

Calto: A Self Sufficient Presence System for Autonomous Networks

Anthony Dang and Bjorn Landfeldt

adang@it.usyd.edu.au and bjornl@it.usyd.edu.au

Abstract— In recent years much attention has been paid to spontaneously formed Ad Hoc networks. These networks can be formed without central management and without the use of the infrastructure the Internet provides. Lack of infrastructure support brings difficulties in information location and management.

This paper describes Calto, a distributed Self Sufficient Presence System for Autonomous Networks. Calto aspires to solve the problems of the currently untouched areas of presence, notification, and location management in Ad Hoc networks. In order to achieve this, Calto combines presence concepts from Second and Third Generation Mobile networks with Instant Messaging Systems subscriber services while providing distributed DNS style functionality in a Peer-to-Peer setting. It also utilizes locality and key nodes to provide Client-Server like functionality. Calto accommodates for true Ad Hoc environments while being scalable and robust in the face of network instability.

Index Terms—Ad Hoc, Peer to Peer, Resource and Service discovery, Presence, Location, Searching, Distributed Networks.

I. INTRODUCTION

NETWORKING environments are continuously changing and evolving. The more recent introduction of wireless networks such as cellular networks, wireless LAN and PANs (Personal Area Networks), and the possible interconnections between these networks have created a new set of challenges [9], [5]. The relatively new paradigm, the Ad Hoc network [6] introduces dynamically changing networks (nodes can join, leave and move) that can be formed without the need of fixed infrastructure¹. Recently there has been much research in regard to efficient network layer routing to nodes in the wireless domain, however, research into presence [13], [14] notification and location in these networks has so far been untouched. For example, nodes in a network contain resources, services and people. In a centralized network when these services, resources or people leave or move to another node in the network they must advertise their availability and current location to the centralized server. Without this functionality, services such as databases, Instant Messaging (IM) [20], [21] and location and presence services will fail in an Ad Hoc environment.

In this paper, we introduce Calto; A Self Sufficient Presence System for Autonomous Networks. In order to provide a presence service in an Ad Hoc environment, Calto integrates Peer-to-Peer (P2P) [2] concepts, a distributed

lookup architecture, subscriber services and a specialized search algorithm. This combination makes up a truly dynamic system in the face of changing network topologies.

We show that Calto scales well and thus can support a large number of nodes without requiring a large amount of signaling. The remainder of the paper is organized as follows. In section II we describe the different technologies that together constitute the system. In section III, we describe the Calto framework. In section IV we explain the presence information distribution, and search protocols. In section V we describe how Calto networks are formed and updated. In section VI we provide an analysis of the scalability of the system before presenting future work in section VII and finally concluding in section VIII.

II. ENABLING TECHNOLOGIES

Location information is paramount in cellular systems [4]. Link layer technologies such as IEEE 802.11 [15], [16] and Bluetooth [7], [8] have allowed for the wireless scene to be extended out of the domain of centralized systems and into the realm of Ad Hoc networks. These networks are envisioned to operate in unstable network environments and without the need for centralized servers. The advent of the Peer to Peer paradigm has made these hopes attainable. P2P networks are application layer overlay networks [1], able to operate using direct links between individual peers in the underlying network infrastructure.

In any networking system it is important to be able to search for objects, services, resources and people. Flooding can be used to this end, but it has the great disadvantage of large traffic overheads. A technique called Rumor Mongering [3] has also been proposed; however the overhead although smaller is still rather high. Distributed lookup architectures such as Tapestry [10], Pastry [12], CAN and Chord [11] have been proposed for the efficient searching of objects in large scale distributed P2P systems. However, these proposals are not suitable for presence information in highly dynamic wireless networks since their algorithms were designed with rather static fixed networking environments in mind.

