledge", *IEEE Trans. on Neural Networks*, 10 (6), pp. 1382-1391, 1999.

[11] J. Töyrylä, *Building NeuroSearch – Intelligent Evolutionary Search Algorithm For Peer-to-Peer Environment*, Master's Thesis, University of Jyväskylä, 2004.

[12] A.-L. Barabási and R. Albert, "Emergence of Scaling in Random Networks", *Science 286* (1999) 509-512.

[13] M. A. Jovanovic, F. S. Annexstein, and K. A. Berman, *Scalability Issues in Large Peer-to-Peer Networks – A Case Study of Gnutella*, Technical report, University of Cincinnati, 2001.

[14] Y. Ivanchenko, *Adaptation of Neural Nets For Resource Discovery Problem in Dynamic And Distributed P2P Environment*, Master's Thesis, University of Jyväskylä, 2004.

[15] A. P. Engelbrecht, *Computational Intelligence: An Introduction*, John Wiley & Sons Ltd, 2002.

PVII

AN ADAPTIVE GLOBAL-LOCAL MEMETIC ALGORITHM TO DISCOVER RESOURCES IN P2P NETWORKS

by

Ferrante Neri, Niko Kotilainen and Mikko Vapa 2007

In Applications of Evolutionary Computing, volume 4448 of *Lectures Notes in Computer Science*, pages 61-70

An Adaptive Global-Local Memetic Algorithm to Discover Resources in P2P Networks

Ferrante Neri[1,2], Niko Kotilainen[1], and Mikko Vapa[1]

[1] Department of Mathematical Information Technology, Agora,
University of Jyväskylä, FI-40014, Finland
`{neferran,npkotila,mikvapa}@jyu.fi`

[2] Dipartimento di Elettrotecnica ed Elettronica, Politecnico di Bari,
Via E. Orabona 4, 70125, Italy
`neri@deemail.poliba.it`

Abstract. This paper proposes a neural network based approach for solving the resource discovery problem in Peer to Peer (P2P) networks and an Adaptive Global Local Memetic Algorithm (AGLMA) for performing the training of the neural network. This training is very challenging due to the large number of weights and noise caused by the dynamic neural network testing. The AGLMA is a memetic algorithm consisting of an evolutionary framework which adaptively employs two local searchers having different exploration logic and pivot rules. Furthermore, the AGLMA makes an adaptive noise compensation by means of explicit averaging on the fitness values and a dynamic population sizing which aims to follow the necessity of the optimization process. The numerical results demonstrate that the proposed computational intelligence approach leads to an efficient resource discovery strategy and that the AGLMA outperforms two classical resource discovery strategies as well as a popular neural network training algorithm.

1 Introduction

During recent years the use of peer-to-peer networks (P2P) has significantly increased and thus demand of high performance peer-to-peer networks is constantly growing. In order to obtain proper functioning of a P2P network a crucial point is to efficiently execute the P2P resource discovery, since an improper resource discovery strategy would lead to overwhelming query traffic and consequently to a waste of bandwidth for each single user.

This problem has been intensively analyzed and several solutions have been proposed in commercial packages and scientific literature. The solutions so far proposed can be classified into two categories: breadth-first search (BFS) and depth-first search (DFS). BFS strategies forward a query to multiple neighbors at the same time whereas DFS strategies forward only to one neighbor.

BFS strategies have been used in Gnutella, where the query is forwarded to all neighbors and the forwarding is controlled by a time-to-live parameter. This parameter is defined as the amount of hops required to forward the query. Two nodes are said to be n hops apart if the shortest path between them has length

n [1]. The main disadvantage of the Gnutella's mechanism is that it generates a massive traffic of query messages when the time-to-live parameter is high. In order to reduce query traffic, Lv et al. [2] proposed the *Expanding Ring*. This strategy establishes that the time-to-live parameter is gradually increased until enough resources have been found. Although use of the *Expanding Ring* is beneficial in terms of query packet reduction, it introduces some delay to resource discovery and thus implies a longer waiting time for the user. Kalogeraki et al. [3] and Menascé [4] proposed that only a subset of neighbors are selected randomly for forwarding. While in [3] a mechanism is proposed which stores the performance of the queries previously done for each neighbor and then uses this memory to direct subsequent queries, in [4] the earlier replies are cached in directory entries and queried prior to using broadcast probability. Yang and Garcia-Molina [1] proposed to heuristically select the first neighbor and further uses BFS for forwarding the query. In Gnutella2 a trial query is sent to the neighbors and estimates how widely the actual query should be forwarded.

In the DFS strategies, selection of the neighbor for query forwarding is performed by means of heuristics. Lv et al. [2] studied the use of multiple random walkers which periodically check the query originator in order to verify whether the query should be forwarded further. Tsoumakos and Roussopoulos [5] proposed using the feedback from previous queries in order to tune probabilities for further forwarding of random walkers. Crespo and Garcia-Molina [6] proposed routing indices, which provide shortcuts for random walkers in locating resources. Sarshar et al. [7] proposed replicating a copy of resources and thus ensure that resource discovery strategy locates at least one replica of the resource.

The main limitation of the previous studies, for both BFS and DFS strategies, is that all the approaches are restricted to only one search strategy. On the contrary, for the same P2P network, in some conditions it is preferable to employ both BFS and DFS strategies. In order to obtain a flexible search strategy, which intelligently takes into account the working conditions of the P2P network, Vapa et al. [8] proposed a neural network based approach (NeuroSearch) which adaptively combines BFS and DFS. In NeuroSearch, a trained neural network is able to map a specific input set to forward decisions in an if-then logic. Thanks to this logic, the resource discovery strategy can be applied also in devices with limited computing power. On the other hand, training neural networks to adapt to various conditions is challenging since it requires training in multiple topological scenarios thus leading to complicated computational requirements. It is therefore fundamental to investigate efficient training algorithms which lead to high performance in a short training time.

2 Problem Description

NeuroSearch [8] is a neural network-based approach which combines different local information units together as an input to multi-layer perceptron (MLP) neural network [9]. The neural network employed in NeuroSearch contains two hidden layers, both having 10 neurons and two different transfer functions in

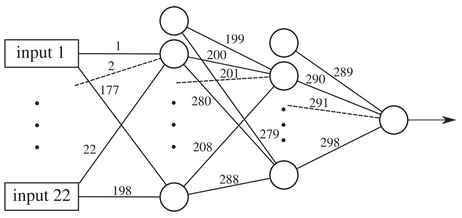


Fig. 1. MLP Neural Network

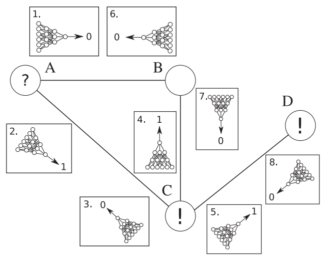


Fig. 2. Query Forwarding

hidden and output layers. The structure of this neural network (see Fig. 1) has been selected on the basis of previous studies carried out by means of the P2PRealm simulation framework [10]. Details regarding the functioning of this neural network are given in [8] and [10]. We characterize the query forwarding situation with a model consisting of 1) the previous forwarding node, 2) the currently forwarding node and 3) the receiver of the currently forwarding node. Upon receiving a query, the currently forwarding node selects the first of its neighbors and determines the inputs, related to that neighbor, of the neural network. The neural network output is then calculated. This output establishes whether or not the query will be forwarded to the neighbor. Next, all other neighbors including the previous forwarding node, are processed in a similar manner by means of the same neural network. Fig. 2, shows an example of the functioning of a P2P network with neural network based forwarding. The circles shown in the figure represent peers of the P2P network. The arcs between the peers represent the Transmission Control Protocol communication links between the peers. The rectangles represent a neural network evaluation for different neighbors. This paper addresses the problem in the training of a neural network (i.e. the determination of the set of weight coefficients W) of the kind in Fig. 1 with the aim summarized in Fig 2. As shown in Fig. 1, the weights can be divided into three categories on the basis of the layer to which they belong to. There are 22 input neurons and 10 neurons on both the hidden layers. Since one input is constant (*Bias*, see [8]) the total amount of weights is $22 * 9 + 10 * 9 + 10 = 298$. The weights can take values within the range $(-\infty, \infty)$. In order to estimate the quality of a candidate solution, the performance of the P2P network is analyzed with the aid of a simulator whose working principles are described in [10] and a certain number n of queries are performed. For each query, the simulator returns two outputs: the number of query packets P used in the query and the number of found resource instances R during the query. At each j^{th} query, these outputs are combined in the following way and F_j is determined:

$$F_j = \begin{cases} 0 & \text{if } P > 300 \\ 1 - \frac{1}{P+1} & \text{if } P \leq 300 \text{ AND } \quad R = 0 \\ 50 * R - P & \text{if } P \leq 300 \text{ AND } 0 < R < \frac{AR}{2} \\ 50 * \frac{AR}{2} - P & \text{if } P \leq 300 \text{ AND } \quad \frac{AR}{2} < R \end{cases} \tag{1}$$

In (1), the amount of Available Resources (AR) instances is constant at each query and the constant values 300 and 50 have been set according to the criterion explained in [8]. It must be noted that due to its formulation each F_j could likely contain several plateaus (see (1)). The total fitness over the n queries is given by $F = \sum_{j=1}^{n} F_j(W)$. It is important to remark that multiple queries ($n = 10$) are needed in order to ensure that the neural network is robust in different query conditions. The querying peer and the queried resource need to be changed to ensure that the neural network is not only specialized for searching resources from one part of the network or one particular resource alone. Therefore, two consecutive fitness evaluations do not produce the same fitness value for the same neural network. Since n queries are required and, for each query, the first forwarding node is chosen at random, fitness F is noisy. This noise is not Gaussian. Let us indicate with $PN(n)$ the distribution of this noise and thus formulate the optimization problem addressed in this paper:

$$\max \left(F\left(W\right) + Z\right) \text{in} \left(-\infty, \infty\right)^{298}; Z \sim PN\left(n\right) \tag{2}$$

3 The Adaptive Global-Local Memetic Algorithm

In order to solve the problem in (2), the following Adaptive Global-Local Memetic Algorithm (AGLMA) has been implemented.

Initialization. An initial sampling made up of S_{pop}^{i} individual has been executed pseudo-randomly with a uniform distribution function over the interval $[-0.2, 0.2]$. This choice can be briefly justified in the following way. The weights of the initial set of neural networks must be small and comparable among each other in order to avoid one or a few weights dominating with respect to the others as suggested in [11], [12].

Parent Selection and Variation Operators. All individuals of the population S_{pop} undergo recombination and each parent generates an offspring. The variation occurs as follows. Associated with each candidate solution i is a self-adaptive vector h_i which represents a scale factor for the exploration. More specifically, at the first generation the self-adaptive vectors h_i are pseudo-randomly generated with uniform distribution within $[-0.2, 0.2]$ (see [11], [12]).

At subsequent generations each self-adaptive vector is updated according to [11], [12]:

$$h_i^{k+1}\left(j\right) = h_i^{k}\left(j\right) e^{\left(\tau N_j(0,1)\right)} \quad \text{for} \quad j = 1, 2 \ldots n \tag{3}$$

where k is the index of generation, j is the index of variable ($n = 298$), $N_j\left(0,1\right)$ is a Gaussian random variable and $\tau = \frac{1}{\sqrt{2\sqrt{n}}} = 0.1659$. Each corresponding candidate solution W_i is then perturbed as follows [11], [12]:

$$W_i^{k+1}\left(j\right) = W_i^{k} + h_i^{k+1}\left(j\right) N_j\left(0,1\right) \quad \text{for} \quad j = 1, 2 \ldots n \tag{4}$$

Fitness Function. In order to take into account the noise, function F is calculated n_s times and an *Explicit Averaging* technique is applied [13]. More specifically, each set of weights for a neural network (candidate solution) is evaluated by means of the following formula:

$$\hat{F} = F^i_{mean} - \frac{\sigma^i}{\sqrt{n_s}} \tag{5}$$

where F^i_{mean} and σ^i are respectively the mean value and standard deviation related to the n_s samples performed to the i^{th} candidate solution.

The penalty term $\frac{\sigma^i}{\sqrt{n_s}}$ takes into account distribution of the data and the number of performed samples [14]. Since the noise strictly depends on the solution under consideration, it follows that for some solutions the value of σ^i is relatively small (stable solutions) and so penalization is small. On the other hand, other solutions could be unstable and score 0 during some samples and give a high performance value during other samples. In these cases σ^i is quite large and the penalization must be significant.

Local Searchers. Two local searchers with different features in terms of search logic and pivot rule have been employed. These local searchers have the role of supporting the evolutionary framework, offering new search directions and exploiting the available genotypes [15].

1)Simulated Annealing. The Simulated Annealing (SA) metaheuristic [16] has been chosen since it offers an exploratory perspective in the decision space which can choose a search direction leading to a basin of attraction different from starting point W_0 and, thus, prevents an undesired premature convergence. The exploration is performed by using the same mutation scheme as was described in equations (3) and (4) for an initial self-adaptive vector h_0 pseudo-randomly sampled in $[-0.2, 0.2]$.

The main reason for employing the SA in the AGLMA is that the evolutionary framework should be assisted in finding better solutions which improve the available genotype while at the same time exploring areas of the decision space not yet explored. It accepts, with a certain probability, solutions with worse performance in order to obtain a global enhancement in a more promising basin of attraction. In addition, the exploratory logic aims to overcome discontinuities of the fitness landscape and to “jump” into a plateau having better performance. For these reasons the SA has been employed as a “global” local searcher.

2)Hooke-Jeeves Algorithm. The Hooke-Jeeves Algorithm (HJA) [17] is a deterministic local searcher which has a steepest descent pivot rule. The HJA is supposed to efficiently exploit promising solutions enhancing their genotype in a meta-Lamarckian logic and thus assist the evolutionary framework in quickly climbing the basin of attractions. In this sense the HJA can be considered as a kind of “local” local searcher integrated in the AGLMA.

Adaptation. In order to design a robust algorithm [15], at the end of each generation the following parameter is calculated:

$$\psi = 1 - \left| \frac{\hat{F}_{avg} - \hat{F}_{best}}{\hat{F}_{worst} - \hat{F}_{best}} \right| \tag{6}$$

where $\hat{F}_{worst}$, $\hat{F}_{best}$, and $\hat{F}_{avg}$ are the worst, best, and average of the fitness function values in the population, respectively. As highlighted in [18], ψ is a fitness-based measurement of the population diversity which is well-suited for flat fitness landscapes. The employment of this parameter, taking into account the presence of plateaus in the fitness landscape (i.e. areas with a very low variability in the fitness values.) ψ, efficiently measures the population diversity even when the range of variability of all fitness values is very small. The population has high diversity when $\psi \approx 1$ and low diversity when $\psi \approx 0$. A low diversity means that the population is converging (possibly in a suboptimal plateau). We remark that the absolute diversity measure used in [14], [19], [20] and [21] is inadequate in this case, since, according to this, the population diversity would be very low most of the time.

Coordination of the local searchers. The SA is activated by the condition $\psi \in [0.1, 0.5]$. This adaptive rule is based on the observation that for values of $\psi > 0.5$, the population diversity is high and therefore the evolutionary framework needs to have a high exploitation of the available genotypes (see [19], [18] and [21]). On the other hand, if $\psi < 0.5$ the population diversity is decreasing and application of the SA can introduce a new genotype in the population which can prevent a premature convergence. In this sense, the SA has been employed as a local searcher with "global" exploratory features. The condition regarding the lower bound of usability of the SA ($\psi > 0.1$) is due to the consideration that if $\psi < 0.1$ application of the SA is usually unsatisfactory since it most likely leads to a worsening in performance.

Moreover, the SA, in our implementation, is applied to the second best individual. This gives a chance at enhancing a solution with good performance without possibly ruining the genotype of the best solution. The initial temperature $Temp^0$ has been adaptively set $Temp^0 = \left|\hat{F}_{avg} - \hat{F}_{best}\right|$. This means that the probability of accepting a worse solution depends on the state of the convergence. In other words, the algorithm does not accept worse solutions when the convergence has practically occurred.

The HJA is activated when $\psi < 0.2$ and is applied to the solution with best performance. The basic idea behind this adaptive rule is that the HJA has the role of quickly improving the best solution while staying in the same basin of attraction. In fact, although evolutionary algorithms are efficient in detecting a solution which is near the optimum, they are not so efficient in "ending the game" of optimization. In this light, the action of the HJA can be seen as purely "local". The condition $\psi < 0.2$ means that the HJA is employed when there are some chances that optimal convergence is approaching. An early application of this local searcher can be inefficient since a high exploitation of solutions having poor fitness values would not lead to significant improvements of the population.

It should be noted that in the range $\psi \in [0.1, 0.2]$ both local searchers are applied to the best two individuals of the population. This range is very critical for the algorithm because the population is tending towards a convergence but still has not reached such a condition. In this case, there is a high risk of premature convergence due to the presence of plateaus and suboptimal basins of attraction or false minima introduced by noise. Thus, the two local searchers are supposed to "compete and cooperate" within the same generation, merging the "global" search power of the SA and the "local" search power of the HJA. An additional rule has been implemented. When the SA has succeeded in enhancing the starting solution, the algorithm attempts to further enhance it by the application of the HJA under supervision of the evolutionary framework.

Dynamic population size in survivor selection. The population is resized at each generation and the S_{pop} individuals having the best performance are selected for the subsequent generation:

$$S_{pop} = S^{f}_{pop} + S^{v}_{pop} \cdot (1 - \psi) , \tag{7}$$

where S^{f}_{pop} and S^{v}_{pop} are the fixed minimum and maximum sizes of the variable population S_{pop}, respectively.

The dynamic population size has two combined roles. The first is to massively explore the decision space and thus prevent a possible premature convergence (see [19]), the second is to *Implicitly Average* in order to compensate for noise by means of the evaluations of similar individuals [13]. According to the first role, when $\psi \approx 0$ the population is converging and a larger population size is required to increase the exploration and possibly inhibit premature convergence by offering new search directions. On the other hand, if the population is spread out in the decision space it is highly desirable that the most promising solution leads the search and that the algorithm exploits this promising search direction. According to the second role, it is well-known that large population sizes are helpful in defeating the noise [22]. Furthermore, recent studies [14], [23] have noted that the noise jeopardizes functioning of the selection mechanisms especially for populations made up of individuals having similar performance, since the noise introduces a disturbance in pair-wise comparison. Therefore, the AGLMA aims to employ a large population size in critical conditions (low diversity) and a small population size when a massive averaging is unnecessary. The algorithm stops when either a budget condition on the number of fitness evaluations is satisfied or ψ takes a value smaller than 0.01.

4 Numerical Results

For the AGLMA 30 simulation experiments have been executed. Each experiment has been stopped after 1.5×10^6 fitness evaluations. At the end of each generation, the best fitness value has been saved. These values have been averaged over the 30 experiments available. The average over the 30 experiments defines the Average Best Fitness (ABF). Analogously, 30 experiments have been carried out with the Checkers Algorithm (CA) described in [11], [12] according to the implementation in [8], and the proposed here Adaptive Checkers Algorithm (ACA) which is the CA with the fitness as shown in (5) and the adaptive population size as shown in (7). For the same P2P network, the BFS according to the implementation in Gnutella and the random walker DFS proposed in [2] have been applied. Table 1 shows the parameter settings for the three algorithms and the optimization results. The final fitness $\hat{F}^b$ obtained by the most successful experiment (over the 30 sample runs), the related number of query packets P used in the query and the number of found resource instances R during the query are given. In addition the average best fitness at the end of the experiments $< \hat{F} >$, the final fitness of the least successful experiment $\hat{F}^w$ and the related standard deviation are shown. Since the BFS follows a deterministic logic, thus only one fitness value is shown. On the contrary, the DFS under study employs a stochastic structure and thus the same statistic analysis as that of CA, ACA and AGLMA over 30 experiments has been carried out.

