

Node-Centric Hybrid Routing for Ad-Hoc Wireless Extensions of The Internet

Soumya Roy and J.J.Garcia-Luna-Aceves
 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 other common nodes to maintain routing information for them by advertising their routing information as in table-driven routing protocols. This reduces the network-wide flooding and the corresponding delay for route set up every time a session needs to be established between a normal node and a netmark. Routes between peer nodes are set up on-demand. A node-centric routing solution is presented based on partial link state information. In this solution, table-driven routing is maintained for netmarks and on-demand routing is supported for common nodes. Simulation results using ns2 show that this approach performs much better than such on-demand routing protocols as DSR, AODV, and SOAR.

I. INTRODUCTION

Table-driven or proactive routing protocols can become expensive in terms of control overhead in mobile ad hoc networks, because each node in the network must maintain routing information for every other network node, although the node occasionally handles traffic destined for some nodes. 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. However, when only a few nodes of the ad hoc network must act as sources and sinks of data packets, maintaining routing information to such nodes on demand and treating those nodes as any other node may not be as attractive as a proactive approach to establishing routing information to them while on-demand routing is used between less accessed nodes. This motivates the interest in a hybrid approach to routing in ad hoc networks.

The Zone Routing Protocol (ZRP) [1] 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 paper, 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 practical scenarios in which certain nodes in an ad hoc network need to host special services that are requested throughout the ad hoc network. For example when ad-hoc networks are wireless extensions of Internet, some nodes need to act as DNS servers, Internet Access points or 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*. Forward and reverse paths between netmarks and common nodes

are maintained constantly, while routes between common nodes are set up on-demand.

The landmark hierarchy [2] is an earlier node-centric approach to hierarchical routing designed for proactive routing in large networks. The key difference between the node-centric routing described in this paper and the landmark hierarchy is that a landmark becomes the address of a common node, while a netmark is a destination that provides services.

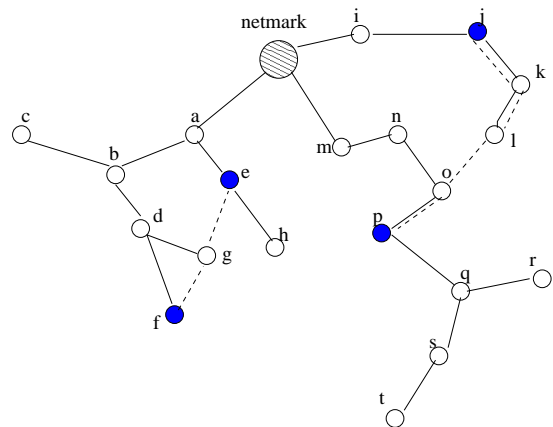


Fig. 1. Figure showing an ad hoc network with a single netmark

Section II introduces our approach to node-centric hybrid routing. In this 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. Section II also shows how an existing on-demand routing protocol can be modified to adopt this approach. Section III addresses the issues that arise with having multiple netmarks in an ad hoc network. Section IV presents the results of our performance comparison of node-centric hybrid routing with purely on-demand routing protocols. Section V concludes the paper.

II. NODE CENTRIC HYBRID ROUTING

A. Hybrid Routing with Proactive Routes to Netmarks

In on-demand routing protocols routes for necessary destinations are set up in a reactive basis using route requests and route replies. Route errors are generated when the routes break necessitating alternate path set ups. The modifications required for any on-demand routing protocol to adopt hybrid routing with proactive routes to netmarks are the following:

1. A netmark can advertise its presence by sending Hellos to enable new neighbors to set up paths to it, and to enable old neighbors to find out whether the netmark is still reachable without

having to depend on link-layer notifications.

2. 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.

3. Route errors and route requests are generated for netmarks irrespective of the presence of traffic to the netmarks.

All the above three techniques make paths to netmarks proactive rather than reactive to traffic.

Now we look at how some of the existing on-demand routing protocols like DSR, AODV or SOAR can be changed to incorporate the above concept of hybrid routing. The ad-hoc on demand distance vector (AODV) protocol is an on-demand distance vector routing protocol in which each routing table entry has an expiration period (*active_route_timeout*) associated with it.

The dynamic source routing (DSR) protocol exchanges routes in the form of paths [11]. DSR keeps routes for destinations until route errors or link layer notifications indicating failure of routes are received. These events are triggered only by the failure of transmission of data packets over links, which implies route failure propagation will not improve by extended caching of netmarks, in which the basic idea is to maintain correct paths in absence of data traffic also.

