BS-8-12

Software Implementation of an On-demand Multipath Routing Protocol for Multimedia Streaming in Mobile Ad hoc Networks

Shinya KOIZUMI, Kenta TANIYAMA, Takeshi MORII, Yukihiro KOTANI, Kazuhiro NOGUCHI and Jiro KATTO
Graduate School of Science and Engineering, Waseda University

1. Introduction

In ad hoc networks, link breaks between nodes happen frequently because of movement of nodes and unstable radio features. To cope with frequent topology changes, various routing protocols were proposed and submitted as RFCs. AODV (Ad hoc On-demand Distance Vector)[1] has clear advantages in its moderate overheads and its route convergence performance. However, AODV forms a single path and must rebuild a route from source to destination when a link break happens. The latency to reconstruct routes is a serious problem for live multimedia applications such as video streaming.

In order to minimize delay for reconstructing routes, some researchers focus on multipath routing protocols and evaluates through simulations [2][3]. Multipath routing protocols show better performance than single path ones in simulation evaluations. However, there were few actual implementations to validate the usage of multipath routing protocols in real world.

In this paper, we extend a single path AODV for multipath and implement it on laptop PCs. Extensions of RREQ/RREP packets and a routing table bring not only loop freedom but also more efficient multipath routing performance.

2. Proposed Protocol

We adopted Kernel AODV [4] developed by NIST for routing protocol implementation. We extend AODV to a multipath routing protocol by using a "source route list," which is attached to the RREQ/RREP packets. Source route list contributes to reliable creation of multiple paths, efficient management of neighbor nodes and maintenance of multiple candidates for data transmission routes even if overhead slightly increases.

2.1 Route Discovery Extensions

(1) Process of RREQ

First, to find routes for a destination node, a source node broadcasts a RREQ packet. When an intermediate node receives the first RREQ packet, similar to AODV, it records a reverse route in its routing table. The intermediate node records its own IP address in the "source route list" field in the RREQ packet and re-broadcast the packet.

If an intermediate node receives a delayed RREQ packet from other neighbors, the node checks a source route field in the packet. This packet would be discarded immediately when the field contains the node's IP address, which means detection of a routing loop. If this address check is passed, the intermediate node accepts the RREQ packet and updates source route list. When the packet does not satisfy the update condition, it is simply discarded.

A routing table is managed by each intermediate node. The *nexthop*, *hopcount*, and *route lifetime* fields are stored for route information in each reverse route.

(2) Process of RREP

Any nodes receiving several RREQ packets generate multiple RREP packets toward a source node by multiple unicasts. When a destination node receives the first RREQ packet, the node unconditionally accepts the packet and generates a RREP packet to set up a primary route similar to AODV. When the node receives a delayed RREQ packet, the node conditionally accepts the packet according to RREQ extension fields and generates a RREP packet to send along a secondary route.

When an intermediate node receives the first RREP packet, the node forwards it to any neighbors over the reverse routes toward a source node by multiple unicasts and updates its routing table. Routing loops can be easily avoided by using a source route list attached. If the node receives a delayed RREP packet, it updates its routing table similar to the RREQ extension case, and discards the RREP packet.

Finally, the fastest RREP for a source node provides a primary route. The others are examined in the source node as well as in intermediate nodes, and some of routes are accepted as backup routes according to the specific metric. We use "delay metric" [3] currently for this implementation. Data transfer begins just after the primary route is established.

2.2 Route Maintenance Extensions

(1) Process of RERR

When a node cannot receive HELLO messages from neighbors, the node detects a link break. If neighbor nodes do not have any backup routes, the nodes invalidate their current routing tables and find precursor lists to send RERR packets to its neighbor nodes. Otherwise, the nodes immediately change a current route to a backup route to avoid new route discovery process.

3. Evaluation and Demonstration

We carry out two kinds of experimentations comparing our proposal with the original AODV; performance evaluations of route recovery latencies and throughputs and demonstration of live streaming using webcam. Both two scenarios are performed in the indoor environment.

3.1 Performance Evaluation

(1) Route Recovery latency

In this section, we compare route recovery latencies using a Ping application over the topology in Figure 1. Node 1 sends 512 byte Ping packets at a rate of 10 packets/s to node 8. Under this

condition, we shut down intermediate nodes to break some links for observing how new route discovery affects on the communication in our implementation.

Figure 2 represents the average recovery time. When a link break occurs, our proposal can reduce the delay of recovery time by changing a primary transmission route to a backup route.