Table 1. Parameter setting and numerical results

PARAMETER	AGLMA	CA	ACA	BFS	DFS
EVOLUTIONARY FRAMEWORK					
S^i_{pop}	30	30	30	–	–
S_{pop}	$\in [20, 40]$	30	$\in [20, 40]$	–	–
sample size n_s	10	–	10	–	–
SIMULATED ANNEALING					
initial temperature $Temp^0$	adaptive	–	–	–	–
temperature decrease	hyperbolic	–	–	–	–
maximum budget per run	600	–	–	–	–
HOOKE-JEEVES ALGORITHM					
exploratory radius	$\in [0.5, 0.01]$	–	–	–	–
maximum budget per run	1000	–	–	–	–
NUMERICAL RESULTS					
P	350	372	355	819	514
R	81	81	81	81	81
$\hat{F}^b$	3700	3678	3695	3231	3536
$< \hat{F} >$	3654	3582	3647	–	3363
$\hat{F}^w$	3506	3502	3504	–	3056
std	36.98	37.71	36.47	–	107.9

Numerical results in Table 1 show that the AGLMA and ACA outperform the CA and that the AGLMA slightly outperformed the ACA in terms of the final solution found. Moreover, the AGLMA clearly outperforms the BFS employed in Gnutella and the DFS.

Figures 3 and 4 show the comparison of the performance. As shown, the AGLMA has a slower convergence than the CA and the ACA but reaches a final solution having better performance. It is also clear that the ACA has intermediate performance between the CA and AGLMA. The ACA trend, in early generations, has a rise quicker than the AGLMA but slower than the CA. On the other hand, in late generations, the ACA outperforms the CA but not the AGLMA. Regarding effectiveness of the noise filtering components, Fig. 4 shows that the ACA and the AGLMA are much more robust with respect to noise than the CA. In fact, the trend of the CA performance contains a high amplitude and

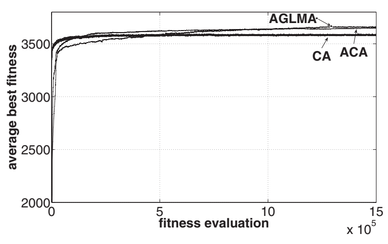


Fig. 3. Algorithmic Performance

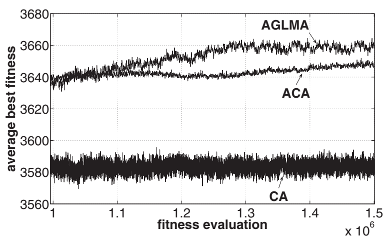


Fig. 4. Performance (zoom detail)

frequency ripple, while the ACA and AGLMA performance are roughly monotonic. Regarding effectiveness of the local searchers, the comparison between the ACA and the AGLMA shows that the AGLMA slightly outperforms the ACA tending to converge to a solution having a better performance.

5 Conclusion

This paper proposes an AGLMA for performing the training of a neural network, which is employed as computational intelligence logic in P2P resource discovery. The AGLMA employs averaging strategies for adaptively executing noise filtering and local searchers in order to handle the multivariate fitness landscape. These local searchers execute the global and local search of the decision space from different perspectives. The numerical results show that the application of the AGLMA leads to a satisfactory neural network training and thus to an efficient P2P network functioning. The proposed neural network along with the learning strategy carried by the AGLMA allows the efficient location of resources with little query traffic. Thus, with reference to classical resource discovery strategies (Gnutella BFS and DFS), the user of the P2P network obtains plentiful amounts of information about resources consuming a definitely smaller portion of bandwidth for query traffic. Regarding performance during the optimization process, comparison with a popular metaheuristic present in literature shows the superiority of the AGLMA in terms of final solution found and reliability in a noisy environment.

References

1. Yang, B., Garcia-Molina, H.: Improving search in peer-to-peer networks. In: Proc. of the 22nd Intern. Conf. on Distributed Computing Systems. (2002) 5–14
2. Lv, Q., Cao, P., Cohen, E., Li, K., Shenker, S.: Search and replication in unstructured peer-to-peer networks. In: Proc. of the 16th ACM Intern. Conf. on Supercomputing. (2002) 84–95
3. Kalogeraki, V., Gunopulos, D., Zeinalipour-Yazti, D.: A local search mechanism for peer-to-peer networks. In: Proc. 11th ACM Intern. Conf. on Information and Knowledge Management. (2002) 300–307

4. Menascé, D.A.: Scalable p2p search. IEEE Internet Computing **7**(2) (2003) 83–87
5. Tsoumakos, D., Roussopoulos, N.: Adaptive probabilistic search for peer-to-peer networks. In: Proc. 3rd IEEE Intern. Conf. on P2P Computing. (2003) 102–109
6. Crespo, A., Garcia-Molina, H.: Routing indices for peer-to-peer systems. In: Proc. of the 22nd IEEE Intern. Conf. on Distributed Computing Systems. (2002) 23–33
7. Sarshar, N., Boykin, P.O., Roychowdhury, V.P.: Percolation search in power law networks: Making unstructured peer-to-peer networks scalable. In: Proc. of the IEEE 4th Intern. Conf. on P2P Computing. (2004) 2–9
8. Vapa, M., Kotilainen, N., Auvinen, A., Kainulainen, H., Vuori, J.: Resource discovery in p2p networks using evolutionary neural networks. In: Intern. Conf. on Advances in Intelligent Systems - Theory and Applications, 067-04. (2004)
9. Engelbrecht, A.: Computational Intelligence-An Introduction. J. Wiley (2002)
10. Kotilainen, N., Vapa, M., Keltanen, T., Auvinen, A., Vuori, J.: P2prealm - peer-to-peer network simulator. In: IEEE Intern. Works. on Computer-Aided Modeling, Analysis and Design of Communication Links and Networks. (2006) 93–99
11. Chellapilla, K., Fogel, D.: Evolving neural networks to play checkers without relying on expert knowledge. IEEE Trans. Neural Networks, **10**(6) (1999) 1382–1391
12. Chellapilla, K., Fogel, D.: Evolving an expert checkers playing program without using human expertise. IEEE Trans. Evol. Computation, **5**(4) (2001) 422–428
13. Jin, Y., Branke, J.: Evolutionary optimization in uncertain environments - a survey. IEEE Transactions on Evolutionary Computation **9**(3) (2005) 303–317
14. Neri, F., Cascella, G.L., Salvatore, N., Kononova, A.V., Acciani, G.: Prudent-daring vs tolerant survivor selection schemes in control design of electric drives. In Rothlauf, F. et al., ed.: Applications of Evolutionary Computing, LNCS. Volume 3907., Springer (2006) 805–809
15. Krasnogor, N.: Toward robust memetic algorithms. In W. E. Hart et al., ed.: Recent Advances in Memetic Algorithms, Springer (2004) 185–207
16. Cerny, V.: A thermodynamical approach to the traveling salesman problem. Journal of Optimization, Theory and Applications **45**(1) (1985) 41–51
17. Hooke, R., Jeeves, T.A.: Direct search solution of numerical and statistical problems. Journal of the ACM, **8** (1961) pp. 212–229
18. Neri, F., Toivanen, J., Cascella, G.L., Ong, Y.S.: An adaptive multimeme algorithm for designing hiv multidrug therapies. IEEE/ACM Transactions on Computational Biology and Bioinformatics, Special Issue on Computational Intelligence Approaches in Computational Biology and Bioinformatics (2007) to appear.
19. Caponio, A., Cascella, G.L., Neri, F., Salvatore, N., Sumner, M.: A fast adaptive memetic algorithm for on-line and off-line control design of PMSM drives. IEEE Trans. on System Man and Cybernetics-part B 37(1) (2007) 28–41.
20. Neri, F., Toivanen, J., Mäkinen, R.A.E.: An adaptive evolutionary algorithm with intelligent mutation local searchers for designing multidrug therapies for HIV. Applied Intelligence, Springer (2007) to appear.
21. Neri, F., Mäkinen, R.A.E.: Hierarchical evolutionary algorithms and noise compensation via adaptation. In S. Yang et al., ed.: Evolutionary Computation in Dynamic and Uncertain Environments, Springer (2007) to appear.
22. Miller, B.L., Goldberg, D.E.: Genetic algorithms, selection schemes, and the varying effects of noise. Evolutionary Computation **4**(2) (1996) 113–131
23. Schmidt, C., Branke, J., Chick, S.E.: Integrating techniques from statistical ranking into evolutionary algorithms. In F. Rothlauf et al., ed.: Applications of Evolutionary Computing. Volume LNCS 3907., Springer (2006) 752–763

PVIII

A MEMETIC-NEURAL APPROACH TO DISCOVER RESOURCES IN P2P NETWORKS

by

Ferrante Neri, Niko Kotilainen and Mikko Vapa 2008

In Recent Advances in Evolutionary Computation for Combinatorial Optimization, volume 153 of *Studies in Computational Intelligence*, pages 113-129

8

A Memetic-Neural Approach to Discover Resources in P2P Networks

Ferrante Neri, Niko Kotilainen, and Mikko Vapa

Department of Mathematical Information Technology, Agora, University of Jyväskylä, P.O. Box 35 (Agora), FI-40014 University of Jyväskylä, Finland
neferran@cc.jyu.fi, niko.kotilainen@jyu.fi, mikko.vapa@jyu.fi

Summary. This chapter proposes a neural network based approach for solving the resource discovery problem in Peer to Peer (P2P) networks and an Adaptive Global Local Memetic Algorithm (AGLMA) for performing in training of the neural network. The neural network, which is a multi-layer perceptron neural network, allows the P2P nodes to efficiently locate resources desired by the user. The necessity of testing the network in various working conditions, aiming to obtain a robust neural network, introduces noise in the objective function. The AGLMA is a memetic algorithm which employs two local search algorithms adaptively activated by an evolutionary framework. These local searchers, having different features according to the exploration logic and the pivot rule, have the role of exploring decision space from different and complementary perspectives. Furthermore, the AGLMA makes an adaptive noise compensation by means of explicit averaging on the fitness values and a dynamic population sizing which aims to follow the necessity of the optimization process. The numerical results demonstrate that the proposed computational intelligence approach leads to an efficient resource discovery strategy and that the AGLMA outperforms an algorithm classically employed for executing the neural network training.

Keywords: Memetic Algorithms, Neural Networks, P2P Networks, Telecommunication, Noisy Optimization Problems.

8.1 Introduction

During recent years the use of peer-to-peer networks (P2P) has significantly increased. P2P networks are widely used to share files or communicate with each other using Voice over Peer-to-Peer (VoP2P) systems, for example Skype. Due to the large number of users and large files being shared communication load induced to the underlying routers is enormous and thus demand of high performance in peer-to-peer networks is constantly growing.

In order to obtain a proper functioning of a peer-to-peer network a crucial point is to efficiently execute the peer-to-peer resource discovery, meaning the search of information (files, users, devices etc.) within a network of computers connected by Internet. An improper resource discovery mechanism would lead to overwhelming query traffic within the P2P network and consequently to a waste of bandwidth of each single user connected to the network.

Although several proposals are present in commercial packages (e.g., Gnutella [151] and KaZaA), this problem is still intensively studied in literature. Resource discovery strategies can be divided into two classes: breadth-first search and depth-first search. Breadth-First Search (BFS) strategies forward a query to multiple neighbors at the same time whereas Depth-First Search (DFS) strategies forward only to one neighbor. In both strategies, the choice of those neighbors receiving the query is carried out by heuristic methods. These heuristics might be stochastic e.g. random selection [152], or based on deterministic rules [153].

BFS strategies have been used in Gnutella [151], where the query is forwarded to all neighbors and the forwarding is controlled by a time-to-live parameter. This parameter is defined as the amount of hops required to forward the query. Two nodes are said to be n hops apart if the shortest path between them has length n [153]. The main disadvantage of the Gnutella's mechanism is that it generates a massive traffic of query messages when the time-to-live parameter is high thus leading to a consumption of an unacceptable amount of bandwidth.

In order to reduce query traffic, Lv et al. [152] proposed the *Expanding Ring*. This strategy establishes that the time-to-live parameter is gradually increased until enough resources have been found. Although use of the *Expanding Ring* is beneficial in terms of query reduction, it introduces some delay to resource discovery and thus implies a longer waiting time for the user. Kalogeraki et al. [154] proposed a Modified Random Breadth-First Search (MRBFS) as an enhancement of the Gnutella's algorithm. In MRBFS, only a subset of neighbors are selected randomly for forwarding. They also proposed an intelligent search mechanism which stores the performance of the queries previously done for each neighbor. This memory storage is then used to direct the subsequent queries. Following the ideas of Kalogeraki et al., Menascé [155] proposed that only a subset of neighbors are randomly selected for forwarding. Yang and Garcia-Molina [153] proposed the Directed BFS (DBFS), which selects the first neighbor based on one of several heuristics and further uses BFS for forwarding the query. They also proposed the use of local indices for replicating resources to a certain radius of hops from a node. In Gnutella2 [156] a trial query is sent to the neighbors and, on the basis of obtained results, an estimate of how widely the actual query should be forwarded is calculated.

In the DFS strategies, selection of the neighbor chosen for the query forwarding is performed by means of heuristics. The main problem related to use of this strategy is the proper choice of this heuristic. A popular heuristic employed with this aim is the random walker which selects the neighbor randomly. The random walker terminates when a predefined number of hops have been travelled or when enough resources have been found. Lv et al. [152] studied the use of multiple random walkers which periodically check the query originator in order to verify if the query should still be forwarded further. Tsoumakos and Roussopoulos [157] proposed an Adaptive Probabilistic Search (APS). The APS makes use of feedback from previous queries in order to tune probabilities for further forwarding of random walkers. Crespo and Garcia-Molina [158] proposed the routing indices, which provide shortcuts for random walkers in locating resources. Sarshar et al. [159] proposed the Percolation Search Algorithm (PSA) for power-law networks. The idea is to replicate a copy of resources to a sufficient number of nodes and thus ensure that resource discovery algorithm locates at least one replica of the resource.

The main limitation of the previous studies, for both BFS and DFS strategies, is that all the approaches are restricted to only one search strategy. On the contrary, for the same P2P network, in some conditions it is preferable to employ both BFS and DFS strategies. In order to obtain a flexible search strategy, which intelligently takes into account the working conditions of the P2P network, Vapa et al. [160] proposed a neural network based approach (NeuroSearch). This strategy combines multiple heuristics as inputs of a neural network in order to classify among all its neighbors those which will receive the query, thus it does not fix a priori the search strategy (breadth-first or depth-first) to be employed. Depending on the working conditions of the P2P network, NeuroSearch can alternate between both search strategies during a single query.

Since NeuroSearch is based on a neural network, it obviously follows that an initial training is needed. The resulting optimization problem is very challenging because neural networks have a large number of weights varying from minus to plus infinity. In addition, in order to obtain a robust search strategy it is required that training is performed in various working conditions of a P2P network. It is therefore required that many queries are executed, thus making the training problem computationally expensive and the optimization environment noisy.

8.2 NeuroSearch - Neural Network Based Query Forwarding

As highlighted above, NeuroSearch [160] is a neural network-based approach for solving the resource discovery problem. NeuroSearch combines different local information units together as an input to multi-layer perceptron (MLP) neural network [161]. Multi-layer perceptron is a non-linear function approximator, which is organized into different layers: an input layer, one or more hidden layers and an output layer. Adjacent layers are connected together with weights, these weights are the parameters of the function approximator to be determined by the learning process. Hidden and output layers contain neurons, which take a weighted sum of outputs from the previous layer and use a non-linear transfer function to produce output to the next layer. NeuroSearch uses two hidden layers, both having 10 neurons and two different transfer functions in hidden and output layers. The structure of this neural network has been selected on the basis of previous studies carried out by means of the P2PRealm simulation framework [162].

We characterize the query forwarding situation with a model consisting of 1)the previous forwarding node, 2)the currently forwarding node and 3)the receiver of the currently forwarding node. Upon receiving a query, the currently forwarding node selects the first of its neighbors and determines the inputs, related to that neighbor, of the neural network. Then, the neural network output is calculated. This output establishes whether or not the query will be forwarded to the neighbor. Next, all other neighbors including the previous forwarding node, are processed in a similar manner by means of the same neural network. If some of the neighbors were forwarded, then new query forwarding situations will occur until all forwarding nodes have decided not to forward query further.

Fig. 8.1 shows the functioning of a P2P network with neural network based forwarding.

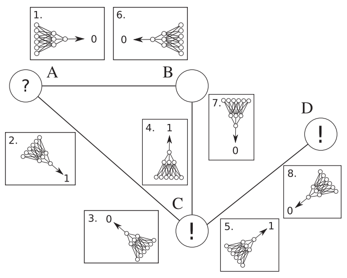


Fig. 8.1. Query Forwarding

The circles shown in the figure represent the peers of the P2P network. The arcs between the peers represent the Transmission Control Protocol (TCP) communication links between the peers. The rectangles represent a neural network evaluation for different neighbors. More specifically, node A denoted with a question mark begins a query. It attempts to forward the query to node B. The neural network in rectangle 1. outputs zero and therefore the query is not forwarded. Instead the second evaluation for node C, shown in rectangle 2, outputs one and the query is forwarded to node C. Then node C attempts to forward the query to neighbor nodes and the nodes B and D receives the query. In the last steps nodes B and D do not forward the query further and the query ends. The query enters nodes C and D denoted with an exclamation mark thereby locating two resource instances.

8.2.1 The Set of the Neural Network Inputs

The MLP uses constant, binary and discrete valued inputs as an information for making forwarding decisions. Each input I_j is a neuron and all 22 inputs I form the input layer.

The following input is constant:

(1) *Bias* takes value 1. Bias is needed in MLP neural networks to approximate functions with non-zero output in case of zero input.

The following inputs are binary:

(2) *Sent* scores 1 if the query has already been forwarded to the receiver. Otherwise it scores 0.
(3) *CurrentVisited* scores 1 if the query has already been received by the currently forwarding node, else it scores 0.
(4) *From* is a binary variable indicating whether a query was received from this receiver. *From* scores 1 if the current query was received from this receiver. Otherwise it scores 0.
(5) *RandomNeighbor* scores 1 for a randomly selected receiver and 0 for other receivers in the current node.
(6) *EnoughReplies* scores 1, if through the query path used by the current query an equal number or more resources have been found as were given in RepliesToGet input parameter (see below). Otherwise *EnoughReplies* scores 0.
The following inputs are discrete:
(7) *Hops* is the number of edges the query has travelled in a P2P network (see definition of *Hops* in section 8.1).
(8) *ToNeighbors* is the number of neighbors connected to the receiver.
(9) *CurrentNeighbors* is the number of neighbors connected to the currently forwarding node.
(10) *FromNeighbors* is the number of neighbors connected to the previous forwarding node.
(11) *InitiatorNeighbors* is the number of neighbors connected to the query initiator.
(12) *NeighborsOrder* is a number associated to each neighbor connected to the forwarding peer. The *NeighborsOrder* is assigned by ascent sorting and enumerating $(0, 1, 2...)$ the neighbors according to their degree. By degree of a peer node we mean the number of neighbors connected to it.
(13) *FromNeighborsOrder*, indicates the *NeighborsOrder* of the previous forwarding node.
(14) *RepliesNow* is the number of replies the query locates in its query path.
(15) *PacketsNow* is the number of packets the query produces in its query path.
(16) *RepliesToGet* is the number of resources that need to be located.
(17) *Forwarded* is the number of times the currently forwarding node has forwarded the query.
(18) *NotForwarded* is the number of times the current node did not forward the query.
(19) *DecisionsLeft* is the number of forwarding “decisions” the current node will still make for the current query message i.e. how many neighbors have not yet been evaluated for forwarding the query message.
(20) *SentCounter* is the number of times the current query has already been forwarded to the receiver.
(21) *CurrentVisitedCounter* is the number of times the query has already been received by the currently forwarding node.
(22) *BranchingResourcesMissing* estimates how many resources on average should still be located from the current query path. First the estimate is set

to the value of *RepliesToGet*. The estimate is updated each time the current node has made all the forwarding decisions. If the current node contained the queried resource, the value is decreased by one. The estimate is then updated depending on whether the current value is positive or negative. In case of a positive value, the current value is divided with the number of neighbors receiving the query. In case of a negative value, the current value is multiplied by the number of neighbors, which will receive the query.

