

Node-Centric Hybrid Routing for Ad Hoc Networks

Soumya Roy and J.J.Garcia-Luna-Aceves
Computer Engineering,
University of California, Santa Cruz
soumya, jj@cse.ucsc.edu *

Abstract

We present node-centric approaches to hybrid routing for ad hoc networks in which normal nodes are distinguished from special nodes, called netmarks, hosting popular network services or functioning as points of attachment to the Internet. With node-centric hybrid routing, netmarks force common nodes to maintain routing information for them by either sending routing updates proactively, or by requiring nodes to maintain on-demand routing entries towards them for extended periods of time. Routes between peer nodes are set up on-demand. Two node-centric routing solutions are presented based on partial link-state information. Simulation results using ns2 show that maintaining table-driven routing for netmarks and on-demand routing for common nodes performs much better than purely on-demand routing protocols based on distance vectors, path information, or link-state information.

1. Introduction

In table-driven or proactive routing protocols for ad-hoc networks, control overhead increases with the size of the network and becomes redundant if the number of communicating peers is much less than the total number of nodes in the network. To address the scaling problem of table-driven routing, on-demand routing protocols have been proposed for ad hoc networks. Nodes running such protocols set up and maintain routes to destinations only if they are active recipients of data packets and generate network-wide queries to establish routes to destinations. However, when only a few nodes of the ad hoc network must act as sources and sinks of data packets, and such choices are very stable, maintaining routing information to such destinations on demand and treating those nodes as any other node may not

be as attractive as a proactive approach. This motivates the interest in a hybrid approach to routing in ad hoc networks.

The Zone Routing Protocol (ZRP) [3] constitutes a framework for hybrid routing in ad hoc networks. ZRP adapts a hierarchical-routing approach based on clusters (called zones) and maintains routes proactively to destinations inside a zone, and on-demand routing is used to establish routing information spanning more than one zone. In this chapter, we advocate a different approach to hybrid routing that is node centric rather than based on zones or areas of the network.

The rationale for a node-centric approach to hybrid routing is that there are many cases in which certain nodes in an ad hoc network need to host special services to other nodes in the ad hoc network, specially when the ad hoc network is a wireless extension of the Internet (e.g., DNS services, Internet access, web proxies). We call those nodes that support special services for the rest of the nodes (and therefore that have a high likelihood of communicating with the rest of the ad hoc network) *netmarks*. The netmark can be fixed as well as mobile, depending on the application scenario. Under a node-centric hybrid routing approach, paths are constantly maintained between nodes and the netmarks. The forward and reverse paths between netmarks and nodes are maintained constantly. The paths between normal nodes are set up on demand.

Section 2 introduces two approaches to node-centric hybrid routing, in which a netmark is distinguished from normal nodes (which can be done through addressing or by having an explicit tagging mechanism). In one approach, a netmark forces the rest of the nodes to maintain their routes to it for long periods of time once they acquire such routes. This amounts to extending the caching of netmark routing information. In another approach, a netmark uses proactive routing updates to push its routing entry into the routing tables of the rest of the nodes in the ad hoc network. To describe our approaches, we assume that the nodes in the ad hoc network form a subnet and each node has a unique identifier, by which routing protocols and other applications can identify it. By looking at the address of the destination

*This work was supported in part by the Defense Research Projects Agency (DARPA) under grant F30602-97-2-0338 and by the US Air Force/OSR under Grant No. F49620-00-1-0330

of any IP packet, any node can determine whether or not the packet is meant for a node outside the subnet.

Section 3 shows how an on-demand routing protocol can be modified to adopt the node-centric approaches proposed in Section 2. When multiple netmarks are present in the network, the way in which nodes affiliate themselves to netmarks has a direct impact on performance of the routing protocols. How to handle multiple netmarks is addressed elsewhere [9].

Section 4 presents the results of our performance comparison of node-centric hybrid routing with purely on-demand routing protocols. The results of our simulation experiments illustrate the benefits of the node-centric hybrid routing approach.

2. Node Centric Hybrid Routing

2.1. Hybrid Routing by Extended Caching

In pure on-demand routing protocols, routers set up paths to other nodes based on the existence of flows with them. Routes are cached once they are obtained using a route discovery mechanism and they are modified when they become invalid due to link failures. Among the on-demand routing protocols proposed in the literature, the basic difference is in how routes are cached and invalidated, and how route changes are reported to other nodes. Extending an on-demand routing protocol to support hybrid routing through extended caching of the routes to the netmarks entails the following two main changes to a pure on-demand protocol:

- A node sends a route request for a netmark whenever it loses all routes to it independently of its traffic to the netmark.
- Any node generates a route error whenever it detects the loss of a route to a netmark independently of the traffic for the netmarks.

