Lookup Service for Peer-to-Peer Systems in Mobile Ad-hoc Networks

Dissertation

submitted in partial fulfillment of the requirements

for the degree of

Master of Technology

by

Kalpesh K. Patel

Roll No: 02329027

under the guidance of

Dr. Sridhar Iyer

and

Dr. Krishna Paul

Kanwal Rekhi School of Information Technology

Indian Institute of Technology, Bombay

Mumbai - 400076.

2004

Dedicated to

My Family

Abstract

A Peer-to-Peer(P2P) system consists of a set of nodes in which every node shares its own resources and services among other nodes in the system. Any node can make query for data, currently available in the network. These peer-to-peer systems are basically overlay networks, work on top of fixed network infrastructure. With the increasing popularity of Mobile Ad-hoc Networks (MANET), it becomes necessary to think of P2P systems which can efficiently work in mobile environments.

Current P2P systems do not fit well into mobile environments where each node have to follow multi-hop path routing and have to face unpredictable mobility. If current P2P systems are deployed on the MANET, application layer and network layer both perform different routing mechanism independent of each other, resulting in multiple layer routing redundancy. Also, routes at the application layer may not be necessarily optimal at the network layer. Thus, we bring down all routing and query lookup mechanism to network layer.

Further, current ad-hoc network protocols do not fulfill the P2P requirements as basically they all are flooding based protocols. Here, we address this problem and propose a novel, controlled-flooding based approach which works at network layer and helps P2P systems to work in mobile environment efficiently.

RINGS is a protocol which implements this approach. RINGS performs query lookup at network layer rather than at application layer, thus eliminating application layer routing overhead. It also reduces query lookup cost by evenly distributing data indices throughout the network, thus reducing network layer routing overhead.

Acknowledgments

I express my sincere gratitude towards my guides **Dr. Sridhar Iyer** and **Dr. Krishna Paul** for their constant help, encouragement and inspiration throughout the project work. Without their invaluable guidance, this work would never have been a successful one. I would also like to thank Kalyan Rao D., Kiran Diwakar, Ranjeeth Kumar, Srinath Perur for their valuable suggestions and helpful discussions. Last, but not the least, I would like to thank the whole KReSIT family which made my stay at IIT Bombay a memorable one.

Kalpesh K. Patel
IIT Bombay
July 9, 2004

Contents

List of Figures

Chapter 1

Introduction

1.1 Peer-to-Peer Systems

Peer-to-Peer are distributed systems without any centralized control or hierarchical organization. Every node in Peer-to-Peer system is functionally equivalent with every other node in the system. The primary goal of P2P system is to share data and services within the network. Most of the P2P protocols can be characterized in three different categories as, protocols with:

1. *Centralized Index Servers*: A set of servers maintain repositories of all the index in the network. All data related queries are answered by these servers. There are many disadvantages using this model including lack of load balancing, single point of failure etc. Because of client/server model, these protocols do not suit with ad-hoc environments. Example, Napster [1].

2. *Flooding-based Approach*: Protocols in this category follow simple broadcast based technique to communicate with peer node in the network. In this model, every client is a server, and vice-versa. These so-called *servents* perform tasks normally associated with both clients and servers. Thus these protocols are a step further to *Centralized Index Servers* based protocols and decentralize the file location process. examples include *Gnutella* [3].

3. *Distributed Hash Index*: Among all P2P protocols, these protocols give better scalability and routing optimization. Still, we observed that performance of these protocols degrades in mobile environments. examples include *Chord* [9], *CAN* [4], etc.

1.2 Mobile Ad-Hoc Networks

Mobile Ad-hoc networks first appeared as DARPA packet radio networks in the early 1970's. They became an important area of research within communication networks and remain a hot topic. To reach their communication partner, they create connections over several intermediate hops as shown in Fig. 1.1. Since wireless access technologies like WLAN 802.11 and Bluetooth are broadly available, ad hoc networks turn from academic considerations into real applications. Both standards are able to create and maintain networks without central entities, and therewith fit into the ad hoc network paradigm of infrastructure free communication.

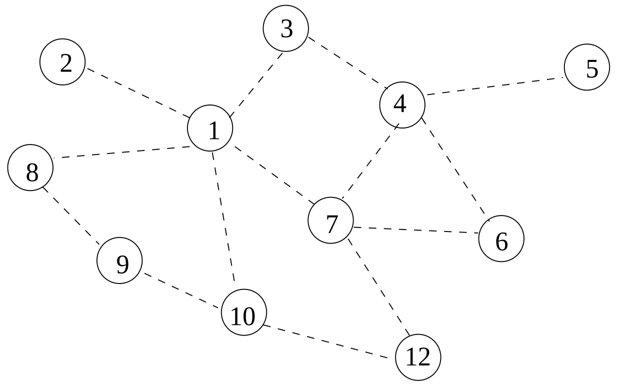

Figure 1.1: Mobile Ad-hoc Network

A MANET consists of mobile nodes which are free to move around arbitrarily. The nodes may be located at different places, and there may be multiple hosts per router. Also, a MANET is an autonomous system of mobile nodes. The system may operate in isolation, or may interface with a fixed network. MANET nodes are equipped with wireless transmitters and receivers using antennas. At a given point of time, depending on the nodes' positions and joining or leaving of nodes, topology of MANET changes accordingly.

1.3 Peer-to-Peer in MANET

A mobile P2P system inherits many of the features of ad-hoc networks:

1. *Self-organizing*: As side effect of the movement of devices in physical space, the topology of a mobile P2P system constantly adjusts itself by discovering new communication links.

2. *Fully decentralized*: Each peer in a mobile peer-to-peer system is equally important and no central node exists.

3. *Highly dynamic*: Since communication end-points can move frequently and independently of one another, mobile P2P systems are highly dynamic.

4. *Low cost*: Wireless ad-hoc networks are built from low-cost transceivers and do not incur charges for provider access and air-time.

At the same place, peer-to-Peer networks realize a comparable approach within fixed network infrastructures. The main characteristic is that P2P networks act independent from the underlying network infrastructure. The utilized TCP/IP only provides a communication platform protocol and does not supply information about the location of requested content or participants. As P2P-nodes independently maintain P2P networks, these overlays show characteristics of an infrastructure free and decentralized network. As an example, Gnutella [3] allow file sharing between distant participants in completely self organizing networks. For signaling, P2P nodes flood search requests and keep alive messages through the network, or distribute them with more sophisticated algorithms to other participants of the network.

Both network architectures are very promising concepts. Although they use different layers for operation, we think that the idea to combine both architectures shows a high potential. The reason being is P2P protocols are not aware of the underlying MANET and assume a fixed network infrastructure, causing additional and unnecessary network traffic.

Common P2P applications have optimized algorithms to find information within their overlay network, but for information exchange they generally rely on TCP and assume stable connections. Whenever connections break, P2P nodes suppose, that their distant communication partners have left the network and switch to other, randomly chosen nodes, which also provide the requested content, or which offer further connectivity to the network to initiate new search requests.

In MANETs, link breaks are common, as all nodes are in motion. Whenever two adjacent nodes move out of each others radio range, the link between both nodes breaks. A MANET protocol not aware of the P2P application, tries to reestablish a new route to the same destination, independent from the necessary effort. Instead of trying to create a new route to the same source of information, other sources could provide the information at less costs after the network topology changed. Therefore the MANET nodes have to report route breaks to the upper P2P node which then decides whether the old source is still utilizable or other connection partners are more appropriate.

