Imposed Route Reuse in Ad Hoc Network Routing Protocols Using Structured Peer-to-Peer Overlay Routing

Himabindu Pucha, *Student Member*, *IEEE*, Saumitra M. Das, *Student Member*, *IEEE*, and Y. Charlie Hu, *Member*, *IEEE*

Abstract—One of the most effective approaches to reducing the routing overhead in mobile ad hoc network routing protocols is to reuse routes discovered for one packet to deliver as many other packets as possible. Numerous techniques have been developed to facilitate route reuse, including caching, overhearing, using partial routes, and replying to route discovery with cached routes. We term such reuses *natural reuse*, as they are passive techniques and do not cause any detour in routing data packets. In this paper, we propose the Dynamic P2P Source Routing (DPSR) protocol which imposes additional reuse of routes by exploiting the synergy between mobile ad hoc networks and peer-to-peer overlay networks. By imposing reuse of a route to a faraway common destination node among a set of nearby source nodes using *localized* communication, DPSR limits the number of the source routes that each node has to discover to $O(logN)$, in contrast to the maximum of N source routes per node in DSR, where N is the number of nodes in the network. Our detailed simulations show that DPSR reduces the routing overhead significantly from imposed route reuse when many nodes communicate with one or a few other nodes. Obtaining the maximum benefit from imposed route reuse, however, requires sustaining traffic patterns beyond the capacity of current ad hoc networks. We conjecture that the full potential of DPSR will manifest as the capacity of ad hoc networks increases in the future from advances in physical layer technologies.

Index Terms—Wireless communication, network protocols, routing protocols, wireless.

1 Introduction

Due to the limited network capacity and available energy, one of the primary challenges faced by routing protocols for wireless mobile ad hoc networks (MANETs) is to minimize the routing overhead by reusing routes discovered for one packet to deliver as many other packets as possible. Further, route reuse is also crucial to reducing the routing delay and increasing the packet delivery ratio, consequently improving protocol scalability. Numerous techniques have been developed to facilitate route reuse. For example, in DSR [19], an on-demand routing protocol, a node can overhear packets being transmitted in its vicinity and utilize the source routes embedded in the packets for its own use. Similarly, the partial routes contained in a source route can be used to send packets with different destinations. We term such reuse *natural reuse* as they do not cause any detour in routing data packets. Moreover, all the previous route reuse techniques (in DSR and other protocols) are *passive*—they only exploit existing opportunities.

- *H. Pucha and S.M. Das are with the School of Electrical and Computer Engineering, Center for Wireless Systems and Applications, MSEE 212, 1285 Northwestern Ave., Purdue University, West Lafayette, IN 47907. E-mail: {hpucha, smdas}@purdue.edu.*
- *Y.C. Hu is with the School of Electrical and Computer Engineering, Center for Wireless Systems and Applications, MSEE 226, 1285 Northwestern Ave., Purdue University, West Lafayette, IN 47907. E-mail: ychu@purdue.edu.*

Manuscript received 15 Feb. 2005; revised 27 Sept. 2005; accepted 1 Nov. 2005; published online 25 Oct. 2006.
Recommended for acceptance by I. Stojmenovic.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number TPDS-0139-0205.

Recent developments in structured peer-to-peer (p2p) overlay networks such as CAN [28], Chord [30], Pastry [29], and Tapestry [35] show a highly scalable way of constructing and maintaining self-organizing, decentralized, and reliable overlay networks in the Internet. In a network of size N, when each of the N nodes communicates with the remaining $N-1$ nodes directly, the total number of distinct links is N^2. However, in a structured p2p overlay of N nodes, each node maintains direct links to $O(\log N)$ other nodes, and every node can reach every other node in $O(\log N)$ hops in the overlay. Effectively, a structured overlay introduces a virtual address space in which routing is performed, i.e., a packet is routed through $O(\log N)$ intermediate nodes in the overlay before reaching the destination node. Since the total number of distinct links in the routing tables of all the nodes is $O(N \log N)$, such a p2p overlay effectively reduces the number of links required for all pairwise communication from N^2 to $O(N \log N)$, suggesting that each link is reused $O(N/\log N)$ times.