We now summarize how extended caching can be applied to specific on-demand routing protocols, namely AODV, DSR and SOAR.

The ad-hoc on demand distance vector (AODV) protocol is based on distance vectors and uses sequence numbers to prevent temporary and permanent loops. AODV supports incremental routing because no routing table loops can be formed. Route Requests (RREQs) are generated by the sources of data packets and forwarded by intermediate nodes. When RREQs are forwarded, reverse routes for the source of the RREQ are installed. Route replies (RREPs) can be sent by the destination or an intermediate node with an unexpired route entry for the destination. The RREP message initiates the creation of a path for the destination in intermediate nodes that forward the RREP back to the

sender of the RREQ. Each routing table entry has an expiration period (*active_route_timeout*) associated with it.

The specification of AODV [7] states that AODV can use the periodic network-layer Hellos or link-layer notifications for determining connectivity with neighbors. When a node detects that its path to a destination has been broken due to a failed link, it sends a route error (RERR) packet to its active predecessors for that destination, i.e. those neighbors known to use the node to forward packets to the destination.

AODV can easily incorporate the idea of extended caching by increasing the expiration period of the route for the netmarks to a very high value or infinity. Every time the paths to a netmark fail due to link failures, the nodes can send RERRs to active predecessors, which travel upstream. When a node loses its route for a netmark, it starts the route discovery mechanism, irrespective of the presence of traffic for netmarks. Royer et al. [10] have suggested a similar foreign agent discovery mechanism using AODV where routes for foreign agents are discovered irrespective of the presence of traffic. However, routes for foreign agents are not maintained differently from the routes to common nodes.

The dynamic source routing (DSR) protocol uses source routes to forward data packets and exchanges routes in the form of paths. Routes are stored in a cache, until an indication that a link in the route is broken is obtained through route error (RERR) messages or link-layer notifications. A route discovery cycle is started by a source if it loses all routes to the required destination. DSR can determine on its own if a link is broken by doing multiple retransmissions, or can depend on the link layer for link-failure notifications during packet transmission failures. Route errors containing information about failed links are sent towards the source of data packets. In DSR, if a node detects the route failure for a netmark while forwarding data packets for a common node, then it does not know how to propagate RERR information for the netmark. Hence, generating route errors independently of traffic to netmarks is not trivial in DSR and flooding RERRs would be expensive. RREQs sent in response to route failures to netmarks cannot be used to replace the functionality of RERRs, because RREQs do not have the information about failed routes. However as in AODV, anytime the route to a netmark is broken, the basic route discovery mechanism of DSR can be easily modified to send RREQs for netmarks, irrespective of the presence of traffic for them.

The source-tree on demand adaptive routing (SOAR) [8] protocol is a link-state routing protocol in which routers exchange minimal source trees in their control packets. A minimal source tree consists of the state of the links along the paths used by the routers to reach active (important) destinations. Important destinations are active receivers, relays or possible relays. Each router uses the minimal source tree of its neighbors and its outgoing links to get the partial

topology of the network. Routing table entries for known destinations are computed using a path selection algorithm on the partial topology and the data packets are forwarded hop-by-hop according to the routing entries. Links are validated using sequence numbers. SOAR uses queries and replies to create routes to unknown destinations. Update packets containing modified minimal source trees are generated when a node decides that its neighbors need to be updated to prevent erroneous forwarding or the formation of routing-table loops in the paths to important destinations.

The notion of *important node* in SOAR can be generalized to incorporate the concept of netmarks by always tagging netmarks as *important*, while the rest of the nodes become important on the basis of the traffic flowing to those nodes. Another variation of this approach is that netmarks would be considered important for longer periods of time than common nodes. Therefore, depending on the level of importance of a particular node, paths to nodes remain fresh for different time-spans. Hence, a simple modification to the basic mechanism of routing information exchange in SOAR enables incorporating the idea of extended caching of routes to netmarks. We call this modification netmark-aware on-demand link state routing (NOLR).

