IMPLEMENTATION OF BOOTSTRAPPING FOR P2P OVERLAYS IN MANETS

Afzal Mawji and Hossam Hassanein

Telecommunications Research Lab, School of Computing, Queen's University
Kingston, Ontario, Canada, K7L 3N6
{mawji, hossam}@cs.queensu.ca

ABSTRACT

Peer–to–peer networks are immensely popular, but bootstrapping them usually requires centralized infrastructure. In fully decentralized environments such as mobile ad hoc networks, the use of centralized solutions are not possible. This paper presents a method of bootstrapping P2P overlay networks running on MANETs, and demonstrates a Java implementation of the algorithm. Simulation results show that the algorithm performs well and the implementation confirms the feasibility of the algorithm.

1. INTRODUCTION

Peer–to–peer (P2P) networks are popular among users and they are used for many different application types, such as file sharing, Voice–over–IP, gaming and instant messaging. Mobile ad hoc networks (MANETs) and peer–to–peer networks share many similarities such as decentralization, dynamic organization and topology, and failure resilience. Therefore, it is a natural evolution to combine them together so that a P2P overlay runs on a cooperative MANET, for which we use the term P2P–MANETs.

Adopting existing P2P overlay techniques and using them in MANETs is undesirable even though they share similarities, because there are also signicant differences. P2P networks are very large–scale with millions of simultaneous users, and are designed as overlays on the Internet, where nodes are immobile. On the other hand, MANETs have far fewer, and proximate nodes, the devices are severely resource–constrained in comparison, particularly in terms of energy, and the links between nodes usually have higher delay.

Before a node can participate in a P2P overlay, it must first "bootstrap" itself into the overlay. Bootstrapping addresses the problem of joining a P2P overlay. The node must first find other nodes that are already part of the overlay and attempt to create an overlay connection with some of them. This problem has largely been ignored in the research literature, even for wired P2P networks. Without a means of bootstrapping, it is impossible to participate in a P2P overlay. Many P2P networks are designed to be fully decentralized with the exception of bootstrapping. Existing bootstrapping techniques usually require a centralized service, which is problematic in MANETs.

This paper discusses a completely decentralized algorithm for bootsrapping MANET nodes into P2P overlays. It makes use of MANET–wide multicasting for both join queries and responses. Nodes cache the responses for future use. The bootstrapping algorithm has been implemented in a small ad hoc network. Simulation results show that the algorithm performs well, while the Java implementation confirms that the algorithm works in a real wireless network under several test scenarios.

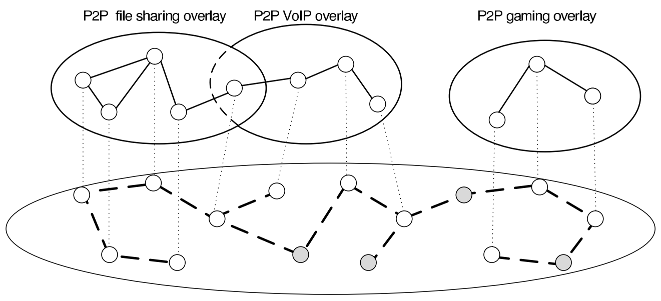


Fig. 1. Multiple overlays on a MANET

The rest of the paper is organized as follows. Section 2 briefly describes the bootstrapping algorithm. Section 3 presents some simulation results. Section 4 discusses the implementation of the algorithm. Section 5 gives our conclusions.

2. ALGORITHM DESCRIPTION

The bootstrapping algorithm has been presented in detail in [1], so we present only a brief overview here.

In order to boostrap, MANET nodes must first discover overlays by finding peers that are already participating in a P2P network. The node first examines its local cache for previously–known peers. If none are found, or if the information is stale, the node multicasts a MANET–wide join request. When a peer receives a request, it may multicast a MANET–wide reply and all nodes in the network will cache this information. The node wanting to join the overlay can then select the best peers to join, based on the overlay topology control algorithm being employed.

In a given MANET, there may be several simultaneous, extant P2P overlays. For example, there may be a file sharing overlay, a VoIP overlay, and a gaming overlay, as shown in Figure 1. The three overlays would all make use of the same underlying mobile ad hoc network. Some MANET nodes may be part of more than one overlay at the same time, while other nodes may not be a member of any overlay at all. The proposed algorithm works regardless of how many overlay networks exist within the MANET. The requesting node may choose to join a specific overlay, or it may choose to query all overlays to determine which ones are available.

