Dynamic NIx-Vector Routing for Mobile Ad Hoc Networks

Young J. Lee and George F. Riley

School of Electrical & Computer Engineering
Georgia Institute of Technology, Atlanta, GA 30332
Email: {young, riley}@ece.gatech.edu
Tel: (404) 894-9041; Fax: (404) 894-9959

Abstract— **We present a new protocol for multi–hop routing in mobile ad hoc networks called Dynamic NIx–Vector Routing (DNVR). Our scheme is based on the *NIx–Vector* concept for efficient routing originally designed for wired networks. DNVR acquires a loop–free route and maintains it on a need basis as do other on–demand protocols. However, DNVR has several new features as compared to other existing reactive routing protocols which lead to more stable routes and better scalability. DNVR effectively validates the stored routes as well as efficiently senses the up-to-date network topology during a route discovery phase by sending a unicast probe packet. To accommodate networks with a high degree of mobility, the routing states are invalidated in a timely manner. DNVR adopts a conservative route discovery strategy by suppressing route requests in some cases, and thus only a few routes are maintained per destination. Moreover, it attains bandwidth efficiency by using a *Neighbor Index* and Medium Access Control (MAC) addresses for routing purposes, which obviates the need for address resolution. We show via simulation that DNVR scales well to a large network with varying traffic load under diverse mobility scenarios. We compare DNVR to the well known *Dynamic Source Routing (DSR)* protocol, which is believed to be one of the most efficient on-demand routing protocols. Simulation results reveal that DNVR is as efficient as DSR while at the same time providing higher packet delivery ratios (up to 46 % higher) and smaller delays (up to 23 % smaller) than DSR in most cases.**

Index Terms— **Mobile ad hoc network, wireless routing, mobile computing**

I. Introduction

A mobile ad hoc network is a collection of mobile nodes that form a network structure without the help of any existing infrastructure or central administration, such as static base stations in cellular networks. With the advancements of wireless communications and computer technologies, the use and application of mobile ad hoc networks is increasing over time. For example, military operations or disaster relief efforts are usually performed without any pre-existing infrastructure. Also commercial applications that need cooperative mobile data exchange can benefit from this mobile ad hoc networking technology [1]. Moreover, this type of network may provide an inexpensive alternative to cellular networks [2].

In recent years, several routing protocols for mobile ad hoc networks have been proposed. The primary goal of such routing protocols is to establish a correct route from a source node to a destination node in an efficient way, to maintain the discovered routes, and react to the network topology changes effectively. Ad hoc network routing protocols can be classified into two major categories according to the way routes are acquired and maintained: table-driven (proactive) and on-demand (reactive) protocols. Among these ad hoc routing protocols, commonly studied protocols include DSR [3] and AODV [4].

In Dynamic Source Routing (DSR) [3], packets are delivered using *source routing*. A DSR packet contains a complete ordered list of IP address for nodes through which it should pass. DSR is an on-demand protocol and therefore finds a path to a destination only when it wishes to send to the destination. The main advantage of this type of protocol is that the intermediate nodes on a path do not need to maintain up-to-date routing information, which removes the need for route advertisement and explicit neighbor detection mechanisms. Further, it is straightforward to obtain loop-free routes with this approach. However, the size of a packet header containing route information grows with the length of a path, which results in high packet overhead and raises scalability concerns. Obviously, this could have a negative impact due to limited bandwidth in wireless networks. There was an effort to reduce this packet overhead using *implicit source routing*[5]. It successfully reduces the packet overhead due to source routing, but shows very similar performance to DSR in other metrics such as packet delivery and latency. Furthermore, DSR does not have a mechanism to invalidate the cached routing information in a timely fashion, which can lead to a routing misbehavior due to stale routes, resulting in poor performance in highly mobile networks.

Ad hoc On-demand Distance Vector (AODV) [4] adopts the *distance vector* routing algorithm. Each node exchanges routing information to attain the up-to-date view of the network only when involved in an active routing path, which makes it an on-demand routing protocol. AODV forwards packets in a hop-by-hop manner. It uses *HELLO* messages to maintain local connectivity at each node. One of the advantages of the protocol is that it scales well to large networks. However, even though the neighbor management associated with local connectivity is done only during an active routing phase, it does increase the signaling overhead of the entire network.

All of these protocols also prefer stored route information

(at route caches or routing tables) in a route discovery phase. Even though this behavior can save some routing overhead, it often fails to sense the up-to-date network topology, which leads to the discovery of invalid paths, resulting in additional packet drops and more delays, especially in case of high mobility.