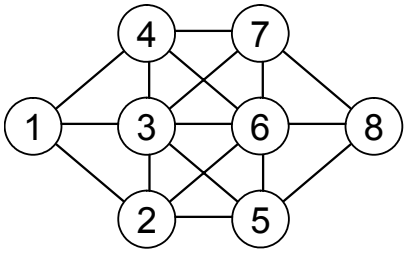

Figure1: Network topology

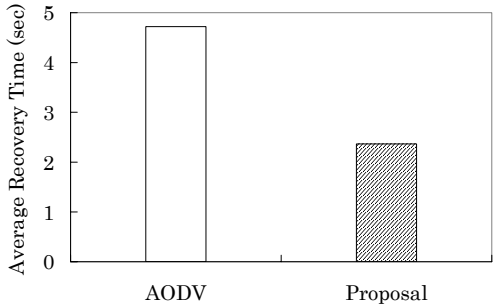

Figure 2: Average recovery time

(2) Comparison of throughputs

Next, we evaluate throughputs from node 1 to node 8 in Figure 3. Nodes 3 and 6 are placed in the third floor, and the others are in the fourth floor. There are a number of routes, which include stable or unstable links. As a characteristic of AODV, the shortest hop but unstable route tends to be selected in route discovery phase or in HELLO message exchanges. However, our proposal uses HELLO messages only for link life time update and unstable links are replaced by accumulated stable backup links. So, the routes constructed by our proposed protocol are less affected than by AODV.

Figure 4 shows average throughputs. At this experiment, we flow TCP streams for 30 sec to measure throughputs. Our proposal has an average throughput of 349 kbps which is 1.2 times higher to the one of AODV in Figure 4. This is because the routes for transmission converge on a stable route more quickly in the proposal than AODV.

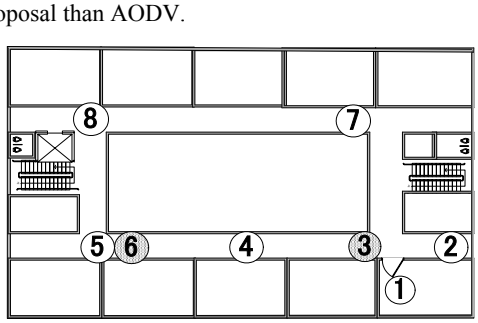

Figure 3: Layout of the testbed

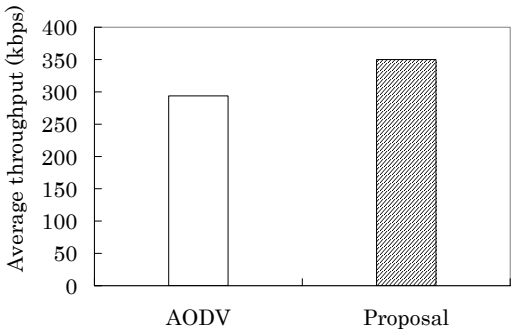

Figure 4: Average throughput

3.2 Demonstration of Live Streaming

Finally, we demonstrated live streaming, attaching webcam to node 8 of the destination. Node 1 initiates route discovery and requests video streaming after route establishment.

In case of AODV, the video quality was not so bad without heavy packet losses. After a while, however, connection was unconsciously broken and timeout error (picture freeze) happened. As mentioned above, this is due to unstable link selection of AODV which causes its time out.

On the other hand, our proposal was able to play video without stress similar to AODV. In addition, we could hardly find packet losses in the played video sequence and we can observe video for long time without timeouts.

4. Conclusions

In this paper, we implemented a testbed of our on-demand multipath routing protocol and evaluated its performance demonstrating live streaming for sensing applications over the testbed. The routes of unstable links are adaptively replaced to stable routes in the routing tables during communication, which will be utilized as backup routes later. Streaming video is played seamlessly without timeouts when route change has occurred by promptly switching to one of backup routes. Performance evaluations and demonstrations show improvement on stable route convergence to original AODV constructing a single path.

As future work, we plan to control transmission rates in MAC layer using received signal to noise ratio with the harmonized rate control in video compression layer.

References

[1] C.Perkins, E.B-Royer, S.Das, “Ad hoc On-demand Routing Vector (AODV) Routing”, RFC3561, July 2003.

[2] M.Marina, S.Das, “On-demand Multipath Distance Vector Routing in Ad Hoc Networks”, IEEE ICNP, November 2001

[3] Y. Sakurai, J. Katto, “AODV Multipath Extension using Source Route Lists with Optimized Route Establishment,” International Workshop on Wireless Ad-hoc Networks (IWWAN), May 2004.

[4] Kernel AODV 'http://w3.antd.nist.gov/wctg/aodv_kernel/'