All nodes in the MANET must join a multicast group which is used for both sending join requests and responding to them. Any routing protocol may be used, for both multicast and unicast messages.

When a node wishes to join a P2P overlay, it multicasts a join request (*mc_join_request* message) to the multicast group. If the node knows which overlay it wishes to join, this can be indicated in the initial request. Otherwise, the join request is considered generic and nodes belonging to all overlays in the MANET may respond.

All peers in the overlay willing to accept a connection from the requesting node then multicast a response (*mc_join_reply* message) back to the group. This response includes the peer's address and any information required by the topology control algorithm, such as hop distance and remaining energy. If a peer in the overlay is not willing to accept any new connections, it simply ignores the request.

All devices cache the response information, even if they are already part of a P2P overlay. Nodes may receive multiple responses per request. They use the selection criteria of the overlay topology control algorithm they are employing to determine how many, and which neighbours they should connect to. Current members of the overlay may use these responses to update their view of the topology. These peers may decide that the peer response they have just received now represents a better neighbour in the overlay than an existing one.

When deciding to join the P2P overlay, a node first consults its local cache and selects peers based on the information contained therein. If there are fewer known peers than the node wishes to connect to, or the information about a peer is believed to be out of date, the node proceeds with a multicast query as described earlier. Otherwise, unicast connection requests (*conn_request* messages) are sent to the desired peers.

Cached information is maintained in a soft state. When a multicast response is received, it is placed in the cache with a fixed time-to-live (TTL) value. This value is then decremented as time passes, and the entry is removed from the cache when it reaches zero. If the cache is full, the new entry will replace the one with the shortest TTL value.

3. SIMULATION RESULTS

This section evaluates the performance of the bootstrapping algorithm and compares it with two alternatives in simulation.

3.1. Simulation Parameters and Metrics

The MANETs are simulated in *ns–2* 2.33 [2], have an area of 1500m $\times$ 1500m, and contain 100 nodes. A transmission rate of 54 Mbps is used. All nodes are evenly distributed in the simulation area. Overlay sizes are varied from 50 to 100 nodes, in increments of 10.

The random waypoint model is used for mobility with nodal velocities distributed according to a uniform distribution with minimum speed of 1 m/s and maximum speed of 3 m/s. The pause time is uniformly distributed with a mean of 60 seconds and the simulation time is set to two hours. MAODV is used as the multicast routing protocol and AODV is used as the unicast routing protocol. The energy consumption model used in the simulations is the linear model proposed by Feeney [3].

Initially the network is in a steady state with all peers, save one, joined. In each simulation experiment, one randomly selected node that is not a member of the overlay attempts to join it, and another randomly selected peer that is a member, leaves it. This process repeats every 30 s.

We test the performance of our algorithm both with and without caching enabled. When caching is turned on, it is not fixed at a pre–specified rate. Instead, we attempt to model a real MANET and determine the natural cache hit rate achieved during bootstrapping. To simulate a P2P file–sharing overlay, constant bit rate (CBR) traffic is being sent between members of the P2P overlay. The traffic consists of 1000 byte packets sent 100 times per second and is started and stopped at random times, between random peers.

We compare our scheme with two other well–known schemes. The first is a flooding algorithm. In this scheme, when a node wishes to join the P2P overlay, it floods the join request to every node in the MANET. The second comparison scheme is Random Address Probing with address knowledge. With the normal Random Address Probing scheme, nodes were virtually never able to connect to the overlay at all due to the low probability of randomly selecting a valid address. Therefore, in order to provide a useful comparison, the Random Address Probing technique was given additional information in the form of the addresses of all nodes existing in the MANET so that a valid random MANET node would be chosen each time. After three minutes without receiving a query response message, RAP gives up. We use three variations of RAP, which we name RAP–1, RAP–5, and RAP–10. The numbers indicate how many addresses the requestor contacts simultaneously. For example, in RAP–5, the node will send probes to 5 random addresses, then wait for a response. If there is no response within 30 s, another 5 addresses will be sent probes, and the process may be repeated until three minutes have elapsed.