In addition, a node in a mobile ad hoc network tends to communicate with more peers than those in a wired network due to the dynamic nature of the network. Thus, frequent *address resolution* services are requested, which may degrade the performance of routing protocols in terms of delay and packet loss [3], [12].

We introduce a new scheme which we call Dynamic NIx-Vector Routing for mobile ad hoc networks (DNVR). DNVR is a pure on-demand routing protocol. It acquires a loop-free route and maintains it on a demand basis as do other on-demand routing protocols [3], [4], [6].

The DNVR protocol, however, has several distinct features from other on-demand routing protocols as follows:

- Validation of the stored route information
- Utilization of probes for efficient sensing of the network
- Management of routing states in a timely fashion
- Suppression of route requests for a conservative route discovery
- Removal of address resolution
- Use of a compact form of source routes

DNVR effectively validates the stored route information before using it as well as efficiently detects the up-to-date network topology in a route discovery phase by utilizing a unicast packet functioning as a probe. To accommodate a high degree of mobility, the routing states are invalidated in a timely manner. DNVR adopts a conservative route discovery strategy by suppressing route requests at destinations, and thus only a few routes are maintained per destination. In addition, DNVR uses MAC addresses instead of IP addresses to identify routing neighbors, which eliminates the need for additional mechanisms such as *Address Resolution Protocol (ARP)* [8] to resolve a neighbor's IP address. [1] DNVR uses a compact form of a source route since it inherits the NIx-Vector routing as a basic routing method, which reduces packet overhead significantly. By virtue of these features, DNVR provides more stable and scalable performance as compared to existing routing methods, with respect to network size, mobility, and traffic volume.

The rest of the paper is organized as follows. In Section II, we overview the wired NIx-Vector routing method. In Section III, we describe the DNVR protocol in detail. Section IV gives the performance evaluation of the new scheme and compares it with DSR. Section V concludes the paper.

II. Overview of Wired NIx-Vector Routing

The wired *NIx-Vector (NV)* routing protocol was introduced to provide an efficient routing method in the Internet [7].

[1]The use of a MAC address for identifying a routing neighbor simply removes ARP, and it operates only from the perspective of per-hop behavior. Note that it does not violate the layered protocol approach.

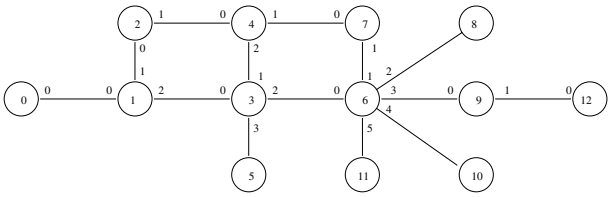

Fig. 1. *Simple routing path*

The NV routing was motivated from the observation that the specification of the IP address for each next hop in a routing path is an excess of information. The NV routing method enables the relevant routing information to be specified with a small number of bits per hop, and thus the same amount of routing information can be specified in a much smaller space.

The NIx-Vector routing method starts with the concept of a *neighbor index (NIx)*. Each router has an ordered set of routing neighbors. For example, if a router has three routing neighbors, it has a set of $\{0, 1, 2\}$. The router selects the next hop from this set. The NV consists of the concatenation of the all *neighbor index (NIx)* values selected on the path from a source to a destination.

A NV is constructed during a creation phase while a packet goes from a source to a destination. During this phase, routers make a normal routing decisions and record the *neighbor index* decisions in the packet header by simply concatenating the *NIx* value to the existing vector, and incrementing the length by the appropriate number of bits. When the packet arrives at the destination, the NV is complete and is returned to the source. Once the NV is available at a source, it is included in all subsequent packets from the source to the destination, and the routers use it to efficiently forward the packets by extracting the appropriate number of bits from the vector and decrementing the length.

Fig. 1 shows a simple network as an example. Each router has varying numbers of neighbors, and each router numbers those neighbors sequentially as shown. Suppose the path for a packet is 0–1–3–6–9–12. The resulting NV for this path is `0 10 10 011 1` (binary).

Once a NV from a source to a destination is created and becomes available to the source, it can be subsequently used for efficient $O(1)$ routing decisions. Each router extracts the appropriate number of bits from the received NV, which specifies the correct NIx for the next hop. This NIx is used to index a table of next hop IP address, interface number, and (optionally) layer 2 address. The packet is then forwarded to this neighbor with no further processing. The extracted NIx values at each router between 0 and 12 would be 0, 2, 2, 3, and 1, resulting in the correct path 0–1–3–6–9–12.