The source tree on demand adaptive routing (SOAR) [5] protocol is a link-state routing protocol in which routers exchange minimal source trees in their control packets, consisting of the state of the links along the paths used by the routers to reach active (important) destinations.

Implementing a network-layer Hello mechanism at the netmarks is easy in any of the above three protocols. Hellos sent by the netmarks in AODV should contain the highest sequence number for the netmarks, so that the receiving node can install new routes for the netmarks. Because DSR sends route errors only to the source of data packets, adding a Hello mechanism does not help DSR, because it does not know how to propagate loss of netmark-routes to other nodes in the network. Some improvement can be achieved if a node keeps track of the neighbors who use it for data delivery to netmarks for the last *pre-defined* amount of time, so that route failures to netmarks can be reported to those predecessors. Unlike in DSR, by increasing the time for which a route is considered important, in both SOAR and AODV, route errors and route requests can be sent irrespective of the presence of traffic.

To propagate new route information for netmarks in SOAR only requires a change in the rules for sending an *update*. Updates are going to be sent not only during path cost increase but also during new netmark-route discovery. Both AODV and DSR need the introduction of a new broadcast control packet, which would propagate paths to netmarks when a route is first found for the netmarks.

B. Hybrid Routing based on SOAR

We have chosen SOAR as the routing protocol to illustrate the benefits of node-centric hybrid routing over on-demand routing because of the following reasons :

1. SOAR has been shown to be more efficient than DSR [5] and our own results, presented in Sec. IV, show that SOAR completely outperforms AODV.

2. Modifications in SOAR to adopt hybrid routing are much simpler than in both DSR or AODV.

The Netmark Enhanced Source Tree (NEST) routing protocol adopts the routing mechanisms of SOAR to maintain proactive routes for netmarks and on-demand routes for other nodes in the network. In the following we provide the details of NEST.

Fig. 2 shows the difference in the control message between NEST and SOAR advertised for the nodes in the network shown in Fig. 1, where every node has a proactive path with the netmark and nodes e and i have on-demand routes set up for f and p respectively. The source tree at e (Fig. 2a) is the tree consisting of links that e uses to reach the netmark and other nodes in the network. Router e advertises a part of this complete source tree to its neighbors, which is called the *minimal* source tree. For SOAR the *minimal* source tree would only consist of links needed to reach nodes with which it has active flows. In this example, e has active flow with f , the minimal source tree advertised by e would be as shown in Fig. 2b. In NEST, even if e does not have active communication with the netmark, it advertises links belonging to the path to it (as shown in Fig. 2c), which implies e always considers netmark as important.

C. Netmark Discovery

In NEST, netmarks send Hello packets to inform their neighbors of their presence. This same effect is achieved by sending beacons at the MAC layer. When a node receives a Hello packet, it assumes the presence of netmark in its neighborhood or the neighbor protocol can send such indication. At the routing layer, if the node does not receive the Hello packet for some predefined interval of time, then it can declare that the link to its neighbor is down. The MAC layer can also notify the routing layer about link failures when it cannot deliver data packets. In the presence of data traffic this mechanism can detect link failures faster.

When a node has a new entry for the netmark, it updates it neighbors about the new route, which in turn will choose to send an *update* if it discovers the netmark for the first time. Therefore, every node in the network will know about the path to the netmark. In case of loss of updates *queries* are used to set up paths with netmarks.

Because netmarks acting as gateways or hosting proxy (or DNS) services will not change over small time scales, we assume all nodes can be pre-configured statically with the netmark addresses.

D. Maintaining Paths in NEST

Fig. 3 illustrates the process of setting up forward and reverse paths for NEST.