In an Ad Hoc environment it should be possible for networks to function even when individual parts become separated, and when they become re-attached. This would involve some sort of a notification system for nodes that require notification of other's presence (e.g. availability and location). This notion of location, notification and presence in an Ad Hoc networks is currently an untouched area. In our work, we use a combination of the above technologies to achieve this functionality and provide a system for locating

¹ Nodes in these networks may be wired or wireless, although a the notion of a truly dynamic environment may be less constrained through wireless connections.

users, nodes and services in highly dynamic wired and wireless environments.

III. CALTO NODE ARRANGEMENT

In an Ad Hoc environment a node may leave the network and in the future rejoin at another location (perhaps geographically distant). This movement is analogous to the GSM (Global System for Mobile Communications) scenario. In GSM a HLR (Home Location Register) plays the role of keeping a table of a node's identifier and its current location. This functionality is akin to that of a Hash table or a lookup service, and can be implemented using distributed lookup technologies. Calto models some of these aspects on Chord as its simplistic design enables various enhancements crucial to Calto's functionality (e.g. efficient searching and adaptation to network merger and partitioning).

A. Basic structure

Calto operates as a P2P application-layer overlay network [1] organized in a ring structure, where each node is ordered in terms of an identifier², and the name space is completely distributed. The architecture provides a distributed lookup service, allowing the insertion, updating, lookup, and deletion of key-values pairs using identifiers as handles to the keys. Examples of searchable identifiers are an email address, IP address, a string name of a service, or even a file name. The identifiers are generated by applying a hash function to the searchable identifier (the identifier known to the person or process conducting a search).

Nodes in Calto are responsible for identifiers that are numerically close to the node's own identifier. For Example, in Figure 1 we can see that because the identifiers 1 and 2 are numerically closer to 1 than to 4, their assignments are to node 1. Node 0 is a special case. The example in Figure 1 depicts a small namespace from 0 to 7. Node 0 is the keeper of identifier 7 because it is mathematically closest to it when the namespace is wrapped around in a ring or circular list.

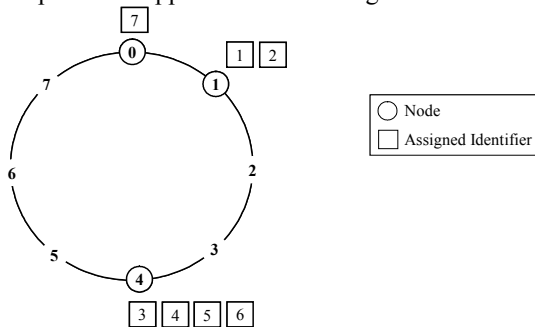


Figure 1 – The Calto Identifier Ring with a namespace of 0-7.

Mirrored copies of node's tables are kept in r consecutive nodes preceding and succeeding a node in the ring. So, when a node leaves a network, the responsibility for its keys is distributed among its immediate successor and predecessors.

² To create node's identifier we take a node's IP address and spread it evenly over a 160bit address space.

In Figure 1 it can be seen that any insertion of a key-value value pair involves, simply, performing a search for a node that has an identifier close to the identifier of the pair, and then inserting the pair at that node. It should be noted that this system can store presence information about anything with a string name. This name can be hashed into an identifier and stored in the system with a mapping to the node identifier that has this service, resource etc.

B. A Multi-Tier Architecture

In any network (large or small) nodes may differ in resource availability (storage capacity, processing power, bandwidth, online availability etc). Calto endeavors to utilize these differences in a hierarchical P2P design, providing scope and flexibility for hybrid [2] client/server functionality (when available) to improve network performance. Specifically, nodes can be elected to be Super Peers, providing a similar function to a server (e.g. Napster hybrid [2], [19] and Kazaa hierarchical [17] approaches). These nodes hold copies of key-value pairs of their children and adjacent Super Peers. This allows peers to make direct connections to Super Peers for faster searching. Desirable properties of these nodes include availability, bandwidth, and in the case of wireless environments, power.

Calto is designed to accommodate for situations where node's resources may be exploited to aid in overall network performance. Super Peers maintain their own logical ring overlaying the Global Ring (see Figure 2), enabling the possibility of information backup and increased performance for information retrieval. The advantage of this design is that it also leaves the network open for the utilization of nodes that may indeed be high powered centralized servers (permanent or otherwise).