P2P networks use multi-hop connections only for search requests. Subsequent information downloads use direct TCP connections between source and destination. The distance between nodes generally does not affect the stability and error free operation of connections. Therefore P2P nodes might create connections to distant nodes, although the searched information might be available more closely as well.

The distances between communication partners in MANETs are counted in number of intermediate hops. It is the most important parameter for route lifetimes. Numerous unnecessary lengthened routes induce additional routing overhead, delay data packet transmissions and therewith greatly reduce the overall network performance. Therefore the application client must classify incoming search replies according to their hop distance to the destination. As a consequence, a combined P2P-MANET approach must be well aware on the one hand of the underlying network infrastructure and on the other hand of the overlaying application.

1.4 Motivation

Despite the fact that ad-hoc networking and P2P computing deal with similar issues, namely discovery, routing and information dissemination, there is not much overlap in the research. Most research on ad-hoc networks focuses on the lower layer of the protocol stack including the link layer, network layer and transport layer. On the other hand, current P2P systems are designed for an Internet-like network infrastructure in which stationary hosts are connected by high bandwidth links. The assumptions, on which these P2P systems are build, are no longer valid in dynamic ad-hoc networking environments. The unique characteristics of such networks require highly adaptable P2P systems that can react to changes in connectivity.

As of today, a lot of work has been done in the area of P2P and MANET independently. P2P protocols and applications are widely available in distributed computing field. All those P2P protocols work as virtual overlay networks on top of fixed network infrastructure. Deploying current P2P protocols on ad-hoc network induces multiple layer redundancy and duplication in terms of messages and communication between nodes. As for example, *Chord* [9] working as overlay network, forms a cluster of nodes on top of TCP/IP layer. The fact is that a pair of nodes, which are neighbors at overlay network, might be far away from each other at network layer as shown in Fig. 1.2. For example, query lookup cost for *Chord* is $log(N)$ at the application layer but due to this fact, the overall cost will be increased.

This problem brings the idea of having communication link between application layer and network layer. If network layer routing information can be provided to application layer, multiple layer redundancy can be avoided.

Here, we suggest a protocol, named RINGS, to optimize overall query lookup cost. RINGS provides query lookup service at network layer rather than at application layer and thus eliminates application layer routing overhead. Further, RINGS provides routing optimization at network layer by *controlled-flooding* approach in which query does not flood the whole network. Thus, by eliminating application layer from routing process and providing network layer routing optimization, RINGS enables P2P systems to work in mobile environments efficiently.

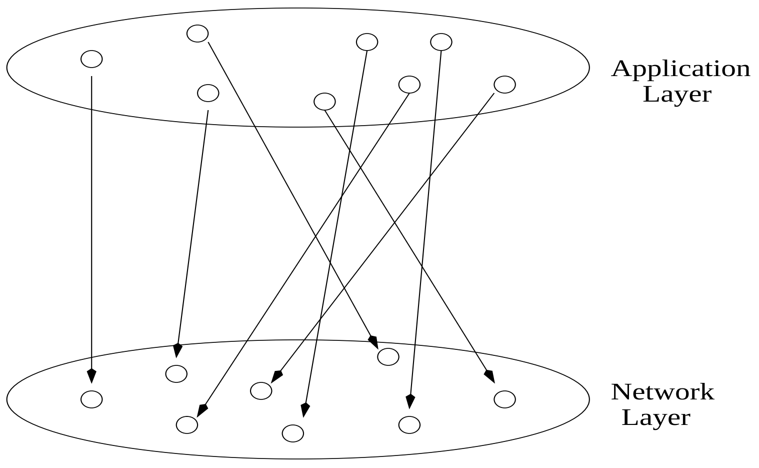

Figure 1.2: Application layer nodes' positions from network layer perspective

1.5 Problem Definition

As we have discussed earlier, the total routing overhead doubles while deploying Peer-to-Peer systems in ad-hoc networks. This overhead is because of separate routing functionality at application layer and at network layer. The overall problem is to reduce this routing redundancy of application layer and network layer, and to optimize routing mechanism.

1.6 Thesis Objective and Scope

The main objective of this work is to enable Peer-to-Peer systems in Mobile environments *effectively*. Peer-to-Peer systems are becoming popular very fast. There are many ad-hoc applications in which these systems may need to be deployed such as military bases, navy camps, on-the-road traffic systems etc. Current P2P protocols are application oriented protocols which fail to perform in mobile environments. This thesis suggests a lookup service to come up with this problem. The scope of this work is mainly limited to multihop networks. The effect of node mobility is discussed but an efficient solution is yet to be given.

1.7 Thesis Outline

Rest of the thesis is organized as follows. Chapter 2 and chapter 3 give an overview of P2P protocols and MANET protocols respectively. Chapter 3 also covers current updates on the field of peer-to-peer mobile ad-hoc networks. The RINGS protocol is explained in Chapter 4 with all details. Chapter 5 checks the correctness of RINGS by analysis and comparison. Simulation results are discussed in chapter 6. Chapter 7 gives directions for future work and concludes the thesis.

Chapter 2

P2P Protocols

This chapter gives a brief overview of popular P2P protocols which include Napster, Gnutella, Chord, CAN, and Pastry.

2.1 Napster

Napster [1] has been among the first protocol which introduced P2P concepts in computer networks. Although it follows some of P2P paradigms, it fails to be characterized as a complete P2P network as it relies on centralized servers. in Napster, a central server stores the index of all the files available within the Napster user community. To retrieve a file, a user queries this central server using the desired file's well known name and obtains the IP address of a user machine storing the requested file. The file is then downloaded directly from this user machine. Thus, although Napster uses a peer-to-peer communication model for the actual file transfer, the process of locating a file is still very much centralized. This makes it expensive and easy to fail.

2.2 Gnutella

The most prominent example for flat routing architectures in the area of Peer-to-Peer networks is the network established by the Gnutella protocol [3] . The Gnutella network is made up of nodes distributed throughout the world, which are interconnected by

TCP/IP connections. Within this virtual overlay network the nodes provide the content and perform routing tasks, to make networking possible.

Every node is connected dynamically to a few number of nodes, depending on the bandwidth of the node's network connection. The messages, routed via these connections can be divided into two categories. One type are the query messages, and the second type are the respond messages. The query messages are used to explore the structure of a terminals neighborhood and to search for user demanded content.

The Gnutella network employs a routing concept, known as viral propagation for the query messages. This means that a node searching for content or exploring the network, sends out a query message, i.e. a QUERY() or a PING() message, to all the neighboring nodes it is currently directly connected to via TCP/IP connections in the virtual overlay network.

The second type of messages, which are used in the Gnutella network, are the respond messages, which are used to answer received query messages. These respond message are of no interest to the rest of the network, and they therefore have to be routed only to the querying node. To avoid flooding, respond messages are routed back to the querying terminal on the same way backwards, the original query message traveled to the receiving node.

Beside the application routed-signaling messages, i.e. the respond and query messages, the content a node is querying for, must also be distributed through the virtual overlay network. However, to minimize the load on the existing overlay network and especially of its nodes/routers, the demanded data is transmitted out-band. Out-band in this context means, that with the address provided in the QUERY_HIT message, a direct-only IP routed connection between the querying and the responding node is established. This connection is used to transmit the content directly between both peers.

The major problem of the Gnutella protocol v0.4 [2] is that parts of the virtual overlay network are flooded to a certain extent with ping and query messages, which causes a high signaling load. To reduce this load, a time-to-live value (TTL) is attached to each query message in the Gnutella protocol v0.4. Thus a certain transmission range for these messages is defined, which prevents the network from being flooded by query messages.

