Scenario-based comparison of Source-Tracing and Dynamic Source Routing Protocols for Ad-Hoc Networks

Jyoti Raju
jyoti@cse.ucsc.edu
Computer Science Department
University of California
Santa Cruz, CA 95064

J.J. Garcia-Luna-Aceves
jj@cse.ucsc.edu
Computer Engineering Department
University of California
Santa Cruz, CA 95064

Abstract—**We present source tracing as a new viable approach to routing in ad hoc networks where routers communicate the second-to-last hop and distance in preferred paths to destinations. We use two source tracing algorithms, a table-driven protocol (BEST) in which routers maintain routing information for all destinations, and an on-demand routing protocol (DST) in which routers maintain routing information for only those destinations to whom they need to forward data. Simulation experiments are used to compare these protocols with DSR, which has been shown to incur less control overhead than other on-demand routing protocols. The simulations show that DST requires far less control packets to achieve comparable or better average delays and percentage of packet delivered than DSR, and that BEST achieves comparable results to DSR while maintaining routing information for all destinations.**

I. Introduction

Ad-hoc networks (or multi-hop packet-radio networks) consist of mobile routers interconnecting hosts. The deployment of such routers is ad-hoc and the topology of the network is very dynamic, because of host and router mobility, signal loss and interference, and power outages. Furthermore, the bandwidth available for the exchange of routing information in ad-hoc networks is far lesser than the bandwidth available in a wired internet.

Routing for ad-hoc networks can be classified into two main types: *table-driven* and *on-demand*. Table driven routing attempts to maintain consistent information about the path from each node to every other node in the network. The Destination-Sequenced Distance-Vector Routing (DSDV) protocol is a table driven algorithm that modifies the distributed Bellman-Ford routing algorithm to include timestamps that prevent loop-formation [13]. The Wireless Routing Protocol (WRP) is a distance vector routing protocol which belongs to the class of path-finding algorithms that exchange second-to-last hop ($predecessor$) to destinations in addition to distances to destinations [11]. This extra information helps remove the "counting-to-infinity" problem that most distance vector routing algorithms suffer from [1]. It also speeds up route convergence when a link failure occurs.

On-demand routing protocols have been designed to limit the amount of bandwidth consumed in maintaining up-to-date routes to all destinations in a network by maintaining routes to only those destinations to which the routers need to forward data traffic. The basic approach consists of allowing a router that does not know how to reach a destination to send a flood-search message to obtain the path information it needs. There are several recent examples of this approach (e.g., AODV [14], ABR [17], DSR [10], TORA [12], SSA [5]) and the routing protocols differ on the specific mechanisms used to disseminate flood-search packets and their responses, cache the information heard from other nodes' searches, determine the cost of a link, and determine the existence of a neighbor. However, all the on-demand routing proposals to date use flood search messages that either: (a) give sources the entire paths to destinations, which are then used in source-routed data packets (e.g., DSR); or (b) provide only the distances and next hops to destinations, validating them with sequence numbers (e.g., AODV) or time stamps (e.g., TORA). One problem with source routing is that it results in long data-packet headers as the network size increases; in addition, source routing will not work with security schemes that encrypt headers. Protocols using sequence numbers and timestamps suffer from inefficiency when nodes fail and lose state, as can happen with high probability in ad-hoc networks.

This work was supported in part by the Defense Advanced Research Projects Agency (DARPA) under grant F30602-97-2-0338.

In this paper, we analyze two source routing protocols for ad-hoc networks that use predecessor and distance information. The first protocol is dynamic source tree (DST) protocol [16], which is an on-demand routing protocol. DST acquires routes to destinations only when traffic for those destinations arrives and there is no known route to the destination. The acquired route does not have to be the shortest path; it has to be valid and of finite metric value. DST does not use source-routed packets or time stamps to validate distance updates. DST uses a source-tracing algorithm similar to the one advocated in prior table-driven routing protocols in which routers maintain routing information for all network destinations [11], [1]. To reduce the number of loops, the source-tracing algorithm allows for complete paths to be checked for loops starting from the sources (hence the name source-tracing). This source tracing is facilitated by maintaining distance and predecessor of the shortest path to all known destinations, which in turn eliminates the counting to infinity problem of the distributed Bellman-Ford algorithm.

The second protocol, Bandwidth Efficient Source Tracing (BEST) protocol [15] is based on source-tracing and is an extension of WRP[11]. It uses unreliable updates and introduces a conservative approach to table-driven routing, i.e., routers send updates only under conditions where routing table loops are suspected. Data packets in both DST and BEST contain only the source and destination addresses and not the entire source routes.

