A Scalable lookup Service for P2P File Sharing in MANET

Mehdi Asadi  
Dept. of Computer Engineering,  
Islamic Azad University, Khamneh  
Branch, Iran  
Asadi_128@yahoo.com

Abolfazl Toroghi Haghighat  
Dept. of Computer Engineering,  
Islamic Azad University, Qazvin  
Branch, Iran  
haghighat@qazviniau.ac.ir

Bager Zarei  
Dept. of Computer Engineering  
Islamic Azad University, Shabestar  
Branch, Iran  
Zarei_bager@yahoo.com

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

Categories and Subject Descriptors

General Terms
Design, Performance.

Keywords
P2P File Sharing, Self-Organizing Networks, MANET environments, Multi-hop routing.

1. INTRODUCTION
P2P system aims to share information among a large number of users without assistance of explicit servers. It has attracted more and more researchers in recent years, motivated by the popularity of file-sharing applications over internet, such as Napster [1] and Gnutella [2]. There are many research issues on P2P networks, among which the searching algorithm is the central topic. Napster uses a central server to maintain index of all information so that every peer should contact the server to lookup the information. Gnutella works in a pure P2P way where every peer broadcasts its query to all its neighbors. Peers form an overlay topology which might be far different from the underlying physical network topology. But this kind of query ‘flood’ takes too much network bandwidth so that scalability issue arises. Some highly structured P2P lookup algorithms have been recently proposed, such as Chord, Pastry, Tapestry and CAN. They all employ a similar technique named Distributed Hashing Table (DHT ), in which every file and peer is assigned a unique key by a hash function. The keys, along with the network address of the peer storing the corresponding files, are evenly distributed among all participating peers. Each peer maintains a routing table and queries are only directed to those peers in the routing table.

For Chord [3], Pastry and Tapestry, the routing table size is $O(d)$ and searching path length is $\log(N)$, where $n$ is the total number of files and peers. For CAN, peers have routing table size of $O(d)$ and searching path length of $O(n^{1/d})$. The fast and accurate searching ability of DHT-based algorithms is very attractive and has been applied to different kinds of applications in wired networks. MANET is another popular research area due to the rapid progress of mobile wireless communication.

MANET is characterized as an infrastructure-less mobile wireless network, in which two mobile nodes communicate with each other through intermediate nodes. Since there is no explicit server, every mobile node should work autonomously. One of the main research issues in MANET is network routing. In proactive (or table-driven) routing protocols, such as DSDV, every mobile node tries to maintain a routing table involving the complete information of network topology. But this needs a lot of computation efforts and communication bandwidth to maintain the accurate routing tables. On the other hand, the reactive (or on demand) routing protocols, such as DSR and AODV, try to find a route to the destination only when it is necessary. The route request is broadcasted throughout the whole network, and the route response is returned when a mobile node knows the route to the requested destination or itself is the destination. We notice that there are some similar features for both P2P file-sharing networks and MANET. If we also call a mobile node in MANET a mobile peer, (a) there is no peer in either P2P network or MANET that acts explicitly as a central server, and every peer should collaborate with other peers in order to make the whole system work. (b) The major problem in P2P and MANET is how to find the requested files or route efficiently. (c) The topology of P2P and MANET is changing frequently because of peer on-off or mobility. (d) In both systems, flooding or broadcasting is employed to some extent in order to exchange files or routing information among different peers, which raises the scalability problem.

There are also some differences between P2P and MANET. For example, (a) P2P refers to the application layer in the protocol stack, while MANET focuses on the network and lower layers. (b) The peers in MANET are mobile and constrained by limited energy, bandwidth and computation power, which is not a big concern in P2P file-sharing systems over internet. (c) For the execution of broadcast, a P2P overlay is a single cast network which only generates virtual broadcast consisting of a number of single cast messages. In contrast, MANET always performs a
physical broadcast. The above similarities and differences between P2P and MANET motivate an interesting and challenging research topic of enabling efficient P2P file-sharing over MANET. The major problem is how to quickly find the requested file in spite of the mobility and the scarcity of power and bandwidth in the underlying MANET. This paper proposes approach combining existing P2P searching protocols and MANET routing protocols. The performance of this approach are evaluated and compared in terms of routing complexity, scalability, maintenance complexity.

The paper is organized as follows. Section 2 briefly reviews progress on P2P mobile networks. Section 3, 4 introduces the proposed approach in details, Simulation and Results. The conclusion is drawn in section 5.

2. Progress on P2P Mobile Networks

We describe some approaches that used in P2P Mobile networks.

2.1 JXME

A first approach trying to make P2P networking in a mobile environment feasible is JXME. This approach intends to keep the objectives of JXTA [4] 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.