3.2. Simulation Analysis

In this section, we present some updated simulations results for the bootstrapping algorithm. Due to space constraints, only select results are shown. The simulation results obtained in all experiments have a 95% confidence level based on 10 independent runs. The confidence intervals are indicated by the error bars in the figures below.

In order for a node to successfully bootstrap itself into a P2P overlay, it must connect to at least one neighbour. Figure 2 shows the success rate for nodes trying to join an overlay. The proposed distributed bootstrapping algorithm performs very well, both with and without caching enabled. It achieves 100% success in most cases. The flooding technique performs reasonably well and its success rate does not change much with the overlay size because the flooding mechanism contacts every MANET node. However, this is also a drawback because it creates excessive network traffic that sometimes prevents bootstrap responses from reaching the requesting node within the timeout period. This is why the flooding technique does not achieve 100% success. The RAP algorithms generally perform less well, particularly RAP–1. RAP–1 contacts very few nodes and as a result, receives very few responses. RAP–5 and RAP–10 do much better since they contact a larger proportion of nodes. As the number of overlay members increases, the RAP algorithms do better because the chances of contacting an overlay member rise. However, even though RAP–10 has a high success rate with 100 nodes, it is not 100% because some peers do not respond to requests since they have already have five neighbours by the time they are contacted. Also, the RAP algorithms' success rates have high variability, as seen by the large error bars, because of their inherently random nature.

The number of received responses to a join query is important because more responses provides the node with a greater selection of potential neighbours to choose from. If only a single response is received, then only one neighbour may be selected. However, if several responses are received, they can be evaluated by the topology control algorithm and the best neighbours chosen. Figure 3 shows that the proposed algorithm receives the highest number of responses per query. The uncached version does slightly better because the responses are more up–to–date. However, both receive many more

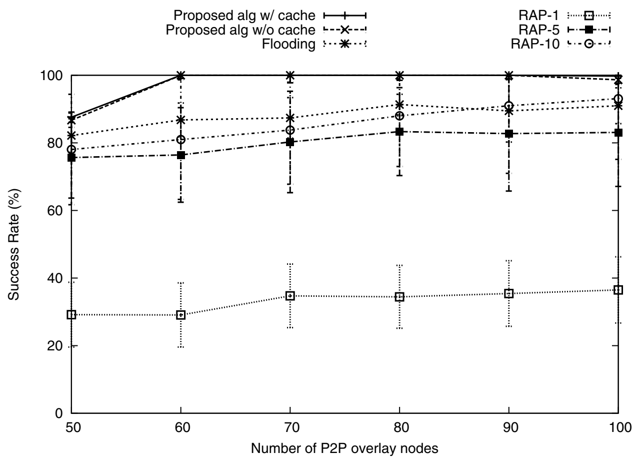


Fig. 2. Bootstrap success rate

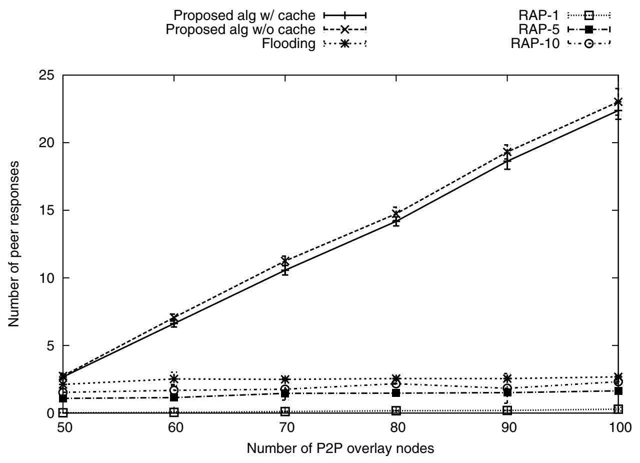


Fig. 3. Number of responses received

responses than the other algorithms, particularly with larger network sizes. As explained earlier, the flooding algorithm creates so much network traffic that requests and responses are dropped, and so the number of responses remains flat with the network size. The RAP algorithms also do not change much with the network size because they will tend to contact the same number of nodes in a given time frame, regardless of the network size.