2.3 Chord

Chord [9] is a scalable, distributed look-up protocol which efficiently locates the node that stores a particular data item or service. Chord provides support for one operation: given a key, it maps the key onto a node. While Chord maps keys onto nodes, traditional name and location services provide a direct mapping between keys and values. A value can be an address, a document, or an arbitrary data item. Chord implements this functionality by storing each key/value pair at the node to which that key maps. Depending on the application using Chord, that node might be responsible for storing a value associated with the key. Chord uses a variant of consistent hashing to assign keys to Chord nodes. Consistent hashing tends to balance load, since each node receives roughly the same number of keys, and involves relatively little movement of keys when nodes join and leave the system.

Previous work on consistent hashing assumed that nodes were aware of most other nodes in the system, making it impractical to scale to large number of nodes. In contrast, each Chord node needs routing information about only a few other nodes. Because the routing table is distributed, a node resolves the hash function by communicating with a few other nodes. In the steady state, in an N-node system, each node maintains information only about $O(logN)$ other nodes, and resolves all lookups via $O(logN)$ messages to other nodes. Chord maintains its routing information as nodes join and leave the system; with high probability each such event results in no more than $O(log^2N)$ messages.

Our analysis and simulation results show that in fact, performance of *Chord* degrades in mobile ad-hoc networks.

2.4 Content-Addressable Network (CAN)

Content-Addressable Network (CAN) [4], as a distributed infrastructure, provides hash table-like functionality on Internet-like scales. In CAN, the node space is mapped on a d-dimensional thorus. The performance of the scheme depends on the tuning parameter d which is the dimension of the thorus. Each node owns a zone of the thorus. To store a pair(key, value), the key is deterministically mapped, using the hash function, to a point in this virtual space. The pair is then stored at the node that owns the zone within which the point lies. To get the value corresponding to key, the node applies the same hash function to map the key onto the point, and retrieve the value from that point. In case the point is not in the zone owned by the node or immediate neighbors, the request gets routed until it reaches the zone where the point lies. To maintain this structure upon nodes departure, CAN must ensure that the zones they occupied are taken over by the remaining nodes. CAN does that through a Takeover algorithm.

2.5 Pastry

Pastry [5] is a scalable, distributed object location and routing substrate for wide-area Peer-to-Peer applications. *Pastry* performs application-level routing and object location is a potentially very large overlay network of nodes connected via the Internet.

2.6 Summary

Although all these protocols decrease the signaling overhead significantly, the major application area is still a fixed network. These approaches do not aim to match the virtual overlay network to the physical network and thus are not utilizable within MANET scenarios.

Chapter 3

Mobile Ad-hoc Routing Protocols

Since the advent of ad-hoc networks numerous protocols have been developed so far. Such protocols have unique characteristics like high power consumption, low bandwidth, high error rates, and multihop routing. All the ad-hoc routing protocols can be categorized as (See Fig. 3.1):

- Table-driven
- Demand-driven (Source-initiated)

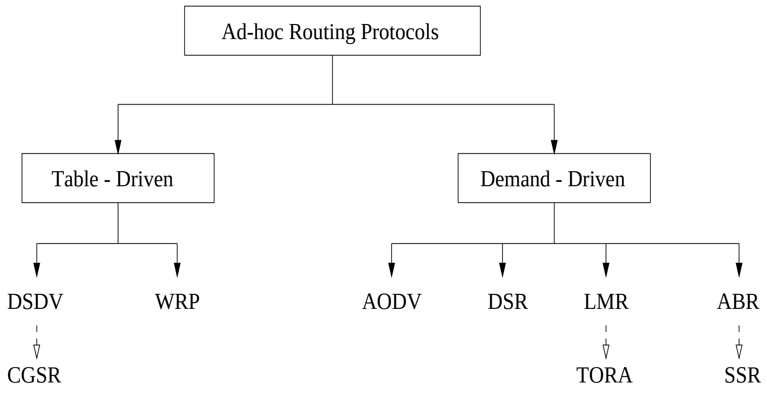

Figure 3.1: Various Ad-hoc Routing Protocols

3.1 Table-driven protocols

Each node in the network has to maintain one or more tables to store routing information in table-driven routing protocols, and they respond to changes in network topology by propagating updates throughout the network in order to maintain a consistent network view. Thus, table-driven routing protocols attempt to maintain consistent, up-to-date routing information from each node to every other node in the network. Important protocols are Destination-Sequenced Distance-Vector Routing Protocol(DSDV) and Wireless Routing Protocol(WRP).

3.1.1 Destination-Sequenced Distance-Vector Routing(DSDV)

The Destination-Sequenced Distance-Vector Routing protocol(DSDV) is a table-driven algorithm which features loop-free routing.

Each mobile node in the network maintains a routing table in which all of the possible destinations within the network and the number of hops to each destination are recorded. Each entry is marked with a sequence number assigned by the destination node. Routing table updates are periodically transmitted throughout the network in order to maintain table consistency.

New route broadcasts contain the address of the destination, the number of hops to reach the destination, the sequence number of the information received regarding the destination, as well as a new sequence number unique to the broadcast. The route labeled with the most recent sequence number is always used. In the event that two updates have the same sequence number, the route with the smaller metric is used in order to optimize the path. Mobiles also keep track of the settling time of routes, or the weighted average time that routes to a destination will fluctuate before the route with the best metric is received. By delaying the broadcast of a routing update by the length of the settling time, mobiles can reduce network traffic and optimize routes by eliminating those broadcasts that would occur if a better route was discovered in the very near future.

Cluster-head Gateway Switch Routing - Instead of a “flat” network, Cluster-head Gateway Switch Routing (CGSR) protocol is a clustered multihop mobile wireless network with several heuristic routing schemes.

3.1.2 Wireless Routing Protocol(WRP)

The Wireless Routing Protocol (WRP) is a table-based protocol with the goal of maintaining routing information among all nodes in the network. Each node in the network is responsible for maintaining four tables:

- Distance table
- Routing table
- Link-cost table
- Message retransmission list (MRL) table

Each entry of the MRL contains the sequence number of the update message, a retransmission counter, an acknowledgment-required flag vector with one entry per neighbor, and a list of updates sent in the update message. The MRL records which updates in an update message need to be retransmitted and which neighbors should acknowledge the retransmission.

Mobiles inform each other of link changes through the use of update messages. An update message is sent only between neighboring nodes and contains a list of updates, as well as a list of responses indicating which mobiles should acknowledge (ACK) the update. Mobiles send update messages after processing updates from neighbors or detecting a change in a link to a neighbor. In the event of the loss of a link between two nodes, the nodes send update messages to their neighbors. The neighbors then modify their distance table entries and check for new possible paths through other nodes. Any new paths are relayed back to the original nodes so that they can update their tables accordingly.

Nodes learn of the existence of their neighbors from the receipt of acknowledgments and other messages. If a node is not sending messages, it must send a *hello* message within a specified time period to ensure connectivity. Otherwise, the lack of messages from the node indicates the failure of that link; this may cause a false alarm. When a mobile receives a *hello* message from a new node, that new node is added to the mobile's routing table, and the mobile sends the new node a copy of its routing table information.