8.2.2 Input Scaling

To ensure that all inputs are in the range of $[0, 1]$ the discrete inputs need to be scaled. The discrete inputs can be classified into three categories according to their original range of variability.

(a) Inputs in the range of $[0, \infty]$ are *Hops*, *NeighborsOrder*, *FromNeighborsOrder*, *RepliesNow*, *PacketsNow*, *RepliesToGet*, *Forwarded*, *NotForwarded*, *DecisionsLeft*, *SentCounter* and*CurrentVisitedCounter* and they are scaled with the function $s(I_j) = \frac{1}{I_j+1}$
(b)Inputs in the range of $[1, \infty]$ are *ToNeighbors*, *CurrentNeighbors*, *FromNeighbors* and *InitiatorNeighbors* and they are scaled with $s(I_j) = \frac{1}{I_j}$
(c)*BranchingResourcesMissing* is in the range of $[-\infty, \infty]$ and it is scaled with the sigmoid function $s(I_{22}) = \frac{1}{1+e^{-I_{22}}}$

The scaled inputs I are then given to the neural network.

8.2.3 Calculation of the Neural Network Output

The neurons on the hidden layers contain the transfer function $t(a) = \frac{2}{1+e^{-2a}} - 1$ where a is the sum of the outcoming weighted outputs from the previous (input or first hidden) layer and *Bias*.

The output layer neuron contains a transfer function

$$u(a) = \begin{cases} 0 \text{ if } a < 0 \\ 1 \text{ if } a \geq 0 \end{cases} \tag{8.1}$$

where a is the sum of the outcoming weighted outputs from the second hidden layer. The output function is thus defined as follows:

$$O = f(I) = u\left(w_{3,1} + \sum_{l=2}^{L} w_{3,l}\, t\left(w_{2,1,l} + \sum_{k=2}^{K} w_{2,k,l}\, t\left(\sum_{j=1}^{J} w_{1,j,k} s_j(I_j)\right)\right)\right) \tag{8.2}$$

where J is the number of inputs, K is the number of neurons on the second layer, L is the number of neurons on the third layer, $w_{1,j,k}$ is the weight from the j^{th} input to k^{th} neuron on the first hidden layer, $w_{2,k,l}$ is the weight from the k^{th} neuron on the first hidden layer to l^{th} neuron on the second hidden layer and $w_{3,l}$ is the weight from the l^{th} neuron on the second hidden layer to the output neuron, $w_{2,1,l}$ is the bias weight associated to the second hidden layer and $w_{3,1}$ is the bias weight associated to the output layer.

Output O can take a boolean value indicating whether the query is forwarded to the neighbor node currently being taken into consideration. The neural network output is calculated separately for each neighbor node and after the calculations, the query is sent to neighbor nodes which had an output value 1.

8.3 The Optimization Problem

The neural network described above is supposed to handle the communication and data transfer between a couple of peers. As in all cases of the neural networks, its proper functioning is subject to correctly executed training. The training of a neural network consists of determination of the set of weight coefficients W. As shown in formula (8.2), the weights can be divided into three categories on the basis of the layer to which they belong to. There are 22 input neurons and 10 neurons on both the hidden layers. Since one input is constant (*Bias*) the total amount of weights is $22 * 9 + 10 * 9 + 10 = 298$. The weights can take values in the range $[-\infty, \infty]$.

8.3.1 Fitness Formulation

In order to estimate the quality of a candidate solution, the performance of the P2P network is analyzed with the aid of a simulator whose working principles are described in [162]. More specifically, the set of weights is given to the simulator and a certain number n of queries are performed. The total fitness over the n queries is given by:

$$F = \sum_{j=1}^{n} F_j(W) \tag{8.3}$$

where F_j is the fitness contribution from each query. It is important to remark that multiple queries are needed in order to ensure that the neural network is robust in different query conditions. In addition, for each query the amount of Available Resources (AR) instances is known. Thus, AR is a constant value given a priori.

For each query, the simulator returns two outputs:

(a) the number of query packets P used in the query
(b) the number of found resource instances R during the query

For details regarding the simulation see [162]. These outputs are combined in the following way in order to determine each F_j:

$$F_j = \begin{cases} 0 & \text{if } P > 300 \\ 1 - \frac{1}{P+1} & \text{if } P \leq 300 \text{ AND } \quad R = 0 \\ 50 * R - P & \text{if } P \leq 300 \text{ AND } 0 < R < \frac{AR}{2} \\ 50 * \frac{AR}{2} - P & \text{if } P \leq 300 \text{ AND } \quad \frac{AR}{2} < R \end{cases} \tag{8.4}$$

In formula (8.4), the constant values 300 and 50 have been set according to the criterion explained in [160].

The first condition in (8.4) ensures that the neural network should eventually stop forwarding the queries. The second condition controls that if no resources are found then the neural network increases the number of query packets sent to the network. The third condition states that if the number of found resources is not enough then the neural network develops only by locating more resources. The fourth condition ensures that when half of the available resource instances are found from the network the fitness grows if the neural network uses fewer query packets. The fourth condition also upperbounds F_j to $50 * \frac{AR}{2} - \frac{AR}{2} = 49 * \frac{AR}{2}$, because it is imposed that when half of the resource instances are found P is at minimum $\frac{AR}{2}$.

Thus, the problem of discovering resources in the P2P network consists of the maximization of F in a 298-dimension continuous space:

$$\max\left(F\left(W\right)\right) \text{in} \left[-\infty, \infty\right]^{298} \tag{8.5}$$

Due to the necessity of ensuring robustness of the neural network in different queries, the fitness value varies with the chosen query. The querying peer and the queried resource need to be changed to ensure that the neural network is not just specialized for searching resources from one part of the network or one particular resource alone. Since n (in our case $n = 10$) queries are required and they are chosen at random, fitness F is noisy. This noise does not have any peculiarity and therefore it can hardly be approximated by a known distribution function. Let us indicate with PN the distribution of this noise and thus re-formulate the problem in equation (8.5)

$$\max\left(F\left(W\right) + Z\right) \text{in} \left[-\infty, \infty\right]^{298}; Z \sim PN \tag{8.6}$$

8.3.2 Features of the Decision Space and the Fitness Landscape

As highlighted above, the optimization problem is highly multivariate and is defined in a continuous domain. It obviously follows that the problem is quite challenging due to a high dimensionality. In addition, presence of the noise enhances the difficulty of the problem because it introduces some "false" optima into the landscape which disturb the functioning of any optimization algorithm [163,164].

Due to the structure of each F_j (see equation (8.4)), the fitness landscape contains discontinuities. In particular, it is relevant to observe that due to the first condition in (8.4) the fitness landscape contains some plateaus with a null value as well as some other areas which take non-null values and contain a variability. In order to give a rough description of the fitness landscape, the following test has been designed. 2 million candidate solutions have been pseudo-randomly sampled by means of a uniform distribution within the decision space. Fig. 8.2 and 8.3 show the histogram and distribution curve, respectively, related to this test. It should be noted that the y-axis has a logarithmic scale. Fig. 8.2 shows that about half the points take a null fitness value and Fig. 8.3 shows that

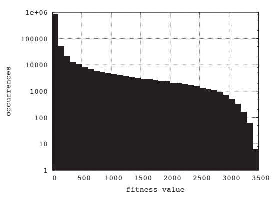


Fig. 8.2. Histogram of Fitness Values

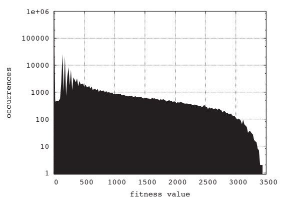


Fig. 8.3. Distribution of Fitness Values

the distribution curve contains a very high, sharp peak in zero and other lower sharp peaks before 500. This obviously means the fitness landscape contains some plateau areas for low fitness values (up to 500) and a variational area for high fitness values. In other words the fitness landscape is locally flat and contains several areas having a small variation in fitness values [165, 166]. This feature of the fitness landscape makes the optimization problem very challenging since many optimization algorithms can easily stagnate or prematurely converge in a suboptimal plateau.

8.4 The Adaptive Global-Local Memetic Algorithm

In order to solve the problem in (8.5), an Adaptive Global-Local Memetic Algorithm (AGLMA) has been implemented.

8.4.1 Initialization

An initial sampling made up of S^i_{pop} individual has been executed pseudo-randomly with a uniform distribution function over the interval $[-0.2, 0.2]$. This choice can be briefly justified in the following way. The weights of the initial set of neural networks must be small and comparable among each other in order to avoid one or a few weights dominating with respect to the others as suggested in [167, 168].

8.4.2 Parent Selection and Variation Operators

All individuals of the population S_{pop} undergo recombination and each parent generates an offspring. The variation occurs as follows. For each candidate solution i is associated a self-adaptive vector h_i which represents a scale factor for the exploration. More specifically, at the first generation the self-adaptive vectors h_i are pseudo-randomly generated with uniform distribution within $[-0.2, 0.2]$.

At the subsequent generations each self-adaptive vector is updated according to [167, 168]:

$$h_i^{k+1}(j) = h_i^k(j)\, e^{(\tau N_j(0,1))} \quad \text{for} \quad j = 1, 2...n \tag{8.7}$$

where k is the index of generation, j is the index of variable ($n = 298$), $N_j(0,1)$ is a Gaussian random variable and $\tau = \frac{1}{\sqrt{2\sqrt{n}}} = 0.1659$. Each corresponding candidate solution W_i is then perturbed according to the following formula [167, 168]:

$$W_i^{k+1}(j) = W_i^k + h_i^{k+1}(j) N_j(0,1) \quad \text{for} \quad j = 1, 2...n \tag{8.8}$$

It is interesting to observe that each component $h_i^k(j)$ of the self-adaptive vector at the k^{th} generation can be seen as the standard deviation of a Gaussian perturbation.

8.4.3 Fitness Function

In order to take into account the noise, function F is calculated n_s times (where n_s stands for number of samples) and an *Explicit Averaging* technique is applied [164, 169]. More specifically each set of weights of a neural network (candidate solution) is evaluated by means of the following formula:

$$\hat{F} = F_{mean}^i - \frac{\sigma^i}{\sqrt{n_s}} \tag{8.9}$$

where F_{mean}^i and σ^i are respectively the mean value and standard deviation related to the n_s samples performed to the i^{th} candidate solution.

The penalty term $\frac{\sigma^i}{\sqrt{n_s}}$ takes into account the distribution of the data and the number of performed samples [170]. Since the noise strictly depends on the solution under consideration, it follows that for some solutions the value of σ^i is relatively small (stable solutions) and so the penalization is small. On the other hand, other solutions could be unstable and score 0 during some samples and a high performance value during other samples. In these cases σ^i is quite large and the penalization must be significant.

8.4.4 Local Searchers

Two local search algorithms with different features in terms of search logic and pivot rule have been employed. These local searchers have the role of supporting the evolutionary framework, offering new search directions and exploiting the available genotypes [171, 172]. The **Simulated Annealing** (SA) metaheuristic [173], [174] has been chosen since it offers an exploratory perspective in the decision space which can choose a search direction leading to a basin of attraction different from where starting point W_0 is. The exploration is performed by using the same mutation scheme as was described in equations (8.7) and (8.8) for an initial self-adaptive vector h_0 pseudo-randomly sampled in $[-0.2, 0.2]$. The main reason for employing the SA in the AGLMA is that the evolutionary framework should be assisted in finding better solutions which improve the available genotype while at the same time exploring areas of the decision space not yet explored. It accepts with a certain probability solutions with worse performance in order to obtain a global enhancement in a more promising basin of attraction. In addition, the exploratory logic aims to overcome discontinuities of the fitness landscape and to “jump” into a plateau having better performance. For these reasons the SA has been employed as a “global” local searcher.

The application of the SA local searcher can be successful in most cases, in the early generations, and in the late generations as well. Moreover, due to its structure the SA can efficiently offer solutions in unexplored basins of attractions and, therefore, prevent an undesired premature convergence. The most delicate issue related to the SA is choice of parameters. The SA has two parameters which are the budget and the initial temperature $Temp^0$. The budget has been fixed at 600 fitness evaluations (in order to have a constant computational cost for the SA). The setting of the initial temperature $Temp^0$ is performed as explained in section 8.4.5. The temperature $Temp$ is reduced according to a hyperbolic law following the suggestions in [175].

The **Hooke-Jeeves Algorithm** (HJA) [176, 177] is a deterministic local searcher which has a steepest descent pivot rule. Briefly the implemented HJA consists of the following. An initial radius d_0 (in our implementation $d_0 = 0.5$) an initial candidate solution W_0 and a direction exploratory matrix are required. In this implementation a standard identity matrix I has been chosen due to the hypercubic features of the decision space. Let us indicate with $I(m, :)$ the m^{th} row of the direction matrix with $m = 1, 2..n$ ($n = 298$).

The HJA consists of an exploratory move and a pattern move. Indicating with W_{cb} the current best candidate solution and with d the generic radius of the search, the HJA during the exploratory move samples the points $W_{cb}(m)+dI$ $(m, :)$ with $m = 1, 2..n$ and the points $W_{cb}(m) - dI(m, :)$ with $m = 1, 2..n$ only along those directions which turned out unsuccessful during the "+" move. Then, if a new current best is found W_{cb} is updated and the pattern move is executed. If a new current best is not found, d is halved and the exploration is repeated.

The HJA pattern move is an aggressive attempt of the algorithm which aims to exploit promising search directions. Rather than centering the following exploration at the most promising explored candidate solution (W_{cb}), the HJA tries to move further [178]. The algorithm centers the subsequent exploratory move at $W_{cb} \pm dI(m, :)$ ("+" or "-" on the basis of the best direction). If this second exploratory move does not outperform $\hat{F}(W_{cb})$ (the exploratory move fails), then an exploratory move with W_{cb} as the center is performed. The HJA stops either when $d < 0.01$ or when the budget condition of 1000 fitness evaluation is reached.

The HJA is supposed to efficiently exploit promising solutions enhancing their genotype in a meta-Lamarckian logic and thus assist the evolutionary framework in quickly climbing the basin of attractions. In this sense the HJA can be considered as a kind of "local" local searcher integrated in the AGLMA.

8.4.5 Adaptation

An adaptation has been implemented taking into account the features of this kind of fitness landscape in order to design a robust algorithm [179, 171]. At the end of each generation the following parameter is calculated:

$$\psi = 1 - \left| \frac{\hat{F}_{avg} - \hat{F}_{best}}{\hat{F}_{worst} - \hat{F}_{best}} \right| \tag{8.10}$$

where $\hat{F}_{worst}$, $\hat{F}_{best}$, and $\hat{F}_{avg}$ are the worst, best, and average of the fitness function values in the population, respectively.

As highlighted in [166], ψ is a fitness-based measurement of the fitness diversity which is well-suited for flat fitness landscapes. The employment of this parameter, taking into account the presence of plateaus in the fitness landscape. ψ, measures the population diversity in terms of fitness and is relative to the range of the fitness values $[\hat{F}_{best}, \hat{F}_{worst}]$ in the population. Thus, even when all fitness values are very similar, leading to $\hat{F}_{best}$ and $\hat{F}_{worst}$ being close to each other, ψ still gives a well scaled measure, since it uses the relative distance of $\hat{F}_{avg}$ from $\hat{F}_{best}$. The population has high diversity when $\psi \approx 1$ and low diversity when $\psi \approx 0$. A low diversity means that the population is converging (possibly in a suboptimal plateau). This parameter has been used in order to control coordination among the local searchers and a dynamic population size.

8.4.6 Coordination of the Local Searchers

ψ has been employed in order to execute an adaptive coordination of the local searchers so as to let them assist the evolutionary framework in the optimization process.

The SA is activated by the condition $\psi \in [0.1, 0.5]$. This adaptive rule is based on the observation that for values of $\psi > 0.5$, the fitness diversity is high and then the evolutionary framework needs to have a high exploitation of the available genotypes (see [180], [166] and [181]). In other words, under this condition the evolutionary framework does not require the assistance of a local searcher. On the other hand, if $\psi < 0.5$ the fitness diversity is decreasing and the application of the SA can introduce a new genotype in the population which can prevent a premature convergence. Basically, the SA has the potential to detect new promising solutions outside a suboptimal plateau into which the population could have fallen. In this sense, the SA has been employed as a local searcher with "global" exploratory features. The condition regarding the lower bound of usability of the SA ($\psi > 0.1$) is due to the consideration that if $\psi < 0.1$ convergence is approaching and the fitness value has already been drastically reduced.

Thus, the SA has the role of exploiting already existing good genotypes but nevertheless to explore other areas of the decision space. Due to its structure, the SA could lead new search directions but its application can lead to a solution which is worse than that which it started with. For this reason, in our implementation it is applied to the second best individual. The initial temperature $Temp^0$ has to be chosen for this local searcher. It is adaptively set to be $Temp^0 = \left|\hat{F}_{avg} - \hat{F}_{best}\right|$. This means that the algorithm does not accept worse solutions when the convergence has practically occurred.

The HJA is activated when $\psi < 0.2$ and is applied to the solution with best performance. The basic idea behind this adaptive rule is that the HJA has the role of quickly improving the best solution while staying in the same basin of attraction. In this light, the action of the HJA can be seen as purely "local". The condition $\psi < 0.2$ means that the HJA is employed when there are some chances that optimal convergence is approaching. An early application of this local searcher can be inefficient since a high exploitation of solutions having poor fitness values would not lead to significant improvements of the population.

It should be noted that in the range $\psi \in [0.1, 0.2]$ both the local searchers are applied to the best two individuals of the population. This range is very critical for the algorithm because the population is tending towards a convergence but still has not reached such a condition. In this case, there is a high risk of premature convergence due to the presence of plateaus and suboptimal basins of attraction or false minima introduced by noise. Thus, the two local searchers are supposed to "compete and cooperate" within the same generation, merging the "global" search power of the SA and the "local" search power of the HJA under supervision of the evolutionary framework.

An additional rule has been implemented. When the SA has succeeded in enhancing the starting solution, the algorithm attempts to further enhance it by the application of the HJA. This choice can be justified by the consideration that when the SA succeeds, it returns a solution having better performance with a genotype (usually) quite different from the starting one and, therefore, belonging to a region of the decision space which has not yet been exploited.

8.4.7 Dynamic Population Size in Survivor Selection

The adaptation controls the population size whose dynamic variation has two combined roles. The first is to massively explore the decision space and thus prevent a possible premature convergence (see [182], [180]), the second is to *Implicitly Average* in order to compensate for noise by means of the evaluations of similar individuals [169]. The population is resized at each generation according to the formula:

$$S_{pop} = S_{pop}^{f} + S_{pop}^{v} \cdot (1 - \psi) , \tag{8.11}$$

where S_{pop}^{f} and S_{pop}^{v} are the fixed minimum and maximum sizes of the variable population S_{pop}, respectively.