III. Dynamic NIx-Vector Routing Protocol

DNVR protocol mainly consists of two parts: NV creation and mobility management. In the NV creation phase, a path from a source to a destination is found, and a NV for the path is constructed. In the mobility management phase, DNVR detects routing failures and performs mobility management functions.

Fig. 2. *NIx structure*

A. Data Structures

DNVR utilizes three types of data structures: NIx-Vector, neighbor table, and NV Forwarding Information Base (NV-FIB). Each structure is detailed in the following sub-sections.

1) NIx and NV: The number of routing neighbors is dynamic due to mobility in a mobile ad hoc network, which makes it difficult to record a NIx with a constant number of bits. To cope with this, we slightly modified the original NIx structure. We introduced an additional prepended field called *color* that specifies the actual length of a NIx. Fig. 2 shows the NIx structure. A NIx has a color field and a neighbor index field. The color field specifies the number of bits for the neighbor index field. The neighbor index field represents the actual index of the neighbor table. The next hop can be determined from this index value at each forwarding node. The length of the color field is fixed (currently 4) while the index field length is variable. The value to be stored in the color field N_b can be computed from (1).

$$N_b = \lfloor log_2 index \rfloor + 1 \tag{1}$$

where $index \in \{1, 2, 3, \ldots\}$. The index starts from one due to the presence of a *hidden bit* [16] in the neighbor index field, where the bit `1` is assumed to always precede the number specified in the neighbor index field. Thus, the actual length of the index field is $N_b - 1$ bits since the color N_b counts the hidden bit, and the index value is obtained by prepending the hidden bit `1` to the number extracted from the index field. Given a 4-bit color, the length of the index field can range from zero to 14 bits.

The 4-bit width of the color field means that a node can accommodate up to 32K (2^{15}) neighbors. Even though this number seems to be large enough to handle every node as a routing neighbor in a realistic mobile ad hoc network, a slight increase in the color field will give us more space. For example, if we have 5 bits for the color field rather than the 4 (only one bit increase), a node will be able to accommodate up to about two billion (2^{31}) neighbors.

A NV is simply a sequence of NIxes with a NV length field. The NV length field represents the number of bits of the NV excluding the length field itself. The length of the NV decreases while a packet with a NV is routed along a path, since each NIx is removed from the NV as it is used. The NV length field also can be used to check whether or not the NV is valid. If the NV length is not positive, then the NV is invalid and cannot be used for routing. Thus, the packet with the invalid NV should be dropped immediately.

2) Neighbor Table: Each neighbor table entry consists of three fields: next hop, interface number, and lifetime. The entries are indexed by the *NIx* value, and thus have $O(1)$ access time. In routing table based approaches, $O(logN)$ access time is typical.

The next hop field contains the MAC address of the next-hop neighbor's interface. This MAC address uniquely identifies a neighbor's interface, and it is used as a link layer unicast address. The interface number specifies which interface to use for communicating with the relevant neighbor. In general, this interface is the one from which a node received routing protocol messages. The lifetime value specifies how long the associated entry will remain valid. An entry is deleted when the lifetime expires. Also an entry can be invalidated even though the lifetime has not expired if the neighbor is no longer reachable.

3) NV-FIB: A NV-FIB is a table storing NVs and the relevant information. When there is a request for sending a data packet, DNVR searches its NV-FIB for a NV with the packet destination. A NV-FIB is indexed by a *path id*. A path id is a set of a source IP address, a destination IP address, and a NV reply number. The NV reply number is generated only by a node that replies to NV request messages. Whenever a node replies with a NV reply, it increments its own NV reply number by at least one and puts the value in the returned message. In DNVR, a path can be uniquely identified by this path id.

Each NV-FIB entity has 5 elements: path id, NV, metric, state, and lifetime. The NV is a representation of the path identified by the path id. The metric field holds the correspondent information regarding the path such as hop count, load factor, etc. The state field indicates the validity of the NV. If it is invalid, the NV cannot be used to route data packets, but can be used for route maintenance purposes.

The meaning of the lifetime depends on the state field. A NV is invalidated when (i) the lifetime expires with a valid state, or (ii) the link to the next hop is determined to be broken. As soon as the NV switches to the invalid state, the lifetime field is refreshed with a new value. When this new lifetime expires, the NV is deleted permanently.

B. NIx-Vector Creation

The NV creation process relies on a flooding scheme using local broadcasts as do most ad hoc routing protocols [3], [4]. Even though recent studies [11] indicate inefficacy and unreliability of local broadcasts, the main focus of this work does not lie on the topic.