3.2 Demand-driven protocols

This type of routing creates routes only when desired by the source node. So, they are called source-initiated on-demand routing protocols. When a node requires a route to a destination, it initiates a route discovery process within the network. This process is completed once a route is found or all possible routes have been examined. Once a route has been established, it is maintained by a route maintenance procedure until either the destination becomes inaccessible along every path from the source or until the route is no longer desired.

3.2.1 Ad-hoc On-demand Distance Vector (AODV)

The Ad-hoc On-demand Distance Vector (AODV) routing protocol builds on the DSDV algorithm previously described. AODV is an improvement on DSDV because it typically minimizes the number of required broadcasts by creating routes on a demand basis, as opposed to maintaining a complete list of routes as in the DSDV algorithm.

When a source node desires to send a message to some destination node and does not already have a valid route to that destination, it initiates a *path discovery* process to locate the other node. It broadcasts a route request (RREQ) packet to its neighbors, and so on, until either the destination or an intermediate node with a "fresh enough" route to the destination is located. AODV utilizes destination sequence numbers to ensure all routes are loop-free and contain the most recent route information. Each node maintains its own sequence number, as well as a broadcast ID. The broadcast ID is incremented for every RREQ the node initiates, and together with the node's IP address, uniquely identifies and RREQ. Along with its own sequence number and the broadcast ID, the source node includes in the RREQ the most recent sequence number it has for the destination. Intermediate nodes can reply to the RREQ only if they have a route to the destination whose corresponding destination sequence number is greater than or equal to that contained in the RREQ.

During the process of forwarding the RREQ, intermediate nodes record in their route tables the address of the neighbor from which the first copy of the broadcast packet is received, thus establishing a reverse path. If additional copies of the same RREQ are later received, these packets are discarded. Once the RREQ reaches the destination or an intermediate node with a fresh enough route, the destination/intermediate node responds by uni-casting a route reply (RREP) packet back to the neighbor from which it first received the RREQ. As the RREP is routed back along the reverse path, nodes along this path set up forward route entries in their route tables which point to the node from which the RREP came. These forward route entries indicate the active forward route. Associated with each route entry is a route timer which will cause the deletion of the entry if it is not used within the specified lifetime. Because the RREP is forwarded along the path established by RREQ, AODV only supports the use of symmetric links.

Routes are maintained as follows. If a source node moves, it is able to re-initiate the route discovery protocol to find a new route to the destination. If a node along the route moves, its upstream neighbor notices the move and propagates a *link failure notification* message to each of its active upstream neighbors to inform them of the erasure of that part of the route. These nodes in turn propagate the *link failure notification* to their upstream neighbors, and so on until the source node is reached. The source node may then choose to re-initiate route discovery for that destination if a route is still desired.

An additional aspect of the protocol is the use of *hello* messages, periodic local broadcasts by a node to inform each mobile nodes of other nodes in its neighborhood.

3.2.2 Dynamic Source Routing (DSR)

The Dynamic Source Routing (DSR) protocol is an on-demand routing protocol that is based on the concept of source routing. Mobile nodes are required to maintain route caches that contain the source routes of which the mobile is aware. Entries in the route cache are continually updated as new routes are learned.

The protocol consists of two major phases: *route discovery* and *route maintenance.* When a mobile node has a packet to send to some destination, it first consults its route cache to determine whether it already has a route to the destination. If it has an unexpired route to the destination, it will use this route to send the packet. On the other hand, if the node does not have such a route, it initiates route discovery by broadcasting a *route request* packet. This route request contains the address of the destination, along with the source node's address and a unique identification number. Each node receiving the packet checks whether it knows of a route to the destination. If it does not, it adds its own address to the *route record* of the packet and then forwards the packet along its outgoing links. To limit the number of route requests propagated on the outgoing links of a node, a mobile only forwards the route request if the request has not yet been seen by the mobile and if the mobile's address does not already appear in the route record.

A *route reply* is generated when a route request reaches either the destination itself, or an intermediate node which contains in its route cache an unexpired route to the destination. By the time the packet reaches either the destination or such an intermediate node, it contains a route record yielding the sequence of hops taken. If the node generating the route reply is the destination, it places the route record contained in the route request into the route reply. If the responding node is an intermediate node, it will append its cached route to the route record and then generate the route reply. To return the route reply, the responding node must have a route to the initiator. If it has a route to the initiator in its route cache, it may use that route. Otherwise, if symmetric links are supported, the node may reverse the route in the router record. If symmetric links are not supported, the node may initiate its own route discovery and piggyback the route reply on the new route request.

Route maintenance is accomplished through the use of route error packets and acknowledgments. *Route error* packets are generated at a node when the data link layer encounters a fatal transmission problem. When a route error packet received, the hop in error is removed from the node's route cache and all routes containing the hop are truncated at that point. In addition to route error messages, acknowledgments are used to verify the correct operation of the route links. such acknowledgments include passive acknowledgments, where a mobile is able to hear the next hop forwarding the packet along the route.

3.2.3 Temporally Ordered Routing Algorithm (TORA)

The Temporally Ordered Routing Algorithm (TORA) is a highly adaptive loop-free distributed routing algorithm based on the concept of link reversal. TORA is proposed to operate in a highly dynamic mobile networking environment. It is source-initiated and provides multiple routes for any desired source/destination pair. The key design concept of TORA is the localization of control messages to a very small set of nodes near the occurrence of a topological change. To accomplish this, nodes maintain routing information about adjacent (one-hop) nodes.

Main problem with TORA is that TORA assumes that all nodes have synchronized clocks which can be accomplished via an external time source such as the Global Positioning System. TORA's metrics are dependent on the logical time of a link failure.

3.2.4 Associativity Based Routing (ABR)

The Associativity Based Routing (ABR) protocol is free from loops, deadlock, and packet duplicates, and defines a new routing metric for ad-hoc mobile networks. This metric is known as the *degree of association stability.* In ABR, a route is selected based on the degree of association stability of mobile nodes. Each node periodically generates a beacon to signify its existence. When received by neighboring nodes, this beacon causes their associativity tables to be updated. For each beacon received, the associativity tick of the current node with respect to the beaconing node is incremented. Association stability is defined by connection stability of one node with respect to another node over time and space. A high degree of association stability may indicate a low state of node mobility and vice-versa.

3.3 Summary of Ad-hoc Protocols

It can be seen from the above mentioned protocols that both type of protocols (Table-Driven and On-Demand) has their own advantages and disadvantages. The table-driven ad-hoc routing approach is similar to the connectionless approach of forwarding packets, with no regard to when and how frequently such routes are desired. It relies on an underlying routing table update mechanism that involves the constant propagation of routing information. However, This is not the case for on-demand routing protocols. When a node using an on-demand protocol desires a route to a new destination, it will have to wait until such a route can be discovered. On the other hand, because routing information is constantly propagated and maintained in table-driven routing protocols, a route to every other node in the ad-hoc network is always available.

3.4 Progress on P2P Mobile Networks

3.4.1 JXME

A first approach trying to make P2P networking in a mobile environment feasible is JXME [13]. This approach intends to keep the objectives of JXTA [15], and to adapt JXTA such that the requirements of a mobile network can be met. However, this approach regards only cellular networks, and does not address the problem of adapting the virtual overlay network to the physical network. JXME is therefore not feasible for a MANET scenario.

3.4.2 Lawrence

Lawrence [14] proposes an approach to use JADE-LEAP in an ad-hoc environment. The author proposes the removal of some mandatory components and adds a Discovery Agent to support ad-hoc networks. Unfortunately this does not address the question of routing, which we think is one of the most critical question in a MANET scenario.