2.2. Hybrid Routing with Proactive Routes

The second approach to hybrid routing consists of maintaining proactive routes for the netmarks, while on-demand routes are used for other nodes. The modifications required for any on-demand routing protocol to adopt this approach are the following:

1. Adding a route for a netmark for the first time necessitates sending updates to neighbors, so that they can also set up new paths to the netmarks.
2. Depending on the protocol, route errors, route requests, or route updates are generated for netmarks, independently of the traffic to them.
3. A netmark can advertise its presence by sending Hellos to enable new neighbors to set up paths to it, and to find out whether the netmark is still reachable without having to depend on link-layer notifications. This enables paths to netmarks to be more proactive, rather than being data-packet driven.

Next we summarize how DSR, AODV or SOAR can be changed to incorporate the above concepts of hybrid routing. Implementing a network-layer Hello mechanism at the netmarks is easy in any of the three protocols.

As mentioned earlier, AODV uses destination sequence numbers to validate routes to destinations. Hellos sent by the netmarks in AODV must contain the highest sequence

number for the netmarks, so that the receiving node can install new direct routes for the netmarks. Given that DSR sends RERRs only to the source of data packets when there is a failure of packet transmissions along a particular link, adding a Hello mechanism will not help DSR, because in such a case the protocol does not have a mechanism to decide how to propagate information about the loss of netmark-routes to other nodes in the network. However, if a node keeps track of the neighbors (i.e. predecessors) that use it for data delivery to netmarks for the last *pre-defined* amount of time, route failures to netmarks can be recursively reported to other nodes through the predecessors. Incorporating the Hello mechanism benefits both AODV and SOAR, because the paths to netmarks remain more up to date.

To propagate new route information for netmarks in SOAR requires only a change in the rules for sending an *update*. Rather than only sending an *update* for a destination when the path cost to it increases, updates are also sent when a route for a new netmark is discovered. Unlike SOAR, both AODV and DSR need the introduction of a new type of control packet that propagates route updates for netmarks to all the nodes in the network when a route to a netmark is first discovered.

3 Hybrid Routing Using SOAR

We have chosen SOAR as the basic routing protocol to illustrate the benefits of node-centric hybrid routing over on-demand routing because SOAR has been shown to be more efficient than DSR [8] and the results presented in Sec. 4 show that SOAR outperforms AODV. Furthermore, the modifications needed in SOAR to adopt hybrid routing are much simpler than the modifications required in DSR and AODV.

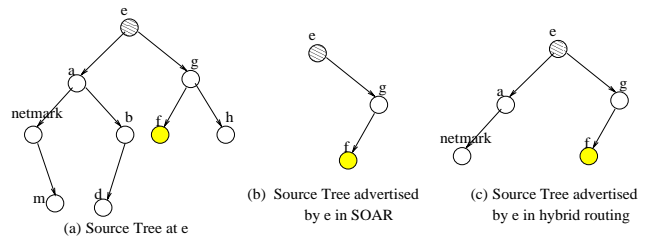


Figure 1. Differences in control information between SOAR and NEST/NOLR

Extended caching of netmarks can be adopted in SOAR by considering paths to different destinations as important for different periods of time. This modification of SOAR is called NOLR (netmark-aware on-demand link state routing).

node in the network, the following mechanism is adopted to set up the reverse paths from a netmark to any common node without introducing any extra control overhead.

When data packets start flowing from a node towards the netmark, the intermediate nodes along the path towards the netmark can set up paths towards the source of the data packets. For example, when the data packet from node a reaches node c and finds the destination is a netmark, then node c adds an entry in its routing table for node a as $[dest = a, nexthop = c]$. Similarly, the netmark keeps a routing-table entry $[dest = a, nexthop = c]$. These routing entries expire after a *Soft_State_Interval*. When link (b, c) breaks, data packets are forwarded along the path $[a, b, d, e, netmark]$ and the netmark replaces the entry $[a, c]$ with $[a, e]$ when the data packets arrive from node e . Similarly, when nodes d and e forward packets, they set up soft-state entries for destination a . Node c removes entry $[a, b]$ after the *Soft_State_Interval* due to the absence of any data packets from node a towards the netmark. Routes for different netmarks from the same node can cut same intermediate nodes. In that case, at an intermediate node, variations of the path traced by the data packets with the same source but with different destinations can lead to route flapping. Therefore, to prevent route flapping, the soft-state entry is not modified till some pre-defined time has elapsed.