In Fig. 3, when c learns of the netmark, it advertises the netmark in its source tree and hence b will know about the netmark and also about neighbor c . When b re-advertises, a will know about b, c , and *netmark*, i.e., the path to netmark and the intermediate nodes b, c . Similarly, b will know about an alternate path [$b, d, e, \textit{netmark}$] from d , but b chooses the path through c as it is of smaller length. If link (b, c) fails, then b can choose the alternate path to netmark through d . The downstream nodes from a mobile node towards the netmark may only know about the upstream predecessor but not about all upstream nodes, e.g.,

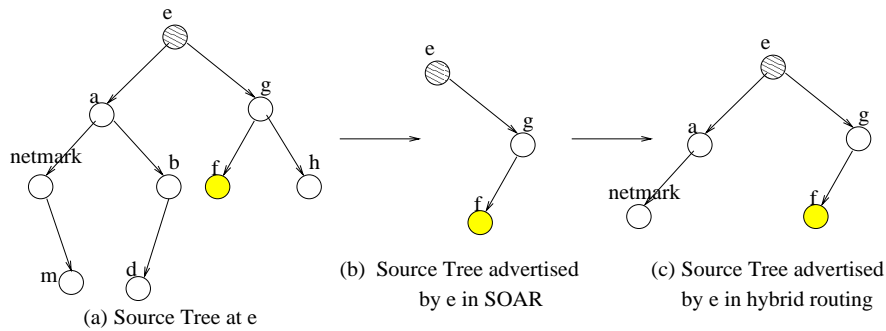


Fig. 2. Figure showing difference in control information in SOAR and NEST

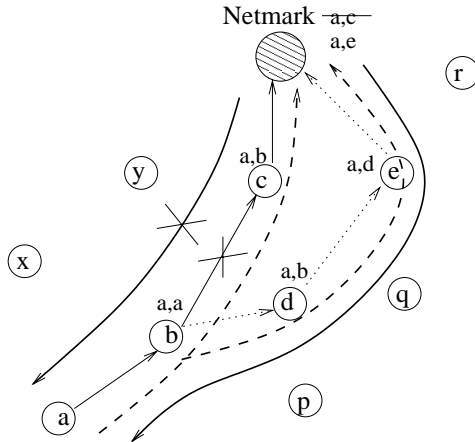


Fig. 3. Setting up of paths between netmarks and nodes

when b advertises its source tree, it may not advertise link to a as a is not an important destination. In such a case, c will know only about b , but not about a . Similarly, the netmark may know about c , but not about nodes a and b . In order to send packets to a , netmark in such a case would have to send a *query* for a . To prevent a *query* to be initiated from the netmark for every mobile node in the network, the following mechanism is adopted to set up the reverse paths 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 a reaches c and finds the destination is a netmark, then c adds an entry in its routing table for a as $[dest = a, nexthop = c]$. Similarly, the netmark will keep an entry $[dest = a, nexthop = c]$. These routing entries will expire after a *soft_state_interval*, such that when link (b, c) breaks, data packets will be forwarded along the path $[a, b, d, e, netmark]$ and netmark will replace the entry $[a, c]$ with $[a, e]$ as the data packets arrive from e . Similarly, when d and e forward packets, they set up soft-state entries for the destination a . Node c removes entry $[a, b]$ after the *soft state interval* due to the absence of any data packets from a towards the netmark. If a node finds that the next hop for the reverse path towards any destination changes, that change is only recorded in the routing table, if the previous route has not been used for *soft_state_interval*. This prevents route-flapping in

case routes for different destinations from the same source pass through same intermediate nodes.

III. MULTIPLE NETMARK SCENARIOS

When multiple netmarks are present in the network, depending on the purpose the netmarks serve, the routing can be adapted to further improve performance of routing protocols. How to do this depends on the way in which nodes affiliate themselves to netmarks.

A. Dynamic Affiliation

A node need not be affiliated with any particular netmark and it can communicate with any one netmark. This can happen when the ad-hoc network is an extension of the Internet, and there are multiple Internet access points. The common nodes in the network can communicate with any access point, as all access points are connected to the Internet. Since packets are forwarded to any netmark, routing becomes efficient in terms of control overhead because (a) redundancy of routes to the Internet reduces the number of expensive route discovery cycles; (b) Anycast route discovery mechanism reduces control overhead. In any route discovery mechanism, *queries* are not required to be sent individually for each netmark. *Anycast* queries can be sent asking for a route to the anycast address of all the netmarks. In such a case, any router who has at least a route to a netmark can reply and in case of availability of multiple routes, the reply would contain the route to the nearest netmark.

B. Static and Hybrid Affiliation

In static affiliation irrespective of the distance of a node from the netmark, a node can be made to always use a particular netmark. Depending on the mobility, a node in order to reach the netmark affiliated to it, might have to use some other node, affiliated to another netmark, which implies every router has to know the routes to all netmarks, though packets from a node would always be forwarded to a particular netmark.