This paper examines the performance of DST, DSR and BEST in simulation scenarios that mimic real world scenarios and using these simulations to conclude that source-tracing can be the basis for a very efficient routing protocol that maintains routing information either on-demand or for all destinations.

Section II presents the network model used in DST and BEST. Section III and section IV give a brief description of DST and BEST, respectively. Section V uses simulations to compare the performance of DSR, DST and BEST using the same movement model used in [3], [4] to compare DSR with other on-demand and table-driven routing protocols.

II. Network Model

A network is modelled as an undirected graph with V nodes and E links. Instead of having interface identifiers, a router has a single node identifier, which helps the routing and other application protocols identify it. In a wireless network, a node has radio connectivity with multiple nodes using a single physical radio link. Accordingly, we map a physical broadcast link connecting a node and its multiple neighbors into point-to-point links between the node and its neighbors. Each link has a positive cost associated with it. If a link fails, its cost is set to infinity. A node failure is modelled as all links incident on the node getting set to infinity.

For the purpose of routing-table updating, a node A considers an-

Report Documentation Page	*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE **2001**	2. REPORT TYPE	3. DATES COVERED **00-00-2001 to 00-00-2001**

4. TITLE AND SUBTITLE **Scenario-based comparison of Source-Tracing and Dynamic Source Routing Protocols for Ad-Hoc Networks**	5a. CONTRACT NUMBER
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **University of California at Santa Cruz,Department of Computer Engineering,Santa Cruz,CA,95064**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**		**6**	

other node B as its neighbor if A receives an update from neighbor B. Node B is no longer node A's neighbor when the medium access protocol at node A sends a signal to DST indicating that data packets can no longer be sent successfully to node B.

Both the source-routing protocols are designed to run on top of any wireless medium-access protocol. Routing messages are broadcast unreliably and the protocol assumes that routing packets may be lost due to changes in link connectivity, fading or jamming. Since DST and BEST only require a MAC indication that data packets can no longer be sent to a neighbor, the need for a link-layer protocol for monitoring link connectivity with neighbors or transmitting reliable updates is eliminated, thus reducing control overhead. If such a layer can be provided with no extra MAC overhead, then DST and BEST can be made more proactive by identifying lost neighbors before data for them arrives, resulting in faster convergence and decreased data packets losses.

III. DST

A node running DST maintains shortest paths in its routing tables to all the destinations it has seen data packets for. Each entry of the routing table contains a destination identifier j, the distance to the destination D_j^i and the predecessor to that destination p_j^i. A node also maintains a *distance table* which contains the routing tables reported by all its known neighbors. A node uses the routing tables of known neighbors along with the link costs to known neighbors to generate its own routing table. A *query table* is maintained to keep track of the flood-search messages started and forwarded for all known destinations. Using information about the last time a flood search message was started and forwarded, a node can prevent broadcasts from continuing indefinitely by dropping queries that follow each other too closely in time. A *data buffer* is used to store packets waiting for routes. The packets are dropped after a timeout.

A routing message broadcasted by a node i contains a vector of entries where each entry corresponds to a route in the routing table; each entry contains a destination identifier j, the distance to the destination D_j^i and the predecessor to that destination p_j^i. DST uses two types of routing messages: $queries$ and $updates$. Like other on-demand routing protocols, DST starts a route discovery process when it has no path to a destination that it receives a data packet for. The route discovery process is started by broadcasting queries to neighbors. These queries are separated by a $query\_send\_timeout$ to prevent a flood of queries when a burst of data packets arrives at a node. A query is forwarded hop-by-hop for a maximum of MAX_HOPS hops or until it reaches a node that has a non-infinite loop-free path to the destination. A loop-free path is guaranteed at a node i by using the following two conditions when picking a neighbor k as the next hop towards a destination j.

1. k offers the shortest distance to all nodes in the path from j to i.
2. the path from j to k does not contain i and does not contain any repeated nodes.

The node that finds the required path sends back a reply update that is forwarded back hop-by-hop to the source of the query, thus creating a route to the destination in all the intermediate nodes it passes through. Reply updates may arrive from multiple nodes. To prevent reply update broadcast storms, an intermediate node does not forward a reply broadcast back towards the source, if the reply update does not decrease the intermediate node's distance to the destination. This procedure allows the source to get multiple disjoint paths to the destination but reduces the number of updates.

When a node i receives a data packet for a destination j it no longer has a route to, it sends updates to all its neighbors reflecting the new infinite distance to j. When a neighbor k gets the update, it processes the update. If the update implies a distance increase to a known destination, then a new update with the changed routing tables is rebroadcasted to the neighbor set. In this manner, the update eventually gets forwarded to the source of the data packets. The source then uses an alternate route, if it has any, or restarts a route discovery process.

