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

Kalpesh Patel, Sridhar Iyer, Krishna Paul

Kanwal Rekhi School of Information Technology, IIT Mumbai, INDIA
{kalpesh,sri,krishna}@it.iitb.ac.in

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. 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, it becomes necessary to think of P2P systems that can efficiently work in mobile environments.
Putting current P2P applications into mobile environments, results in multiple layer flooding because these applications will maintain different peer routes than that of by network layer. Also, routes at the application layer may not be necessarily optimal at the network layer. Here, we address this problem and propose a novel, controlled-flooding based protocol named *RINGS* which works at network layer and helps P2P systems to work in mobile environment efficiently. *RINGS* reduces query lookup cost by evenly distributing data indices throughout the network, thus reducing network layer routing overhead.

1 Motivation

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*[3] 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. 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 an efficient network layer routing protocol. If application layer tasks can be assigned to network layer, routing redundancy can be avoided. Thus, we propose 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* follows *controlled-flooding* approach at the network layer, so query does not flood the whole network.

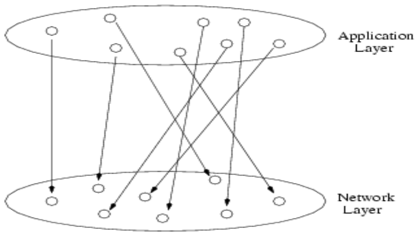

Fig. 1. Positions of application layer nodes from network layer perspective

Rest of the paper is organized as follows. Section 2 gives an overview of current P2P protocols. Section 3 contains the detailed *RINGS* protocol. Analysis and comparisons are made in section 4. Section 5 shows simulation results. Section 6 concludes the work.

2 Related Work

A handful of P2P protocols and architectures have been proposed so far. Nearly all of them fail to match with mobile environments.

Napster[1] has been among the first protocols, 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. While *Gnutella 0.4*[2] follows simple broadcast based technique to communicate with peer node in the network. Gnutella, if deployed in ad-hoc networks, results in *multiple layer flooding* as ad-hoc network layer protocols also are based on flooding. *Gnutella 0.6*, an enhanced version is available which decreases the flooding overhead significantly, but still overall flooding is high.

Chord[3] is a scalable, distributed look-up protocol, which efficiently locates the node that stores a particular data item or service. Our analysis and simulation results show that in fact, performance of *Chord* degrades in mobile ad-hoc networks. *CAN*[4] essentially offers the same functionality as a hash table. i.e. it maps 'keys' to 'values'. Functionally, it is very much similar to *Chord*.

A protocol named *MPP* in [5] emphasizes on communication between application layer and network layer to deploy Peer-to-Peer systems in mobile environments. The problem is, *MPP* broadcasts the query throughout the network and gets the result. There is a need of network layer protocol, which can reduce this flooding.

3 Protocol in Detail

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.

3.1 Architecture

RINGS, being a network layer protocol, has to communicate with application layer to facilitate data query and results. So, we assume that communication is happening between application layer and network layer. [5] has already suggested a protocol, which establishes this communication.

Whenever a node joins the network, it originates 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.

3.2 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. The fields of ADV_PACKET include *source_ID*, *sender_ID*, *seq_number*, *index_size*, *route_to_source*, and *hop_count*. Here, *sender_ID* is the identifier of the node, which forwards the packet whereas *route_to_source* is a complete route from the current node to source. To identify unique packet, ADV_PACKET includes sequence number. Sequence number along with source_ID uniquely identifies the packet.
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. 2

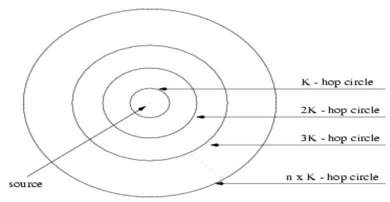

Fig. 2. A set of imaginary circles for a node

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 $n = 1, 2, 3, ...$
3. Above procedure is followed by each node in the network.

3.3 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.

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 the *Advertisement packet* fields.

3.4 Query Lookup

Whenever a node wants a particular data, 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.

3.5 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 inside the nearest circle which can satisfy its query, even if L number of nodes from nearest circle move away from the network where L is,

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

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

4 Analysis and Comparison

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 [3] 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 [6], 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,

$$L_{chord} = d * log(N) \tag{2}$$

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. 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,

$$L_{RINGS} = log(N) * (1 + (log(N) - 1) + (log(N) - 1)^2 +$$

$$... + (log(N) - 1)^{K/2-1}) \tag{3}$$

Comparison of both costs is shown graphically in Fig. 3 in next section.

5 Simulation Results

5.1 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.

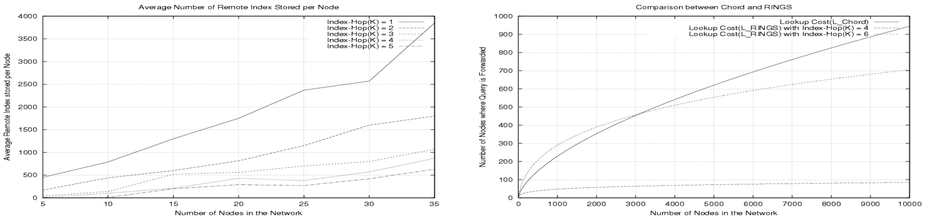

Fig. 3. Simulation Results

5.2 Results

As indicated earlier, 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*. First graph of Fig. 3 shows different measures of average remote indices per node for the different value of *Index-Hop*. It can be observed that the value $K = 4$ balances between the *lookup cost* and *per node storage requirement*. Second graph of Fig. 3 shows the comparison between Eq. (2) and (3). It indicates that *lookup cost* 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*.

6 Conclusion

We observed that current P2P protocols fail to perform well in mobile ad-hoc networks. So, we suggest a simple but effective protocol *RINGS*, which optimizes network layer query routing. *RINGS* reduces query lookup cost by evenly distributing data indices throughout the network. Our initial simulation results show the improvement from the suggested protocol.

References

1. Napster Protocol Specification. June 2000.
2. Gnutella protocol specification v0.4. 2002.
3. Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for Internet applications. In Proceedings of ACM SIGCOMM, pages 149-160. ACM Press, 2001.
4. Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker. A scalable content addressable network. In Proceedings of ACM SIGCOMM 2001.
5. Florian Niethammer Rudiger Schollmeier, Ingo Gruber. Protocol for peer-to- peer networking in mobile environments. In Proceedings of IEEE International Conference on Computer Communications and Networks (ICCCN'03), October 2003.
6. 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.