A NV creation process is invoked when a node has packets to send and does not have a NV for the destination. The node then stores the packets in a buffer for pending packets and initiates a NV creation process by broadcasting a NV request (NVREQ) message. When a node receives a NVREQ and it is the specified destination, it returns a NV reply (NVREP) message to the source. Otherwise, the node propagates the NVREQ by broadcasting or unicasting if the request has not

been processed by the node before. This action is described in detail later.

In this paper, we assume that every link involved in a network is bidirectional. In other words, we assume that the link layer protocol does not allow packet transmission over unidirectional links. The IEEE 802.11 [15] falls into this category. Thus, symmetric routing is a basic assumption where the reverse of a path from node A to node B is the path from B to A.

1) Detailed Operation: A NVREQ message carries a source IP address; a destination IP address; a NVREQ sequence number; routing metric information; a reverse NV; and the MAC address of a forwarding node's interface. When a node receives a NVREQ message, it first checks if it has already processed the request by examining a unique NVREQ identifier, a pair of <source IP address, NVREQ sequence number>. If the node has already processed the request, it drops the request and does not re-broadcast the packet. Otherwise, the node then processes the routing metric field of the NVREQ by incrementing the hop count by one.

The node then reads the MAC address of the forwarding (previous hop) node's interface from the NVREQ and adds it to the neighbor table. This action naturally returns a neighbor table index for the added node. This index is converted to a NIx as follows: the number of bits for the index, N_b, is computed by (1), and N_b is then prepended at the index as color. This NIx is concatenated to the *reverse NV* field of the NVREQ. The reverse NV is used to return a NVREP to the source later. The node then replaces the corresponding field of the NVREQ with the MAC address of its interface from which it received the request and re-broadcasts the request through the interface. This process is repeated until the NVREQ reaches the destination node.

When the NVREQ arrives at a node that is not the request target but does have a NV for the target, DNVR shows different behavior from other on-demand routing protocols. Instead of immediately replying to the request, the intermediate node forwards the request by *unicasting* to the target using the stored NV. By doing so, the stored route information is validated before it can be used, and the up-to-date network topology can be probed. On the other hand, the immediate reply relies on the stored route state that may be inaccurate, and it can incur additional packet drops and delays. In the new scheme, the unicast NVREQ message serves as a probe. If the packet carrying the NVREQ successfully reaches the target, it means that the probed path is still valid and the corresponding NV can be used for routing. Otherwise, that path is no longer valid, and thus other paths should be found. Moreover, with this method, more accurate route metrics (other than hop count) can be obtained since every reply comes from the target through all the intermediate nodes on the path.

It appears at first glance that this different behavior of validating the stored route information will take additional time to acquire a route and consequently cause more packet latency. Even though it might slightly increase the route acquisition time, it does not contribute to the overall delay performance of the protocol very much while resulting in higher packet delivery performance. This is verified through the simulation, which is presented later.

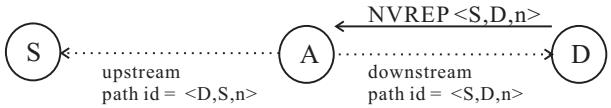

Fig. 3. *Forward and reverse NVs*

When the NVREQ message finally reaches the destination, a NVREP message is constructed and returned to the originator of the request. DNVR adopts a conservative route discovery strategy. Whenever a destination node replies to a NVREQ, it records how many times it replied to the same NVREQ, and checks if the value does not exceed a *reply threshold* value. If the node already replied the threshold value times, it simply drops the request. By doing so, only a few paths are maintained per destination in DNVR. For example, if a node has a reply threshold value *one*, the node replies only to the NVREQ that arrives first.

A NVREP message carries a source IP address; a destination IP address; a NV reply number; routing metric information; a NV; and the MAC address of a forwarding node's interface. Before constructing the NVREP, the destination node constructs a NIx from the NVREQ and completes the reverse NV by concatenating the NIx. The destination then puts in the corresponding field of the NVREP the MAC address of its interface from which it received the request. The routing metric information is copied to the relevant field of the NVREP from the NVREQ, and the NV reply number maintained at the node is incremented and put in the corresponding field of the NVREP. Finally, the reverse NV in the NVREQ is copied to the NV routing header of the NVREP, which is used to return the NVREP to the requester. This routing action using NV is detailed in the later section.