There can be scenarios where the affiliations can be hybrid i.e. both static and dynamic. For example, if the mobile router wants to reach a particular proxy server, which one of the netmarks host in its local subnet, forwarding packets to the netmark hosting the proxy server would be more efficient than forwarding to the other netmarks which can be connected to different networks. Packets meant for networks remote to any of the netmarks can be forwarded to any one of them. In terms of routing

the approach will be similar in both static and hybrid affiliation scenarios.

IV. PERFORMANCE EVALUATION

We have evaluated the performance of NEST with the pure on-demand routing protocols SOAR, DSR and AODV using the ns2 network simulator [12]. Following are the constants required in NEST for netmark discovery:

```
Hello_Interval (interval between Hello packets): 3 secs
Dead_Time_Interval (interval after which the netmark can be considered) : 9 secs
soft_state_duration (timer value for the soft-state routing entry) : 1 sec
```

Promiscuous mode of operation has been disabled due to practical purposes [5]. 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 the 2 Mbps DSSS radio interface (Lucent WaveLan Direct-Sequence Spread-Spectrum). The radio range of the radio is 250m.

Nodal movement in the simulation occurs according to the random waypoint model [6] within a rectangular area of 1000mx500m.

We have introduced INTNET traffic model for performance evaluation. This traffic model, which simulates the web traffic, is more realistic compared to the traffic models used in [6], [7]. It consists of sequences of FLOW_OFF and FLOW_ON (Fig. 4) periods where the OFF periods correspond to the user's think time, while the ON period represents download time. During the FLOW_ON period, there exists a *cbn* traffic and there is no packet flow during the FLOW_OFF period. Following are the parameters that defines the model:

```
FLOW_ON period      : Uniform Dist (30,120) secs
FLOW_OFF period     : Uniform Dist (50, 120) secs
Packet Size         : 66 bytes
Rate                : 3, 5 packets/sec per node
```

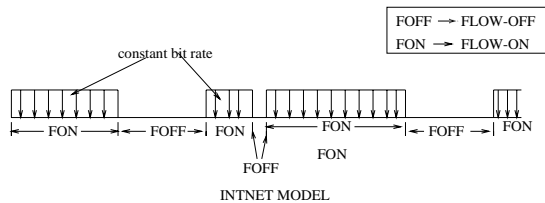


Fig. 4. Internet Traffic Flow Model

In our experiment all nodes have continuous flows with the netmark while there are four random flows (of duration 200 secs) between any two randomly selected nodes. All the flows are bi-directional in nature.

A. Performance Criteria

We have evaluated the routing protocols based on the following metrics: (a) Packet Delivery Percentage, (b) Control Packet Overhead (c) Average Hop Count (d) End-to-End Delay (f) Number of Queries, Replies.

B. Single Netmark Scenario

There is a single netmark in the network, which is placed at co-ordinates (500, 250) and is fixed throughout the simulation time. The *pause time* 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 having a different randomly generated mobility scenario but same traffic model (this is also true for subsequent experiments).

Most of the findings on AODV, SOAR and DSR from our experiment conform to the results published in [7], [5] and [6]. However, contrary to the findings in [7], [6] an interesting result for the INTNET model 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 compared to scenarios where the traffic pattern is uniform. That effectively leads to significantly higher number of cached replies which amount to higher control overhead in DSR than in AODV. SOAR produces much less control packets compared to DSR or AODV under all mobility scenarios with varying loads because SOAR resorts to less amount of route discovery. Because of the fact SOAR and DSR both can use stale information, under heavy load scenarios and high mobility, SOAR and DSR suffer slightly in terms of data delivery compared to AODV.

