Performance Evaluation of Two Congestion Control Mechanisms with On-Demand Distance Vector (AODV) Routing Protocol for Mobile and Wireless Networks*.

Azzedine Boukerche

University of Ottawa, Canada
boukerche@site.uttowa.ca

Abstract. In this paper, we focus upon the congestion control problem in on-demand distance vector routing (AODV) protocol for mobile and wireless ad hoc networks. We propose to study two mechanisms to deal with the congestion problem within AODV routing protocol. First, we investigate a randomized approach of AODV protocol. then we present a preemptive ad hoc on-demand distance vector routing protocol for mobile ad hoc networks. We discuss the implementation of both algorithms, and report on the performance results of simulation of several scenarios using the *ns-2* ad hoc network simulator.

1 Introduction

Ad hoc wireless networks are expected to play an increasingly important role in future civilian and military settings where wireless access to wired backbone is either ineffective or impossible. However, frequent topology changes caused by node mobility make routing in ad hoc wireless networks a challenging problem [3,4,5,6].

In this paper, we are concerned with the congestion problem and packets dropped that are mainly due to the path break in the ad hoc network where the topology changes frequently, and the packet delay due to finding a new path and the transfer time. As a consequence, we present a Randomized AODV protocol (*R-AODV*), and $P_r AODV$, an extension to the AODV routing protocol using a preemptive mechanism. While *R-AODV* uses a randomized approach to divert routing packets from congested paths to reliable and less congested paths, $P_r AODV$ scheme a preemptive approach to find a new path before the existing path breaks, thereby switching to this new path. Our main focus in this paper is to study the feasibility and the effectiveness of both congestion control mechanisms in the proposed protocol.

* Part of this research is supported by Canada Research Chair program and Texas Advanced Research Program grant ATP/ARP

H. Kosch, L. Böszörményi, H. Hellwagner (Eds.): Euro-Par 2003, LNCS 2790, pp. 1099–1108, 2003.

2 The Basic AODV Protocol

AODV uses an on-demand based protocol to discover the desired path, while using hello packets to keep track of the current neighbors. Since it is an on demand algorithm, it builds routes between nodes only upon request by the source nodes. It maintains these routes as long as they are needed by the sources. It is loop-free and self-starting. Earlier studies have also shown that AODV protocol scales quite well when we increase the number of mobile nodes [2]. AODV allows mobile nodes to obtain routes quickly for new destinations and does not require mobile nodes to maintain routes to destinations that are not in active communication. It also allows mobile nodes to respond to situations associated with broken link and changes in network topology in a timely manner, as we shall see later.

AODV uses a destination broadcast id number to ensure loop freedom at all times, since the intermediate nodes only forward the first copy of the same request packet. The sequence number is used in each node, and the destination sequence number is created by the destination for any route information it sends to requesting nodes. It also ensures the freshness of each established route. The route is updated if a new reply is received with the greater destination sequence or the same destination sequence number but the route has fewer hops. Therefore, this protocol will select the freshest and shortest path at any time.

• **Path discovery**: When a route is needed, the source invokes a path discovery routine. It has two steps: the source sends out a request and the destination return a reply. In what follows, we will discuss these two steps.

Request phase: When a route to a new destination is needed, the node uses a broadcast RREQ to find a route to that destination. A route can be determined when the RREQ reaches either the destination itself or an intermediate node with a *fresh* route to the destination. The *fresh* route is an unexpired route entry for the destination associated. The important step during the request phase is that a *reverse path* from the destination to the source can be set up. When a source node wants to find a path to a destination, it broadcasts a route request packet to its neighbors. The neighbors update their information for the source node, set up backwards pointers to the source node in their route tables, and rebroadcast the request packet.

Reply phase: When the request arrives at the destination or at an intermediate node that has a path to that destination a reply packet is returned to the source node along the path recorded. While the reply packet propagates back to the source, nodes set up forward pointers to the destination, and set up the *forward path.* Once the source node receives the reply packet, it may begin to forward data packets to the destination.