2.2 LAWRENCE

LAWRENCE proposes an approach to use the JADE-LEAP in 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 this is one of the most critical questions in a MANET scenario.

2.3 PROEM

Proem [5] 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, HTTP and employ XML for the representation of the message. However, it also does not provide any mechanism to adapt its virtual to the physical topology. Thus PROEM may not be feasible in large MANET scenario.

2.4 ORION

ORION [6] provides full P2P functionalities in a MANET environment. The basis of ORION is AODV and the Simple Multicast and Broadcast protocol for MANET. ORION provides an application layer routing protocol causing unnecessary overhead. Hybrid solutions, relying on base stations and centralized elements that are not suitable. Central elements cause higher costs.

2.5 MPP

A protocol named MPP in [7] emphasized on communication between application layer and network layer to deploy P2P systems effectively in mobile environments.

This protocol establishes an intermediate layer which helps in passing queries from application layer to network layer. Although this protocol is an important step towards an efficient P2P applications on MANET, it still relies on flooding the network while query lookup. There is a need of method which can control this network flood.

3. Suggested Method

This method provides lookup service which operates at the network layer. It assumes that all lookup queries are forwarded from application layer to network layer to enables our method lookup service. This method is a proactive protocol which spreads files index into the network beforehand. Files 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 happens with current P2P protocols.

Method being at network layer protocol has to communicate with application layer to facilitate file query and results. We assume that communication is happening between application layer and network layer. [8] has already suggested a protocol which establishes this communication. We 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 files internally and tags it with the packet. Lookup Packet is used while files query processing. Depending upon the packet type, network layer forms either Query-Packet or ADV-Packet and forwards it to neighbors. After getting ADV-Packet, nodes need to spread index into the network and then Sequence of events can be described:

- Network layer forms ADV-Packet and forwards it to neighbors. ADV-Packet header include:
  - Source-ID: The node which actually has the data or files that generates index of its files and tags it along with the ADV-Packet.
  - Sender-ID: The node which forwards the packet.
  - Index-Size: Contains the size of the index.
  - Seq-Num: Seq-Num is maintained to eliminate receiving duplicate packets.
  - Hop-Count: This value is incremented every time an intermediate node is visited.
  - Route-to-Source: This contains a complete route from the current node to source.

- This method 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 \((s+K)^{th}\) -Hop circle

---

1 Mobile Peer-to-Peer Protocol
stores the index and forward the packet further, where \( n=1, 2, 3 \), \( \ldots \). Other nodes simply forward the packet. We assume that network nodes have sufficient storage memory to store remote indices. The above process forms a set of virtual circle of nodes with the center indicate the source of the files.

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 \( n=1, 2, 3, \ldots \). In between, to identify unique packet, ADV-Packet fields includes sequence number.

Every node maintains a routing table. The routing table entry as shown in Figure 4:

![Routing Table Entry](image)

The main purpose of routing table is to avoid receiving duplicated packet. If node receives packet with same Source-ID and same Seq-Num again within TTL\(^2\) time, node drops the packet. Each time node receives an ADV-Packet, it updates routing table entry depending on ADV-Packet fields. If current entry in routing table contains same Source-ID but Seq-Nm 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 files and data all-over in the network. Whenever a nod wants a file or service, it forwards Query-Packet to its neighbors. The query gets the results within maximum \( \frac{K}{2} \) hops and query does not flood throughout the network. As for example if \( K = 4 \), it means \( n*4 \) hops away node has stored the index. Thus the upper limit on the lookup cost for any node in the network reduces to 2 hops.

ADV-Packet maintains route to the source node which contain actual files or services. Along with forwarding packet ot node, a reverse path is also maintained. A node which stores the index, responds with query result along with route to source. It can be shown that this route remains to be optimal. As for example suppose Q makes query request. In respond, R reply with complete route to source with other information. Then Q can communicate with node S using the route. Because in MANET nodes move frequently in network, it is difficult to trace them. With our method mobility have little impact on performance.

We assume that number of neighbors for each node in mobile network can be taken as \( \log(N) \), where \( N \) is the number of nodes in 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. In general, a node gets at least one node which can satisfy its query even if \( M \) number of nodes from nearest circle leaves the network where \( M \) is:

\[
M = (\log(N) - 1)^{\frac{K}{2}}, \quad \text{Where } K = 2, 4, 6, \ldots
\]

Our method reduces advertisement broadcast by sending files or data updates to only those nodes which contain the index.

### 3.1 Node joining

If a new node joins the network or move from one circle to another circle, method 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. Figure 6.1 shows node joining and figure 6.2 shows pseudo code for node joining.