4. IMPLEMENTATION

The bootstrapping algorithm has been implemented as a Java application using the Microsoft Mesh Networking Academic Resource Toolkit [4] for multihop communication. The toolkit implements an ad hoc routing protocol that uses link quality measurements. It also implements a virtual network adapter that enables higher level software to run unmodified over the ad hoc network.

It is not practical to test large MANET networks, so no specific performance data was measured and instead the focus was on proving the feasibility and correctness of the algorithm for small networks. The bootstrapping application uses the multicast algorithm presented in [5].

The bootstrap algorithm is implemented in several Java classes [6], of which we now describe the main ones. The Client class is used initiate join queries, the Server class is used to handle all incoming mesages, and the SendQuery class is used by both the Client and Server classes to transmit packets. It takes care of the details of determining whether to multicast or unicast the packet. Packet exchange between nodes is done via UDP.

The Client class is responsible for sending out join queries. This thread will first look in the host's cache table to see if it already knows of some nodes that are members of the desired overlay. If it has knowledge of an overlay member from previous responses then the Client will send a unicast message to the peer for a join request. If the overlay it is looking for has no entries in the table or there are too few peers a join request message is multicast to the group. This join request message contains the message header, TTL, overlay name, packet sequence number, peer score, and source IP address.

The message header specifies the type of message and can be either a multicast message to all peers in the MANET, or to a specific peer directly. The TTL field is initialized to nine when the packet first leaves the host and is decremented after every hop. If the TTL is equal to zero the intermediate node will not prepare the packet for multicasting and will instead drop it. The overlay name is used to describe the type of the overlay the client is looking for. The packet sequence number is used to identify packets from a particular node. The peer score enables nodes to evaluate each other and is essentially a utility value. The Source IP address is the address of the node that first generated the packet; this address remains the same for the entire life span of the packet.

The Server class is used to make decisions about incoming and outgoing packets, provide service to other nodes, and decide what peers to connect to based on the score they provide. Once the Server receives a packet it will extract the information from it. Next, the sequence number of the packet that was received is checked and if it has been received before, the packet is dropped. Finally, based on the header value, the packet will be processed accordingly. If it is a multicast packet, the packet will be sent according to the multicast algorithm. Next, the Server checks if it is an overlay member and if it is able to allow the requestor to join it as a neighbour. If so, a response is sent back to the sender, and at the same time to all the other nodes in the network.

If the packet header indicates that this is a join query reply message, it is multicasted back out. In addition, the new peer is added to the cache table. If the message is a join request, the new peer is added.

The application was tested under different network configurations, which are described below. The tests were successful and the results met with our expectations. The equipment used during the tests included several desktops and laptops running Windows XP SP 2.

Three network configurations were used to test the bootstrapping application and verify its functionality. First, a configuration of three nodes, arranged in a line was tested. One node was the client node and the device at the other end of the line was the server node. A multicast query was sent, and the server multicasted back its response information. After this phase, the two nodes exchanged a unicast confirmation and acknowledgement.

Next, a longer line configuration of five nodes was tested. Again, the device at one end was the client node and the device at the other end of the line was the server node. All packets were required to pass through four hops to reach the destination. This configuration tested the performance over multiple hops, and the results proved to be successful.

Finally, a multi–node network configuration was tested to verify

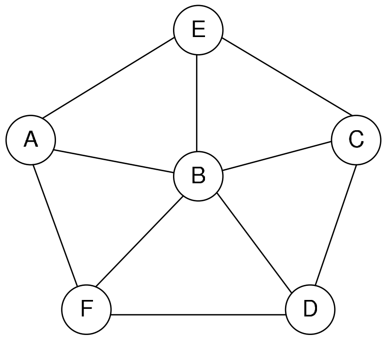
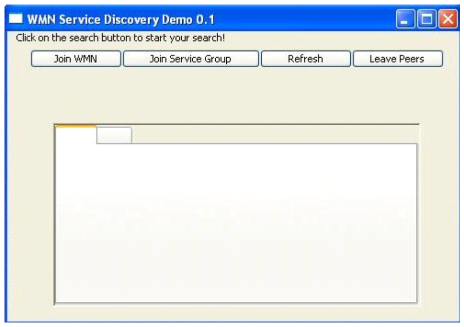


Fig. 4. Multi–Node Configuration



```
Click on the search button to start your search!
```