Since DST assumes unreliable updates, it may be the case some updates are lost resulting in data packet looping. To prevent data packet looping, two more conditions are added to prevent data packet looping. A data packet is dropped and an update is sent if

A. The data packet is sent by a neighbor that is in the path from the present node to the destination of the data packet.

B. The path implied by the neighbor's distance table entry is different from the path implied in the routing table.

Permanent looping can occur when nodes are unaware of the latest changes in their neighbor's routing tables. The use of conditions A and B can be explained with the help of an example shown in Fig. 1.a. The node addresses are marked in bold font. Node j is the required destination. The path to j implied by traversing predecessors from j is marked in italics. Initially, all nodes have loop-free routes. The loss of links (i, j) and (m, j) and the loss of update packets from i and m can result in a loop shown in Fig. 1.b. When i gets a data packet from k, it finds that its distance table entry for k implies the path ij, while i's own path implies $ilmj$ which is different from ij. Therefore due to condition B, the data packet is dropped and a unicast routing update is sent resulting in k setting its path to kmj. Now, when k gets a data packet from m, it sends a unicast update to m because m is its successor on the path to j. This follows from condition A. When m gets the update, it detects a loop and resets its distance to infinity, thus breaking the loop.

IV. BEST

BEST has a routing table and distance table with the same functionality as in DST. BEST does not require a data buffer or a query table as it is a table-driven routing protocol; data packets are dropped if no path exists. The only type of packets used in BEST are updates that are unreliably broadcast and contain (distance,predecessor) tuples for all the destinations. BEST differs from WRP in allowing unreliable updates and in specifying different conditions to send updates. We focus our description of BEST on how updates are sent, because source tracing has been used extensively in the past in table-driven protocols [1], [11], [7]

The processing of an update in BEST is done in the same manner as in DST. When an update from neighbor k is received, the entries in the distance table corresponding to neighbor k are updated. The paths to each destination are then recomputed. BEST sends updates only if any of the following conditions have been met.

1. A node discovers a new destination with a finite and valid path to the destination.
2. A node loses the last path to a destination.
3. A node suffers a distance increase to a destination.

From the above conditions, it follows that an update is not sent if a next hop to destination changes. It is also not sent if the distance to a destination decreases. However, an update is sent when the distance to a destination increases, because this condition has the potential to cause a loop.

Conditions A and B used in DST are also incorporated in BEST to prevent data packet looping. These rules are much simpler than those introduced in STAR [8] which uses the link-state information in source trees, rather than distance and second-to-last-hop information to a destination in the tree.

V. Performance Evaluation

We ran simulations for two different experimental scenarios to compare the average performance of DST, DSR and BEST. These simulations allowed us to independently change input parameters and check the protocol's sensitivity to these parameters. All three protocols are implemented in CPT, which is a C++ based toolkit that provides a

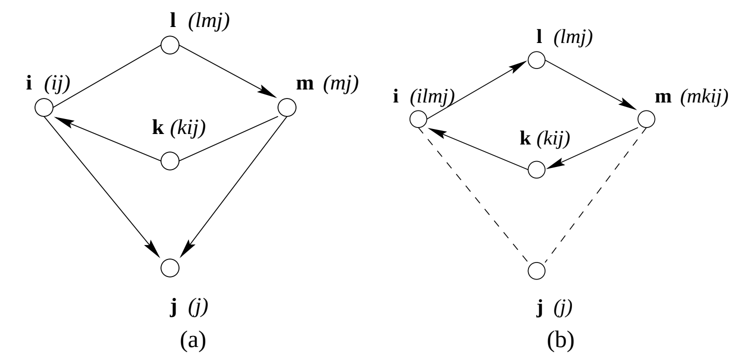


Fig. 1. Creation of a permanent loop in BEST due to unreliable updates

wireless protocol stack and extensive features for accurately simulating the physical aspects of a wireless multi-hop network. The protocol stack in the simulator can be transferred with a minimal amount of changes to a real embedded wireless router. The stack uses IP as the network protocol. The routing protocols directly use UDP to transfer packets. The link layer implements the IEEE 802.11 standard [2] and the physical layer is based on a direct sequence spread spectrum radio with a link bandwidth of 1 Mbit/sec.

To run DSR in CPT, we ported the DSR code available in the $ns2$ [6] wireless release. There are two differences in our DSR implementation as compared to the implementation used in [3]. Firstly, we do not use the $promiscuous$ listening mode in DSR. We, however, implement the promiscuous learning of source routes from data packets. This follows the specification given in the Internet Draft of DSR. Our reason for not allowing promiscuous listening is that, besides introducing security problems, it cannot be supported in any IP stack where the routing protocol is in the application layer and the MAC protocol uses multiple channels to transmit data. The second difference in our implementation is that since the routing protocol in our stack does not have access to the MAC and link queues, we cannot reschedule packets that have already been scheduled over a link (for either DSR, DST or BEST). Tables I and II show the constants used in the implementation of DSR and DST, respectively.