The reverse routes are set up towards the source based on the flow of data packets only if the destination of the data packets is a netmark. This is because each node maintains up-to-date paths only to netmarks. Hence, given that the paths from a node to any other node may not be current, the reverse path between any two common nodes would lead to data packet losses. In steady state, the reverse route from a netmark is essentially the same as the forward path. Therefore, if the forward routes to the netmarks are correct, the reverse routes are going to be also correct.

In the absence of a flow of data packets from a common node to the netmark, the netmark must resort to queries to find paths to the destinations. The number of these queries is reduced drastically if the flow between the netmark and the node is mainly bi-directional e.g. in *tcp* like flows.

When a node receives a packet from the application layer and it is meant for a node in the ad hoc network, it forwards it to the next hop as indicated in the routing table entry, provided that the node has a route to the destination. All nodes can determine when the packets are meant for a node outside the ad hoc network by looking at the IP address of the destination, and forward such packets towards the netmark.

4. Performance Evaluation

4.1. Simulation Model

We compare the performance of node-centric routing approaches NEST and NOLR with the performance of pure on-demand routing protocols SOAR [8], DSR [5] and AODV [7] using the ns2 network simulator. For DSR, we used the code available with the ns2 simulator [4]. For AODV, we used the code available from the implementation of Marina [6] and the constants provided with the code. SOAR has been implemented according to the specifications provided by Roy and Garcia-Luna-Aceves [8]. NOLR is the modification of SOAR with extended caching of routing information for netmarks. NEST uses the same constants used in SOAR [8] along with three additional constants: *Hello_Interval*, *Dead_Time_Interval* and *Soft_State_Interval*. *Hello_Interval* (three secs) is the interval between sending of two consecutive Hello packets by the netmark. *Dead_Time_Interval* (nine secs) is the time interval for which if a node does not receive any packet from a neighboring netmark, the link to the netmark is considered to be down. When a broadcast control packet is sent, the netmark can defer the next transmission of Hello packet for a time equal to the *Hello_Interval*. *Soft_State_Interval* (one sec) is the maximum time a soft-state routing entry stays in the routing table, without being refreshed.

DSR, AODV, SOAR, NOLR and NEST do not depend on the link layer for neighbor discovery. All protocols use link-layer indications about link-failures when data packets cannot be delivered along particular links. Use of link-layer information for discovering neighbors can significantly improve the performance of routing-layer protocols. However, because our objective is to test the routing protocols as stand-alone protocols, we have not considered the effects of MAC layer interactions on the routing protocols' performance and promiscuous mode of operation has been disabled. The link layer protocol used is the IEEE802.11 distributed co-ordination function (DCF) for wireless LANs, which uses a RTS/CTS/DATA/ACK pattern for all unicast packets and DATA packets for all broadcast packets. The physical layer approximates a 2 Mbps DSSS radio interface. The radio range of the radio is 250m. We assume a netmark does not change its *netmark* status during the entire length of the simulation. Nodal movement occurs according to the random waypoint model introduced in [1]. The speed of a mobile node during its movement is uniformly distributed between 0 and 20m/sec.

We use two traffic models for performance evaluation, which we call (a) the INTNET model and (b) the RELIEF model. These traffic models are more realistic compared to the traffic models used in prior analyses [1], [2], in which continuous CBR traffic flows exist between randomly cho-

sen nodes making the traffic pattern more or less uniform throughout the network.

The INTNET model is similar to the scenario of using ad hoc networks as wireless extensions of the Internet. The communication is mainly from each of the common nodes towards the netmark hosting commonly-accessed servers or acting as access point to the Internet. The number of flows between mobile nodes only is much less compared to the number of flows between nodes and netmark. The traffic pattern is based on a FLOW_OFF/ON model, with the parameters as given in Table 1.

During the FLOW_ON period, there exists *cbr* traffic and there is no packet flow during the FLOW_OFF period. The motivation behind simulating the FLOW_OFF/ON model, rather than a model in which the flows are on continuously, is that Web traffic consists of FLOW_ON/OFF periods, where the OFF periods correspond to the user’s think time, and the ON period represents download time. In our experiments with the INTNET model, there are four random flows between any two randomly selected common nodes at any time. The duration of these flows is always 200 secs and all the flows are bi-directional in nature.