The reduction of the routing state to $O(\log N)$ in a structured p2p overlay is obtained by sending a packet destined to any node in the network to one of the $O(\log N)$ nodes contained in the routing table. Each of these intermediate nodes contains state to route the packet further. This process is repeated till the packet reaches the destination. Furthermore, if the structured overlay is *proximity-aware* [6], the $O(\log N)$ nodes maintained at each node are among the closest candidates in the network proximity space. Thus, a packet from any node is routed to an intermediate node that is closest to the sender and contains further state to route the packet. If several nearby

nodes send packets to the same destination, all these packets will converge at a node near the source nodes that maintains further state to route the packet. This effect is termed *local route convergence* [7], [13]. When such a proximity-aware structured overlay is used in MANETs, it is expected to *impose* route reuse when several nearby nodes communicate with the same faraway node. In addition to passive natural reuse, this *active* technique of imposing route reuse can further reduce control overhead and potentially improve the scalability of routing in MANETs.

The primary challenge with using a structured p2p routing protocol as the network layer in MANETs is that p2p overlays in the wired Internet rely on the IP routing infrastructure to perform hop-by-hop routing between neighboring nodes in the overlays, whereas such an infrastructure does not exist in MANETs.

To overcome this problem, we propose the Dynamic P2P Source Routing (DPSR) [17] protocol which seamlessly integrates functions performed by a structured p2p overlay routing protocol (Pastry) operating in a logical namespace and by a MANET routing protocol (DSR) operating in the physical namespace. The key idea of the integration is to bring the proximity-aware structured p2p routing protocol to the network layer of MANETs via a one-to-one mapping between the IP addresses of the mobile nodes and their nodeIds in the namespace and replacing each routing table entry which used to store a (nodeId, IP address) pair in Pastry with a (nodeId, source route) pair. DPSR first routes the packets from nearby sources to a common faraway destination node to a common node nearby the sources, which effectively serves as a clusterhead and reuses the same route to the common destination node to route all the packets. In doing so, DPSR effectively *imposes* reuse of a long route for routing packets originated from nearby senders and going to a common destination. Since the route discovery for a long route is presumably more costly than for short routes, i.e., the routes to reach the common nearby node, such imposed reuse is expected to reduce the overall route discovery overhead. In essence, DPSR makes nodes use *localized* communication to route packets from multiple nearby senders to nearby clusterheads which then forward the packets to common faraway destinations and, in doing so, reduces the need for and, thus, the cost of flooding-based route discoveries.

Despite its apparent simplicity, implementing DPSR is not as straightforward as putting Pastry and DSR together. Pastry is designed for the Internet and has many features that work inefficiently in MANETs such as periodic pinging and a high overhead joining algorithm. Many of these features need to be redesigned for DPSR in order to be feasible for use in MANETs. In particular, the routing structures of Pastry and DSR need to be seamlessly integrated so that they can operate with an up-to-date view of the physical topology which changes dynamically due to mobility. Additionally, DSR route cache structures involve carefully tuned parameters to maintain the freshness of routes without compromising their availability. These mechanisms need to be carefully merged into the DPSR routing structures to optimize performance.

Apart from the implementation challenges, evaluating DPSR and demonstrating the benefit of DPSR from imposed route creates an exuberant challenge by itself. Similarly to the structured p2p overlays in the Internet, the existence and the extent of imposed reuse in DPSR critically depend on two factors. First, the communication pattern in the network needs to be many-to-many for maximum route reuse to happen. However, such traffic patterns are rarely simulated in ad hoc network routing protocol research (for example, [5], [9], [15], [18], [20]), primarily due to the high bandwidth consumption of such traffic patterns. Second, as the maximum route reuse in the presence of all-to-all communication pattern is $O(N/\log N)$, the benefit of imposed route reuse is expected to increase with the network size N. However, the vast majority of the simulation studies of routing protocols for mobile ad hoc networks have restricted their evaluation to a network size of around 100 nodes (for example, [5], [9], [15], [18], [20]), due to the capacity limitation of such networks [14] and also the scalability of the routing protocols.