Under all scenarios, NEST has been found to perform much better compared to any other purely on-demand routing protocol, both in terms of data delivery and control overhead. NEST (Fig. 5a and Fig. 6a) always delivers more packets compared to other protocols, with the effect being more prominent under heavy load. The effect of overloading the network does not affect NEST, because each node always tries to maintain correct paths to the netmark and therefore, under heavy loads, NEST loses much fewer data packets than other protocols due to wrong route information. NEST paths are more accurate than paths in SOAR, because the netmark in NEST advertises itself periodically to force its routing information in other nodes and nodes using NEST update their neighbors when they first discover routes to netmarks. This conclusion is validated by the results of Fig. 5f and Fig. 6f (NEST-U and SOAR-U), where we see that more *updates* are needed in SOAR compared to NEST to purge wrong link state information. On an average, NEST produces around 30% fewer *updates* than SOAR. We also find that NEST (NEST-Q and SOAR-Q in Fig. 5f and Fig. 6f) reduces the number of *queries* compared to SOAR, which is desirable in large networks where *queries* can be expensive. The reduction of *queries* also leads to reduction of *replies* in NEST (Fig. 5e and Fig. 6e). *Queries* are still sent by NEST for discovering routes on demand and during network partitioning when NEST needs to probe the network. We also see from Fig. 5c and Fig. 6c that on an average hop count in NEST is the smallest. This is because NEST detects the presence of netmarks much faster. However to conserve network bandwidth, as the nodes in NEST do not advertise route changes when distances decrease, the path length in NEST can still be sub-optimal.

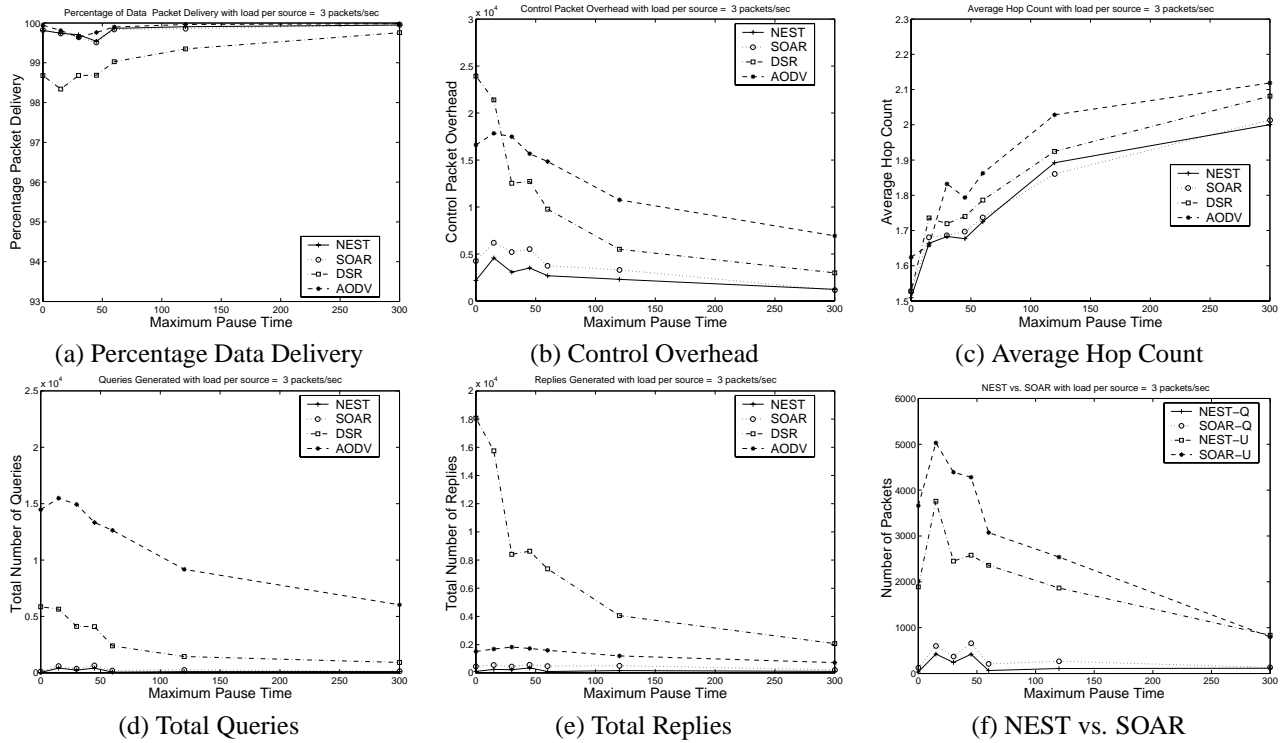


Fig. 5. Performance of NEST, SOAR, DSR, AODV in a 31node Network at load per node of 3 packets/sec with fixed network

