THE EFFECTS OF CACHE ORGANIZATIONS ON THE PERFORMANCE OF ON-DEMAND ROUTING PROTOCOLS IN AD HOC NETWORKS

Wenjing Lou and Yuguang Fang

Wireless Networks Laboratory (WINET)
Department of Electrical and Computer Engineering
University of Florida
Gainesville, FL 32611

Email: wjlou@ufl.edu , fang@ece.ufl.edu

ABSTRACT

Route caching strategy is important in an on-demand routing protocol in wireless mobile ad hoc networks. While high routing overhead usually has a significant performance impact in low bandwidth wireless network, a good route caching strategy can reduce routing overhead by making use of the available route information more efficiently. In this paper, we study the effects of two cache organizations, "link cache" and "path cache", on the performance of on-demand routing protocols through simulation. The effect of a static link timeout mechanism is studied and an adaptive link timeout mechanism is proposed. The adaptive timeout mechanism aims to keep tracking of the "optimal" link lifetime under varied node mobility levels by dynamically adapting the estimated link lifetime using a moving average from the real link lifetime statistics. We present the simulation results in terms of routing overhead, packet delivery ratio, and end-to-end delay. The results indicate that without an appropriate stale link removal mechanism, a link cache organization may suffer severe performance degradation because of the large number of route error messages generated. However, with an appropriate timeout mechanism, the link cache organization reduces the routing overhead significantly and outperforms the path cache when the network traffic load is high. Our simulation is based on the Dynamic Source Routing (DSR) protocol.

I. INTRODUCTION

An ad hoc network is an infrastructureless multi-hop mobile wireless network. Its self-configuring, self-organizing nature, and its capability to be promptly deployed without any wired base stations or infrastructure support have made ad hoc network very attractive in tactical and military applications, where fixed infrastructures are not available or reliable, and fast network establishment and self-reconfiguration are required. Examples include the tactical communication in a battlefield and the disaster rescue after an earthquake.

Due to the independent moving of the nodes, the network topology of an ad hoc network can be changed dramatically. Traditional Internet routing protocols are no longer effective in ad hoc networks. It presents a great challenge for a routing protocol to keep up with the frequent and unpredictable topology changes. Much work has been done since the mobile ad hoc networking (MANET) working group was formed within the Internet Engineering Task Force (IETF) to develop a routing framework for IP-based protocols in ad hoc network. Existing ad hoc routing protocol can be generally categorized into two classes: table-driven (or proactive) and demand-driven (reactive) [1]. Most of the performance studies indicate that on-demand routing protocols perform better than table-driven routing protocols [2][3][4]. The major advantage of the on-demand routing comes from the reduction of the routing overhead, as high routing overhead usually has a significant performance impact in low bandwidth wireless network.

However, on-demand routing also has its disadvantages. First of all, since a route is discovered on a needed basis, the packet cannot be sent before such a route has been found. Since the source node has no idea about where the destination is, the protocol has to search the entire network to find the destination. This is a very costly operation. It also adds the latency to the packet delivered. Secondly, in order to avoid the need to rediscover each routing decision for each individual packet, any on-demand routing protocol must utilizes some type of routing cache to cache the routes previously discovered. However, due to the frequent topology changes and lacking of timely update mechanisms on a regular basis, the cache itself may contain out-of-date information indicating links that are actually no longer exist. This stale data represents a liability that may degrade performance rather than improve

This work was supported in part by the Office of Naval Research Young Investigator Award under grant N000140210464 and the Office of Naval Research under grant N000140210554.

0-7803-7625-0/02/$17.00 ©2002 IEEE.

it [5]. There is a tradeoff between using the cached information -- valid data may reduce route discovery cost which will contribute to the improved performance while stale data will cause more errors, longer delay and higher packet loss.