The RELIEF traffic model is used to simulate traffic in relief or battlefield scenarios, where the group members report to the group leaders while the group members also exchange information. The group leader is the netmark contacted more frequently compared to other nodes. There are four random flows between common nodes and there are at most six random flows from a common node towards the netmark. We divided the set of common nodes into five groups and only one member in the group can talk at a time with the netmark. The traffic pattern per group is also like the FLOW_OFF/ON model. The packet arrivals during FLOW_ON period follow an interrupted deterministic process (IDP) like voice traffic. The ON/OFF periods during a FLOW_ON period correspond to talkspurt/silence periods of the speaker. The parameters for the RELIEF model are as given in Table 1.

Constants	RELIEF Model	INTNET model
FLOW_ON period	uniform (30,150) secs	Uniform Dist (30,120) secs
FLOW_OFF period	uniform (10, 20)	Uniform Dist (50, 120) secs
Packet Size	66 bytes	66 bytes
Rate	17 packets/sec (9kbps)	2, 3, 4 ,5 packets/sec per node
talkspurt	350ms	
silence	650ms	

Table 1. Constants for Flows in RELIEF and INTNET models

We evaluate the routing protocols based on packet delivery ratio, control packet overhead, average hop count, end-to-end delay, and the number of queries and replies sent by each protocol.

4.2. Experimental Scenario 1

The first scenario consists of a network of 31 nodes moving over a rectangular area of 1000mx500m. There is a single fixed netmark in the system, which is placed at coordinates (500, 250). The pause time of other nodes is uniformly distributed between zero and a maximum value, which can be one of 0, 15, 30, 45, 60, 120 and 300 seconds. The simulation length is 600 secs, while the results are presented on the basis of at least 3 simulation runs, where each run is with the same INTNET traffic model but with a different randomly generated mobility scenario (this is also true for subsequent experiments). Performance results are presented for a load of five packets/sec.

Most of the results for AODV, SOAR and DSR conform to the results published previously for those protocols [2], [8] and [1]. As shown in Fig. 3(b), AODV’s control overhead is found to be significantly higher than DSR’s or SOAR’s, except for the high mobility scenarios. AODV’s control overhead consists primarily of queries (Fig. 3(d)), while the control overhead of DSR consists mainly of replies (Fig. 3(e)). This is because AODV resorts to the route discovery mechanism more often than DSR, while DSR sends multiple replies to queries. Contrary to the findings in [2], [1] an interesting result is that AODV’s control overhead in highly mobile scenarios is lower than DSR’s. Because each node in the INTNET model sends and forwards packets for a netmark, the number of cached entries for the netmark is comparatively higher in DSR than in scenarios where the traffic pattern is uniform. That effectively leads to significantly higher number of cached replies (many of which contain stale routes) which amount to higher control overhead in DSR than in AODV. In low mobility scenarios in which path information becomes stale less often, the effect of injecting old routes due to multiple replies is much smaller.

SOAR produces much fewer control packets compared to DSR or AODV under all mobility scenarios. The reason behind this is that SOAR resorts to fewer route discovery queries than AODV or DSR, because of the redundancy in the exchanged routing information in the control packets specifying minimal source trees, and because of the use of mostly local updates to solve path breakage, rather than sending route error messages to the source of data packets. Because SOAR and DSR can use stale information, SOAR and DSR deliver slightly fewer data packets compared to AODV under high mobility. The performance degradation is less in SOAR compared to DSR, because SOAR uses sequence numbers to validate link–state information and DSR uses explicit route error messages to invalidate link information.

The performance of NOLR was found to be almost identical to SOAR’s. Accordingly, for clarity, Fig. 3 does not