While the NVREP is routed along the reverse path from the destination to the source, the requested NV is constructed. Each node on the path reads the MAC address field of the NVREP and adds it to the neighbor table. The resulting NIx is concatenated to the NV field of the NVREP. The NV and the reverse NV (the NV in the routing header) are then copied and stored at the NV-FIB of each intermediate node. The NV represents the requested downstream path while the reverse NV represents the upstream of the path. Each NV is stored with a corresponding path id (Fig. 3). This process is repeated until the NVREP arrives at the request originator. The requester extracts the NV from the NVREP and stores it with the corresponding information at the NV-FIB. The requester then can transmit the pending packets to the destination using that stored NV.

2) Loop Freedom: For loop detection, it is enough to uniquely identify a NVREP. A NVREP can be identified by its requester, the responder, and the NVREP number, i.e., a triple <source IP address, destination IP address, NVREP number>, which is a path id (Section III-A.3). Let us suppose that node

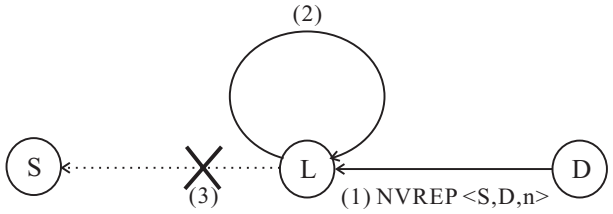

Fig. 4. *Loop detection in DNVR*

S requested a NV for node D, and thus D replies with a NVREP number n (Fig. 4). In this example, the NVREP can be identified by $< S, D, n >$. Each intermediate node on the path stores a relevant routing state with the path id when it receives the NVREP as described in Section III-B.1.

If a node, say L, finds a state already existing with the same path id while processing the NVREP, it assumes a loop is detected and does not forward the NVREP toward the requester S. The routing states that have been stored at other nodes between L and D on the path are never brought up to the active state and eventually removed. Thus, any route with loops is neither found nor used in DNVR.

C. Routing using NIx-Vector

A NV is a compact representation of a path, where the NV is a sequence of NIx values. As explained in Section III-A.1, a NIx consists of a 4-bit color field and a neighbor index field with variable length.

When a node receives a data packet, it first reads the 4-bit color field from the NIx. Let us call the value of the color field N_b. The node then reads $N_b - 1$ bits afterward. The hidden bit `1` is then prepended to the extracted bits, and the resulting bits constitute the neighbor index for the next-hop neighbor. This neighbor index returns the MAC address of the next-hop node's interface from the neighbor table. After the node gets the MAC address, it forwards the packet to the corresponding neighbor by link layer unicast if the packet is not destined for the node.

D. Mobility Management

The mobility management function of DNVR can be decomposed into two parts: route maintenance and neighbor management. The route maintenance deals with how to detect a routing failure, to report the error to corresponding nodes, and to find a new route. The neighbor management is concerned with how to detect a neighbor and when to add or invalidate a neighbor in the neighbor table at each node. The detailed operation is discussed in the following sections.

1) Route Maintenance: DNVR assumes notification of packet transmission failure from the link layer. In other words, it utilizes the link layer notification function in case a packet transmission fails. However, if this link layer feedback function cannot be supported for some reason, DNVR can optionally utilize overheard packets to detect the routing failure.

When a node detects a broken link, it notifies corresponding nodes of the failure by originating NV error (NVERR) messages. This error message contains (i) the path id for the path that has been invalidated due to the error and (ii) the IP address of the node that detected the routing failure. The error detecting node searches its NV-FIB for NVs containing this dead link. Each found NV is invalidated, and the source of that invalidated NV is notified with the NVERR message of the failure. In order to return the NVERR, the path id for the reverse path (upstream to the source) is constructed by exchanging the source with the destination part of the downstream path id, and then the error detecting node searches its NV-FIB by the reversed path id to retrieve the NV for the upstream path. That found NV is used to forward the NVERR to the source.

When each intermediate node receives the NVERR, it first checks if it has a stored NV with the path id specified in the NVERR, and then invalidates the NV if it has one. Finally, the source node becomes aware of the routing failure and invalidates the corresponding NV as soon as it receives the NVERR. The source node will invoke a new NV creation process if there are more data packets for the destination.

2) Neighbor Management: In DNVR, routing neighbors at each node are managed in a reactive fashion. It does not actively detect neighbors nor monitor the current state of existing neighbors. The neighbors are passively detected and monitored.