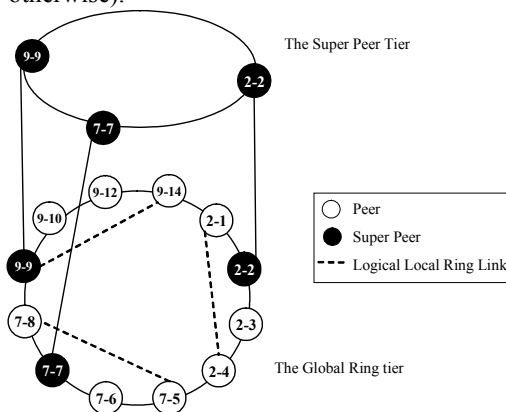


Figure 2 - Local and Global Ring Overlays and the Super Peer Tier

C. Exploiting Locality

Calto endeavors to utilize physical locality and resources to increase network performance, and reduce the cost of sending messages across the physical and data link layers. Calto does this by allowing multiple logical (locality based) rings overlay its Global Ring structure. The aim is to allow Calto rings to function independently in a Local Ring in addition to operating in a Global Ring (see Figure 2).

D. Addressing

When a node leaves and rejoins the network in a different location (with different IP address), a session identifier is used to place the node in the network, enabling the exploitation of locality when nodes move. Searching for a node merely involves searching for its original identifier which will map to its current location (IP address).

To distinguish between Local Rings each node has the identifier of the initial Super Peer as a prefix to their own. This prefix acts as a domain name, and if the Super Peer is indeed a permanent centralized server, then this scheme compliments the current domain scheme of the Global Internet, and can be easily integrated.

IV. PRESENCE AND SEARCHING PROTOCOLS

When a node joins the network it inserts into the system an identifier in the form “nodeID-fn(‘HLR’)” where nodeID is the node’s identifier, fn() is a hash function such as SHA-1, and “HLR” is a string. The value of the pair is simply the node’s current IP address. This pair allows a node to change location, as its HLR will always be a node close to “nodeID” which holds a mapping to its current address. Therefore, a search for a node is firstly a search for its HLR (the node that holds the mapping).

The concept of HLR is taken from GSM networks with the unique exception that this notion of a HLR is merely a construct assigned to a node on the network. In effect, a HLR in Calto may join and leave the network at any time. The nodes to which it is HLR to are then implicitly assigned to one of the relative backup nodes to the HLR, as a search would result in one of these nodes..

A. Subscribing to Presence

Paramount is the need for finding, retrieving, and subscribing to changes in the presence information (e.g. "online at address X" or "offline") of users, nodes or services. In this paper we give the example using network nodes. In centralized Instant Messaging (IM) systems, presence is flagged in a centralized server. This enables any node that comes online to query the server for the nodes in its contact list (nodes to whose presence it has subscribed). Rather than searching and fetching information about these nodes’s presence,

Calto allows for functionality similar to centralized IM systems. When a node comes online, it can deposit a presence flag in a location in the network for the nodes that have subscribed to it, but are not currently online. We call this location a Presence Box (analogous to an email box). If a node has subscribed to others that come online while the node is offline, they will deposit their identifiers and locations in the node’s Presence Box. The insertion identifier is similar to that of a node’s HLR. Specifically, the identifier is in the form “nodeID_{sub}-fn(‘Presence Box’)”. The keeper (and relative backup nodes) of this identifier would merely concatenate this new flag to a list. When the node comes online it searches for “nodeID_{sub}-fn(‘Presence Box’)” and fetches the list of all the

nodes to which it has subscribed that are currently online. Note that this state information is not by any means the only type of information Calto can relay. Any information that can be associated with a presence service can be stored in the Presence Box.

B. Search Algorithm

Calto’s search algorithm is similar to that of Chord’s, as each node stores information about only a small number of other nodes in the network.