TABLE I
CONSTANTS USED IN DSR SIMULATION

Time between ROUTE REQUESTS (exponentially backed off)	500 msec
Size of source route header carrying carrying n addresses	$4n$+4 bytes
Timeout for Ring 0 search	30 msec
Time to hold packets awaiting routes	30 sec
Data buffer size	50 pkts

TABLE II
CONSTANTS USED IN DST SIMULATION

Query send timeout	5 sec
Time out for ring 0 search	30 msec
Data packet timeout	30 sec
Data buffer size	50
MAX_HOPS	17

A. Scenarios used in comparison

We compared DSR, DST and BEST using two types of scenarios. In both scenarios, we used the "random waypoint" model described in [3]. In this model, each node begins the simulation by remaining stationary for *pause time* seconds and then selects a random destination and moves to that destination at a speed of 20 m/s. Upon reaching the destination, the node pauses again for *pause time* seconds, selects another destination, and proceeds there as previously described, repeating this behavior for the duration of the simulation. We used the speed of 20m/s (72 km/hr), which is the speed of a vehicle, because it has been used in simulations in earlier papers [3], [4] and thus provides a basis for comparison with other protocols. All simulations are run for 900 seconds. In both scenarios, we used a 50 node ad-hoc network, moving over a flat space of dimensions 7 X 6 miles (11.2 X 9.7 km) and initially randomly distributed with a density of approximately one node per square mile.

Two nodes can hear each other if the attenuation value of the link between them is such that packets can be exchanged with a probability p, where $p > 0$. Attenuation values are recalculated every time a node moves. Using our attenuation calculations, radios have a range of approximately 4 miles (135 db).

B. Metrics used

In comparing the protocols, we used the following metrics:

- *Packet delivery ratio*: The ratio between the number of packets received by an application and the number of packets sent out by the corresponding peer application at the sender.
- *Control Packet Overhead*: The total number of routing packets sent out during the simulation. Each broadcast packet/unicast packet is counted as a single packet.
- *Hop Count*: The number of hops a data packet took from the sender to the receiver.
- *End to End Delay*: The delay a packet suffers from leaving the sender application to arriving at the receiver application. Since dropped packets are not considered, this metric should be taken in context with the metric of packet delivery ratio.

Packet delivery ratio gives us an idea about the effect of routing policy on the throughput that a network can support. It also is a reflection of the correctness of a protocol.

Control packet overhead has an effect on the congestion seen in the network and also helps evaluate the efficiency of a protocol. Low control packet overhead is desirable in low-bandwidth environments and environments where battery power is an issue.

In ad-hoc networks it is sometimes desirable to reduce the transmitting power to prevent collisions. This will result in packets taking more number of hops to reach destinations. However, if the power is kept constant, the distribution of the number of hops data packets travel through is a good measure of routing protocol efficiency.

Average end-to-end delay is not an adequate reflection of the delays suffered by data packets. A few data packets with high delays may skew results. Therefore, we plot the cumulative distribution function of the delays. This plot gives us a clear understanding of the delays suffered by the bulk of the data packets. Delay also has an effect on the throughput seen by reliable transport protocols like TCP.

C. Performance results

C.1 Scenario 1

Scenario 1 mimics the behavior of an emergency network or a network set up for military purposes. Scenario 1 is almost identical to to the one presented in [3], barring any differences due to implementation of the MAC protocols.

We have 20 random data flows, where each flow is a peer-to-peer constant bit rate (CBR) flow with a randomly picked destination and the data packet size is kept constant at 64 bytes. Data flows were started at times uniformly distributed between 20 and 120 seconds and they go on till the end of the simulation at 900 seconds. We did 7 runs of the simulation where each run had different sets of source-destination pairs. The total load on the network is kept constant at 80 data packets per second (40.96 kbps) to reduce congestion. Our rationale for doing this is that increasing the packet rate of each data flow does not test the routing protocol. On the other hand, having flows with varying destinations does so. We also vary the pause times: 0, 30, 60, 120, 300, 600 and 900 seconds as done in [3].

Fig. 2.a shows the control packet overhead for varying pause times. An obvious result is that the control packet overhead for all the three protocols reduces as the pause time increases. BEST and DST are about 34 % better than DSR at pause time zero. At low rates of movement, DST is a clear winner with one third the control packet overhead of BEST and one tenth the control packet overhead of DSR. Clearly, the fact that the updates in DST contain the entire routing table, means that nodes running DST have a higher chance of knowing paths to destinations for whom no route discovery has been performed in the past. We are able to mimic the behavior of table-driven routing protocols in low topology change scenarios, in that we almost have information about the entire topology with very few flood searches.