A neighbor is detected and added to the neighbor table only during a NV creation phase. Once a neighbor is detected and added, that neighbor can be used as a next hop for routing. The lifetime value is refreshed whenever the neighbor is used for routing. A neighbor is invalidated when the lifetime for the neighbor expires, or the node experiences a transmission failure over the link to the neighbor. This action results from three possibilities: (i) the neighbor is no longer used for routing, or (ii) the neighbor has moved out of the node's transmission range, or (iii) the link to the neighbor is highly congested. In either case, the node invalidates the neighbor. In case of (ii) or (iii), the node will detect a failure to route a packet and send a NVERR message toward the source node for the packet.

IV. Performance Comparison

This section compares the performance of DNVR and DSR. The simulation models for the MAC and the routing protocols are explained, and the performance results are presented and discussed.

A. The Simulation Environment

All of our simulations were performed using the *Georgia Tech Network Simulator (GTNetS)* [13]. GTNetS is a scalable simulation tool designed specifically to support large–scale simulations. The design of the simulator closely matches the design of real network protocol stacks and hardware. Moreover, the simulator is implemented completely in object-oriented C++, which leads to easy extension for new or modified behavior of existing simulation models. For more information, refer to the GTNetS web page at [14].

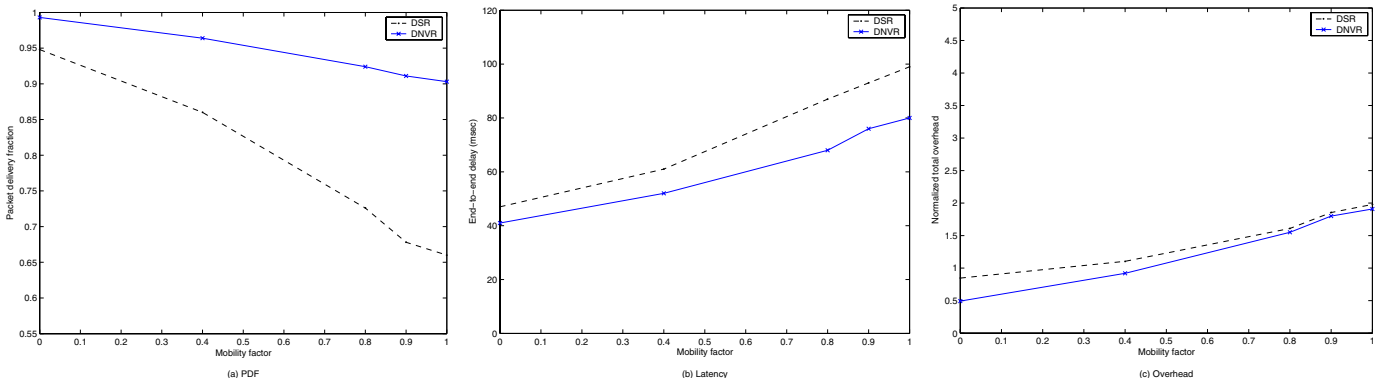

Fig. 5. *20 flows*

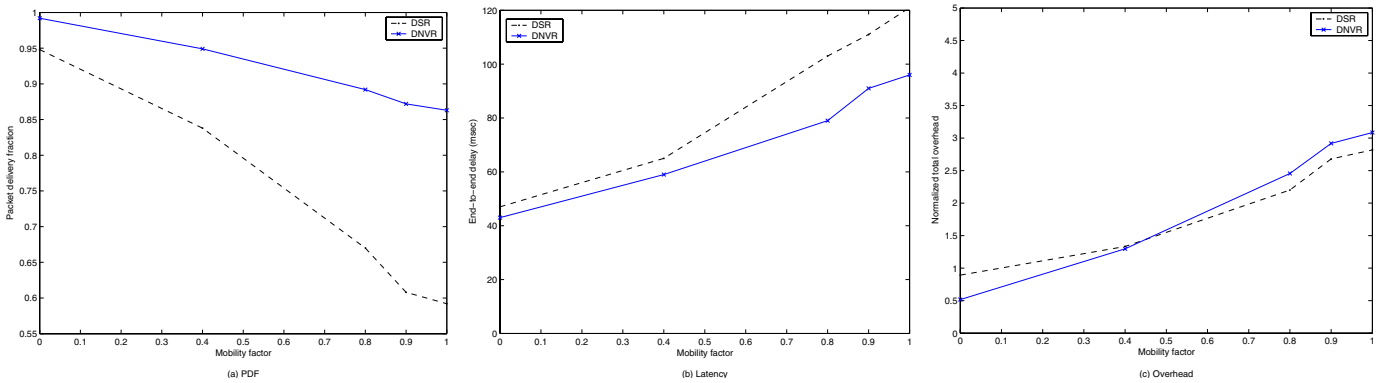

Fig. 6. *40 flows*

The distributed coordination function (DCF) of the IEEE 802.11 [15] standard was used as the MAC protocol in the simulation. Each routing protocol model has a send buffer of 64 data packets with 30 sec timeout. After the timeout, the packet is expunged from the send buffer. The send buffer holds pending packets while waiting for route reply. In addition, each wireless interface of a node has a queue known as *interface queue*. The interface queue can hold maximum 50 packets. This queue gives higher priority to routing protocol messages than data packets.

We constructed 150 different scenarios for the simulations. Each scenario is a set of mobility patterns, traffic patterns, number of traffic flows, and pause time. The mobility model used was random-waypoint [9]. To measure the performance of the protocols, a 200-node model was used. [2] We used 20 and 40 traffic flows with a rate of 4 and 2 packets/sec respectively. All traffic was created with a constant-bit-rate (CBR) data source, and all packets were 512 bytes.

In the simulation, each mobile node is placed within a rectangle of 3200 m $\times$ 900 m. Initially a mobile stays in a randomly selected location during the pause time, then selects a random target and moves to the destination with a uniformly distributed speed between 0 and 20 m/sec. Once it reaches the target, the mobile stays again during the pause time. Each simulation executed for 500 simulated seconds.