We have performed a detailed simulation study of DPSR to evaluate its performance benefits from imposed route reuse. Our simulation results show that, for mobile networks with many-to-one or many-to-some traffic patterns, DPSR significantly reduces the routing overhead while providing comparable packet delivery performance to DSR. The benefit from DPSR increases with the network size. In addition, for the any-to-any traffic patterns widely used in previous protocol studies (e.g., [5], [9]), DPSR is comparable in performance to DSR. Although DPSR provides significant gains in the targeted application scenarios, we also find that sustaining the traffic patterns needed to demonstrate the *maximum* benefit of DPSR is beyond the capacity of current ad hoc networks. We conjecture that the full potential of DPSR will manifest as the capacity of ad hoc networks increases in the future from advancement in physical layer technologies.

The rest of the paper is organized as follows: Section 2 presents a summary of DSR and Pastry. Section 3 presents the design of DPSR. Section 4 describes the methodology used in the evaluation. Sections 6, 7, and 8 describe the numerous discoveries we encountered in comparing DPSR with DSR. Section 9 discusses application scenarios that can benefit from imposed route reuse of DPSR under the capacity limitation of current ad hoc networks. Finally, Section 10 discusses additional related work and Section 11 concludes the paper.

2 Background

The key idea of DPSR is to impose route reuse in multihop routing in MANETs by leveraging the high route reuse of structured p2p overlay routing protocols developed for the Internet. DPSR's design seamlessly integrates the DSR protocol for MANETs with a structured p2p overlay routing protocol, Pastry. In the following, we give a brief overview of DSR and Pastry.

2.1 DSR

DSR [19] is a representative multihop routing protocol for ad hoc networks. It is based on the concept of source

routing in contrast to hop-by-hop routing. It includes two mechanisms: *route discovery* and *route maintenance*.

Route discovery is the process by which a source node discovers a route to a destination for which it does not already have a route in its cache. The process broadcasts a ROUTE REQUEST packet that is flooded across the network in a controlled manner. In addition to the address of the original initiator of the request and the target of the request, each ROUTE REQUEST packet contains a route record, which records the sequence of hops taken by the ROUTE REQUEST packet as it propagates through the network. ROUTE REQUEST packets use sequence numbers to prevent duplication. The request is answered by a ROUTE REPLY packet either from the destination node or an intermediate node that has a cached route to the destination. To reduce the cost of the route discovery, each node maintains a cache of source routes that have been learned or overheard, which it uses aggressively to limit the frequency and propagation of ROUTE REQUESTS. The route maintenance procedure monitors the operation of the route and informs the sender of any routing errors. If a route breaks due to a link failure, the detecting host sends a ROUTE ERROR packet to the source, which, upon receiving it, removes all routes in its cache that use the hop in error.

Optimizations for the route discovery mechanism include: 1) The route discovery mechanism is made more efficient by caching overheard and forwarded routing information, 2) each node attempts to reply to ROUTE REQUESTS using its cached routes, thereby limiting the propagation of ROUTE REQUESTS, and 3) ROUTE REPLY storms caused by nodes replying from their caches are prevented by delaying the reply by a period proportional to the number of hops to the destination. This increases the probability that the source receives the shortest route first.

Optimizations for the route maintenance mechanism include:

1. every node helps maintain shorter routes by sending gratuitous ROUTE REPLYS if it knows of a shorter route to the destination than the one being used in the packet it overheard or forwarded,
2. each node always attempts to salvage a data packet that caused a ROUTE ERROR,
3. ROUTE ERROR packets received by a source node are piggybacked on its next ROUTE REQUEST to ensure increased spreading of information about stale routes, and
4. ROUTE ERROR packets that are forwarded or eavesdropped on are used to invalidate routes with broken links from the route cache.

The original design of DSR [5] uses a *path cache* which stores whole source routes. An alternative graph-based cache design, called a *link cache* [18], stores individual links of routes to build a topological graph of the network. This potentially increases the effectiveness of the cache since it enables DSR to construct routes (using the graph) that were neither overheard nor discovered.

Natural route reuse. DSR has a significant amount of natural route reuse. In DSR, a route could be discovered either *explicitly*, e.g., contained in the route replies to a flooding of explicit route requests, or *implicitly*, e.g., from overhearing control and data packets for neighboring nodes or from snooping and extracting source routes embedded in the data packets being forwarded. The implicit route discovery reuses routes discovered by other nodes. Further, each implicitly or explicitly discovered route is not only used as a full source route, but can also be used to send data packets to all the intermediate nodes along the full source route using the prefixes of the full source route, i.e., the embedded *partial routes*. An extreme example of route reuse appears in the graph-based cache of DSR [18] which allows additional routes to be constructed that are neither discovered nor overheard.

