ODMRP-ASYM (On Demand Multicast Routing Protocol) For Linux Implementation

Eric Bostrom, Jason Lin, Joon-Sang Park
{ebostrom, jasonl, jspark}@cs.ucla.edu

University of California, Los Angeles
Los Angeles, CA 90095

Abstract

This paper presents our work with an ODMRP (on demand multicast Routing Protocol) to extend its functionality to work with asymmetric links in a Linux Environment. We were given the functional code for the ODMRP protocol, as well as a test-bed to work with which consisted of five Dell Latitude laptops equipped with Lucent Wireless PCMCIA 802.11 cards. Our implementation consists of adding to the existing ODMRP code two packet definition types, as well as handlers for these packet types. The packet types are the new Loop Discovery Packet and Loop Marking Packet. These are used to detect loops in order to solve the asymmetric link problem in routing. This is done in the reply phase of the path discovery for ODMRP. When the node sending the Join Table Packet times out while waiting for an acknowledgement, the Loop Discovery Packet is then transmitted from that node. This packet goes around the loop collecting IP addresses of the nodes in the loop, and when the node that timed out sees this packet again, the loop has been discovered. That node then sends out the Loop Marking Packet, which marks all the nodes in that loop as forwarding nodes. Through this procedure, the forwarding group is created even with asymmetric links, and the sending node is able to have an established path to the receiving node. Much of the difficulty of this implementation was in the configuration of the test-bed to be able to enable the laptops to communicate correctly with each other.

1. Introduction

Most routing protocols developed for wireless ad-hoc networks do not consider the existence of asymmetric links—namely, links with different characteristics (e.g., range, quality, etc) in the two directions. The most common cause of this asymmetry is the difference in transmitting power from one node to another, but other causes may be the presence of external interference. In the extreme case, the asymmetric link becomes a unidirectional link, i.e. a link that connects node A to node B, but not node B to node A. Asymmetric links are often found in the real deployment of wireless ad-hoc networks as shown in recent experimental studies.

Other sources of asymmetric links include directional antennas and power (or topology) control algorithms. Directional antennas, which concentrate radio signals toward a specific direction instead of spreading signals out to a circular area surrounding the transmitter, is devised to overcome performance limitations of the conventional omni-directional antenna system. All links using the directional antenna system are inherently asymmetric. Power control algorithms are used to protract the lifetime of a network. In some wireless ad-hoc network scenarios, such as sensor networks with intrinsically limited resources, it is important to use power efficiently so that these networks can have longer lifetimes. Most power control protocols disallow or disregard asymmetric links, yet some yield asymmetric links.
On-demand routing protocols using reverse path technique that backtrack upstream node will not work if they confront asymmetric links in the route discovery stage. This problem is referred to as the asymmetric link problem in routing. Recently, a number of researchers investigated the impact of asymmetric links on the performance of unicast routing protocols and extended them to support asymmetric links.

There are two ways to tackle the asymmetric link problem. Most of the existing work deals with asymmetric links by simply detecting and eliminating them. This approach is natural for unicast routing in that the point-to-point data transmission (i.e. unicast) in the standard wireless MAC protocol, IEEE 802.11 DCF, works well only with bi-directional links as it requires RTS/CTS/ACK handshaking. The other approach is to exploit asymmetric links, which we advocate because of the following reasons. First of all we believe that routing protocols should guarantee to find a path if a path exists. If a routing protocol fails to find an existing route, it is considered to have a serious flaw. The problem with the elimination of asymmetric links is that one cannot get 100% route discovery guarantee in sparse networks even if the topology is connected. Partitioning is also a consideration in network performance, which makes disregarding asymmetric links unacceptable.

In Section 2, a background is given about how our protocol supports asymmetric links in the multicast protocol. Section 3 gives us further details regarding the implementation. Section 4 describes our testbed and testbed configuration. Section 5 describes the shortcomings of our implementations, and possibly future work to correct these. And finally Section 6 contains our conclusions about this project.

2. Background

ODMRP-ASYM was based on the ODMRP, which was developed at the Wireless Adaptive Mobility Lab (WAM) at UCLA. ODMRP uses on-demand routing to techniques to improve channel overhead and efficiency as well as improve scalability. ODMRP uses a mesh-based structure to ensure that only a subset of nodes are responsible for forwarding multicast packets along a shortest path between any member pairs. This forwarding group mesh-based concept is a departure from a tree-based multicast scheme. On-demand routing schemes flood to discover routes as needed. Generally, flooding by the source is used to discover a path to the destination. ODMRP uses this forwarding group to building a forwarding mesh by creating a set of nodes to forward multicast data on the shortest paths between any member pairs. ODMRP-ASYM is an extension of ODMRP that enables routing with asymmetric links and around the loops.

The purpose of Loop Detection procedure is to find an alternate path from a node which does not have a link to an upstream node. Basically, a packet called Loop Detect Packet (LDP) is flooded with the limited TTL (Time-To-Live) which is the expected maximum size of the loop. Experimentally, we choose 6 for this TTL. When the LDP gets back to its sender, it includes a list of node addresses in the order in which they saw the packet.

In original ODMRP, different from Join Query packets, every Join Reply packet is to be acknowledged since the reliable delivery of the Join Query packets is critical for establishing a forwarding path. If a Join Reply is not acknowledged, it is retransmitted at most two times. We utilize this ACK scheme to detect asymmetric link on the fly. We consider a link to be asymmetric and initiate the loop detection procedure if a Join Reply is not ACKed.
Once a valid loop is found by the LDP, the next step is to flag the nodes in the loop which are parts of the downstream path to the destination as forwarding nodes. To mark the nodes in the loop, we circulate a Loop Mark packet around the loop. The node that detects the loop, node B in figure 3, creates a Loop Mark packet by copying the loop node list in Loop Discovery Packet and transmits it to the next node in the loop, in our case, node C. A node placed after the loopsunit in the list should be flagged as forwarding node. On receiving Loop Marking Packet, the loopsunit node designated in the packet sends out a normal Join Reply packet towards multicast sources.