![Figure 6.1: Node Joining](image)

For every neighbor $i$ of new node $M$ {
    For every neighbor $j$ of new node $M$ {
        If ($i \neq j$ \&\& $j$ 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 $M$;
    }
}

![Figure 6.2: Node joining Pseudo-code](image)

### 3.2 Node Deletion

If any node moves away from the network, the route to source is broken. By means of Index Reallocation, our approach 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 figure 7.1 shows one of the route in the network with same source node $S$. Here node $A$ itself can be source node or can be intermediate node. Node $Z$ is an intermediate node which contains source node’s index. Then suppose node $B$ goes off. Node $A$ gets information about it and starts a search for an alternative route for node $C$.Figure 7.2 indicates such an alternative route. Indices have 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 trough the newly established route and moves indices, stored at node $Z$ and to previous node which fall in the circle of Index-Hop $K$. Pseudo-code for this process is shown in figure 7.3.

![Figure 7.1: After Index Advertisement](image)

```plaintext
Void update {
    For every neighbor $i$ of node $B$ {
        For every neighbor $j$ of node $B$ {
            $j$.route = route_discovery();
            Bool_Index_Reallocation = 0;
            Count = 0;
            For every node in $j$.route {
                Count ++;
                If (Bool_Index_Reallocation == 1)
                    If (! (node have the source index )) {
                        Update_Index_Reallocation_Route ;
                        continue ;
                    } // end of if
                Else {
                    Follow the Index_Reallocation_Route and Move index to Index_Reallocation_Route.
                    Source Delete the stored index ;
                } // end of else
            } // end of inner_if
            Count = 0;
        } // end of inner_for
    } // end of middle_for
    } // end of outer_for
}
```

B : Node which goes off the network
K_Value : Index-Hop

![Figure 7.2: Node Deletion and Index Reallocation](image)

### 3.3 Query Lookup Cost

Current P2P protocols are application layer protocols. If these protocols used in mobile ad-hoc environments, the overhead of
network layer also be added in overall routing overhead. From network layer perspective, neighbor’s node at application layer may not be neighbor’s nodes in network layer. Application layer has different routes than that of network layer. So, lookup complexity or cost between source node and destination node is sum of application layer lookup cost and network layer routing cost that which really is larger. For example application layer cost in Chord protocol is \( \log(N) [3] \) but the total lookup complexity or cost is larger. In order to count network layer hops, we need to find out the average distance between any two nodes at the application layer [9]. Average distance \( d \) can be taken as:

\[
d = 2\alpha \cdot \frac{\sqrt{N}}{\pi}
\]

Where \( N \) is total number of nodes in network and \( \alpha \) is a constant that equal or larger than 2. Total lookup cost for Chord protocol is,

\[
\text{Length}_\text{Chord} = d \cdot \log(N)
\]

In our presentation any node has to contact nearest Index-Hop circle to get the files index, which is maximum of \( \frac{k}{2} \) hop away.

For each node we take average number of neighbors as \( \log(N) \), that we can say each node forwards query to its \( \log(N) \) neighbor nodes. Thus query lookup cost for first node is \( \log(N) \). In turn, each node of \( \log(N) \) nodes forwards query to its neighbors. If we continue this way query will be forwarded to number of nodes before reaching the nearest Index-Hop circle, where this number is,

\[
\text{Length} = \log(N) \cdot [1 + (\log(N) - 1) + (\log(N) - 1)^2 + \ldots + (\log(N) - 1)^{\frac{K}{2} - 1}]
\]

4. Simulation and Results

Various scenarios has been taken and simulated with NS2 to show efficiency of our method. Nodes are distributed randomly in the network. Each node in the network was assigned index value from random distribution. All experiments were run for different values of Index-Hop K that K=1 to K=6 that we used only values which have advantage for us in network, because if K= 1 to K=3 we receive broadcast advantage without controlled-flooding and each node must have a large memory storage to save all other nodes indices in the network. Results of our method cost are shown in Figure 8. With lookup to Figure 8 and Figure 9, we notice that if K=4 there is balances between the files lookup cost and per node storage. Comparison of our method cost and Chord protocol cost (Chord Protocol is one of most important protocols that works in MANETs efficiently) is shown in Figure 10.

5. Conclusion

Increasing popularity of P2P systems and MANETs gives attention to combine both systems. Thus having a community which can share their files and data in mobile environments.

Current P2P protocols fail to perform well with this combination, because there is multi-layer routing redundancy. So we suggest simple method which bring down routing and query lookup mechanisms from application layer to network layer and optimize network layer query routing. Our method reduces query lookup complexity by evenly distributing files indices throughout the network.
6. Reference