Fig. 5. The GUI on startup

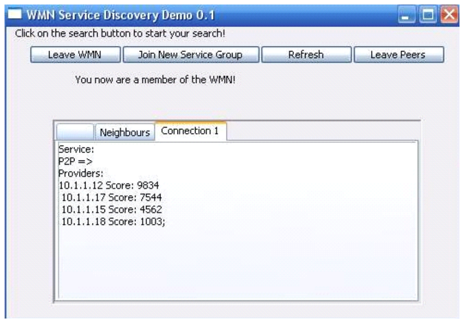
that the bootstrapping service could find multiple peers and choose the best neighbours (see Figure 4). Nodes tried to connect to peers with the most energy remaining. This test required a P2P file–sharing application in order to check the results. Node B is the client node and nodes A, E, F, C, and D run the same service. The nodes start joining the P2P–MANET one at a time and all have the same initial score. The P2P application is set up to connect to only the three best peers at a time. As nodes communicate over the MANET, their energy level gradually decreases. Assuming that all peers start at the same energy level, the node that joins the network last has the highest energy level, and hence the highest score. In our configuration, node A was the first to join the network and node C joined the network last. The test output showed that the peers performed as expected. Node C was listed as the one with the highest score in the peer table, and replaced node A which had the lowest score.

A GUI was implemented for the bootstrap application in order to verify its functionality. Figure 5 shows the screen on startup. Figure 6 shows the results of a query for “P2P” appearing in a new tab. Four peers have responded and they are ranked by score. The application was designed with the idea of multiple P2P overlays existing, and this is shown in Figure 7. A gaming overlay, a P2P file–sharing overlay, and a user sharing a printer have been found.

5. CONCLUSIONS

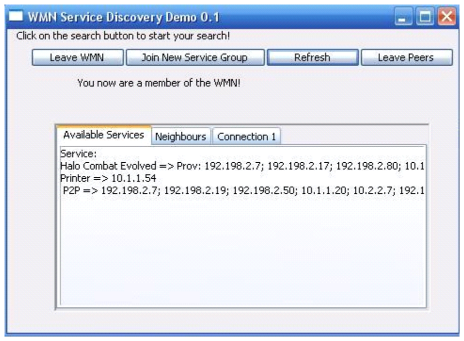
This paper presented a decentralized bootstrapping algorithm for P2P–MANETs, along with a discussion of its implementation. Nodes multicast their requests MANET–wide, peers respond with a MANET–wide multicast, and all nodes cache the responses for possible future use.

Simulation results demonstrate that the algorithm outperforms competing systems. The Java implementation verifies the feasibility of the algorithm for use in practical P2P–MANETs.



```
Click on the search button to start your search!
You now are a member of the WMN!
Service:
P2P =>
Providers:
10.1.1.12 Score: 9834
 10.1.1.17 Score: 7544
 10.1.1.15 Score: 4562
 10.1.1.18 Score: 1003;
```

Fig. 6. Query responses



```
Click on the search button to start your search!
You now are a member of the WMN!
Service:
Halo Combat Evolved => Prov: 192.198.2.7; 192.198.2.17; 192.198.2.80; 10.1
Printer => 10.1.1.54
 P2P => 192.198.2.7; 192.198.2.19; 192.198.2.50; 10.1.1.20; 10.2.2.7; 192.1
```

Fig. 7. Multiple overlays

6. REFERENCES

[1] A. Mawji and H. Hassanein, “Bootstrapping P2P overlays in MANETs,” in *IEEE Global Telecommunications Conference (GLOBECOM)*, 2008.

[2] ns 2, “The network simulator,” http://www.isi.edu/nsnam/ns/.

[3] L. M. Feeney, “An energy–consumption model for performance analysis of routing protocols for mobile ad hoc networks,” *Mobile Networks and Applications*, vol. 6, no. 3, pp. 239–250, 2001.

[4] Microsoft Research, “Mesh networking academic resource toolkit,” http://research.microsoft.com/en–us/projects/mesh/, 2007.

[5] H. Hassanein, Y. Yang, and A. Mawji, “A new approach to service discovery in wireless mobile ad hoc networks,” *Int. J. Sensor Networks*, vol. 2, no. 1/2, pp. 135–145, 2007.

[6] P. Kolomitro, “Service discovery system for use in wireless mesh networks,” 2008, Queen’s University, CISC 499 report.