The coefficient ψ is then used to dynamically set the population size [183,184] in order to prevent a premature convergence and stagnation. According to the first role, when the population is highly diverse a small number of solutions need to be exploited. When $\psi \approx 0$ the population is converging and a larger population size is required to increase the exploration. The main idea is that if a population is in a suboptimal plateau an increase of the population size enhances the chances of detecting new promising areas of the decision space and thus prevent premature convergence. On the other hand, if the population is spread out in the decision space it is highly desirable that the most promising solution leads the search and that the algorithm exploits this promising search direction.

According to the second role, it is well-known that large population sizes are helpful in defeating the noise (*Implicitly Averaging*) [185,186]. Furthermore, recent studies [187,170] have noted that the noise jeopardizes proper functioning of the selection mechanisms, especially in cases of low fitness diversity since the noise introduces a disturbance in pair-wise comparison. Therefore, the AGLMA

```
Pseudo-Random Initial Sampling of the weights $W$ and self-adaptive parameters $h$;
Fitness evaluation of the initial population by $\hat{F} = F^i_{mean} - \frac{\sigma^i}{\sqrt{n_s}}$;
Calculate $\psi = 1 - \left| \frac{\hat{F}_{avg} - \hat{F}_{best}}{\hat{F}_{worst} - \hat{F}_{best}} \right|$;
while budget conditions and $\psi > 0.01$
  for all the individuals $i$
    for all the variables $j$
      $h_i(j) = h_i(j) e^{(\tau N_j(0,1))}$;
      $W_i(j) = W_i + h_i(j) N_j(0,1)$;
    end-for
  end-for
  Fitness evaluation of the population by $\hat{F} = F^i_{mean} - \frac{\sigma^i}{\sqrt{n_s}}$;
  Sort the population made up of parents and offsprings according to their fitness values;
  if $\psi \in [0.1, 0.5]$
    Execute the SA on the individual with the $2^{nd}$ best performance;
    if $\psi < 0.2$
      Execute the HJA on the individual with the best performance;
    end-if
    if the SA succeeds
      Execute the HJA on the individual enhanced by the SA;
    end-if
  end-if
  Calculate $S_{pop} = S^f_{pop} + S^v_{pop} \cdot (1 - \psi)$;
  Select the $S_{pop}$ best individuals to the subsequent generation;
  Calculate $\psi = 1 - \left| \frac{\hat{F}_{avg} - \hat{F}_{best}}{\hat{F}_{worst} - \hat{F}_{best}} \right|$;
end-while
```

Fig. 8.4. AGLMA pseudo-code

aims to employ a large population size in critical conditions (low diversity) and a small population size when a massive averaging is unnecessary.

After the calculation of S_{pop} in equation (8.11), the AGLMA selects for the subsequent generation, among parents and offspring, the S_{pop} candidate solutions having the best performance.

The algorithm stops when either a budget condition on the number of fitness evaluations is satisfied or ψ takes a value smaller than 0.01.

Fig. 8.4 shows the pseudo-code of the AGLMA.

8.5 Numerical Results

For the AGLMA 30 simulation experiments have been executed. Each experiment has been stopped after 1500000 fitness evaluations. At the end of each generation, the best fitness value has been saved. These values have been averaged over the 30 experiments available. The average over the 30 experiments defines the Average Best Fitness (ABF). Analogously, 30 experiments have been carried out with the Checkers Algorithm (CA) described in [167, 168] according to the implementation in [160], and the ACA which is the CA with the fitness as shown in (8.9) and the adaptive population size as shown in (8.11). In addition a standard real valued Genetic Algorithm (GA) has been run for the problem under study. The GA employs an arithmetic blend crossover and a Gaussian mutation. For the same P2P network, the BFS according to the implementation in

Gnutella and the random walker DFS proposed in [152] have been applied. Table 8.1 shows the parameter settings for the three algorithms and the optimization results. The final fitness $\hat{F}^b$ obtained by the most successful experiment (over the 30 sample runs), the related number of query packets P used in the query and the number of found resource instances R during the query are given. In addition the average best fitness at the end of the experiments $< \hat{F} >$, the final fitness of the least successful experiment $\hat{F}^w$ and the related standard deviation are shown. Since the BFS follows a deterministic logic, thus only one fitness value is shown. On the contrary, the DFS under study employs a stochastic structure and thus the same statistic analysis as that of GA, CA, ACA and AGLMA over 30 experiments has been carried out.

Numerical results in Table 8.1 show that the methods employing the neural network approach are more promising than the classical methods for P2P networks. Moreover, AGLMA and ACA outperform the CA and the AGLMA slightly outperformed the ACA in terms of final solution found. The GA performed significantly worse than the other optimization algorithms.

Fig. 8.5 shows a graphical representation of the solution in the most successful experiment (over the 30 carried out) returned by the proposed AGLMA. An index of the weights are shown on the x-axis and the corresponding weight values are shown on the y-axis (see the crosses in figure).

As shown in Fig. 8.5, according to AGLMA, we propose a neural network having a set of 298 weights, which take small values. More specifically, the proposed neural network contains 296 weight values between -1 and 1. On the contrary, two weights belonging to the first hidden layer take the values of around -1.5 and 1.5.

Table 8.1. Parameter setting and numerical results

PARAMETER	AGLMA	CA	ACA	GA	BFS	DFS
EVOLUTIONARY FRAMEWORK						
S^i_{pop}	30	30	30	30	–	–
S_{pop}	$\in [20, 40]$	30	$\in [20, 40]$	30	–	–
sample size n_s	10	–	10	–	–	–
SIMULATED ANNEALING						
initial temperature $Temp^0$	adaptive	–	–	–	–	–
temperature decrease	hyperbolic	–	–	–	–	–
maximum budget per run	600	–	–	–	–	–
HOOKE-JEEVES ALGORITHM						
exploratory radius	$\in [0.5, 0.01]$	–	–	–	–	–
maximum budget per run	1000	–	–	–	–	–
NUMERICAL RESULTS						
P	350	372	355	497	819	514
R	81	81	81	85	81	81
$\hat{F}^b$	3700	3678	3695	3366	3231	3536
$< \hat{F} >$	3654	3582	3647	2705	–	3363
$\hat{F}^w$	3506	3502	3504	0	–	3056
std	36.98	37.71	36.47	1068	–	107.9

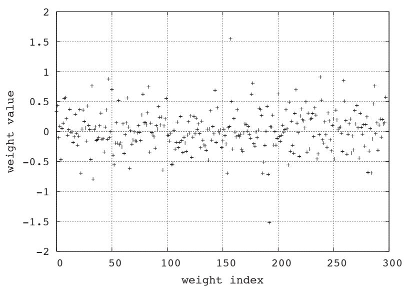
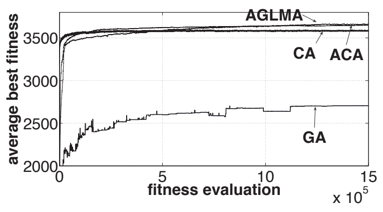


Fig. 8.5. Distribution of Neural Network Weights



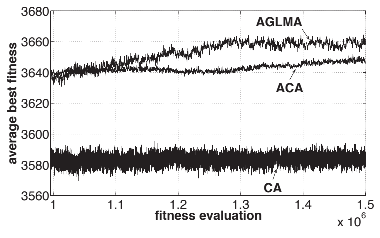


Fig. 8.6. Comparison of the algorithmic performance

Fig. 8.7. Comparison of the algorithmic performance (zoom)

Fig. 8.6 shows the comparison of the performance over the 1.5×10^6 fitness evaluations and Fig. 8.7 shows a zoom detail of the algorithmic performance.

Fig. 8.6 shows that the AGLMA has a slower convergence than the CA and the ACA but it reaches a final solution having better performance. It is also clear that the ACA has intermediate performance between the CA and AGLMA. The ACA trend, in early generations, has a rise quicker than the AGLMA but slower than the CA. On the other hand, in late generations, the ACA outperforms the CA but not the AGLMA. As shown in Fig. 8.6, the GA performed much worse than the CA structured algorithms (CA, ACA, AGLMA) also in terms of convergence speed.

It can be remarked that the ACA can be seen as an AGLMA which does not employ local searchers but only executes *Implicit* (dynamic population size) and *Explicit Averaging* (n_s re-samples and modified fitness). In other words, the ACA does not contain the memetic components but does contain the noise filtering components. Fig. 8.7 shows that the ACA and the AGLMA are much more robust to noise than the CA. In fact, as shown in Fig. 8.7, the trend of the CA performance contains a high amplitude (about 20) and frequency ripple around a mean value, while the ACA and AGLMA performance are roughly monotonic. The oscillatory trend of the CA performance is due to an incorrect estimation of candidate solutions. The quick initial rise of the CA performance is, according to our interpretation, also due to an overestimation of an unstable solution. On the contrary, the ACA and the AGLMA efficiently filter the noise and select only reliable solutions for the subsequent generations.

Regarding effectiveness of the local searchers, the comparison between the ACA and the AGLMA shows that the AGLMA slightly outperforms the ACA tending to converge to a solution having a better performance. Moreover it is shown that after 1.5×10^6 fitness evaluations, the trend of the AGLMA still continues to grow whilst the other trends seem to have reached a final value.

8.6 Conclusion

This chapter proposes an Adaptive Global Local Memetic Algorithm (AGLMA) for performing the training of a neural network, which is employed as computational intelligence logic in P2P resource discovery. The AGLMA employs averaging strategies for adaptively executing noise filtering and local searchers in order to handle the multivariate fitness landscape. These local searchers execute the global and local search of the decision space from different perspectives. The numerical results show that the application of the AGLMA leads to a satisfactory neural network training and thus to an efficient P2P network functioning. The comparison with two popular metaheuristics present in literature shows that the proposed approach seems to be promising in terms of final solution found and reliability in noise environment. Matching with another algorithm with intermediate features highlights the effectiveness of each algorithmic component integrated in the proposed algorithm.

The proposed neural network along with the learning strategy carried by the AGLMA allows the efficient location of resources with little query traffic. Thus, the user of the P2P network obtains plentiful amounts of information about resources without consuming a large portion of his own bandwidth for query traffic.

Acknowledgements

We wish to thank Teemu Keltanen and Andrea Caponio for their kind support in analyzing the data.

References

[151] A. Oram, Ed., *Peer-to-Peer: Harnessing the Power of Disruptive Technologies.* Sebastopol, CA: O'Reilly & Associates, 2001.

[152] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker, "Search and replication in unstructured peer-to-peer networks," in *Proceedings of the 16th International Conference on Supercomputing.* ACM Press, 2002, pp. 84–95.

[153] B. Yang and H. Garcia-Molina, "Improving search in peer-to-peer networks," in *Proceedings of the 22nd International Conference on Distributed Computing Systems (ICDCS'02)*, 2002, pp. 5–14.

[154] V. Kalogeraki, D. Gunopulos, and D. Zeinalipour-Yazti, "A local search mechanism for peer-to-peer networks," in *Proceedings of the 11th International Conference on Information and Knowledge Management.* ACM Press, 2002, pp. 300–307.

[155] D. A. Menascé, "Scalable P2P search," *IEEE Internet Computing*, vol. 7, no. 2, pp. 83–87, March-April 2003.

[156] A. Fisk, "Gnutella dynamic query protocol v0.1," *Gnutella Developer's Forum*, May 2003.

[157] D. Tsoumakos and N. Roussopoulos, "Adaptive probabilistic search for peer-to-peer networks," in *Proceedings of the Third IEEE International Conference on P2P Computing (P2P2003).* IEEE Press, 2003, pp. 102–109.

[158] A. Crespo and H. Garcia-Molina, "Routing indices for peer-to-peer systems," in *Proceedings of the 22nd IEEE International Conference on Distributed Computing Systems (ICDCS'02).* IEEE Press, 2002, pp. 23–33.

[159] N. Sarshar, P. O. Boykin, and V. P. Roychowdhury, "Percolation search in power law networks: Making unstructured peer-to-peer networks scalable," in *Proceedings of the Fourth International Conference on P2P Computing (P2P'04).* IEEE Press, 2004, pp. 2–9.

[160] M. Vapa, N. Kotilainen, A. Auvinen, H. Kainulainen, and J. Vuori, "Resource discovery in P2P networks using evolutionary neural networks," in *International Conference on Advances in Intelligent Systems - Theory and Applications, 067-04*, 2004.

[161] A. Engelbrecht, *Computational Intelligence - An Introduction.* J. Wiley and Sons, 2002.

[162] N. Kotilainen, M. Vapa, T. Keltanen, A. Auvinen, and J. Vuori, "P2PRealm - peer-to-peer network simulator," in *International Workshop on Computer-Aided Modeling, Analysis and Design of Communication Links and Networks.* IEEE Communications Society, 2006, pp. 93–99.

[163] K. Deb, *Multi-objective Optimization using Evolutionary Algorithms.* Chichester, UK: John Wiley and Sons LTD, 2001, pp. 147–149.

[164] J. Branke, *Evolutionary Optimization in Dynamic Environments.* the Netherlands: Kluwer A. P., 2001, pp. 125–172.

[165] B. Derrida and L. Peliti, "Evolution in a flat fitness landscape," *Bulletin of Mathematical Biology*, vol. 53, pp. 355–382, 1991.

[166] F. Neri, J. Toivanen, G. L. Cascella, and Y. S. Ong, "An adaptive multimeme algorithm for designing HIV multidrug therapies," *IEEE/ACM Transactions on Computational Biology and Bioinformatics, Special Issue on Computational Intelligence Approaches in Computational Biology and Bioinformatics*, vol. 4, no. 2, pp. 264–278, 2007.

[167] K. Chellapilla and D. Fogel, "Evolving neural networks to play checkers without relying on expert knowledge," *IEEE Transactions on Neural Networks,*, vol. Vol 10, No 6, pp. pp. 1382–1391, 1999.

[168] K. Chellapilla and D. Fogel, "Evolving an expert checkers playing program without using human expertise," *IEEE Transactions on Evolutionary Computation,*, vol. Vol 5, No 4, pp. pp. 422–428, 2001.

[169] Y. Jin and J. Branke, "Evolutionary optimization in uncertain environments - a survey," *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 3, pp. 303–317, 2005.

[170] F. Neri, G. L. Cascella, N. Salvatore, A. V. Kononova, and G. Acciani, "Prudent-daring vs tolerant survivor selection schemes in control design of electric drives," in *Applications of Evolutionary Computing, Lecture Notes in Computer Science*, F. Rothlauf et al., Ed., vol. 3907. Springer, 2006, pp. 805–809.

[171] N. Krasnogor, "Toward robust memetic algorithms," in *Recent Advances in Memetic Algorithms*, W. E. Hart, N. Krasnogor, and J. E. Smith, Eds. Berlin, Germany: Springer, 2004, pp. 185–207.

[172] Y. S. Ong and A. J. Keane, "Meta-lamarckian learning in memetic algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 8, pp. 99–110, 2004.

[173] S. Kirkpatrick, C. D. J. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, no. 220, pp. 671–680, 1983.

[174] V. Cerny, "A thermodynamical approach to the traveling salesman problem," *Journal of Optimization, Theory and Applications*, vol. 45, no. 1, pp. 41–51, 1985.

[175] H. Szu and R. Hartley, "Fast simulated annealing," *Physics Letters A*, vol. 122, pp. 157–162, 1987.

[176] R. Hooke and T. A. Jeeves, "Direct search solution of numerical and statistical problems," *Journal of the ACM,*, vol. Vol 8, pp. pp. 212–229, 1961.

[177] Kaupe, F. Jr., ""Algorithm 178: direct search"," *Communications of the ACM,*, vol. Vol 6, No 6, pp. pp. 313–314, 1963.

[178] C. T. Kelley, *Iterative Methods of Optimization.* Philadelphia, USA: SIAM, 1999, pp. 212–229.

[179] N. Krasnogor, B. Blackburne, E. Burke, and J. Hirst, "Multimeme algorithms for proteine structure prediction," in *Proceeding of Parallel Problem Solving in Nature VII.* Lecture Notes in Computer Science, Springer-Verlag, 2002.

[180] A. Caponio, G. L. Cascella, F. Neri, N. Salvatore, and M. Sumner, "A fast adaptive memetic algorithm for on-line and off-line control design of pmsm drives," *IEEE Transactions on System Man and Cybernetics-part B, special issue on Memetic Algorithms*, vol. 37, no. 1, pp. 28–41, 2007.

[181] F. Neri and R. A. E. Mäkinen, "Hierarchical evolutionary algorithms and noise compensation via adaptation," in *Evolutionary Computation in Dynamic and Uncertain Environments, Studies in Computational Intelligence*, S. Yang, Y. S. Ong, and Y. Jin, Eds. Springer, 2007, pp. 345–369.

[182] A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computation.* Berlin, Germany: Springer-Verlag, 2003.

[183] T. Bäck, "The interaction rate of mutation rate, selection, and self-adaptation within a genetic algorithm," in *Proceedings of Parallel Problem Solving from Nature (PPSN-II).* Elsevier Science, 1992, pp. 85–94.

[184] A. E. Eiben, R. Hinterding, and Z. Michaelwicz, “Parameter control,” in *Evolutionary Computation 2, Advanced Algorithms and Operators*, T. Baeck, D. B. Fogel, and Z. Michaelwicz, Eds. Institute of Physics Publishing, 2000, pp. 170–187.

[185] J. M. Fitzpatrick and J. J. Grefenstette, “Genetic algorithms in noisy viroments,” *Machine Learning*, vol. 3, pp. 101–120, 1988.

[186] B. L. Miller and D. E. Goldberg, “Genetic algorithms, selection schemes, and the varying effects of noise,” *Evolutionary Computation*, vol. 4, no. 2, pp. 113–131, 1996.

[187] C. Schmidt, J. Branke, and S. E. Chick, “Integrating techniques from statistical ranking into evolutionary algorithms,” in *Applications of Evolutionary Computing*, F. Rothlauf et al., Ed., vol. LNCS 3907. Springer, 2006, pp. 752–763.

PIX

MOBILE CHEDAR – A PEER-TO-PEER MIDDLEWARE FOR MOBILE DEVICES

by

Niko Kotilainen, Matthieu Weber, Mikko Vapa and Jarkko Vuori 2005

In Proceedings of the Third IEEE International Conference on Pervasive Computing and Communications Workshops, pages 86-90

Mobile Chedar - A Peer-to-Peer Middleware for Mobile Devices

Kotilainen N.[*], Weber M., Vapa M.[+] and Vuori J.
University of Jyväskylä, P.O.Box 35 (Agora), 40014 University of Jyväskylä
[npkotila, mweber, mikvapa, mimic]@jyu.fi

Abstract

This paper presents the Mobile Chedar Peer-to-Peer middleware for mobile peer-to-peer applications. The middleware is an extension to the Chedar peer-to-peer network allowing mobile devices to access the Chedar network and also to communicate with other Mobile Chedar peers. Currently, Mobile Chedar uses Bluetooth to connect to Chedar gateway peers, because Bluetooth is now the most widespread short-range radio technology in mobile phones. We also introduce in this paper one example of a Mobile Peer-to-Peer application for cooperative lecture notes taking, which is based on Mobile Chedar.

1. Introduction

Coulouris et al. [3] define distributed system as a system in which components located at networked computers communicate and coordinate their actions only by message passing. Peer-to-Peer networks (P2P) are an instance of distributed systems.