In this paper, we study the effects of different cache organizations on the performance of on-demand routing protocol in an ad hoc network. Two types of the cache organizations, a path cache and a link cache, are studied. A path cache is one in which each cache entry represents an entire path from source to destination, while in a link cache each individual link is referred to as a cache data unit. A link cache has the potential to utilize the obtained route information more efficiently. However, without a good link update mechanism, stale links may cause more route errors. Marina and Das [6] studied a few techniques to improve cache correctness in DSR. Their study was based on a path cache structure. Hu and Johnson [7] have conducted a comprehensive study on the caching strategies in on-demand routing protocols for wireless ad hoc networks. However, their study on the timeout mechanisms is limited to a fixed level of node mobility. A static "optimal" lifetime is assigned to each node with this mobility level. Their adaptive timeout mechanism is also limited to fine tune-up of the lifetime within the same level of node mobility. In this paper, we study two types of link update mechanisms, a static timeout mechanism and an adaptive timeout mechanism. The static timeout mechanism expires a link using a statically assigned lifetime, while in the adaptive timeout mechanism, we propose to adapt the link timeout interval to various node mobility levels based on the estimation from a moving average of real link lifetime statistics. The performance of different cache organizations is evaluated through simulations based on the Dynamic Source Routing (DSR) protocol [8], since it is a well performed and entirely on-demand routing protocol.

II. METHODOLOGY

A. DSR Protocol and Cache Organizations

DSR is an on-demand routing protocol that is based on the concept of source routing. The mechanisms and operation of DSR are well defined in [8]. A slight modification in our simulation is that the destination node will reply to all the route requests received rather than only reply to the first one. Actually this is done in most simulation implementations as DSR is capable of caching multiple paths to a certain destination and the replies from the destination most accurately reflect the up-to-date network topology.

Besides the basic functions, more optimization mechanisms are proposed and added to DSR protocol. These optimizations include gratuitous route replies, salvaging, gratuitous route errors, snooping, tapping, etc. [5]. Most of them have been included in our simulation implementation.

In DSR, the route returned to the source is a complete path leading to the destination. By caching each of these paths separately, a "path cache" organization can be formed. A path cache is very simple to implement. When a route is needed, the path cache data structure can be efficiently searched for any path leading to that destination. This type of cache organization has been extensively studied through simulations [2][9]. In this paper, we consider an alternative type of organization, a "link cache", in which each node keeps a unified graph data structure representing this node's current view of the network topology, the route returned to this node is decomposed into individual links and represented in the graph data structure [7]. When a route is needed, a graph search algorithm, such as the Dijkstra shortest path algorithm or the breath-first-search (BFS) shortest path algorithm, is executed to find a path to the destination.

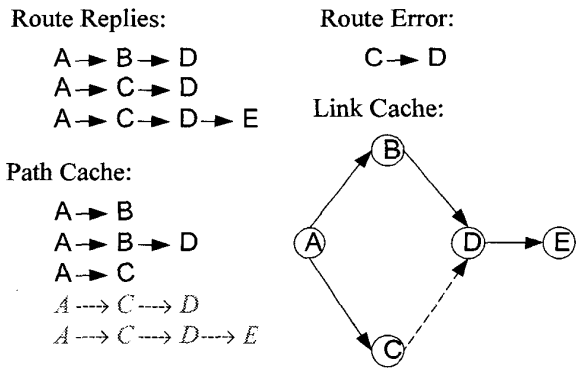


Figure 1 Illustration of path cache and link cache

Compared with a path cache organization, a link cache has the potential to utilize the route information more efficiently. Given the same amount of route reply information, the routes existing in a path cache can always be found in a link cache, while the link cache may have new routes that a path cache does not include by connecting individual links differently. In a path cache, a complete route will be removed (or is truncated before the broken link, depending on the path representation in the cache) when one of its links broke. While in a link cache, only the broken link is removed. The rest of the links on

that route are still available to form new routes. Consider the situation illustrated in Figure 1, when link C->D is broken, there will be no route to destination E exist in a path cache. A route discovery process has to be initiated if a route to destination E is desired. While by removing only link C->D, a link cache still has a route A->B->D->E to the destination E. A route discovery can be avoided. Therefore, with a link cache, potential reduction in the costly route discovery operation can be expected.

B. Link Timeout Mechanisms

As nodes in an ad hoc network are capable of moving, a links has a limited lifetime. Current existing links may no longer be valid when the two end nodes moving out of the transmission range of each other. Due to the on-demand manner of the routing protocol, the link status will not be updated until they are used. However, using an actually broken route will cause a number of route errors to be generated and potential packet loss. So taking advantage of using active links individually has to be combined with a mechanism that removing stale links timely to avoid route errors. A natural choice is to combine a timeout policy to the link cache such that each link is given an appropriate lifetime when it enters the link cache, and is removed when its lifetime runs out. The lifetime estimation becomes a critical issue for such a link cache organization. The lifetime assigned to the link should properly reflect the expected value of its real lifetime. If this value is assigned too small, links expire too quickly before they really break, more costly route discoveries would have to be performed. On the other hand, if the value is too large, links break early before the timers expire, more route errors will be caused, which would degrade the overall performance of the protocol.