As shown in Fig. 2.b, the percentage of data packets delivered is almost the same for DST and BEST. At lower pause times, DSR has the same packet delivery ratio as DST and BEST. However, as the pause time decreases, DSR suffers due to data packets getting dropped at the link layer, indicating that the routes provided in the source routes are not correct any more. At lower pause times, links get broken faster. Even though this results in higher control overhead, the routes obtained are relatively new. As mentioned earlier, we keep the load on the network constant. Since this load is divided among a large number of flows, we see very little congestion and therefore most packets get through at higher pause times during which the topology is close to static.

For Fig. 2.c we collated the hop count values for data packets during all pause times and plotted the hop distribution. All three protocols have almost the same number of one hop packets, indicating that the zero hop query is very effective in getting routes to neighbors. However, for the number of hops greater than one, we see that BEST performs the best. This is expected of a table driven routing protocol that tries to maintain valid routes at most times. DST's behavior is slightly worse than BEST. DSR on the other hand sends packets through longer routes. This is a direct consequence of the fact that after the initial query-reply process DSR pretty much uses the route it caches, without trying to better them.

Fig. 2.d shows the cumulative delay of all the protocols. The graphs shown are logarithmic in time to accommodate the wide variation. We see that BEST performs better than DSR or DST, with DST being very close. Almost all packets are sent within 4 seconds in BEST and within 8 seconds in DST. Some packets in DSR take almost 30 seconds. This is because a packet is allowed to stay in a buffer for a maximum of 30 seconds before it is dropped. These are packets that found the path just in time.

C.2 Scenario 2

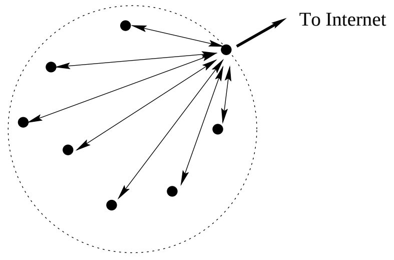


Fig. 3. Scenario 2

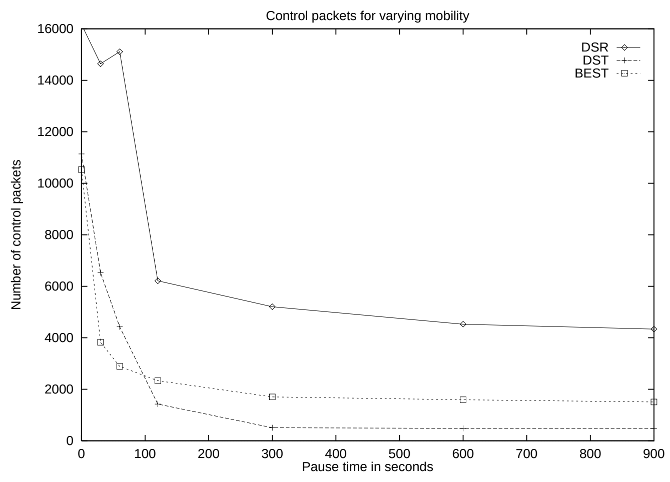
Scenario 2 mimics the applications of ad-hoc networks as wireless extensions to the Internet. In this case, one or two nodes act as points of attachment of the ad-hoc network to the Internet. Accordingly, all Internet traffic travels to and from the attachment points as shown in Fig. 3. To model this situation, we pick one node as the point-of-attachment to the Internet for a simulation run of 900 seconds and we do five such runs and plot our results. During each run, the sender node first establishes a low rate connection (5.85 kbps) with the point-of-attachment. Immediately after the forward connection is established, the backward connection is started from the point-of-attachment to the sender. This connection has a higher rate of 40.96 kbps. Each pair of connections lasts for 300 seconds. In each epoch of 300 seconds, we start seven pairs at random times. This setup closely resembles number of nodes accessing the Web through the point-of-attachment. We run our simulations for two pause times, 0 (continuous movement) and 900 (no movement).

Fig. 4.a and Fig. 4.b show the results for the case of continuous movement. We see that BEST has almost double the control packet overhead of DST or DSR. The protocol is essentially reacting to the high rate of topology changes. The traffic does not seem to influence the behavior of BEST, because the same information needs to be maintained no matter what point-of-attachment is used. DSR and DST have almost the same behavior in terms of control overhead. DSR performs well in this traffic pattern, because with every flood search towards the point-of-attachment, the point-of-attachment learns the reverse path to the source from the source route accumulated in the queries. Another reason is that the fast changing topology forces out stale routes from DSR caches. This also results in DSR sending about 10 % more data packets than DST or BEST as shown in Fig 4.b.