• **Data Packet Forwarding:** After the reply packet returns from the destination, the source can begin sending out the packet to the destination via the new discovery path, or the node can forward any enqueued packets to a destination if a *reverse path* is set up and the destination of the reverse path is the destination of the path.

A route is considered *active* as long as there are data packets periodically traveling from the source to the destination along that path. Once the source stops sending data packets, the links will time out and eventually be deleted from the intermediate node routing tables.

• **Dealing with the broken links:** If a link breaks while the route is active, the packets that are flowing in that path can be dropped and an error message is sent to the source or a local repair routine will take over. If the node holding the packets is close to the destination, this node invokes a local repair route. It enqueues the packet and finds a new path from this node to the destination. Otherwise, the packets are dropped, and the node upstream of the break propagates a route error message to the source node to inform it of the now unreachable destination(s). After receiving the route error, if the source node still desires the route, it can re-initiate a route discovery mechanism. However, we can add a preemptive protocol to AODV and initiate a rediscovery routine before the current routes goes down. In the next section, we will discuss this preemptive approach in more detail.

3 Randomized AODV (*R-AODV*) Protocol

AODV protocol is extended with a drop factor that induces a randomness feature to result in Randomized Ad-Hoc On-Demand Routing (*R-AODV*) protocol. During the route discovery process, every intermediary or router nodes between the source and the destination nodes makes a decision to either broadcast/forward the RREQ packet further towards the destination or drop it. Before forwarding a RREQ packet, every node computes the drop factorwhich is a function of the inverse of the number of hop counts traversed by the RREQ packet. This drop factor lies in the range of 0 to 1. Also, the node generates a random number from 0 to 1. If this random number is higher than the drop factor, the node forwards the RREQ packet. Otherwise, the RREQ packet is dropped. Dropping of RREQ packets does not necessarily result in a new route discovery process by the source node. This is due to the fact that the original broadcast by the source node results in multiple RREQ packets via the neighbors and this diffusing wave results quickly in a large number of RREQ packets traversing the network in search of the destination. A major proportion of these packets are redundant due to the fact that in the ideal case, a single RREQ packet can find the best route. Also, a number of these packets diffusing in directions away from the destination shall eventually timeout. Hence, in *R-AODV*, the aim is to minimize on these redundant RREQ packets, or alternatively, drop as much as possible of these redundant RREQ packets. The drop policy is conservative and its value becomes lesser with higher number of hops. As RREQ packets get near the destination node, the chances of survival of RREQ packets is higher. Hence, the first phase of the route discovering process, that is, finding the destination node, is completed as soon as possible and a RREP packet can be transmitted from the destination node back to the source node.

In *R-AODV*, the dropping of redundant RREQ packets reduces a proportion of RREQ packets that shall never reach the destination node, resulting in a decrease of network congestion. Hence, the ratio of the number of packets received by the nodes to the number of packets sent by the nodes, namely, throughput, should be higher in *R-AODV* compared to AODV.

The following algorithm is used in the decision making process of whether to drop the RREQ packets by the intermediary or routing nodes.

```
Step 1: Calculate drop_factor
        drop_factor = (1/(Hop_count_of_RREQ_packet + 1))
Step 2: Calculate a random value in the range of 0 to 1.
Step 3: If (random_value > drop_factor) then broadcast/forward RREQ_packet
        else drop RREQ_packet
```

4 Adaptive Preemptive AODV Protocol

In this section, we introduce the preemptive protocol first, then we discuss how this preemptive protocol is added into the original AODV,