In this paper, we first study the static lifetime assignment, in which when a link enters the link cache, it is assigned a predefined static lifetime *T1* seconds. During its lifetime, if the link is used to send packets, the lifetime of this link will be adjusted so that it won't expire in the future *T2* seconds. Then based on the observations from the static lifetime experiments, we propose and study an adaptive lifetime estimation scheme that adaptively estimate the link lifetime based on the moving average of the pervious collected lifetime statistics.

In our adaptive link lifetime scheme, each link (i,j) in the link cache is associated with three clock type attributes, *born*, *lastUsed*, and *liveTo*. Attribute *born* indicates the time when a new link enters the link cache. It is updated when a route reply is received and a new link is found in the route. Attribute *lastUsed* is the time stamp when the link is last used to forward a packet. Attribute *liveTo* is the predicted time at or after which the link expires. The statistical lifetime data are collected whenever a link is removed from the link cache. If it is removed because of the reception of a route error, the lifetime l is calculated as

$$l = CurrentTime() - link[i,j].born$$

or if it is removed because of timeout, lifetime l is calculated only if this link has ever been used during its lifetime,

$$l = link[i,j].lastUsed - link[i,j].born$$

LIFETIME is the variable indicating the estimation of the link lifetime. It is initially assigned a static value and is adjusted dynamically using a moving average method whenever a lifetime datum l is collected,

$$LIFETIME = (1-\alpha) * LIFETIME + \alpha * l$$

where α is the 1st order moving average parameter.

III. SIMULATION AND RESULTS

A. Simulation Framework

The simulation of our link cache DSR protocol is implemented within the GloMoSim library [10]. The GloMoSim library is a scalable simulation environment for wireless network systems using the parallel discrete-event simulation language called PARSEC. The link layer model is the Distributed Coordination Function (DCF) of the IEEE 802.11 wireless LAN standard. The radio model uses the channel characteristics similar to Lucent's WaveLAN product. Radio propagation range for each node is 250 meters and channel capacity is 2 Mbits/sec. The network simulated consists of 50 nodes. The simulation area is 700x700 square meters. Each simulation is executed for 15 simulated minutes.

The random waypoint mobility model is used in the simulation [2]. In this model, a node selects a destination randomly within the simulated territory, moves to that destination at a speed uniformly distributed between 0 to 20 m/sec, stops there for a predefined pause time and then repeats this behavior for the duration of the simulation. The different node mobility levels are achieved by changing the values of pause time.

The traffic used in this simulation is constant bit rate (CBR) UDP sessions. The source-destination pairs are chosen randomly among the 50 nodes. The number of communication sessions is varied to change the offered load in the network. All the data packets are 64 bytes and are sent at a speed of 4 packets/sec. The reason that we choose 64 bytes small packet size is because our focus is on routing protocol's capability of tracking the topology change. Small packet size will factor out the effects of

other reasons such as the network congestion [2].

We evaluate the performance of the link cache DSR protocol with static lifetime assignments and the proposed adaptive lifetime scheme. With static lifetime assignments, we execute multiple simulations for different (*T1,T2*) values as (3,1), (6,1), (12,2), (25,3), (50,5), (100,10), (900,-). For each (*T1,T2*) value, we execute multiple simulation runs with various traffic load conditions (10,20,30,40,50 sources) and various node mobility levels (pause time = 0s, 30s, 60s, 120s, 240s, 480s, 900s). For comparison purpose, we also evaluate the conventional path cache DSR protocol. The traffic load and the node mobility scenarios are identical across different variations of DSR protocol.