1) Routing Tables

To allow for under $O(\log_2 N)$ scalability, the selection of these nodes is based on exponentially increasing distance (node hops). Specifically, each node holds in its routing table the nodes that are $2^0, 2^1, 2^2, 2^3, \dots, 2^{\log N - 1}$ hops from its current position in the ring, where N is the number of nodes in the network.

Note that in Chord, nodes only hold routing entries in the clockwise direction. In Calto we wish to accommodate for network partitioning and independent operation of the partitioned networks. Thus, in Calto routing entries are for both clockwise and anticlockwise directions (see Figure 3 a)). This is necessary since Ad Hoc environments are highly dynamic and segmentation is likely to occur at times. Therefore by allowing nodes to know about other nodes in both directions around the ring, there is a higher chance of either repairing the ring, or allowing the formation of individual rings from the broken off sections.

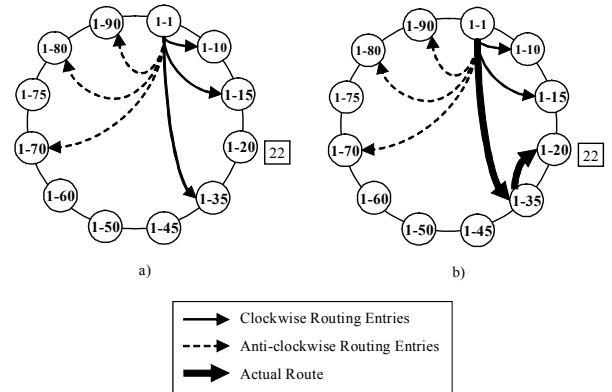


Figure 3 - Calto's Search

2) Searching

When a node initiates a query it will search its routing tables for a node with an identifier that is numerically closest to the search identifier. It then passes the query to that node, and the query is performed recursively until it is routed to the node that contains the key-value pair the identifier belongs to.

It should be noted that although these routes may overshoot the node that is responsible for the queried identifier, the namespace is significantly reduced at each step (see Figure 3 b)). In fact, at each hop we reduce the remaining number of possibilities by half in each direction. So we have a search time of $\frac{1}{2} \log_2 N$.

As an optimization, if the node is a member of a LAN, before it routes a query, it may perform a Local broadcast check for any nodes on its LAN that might have an identifier closer to the target than its own identifier. Even if the query still requires a node to route to outside of the LAN, the number of node hops has decreased.

It should be noted that Calto is not merely limited to searching for nodes. The system is capable of finding email addresses, IP addresses, files, and any string name of a service, resource, device or person. For example, we can search for a person on the network by using their known email address.

We firstly search for a node in the network that has an identifier that is numerically close to “fn(‘email’)”, where email is the email address of the desired person. This identifier will map to the desired person’s nodeID. All that is needed now is to search for “nodeID-fn(‘HLR’)” to get the current IP address. The same will apply to finding files on the network. The identifier “fn(‘filename’)” can be used to find the identifiers of the node(s) that hold this file.

V. CALTO NETWORK FORMATION

A. Insertion into the network

Assuming a node can contact any other node in the network, it will be able to find the place in the ring to insert itself through a search. This may involve a Chord-style search, or directly contacting a Super Peer. The procedure for the new node is to contact the nodes that will be its successor and predecessor in the ring, and have them populate the new node’s routing tables.

A new node X needs to know about nodes that are $2^0, 2^1, 2^2, 2^3, \dots, 2^{\log N - 1}$ hops away from it. If the X is inserted between nodes Y and Z , then X will take on half the routing entries of these two nodes. That is, node Y (the immediate node successor in the clockwise direction) will know about the nodes that are needed in the anti-clockwise direction (and vice versa for node Z). Therefore, all that is needed is for the new node X to contact these nodes to update their routing entries to point to it and not Y . The same approach is applied to the node’s new predecessor Z (the node in the anti-clockwise direction).