3. Implementation Details

The diagram below shows the structure of our Loop Discovery Packet which includes the layers of IP, IGMP, and our loop discovery information (shown in figure 1). In the loop discovery layer, there are many similarities to the JRP packet. The packet type, hop count, group IP, sequence number, source IP, and the intermediate hops IP address array are all used in similar fashion. The intermediate hop array is used to keep the path from the source to destination. This array is crucial to marking nodes as forwarding later with a loop-marking packet. Once the algorithm has located the smallest valid loop, that path from the LDP packet is used again to traverse the loop and mark those nodes as forwarding nodes.

The Loop Marking Packet (LMP) is the same as the LDP packet with only a different type in packet header. It requires slight changes to the handler functions when compared to the LDP handler code. As the node travels around the loop, each node found that is contained in the hop IP array is marked as forwarding, since it was part of the loop found using the Loop Discovery Packet. A node marked as forwarding is used later to forward the packets on the path to the destination.

<table>
<thead>
<tr>
<th>version</th>
<th>TOS</th>
<th>length</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>offset</td>
<td></td>
</tr>
<tr>
<td>TTL</td>
<td>protocol</td>
<td>checksum</td>
</tr>
<tr>
<td>source address</td>
<td></td>
<td></td>
</tr>
<tr>
<td>destination address</td>
<td></td>
<td></td>
</tr>
<tr>
<td>type</td>
<td>max response time</td>
<td>checksum</td>
</tr>
<tr>
<td>group address</td>
<td></td>
<td></td>
</tr>
<tr>
<td>type</td>
<td>hop count</td>
<td>reserved</td>
</tr>
<tr>
<td>group IP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sequence number</td>
<td></td>
<td></td>
</tr>
<tr>
<td>source IP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>hop IP[n]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(a) Loop Discovery Packet LDP Format

Figure 2. The Loop Discovery Packet Format.
4. Testbed

Our test-bed consists of five Dell Inspiron Laptops equipped with 802.11b cards, which function in ad-hoc mode. To create the asymmetric links, a small program called iptables selectively drops packets from certain nodes at each node in our topology. This simulates asymmetric links via our ad-hoc setup. The actual setup of these laptops to run the correct topology we desired was time consuming and difficult. Setting up the test-bed was a major hurdle for this project. A script was written for each laptop to set this topology up so each time we worked on our test-bed it would be consistent.

The test-bed consists of nodes S and D functioning as source and destination nodes, respectively. The intermediate nodes A, B, and C contain the new ODMRP-ASYM routing code for the asymmetric links from A->B and C->A.

To provide a good visual demonstration of our protocol, we used a small program to simulate a video conferencing session on both the sender and destination sides. This program shows transfer metrics such as frames per second and bits per second. With the asymmetric links, we run the original implementation of ODMRP to show that it fails in a topology with asymmetric links. However, with our new ODMRP-ASYM, the video application shows some throughput over asymmetric link topology.

5. Non-Compliance to Exact Specification

Our protocol is not exactly compliant to the Specification for ODMRP-ASYM. Some of this is due to the fact that the original ODMRP code given to us was not fully compliant also. Therefore, we did not want to spend the majority of our time fixing the existing code, but rather showing that the ODMRP-ASYM protocol could be implemented. Thus, we document our shortcomings in this project so that perhaps future work could improve upon our project.

Acknowledgements were not implemented in the original protocol, and should have been done. We fixed this by listening for a passive acknowledgement. However, the exact protocol states that the JTP should be resent two times, and we should wait for 3 timeouts on acknowledgements before we call our Loop Discovery procedure. Also, these acknowledgements are not supposed to be passive acknowledgements, but rather normal acknowledgments. Also, we only wait for one timeout to call our procedure. These small inconsistencies are fairly easy to fix.

Figure 3. Our testbed configuration of nodes and links.
Another issue is that there are two protocols actually running. There is a sender/receiver protocol, and there is an intermediate node protocol. We have only modified the intermediate node protocol to be able to handle asymmetric links. This can easily be extended to the source and destination nodes also.

Also, our protocol only currently supports one loop. Since loops are rare, this may not be a problem. However, the specification calls for our procedure to be more thorough. It must disregard invalid loops, as shown in figure 4 below.

![Figure 4. Valid and Invalid Loops in the topology.](image_url)

Another of our shortcomings is due to poor commenting of code. In the JTP code, the commenting stated that it did not really contain the true Multicast group IP address, but rather did not use it and simply stored the source IP address. This would allow only one multicast group to use this protocol over these nodes. The comments stated that this needed to be fixed, but was not currently in working order. Thus, in our LDP packets and LMP packets, we thought we did not have access to the Multicast Group IP address. However, when our protocol was finished, (we hard coded the group IP) we discovered that the Group IP actually was present in the JTP packet. The problem in our LMP packet can now easily be fixed by taking the Group IP address from the JTP packet and putting it into our LDP and LMP packets. After this is done, then the LMP packet can correctly mark the packets as forwarding nodes for that particular group. Then, there can be many groups broadcasting over the topology concurrently.

6. Conclusion

This project was a success in that we were able to implement a scaled down version of ODMRP with support for asymmetric links in the Linux environment. The framework for a more robust version of our ODMRP-AYSM has been set, and thus future work includes extending our code to handle all that the ODMRP-AYSM protocol specifies.
7. References