P2P networks allow the sharing of resources over the Internet. The resources can be for example, computing power, storage space, network bandwidth, printers etc. Another main feature of P2P networks is that all the tasks and responsibilities for managing the network are shared between the peers. This means that there is no single control entity responsible for providing the services.

Middleware provides an application programming interface (API) for accessing message passing functionalities and other common services needed in distributed systems. The main benefit is that by using middleware programmers can speed up the development of applications because the features needed for distribution are already provided by the middleware.

This paper describes one implementation of a mobile peer-to-peer (MP2P) middleware enabling information sharing in a mobile environment. The proposed middleware is an extension to a non-mobile Chedar P2P network. In the following sections we discuss the related work on MP2P, introduce the Chedar P2P middleware, the Mobile Chedar MP2P middleware with an API for MP2P applications and finally present a co-operative learning application using the Mobile Chedar.

2. Related Work

Proem [5] is a mobile middleware providing solution for developing and deploying applications for mobile ad hoc networks. In Proem, middleware is responsible for presence and discovery services as well as being an identity, data space and community manager. Proem has been designed for mobile peers in ad hoc networks whereas in Mobile Chedar also peers with fixed P2P network connections are supported. The current prototype of Proem uses Wireless Local Area Network (WLAN) for communication and has been implemented using Java.

7DS [12] is a Java based data prefetching tool for mobile devices. It allows mobile users to advertise data items of their mobile devices and to query other user's data items through WLAN connections. 7DS works only on IP networks and is designed for disseminating rather static content. IP multicast is used for querying the peers.

XMIDDLE [10] is a reflective middleware enabling transparent sharing of XML documents between mobile peers. XMIDDLE does not use a fixed P2P infrastructure and therefore differs from our approach. Because data structure consists of XML trees, modifications to the branches of XML tree are fine-grain for example compared to modification of files. In Mobile Chedar, resource queries are matched using XPath expressions as in XMIDDLE, but modifications to the data stream are not guaranteed to have the same order in all peers. XMIDDLE solves the problem by allowing user to resolve the update conflicts. The current XMIDDLE prototype is based on WLAN and has been implemented using Java.

[*] The work of N. Kotilainen is supported by Innovations in Business, Communication and Technology (InBCT) project.
[+] The work of M. Vapa is supported by Graduate School in Electronics, Telecommunications and Automation (GETA).

MOBY [4] is a service network enabling access to services on wide area networks. The framework is built using Jini and Jini Technology Surrogate Architecture Specification. MOBY's approach seems to be the closest to ours. However, there are some differences in the design choices. In MOBY resources are registered to Jini Lookup Service, which is located in the local area network. In Mobile Chedar the mobile peers store their resources and no registration of resources to external server is needed. MOBY's P2P network is based on super-peer architecture i.e. the network is divided into domains by Mnode super-peers whereas in Chedar all peers are equal. Communication between Mnodes is handled using UDP, but in Chedar network connections are established using TCP. MOBY uses IP addresses to identify peers and Mobile Chedar, relying on Bluetooth service discovery, does not require IP addressing of mobile devices. Resource discovery is handled in MOBY using expanding ring [9] between Mnodes while Chedar currently uses breadth-first search. In overall, MOBY is designed more like a fixed overlay, because the links between Mnodes are preconfigured compared to the autonomous overlay approach used in Chedar P2P network.

Mobile Chedar is an extension to existing peer-to-peer network and therefore differs from the MP2P software presented in the literature. The middleware provides mechanisms for data streaming which is unique feature among the considered related work. Also Mobile Chedar uses Bluetooth as a transmission technology in contrast to WLAN used in other studies.

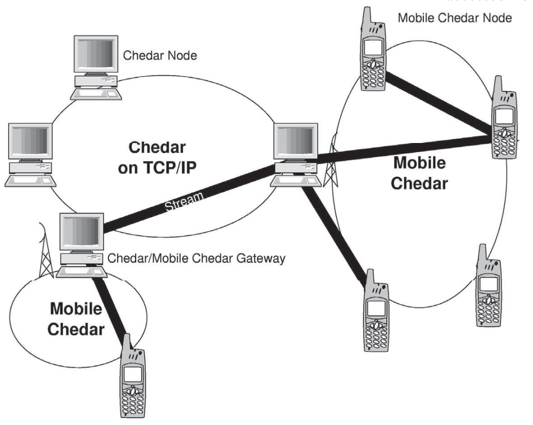


Figure 1. Stream delivery between Chedar and Mobile Chedar nodes

3. Mobile Chedar – A Mobile Peer-to-Peer Middleware

3.1. Chedar

Chedar (CHEap Distributed ARchitecture) is a peer-to-peer middleware designed for peer-to-peer applications. The goal of the Chedar software is to provide a convenient API for peer-to-peer application developers. For example Chedar can be used to locate unused resources in a computer network that could be used for a given purpose; one could thus locate idle computers with a given characteristics in order to run computationally intensive calculations. Chedar nodes maintain resources of different types: data (files), software (e.g. specific applications or operating systems) and hardware (e.g. CPU, printers and displays). It also provides platform independence and quick adaptation to new hardware. Chedar has been programmed with Java 2 Standard Edition and is currently being used for speeding up the computations of NeuroSearch resource discovery algorithm [13] with P2P Distributed Computing application (P2PDisCo) [6] and for studying distributed data fusion in peer-to-peer environment [11].

Each Chedar node is identified with a unique identifier (Chedar ID). The nodes maintain a database of locally available resources shared by the owner of the device. These resources can include for example files and databases, software running on the device that can be accessed or used by remote users, and hardware characteristics of the device. Also, remote resources discovered on the network can be added to the database combined with information about their owner identified by Chedar ID and meta-information about themselves. Meta-information can contain e.g. type and path for the files, name and version for applications or any useful description for the hardware depending on the application, which uses the information. The resource database is stored as an XML document using a specific DTD. This organization of data allows making rich and complex queries to the database in the form of XPath expressions.

3.2. Mobile Chedar

With the advent of mobile computing and the inherent peer-to-peer properties of mobile ad hoc networks, Chedar has been extended to the mobile platform as Mobile Chedar. It thus provides functionalities for registering resources on the mobile device and for querying resources from other peers. Mobile Chedar is implemented using Java 2 Micro Edition (J2ME), which is suitable for mobile wireless devices and has spread widely among new mobile phone models. Mobile Chedar uses Bluetooth [1] as a transmission technology for connecting to other peers, because Bluetooth is the most common available short-range radio frequency wireless protocol stack on today's mobile devices.

Current Bluetooth implementations have a restriction that nodes can be connected to only one piconet at a time [8]. Therefore the only topology that is available for constructing Bluetooth network is star-shaped. One device functions as a master and others as slaves. In Mobile Chedar one node connected to piconet can be e.g. a workstation with a Bluetooth adapter and an Internet connection working as a Mobile Chedar/Chedar gateway node. Through the Internet connection it keeps contact with other Chedar nodes and through Bluetooth it can communicate with other Mobile Chedar peers.

A common use case for Mobile Chedar is the querying of a resource located on Chedar nodes or on other Mobile Chedar nodes through the gateway peer and then using the found resource. Chedar nodes can provide streamable resources to Mobile Chedar peers and depending on the device capabilities of the Mobile Chedar node they can subscribe to these streams. Currently text and picture are supported and in the future also audio and video streams will be supported also. Multiple peers can simultaneously subscribe to the same stream and after subscribing they also start to publish the stream as a resource. Therefore it is enough for a peer to locate one peer that provides the requested stream. This kind of a streaming is called end system multicasting [2]. Also, because streams are duplex, the data written to the streams by peers will be delivered to all other peers currently subscribed. Duplex streams can be realized by flooding all data inserted to the stream along the multicast tree to all other participants in the tree. However, the order of the data is not preserved and it is handled in a First-In-First-Out manner. Totally ordered delivery of data would require more complex implementation in this kind of environment [3]. Figure 1 illustrates a stream delivery between Chedar and Mobile Chedar peers.

Neighbor discovery is a prerequisite for resource queries. Since the nodes are able to communicate with each other using a wireless channel, it is easy to discover all nodes within range of the radio frequency transceiver using Bluetooth's Service Discovery Protocol (SDP). The discovery of resources is performed as one hop query, tagged with a unique Message-ID, to all the nodes within Bluetooth range. If the query arrives to a Chedar/Mobile Chedar gateway node, it checks whether the query has already been received: if not, it is forwarded to all of its Chedar neighbors with default time-to-live; otherwise, the query is discarded. If the query message matches one of the resources owned by the node, the node replies to the neighbor from which it received the query with the same Message-ID as in the query message. The reply message then travels back to the originator of the query on the same path as the query traveled on. Once the location of the resource (or locations, if there exists multiple instances of the same resource in the network) is known, Mobile Chedar informs the application, which decides how to acquire or use the resource.

4. Mobile Chedar Application Programming Interface

Mobile Chedar provides the following API for MP2P applications:

register(String resourceidentifier)	Adds a resource to the resource database.
unregister(String resourceidentifier)	Removes a resource from the resource database.
connected()	Checks if Mobile Chedar is connected to other Chedar nodes.
query(String resourceidentifier)	Executes a query.
subscribe(Resource resource)	Subscribes to the found resource.
unsubscribe(Resource resource)	Unsubscribes from a subscribed resource.
send(Resource resource, Message data)	Sends data to the subscribed resource.

The MP2P applications must implement the following methods:

resourceFound(Resource resource)	Informs the application when the query has located a matching resource.
receive(Resource resource, Message data)	Informs the application when new data has arrived to a subscribed resource.

5. Mobile Peer-to-Peer Learning Environment

Mobile Peer-to-Peer Learning Environment (MP2PLE) [7] is designed for collaborative note taking during lectures as a test application for Chedar peer-to-peer network and Mobile Chedar middleware. The MP2PLE user interface contains a text area displaying the current state of notes and provides means for users to edit them. With MP2PLE, the mobile device user may create a new stream for other participants to join or subscribe to an already existing one by executing a query. After subscription the user is allowed to modify any part of the notes by selecting a paragraph and submitting the changes. Whenever the data is being changed it is streamed to other participants subscribed to the same stream. At the moment, each user can be only subscribed to one stream at a time. The user interface of MP2PLE is shown in figure 2.

There are two common use cases for such kind of an application. Firstly, it serves as a personal note-taking tool to store lecture notes. Secondly, people who do not take notes can benefit from other user's notes, either during the lectures, or later, e.g. from home by accessing Mobile Chedar nodes through a gateway node.

MP2PLE has certain limitations in the current design. The tiny user interface is problematic and provides only primitive means to take notes e.g., pictures cannot be drawn and course presentation material cannot be integrated with MP2PLE. Also, taking lecture notes is difficult because of the small keypads in mobile phones. These limitations can only be overcome if larger screen sizes and more convenient input devices are being used.

Bluetooth does not allow multi-hop with current mobile phones because the device can only belong to one piconet at a time. To support multiple devices in a classroom one solution would be to use the approach presented in [8] and to equip Bluetooth base stations with two radio chips. They could, for example, be plugged into power supplies inside the classroom and equipped with Mobile Chedar middleware to function as relaying devices for queries. Another solution would be to use different transmission technology e.g., WLAN, which however is not yet available in low-cost mobile terminals.

6. Conclusion

Mobile peer-to-peer enables new kind of applications taking advantage of emerging short-range radio technologies and allowing collaborative resource sharing between peers. This paper describes one way to construct peer-to-peer networks with support for mobile devices and demonstrates the feasibility with a prototype implementation. The future work of Mobile Chedar and MP2PLE includes the support of audio and video streams and determining the feasibility of the approach with practical experiments.

7. References

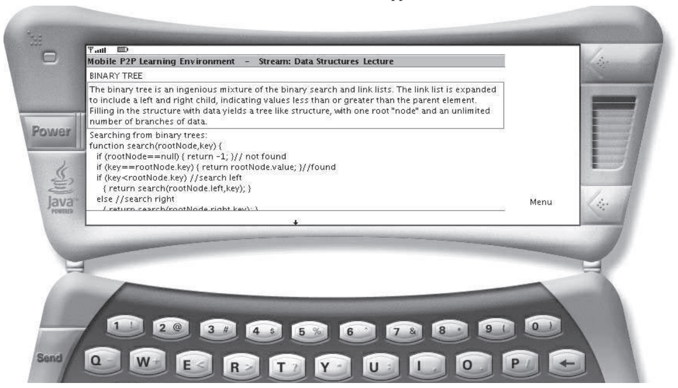
[1] Bluetooth Special Interest Group, "Bluetooth Core Specification v1.2", March 2004

[2] Y. Chu, S. G. Rao, and H. Zhang, "A Case for End System Multicast", International conference on Measurement and modeling of computer systems, SIGMETRICS, Proceedings, 2000, pp. 1-12.

[3] G. Coulouris, J. Dollimore, and T. Kindberg, "Distributed Systems – Concepts and Design", 3rd Edition, Addison-Wesley, 2001.

[4] T. Horozov, A. Grama, V. Vasudevan, and S. Landis, "MOBY - A Mobile Peer-to-Peer Service and Data Network", International Conference on Parallel Processing, Proceedings, 18-21 August 2002, pp. 437-444.

[5] G. Kortuem, "Proem: a middleware platform for mobile peer-to-peer computing", Mobile Computing and Communications Review, ACM, Volume: 6, Issue: 4, October 2002, pp. 62-64.



Mobile P2P Learning Environment – Stream: Data Structures Lecture

BINARY TREE

The binary tree is an ingenious mixture of the binary search and link lists. The link list is expanded to include a left and right child, indicating values less than or greater than the parent element. Filling in the structure with data yields a tree like structure, with one root "node" and an unlimited number of branches of data.

Searching from binary trees:
```
function search(rootNode,key) {
  if (rootNode==null) { return -1; }// not found
  if (key==rootNode.key) { return rootNode.value; }//found
  if (key<rootNode.key) //search left
    { return search(rootNode.left,key); }
  else //search right
    { return search(rootNode.right,key); }
```

Figure 2. User Interface for Mobile Peer-to-Peer Learning Environment

[6] N. Kotilainen, M. Vapa, M. Weber, J. Töyrylä, and J. Vuori, "P2PDisCo – Java Distributed Computing For Workstations Using Chedar Peer-to-Peer Middleware", University of Jyväskylä, 2004.

[7] J. Kurhinen, M. Vapa, M. Weber, N. Kotilainen, and J. Vuori, "Short Range Wireless P2P for Co-Operative Learning", The 3rd International Conference on Emerging Technologies and Applications, Proceedings, 16-18 October 2004, pp. 141-145.

[8] M. Leopold, M. B. Dydensborg, and P. Bonnet, "Bluetooth and Sensor Networks: A Reality Check", The First International Conference on Embedded Networked Sensor Systems, Proceedings, November 2003, pp. 103-113.

[9] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker, "Search and Replication in Unstructured Peer-to-Peer Networks", Proceedings of the 16th International Conference on Supercomputing, ACM Press, 2002, pp. 84-95.

[10] C. Mascolo, L. Capra, S. Zachariadis, and W. Emmerich, "XMIDDLE: A Data-Sharing Middleware for Mobile Computing", International Journal on Wireless Personal Communications, Kluwer Academic Publisher, Volume: 21, Issue: 1, April 2002, pp. 77-103.

[11] S. Nazarko, "Evaluation of the data fusion methods using Kalman filtering and Transferable Belief model", Master's Thesis, University of Jyväskylä, 2002.

[12] M. Papadopouli, and H. Schulzrinne, "Design and Implementation of a Peer-to-Peer Data Dissemination and Prefetching Tool for Mobile Users", First New York Metro Area Networking Workshop, IBM T. J. Watson Research Center, Hawthorne, New York, 12 March 2002.

[13] M. Vapa, N. Kotilainen, A. Auvinen, H. Kainulainen, and J. Vuori, "Resource Discovery in P2P Networks Using Evolutionary Neural Networks", International Conference on Advances in Intelligent Systems – Theory and Applications, Proceedings, IEEE Press, Luxembourg, 15-18 November 2004.

PX

P2PDISCO – JAVA DISTRIBUTED COMPUTING FOR WORKSTATIONS USING CHEDAR PEER-TO-PEER MIDDLEWARE

by

Niko Kotilainen, Mikko Vapa, Matthieu Weber, Joni Töyrylä and Jarkko Vuori
2005

In Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium

P2PDisCo – Java Distributed Computing for Workstations Using Chedar Peer-to-Peer Middleware

Kotilainen N.[*], Vapa M.[+], Weber M., Töyrylä J. and Vuori J.
University of Jyväskylä, P.O.Box 35 (Agora), 40014 University of Jyväskylä
[niko.kotilainen, mikko.vapa, mweber, joni.toyryla, jarkko.vuori]@jyu.fi

Abstract

This paper introduces Peer-to-Peer Distributed Computing (P2PDisCo) software, which provides an interface for distributing the computation of Java programs to multiple workstations. P2PDisCo can be used to distribute any Java program that uses files for storing input and output parameters without significant code modifications to the Java program itself. P2PDisCo has been built over Chedar peer-to-peer middleware and is currently being used for speeding up the training of neural networks with evolutionary algorithm.

Keywords: peer-to-peer, distributed computing, Java, P2PDisCo, Chedar P2P middleware

1. Introduction

Peer-to-Peer (P2P) networks allow sharing of resources over the Internet. The resources can be for example, computing power, storage space, network bandwidth, printers etc. For sharing computing power, peer-to-peer networks are a natural choice, because many workstations are running idle most of the time.

In contrast to clusters, in P2P networks all the tasks and responsibilities for managing the network are shared between the peers. This means that there exists no single control entity responsible for providing the services. Also, because P2P networks do not require a dedicated hardware, distributing computation among workstations is usually a cost-effective solution.

Distributed computing on workstations is mostly known of SETI@home [1], which uses master-slave architecture for distributing the analysis of radio signals obtained from space to workstations. In this paper we present a peer-to-peer system, in which all the connected peers can work as master nodes initiating computations and also as slaves processing computations when idling.

The paper is structured as follows. Section 2 describes the related work in the area of P2P distributed computing. Section 3 introduces Chedar peer-to-peer middleware and section 4 the P2PDisCo peer-to-peer distributed computing software and its application programming interface (API). In section 5 we discuss the experiences gained from the use of P2PDisCo for distributing the training of neural network with evolutionary optimization algorithm and the planned future work. Section 6 concludes the paper.

2. Related Work

Nowadays there are many alternatives for distributed computing using Java programming language. One class of software is formed by programming language independent distributed computing tools that support Java. An example of such software is Globus Toolkit [2], in which Java Commodity Grid kit [3] provides an interface for accessing Globus services using Java programs. Globus contains mechanisms for code mobility which poses risks on security because the downloaded code needs to come from a trusted source or otherwise guaranteed to not be malicious. In our approach we have avoided the use of mobile code and the installation of the executed Java Archive file (jar) is done semiautomatically using copying scripts or by the user who donates computing power for the Chedar peer-to-peer network. Also Globus uses centralized indexes for resource discovery whereas in P2PDisCo the resource discovery is decentralized and provided by the Chedar peer-to-peer network.

Programming language dependent class of Java distributed computing can be divided in two: Java extensions and Java libraries. Java extensions such as JavaParty [4] provide special distribution mechanisms requiring changes to the Java compiler and/or Java Virtual Machine (JVM). This is a drawback considering the difficulty of executing the distributed code. Java libraries provide special class libraries for the distribution without

[*] The work of N. Kotilainen is supported by Innovations in Business, Communication and Technology (InBCT) project.
[+] The work of M. Vapa is supported by Graduate School in Electronics, Telecommunications and Automation (GETA).

need for modifications to the Java compiler or JVM. Therefore Java libraries are easier to deploy. An example of such library is JavaSymphony [5] as well as P2PDisCo presented in this paper. In JavaSymphony all the computing resources are centrally configured under JS-Shell whereas in P2PDisCo no central management exists.