This procedure allows for $2 \log_2 N$ nodes to have their routing tables updated, to reflect presence of the new node. It should be noted that it is impossible to update routing tables of all the nodes in a large dynamically changing network without considerable overhead. So the routing entries will approximate the desired node hop distribution. Periodic updating can also be applied by allowing a node to ask its 2^{th} nodes in its routing table if it is their 2^{th} node in the opposite direction.

B. Duplication and departures

In Calto, each node in the network maintains a subset of the key-value pairs, as well as routing table entries that point to a

subset of preceding and succeeding nodes. In a dynamic environment there will be nodes joining and leaving frequently. If we wish to distribute information storage, there must be some duplication to allow for nodes to take over the responsibility of the information of nodes that have departed. In Calto when a key-value insertion is made, the pair is inserted at r consecutive nodes on both sides of the actual keeper of the pair (The quantity r is a configuration parameter that depends on the degree of redundancy required). When a node leaves the network then its successor and predecessor take on the responsibility (implicit though resulting searches) of the key-value pairs that are closest to their identifiers.

C. Super Peer departures

Super Peers may leave the network like any other node. For this reason we introduce Backup Super Peers on each Local Ring. This enables the promotion of a Backup Super Peer to Super Peer status which will save on traffic overheads compared to a complete migration of a Local Ring to another Super Peer.

D. Dynamically changing topologies

To function in a dynamically changing environment Bluetooth allows its Piconets to form and merge independently and function independently when a partition occurs. Calto also adapts to these situations, allowing for previously formed LANs, WANs or MANETs (Mobile Ad Hoc Networks) to merge and partition. Calto also caters for the possibility for rings formed independently to merge and function as one whole network. When two nodes from different networks are able to contact each other we logically break each ring at their start and end points (lowest and highest identifier found by search or Super Peer query), and attach each start point to the end point of the other broken ring, forming a larger ring. In

Figure 4 a),

Figure 4 b) and

Figure 4 c) we can see this progression of joining two networks. As can be seen in

Figure 4 b) and

Figure 4 c) we maintain the Local Ring connections (dotted lines) after the merger into the larger ring. This is to allow the original rings to function, taking advantage of locality. In addition, the Super Peers of the merging rings also get inserted in the correct places on the Super Peer tier, as can be seen in

Figure 4 d).

Calto also allows for rings to be split. Nodes at the ends of each broken arc can search for each other and then form two independent (smaller) rings. In addition, Calto also allows for Local Rings to be *logically* split. This is to take advantage of nodes in the Local Rings that have the ability to be utilized as Super Peers.

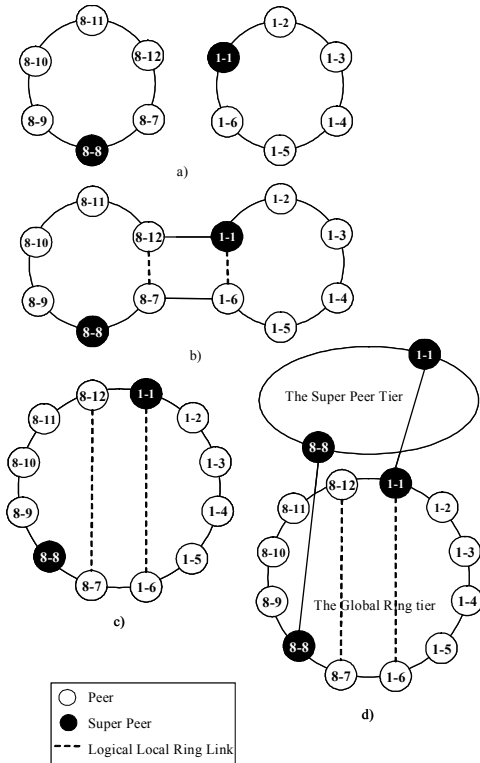


Figure 4 – Merging networks

VI. ANALYSIS

A. Search Scalability

Calto's search algorithm scalability performs with an upper bound of $O(\log_2 N)$ node hops where N is the number of nodes in the network. The convergence distance at each step in the algorithm to the target identifier holds true for the following relation:

$$|k_i| < \left\lfloor \frac{k_{i-1}}{2} \right\rfloor,$$

where k is the converging distance to the target identifier.