B. Simulation Results

We first examine the effects of different cache schemes on the routing overhead generated. The routing overhead is calculated as the number of control packets transmitted by the protocol. It is counted as per hop basis. As shown in Figure 2 and 3, the results confirm our expectation that small values of lifetime cause increased number of route requests but decreased number of route errors, while large values of lifetime cause decreased number of route requests but increased number of route errors. When the lifetime is within an appropriate range (in our experiments, 6<=T1<=50), the overall control messages are quite balanced. However, when the lifetime value becomes very large (in our experiments, T1>=50), the dramatically increased route errors would overwhelm the slightly decreased route requests. The total number of control messages increases significantly. In general, with an appropriate static link lifetime, the number of control packets used by the link cache scheme is less than that used by the path cache scheme. With the increased network traffic load, the reduction becomes more obvious. We also observe that the proposed adaptive cache scheme tracks the "optimal" link lifetime quite well. Although it is not always optimal, it keeps the routing overhead "sub-optimally" low under various mobility level conditions. The reduced routing overhead, especially in heavy traffic situation, will contribute to the improved performance.

Next we evaluate the application level performance metric – packet delivery ratio (the end-to-end throughput). The packet delivery ratio is the fraction of packets that are received at corresponding destination over those sent at the source. There are two major situations that a packet may drop. One is due to the stale routes. A stale route in the cache may direct a packet to an actually broken link, which may cause the packet be dropped. To reduce such packet drops, small lifetime value is preferred because small lifetime value could expire stale routes and reduce the chances that a stale route is used. The other type of packet loss is due to the heavy collisions in the MAC layer that cause a packet drop after failing a certain number of attempts to transmit the packet. A route discovery process can cause a large number of route requests and route replies generated within a short time, thus cause increased interference to data traffic at MAC layer. When the traffic load is high, the packet loss caused by collisions becomes more severe. To reduce the packet loss due to this reason, a large lifetime value is preferred because a large lifetime value could minimize the number of route discovery performed. Figure 4 summarizes the performance of packet delivery ratio under various network traffic loads and node mobility levels. We observe that, in general, the link cache scheme outperforms the path cache scheme when the network load is high. Again, the heavier the load, the wider the performance gap. However, when the traffic load is low, the performance comparison between the two types is not certain.

Packet latency is another important performance metric. The packet latency is also called end-to-end delay, which is the latency between a packet sent at the source and received at the destination. The packet latency is only calculated for packets that are successfully delivered. Besides the ordinary transmission delay, propagation delay, and queuing delay, which widely exist in all IP networks, there are two types of latency caused particularly by ad hoc on-demand routing protocols. One is the latency the protocol takes to discover a route to a destination when there is no known route to that destination. This type of latency is due to the on-demand behavior of the routing protocol and exists in all such protocols. The other one is the latency for a sender to "recover" when a route used breaks. The latency resulting from broken routes could be very large because the amount of latency is the addition of the following three parts, the time for a packet travel along the route to the node immediately before the broken link, the time for that node to detect the broken link, and the time for a route error message to travel from that node back to the source node. Among them, the time to detect a broken link could be very large because the failure of the node can only be determined after having made a certain number of attempts to transmit the packet over the broken link but failed to receive a passive or explicit acknowledgement of success. Figure 5 shows the performance comparison of packet latency across different variations of DSR protocol. We observe that, with an appropriate link lifetime, the packet latency incurred by the link cache scheme is kept relatively low compared to the path cache scheme. Especially when the network traffic load grows high, the advantage of the link cache becomes clearer. Since the small link lifetime

value tends to expire links earlier than they actually break, the possibility of using broken links are reduced, thus the latency caused by broken routes is minimized. Therefore, we observe that small lifetime values result in low packet latency. Correspondingly, without an appropriate timeout mechanism (e.g. the link lifetime assigned is too large), packets suffer abnormally large latency because too many broken routes are used. Again we observe that the proposed adaptive cache scheme maintains comparably low latency under various mobility level conditions.