3.4.3 PROEM

Proem [12] is a platform independent architecture for building diverse mobile applications ranging from ad-hoc meeting support to classical P2P applications like MP3 file sharing and instant messaging. Its messages are based on TCP, UDP or HTTP and employ XML for the representation of the messages. However, it also does not provide any mechanism to adapt its virtual to the physical topology. Thus PROEM may not be feasible in larger MANET scenarios.

3.4.4 ORION

ORION [11] provides full P2P functionalities in a MANET environment. The basis of ORION is AODV [16] and the Simple Multicast and Broadcast Protocol for MANET [17]. ORION provides an application layer routing protocol causing unnecessary overhead.

Hybrid solutions, relying on base stations and further centralized elements, are from our point of view not suitable. Central elements cause higher costs and additionally can in most cases not be administered by the user itself.

3.4.5 MPP - Mobile Peer-to-Peer Protocol

A protocol named *MPP* in [7] emphasizes on communication between application layer and network layer to deploy Peer-to-Peer systems effectively in mobile environments. This protocol establishes an intermediate layer which helps in passing queries from application layer to network layer. Abstract view of this protocol is shown in Fig. 3.2.

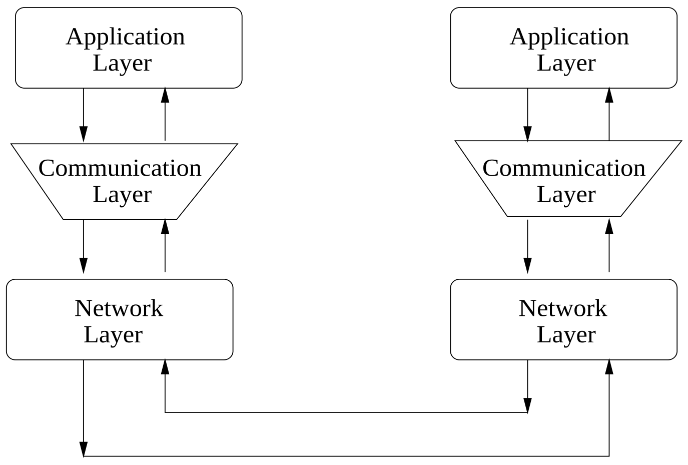

Figure 3.2: Communication between Application layer and Network layer

Although this protocol is an important step towards an efficient P2P applications on Mobile Ad-hoc Networks, it still relies on flooding the network while query lookup. There is a need of protocol which can *control* this network flood. Our work suggests such protocol called RINGS.

Chapter 4

RINGS: Protocol in Details

RINGS provides lookup service which operates at the network layer. It assumes that all lookup queries are forwarded from application layer to network layer to enable RINGS lookup service. RINGS is basically a *proactive* protocol which spreads data index into the network beforehand. Data indices are stored in the network such a way that lookup cost reduces to a constant factor. Thus, it avoids query lookup flooding as it generally happens with current P2P protocols.

4.1 Architecture

RINGS, being a network layer protocol, has to communicate with application layer to facilitate data query and results. We assume that communication is happening between application layer and network layer. [7] has already suggested a protocol which establishes this communication. We also assume that nodes in the network have sufficient amount of memory storage to store routing table and other information.

Whenever a node joins the network, it generates a request packet and sends it down to network layer. Request packet can be either of two types: *Advertisement Packet* or *Lookup Packet*. For the *Advertisement Packet*, node generates index of its own data internally and tags it with the packet. *Lookup Packet* is used while data query processing. Depending upon the packet type, network layer forms either QUERY_PACKET or ADV_PACKET and forwards it to neighbors.

4.2 Data Advertisement

RINGS is basically a proactive protocol. After getting *Advertisement Packet*, node needs to spread index into the network. Further sequence of events can be described as:

1. Network layer forms a packet called ADV_PACKET and forwards it to neighbors. ADV_PACKET header fields include,

    - *source_ID*: *source_ID* is the node which actually has the data. *source_ID* generates index of its data and tags it along with the *ADV_PACKET*.
    - *sender_ID*: Here, *sender_ID* is the identifier of the node which forwards the packet.
    - *seq_number*: *seq_number* is maintained to eliminate receiving duplicate packets.
    - *index_size*: *index_size* contains the size of the index.
    - *route_to_source*: *route_to_source* is a complete route from the current node to source. The functionality of this field is explained later.
    - *hop_count*: The value of *hop_count* is incremented every time an intermediate node is visited.

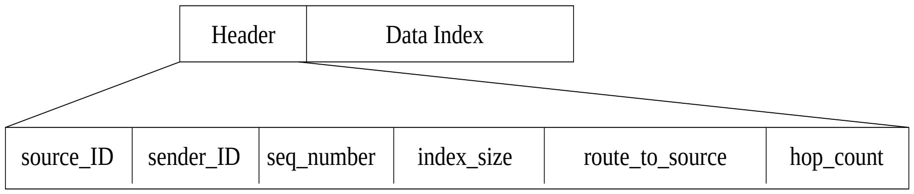

Figure 4.1: ADV_PACKET packet header format

2. RINGS Protocol selects a number called *Index-Hop* "K" at the start of the network. This number remains same for all nodes throughout the network. Nodes, lying on $(n * K)^{th}$-hop circle stores the index and forward the packet further, where $n = 1, 2, 3...$ Other nodes simply forward the packet. At this point of time, we assume that network nodes have sufficient storage space to store remote indices. The

above process forms a set of imaginary circles of nodes with the center indicates the source of the data. See Fig. 4.2

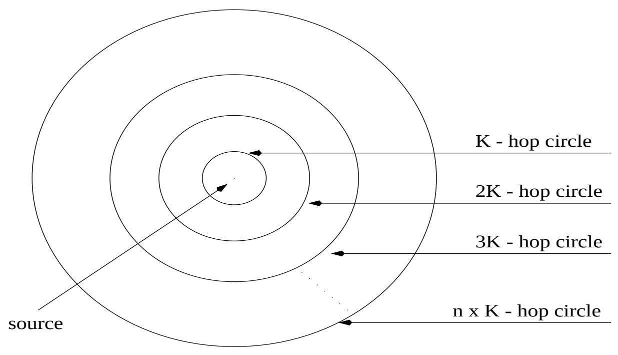

Figure 4.2: A set of imaginary circles for a node

For example, if the value of *Index-Hop* is kept 2, all nodes on the circle of $n * K$ hops will store the index where $n = 1, 2, 3, ...$ In between, to identify unique packet, ADV_PACKET field includes sequence number. Sequence number along with source_ID uniquely identifies the packet.