There are also some implementations of Java distributed computing that use peer-to-peer network for locating the resources. In such design the resource index has been decentralized and peers cooperatively route resource queries among each other. An example of such system is GT-P2PRMI [6] which allows Remote Method Invocation (RMI) lookups to be performed through an extended version of RMIRegistry called P2PRMIRegistry. P2PRMIRegistry is used to form the overlay network, for binding and publishing the remote methods and for looking up the published remote methods.

3. Chedar P2P Middleware

Chedar (CHEap Distributed ARchitecture) is peer-to-peer middleware designed for peer-to-peer applications. Chedar constructs a pure peer-to-peer network using topology management algorithms and provides functionalities for locating resources in the network. The original goal of Chedar was to locate unused resources in a computer network that could be used for a given purpose; one could thus locate idle computers with a given characteristics in order to run computationally intensive calculations. It has then been extended to handle any type of resource: data (files), software (e.g. operating systems or specific applications) and hardware (e.g. computers, printers and displays). Implementation of Chedar is based on Java programming language, thus the software is platform independent and provides easy adaptation to different hardware.

Chedar nodes are identified with a pseudo-unique identifier called Chedar ID. Each node maintains a database of locally available resources shared by the owner of the device. These resources can include for example files and databases, software running on the device that can be accessed or used by remote users, and hardware characteristics of the device. Also, remote resources discovered on the network can be added to the database combined with information about their owner identified by Chedar ID and meta-information about themselves. Meta-information can contain e.g. type and path for the files, name and version for applications or any useful description for the hardware depending on the application, which is using the information. The resource database is stored as an XML document using a specific DTD. This organization of data allows making rich and complex queries to the database in the form of XPath expressions.

Chedar node keeps a list of neighbors it is connected to through TCP sockets. TCP provides reliable data delivery between the end points and thus also the disappearance of a neighbor can be detected. The neighbor list is updated based on heuristics such as the number of relayed query replies and the actual query replies provided by the neighbor to form an efficient topology for resource discovery. Currently, we use as a query mechanism breadth-first search algorithm (BFS), where query is forwarded to each neighbor except the one from which the query was received. Also, if the query has already been received it is not forwarded further. The number of hops that a query can take is limited in BFS with a time-to-live value. BFS is suitable for small-sized networks and guarantees to locate all resources from the network within the time-to-live horizon, but if the network grows larger the time-to-live value has to be decreased. In our experiments the network size of 200 workstations with 100 Mb/s Ethernet connections the query traffic has not yet posed a significant problem and therefore a more efficient version of the query algorithm has not been implemented.

Each query contains a Message-ID and a query XPath description. Whenever a query enters a Chedar node, the node checks its resource database whether it contains a resource matching the XPath expression. If resource is found, a reply message is sent back using the route, which the query came from. To properly relay the reply message back to the query originator, the message needs to contain the same Message-ID as the query did. After the query originator receives replies, it notifies P2PDisCo application, which can react to collected replies. Because the replies also contain the address of replying node, the intermediate nodes along the path as well as the querier learns new nodes in the network without being directly connected to them. This information can be later used for topology updates.

For communicating between two peers, Chedar provides a point-to-point communication protocol allowing basic message passing primitives to be executed by P2P applications. The protocol uses the same path as reply message to deliver messages between peers.

4. P2PDisCo

Peer-to-Peer Distributed Computing (P2PDisCo) was developed for distributing computationally intensive evolutionary optimization method to university workstations. The workstations are basically staying idle most of the time and therefore utilizing their processing power only at times when they are not in use does not interfere normal use of the computers.

P2PDisCo provides Distributed-interface definition with methods for starting and stopping the distributed application and checking whether the application is currently running. This interface needs to be implemented by the distributed application and is invoked by P2PDisCo.

The application is also required to read its parameters from a file and write its output to a file provided by P2PDisCo. The idea is that P2PDisCo pretends to the application that the application is reading all input data from files, but instead the file is delivered from Chedar and thus the application does not see a difference whether it is running remotely or locally. Also this ensures that the computing node does not need to store any data on its hard drive because Chedar delivers the data produced by application through TCP connections to the master node. After receiving data, master writes the received data to files on hard drive. The architecture of P2PDisCo and Chedar is shown in Figure 1.

5. Application Experiences And Future Work

P2PDisCo is at the moment used for speeding up the computations of NeuroSearch resource discovery algorithm [7] and it is deployed to more than 200 workstations of University of Jyväskylä in Agora building. NeuroSearch is a neural network algorithm, which is optimized using iterative evolutionary algorithm. It has been found that the evolutionary algorithm is a stable optimizer, but it requires much more computing power than for example back-propagation algorithms traditionally used for neural network training. The selection of evolutionary algorithm was needed because in

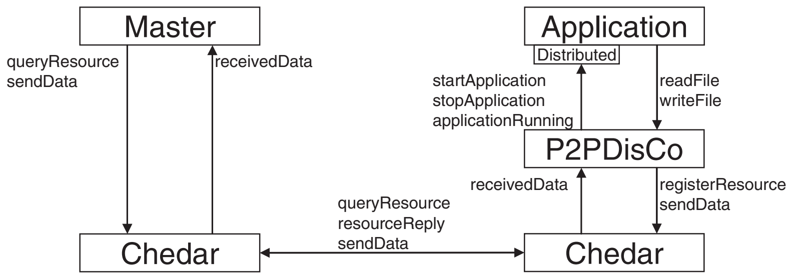


Figure 1. Architecture of P2PDisCo and Chedar.

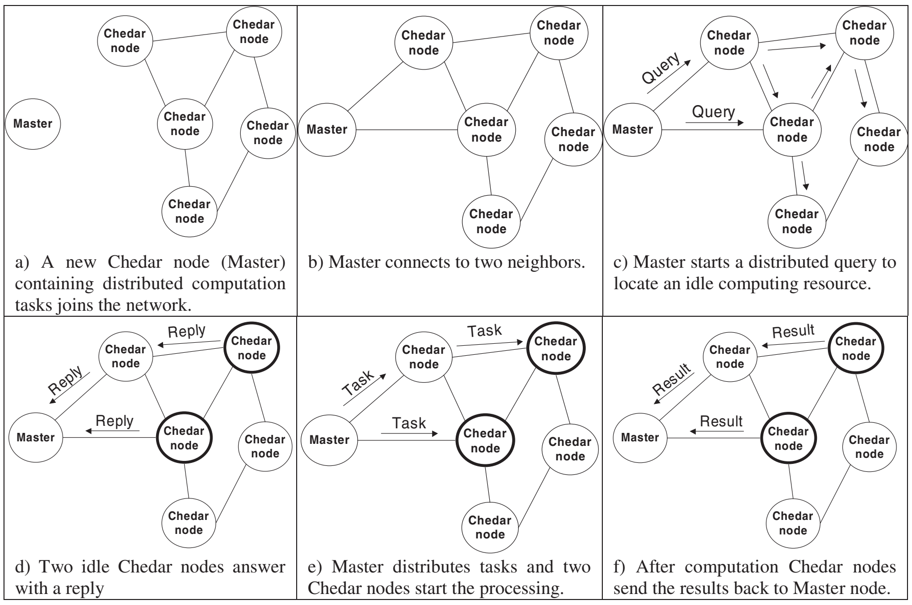
the peer-to-peer resource discovery problem good input-output pairs are unknown and therefore no proper data for supervised training is available.

The process of distributing the computation and collecting results using P2PDisCo is shown in Figure 2. When a peer (denoted as Master) joins the network, it needs to locate other peers to connect. This is currently handled by using a predetermined list of IP addresses and ports. If the peer has already been connected earlier to Chedar network it uses history data for connecting to already learned peers' addresses. Then master starts a query looking for idle computing resource and those peers that are ready for computing answer. Master selects which of the located nodes it uses for computing and distributes tasks to these nodes, which start the computing. During the computation, results are sent when memory buffer is full (currently at 256 KB) to master node and therefore no data is written to computing nodes. Also, this ensures that if the computing node is reset the computation results are still saved to the point of last full memory buffer update.

Because of security concerns the distributed application has been beforehand installed to the computers and it is not automatically delivered during the task distribution. In the task distribution only the execution parameters i.e. configuration files are transferred. Also, currently the IP addresses of master nodes are restricted such that only certain IP addresses are allowed to start computations.

Based on half a year's usage of P2PDisCo it seems that the only major problem with P2PDisCo is the updating of the distributed application. Now it requires that Windows computers are updated with a script and the service running in Windows needs to be stopped and started again. At the moment there is no good solution how to avoid this. Fortunately, the computers are under central administration and the updating can be done from one computer.

As a future work, we are planning to add automatic resuming mechanism for computation, if computing node leaves the network. Now the computing is only restarted, but by checkpointing the state of current execution the computation could be resumed in another computing node from the point when connection was lost. Perhaps, in the future P2PDisCo also allows master to be disconnected for a while and collecting results afterwards from the computing nodes. Also, in the future we are extending the API of P2PDisCo to allow direct communication between computing nodes. This makes it possible to parallelize the evolutionary algorithm for multiple computers with other architectures than master-slave, such as the panmictic model [8].



a) A new Chedar node (Master) containing distributed computation tasks joins the network.

b) Master connects to two neighbors.

c) Master starts a distributed query to locate an idle computing resource.

d) Two idle Chedar nodes answer with a reply

e) Master distributes tasks and two Chedar nodes start the processing.

f) After computation Chedar nodes send the results back to Master node.

Figure 2. The process of distributing computation and gathering the results using P2PDisCo.

6. Conclusion

P2PDisCo provides decentralized architecture for distributed computing of Java programs. P2PDisCo is built on top of Chedar P2P middleware and requires only minor modifications to turn an existing Java application to a distributed one. Based on the experience of training NeuroSearch neural network algorithm, the system seems to perform well in a network of few hundred workstations. As future work resuming mechanism and extension of API to support direct communication between computing nodes is planned.

7. References

[1] SETI@home - The Search for Extraterrestrial Intelligence, http://setiathome.ssl.berkeley.edu/

[2] Foster I. and Kesselman C., "Globus: A Metacomputing Infrastructure Toolkit", *The International Journal of Supercomputer Applications and High Performance Computing*, 11(2), 1997, pp. 115-128.

[3] Von Laszewski G., Foster I., Gawor J., and Lane P., "A Java Commodity Grid Kit", *Concurrency and Computation: Practice and Experience*, 13(8-9), 2001, pp. 643-662.

[4] Philippsen M. and Zenger M., "JavaParty: Transparent Remote Objects in Java", *Concurrency and Computation: Practice and Experience*, 9(11), 1997, pp. 1225-1242.

[5] Fahringer T., "JavaSymphony: A System for Development of Locality-Oriented Distributed and Parallel Java Applications", *IEEE International Conference on Cluster Computing*, 2000.

[6] Chang T. and Ahamad M., "GT-P2PRMI: Improving Middleware Performance Using Peer-to-Peer Service Replication", *10th IEEE International Workshop on Future Trends of Distributed Computing Systems*, 2004.

[7] Vapa M., Kotilainen N., Auvinen A., Kainulainen H., and Vuori J., "Resource Discovery in P2P Networks Using Evolutionary Neural Networks", *IEEE International Conference on Advances in Intelligent Systems – Theory and Applications*, 2004.

[8] Alba E. and Tomassini M., "Parallelism and Evolutionary Algorithms", *IEEE Transactions on Evolutionary Computation*, 6(5), 2002, pp. 443-462.

PXI

P2PREALM – PEER-TO-PEER NETWORK SIMULATOR

by

Niko Kotilainen, Mikko Vapa, Teemu Keltanen, Annemari Auvinen and Jarkko Vuori 2006

In Proceedings of the 11th International Workshop on Computer-Aided Modeling, Analysis and Design of Communication Links and Networks

P2PRealm – Peer-to-Peer Network Simulator

Niko Kotilainen, Mikko Vapa[+], Teemu Keltanen, Annemari Auvinen[+] and Jarkko Vuori
Department of Mathematical Information Technology, University of Jyväskylä
P.O.Box 35 (Agora), 40014 University of Jyväskylä, Finland
firstname.lastname@jyu.fi

***Abstract*—Peer-to-Peer Realm (P2PRealm) is an efficient peer-to-peer network simulator for studying algorithms based on neural networks. In contrast to many simulators, which emphasize on detailed network simulation, the speed of simulation in P2PRealm is essential, because neural networks require a time consuming training phase. Efficiency has been obtained by optimizing training loops inside the simulator, using Java Native Interface (JNI) as well as distributing the simulator to hundreds of workstations using the P2PDisCo platform. In this paper we describe the architecture of P2PRealm and its input/output interfaces. Also, we present the mechanisms used for internally optimizing the implementation and the configuration used for distribution. Finally, we present the use of P2PRealm with the P2PStudio network visualization tool.**

Keywords - peer-to-peer; P2PRealm; network simulation; research infrastructure; neural networks, optimization methods;

I. INTRODUCTION

Peer-to-Peer (P2P) algorithms have been studied at least using three different approaches. These are crawlers, emulators and simulators. Crawler is an implementation of a peer node specially designed for P2P networks research. A crawler can collect data passing through it to get a local view of the P2P network. By deploying multiple crawlers, a bigger part of the peer-to-peer network can be monitored. However, this approach is not able to gather the global view of the network, because the behavior of nodes which are not connected to crawlers is unknown. This problem is solved by emulators, which contain the implementation of a peer node and are used to build a complete P2P network. Multiple emulators can be deployed inside one workstation usually providing quite large P2P networks with only a few workstations.

Even though emulators can be used to get the global view, they are restricted to slow execution, because messages need to be passed between emulator processes through network protocols such as TCP. The third option, simulator, contains an abstracted implementation of peer nodes equivalent to emulators, but uses local data structures for message passing. The use of local data structures significantly increases the speed of execution and therefore is well-suited approach for computationally intensive algorithm studies. The downside of the approach is the inaccuracy of results compared to real-world P2P networks and the difficulty of modeling user behavior. This knowledge can only be obtained using crawlers or by monitoring network traffic inside routers.

Developing peer-to-peer resource discovery and topology management algorithms based on neural networks are computationally intensive tasks. For example, it takes a week for one low-cost workstation to train a good neural network based resource discovery algorithm for a rather small P2P network [23]. Therefore, in computationally intensive algorithmic research the most important factor to consider for a simulator is speed.

This paper describes an end product of a process where emulators were first used for studying P2P algorithms and later re-implemented as an efficient simulator to decrease the time used for execution. Latest improvement of the simulator is the distributed execution on a Peer-to-Peer Distributed Computing platform (P2PDisCo) [11] allowing us to parameter sweep different features of neural network based P2P algorithms.

The paper is structured as follows. Section 2 compares existing P2P simulators with our work and states their differences. Section 3 introduces P2PRealm network simulator and section 4 describes its input and output interfaces. Section 5 describes the main use case of P2PRealm: the training of evolutionary neural networks for P2P resource discovery. Section 6 describes the internal modifications used for optimizing the code and the combination of P2PRealm with P2PDisCo platform. Section 7 illustrates the use of P2PStudio for visualizing the output of P2PRealm and section 8 concludes the paper.

II. RELATED WORK

There are various network simulators available for studying P2P networks. However, many of these simulators are not primarily designed for speed and none of them contains functionalities for neural networks. Because the speed is the most important factor in our simulation environment, it is obvious that abstractions on the level of details are necessary. Packet-level simulators model the P2P protocols with precise protocol headers and field structures, whereas message-level simulators only take into account the number and sizes of the

[+] The works of M. Vapa and A. Auvinen are supported by Graduate School in Electronics, Telecommunications and Automation (GETA).

packets. While packet-level simulation is a desirable feature, it is still often too expensive in terms of computing resources. In addition to speed, other desirable features in our simulations are compilation of statistics on simulation results and visualization of P2P networks. From this viewpoint, we next overview some of the existing P2P simulators.

A. NS-2

NS-2 [15] is one of the most widely used network simulators. The NS-2 is object-oriented discrete event simulator, which closely follows the architecture of the OSI model. It suits well for simulating packet switched networks and small scale networks. The Parallel and Distributed Simulation (PADS) research group has developed an extension that allows network simulation to be run in parallel on multiple machines [17]. Being very detailed simulator, it still does not scale well enough and is slow from a computational point of view. In addition, adding new modules is not straightforward, because of it's complex module structure [14].

B. PLP2P

Packet-level Peer-to-Peer Simulator (PLP2P) [6] provides a framework for other packet-level simulators, e.g. NS-2, in order to provide detailed model of the underlying network. This is done with wrappers, which translate P2P events into underlying packet-level simulator. The authors assert that abstracting low-level details can impact the simulation results to a large extent. The scalability problem of packet-level simulations is solved by running simulations on parallel machines. Nevertheless, as training neural networks requires substantial part of available computing power in order to get result within reasonable time, we need to abstract the level of details of the P2P network.

C. QueryCycle

The QueryCycle simulator [19] is specialized to file-sharing simulations. It has realistic models for content distribution, query activity, download behavior etc. The content distribution is based on a model, where each file belongs to one category and that category is defined by the popularity of the file. Simulations proceed in query cycles representing the time period between issuing a query and receiving a response. Generated queries are passed into a queue and handled on a First-In-First-Out basis.

D. 3LS

3LS [21] is an open-source simulator for overlay networks designed to overcome the problems of extensibility and usability. The system is separated to three architectural levels: a network model, a protocol model and a user model. The network model uses a two-dimensional matrix as a storage of distances between the nodes. The protocol model defines the current protocol being simulated. The user model is the input interface for the user. The 3LS uses most of the memory resources to a graphical interface as the simulator uses main memory to store each event executed for visualization and this limits the system to less than a thousand nodes on a low-cost workstation [14].

E. PeerSim

PeerSim [18] has been developed especially with scalability and support for dynamicity in mind. PeerSim is Java-based and has two simulation engines, one is cycle-based and the other is event driven. Cycle-based engine allows scalable simulation but is not very accurate. Handling large-scale overlay networks requires simplifying assumptions about the simulation details. For example, the details of the transport communication protocol stack are not taken into account. Event driven engine supports dynamicity and is more realistic, but decreases the scalability of the simulation. The abstractions of cycle-based simulations are similar to ours. The difference is that when PeerSim uses the benefits of abstraction for high scalability, we use it to increase the computational efficiency. Parallel execution is a necessity in order to PeerSim to be useful for our research.

F. NeuroGrid

The NeuroGrid simulator [8] was initially designed to support comparative resource search simulations between FreeNet [5], Gnutella [16] and NeuroGrid [8] systems. The simulator is single-threaded, Java-based and uses discrete events. Several protocols are now available for NeuroGrid e.g., Domain Name Service (DNS) and a distributed e-mail protocol. NeuroGrid supports property files that specify the parameters of the simulation to run. This includes the protocol to simulate, the parameters of the network and the amount of searches.

NeuroGrid would be a promising simulator for our research if it wasn't single-threaded and non-parallel.

G. GPS

The General Peer-to-Peer Simulator (GPS) [24] is aiming to respond to a call for extensible framework for simulating P2P networks efficiently and accurately. Efficiency is accomplished with message-level simulation instead of packet-level simulation. Improvement to the level of detail is achieved by tracking the network infrastructure and using a macroscopic mathematical model to obtain accurate estimate of the message behavior, e.g. TCP. The GPS also models downloads of the files, which is often left out from the simulators. GPS is extensible for modeling any P2P protocol, integration with a GUI and network visualization and provides support for topology generation tools.