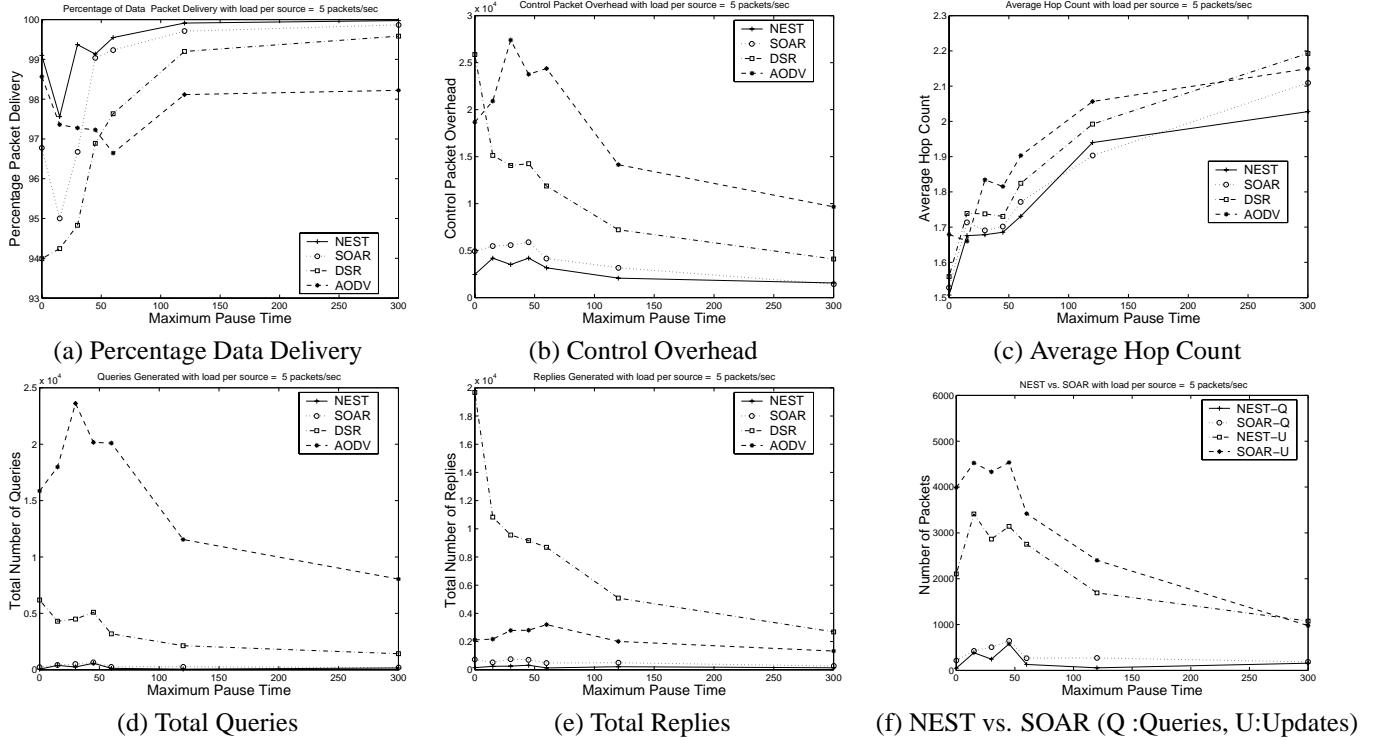


Figure 3. Performance of NEST, SOAR, DSR, AODV in a 31node Network at load per node of 5 packets/sec with fixed network

show the results obtained for NOLR. Unlike SOAR, NOLR maintains routing information for networks for longer periods of time compared to the time for maintaining information for other nodes. The reason why NOLR and SOAR exhibit the same performance in this scenario is that each node either sends or forwards packets for the network the vast majority of time. Therefore, any node in SOAR ends up treating the network as *important* throughout the simulation.

Under all scenarios, NEST performs much better compared to all purely on-demand routing protocols, both in terms of data delivery and control overhead. NEST (Fig. 3(a)) delivers more packets compared to other protocols. In NEST each node always maintains correct paths to the network. Therefore, NEST loses much fewer data packets than AODV, even though AODV attempts to avoid using stale routing information. SOAR maintains information for networks for significant periods of time; however, NEST paths are more accurate, because the network advertises itself periodically to force its routing information in other nodes and nodes using NEST update their neighbors when they first discover routes to networks. This conclusion is validated by the results of Fig. 3(f), in which we see that more updates are needed in SOAR compared to NEST to purge stale link-state information. On an average, NEST produces around 30% fewer updates than SOAR. We also

find that NEST (Fig. 3(f)) produces fewer queries compared to SOAR, which leads to a reduction of replies in NEST (Fig. 3(e)). Queries are still sent by NEST for discovering routes on-demand with common nodes and for probing the network when the network becomes unreachable due to network partitions. We also see from Fig. 3(c) that the average hop count in NEST is the smallest, because NEST detects the presence of networks much faster.