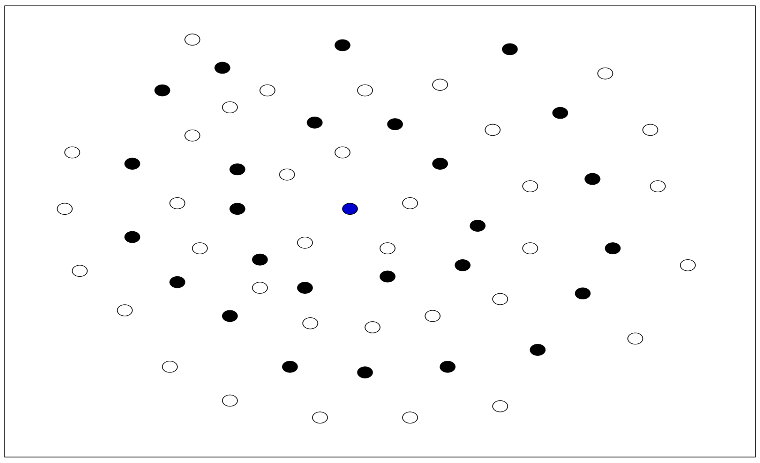

Figure 4.3: An example of 2-hop protocol

3. Above procedure is followed by each node in the network.

4.3 Routing Table

Every node maintains a routing table. The routing table entry contains source_ID, sender_ID, sequence number, TTL(Time-To-Live), hop-count as shown in Fig. 4.4.

source_ID	sender_ID	seq_number	TTL	hop_count

Figure 4.4: Routing table entry

The main purpose of routing table is to avoid receiving duplicate packets. If node receives packet with same *source_ID* and same *sequence number* again within TTL time, node drops the packet. Each time node receives an ADV_PACKET, it updates routing table entry depending on *Advertisement packet* fields. If current entry in routing table contains same *source_ID* but *sequence number* is less than that of received packet, table entry is updated accordingly. If entry is not found for received packet's *source_ID*, new entry is made into routing table. These updates eliminate the possibility of spreading data all-over in the network.

4.4 Query Lookup

Whenever a node wants a particular data or service, it forwards the QUERY_PACKET to its neighbors. In any case, the query gets the results within maximum of $K/2$ hops. Thus, query does not flood throughout the network. As for example, if the value of K is kept 4, it means $n * 4$ hops away nodes has stored the index. Thus the upper limit on the *lookup cost* for any node in the network reduces to 2 hops.

4.5 Route Maintenance

ADV_PACKET maintains route to the source node which contains actual data or services. Along with forwarding packet to nodes, a reverse path is also maintained. A node which stores the index, keeps this route to source as well. Whenever a node sends query, node which contains index, responds with query result along with route to source. It can be shown that this route remains to be *optimal*. As for example shown in Fig. 4.5, suppose R makes query request. In reply, P responds with complete route to source with other information. Now R can communicate with node S using the route.

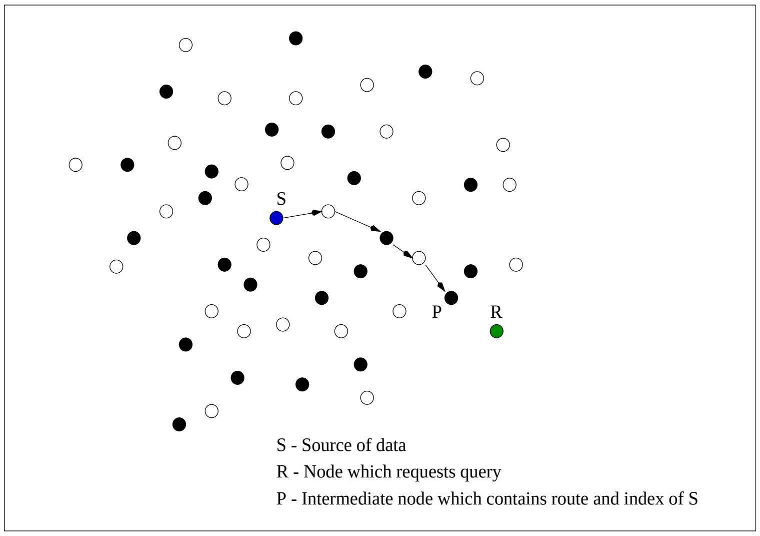
S
P
R
S - Source of data
R - Node which requests query
P - Intermediate node which contains route and index of S

Figure 4.5: An example of route maintenance

4.6 Mobility

Until now, we have assumed that all nodes in the network are static. If nodes move frequently in the network, it is difficult to trace them. The advantage of RINGS protocol is, mobility have none or little impact on performance of the protocol. It has been shown that, in order to achieve maximum *throughput* and *connectivity*, number of neighbors per node in mobile network can be taken as $log(N)$, where N is the number of nodes in the network. Given that, if the value of *Index-Hop* K is kept to 2, a node can get the query results from maximum of $log(N)$ as average number of neighbors is taken as $log(N)$. Out of that, if as much as $log(N) - 1$ nodes leave from or move in the network, a node can get results from at least one node. Extending this equation, we can conclude that, in general, a node gets at least one node which can satisfy its query even if L number of nodes from nearest circle leave the network where L is,

$$L = (log(N) - 1)^{K/2} \tag{4.1}$$

where $K = 2, 4, 6, ...$

4.7 Optimization

4.7.1 Data Updates

RINGS reduces advertisement broadcast by sending data updates to only those nodes which contain the index. To achieve this, *reverse path* from node, which has stored the index, to source is maintained.

4.7.2 Node Joining

RINGS provides a mechanism which allows newly joined node to stabilize in the network. If a new node joins the network, RINGS assigns it the indices of some nodes. This assignment ensures that indices are evenly distributed in the network. Whenever a new node joins the network, it searches for neighbors and gets the neighbor list. After getting the list, node sends this list to each of its neighbors. Now every neighbor node will check how many of the other neighbor nodes are neighbor of itself. For example, a neighbor node i checks against another neighbor node j in Fig. 4.6. If j is neighbor of i also, it means that there are some routes at node j which start from source node to node j via node i. It means, new node falls on the same circle of j for a particular source node. So, RINGS allocates all such routes and indices from j to newly joined node. Fig. 4.7 explains this process through pseudo-code.

4.7.3 Node Deletion

RINGS implements a process named *Index Reallocation* to facilitate node deletion event. *Index Reallocation* process reallocates index into appropriate node if any node has moved away from the network.

As indicated in previous section, RINGS maintains route from the node which has stored the index to the source node. If any node moves away from the network, the

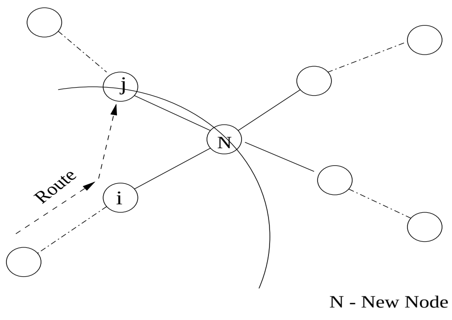

Figure 4.6: Node Join

```
for every neighbor i of new node N {
   for every neighbor j of new node N {
     if ( i != j && j is neighbor of i )
        allocates all indices and routes which are stored in j and
        which has next hop(in the direction of source node) as i, to N;
   }
}
```

Figure 4.7: Pseudo-code for Node joining

route to source is broken. by means of *Index Reallocation*, RINGS tries to re-establish this route and reallocates indices to appropriate nodes. Whenever a node goes off, it informs neighbor nodes about its in-availability. The neighbors search for an alternative route which had deleted node as an intermediate hop previously.

For example, 4.8 shows one of the routes in the network with some source node S. Here, node A itself can be source node or can be an intermediate node. Node B is an intermediate node which contains source node's index. Now, suppose node X goes off. Node A gets information about it and starts a search for an alternative route for node Y. Fig. 4.9 indicates such an alternative route. Indices has to be now reallocated using this route. After getting the route, node A initializes update packet with *Index-hop* value is 0. Update packet traverses through the newly established route and moves indices, stored at node B, to previous node which falls in the circle of *index-hop* K. Pseudo-code for this process is shown in 4.10.