As mentioned previously, each node in the network holds in its routing table nodes that are exponentially further apart from it, in terms of node hops, on both sides. These are the nodes that are $2^0, 2^1, 2^2, 2^3, \dots, 2^{\log N - 1}$ hops away. We assert that at every step i , in the algorithm, the maximum node hop 2^i can be at most $2^{(\log N - 1) - i}$ where $2^{\log N - 1}$ is the largest leap in the routing table. Specifically at each step, it is as if we removed 2^i from the possible set of left over routes in all the node's routing tables. This is because the distance between the node and 2^i hops away is equal to the distance between the nodes at 2^i and 2^{i-1} in the routing table. Therefore, for every 2^i we remove from the possible set of next routes, we halve the remaining number of nodes to search. Thus, the

convergence is linear with an upper bound of $O(\log_2 N)$, and an average of $O(\frac{1}{2} \log_2 N)$.

B. Traffic Analysis

Every node has a routing table with at most $O(\log_2 N)$ entries. In a large network, no node can be sure of the number of nodes in the system. So on average each node is in the routing table of $O(\log_2 N)$ nodes.

When a node in Calto joins the network, there must be $O(\log_2 N)$ messages to initialize its routing table, and to update the routing tables of others in the clockwise direction. A further $O(\log_2 N)$ messages are needed in the anticlockwise direction due to our bidirectional routing. In addition a message must be passed to the Super Peer of the Local Ring. Ideally this contact procedure will be implemented using direct connections. Therefore, a node joining or leaving in Calto requires $O(2 \log_2 N)$ messages for routing entries and 1 message for Super Peer notification. It should be noted that in a dynamically changing network we may have the case where a node's routing tables are unable to be updated fast enough to reflect the true topology of the network. Scalable routing is still possible as long as the routing entries still approximate the distances in terms of hops.

To provide an example of actual traffic load we give the following example. Assuming an implementation of IPv6 (128bit IP), a Calto identifier size of 160bits, and accounting for various bit flags that may be needed in the implementation, we have a packet size of approximately 300bits. In a network of size 1,000 nodes ($N = 1,000$), the amount of traffic on the network would be $300 \times 2 \log_2 N$, (approximately 3bytes).

VII. FUTURE WORK

Application-layer routing can sometimes utilize the physical layer poorly. Much research needs to be done in applying application-layer routing in conformance with, and the effects of network layer and link layer routing protocols, especially Ad Hoc routing protocols.

Calto allows the selection of Super Peers in the network. The selection criteria for electing these nodes will vary for each situation. For example, in a wireless environment we have limited bandwidth, varying energy resources and changing levels of mobility. There is a trade-off between these resource constraints that is currently not fully understood and needs to be investigated.

In the wired domain nodes with high bandwidth potential would be ideal as Super Peers as any request would be quickly responded to. On the other hand, if a node has statistically proven to be very sparse in its online availability, then it would not be a good choice to elect it as a Super Peer. The trade-off here is in terms of bandwidth and online availability. Although it is not expected that wired networks would be as unstable as wireless ones, there is still a need to examine the

overhead of Calto's robustness protocols in the face of an extremely frequently changing environment.

VIII. CONCLUSION

The goal of this project was to design an architecture capable of providing the functionality of presence, notification and searching of traditional network models in an Ad Hoc network environment. To date, there has been no research in this area. Calto's unique design allows for application layer overlay networks to form, merge and partition, and is a networking architecture designed for all environments (wired or wireless).

Calto models some of its fundamental attributes (such as its circular topology and search) on the Chord architecture, but there are some fundamental differences. Chord prefers to randomly place nodes in its overlay ring, where Calto endeavors to keep local nodes close together to benefit from topological proximity. While Chord cites advantages of robustness in their method, the research for this paper has also aimed at providing an architecture that would be portable and dynamic (ability to merge and partition) between all domains, wired and wireless, with scarce or plentiful resources. Utilizing locality would be clearly beneficial to a wireless network as bandwidth and power are scarce, and also to a wired environment where LAN resources are able to be exploited. Though, future research in Calto needs to take place to examine the overhead of its robustness protocols in the face of a massively changing environment.