The preemptive protocol initiates a route rediscovery before the existing path breaks. It overlaps the route discovery routine and the use of the current active path, thereby reducing the average delay per packet. During the development of P_rAODV development, we have investigated several preemptive mechanisms. In this paper, we will settle on the following two approaches:
(i) Schedule a rediscovery in advance: In this approach, when a reply packet returns to the source through an intermediate node, it collects the information of the links. Therefore, when the packet arrives at the source, the information about the condition of all links will be known, including the minimum value of the lifetime of the links. Hence, we can schedule a rediscovery $T_{rediscovery}$ time before the path breaks.
(ii) Warn the source before the path breaks: Some mechanisms are needed to take care of finding which path is likely to break. We can monitor the signal power of the arrived packets as follows: when the signal power is below a threshold value, we begin the ping-pong process between this node and its immediate neighbor nodes. This node sends to its neighbors in the upstream a hello packet called *ping*, and the neighboring nodes will respond with a hello packet called *pong*. Such *ping-pong* messages should be monitored carefully. In our approach, when bad packets were received (or we timeout on ping packets) a warning message should be sent back to the source during the monitoring period. Upon receiving a warning message, a path rediscovery routine is invoked.

In our preemptive AODV protocol, we combine the above two preemptive mechanisms, and add them to the original AODV. P_rAODV protocol is built based on the following assumptions. Future work will directed to eliminate such assumptions.

- *Congestion is caused only due to the path breaks.* Other factors such as the limited bandwidth available on wireless channels and the intensive use of resources in an ad hoc network cannot affect our protocol. We assume that the network has enough bandwidth available, and each node has enough capacity to hold packets.

- *The route discovery time can be estimated.* We estimate the discovery time by monitoring the request and reply packets, as the sum of the time required to process and monitor these two packets.
- *We can detect when a link breaks* by monitoring the transfer time of packets between the two neighboring nodes.

Based on the above assumptions, we present our P_rAODV as the following:
(i) Schedule a rediscovery: We have added an additional field, which we refer to as a lifetime, in the reply packet to store the information of the links along the new discovered path. Each link associates a time-to-live, a parameter that is set up in the request phase. The lifetime field contains the minimum value of the time-to-live of the links along a path. The reply packet also has an additional field, a timestamp, which indicates the time that the request packet was sent out.

Let us now see how we compute RTT, the estimated time needed to discover a new path. When the reply packet arrives at the source node, it computes RTT as $(arrivalTime - timestamp)$, then schedule a rediscovery at time $(lifetime - RTT)$. We use RTT as our estimated value for the time needed to discover a new path.
(ii) Warning the source: Recall that a preemptive protocol should monitor the *signal power* of the receiving packet. In our approach, we monitor the transfer time for the hello packets. According to [1], $P_n = \frac{P_0}{r^n}$, where P_0 is a constant for each link and n is also a constant, we can see that the distance, which connects to the transfer time of the packet via each link, has a one-to-one relationship to the signal power. Thus, by monitoring the transfer time of packets via each link, we can identify when the link is going to break.

In our protocol, when a node (destination) receives a real packet other than AODV packets, it sends out a pong packet to its neighbor (source), i.e., where the packet comes from. Note that there is no need to send this pong packet if we can include a field in the real packet to store the current time the packet is sent. However, in the original packet, it doesn't have such a field, hence, we need an additional *pong* packet to get such information. In both ping and pong packets, we include a field containing the current time that the packet is sent. When the source monitoring the transfer time is greater than some specific value c_1, it begins the *ping-pong* process. If more than two pong packets arrived with the transfer time greater than a constant c_2, the node will warn the source of the real packet.

Upon receiving the warning message, the source checks if the path is still active. This can easily be done, since each route entry for a destination associates a value, rt_last_send, which indicates if the path is in use or not. If the path is still in use, then a rediscovery routine is invoked.

Theoretically, these two cases can take care of finding new paths before current paths break. The first case is quite helpful if the data transmission rate is slow. If only one data packet is sent out per time-expired period, then even the current route is still needed. Note that this entry may not have packets to go through it during the next time period. This can happen if the interval between

two continuous packets is greater that the timeout period of the current route. The second case can be used to find a new path whenever a warning message is generated. However, those two cases are also based on some assumptions, such as the span of a link which can be estimated by the time out of ping packets and the values of those constants are accurate. Once we break those assumptions, the test results will be unexpected.