IV. CONCLUSIONS AND ONGOING RESEARCH

In this paper, we study the effects of the route cache organizations on the performance of on-demand routing protocols in ad hoc networks. We base our simulation on DSR, the well-evaluated on-demand routing protocol in ad hoc networks. We first study the link cache performance with static lifetime assignments. The results indicate that an appropriate timeout mechanism is critical on the performance of such a link cache organization. A link cache with an appropriate timeout mechanism could make use of the available route information more efficiently, thus improve the protocol performance. However, without an appropriate timeout mechanism, a link cache may cause dramatically increased route errors and consequent performance degradation. We also found that the link cache organization performs better when network traffic load is high. Based on these observations, we propose an adaptive link timeout mechanism. The adaptive link lifetime estimation scheme aims at tracking the "optimal" link lifetime under varied node mobility conditions. The performance of the proposed strategy is compared with the conventional "path cache" DSR. The results show that when the number of traffic sources increases, the proposed adaptive link cache DSR outperforms the path cache DSR, with wider performance gap with increasing load. As the cache organization is a local implementation decision at each node, all the protocol control messages for route discovery and maintenance mechanisms remain the same, we suggest that switching between the two types of cache organizations dynamically in response to the network load condition will provide a good way to improve the overall performance of the DSR protocol.

The simulation results shown in this paper are just the preliminary results of out study. More systematic study on the lifetime statistics and finer tune-up of the parameters are still undertaken.

REFERENCES

[1] E. M. Royer, C-K Toh, A review of current routing protocols for ad hoc mobile wireless networks, *IEEE Personal Communications*, pp.46-55, April 1999

[2] J. Broch, D. Maltz, D. Johnson, Y-C. Hu, J. Jetcheva, A performance comparison of multi-hop wireless ad hoc network routing protocol, *The 4th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, pp. 85-97, Oct 1998

[3] P. Johansson, T. Larsson, N. Hedman, B. Mielczarek, M. Degermark, Scenario-based performance analysis of routing protocols for mobile ad hoc networks, *The 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, pp.195-206, Aug 1999

[4] S. R. Das, etc., Comparative performance evaluation of routing protocols for mobile ad hoc networks, *The 7th International Conference on Computer Communication and Networks (IC3N)*, pp. 153-161, Oct 1998

[5] D. A. Maltz, J. Broch, J. Jetcheva, D. B. Johnson, The effects of on-demand behavior in routing protocols for multihop wireless ad hoc networks, *IEEE Journal on Selected Areas in Communications*, vol.17, no.8, pp.1439-1453, Aug 1999

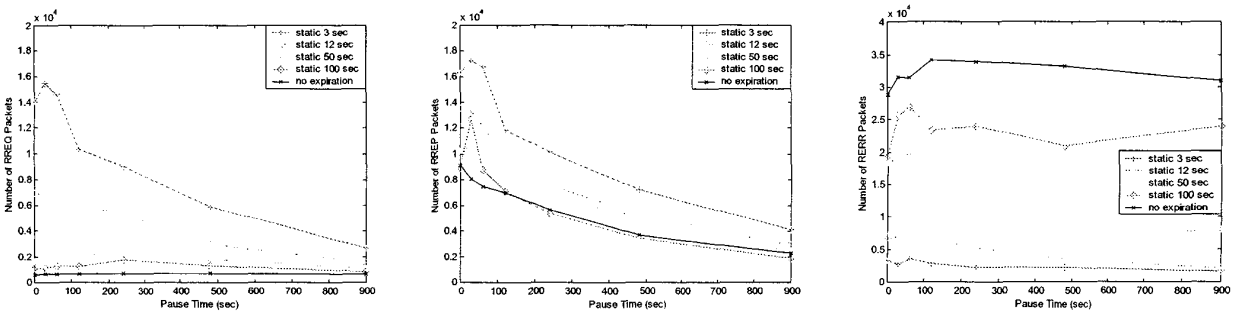
[6] M. K. Marina, S. R. Das, "Performance of routing caching strategies in dynamic source routing", *2001 International conference on distributed computing systems workshop*, 2001

[7] Y-C. Hu, D. B. Johnson, Caching strategies in on-demand routing protocols for wireless ad hoc networks, The 6th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom), Aug 2000

[8] D. B. Johnson, D. A. Maltz, Y-C. Hu, J. G. Jetcheva, The dynamic source routing protocol for mobile ad hoc networks, *IETF Internet Draft*, draft-ietf-manet-dsr-05.txt, Mar 2001