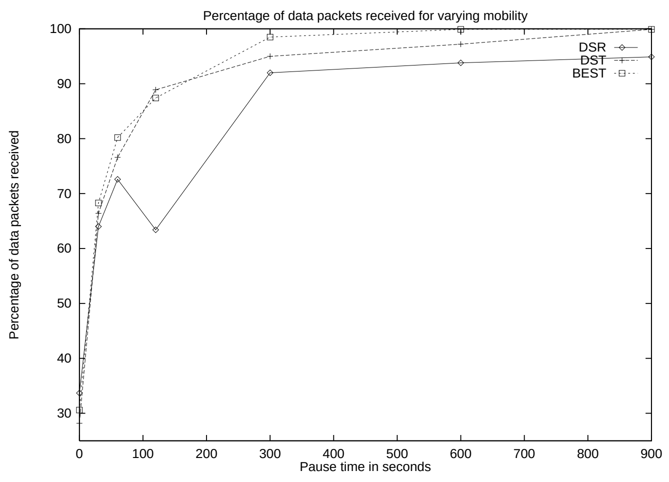
Fig. 4.c and Fig. 4.d show us the results for the static case. This scenario is important because it resembles a static community network, e.g., households with wireless routers used to reach the Internet through an access point. In this case, BEST incurs about 3 times more control overhead than DST, whereas DSR incurs 14 times more control overhead than DST. DST performs this well because the entire network knows the path to the point of attachment with a single flood search. Since there are no topology changes, there is no need for another flood search. BEST also performs much better for a static network than for a dynamic one. No topology changes mean no table driven updates after the initial updates sent when the network comes up. The surprising result is the really bad behavior by DSR, most of which seems to be driven by increase in flood searches caused by old routes. A similar behavior is seen in terms of the ratio of data packets received. DST and BEST lose very few packets, while DSR seems to lose about 50% of them. As congestion due to control packets increases, we observe more and more data packets being dropped.

VI. CONCLUSIONS

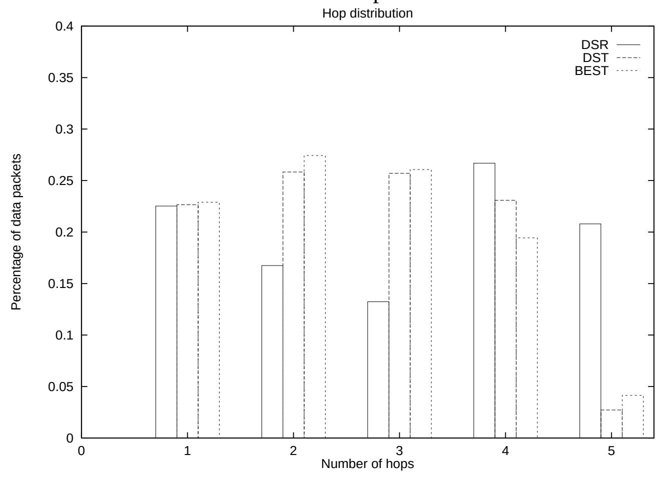
We presented source tracing as an approach to achieve efficient routing in ad hoc networks using either on-demand routing or table-driven