4.3. Experimental Scenario 2

The second scenario focuses on the effect of network mobility. It consists of a network of 30 nodes and one network with common nodes moving at a speed uniformly distributed between 5m/s and 20m/s. Pause times are uniformly distributed between 0secs and 30secs. Three different movement scenarios for the network are analyzed while keeping the mobility pattern for other nodes the same. The network in these scenarios is either static (model *s*), mobile (model *m*, the network moves over a rectangular area (250,250)) or very mobile (model *vm*, the network can move over the entire area (1000,500)). Because most of the traffic is towards the network, the routing protocols would be more stressed to maintain routes as the mobility of the network increases. The network moves with a speed similar to the speed of common nodes with pause time between 10 and 30 secs. We use the INTNET and RELIEF traffic

models, which are indicated as REL and INT in Fig. 4. Accordingly, a static netmark model with INTNET traffic pattern is indicated as *sINT*, while a *vm* model with RELIEF traffic pattern is represented as *vmREL*.

For this scenario, the results for AODV are significantly worse than for the rest of the protocols. Accordingly, the results for AODV are not shown in order to show in more details the performance differences among the other protocols. From Fig. 4 we see that there is no appreciable difference in the performance of the routing protocols, NEST, SOAR and DSR for the *m* and *s* model because of the limited mobility of the netmark in *m* model. The performance of all the routing protocols suffers when the netmark becomes totally mobile. From Fig. 4, we find that the INTNET model produces more stress on the routing protocols than the RELIEF model does, with DSR being affected the most.

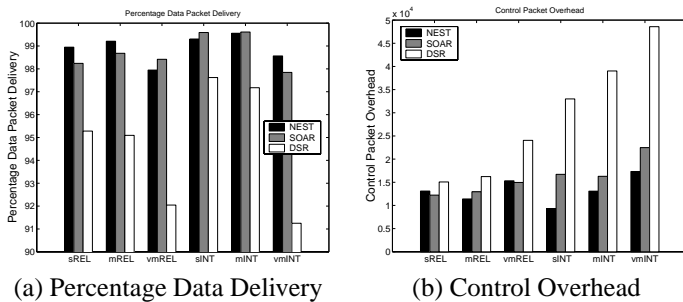


Figure 4. Performance in a 3Node Network with varying mobility models for netmark and two different traffic models

From Fig. 4(a), we see that SOAR and NEST deliver on an average the same number of data packets in both traffic models. DSR’s percentage data delivery is 4%-7% smaller than the data delivery achieved by SOAR and NEST, with the performance becoming worse with higher mobility of the netmarks. This is because of packet losses due to unavailability of routes to forward data packets at intermediate routers, which implies that DSR suffers due to stale path information.

DSR’s control overhead is comparable to that of SOAR or NEST for the *sREL* or *mREL* models (Fig. 4(b)). DSR sends significantly more control packets for the INTNET model, where DSR utilizes redundancy in routing information less efficiently than SOAR or NEST.

SOAR and NEST have similar control overhead for the RELIEF model, though in the INTNET model NEST outperforms SOAR and DSR. This is because the RELIEF model has fewer flows (around six) towards the netmark compared to the INTNET model, in which theoretically any node can communicate any time with the netmark. When the number of flows is smaller, fewer links are used for active data delivery. Because detections of link failures are

triggered only by the failure of transmission of data packets, with fewer flows more links remain stale in the topology table. Therefore, if the number of flows between common nodes is the same as the number of flows between netmarks and common nodes, the Hello mechanism does not improve the condition because SOAR and NEST require almost the same number of updates to purge wrong routing information. This indicates that the node-centric approach to proactive route maintenance can improve the performance of the network, however the degree of performance improvement depends on the amount of communication between common nodes and the netmark.

We also observe that netmark mobility does not impact the performance of NEST more than the performance of purely on-demand approaches, which could have been an argument for using an on-demand approach rather than a hybrid approach when netmarks are very mobile.

Because voice traffic is delay sensitive, we analyzed the delay performance for each of the routing protocols (Table 2) for the RELIEF traffic model, where voice traffic is used. The results presented are for a randomly chosen run, so as not to average out the high frequency components of individual runs. This is important for voice traffic performance, because worst-case performance results are required for quality assurance. Following conclusions can be drawn from the data available from Table 2:

- Range (the difference between minimum and maximum delay) is significantly high under all cases. This is because the network becomes partitioned, and the node discovery mechanism is not very fast in on-demand routing protocols and can become really slow as the timeouts for resending queries increases non-linearly. The range for DSR for Mobile Relief Model (Table 2) is as high as 49 secs. Though NEST maintains proactive routes with the netmark, the range for NEST is also high because it uses on-demand routes to common nodes.
- As expected, there is an increase in delay when the netmark is more mobile, because nodes have more stale routes.
- NEST has better delay performance than SOAR in terms of percentile values because NEST has fewer queries and the paths tend to be more up to date, thereby spending less time queueing packets either at the routing layer or the link layer. This indicates that the hybrid routing approach helps to reduce the end-to-end delay of data packets.