5 Simulation Experiments

We have implemented both *R-AODV* and $P_r AODV$ protocols using *ns-2, version 2.1b8*. The implementation of both algorithms is based on the CMU Monarch extension. The source-destination pairs are spread randomly over the network. By changing the total amount of traffic sources, we can get scenarios with different traffic loads. The mobility models used in our experiments to investigate the effect of the number of nodes are 500m x 500m field with different number of nodes. Each node starts its journey from a random location to a random destination with a randomly chosen speed uniformly distributed between 0-20m/sec. Once the destination is reached, another random destination is targeted after a pause. Varying the pause time changes the frequency of node movement. In each test, the simulation time is 1000 seconds. The experimental results were obtained by averaging several trial runs.

In order to evaluate the performance of *R-AODV* and $P_r AODV$ ad hoc routing protocols, we chose the following metrics:

- *Routing path optimality*: the difference in the average number of hops between the optimal (shortest) routing path and the real routing path.
- *Throughput:* (i.e., packet delivered ratio) is a ratio of the data packets delivered to the destination to those generated by the CBR sources.
- *Average delay per packet:* this includes all possible delays caused by buffering during route discovery latency, queuing at the interface queue, retransmission delays at the MAC, and propagation and transfer time.
- *Packet arrived:* this is the number of packets that are successfully transferred to the destination.

5.1 Simulation Results

In this section, we report on the performance results of simulation of several workload models of several scenarios we have used to evaluate the performance the both *R-AODV* and the preemptive ad hoc routing protocols.

AODV vs. R-AODV. Recall that the use of the drop factor in *R-AODV* will to a drop of some forwarding packets but the drop rate is conservative enough not to let the source node to result in a loop of initiation of route discoveries.

The routing path optimality is illustrated in figure 1. As we can see, the routing path optimality is better in *R-AODV* compared to AODV on an average.

We also observe that the routing path optimality is about 20% better in *R-AODV* compared to AODV on an average. larger percentage were observed with a larger field. This is mainly due to the induction of the dropping factor in *R-AODV* to reduce the node congestion.

Figures 2 shows the comparison of AODV with *R-AODV* on packet overflow due to the limited packet buffer space. It is observed that the number of packets overflowed in *R-AODV* is lower on the average. This is due to the induction of the drop factor in *R-AODV* to decrease node congestion. recall that the packets overflow is dependent on the dynamic network topology. It decreases with an increase in the pause time of the nodes in both AODV and *R-AODV*. This implies that for more stable networks, wherein the nodes are static for higher quantum of time, the latency in finding a route to a node is relatively less.

The throughput is given in figures 3. As we can see, *R-AODV* exhibits a higher throughput on the average since most of the forwarding packets that were dropped were redundant packets and allows more packets to reach their destination without being dropped due to reduce node congestion. The higher stable network dynamics result in a higher throughput. Overall, since throughput is higher for *R-AODV*, lesser number of packets that were sent out by the nodes in the network were lost during the route discovery process, thus reducing network congestion in *R-AODV* when compared to AODV.

Figure 4 show improved delay performance in most cases in *R-AODV*. Especially, for larger networks and higher mobility rate, the performance improvement shows a obvious contrast compared to AODV. For high mobility rate, the routes are frequently renewed with the newly discovered neighbors. Consequently the route update rate increases and *R-AODV* has more chances to find alternative paths due to decrease node congestion. The delay drops as the pause time increase. A main reason attributed to this observation is that as node mobility decrease, the route update interval increases since low node mobility results in less route updates. Thus, delay increases depending on node mobility. The overhead is presented in figure 5 which show similar characteristics for a given pause time in both *R-AODV* and AODV.

AODV vs. $P_r AODV$. In general, in both $AODV$ and $P_r AODV$, the number of nodes does affect their performance. We have observed that with the preemptive protocol, $P_r AODV$ can significantly improve the performance of the throughput, the average delay per packet, and the packet delivered. The pause time and the packet rate can also affect its performance. In the course of our experiments, we have observed that $P_r AODV$ has much better performance than AODV when the packet rate is low.