Calto is the first known attempt of implementing presence awareness in Ad Hoc Networks. Its unique integration of Peer-to-Peer design, distributed lookup architectures, subscriber services of IM systems, a specialized search algorithm and cellular network concepts makes it truly unique and dynamic in the face of changing topologies.

REFERENCES

- [1] David G. Andersen, Hari Balakrishnan, M. Frans Kaashoek, Robert Morris, "Resilient Overlay Networks", Proc. 18th ACM SOSP, Banff, Canada, October 2001
- [2] Rüdiger Schollmeier, "A Definition of Peer-to-Peer Networking for the Classification of Peer-to-Peer Architectures and Applications", In Proceedings of the IEEE 2001 International Conference on Peer-to-Peer Computing (P2P2001), Linköping, Sweden, August 27-29, 2001.
- [3] "The Cost of Application-level Broadcast in a Fully Decentralized Peer-to-peer Network" Marius Portmann, Aruna Seneviratne ISCC, Sicily, Italy 2002
- [4] Mouly, Michel, and Marie-Bernadette Pautet. The GSM System for Mobile Communications. France: 1992.
- [5] George. H. Forman and John Zahorjan. The Challenges of Mobile Computing. IEEE Computer, 27(4):38-47, April 1994.
- [6] Carlo Kopp, "Ad Hoc Networking" Published in "Systems" June 1999 pp33-40, <http://www.csse.monash.edu.au/research/san/AdHocNetworks.pdf>
- [7] Jaap C. Haartsen, The Bluetooth Radio System, IEEE Personal Communications, Vol 7, No 1, pp 28-36, February 2000
- [8] C. Bisdikian, An Overview of the Bluetooth Wireless Technology. IEEE Communications 39(12), 86-94, 2001.
- [9] J.D. Solomon, Mobile IP: The Internet Unplugged, Prentice Hall, 1998 Charles E. Perkins, Mobile Networking Through Mobile IP, <http://www.computer.org/internet/v2n1/perkins.htm>
- [10] B. Y. Zhao, J. D. Kubiatowicz, and A. D. Joseph. Tapestry: An infrastructure for fault resilient wide-area location and routing. Technical Report UCB//CSD-01-1141, U. C. Berkeley, April 2001.
- [11] Ion Stoica, Robert Morris, David Liben-Nowell, David R. Karger, M. Frans Kaashoek, Frank Dabek, Hari Balakrishnan, Chord: A Scalable Peer-to-peer Lookup Protocol for Internet Applications. To Appear in IEEE/ACM Transactions on Networking.
- [12] Rowstron and P. Druschel, "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems". IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), Heidelberg, Germany, pages 329-350, November 2001.
- [13] "Common Presence and Instant Messaging (CPIM)", D. Crocker, 14-AUG-02. <http://www.ietf.org/internet-drafts/draft-ietf-impp-cpim-03.txt>
- [14] RFC2779] "Instant Messaging / Presence Protocol Requirements", RFC 2779, February 2000, <http://www.ietf.org/rfc/rfc2779.txt>
- [15] [White Paper] 802.11a: A Very-High-Speed, Highly Scalable Wireless LAN Standard <http://www.proxim.com/learn/library/whitepapers/pdf/80211a.pdf>
- [16] [White Paper] RangeLAN802.11 White Paper- The IEEE 802.11 Wireless Standard <http://144.16.85.58/resources/networking/wireless/80211wp.pdf>
- [17] Kazaa <http://www.kazaa.com>
- [18] The Peer-to-Peer Working Group - <http://www.peer-to-peerwg.org>
- [19] Napster <http://www.napster.com>
- [20] ICQ Instant Messenger <http://www.icq.com>
- [21] Microsoft Instant Messenger <http://messenger.msn.com>