B. Performance Results

To reflect various aspects of the routing protocols, following performance metrics were used:

- Packet delivery fraction (PDF)
- Packet latency (end-to-end delay) [3]
- Normalized total overhead (byte count)

The packet delivery fraction is the ratio of the number of received data packets at the destinations to the number of data packets generated by the CBR sources. This metric captures the throughput of the network. The packet latency is the average time taken to transfer a data packet from a CBR source to its target. The route acquisition time is also reflected in the packet latency.

In order to capture the routing overhead, the normalized total overhead is introduced. The total overhead is the byte count for the routing messages and the header of data packets. The header includes the IP header and the routing header of DSR and DNVR. Then, the total overhead is normalized to the byte count of the received data packets. Hence, this metric represents the efficiency of a routing protocol, and it is a measure of how many bytes of routing overheads are needed to receive a byte of data.

[2]DSR is designed for networks of up to about 200 nodes [10]. For a fair comparison, we did not perform experiments for larger networks.

[3]We did not model the ARP effect in the simulation. It might cause additional delays for protocols that need address resolution if modeled.

In the simulation results, each data point represents an average of 30 runs with different mobility patterns and traffic patterns. To insure a fair comparison however, an identical set of mobility and traffic patterns was applied to both routing protocols in each simulation. Each result is plotted over the *mobility factor*. The mobility factor was introduced to represent the network mobility with respect to the pause time. It is defined as $(simulated\ time - pause\ time)/simulated\ time$. Thus, the value 0 means no mobility, and the value 1 the highest mobility.

As shown in Fig. 5(a) and Fig. 6(a), both protocols show above 90 % PDF with no mobility. As the network mobility increases, however, the two protocols show much different aspects. The PDF differential becomes even bigger as the network mobility and the traffic volume increase, which shows that DNVR scales better than DSR with respect to mobility and traffic volume. With the highest mobility, the PDF of DNVR is 37 % higher than that of DSR for 20 flows, and 46 % higher for 40 flows. On the average, DNVR demonstrated 23 % higher PDF than DSR.

As can be seen in Fig. 5(b) and Fig. 6(b), DNVR shows smaller packet latency for every mobility factor. The delay performance gap becomes bigger as mobility increases. DNVR demonstrated up to 23 % smaller delay than DSR. On the average, DNVR showed about 18 % smaller delay than DSR. It also turned out that DNVR consumed about 35 % less time to acquire routes than DSR even if the detailed result is omitted due to lack of space. But the portion of each protocol's route acquisition time out of the total delay was similar, which was around 30 %.