In our experiments, we set the traffic flow to 10-pairs, the interval traffic rate to 4, and the pause time to 600. As illustrated in Figure 6 our results indicate that that the number of nodes does not have any affect on both protocols, $AODV$ and $P_r AODV$. However, the preemptive AODV protocol indicates that it is more stable than AODV. This is mainly due to the fact that $P_r AODV$ balances the traffic quite efficiently to a new path before the current one breaks. The best

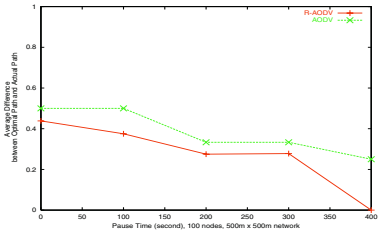

Fig. 1. Average Route Path Optimality – 500m x 500m

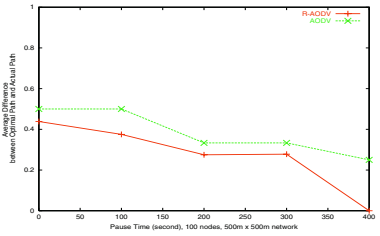

Fig. 2. Overflow – 500m x 500m

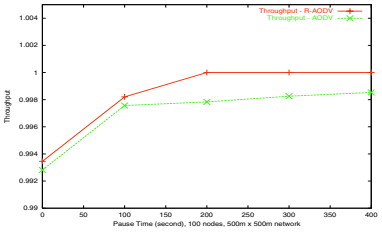

Fig. 3. Throughput – 500m x 500m

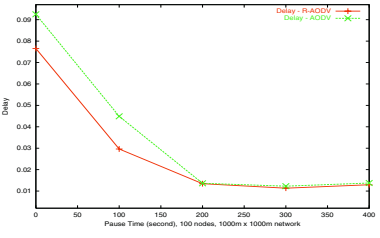

Fig. 4. Delay – 500m x 500m

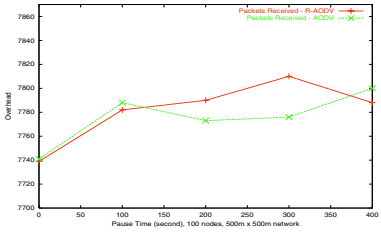

Fig. 5. Overhead – 500m x 500m

performance is observed when the number of nodes is set to 35, as illustrated in Figure 6. In our experiments, we have noticed that this type of behavior is always true with only 5 traffic pairs as well.

In our next set of experiments, we wish to study the effect of the number of traffic on the performance of of P_rAODV. Based upon our initial experiments, (see above), we choose a scenarios with 35 nodes and 5 traffic pairs. We also set the pause time at 600 seconds. Our results are as indicated in Figure 7. In both protocols, we observe an increase of the the number of the packet arrivals as we increase amount of traffic. The throughput obtained with P_rAODV is lower when compared to the AODV protocol, and the delay obtained with the preemptive AODV protocol is lower when compared to the original AODV. This is mainly due the preemptive paradigm used in P_rAODV and the number of packets that are dropped during the simulation.