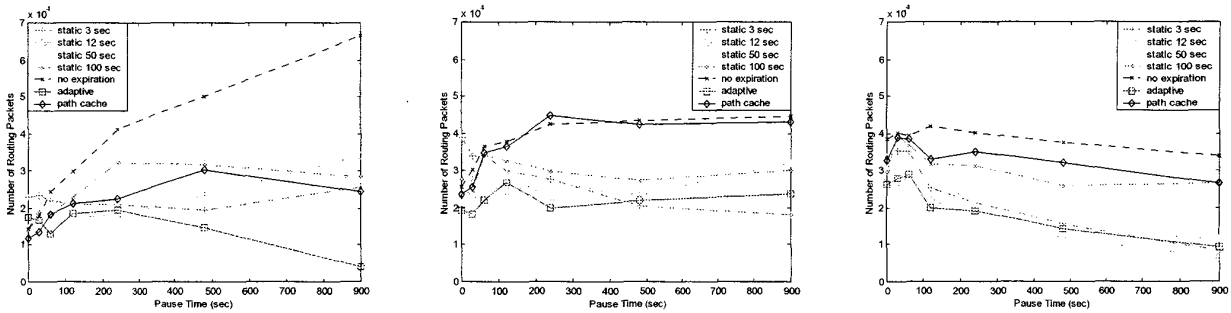
[9] C. E. Perkins, E. M. Royer, S. R. Das, M. K. Marina, Performance comparison of two on-demand routing protocols for ad hoc networks, *IEEE Personal Communications*, pp.16-28, Feb 2001

[10] M. Takai, L. Bajaj, R.Ahuja, R. Bagrodia, M. Gerla, GloMoSim: a scalable network simulation environment, Technical Report 990027, UCLA, Computer science department, 1999



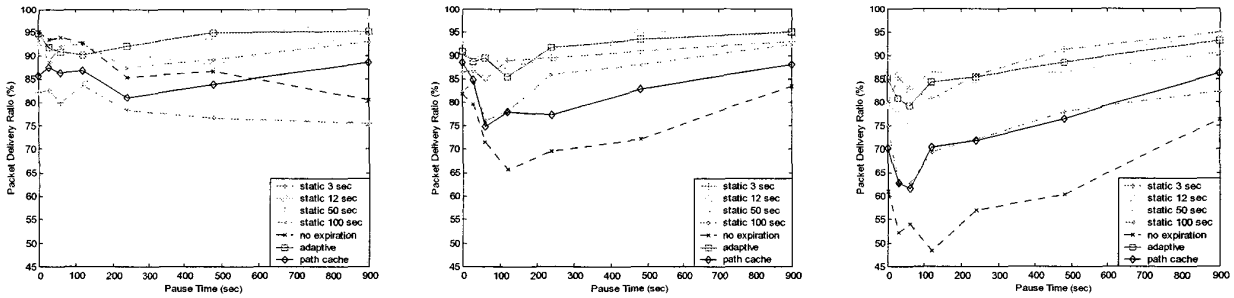
(a) Route Request (b) Route Reply (c) Route Error

Figure 2. Routing Overhead (Separated, 50 sources case)



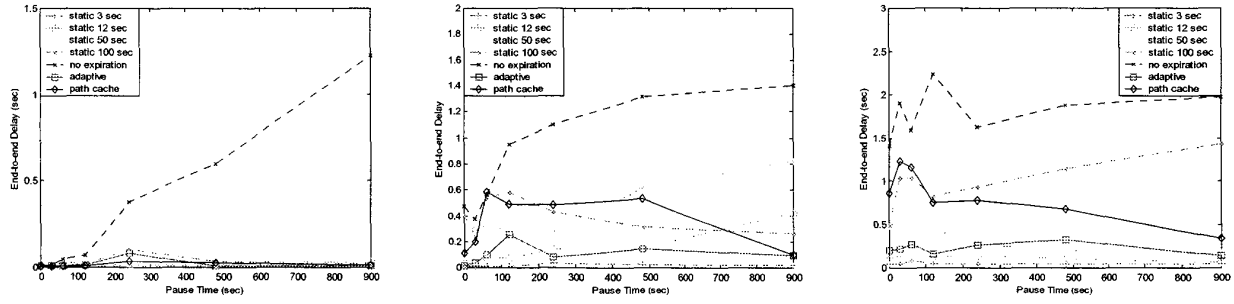
(a) 20 sources (b) 40 sources (c) 50 sources

Figure 3. Routing Overhead (Total)



(a) 20 sources (b) 40 sources (c) 50 sources

Figure 4. Packet Delivery Ratio



(a) 20 sources (b) 40 sources (c) 50 sources

Figure 5. End-to-end Delay