Fig. 5(c) and Fig. 6(c) reveal that the efficiency of both protocols is very similar. As can be seen in Fig. 5(c), the total overhead of DNVR is somewhat lower than that of DSR for 20 flows without regard to mobility. Fig. 6(c) shows that, for 40 flows, DNVR produces a little bit lower overhead than DSR with mid mobility, and slightly higher overhead with high mobility. In the presence of high mobility and a large volume of traffic, it is possible that the network seems partitioned temporarily to the routing protocol when it fails to find a route to a specific destination. In this case, on-demand routing protocols attempt route request retries repeatedly. DSR experiences these route discovery failures less frequently than DNVR due to the high hit rate of route caches even though these replies from route caches contain stale routes. On the other hand, DNVR issues and consequently propagates more route request messages in this case. This is the reason that DNVR shows a little bit higher total overhead in the case. On the average, DNVR showed 10 % lower overhead than DSR for 20 flows, and 4 % higher for 40 flows. Overall, both protocols demonstrated very similar protocol efficiency without regard to mobility and traffic volume.

V. Conclusions

We presented a new routing method for mobile ad hoc networks. DNVR behaves in a reactive fashion to acquire loop-free routes and maintain them as do other reactive routing protocols. However, it is differentiated from other existing on-demand routing protocols in that (i) the stored route information is validated before being used, and the up-to-date network topology is probed in an efficient way during NV creation, (ii) the routing states are managed in a timely fashion, (iii) a conservative route discovery strategy is adopted by suppressing route requests, and thus only a few routes are maintained per destination, (iv) address resolution is obviated by using a NIx and MAC addresses for routing purposes, and (v) a NV routing header is a compact form of a source route, which significantly reduces the packet overhead due to source routing. By virtue of these features, DNVR provides more stable and scalable performance with respect to network size, mobility, and traffic volume.

We showed via simulation that our method scales well to a large network under various scenarios especially with a high degree of mobility and a large volume of traffic. We compared DNVR to DSR through extensive simulations and showed that DNVR is as efficient as DSR in terms of normalized total overhead while achieving higher packet delivery and smaller packet latency in most cases. We believe that DNVR can be another design choice among the existing on-demand routing protocols for mobile ad hoc networks in that it is designed to provide scalability, high bandwidth efficiency as well as low overhead.

References

[1] S. Corson and J. Macker, "Mobile ad hoc networking (MANET): routing protocol performance issues and evaluation considerations," RFC 2501, IETF, January 1999.

[2] H.-Y. Hsieh and R. Sivakumar, "On using the ad-hoc network model in cellular packet data networks," *ACM MobiHoc 2002,* June 2002.

[3] D. Johnson and D. Maltz, "Dynamic source routing in ad hoc wireless networks," *Mobile Computing, edited by T. Imielinski and H. Korth, Ch. 5, pp. 153-181,* Kluwer Academic Publishers, 1996.

[4] C. Perkins and E. Royer, "Ad hoc on-demand distance vector routing," *IEEE WMCSA'99,* February 1999.

[5] Y.-C. Hu and D. Johnson, "Implicit source routes for on-demand ad hoc network routing," *ACM MobiHoc 2001,* October 2001.

[6] V. Park and S. Corson, "A highly adaptive distributed routing algorithm for mobile wireless networks," *IEEE INFOCOM'97,* April 1997.

[7] G. Riley, M. Ammar, and E. Zegura, "Efficient routing using NIx-Vectors," *IEEE HPSR 2001,* May 2001.

[8] D. Plummer, "An ethernet address resolution protocol: or converting network protocol addresses to 48.bit ethernet address for transmission on ethernet hardware," RFC 826, IETF, November 1982.

[9] J. Broch, D. Maltz, D. Johnson, Y.-C. Hu, and J. Jetcheva, "A performance comparison of multi-hop wireless ad hoc network routing protocols," *ACM MobiCom'98,* October 1998.

[10] D. Johnson, D. Maltz, and Y.-C. Hu, "The dynamic source routing protocol for mobile ad hoc networks (DSR)," Internet Draft, IETF, February 2003, work in progress.

[11] S. Das, C. Perkins, and E. Royer, "Performance comparison of two on-demand routing protocols for ad hoc networks," *IEEE INFOCOM 2000,* March 2000.

[12] C. Carter, S. Yi, and R. Kravets, "ARP considered harmful: manycast transactions in ad hoc networks," *IEEE WCNC 2003,* March 2003.

[13] G. Riley, "The Georgia Tech Network Simulator," *ACM SIGCOMM MoMeTools'03,* August 2003.

[14] http://www.ece.gatech.edu/research/labs/MANIACS/GTNetS

[15] IEEE Computer Society, "802.11: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications," June 1997.

[16] IEEE Standards Committee 754, "IEEE standard for binary floating-point arithmetic," ANSI/IEEE standard 754-1985, ANSI/IEEE, New York, 1985, reprinted in SIGPLAN notices, 22(2):9-25, 1987.