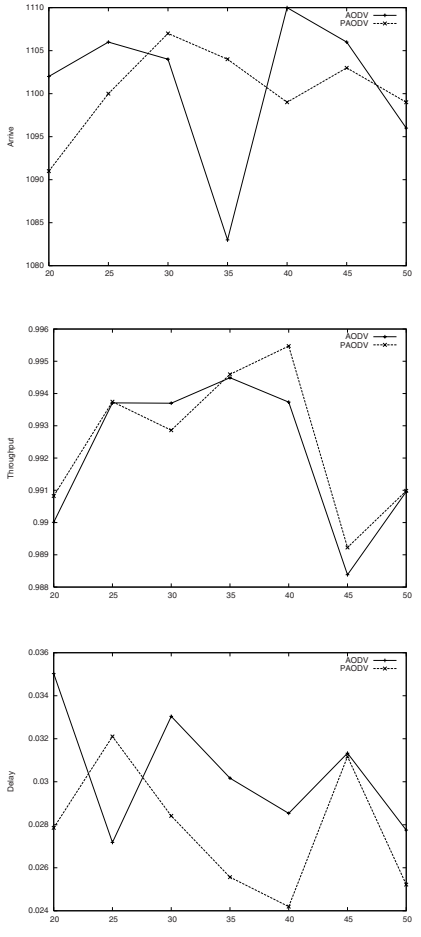

Fig. 6. Effect of the number of nodes

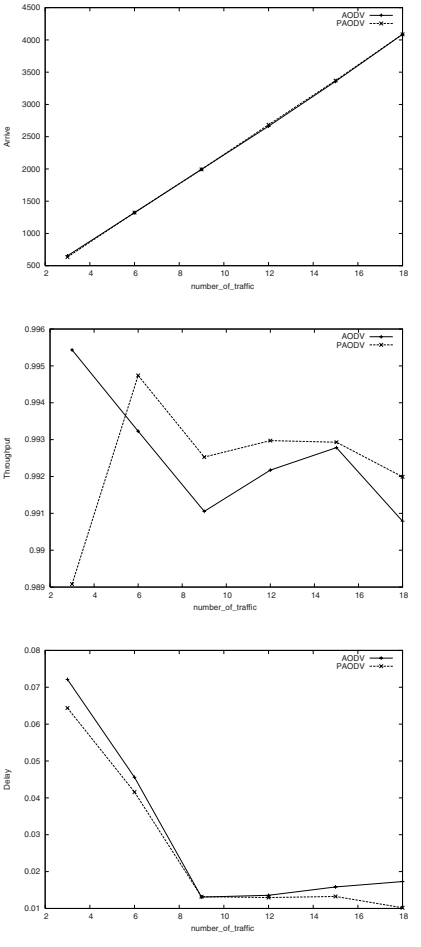

Fig. 7. The effect of the amount of traffic pairs

6 Conclusion

In this paper, we have studied the congestion problem in mobile and wireless networks. and we have presented two congestion control mechanisms for AODV ad hoc routing protocol, which we refer to as, *R-AODV*, a Randomized AODV protocol, and $P_r AODV$, an extension to the AODV routing protocol using a preemptive mechanism. We have discussed both algorithms and their implementations. We have also reported a set of simulation experiments to evaluate their performance when compared to the original AODV on-demand routing protocol.

Our results indicate that both *R-AODV* and $P_r AODV$ protocols are easy to scale, the number of mobile nodes does not affect their performance. and the traffic is quite well balanced among the availiable paths in the network. While our results indicate that both protocols increase the number of arrival packets within

the same simulation time, they decrease the delay per packet when compared to the original AODV scheme.

References

1. Preemptive Routing in Ad Hoc Network URL: http://opal.cs.binghamton.edu/ nael/research/papers/mobicom01.pdf
2. A. Boukerche, "Simulation Based Comparative Study of Ad Hoc Routing Protocols" *34th IEEE/ACM/SCS Annual Simulation Symposium*, April 2001, pp. 85–92.
3. A. Boukerche, S. K. Das, and A. Fabbri "Analysis of Randomized Congestion Control with DSDV Routing in Ad Hoc Wireless Networks" *Journal of Parallel and Distributed Computing*, pp. 967–995, Vo. 61, 2001.
4. D.B. Johnson, D.A. Maltz, Dynamic Source Routing in Ad Hoc Wireless Networks, in: *Mobile Computing*, Editors: T. Imielinski and H.F. Korth (Kluwer Academic Pub. 1996), Ch.5, 153–181.
5. C.E. Perkins, Ad Hoc On Demand Distance Vector (AODV) Routing, *IEFT Internet Draft*, available at: `http://www.ieft.org/internet-drafts/draft-ietf-manet-aodv-02.txt` (November 1998).
6. C.E. Perkins, P. Bhagwat, Highly Dynamic Destination-Sequenced Distance Vector Routing (DSDV) for Mobile Computers, *Proceedings of ACM SIGCOMM'94*, (1994), 234–244.