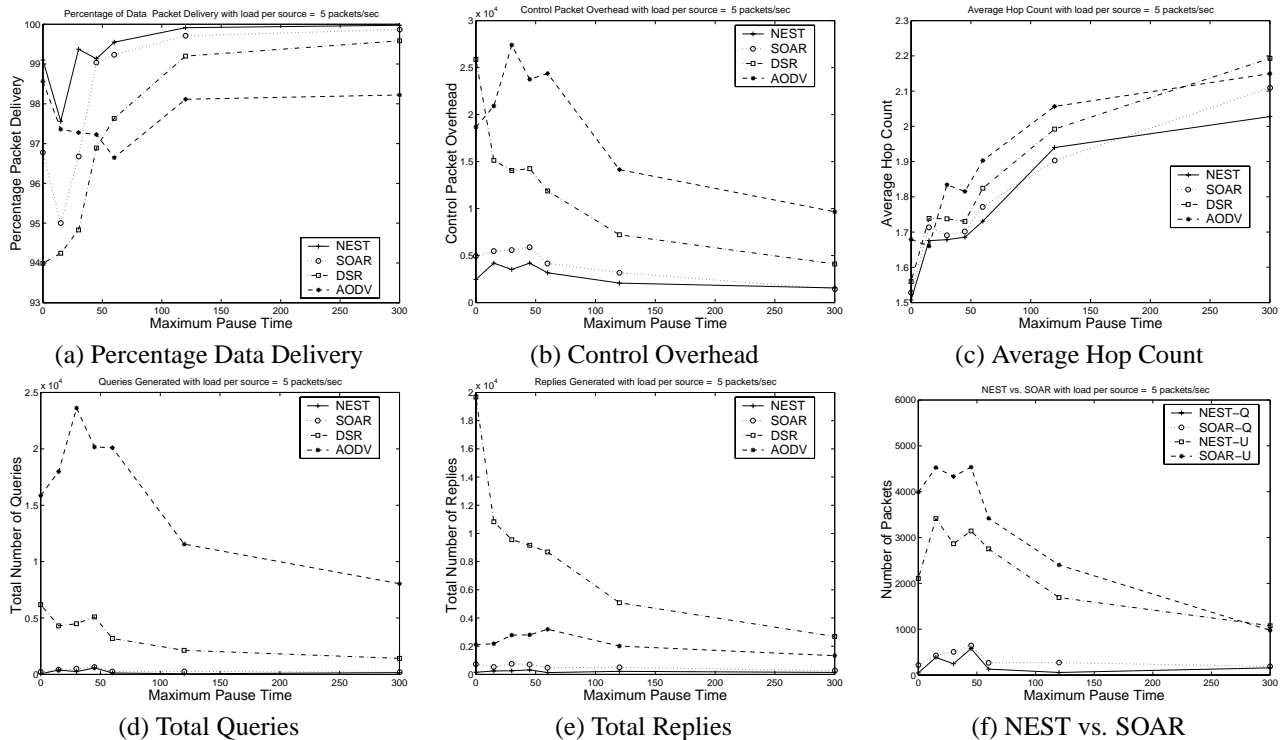


Fig. 6. Performance of NEST, SOAR, DSR, AODV in a 31node Network at load per node of 5 packets/sec with fixed network

C. Multiple Network Scenario

In this scenario there are two networks along with 30 mobile nodes. The networks are placed at coordinates (250, 250) and (750, 250) and are fixed throughout the simulation. The traffic pattern is according to the INTERNET model. Packets for the Internet are always forwarded to the nearest network. In case

of unavailability of routes to any network, *anycast queries* (as discussed in Sec III) are sent and routes are established based on the *anycast* replies. We compare the performance of NEST with SOAR (denoted as *anycast-enabled SOAR* or A-SOAR in Fig. 7). Like A-SOAR, if needed, NEST also uses *anycast* queries for networks.