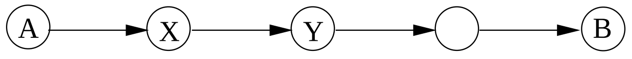

Figure 4.8: After Index Advertisement

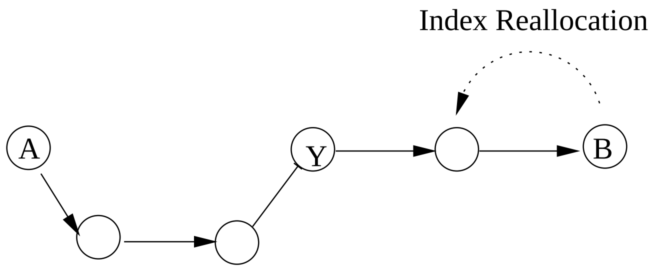

Figure 4.9: Node Deletion: Index Reallocation

4.8 Summary

This chapter explains RINGS protocol in detail. The main aim of RINGS protocol is to decrease query lookup cost by distributing data indices evenly throughout the network.

```
void update{
    for every neighbor i of node X { // X went off from the network
        for every neighbor j (j != i) of node X {
          j.route = route_discovery(j);
          bool_index_reallocation = 0, count = 0;
          for every node in j.route {
             increment count by 1;
             if ( bool_index_reallocation == 1 )
               if not ( node have the source index )
                  update index_reallocation_route;
                  continue;
               else
                  follow the index_reallocation_route and
                  move index to index_reallocation_route.source;
                  delete the stored index;
             else
               if ( count == KVALUE )  // KVALUE is index-hop
                  if not ( node have the source index )
                     bool_index_reallocation = 1;
                     index_reallocation_route.source = node.id;
                     continue;
                  count = 0;
          } //end of inner-most ‘for’ loop
        }
    } //end of outer-most ‘for’ loop
}  //end of update function
```

Figure 4.10: Pseudo-code for Node Deletion

Chapter 5

Analysis and Comparison

5.1 Query Lookup Cost

As mentioned earlier, current P2P protocols are application layer protocols. If these protocols are put in mobile ad-hoc environments, the overhead of network layer should also be counted in overall routing overhead. Now from network layer perspective, neighbors at the application layer may not be neighbors at network layer. In worse case, they could be at the different edges of the network. Application layer has different routes than that of network layer. So, the effective lookup cost between source and destination is the sum of application layer lookup cost and network layer routing cost, which indeed is larger. For example, lookup cost for *Chord* protocol is $log(N)$ at the application layer(please refer [9] for more details) but the overall lookup cost is larger.

In order to count network layer hops, we need to find out the average distance between any two nodes at the network layer. According to [10], the average distance d between any two nodes at the network layer can be taken as,

$$d = 2\alpha \left(\sqrt{N/\pi} \right)$$

where α is a constant and N is total number of nodes in the network.

So, effective lookup cost L_{chord} for *Chord* protocol is,

$$L_{chord} = d * log(N) \tag{5.1}$$

In RINGS, any node has to contact nearest *Index-Hop* circle to get the data index, which is maximum of $K/2$ hops away. Taking average number of neighbors per node as $log(N)$, we can say that node forwards query to $log(N)$ nodes. Thus query lookup cost at first hop is,

$$L_{RINGS} = log(N)$$

In turn, each of $log(N)$ nodes forwards query to its neighbors. Continuing this way, query will be forwarded to L_{RINGS} number of nodes before reaching to nearest *Index-Hop* circle, where L_{RINGS} is,

$$L_{RINGS} = log(N) * (1 + (log(N) - 1) + (log(N) - 1)^2 + \\ ... + (log(N) - 1)^{K/2-1}) \tag{5.2}$$

Comparison of both the costs is shown graphically in Fig. 6.2 in next chapter.

5.2 Comparison with Current P2P protocols

Major functionalities of P2P protocols can be summarized in the table below:

Scheme	Query	Memory	Messages
Napster	1	N	1
Gnutella	$logN$	1	N
Chord	$logN$	$logN$	$logN$
CAN	N^2	*const*	N^2
Pastry	$logN$	$logN$	$logN$

Table 5.1: P2P Protocols

Out of these protocols, Napster does not fit into MANET environment. Gnutella networks have a query time of $logN$ whereas the network load is extremely high since a query must travel all the nodes in the network to be answered. RINGS limits this flooding by answering query within $K/2$ hops. CAN performance can be worse in MANET as the query lookup cost and messages increase exponentially. Chord and Pastry reduce lookup cost and messages significantly but works at application layer rather than at network layer which incurs multi-layer overhead.

The main objective of RINGS protocol is to reduce query lookup cost. This optimization comes at the cost of higher memory space requirement at the node. For example, if *index-hop* value is kept 4 in RINGS, every node has to store approximately $1/4^{th}$ of the total indices of the network. But as indicated previously, current mobile node's memory capabilities are increasing dramatically, it is acceptable to assume enough memory space at the network node.

Chapter 6

Simulation and Results

6.1 Simulation Setup

RINGS protocol has been simulated independently of any existing network simulators like NS2 etc. Various scenarios has been taken to check the efficiency of RINGS in terms of, number of nodes store indices for a node in the network, total remote indices per node etc. Nodes are distributed in the network randomly and RINGS protocol is performed on them. Each node in the network was assigned index value from random distribution. All these experiments were run for different values of *Index-Hop* K starting from $K = 1$ to $K = 5$. The results from initial simulation indicate that performance can be improved by using RINGS.

6.2 Results

As indicated in RINGS, each node stores indices of some other nodes. We want to measure how the value of these average remote indices per node varies over the value of *Index-Hop*. Fig. 6.1 shows different graphs for the different value of *Index-Hop*. In turn, each graph shows the relationship between average remote indices per node and number of nodes in the network. It can be observed that the value $K = 4$ balances between the *lookup cost* and *per node storage requirement.*

Fig. 6.2 graphically shows the comparison between eq. (5.1) and (5.2). It indicates

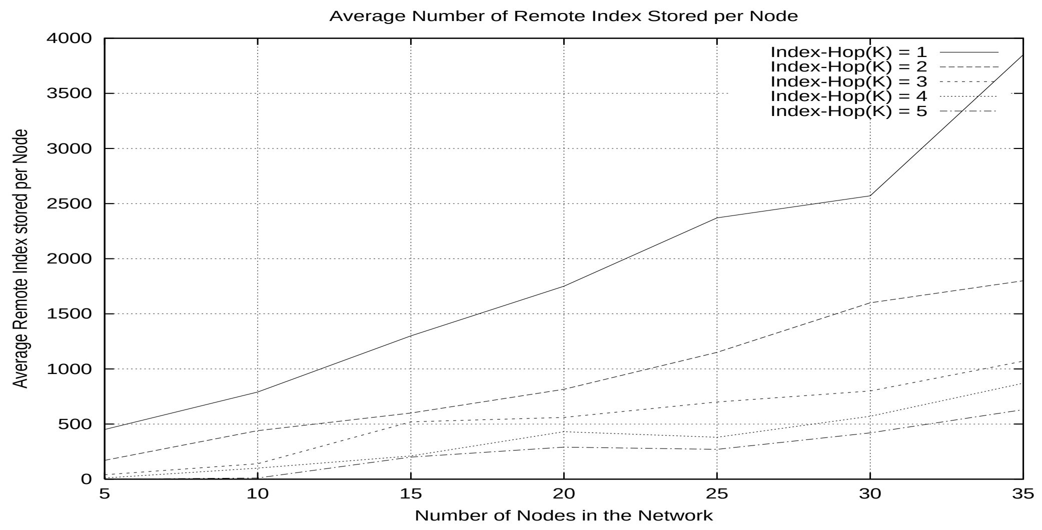

Figure 6.1: Average number of remote index stored per node

that *lookup cost*(number of hops) decreases significantly in RINGS if the value of *Index-Hop* is kept 4 while *Chord* performs better than *RINGS with Index-Hop = 6*, if number of network nodes is kept low. If number of nodes increases, performance of RINGS increases as compare to *Chord*.