The GPS is still in it's early stages and details about scalability, usability and performance are scarce. The GPS has also only been used for simulating BitTorrent, where resource discovery is not an essential problem. It is single-threaded, but according to the authors their aim is to include multi-threading into the simulator in the future.

H. Summary

A comparison of the different characteristics for reviewed P2P simulators is shown in Table I. Overlay with Routers column tells if the simulator contains both the logical overlay network topology and the underlying router structure of the physical network. After surveying the existing literature about P2P simulators it is obvious that there is a need for standardization in the area of P2P simulation [7]. The field is highly fragmented and most of the current projects use their own simulators tailor-made for their purposes. One of the problems of the widely used simulators is their complexity and therefore poor scalability for P2P simulation purposes.

Although our project strongly supports the call for general open-source P2P simulator that is easily extensible and even some attempts for such a simulator have recently appeared, our research area is still too specified to be implemented satisfactorily in any other way than building a specifically optimized simulator. Training neural nets is computationally very demanding and requires parallel computing and simplified network simulation. To the best of our knowledge P2PRealm is the only message-level simulator that allows simulations on parallel machines. In addition, there is no other neural networks based P2P algorithms that we know of and neither simulators supporting neural network algorithms.

The problem of specifically built simulators is that the results are not exactly similar with the ones made by other simulators. This is a compromise that had to be made. In P2PRealm the most common P2P resource discovery algorithms are implemented to allow comparison with the ones neural networks create. This provides the baseline for results obtained in other simulators.

III. PEER-TO-PEER REALM

Peer-to-Peer Realm (P2PRealm) is a Java based peer-to-peer network simulator designed for optimizing neural networks used in P2P networks. The simulator has been developed in Cheese Factory peer-to-peer research project [4]. With the simulator, it is possible to determine a certain P2P network scenario and requirements for a resource discovery or topology management algorithm and get as an output a neural network optimized for that scenario. For example, a P2P network scenario could contain Gnutella's [16] topology, resource distribution and query pattern and the requirements could state that we want an algorithm, which needs to locate certain amount of resources (say 150) using as few query packets as possible. The end result would be an adapted resource discovery algorithm for that particular P2P network scenario. The first results of this kind of an algorithm development was reported in [23]. Also, the simulator contains implementations of various P2P resource discovery algorithms such as Breadth-First Search [13], Random Walker [12], Highest Degree Search [1,22] and optimal path K-Steiner Tree approximation [22]. These algorithms can be used as performance measures for neural network based algorithms or for studying their performance in different P2P network scenarios. The simulator has also been used for studying topology management algorithms for P2P networks [3].

The simulator is divided into four parts: P2P network, P2P algorithms, neural network optimization and input/output interface. P2P network contains the characteristics of a P2P network including the network topology, distribution of resources and query patterns of P2P network users. P2P algorithms contains the implementations of various resource discovery and topology management algorithms. Neural network optimization takes care of neural network structure and different optimization algorithms used for training the neural network structure. Input/output interface is used for reading configuration files and for outputting the statistics of training and final results. The final results consist of the optimized

TABLE I. CHARACTERISTICS OF THE CURRENT UNSTRUCTURED P2P NETWORK SIMULATORS

	Level of Detail	Parallel	Scalability	Overlay with Routers	Dynamic Network	Programming Language
NS-2	Packets	Yes	Very low	Yes	No	C++
PLP2P	Packets	Yes	Medium	-	-	C++
QueryCycle	Messages	No	?	Yes	Yes	Java
3LS	Messages?	No	Very low (<1000 peers)	Yes	?	Java
PeerSim	Messages	No	Very high (10^6 peers)	Yes	Yes	Java
NeuroGrid	Messages	No	High (300 000 peers)	No	Yes	Java
GPS	Messages	No	?	No	Yes	Java
P2PRealm	Messages	Yes	Medium (100 000 peers)	No	Yes	Java

neural network and the used query paths for different queries.

IV. Input and Output Interfaces

The following information is required as an input to P2PRealm (described in a configuration file):

- P2P network topologies containing the resource distribution
- Query pattern
- P2P resource discovery algorithm
- Percentage of available resource instances to be located in each query
- Number of queries executed in each training generation
- Neural network inputs
- Number of training generations, number of neural networks and the neuron structure of neural networks
- Optimization method

As an output Peer-to-Peer Realm (P2PRealm) provides the following files:

- The used topology and neighbor distribution
- A trace of training process with separate files for training and generalization sets
- The best and all neural networks of each generation
- Query routes started from each node of the P2P network
- Configuration file, which was given as an input

V. Training NeuroSearch

Next, we briefly describe how P2PRealm can be used for P2P algorithm development. As an example we use NeuroSearch resource discovery algorithm [23], but other algorithms for example topology management algorithms based on neural networks could be used [9].

NeuroSearch resource discovery algorithm uses local information about query situation in a peer-to-peer network to decide if query should be forwarded to a neighboring peer node or not. The local information can be e.g. number of hops the query has traveled, number of replies still needed to be located etc. The forwarding process is illustrated in the following algorithm:

```
1.    One peer node starts a query specifying a
      resource it wants to locate.
2.    For each neighbor the node has, do the
      following:
2.1       Fill all the input fields of neural network.
2.2       Compute the output of neural network.
2.3       If output is greater than zero, then forward
          the query to neighbor and increase the
          number of sent query packets by one.
3.    A forwarded query packet arrives to peer node. If
      this is the first query packet arriving to this node,
      check whether the peer node contains a resource
      being queried. If peer has the queried resource
      then increase the number of found resources by
      one.
4.    Go to step 2.
```

The algorithm terminates when there are no more query packets to process. At the end the quality of neural network is determined by the number of found resources and the number of query packets used.

To get good neural networks, they need to be trained so the algorithm has to be executed many times (typically millions query executions). There are various neural network weight adjusting algorithms and depending on the used methods the training times can vary a lot. Still, all optimization methods have in common iterative behavior and therefore executing the algorithm efficiently is an important feature of the simulator.

The internal execution loops of P2PRealm used for training NeuroSearch are illustrated in Fig. 1. Each simulation run can have multiple simulation cases, where each case has its own environment parameters according to the input information described in section 4. Furthermore, each case produces NeuroSearch resource discovery algorithm optimized to this environment accordingly. With multiple cases it is possible to do parameter sweeps and to eliminate the need of starting the simulator manually each time one wants to use multiple training environments. The execution of different cases can also be distributed on Peer-to-Peer Distributed Computing platform [11] further described in section 6.

Execution of one case is divided into three different sections:

- Training of the neural networks
- Analyzing the training of best neural network in generalization environment
- Analyzing routes of the best neural network after the training

First, the case has its P2P networks, neural networks and other parameters initialized. Then the simulator proceeds to the training phase. In each generation multiple neural networks are evaluated by forwarding queries according to the resource discovery algorithm presented above. The queries are forwarded in one or more P2P networks and statistics of the query performance of each neural network is recorded at the same time. Usually between generations it is worth to do more specific analysis of the best neural network in generalization

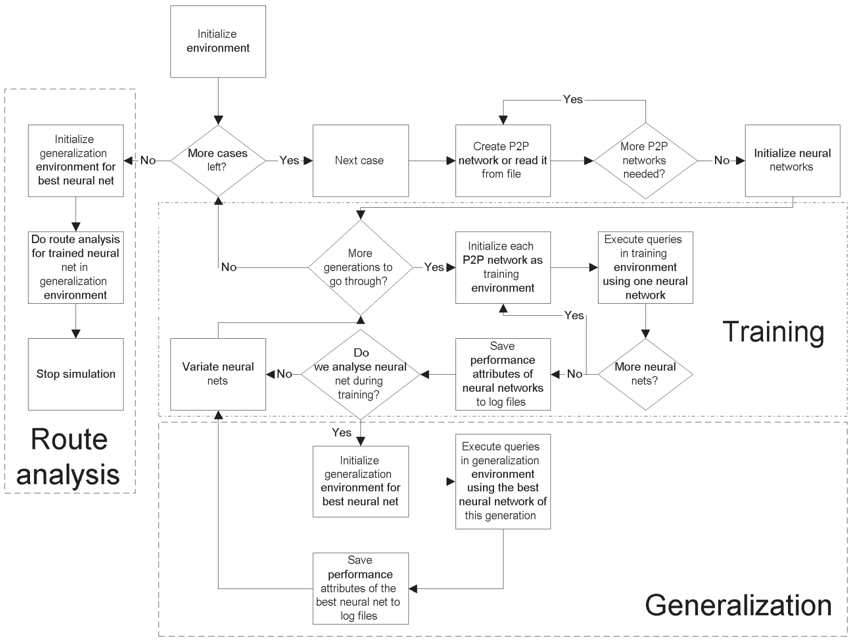

Initialize environment
More cases left?
Yes
No
Next case
Create P2P network or read it from file
More P2P networks needed?
Initialize neural networks
Initialize generalization environment for best neural net
Do route analysis for trained neural net in generalization environment
Stop simulation
Route analysis
More generations to go through?
Initialize each P2P network as training environment
Execute queries in training environment using one neural network
More neural nets?
Save performance attributes of neural networks to log files
Do we analyse neural net during training?
Variate neural nets
Training
Initialize generalization environment for best neural net
Execute queries in generalization environment using the best neural network of this generation
Save performance attributes of the best neural net to log files
Generalization

Figure 1. Execution Loops of P2PRealm

environment, where we can determine how the same neural network performs in an unknown environment. Generalization environment can be used to control when neural network is specializing too much on training environment and loses the ability to perform well in unknown but similar environments than the training environment.

After the evolution has proceeded the predetermined amount of generations, the simulator moves to the last phase of the process: route analysis. In the route analysis the same generalization environment is initialized as earlier for the best neural network, but now the queries start from each peer node at a time. The query paths produced by these queries are recorded and written to files to get accurate data about input and output values of neural network during a query. Finally, when routes have been recorded, the simulation ends.

VI. Speeding the Execution with P2PDisCo

The first implementation of P2PRealm used approximately one week for training the neural networks on a desktop computer. This was a severe limitation in research because it forced to study only small P2P networks and still getting results was very time consuming.

We started the internal code optimization process to see how much can be saved by optimizing internal loops of the simulator. After P2PRealm was profiled we found that the use of Vector object instead of Array in Java consumed lots of time (in particular getting the size of a vector through method call). Java container classes such as HashMap and HashTable can contain only objects and therefore reimplementing them to store only primitive values saved some execution time. Also we found that caching results of different method calls to avoid new method calls resulted in significantly faster execution times. The total time decreased to about 60% with these optimizations.

Java bytecode is interpreted in Java virtual machine yielding slower execution compared to compiled code. Java Native Interface [20] has been developed to allow native code for example compiled C++ to be executed from a Java program. We reimplemented the calculation of neural network output with C yielding an execution time about 70-80% compared to first version of P2PRealm. Combining both the internal code optimization techniques and Java Native Interface implementation of neural network output calculation, we thus achieved execution time of about 50% compared to first version of P2PRealm.

This was however not enough, because reducing execution time of one simulation case from a week to 3-4 days was still quite slow. As a solution, we started developing Peer-to-Peer Distributed Computing platform (P2PDisCo) [11] allowing the distribution of simulation cases to multiple machines.

Earlier in our project [4] we had developed Chedar P2P middleware [2], which provided the basis for building P2PDisCo on top of it. P2PDisCo allows the workstations joined in a Chedar P2P network to publish certain distributed computing application as a resource in Chedar P2P network. When other Chedar nodes find this resource, it can be used to deliver needed input files to computing nodes and the produced output files to the node, which started the computations. For further information on the behavior of P2PDisCo the reader is referred to [11].

The speed up of execution with P2PDisCo is nearly linear, because each simulation case is delivered to different workstation. In university environment it is easy to locate machines, which are idle most of the time, so getting hundred of machines (and thus 100 times faster execution) was relatively easy scaling the research process to much faster rates. The resulting architecture is shown in the Fig. 2. Master denotes the peers, which create simulation cases and P2PRealm denotes the peers, which compute these cases.

VII. Visualization of Data using P2PStudio

Peer-to-Peer Studio (P2PStudio) [10] is a monitoring, controlling and visualization tool for P2P networks research. When combined with P2PRealm only visualization features can be used, because current version of simulator does not provide monitoring data during execution of a simulation. For visualization, P2PStudio provides functionalities to draw network topology and different graphs e.g., neighbor distribution of the topology. Also, the location of resources and query paths can be illustrated on a screen to qualitatively analyze how algorithms are performing. In case, that the simulation contains neural network, the input and output values of a certain query will be shown in a separate table. A screenshot of P2PStudio is shown in Fig. 3 and the specific features of P2PStudio are described in separate article [10].

VIII. Conclusions

Peer-to-Peer Realm is a simulator for studying P2P networks. Its unique functionalities contain training methods for neural networks and optimized speed of execution. By combining P2PRealm with other tools developed in our project, the simulator can grow to a large-scale distributed P2P research environment.

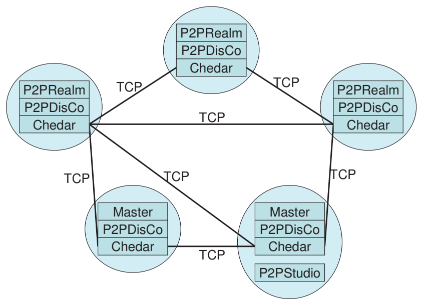


Figure 2. Architecture of P2PRealm combined with P2PDisCo, Chedar and P2PStudio

The future work of P2PRealm includes the parallelization of simulation such that multiple computers can process the same simulation task. Now only one simulation task can be allocated to a certain computer and speed ups are gained only when multiple cases are being simulated. Also, with the advent of multi-core processors for desktop machines, we are going to implement threaded version of simulator to support multiple processors within a single computer. For P2P network visualization, P2PStudio's user interface can be replaced in the future to support large P2P networks to be visualized and better usability of the program. Also the list of improvements for P2PRealm contain different query distributions and new input types for neural networks. As a longer term goal, we are aiming to combine neural network based topology management algorithms with neural network based resource discovery algorithms to study optimal construction of P2P networks.

References

[1] Adamic L., Lukose R. and Huberman B., Local Search in Unstructured Networks, *Handbook of Graphs and Networks: From the Genome to the Internet*, Wiley-VCH, 2003, 295-317.

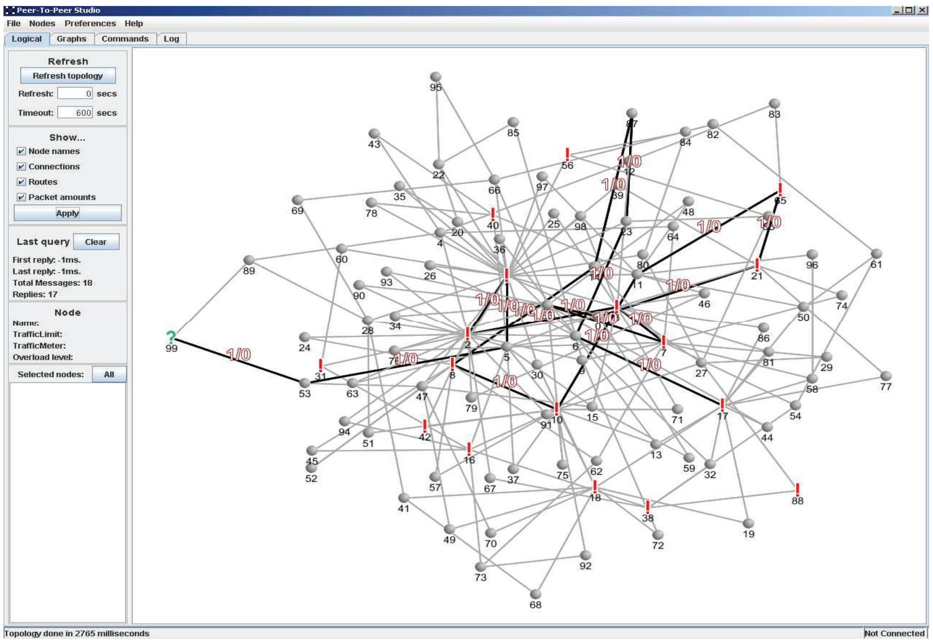
[2] Auvinen A., Vapa M., Weber M., Kotilainen N. and Vuori J., "Chedar: Peer-to-Peer Middleware", Proceedings of the 20th IEEE International Parallel and Distributed Processing Symposium (IPDPS 2006), Rhodes Island, Greece, 2006.

[3] Auvinen A., Vapa M., Weber M., Kotilainen N. and Vuori J., "New Topology Management Algorithms for Unstructured Peer-to-Peer Networks", unpublished.

[4] Cheese Factory –project, http://tisu.it.jyu.fi/cheesefactory

[5] Clarke I., Sandberg O., Wiley B. and Hong T., "Freenet: A distributed anonymous information storage and retrieval service", *Proceedings of Workshop on Design Issues in Anonymity and Unobservability (ICSI),* Berkeley, CA, USA, 2000.

[6] He Q., Ammar M., Riley G., Raj H. and Fujimoto R., "Mapping Peer Behavior to Packet-level Details: A Framework for Packet-level Simulation of Peer-to-Peer Systems", *Proceedings of the 11th IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer Telecommunications Systems (MASCOTS 2003),* Orlando, USA, 2003.



Refresh: 0 secs

Timeout: 600 secs

✓ Node names

✓ Connections

✓ Routes

✓ Packet amounts

First reply: -1ms.
Last reply: -1ms.
Total Messages: 18
Replies: 17

Name:
TrafficLimit:
TrafficMeter:
Overload level:

Topology done in 2765 milliseconds

Not Connected

Figure 3. P2PStudio User Interface for P2PRealm

[7] Joseph S., "An Extendible Open Source P2P Simulator", *P2P Journal*, November 2003, 1-15.

[8] Joseph S. and Hoshiai T., "Decentralized Meta-Data Strategies: Effective Peer-to-Peer Search", *IEICE Transactions on Communications*, Vol.E86-B, No.6, 1740-1753.

[9] Keltanen T., "NeuroTopology: Topology Management Algorithm for P2P Networks", unpublished.

[10] Kotilainen N., Vapa M., Auvinen A., Weber M. and Vuori J., "P2PStudio - Monitoring, Controlling and Visualization Tool for Peer-to-Peer Networks Research", unpublished.

[11] Kotilainen N., Vapa M., Weber M., Töyrylä J. and Vuori J., "P2PDisCo - Java Distributed Computing for Workstations Using Chedar Peer-to-Peer Middleware", *Proceedings of the 19th IEEE International Parallel & Distributed Processing Symposium (IPDPS 2005)*, Denver, Colorado, USA, 2005.

[12] Lv Q., Cao P., Cohen E., Li K. and Shenker S., Search and Replication in Unstructured Peer-to-Peer Networks, *Proceedings of the 16th International Conference on Supercomputing*, ACM Press, 2002, 84-95.

[13] Lynch N. *Distributed Algorithms*, Morgan Kauffmann Publishers, 1996.

[14] Montresor A., Di Caro G. and Heegaard P., "Architecture of the Simulation Environment", Technical Report: D11, BISON project, University of Bologna, 2003.

[15] NS-2, http://www.isi.edu/nsnam/ns/

[16] Oram A., *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*, O'Reilly Media, 2001.

[17] PDNS - Parallel/Distributed NS, http://www.cc.gatech.edu/computing/compass/pdns/

[18] PeerSim, http://peersim.sourceforge.net/

[19] Schlosser M., Condie T. and Kamvar S., "Simulating a P2P File-Sharing Network", *1st Workshop on Semantics in Grid and P2P Networks,* 2002.