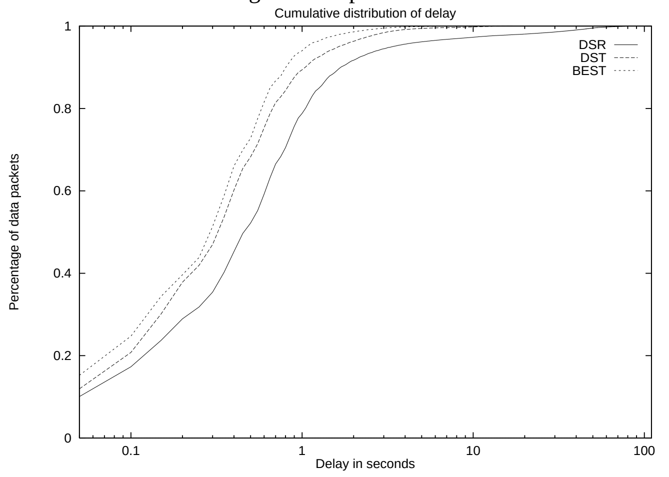
(a)
Number of control packets sent



(b)
Percentage of data packets received



(c)
Hop count distribution



(d)
Cumulative delay distribution

Fig. 2. Results for 20 sources picking random destinations for peer-to-peer flow

routing protocols. Simulations were used to compare DST, BEST and DSR, which is one of the most efficient on-demand routing protocols. The results showed that DST provides comparable average delays and packet delivery ratios while incurring far less control overhead than DSR or BEST. In our first scenario, which closely resembled an ad-hoc scenario for a battlefield or an emergency situation, DST had about one-tenth the control overhead of DSR while delivering packets with the same efficiency as BEST, which is table-driven. BEST, has about one-third the control overhead of DST while having the best results for hop count and delay. For the second scenario, which is comparable to community networks accessing the Internet via wireless links, DSR had almost 14 times more overhead than DST, which suggests that DST is an ideal solution for static community networks. In static networks, the poor performance of DSR in terms of delay and throughput suggests that it needs a mechanism to flush out stale routes in static scenarios. In scenario 2, BEST has double the overhead of DSR and DST when all the nodes are moving. This suggests that a table-driven routing protocol is a wrong choice for scenarios with high topology change and few destinations. On the other hand, BEST delivers almost all the packets and has one fourth the control overhead of DSR for the static version of scenario 2, which implies that it may be used as a solution for community networks, though DST is a better option.

Given that BEST provided good results for application-oriented metrics like hop count and delays, which are of vital significance for QoS sensitive flows, it appears that an ideal routing protocol would have to use table-driven updates for certain sources and on-demand approach for others. This can be achieved with the proper combination of source tracing rules.

References

[1] S.P.R. Kumar C. Cheng, R. Reley and J.J. Garcia-Luna-Aceves. A Loop-Free Extended Bellman-Ford Routing Protocol without Boumcing Effect. *ACM Computer Communications Review*, 19(4):224–236, 1989.

[2] IEEE Computer Society LAN MAN Standards Committee. *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*. The Institute of Electrical and Electronics Engineers, 1997. IEEE Std 802.11.

[3] J. Broch et. al. A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols. In *Proc. ACM MOBICOM 98*, Dallas, TX, October 1998.

[4] Per Johansson et. al. Scenario Based Performance Analysis of Routing Protocols for Mobile Ad-Hoc Networks. In *Proc. ACM Mobicom'99*, Seattle, Washington, August 1999.

[5] R. Dube et. al. Signal Stability-Based Adaptive Routing (SSA) for Ad-Hoc Mobile Networks. *IEEE Pers. Commun.*, February 1997.

[6] Kevin Fall and Kannan Varadhan. *ns notes and documentation*. The VINT Project, UC Berkeley, LBL, USC/ISI and Xerox PARC, 1999. Available from http://www-mash.cs.berkeley.edu.

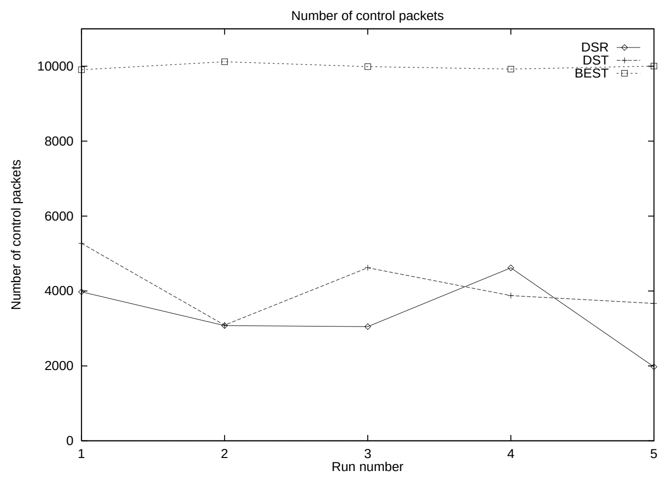
[7] J.J. Garcia-Luna-Aceves and S. Murthy. A Path Finding Algorithm for Loop-Free Routing. *IEEE/ACM Trans. Networking*, February 1997.

[8] J.J. Garcia-Luna-Aceves and M. Spohn. Source-Tree Routing in Wireless Networks. In *Proc. IEEE ICNP 99, 7th International Conference on Network Protocols*, Toronto, Canada, 1999.