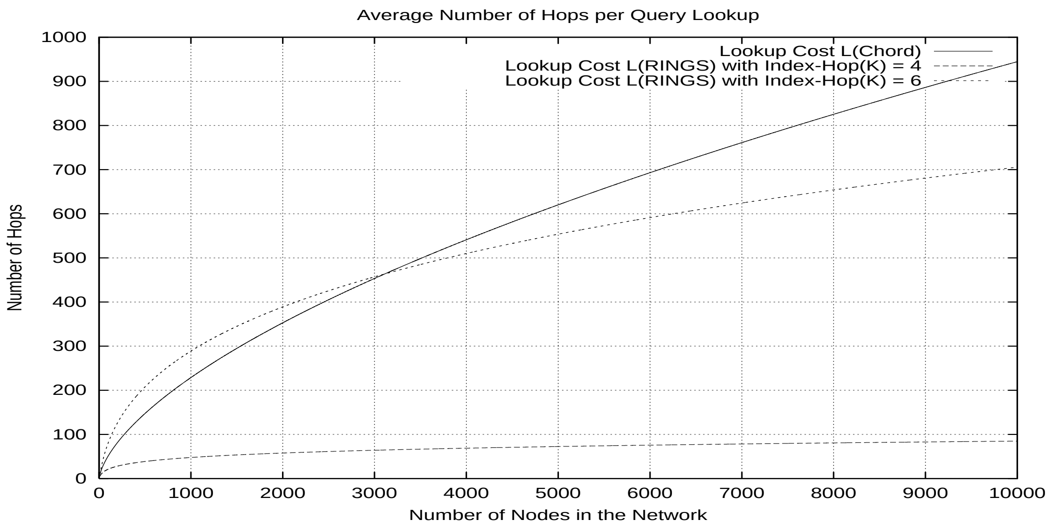

Figure 6.2: Average number of hops per query lookup

Chapter 7

Directions for Future Work and Conclusion

7.1 Possible Applications

Besides current P2P applications exist for sharing media files, there are other possible applications and scenarios for peer-to-peer mobile ad-hoc networks. For example:

- Tele-conferences and Workshops : A system of short life-span can be deployed for conferences and workshops with large number of participants, in which lecturers and participants can share lecture notes, Frequently-Asked-Questions notes, etc.

- Database on The *Road* : These kind of applications can be plugged into moving vehicles in urban areas. Vehicles can share important information like traffic data, route availability, locations of important places etc.

- Applications for Battle Fields : Military and Navy camps can deploy such kind of applications because of their unique characteristics like, keep on moving, should be easily deployed, important data should be reached to every systems and should be updated constantly.

7.2 Directions for Future Work

The suggested protocol specifically focuses on *lookup cost optimization*, which at the same time, incurs more storage space requirement. This requirement pays extra overhead to the node. There can be number of ways to improve this routing protocol to handle such requirements. The important thing is, the exact problem is clearly identified. So, further work can be carried out to solve it in different ways.

Furthermore, *effective mobility handling* is a big task. We have discussed various impacts of mobility like node joining, node leaving etc. but not thoroughly checked for the efficiency.

7.3 Conclusion

Increasing popularity of P2P systems and Mobile Ad-hoc networks gives attention to combine both systems and thus having a community which can share their data in mobile environments. Current P2P protocols fail to perform well with this combination. One of the reasons for that is, it results in multi-layer routing redundancy. So, we suggest a simple but effective protocol named RINGS, which brings down routing and query lookup mechanisms from application layer to network layer and optimizes network layer query routing. RINGS reduces query lookup cost by evenly distributing data indices throughout the network. Our simulation results show the improvement from the suggested protocol.

Bibliography

[1] *Napster Protocol Specification*, `http://opennap.sourceforge.net/napster.txt`.

[2] *Gnutella Protocol Specification*, 0.4 edition, 2002. `http://www.clip2.com/GnutellaProtocol104.pdf`.

[3] R. Manfredi. T. Klingberg. *Gnutella 0.6 RFC* June 2002. `http://rfcgnutella.sourceforge.net/draft.txt`

[4] sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Schenker. *A scalable content-addressable network.* In Proceedings of the 2001 conference on applications, technologies, architectures, and protocols for comuputer communications, pages 161-172. ACM Press, 2001.

[5] Antony Rowstron and Peter Druschel. *Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems.* Lecture Notes in Computer Science, 2218:329+, 2001.

[6] E. Royer and C. Toh. *A review of current routing protocols for ad-hoc mobile wireless networks.* 1999.

[7] Rudiger Schollmeier, Ingo Gruber, Michael Finkenzeller. *Protocol for Peer-to-Peer Networking in Mobile Environments.* In Proceedings of IEEE International Conference on Computer Communications and Networks (ICCCN'03), October 20-22, 2003, Dallas, USA.

[8] Rudiger Schollmeier, Ingo Gruber, Michael Finkenzeller. *Routing in Mobile Ad Hoc and Peer-to-Peer Networks, A Comparison.* In Networking 2002, International Workshop on Peer-to-Peer Computing, May 19-24, 2002, Pisa, Italy.

[9] Ion Stoica, Robert Morris, David Karger, M. Francs Kaashoek, and Hari Balakrishnan. *Chord: A scalable peer-to-peer lookup service for internet applications.* In Proceedings of the 2001 conference on applications, technologies, architectures, and protocols for computer communications, pages 149-160. ACM Press, 2001.

[10] Sverrir Olafsson. *Scalability, Capacity and Local Connectivity in Ad Hoc Networks.* Mobile and Wireless Communications: Key Technologies and Future Applications(P. Smyth (Ed.)), IEE-BT Exact Communication Technology.

[11] A. Klemm, C. Lindemann, O. P. Waldhorst. *A Special-Purpose Peer-to-Peer File Sharing System for Mobile Ad Hoc Networks.* Proceedings of the Workshop on Mobile Ad Hoc Networking and Computing(MADNET 2003). Sophia-Antipolis, France. 2003.

[12] G. Kortuem, J. Schneider. *An Application Platform for Mobile Ad-hoc Networks.* Proceedings of the Workshop on Application Models and Programming Tools for Ubiquitous Computing (UBICOMP 2001). Atlanta, Georgia. 2001.

[13] A. Arora. *JXTA for J2ME^{TM} - Extending the Reach of Wireless With JXTA Technology.* White Paper. 2002.

[14] J. Lawrence. *LEAP into Ad-Hoc Networks.* Proceedings of the Workshop on Ubiquitous Agents on Embedded, Wearable, and Mobile Devices. Bologna, Italy. 2002.

[15] M. Duigou. *JXTA v2.0 Protocols Specification.* IETF Internet Draft. 2003.

[16] C. Perkins, E. Royer, S. Das. *Ad Hoc On-Demand Distance Vector(AODV) Routing.* IETF Internet Draft. 2002.

[17] J. Jetcheva, Y. Hu, D. Maltz, D. Johnson. *A Simple Protocol for Multicast and Broadcast in Mobile Ad Hoc Networks.* IETF Internet Draft. 2001.