[20] Sun Microsystems, Java Native Interface Specification, http://java.sun.com/j2se/1.5.0/docs/guide/jni/spec/jniTOC.html

[21] Ting N. and Deters R., "3LS - A Peer-to-Peer Network Simulator", *Proceedings of the 3rd International Conference on Peer-to-Peer Computing (P2P 2003)*, IEEE Press, 2003, 212-213.

[22] Vapa M., Auvinen A., Ivanchenko Y. Kotilainen N. and Vuori J., "Optimal Resource Discovery Paths of Gnutella2", unpublished.

[23] Vapa M., Kotilainen N., Auvinen A., Kainulainen H. and Vuori J., "Resource Discovery in P2P Networks Using Evolutionary Neural Networks", *International Conference on Advances in Intelligent Systems - Theory and Applications (AISTA 2004)*, Luxembourg, 2004.

[24] Yang W. and Abu-Ghazaleh N., "GPS: A General Peer-to-Peer Simulator and its Use for Modeling BitTorrent", *Proceedings of the 13th Annual Meeting of the IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS '05)*, Atlanta, USA, 2005.

PXII

P2PSTUDIO – MONITORING, CONTROLLING AND VISUALIZATION TOOL FOR PEER-TO-PEER NETWORKS RESEARCH

by

Niko Kotilainen, Mikko Vapa, Annemari Auvinen, Matthieu Weber and Jarkko Vuori 2006

In Proceedings of the ACM international workshop on Performance monitoring, measurement, and evaluation of heterogeneous wireless and wired networks, pages 9-12

P2PStudio – Monitoring, Controlling and Visualization Tool for Peer-to-Peer Networks Research

Niko Kotilainen, Mikko Vapa, Annemari Auvinen, Matthieu Weber, Jarkko Vuori
Department of Mathematical Information Technology
University of Jyväskylä, Finland
firstname.lastname@jyu.fi

ABSTRACT

Peer-to-Peer Studio has been developed as a monitoring, controlling and visualization tool for peer-to-peer networks. It uses a centralized architecture to gather events from a peer-to-peer network and can be used to visualize network topology and to send different commands to individual peer-to-peer nodes. The tool has been used with Chedar Peer-to-Peer network to study the behavior of different peer-to-peer resource discovery and topology management algorithms and for visualizing the results of NeuroSearch resource discovery algorithm produced by the Peer-to-Peer Realm network simulator. This paper presents the features, the architecture and the protocols of Peer-to-Peer Studio and the experience gained from using the tool for peer-to-peer networks research.

Categories and Subject Descriptors

C.4 [**Performance of Systems**]: Measurement techniques

General Terms: Measurement, Performance.

Keywords

peer-to-peer; P2PStudio; monitoring tool; research infrastructure.

1. INTRODUCTION

Peer-to-Peer (P2P) networks consist of a set of peer nodes. Each peer node makes decisions on where to connect and where to forward resource queries resulting in a complex self-organizing network. Studying how different algorithms are performing requires collecting data from the entire P2P network to obtain a global view. In P2P networks research people have used crawlers [5,9] to collect data locally available for some peer nodes. This approach however is only able to gather a portion of the P2P network's behavior, because some of the peers might not accept any new connections requested by the crawlers. Also, the crawlers can only gather information, which is accessible by the P2P protocol and thus they do not have direct means to control the peer's actions.

In our approach, we use a centralized server to contact peers in the P2P network and to set filters to the peers for what events the peers need to report back to the server. This allows measuring different properties from the P2P network extensively and globally. The graphical user interface presents the collected data visually thus making the interpretation easier compared to reading plain text log files. In contrast to crawlers, we note that our work is the first attempt to create a P2P research environment, which provides strict control mechanisms and accurate measurements for studying the behavior of different P2P algorithms.

To monitor the events of a P2P network a specific monitoring interface needs to be implemented in the peer nodes. This interface is used for setting different event logging options and for accepting incoming connections for data delivery from the centralized server. However, in presence of a large P2P network the centralized server can have lots of connections to manage and presents a potential performance bottleneck in our approach compared to local gathering of data done by crawlers. This architecture can however be scaled up by using multiple servers as is common in studies with crawlers [9].

The rest of the paper is organized as follows. Section 2 presents P2PStudio, its features, architecture and protocols. Section 3 describes how P2PStudio has been used in peer-to-peer networks research for studying the performance of peer-to-peer resource discovery and topology management algorithms. Conclusions and future work are discussed in Section 4.

2. PEER-TO-PEER STUDIO

The Cheese Factory –project [3] has implemented a Java-based peer-to-peer computing platform called Chedar [1]. Chedar can be used to build a network of workstations where each node provides and consumes resources such as computing power, files and devices. Currently, Chedar is used as a middleware for P2P Distributed Computing applications [7]. Chedar has also been extended to support mobile devices [8]. In order to test and monitor the Chedar network there was a need for a tool that enables to remotely control and monitor each peer and workstation in a centralized way. By executing the Guardian student project [4], the first version of Peer-to-Peer Studio was developed in 2002.

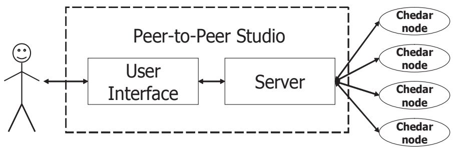


Figure 1. Components of Peer-to-Peer Studio.

P2PStudio is Java-based and it is divided into two separate programs as shown in Fig. 1: the user interface (UI) and the server. The graphical UI connects to the server program and uses it to carry out the commands entered by the user. The server program takes care of all of the communication between the UI and Chedar nodes. It also manages the data sent from Chedar nodes. Dividing the application into two programs allows mobility of the UI from the dedicated hardware of the server. For example the server might have privileges to connect to Chedar nodes through firewalls and an UI residing on a laptop only needs to be able to connect to the server.

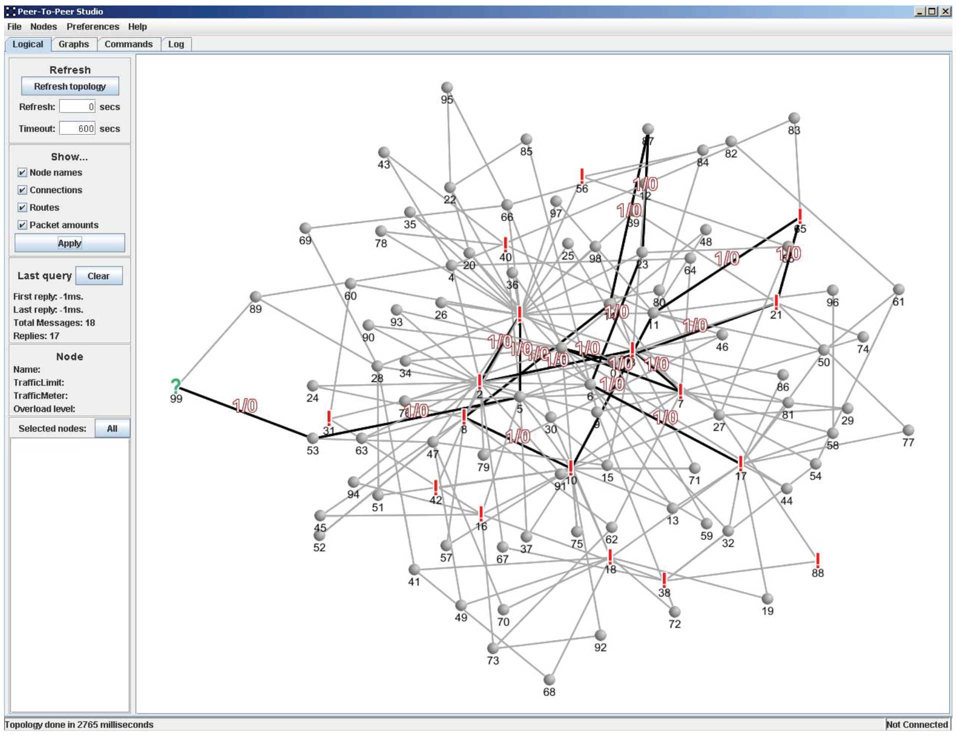
UI communicates with the server, sends requests to Chedar nodes, displays data from the server to the user e.g., by visualizing the network topology and showing diagrams. The UI also allows the management of Chedar nodes. Server forwards the commands sent by the UI, gathers information from the Chedar network and passes on requested data to the UI.

2.1 User Interface

The user interface draws a logical topology of the monitored network as shown in Fig. 2. From the zoomable topology view the user can select nodes and for example check their values, command queries to be sent and modify the resources owned by the nodes. Nodes can also be grouped together to ease the execution of a certain action to multiple nodes. Information on the last executed query is also shown in the topology view. The topology is generated using the WTS Veivi component from WTS Networks [12]. The component creates a visualization of network topology from a set of nodes and links optimized to minimum number of overlapping links. The topology is refreshed whenever the user desires or after a set interval.

Another feature of the UI is to show graphs of the monitoring data as shown in Fig. 3. Currently, the only graph implemented is the neighborhood distribution, but other graphs are relatively easy to be plugged in. Graphs are formed by combining multiple events into a single value, like in the neighborhood distribution, where individual neighbor amount notifications are counted and the frequency of certain value creates one data point in the graph. Graphs can be zoomed and shown also in a logarithmic scale.

The log feature of the UI allows the user to keep track of the Chedar network's actions almost in real time. Log presents the event messages coming from the Chedar nodes. The events are notifications of certain network events, for example forwarded queries, new neighbor connections or dropped messages because of congestion in a Chedar node.



Refresh: 0 secs

Timeout: 600 secs

☑ Node names
☑ Connections
☑ Routes
☑ Packet amounts

First reply: -1ms.
Last reply: -1ms.
Total Messages: 18
Replies: 17

Name:
TrafficLimit:
TrafficMeter:
Overload level:

Topology done in 2765 milliseconds

Not Connected

Figure 2. Topology view.

The user can also send commands to the server or to Chedar nodes via the server by typing commands in the User Interface-to-Server Message Protocol (UMP) format (for more details see the Section 2.3). The Commands view allows the user to see the sent data and the received messages from the nodes. Also batch files can be executed via the commands view. Batch files are useful when a certain peer-to-peer query pattern and measurement scenario needs to be executed multiple times.

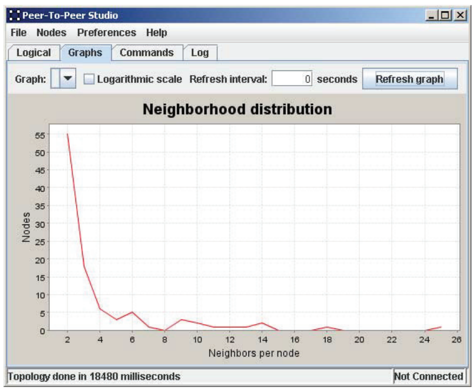


Figure 3. Graph view.

The UI can be run online as well as offline especially for demonstrations. For offline use there is a recording feature allowing the user to record actual monitoring data coming from the server to a file and later retrieve the recorded data in offline state. The UI also allows the user to create Chedar node groups and manage connections.

The functioning of the UI is quite simple. When data is received from the server it is checked and forwarded to the addressed component of the UI. The data will be presented to the user in a form of topology, graph or text depending on the view. Sending data is also rather straightforward. The user assigns a command and it is sent to the server for further handling.

2.2 Server

The server program is divided into two main components: stateless connection manager and stateful data manager. The connection manager is the part of the server which takes care of all connections. It forwards the contents of the packets without interpreting them, only adding metadata about the time the packet was received and Chedar node's IP address and port. A packet can arrive to the server either from the UI or from a Chedar node. It arrives first to the connection manager which forwards it to the data manager if necessary, otherwise directly to UI or to Chedar node(s).

The data manager is responsible for temporarily saving data coming from Chedar nodes and for combining multiple individual replies to a single reply for UI. For example to construct a neighbor distribution graph, data manager needs to collect individual neighbor amounts from Chedar nodes and build the graph data for UI. This lightweight architecture of the server allows scaling to hundreds of Chedar nodes.

2.3 Protocols

User Interface, Server and Chedar nodes use three different protocols for communication. One binary protocol was developed as a container for two message protocols, one XML protocol for communication between the server and the Chedar nodes as well as one XML protocol for communication between the UI and the server. Both XML protocols are on the top of the binary protocol as illustrated in Table 1. The binary protocol is always on the top of TCP.

Table 1. LAYERS OF THE PROTOCOLS.

Message Protocol (GMP or UMP)	*XML*
Packet Transmission Protocol (GPTP)	*Binary*
TCP	

1) Guardian Packet Transmission Protocol (GPTP)

The Guardian Packet Transmission Protocol (GPTP) is a binary protocol used between the UI and the server as well as between the server and the Chedar nodes. The GPTP packets are composed of a fixed-size 64-bit header and a data part, which varies in size. The header identifies the packet as a part of the Guardian-to-Chedar protocol and specifies the size of the data part in bytes. Without a specified data size, parsing an incoming XML message from a stream would be harder. An example of a GPTP message is shown in Table 2.

Table 2. GUARDIAN PACKET TRANSMISSION PROTOCOL.

`32 bit synchronization header, 0x47324350 (G2CP)`
`32 bit size field, network byte order, (1234)`
`Byte data`

2) Guardian Message Protocol (GMP)

The Guardian Message Protocol (GMP) is used between the server and the Chedar nodes on the top of the Guardian Packet Transmission Protocol. Each GMP message is a complete XML document. The header is a standard XML declaration, and the body is composed of a root element which specifies the type of message, and a variable content.

Here is the structure of GMP message:

Header: XML declaration

<?xml version="1.0" encoding="UTF-8"?>

Body

Root element: <request/> OR <reply/> OR <event/>

Content: various requests, replies or events as

XML elements and/or attributes

There are three types of messages in the Guardian Message Protocol:

Request	message is sent by the server to a Chedar or a Workstation node.
Reply	message is sent by a Chedar or a Workstation node to the server.
Event	message is sent by a Chedar node to the server.

The request/reply pair forms a synchronous message exchange initiated by the server. The reply is not mandatory. Event messages can arrive from the Chedar nodes at any time.

3) User Interface-to-Server Message Protocol (UMP)

The User Interface-to-Server Message Protocol (UMP) is used between the UI and the server on top of the Guardian Packet Transmission Protocol. UMP uses similar message structure as GMP. The difference between UMP and GMP is in the XML elements and attributes. For example the UMP contains elements for sending a certain GMP message to all Chedar nodes.

3. P2PSTUDIO IN PEER-TO-PEER NETWORKS RESEARCH

At first, P2PStudio was developed to collect data from a Chedar network [1] consisting of tens of workstations. Experimenting with self-organization of topology and different resource discovery algorithms however usually requires a controlled environment to obtain results that are repeatable. Creating exactly same starting conditions for each test in a network of workstations is problematic, because of differencies in hardware and network traffic. Also, having each Chedar node pack and send data over the network is significantly slower than executing algorithms in a simulator, where only local data structures are being used.

Therefore, the use of P2PStudio was extended by creating the Peer-to-Peer Realm (P2PRealm) network simulator [10,6]. P2PRealm is Java-based and contains functionalities for creating peer-to-peer network scenarios with different topologies, resource distributions and query patterns, executing different resource discovery and topology management algorithms, and collecting various statistics of the execution to log files. In addition to textual viewing of log files, P2PStudio can be used for graphical viewing e.g., to plot how queries spread in the network and what kind of topologies emerge from the execution of algorithms.

A special use case for P2PStudio and P2PRealm is the development of the NeuroSearch resource discovery algorithm [11], which is based on neural networks. Optimizing neural networks requires not only simulation of a certain scenario once, but usually thousands of times to reach a near-optimum state in learning. Therefore network simulators, such as Ns-2 [2], which are based on scripting languages and mainly developed for detailed protocol studies are not fast enough. For studying the behavior of neural networks, P2PStudio provides a view containing the inputs of neural network and the corresponding output decisions.

4. CONCLUSIONS AND FUTURE WORK

P2PStudio is a well-established research tool for peer-to-peer networks research providing functionalities for peer-to-peer network monitoring, controlling and visualization. P2PStudio has been used with two different peer-to-peer software, Chedar and P2PRealm, for algorithm development. The centralized architecture of P2PStudio is a potential bottleneck for scalability in the future when the size of the P2P networks being studied grows. As a future work we envision changes in the architecture to support multiple servers as well as adding new functionalities to UI to determine certain network characteristics such as diameter, shortest paths and multiple distinct paths between nodes.

5. ACKNOWLEDGMENTS

The authors would like to thank the other members of the Guardian student project: Joni Töyrylä, Jussi Rastas and Ville Pentti. Niko Kotilainen was supported by the InBCT-project and Mikko Vapa and Annemari Auvinen were supported by the GETA graduate school.

6. REFERENCES

[1] A. Auvinen, M. Vapa, M. Weber, N. Kotilainen, and J. Vuori, “Chedar: Peer-to-Peer Middleware”, *Proceedings of the 20th IEEE International Parallel & Distributed Processing Symposium (IPDPS 2006)*, Rhodes Island, Greece, Arpil 2006.

[2] L. Breslau, D. Estrin, K. Fall, S. Floyd, J. Heidemann, A. Helmy, P. Huang, S. McCanne, K. Varadhan K. , X. Ya, and Y. Haobo, “Advances in network simulation”, IEEE Computer, Vol. 33, Issue 5, pp. 59-67, 2000.

[3] Cheese Factory – Peer-to-Peer Computing Project, tisu.it.jyu.fi/cheesefactory.

[4] Guardian project, www.mit.jyu.fi/opiskelu/sovellusprojektit/guardian/.

[5] M. A. Jovanovic, F. S. Annexstein, and K. A. Berman, “Scalability Issues in Large Peer-to-Peer Networks – A Case Study of Gnutella”, Technical report, University of Cincinnati, 2001.

[6] N. Kotilainen, M. Vapa, A. Auvinen, T. Keltanen, and J. Vuori, ”P2PRealm – Peer-to-Peer Network Simulator”, *Proceedings of the 11th International Workshop on Computer-Aided Modeling, Analysis and Design of Communication Links and Networks (CAMAD 2006)*, Italy, June 2006 .

[7] N. Kotilainen, M. Vapa, M. Weber, J. Töyrylä, and J. Vuori, "P2PDisCo – Java Distributed Computing for Workstations Using Chedar Peer-to-Peer Middleware*", Proceedings of the 19th IEEE International Parallel & Distributed Processing Symposium (IPDPS 2005)*, Denver, Colorado, USA, 2005.

[8] N. Kotilainen, M. Weber, M. Vapa, and J. Vuori, "Mobile Chedar - A Peer-to-Peer Middleware for Mobile Devices", *Workshops Proceedings of the Third IEEE Conference on Pervasive Computing and Communications (Percom 2005)*, pp. 86-90, Kauai Island, Hawaii, USA, 2005.

[9] D. Stutzbach, R. Rejaie, “Capturing Accurate Snapshots of the Gnutella Network”, *Proceedings of the 8th IEEE Global Internet Symposium*, Miami, Florida, 2005.

[10] J. Töyrylä, "Building NeuroSearch - Intelligent Evolutionary Search Algorithm For Peer-to-Peer Environment", Master's Thesis, University of Jyväskylä, 3.9.2004.

[11] M. Vapa, N. Kotilainen, A. Auvinen, H. Kainulainen, and J. Vuori, "Resource Discovery in P2P Networks Using Evolutionary Neural Networks", *International Conference on Advances in Intelligent Systems - Theory and Applications (AISTA 2004)*, Luxembourg, 2004.

[12] WTS Networks, www.wts.fi