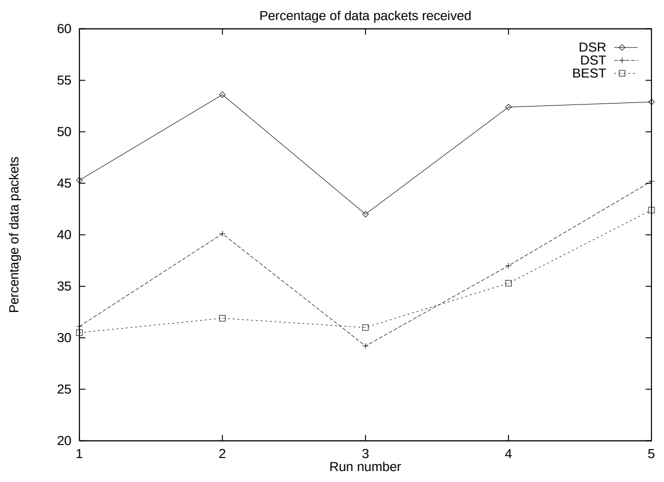
[9] Z. Haas and M. Pearlman. The Performance of Query Control Schemes for the Zone Routing Protocol. In *Proc. ACM SIGCOMM '98*, Vancouver, British Columbia, August 1998.

[10] D. B. Johnson and D. A. Maltz. Dynamic Source Routing in Ad-Hoc Wireless Networks. *Mobile Computing*, 1994.

[11] S. Murthy and J.J Garcia-Luna-Aceves. An Efficient Routing Protocol for Wireless Networks. *ACM Mobile Networks and Applications Journal*, 1996.



(a)

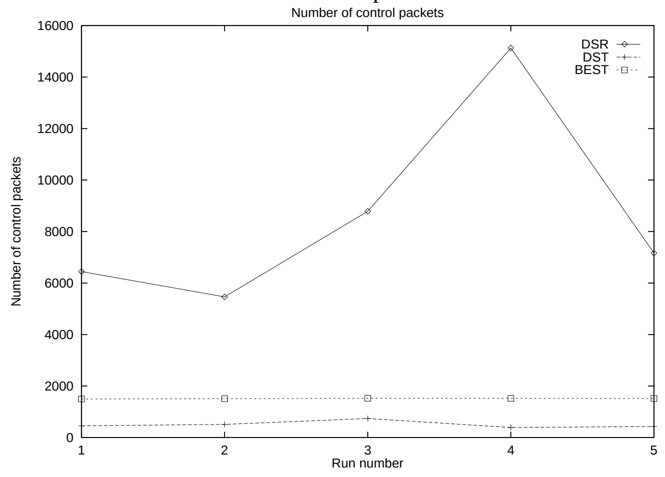


(b)

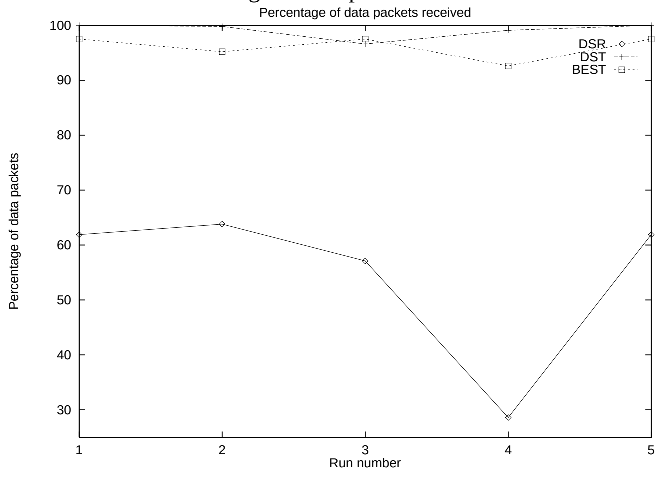
All nodes moving

Number of control packets sent

Percentage of data packets received



(c)



(d)

Static topology

Number of control packets sent

Percentage of data packets received

Fig. 4. Results for single point of attachment

[12] V. D. Park and M. S. Corson. A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks. In *Proc. IEEE INFOCOM'97*, Kobe, Japan, April 1997.

[13] C. E. Perkins and P. Bhagwat. Highly Dynamic Distance-Sequenced Distance-Vector(DSDV) for mobile computers. *Computer Communication Review*, 24(4):234–244, October 1994.

[14] C. E. Perkins and E. M. Royer. Ad Hoc On-Demand Distance Vector Routing. In *Proc. of IEEE WMCSA'99*, New Orleans, LA, 1999.

[15] J. Raju and J.J. Garcia-Luna-Aceves. A comparison of on-demand and table driven routing for ad-hoc wireless networks. In *Proc. ICC 2000*, June 2000.

[16] J. Raju and J.J Garcia-Luna-Aceves. Efficient On-Demand Routing Using Source-Tracing in Wireless Networks. In *Proc. IEEE Globecom 2000*, November 2000.

[17] C.K. Toh. Associativity-Based Routing for Ad-Hoc Mobile Networks. *Wireless Personal Communications Journal, Special Issue on Mobile Networking and Computing Systems, Kluwer Academic Publishers*, 4(2):103–109, Mar. 1997.