5. Conclusions

We have presented node-centric approaches to hybrid routing for ad hoc networks that distinguish between nor-

Table 2. End to End Delay Distribution of the Voice Traffic For Different Netmark Mobility Models

Percentile	Static Relief Model (sREL)			Mobile Relief Model (mREL)			Very Mobile Relief Model (vmREL)		
	NEST (s)	SOAR (s)	DSR (s)	NEST (s)	SOAR (s)	DSR (s)	NEST (s)	SOAR (s)	DSR (s)
90	0.0091	0.0110	0.0087	0.0101	0.0151	0.0121	0.0611	0.0834	0.0986
95	0.1136	0.1503	0.0734	0.1067	0.1768	0.0917	0.3222	0.3930	0.3695
97	0.2682	0.3301	0.2226	0.2150	0.3799	0.2087	0.6832	0.8593	0.6371
range	19.89	13.7335	19.329	2.5041	14.4443	49.083	17.28	23.4054	13.119

mal nodes and special nodes called netmarks, which host popular network services or function as points of attachment to the Internet. With node-centric hybrid routing, netmarks force other common nodes to maintain routing information for them by either advertising their routing information as in table-driven routing protocols, or by requiring nodes to maintain routing entries towards them for extended periods of time. Routes between peer nodes are set up on-demand. We have evaluated the changes needed to incorporate node-centric hybrid routing in the basic mechanism of routing for some pure on-demand routing protocols, namely AODV, DSR and SOAR and compared the performance of AODV, DSR and SOAR with the hybrid approaches, NEST and NOLR (which have been adapted from SOAR) using ns2.

On the basis of ns2 simulations, we have found that, if a node in the ad hoc network acts as a source or relay of data packets for significant portion of its lifetime, the benefit of extending caching information in a purely on-demand routing protocol is not noticeable. However, maintaining proactive routes as in NEST offers better performance than any on-demand routing protocol, both in terms of data delivery and control packet overhead when the traffic flow is mostly from common nodes towards the netmark. We have also found that the performance of NEST is not affected by the mobility of netmarks.

References

- [1] J. Broch, D. A. Maltz, D. B. Johnson, Y. C. Hu, and J. Jetcheva. A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols, 1998. ACM Mobicom.
- [2] S. R. Das, C. E. Perkins, and E. M. Royer. Performance Comparison of Two On-Demand Routing Protocols for Ad-Hoc Networks, 2000. IEEE Infocom.
- [3] Z. Haas and M. R. Pearlman. The Zone Routing Protocol (ZRP) for Ad Hoc Networks, June 1999. <http://www.ee.cornell.edu/haas/Publications/draft-ietf-manet-zone-zrp-02.txt>.
- [4] ISI-NS. The network simulator - ns-2.1b6, <http://www.isi.edu/nsnam/ns/>, 25 May, 2000.
- [5] D. B. Johnson and D. A. Maltz. Dynamic Source Routing in Ad-Hoc Wireless Networks, 1994. Mobile Computing.
- [6] M. Marina. Aodv code for cmu wireless and mobility extensions to ns-2, last updated on 12/07/2000. <http://www.eecs.uc.edu/mmarina/aodv/>.

- [7] C. E. Perkins, E. M. Royer, and S. R. Das. Ad Hoc On-Demand Distance Vector (AODV) Routing, March, 2002. Mobile Ad Hoc Networking Working Group, draft-ietf-manet-aodv-10.txt.
- [8] S. Roy and J. J. Garcia-Luna-Aceves. Using Minimal Source Trees for On-Demand Routing in Ad Hoc Networks, 2001. IEEE Infocom.
- [9] S. Roy and J. J. Garcia-Luna-Aceves. Node-Centric Hybrid Routing for Ad Hoc Wireless Extensions of The Internet , 2002. IEEE Globecom.
- [10] E. M. Royer, Y. Sun, and C. Perkins. Global Connectivity for IPv4 Mobile Ad hoc Networks, 2001. draft-ietf-manet-globalv4-00.txt.