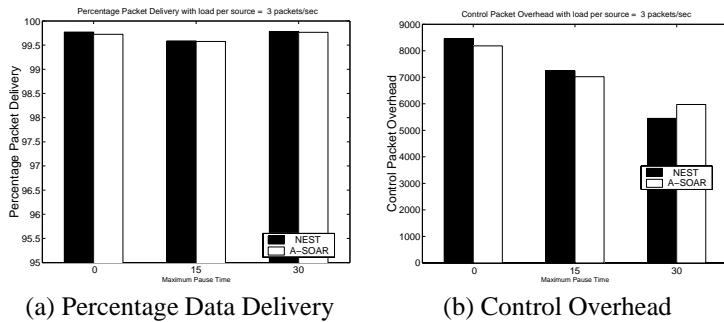


Fig. 7. Performance of NEST and SOAR for a network with 30 nodes and 2 netmarks

From Fig. 7, we see that SOAR performs as well as NEST in terms of data delivery and control overhead under all three mobility scenarios. This is in contrast to the results presented in the previous scenario, where SOAR produces higher control overhead than NEST. This improvement of performance in SOAR can be attributed to the following three reasons: (a) If a route to a given netmark is not available, packets can still be sent to the other netmark, which helps in reducing the number of *queries*; (b) anycast route replies help to reduce flooding of *queries* and speed up route discovery; and (c) reducing the number of *query-reply* packets helps to prevent old link-state information to be injected into the network, which helps to reduce the number of updates.

V. CONCLUSIONS

Here we have presented a node-centric approach to hybrid routing for ad hoc networks that distinguishes between normal nodes and special nodes, called netmarks, hosting popular network services or functioning as points of attachment to the Internet. 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 DSR, SOAR and AODV and compared the performance of AODV, DSR and SOAR with the hybrid approach, NEST using ns2.

On the basis of ns2 simulations when the traffic flow is mostly from common nodes towards the netmark, maintaining proactive routes as in NEST has been found to be always an attractive method of routing than any other on demand routing protocol both in terms of data delivery and control packet overhead. In a moderately-sized network served by multiple netmarks, the performance of on-demand routing protocols can be significantly improved by maintaining routes to any of the netmarks and sending anycast *queries* asking for a route to the nearest netmark. Future work will compare anycasting techniques with node-centric hybrid approaches in big networks serviced by multiple netmarks.

REFERENCES

- [1] Z. Haas and M. R. Pearlman, "The zone routing protocol (zrp) for ad hoc networks," in <http://www.ee.cornell.edu/haas/Publications/draft-ietf-manet-zone-zrp-02.txt>, June 1999.
- [2] P. Tsuchiya, "The Landmark Hierarchy : A New Hierarchy for Routing in Very Large Networks," in *ACM Sigcomm*, 1988.
- [3] C. E. Perkins, E. M. Royer, and S. R. Das, "Ad Hoc On-Demand Distance Vector(AODV) Routing," in *draft-ietf-manet-aodv-08.txt*, March, 2001.
- [4] D. B. Johnson and D. A. Maltz, "Dynamic Source Routing in Ad-Hoc Wireless Networks," *Mobile Computing*, 1994.

- [5] S. Roy and J.J. Gracia Luna Aceves, "Using Minimal Source Trees for On-Demand Routing in Ad Hoc Networks," in *IEEE Infocom*, Anchorage, Alaska, 2001.
- [6] 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.
- [7] C. E. Perkins S. R. Das and E. M. Royer, "Performance Comparison of Two On-Demand Routing Protocols for Ad-Hoc Networks," in *Proc. of IEEE Infocom 2000*, Tel Aviv, Israel, Mar 2000.
- [8] K. Leibnitz D. Staehle and P. T. Gia, "Source traffic modeling of wireless applications," in *CNO Activity Report 1999/2000*, June, 2000.
- [9] C. E. Perkins, E. M. Royer, and S. R. Das, "Ad Hoc On-Demand Distance Vector(AODV) Routing," in *draft-ietf-manet-aodv-10.txt*, January, 2002.
- [10] E. M. Royer, Y. Sun, and C. Perkins, "Global Connectivity for IPv4 Mobile Ad hoc Networks," in *draft-ietf-manet-globalv4-00.txt*, November, 2001.
- [11] Y.C. Hu and David Johnson, "Caching Strategies in On-Demand Routing Protocols for Wireless Ad Hoc Networks," in *ACM Mobicom*, Boston, Massachusetts, 2000.
- [12] "The network simulator - ns-2," in <http://www.isi.edu/nsnam/ns/>, ns-2.1b6.
- [13] M. Marina, "Aodv code for cmu wireless and mobility extensions to ns-2," in <http://www.ececs.uc.edu/mmarina/aodv/>, last updated on 12/07/2000.
- [14] C.R. Baugh, J. Huang, R. Schwartz, and D. Trinkwon, "Traffic model for 802.16 tg3 mac/phy simulations," in <http://ieee802.org/16>, 2001.