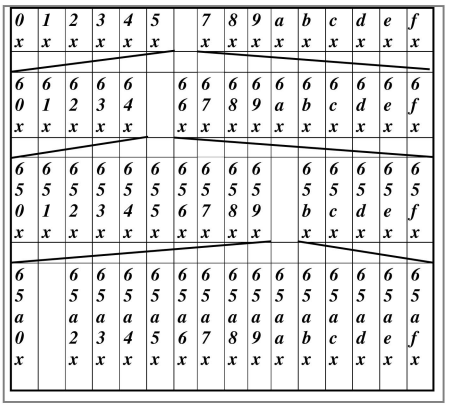

Fig. 1. Routing table of a Pastry node with nodeId $65a1x$, $b = 4$. Digits are in base 16, x represents an arbitrary suffix.

2.2 Pastry

Pastry [29] is one of several proximity-aware structured p2p routing protocols [6]. Each Pastry node has a unique, uniform randomly assigned *nodeId* in a circular 128-bit identifier space. Given a 128-bit key, Pastry routes an associated message toward the live node whose nodeId is numerically closest to the key.

Node state. For the purpose of routing, nodeIds and keys are thought of as a sequence of digits in base 2^b (b is a configuration parameter with typical value 4). A node's routing table is organized into $128/b$ rows and 2^b columns. The 2^b entries in row n of the routing table contain the IP addresses of nodes whose nodeIds share the first n digits with the current node's nodeId; the $n+1$th nodeId digit of the node in column m of row n equals m. The column in row n that corresponds to the value of the $n+1$th digit of the current node's nodeId remains empty. Fig. 1 depicts a sample routing table.

A routing table entry is left empty if no node with the appropriate nodeId prefix is known. The uniform random distribution of nodeIds ensures an even population of the nodeId space; thus, on average, only $\lceil log_{2^b} N \rceil$ levels are populated in the routing table. Each node also maintains a *leaf set*. The leaf set is the set of l nodes with nodeIds that are numerically closest to the current node's nodeId, with $l/2$ larger and $l/2$ smaller nodeIds than the current node's id. A typical value for l is approximately $\lceil 8 * log_{16} N \rceil$. The leaf set ensures reliable message delivery and is used to store replicas of application objects.

Message routing. At each routing step, a node seeks to forward the message to a node whose nodeId shares with

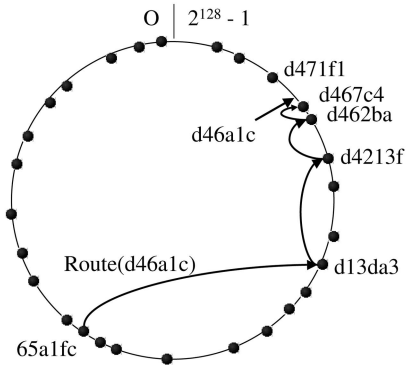

Fig. 2. Routing a message from node $65a1fc$ with key $d46a1c$. The dots depict live nodes in Pastry's circular namespace.

the key a prefix that is at least one digit (or b bits) longer than the current node's shared prefix. If no such node can be found in the routing table, the message is forwarded to a node whose nodeId shares a prefix with the key as long as the current node, but is numerically closer to the key than the current node's nodeId. Several such nodes can normally be found in the routing table; moreover, such a node is guaranteed to exist in the leaf set unless the message has already arrived at the node with numerically closest nodeId or its immediate neighbor. Further, unless all $l/2$ nodes in one half of the leaf set have failed simultaneously, at least one of those nodes must be live.

Fig. 2 shows the path of an example message. Analysis [7], [29] shows that the expected number of forwarding hops is slightly below $\lceil log_{2^b} N \rceil$, with a distribution that is tight around the mean. Moreover, simulation shows that the routing is highly resilient to node failures.

Node join. An arriving node with a newly chosen nodeId X initializes its state by contacting a nearby node A (according to the proximity metric) and asking A to route a special message with X as the key. This message is routed to the existing node Z whose nodeId is numerically closest to X. X then obtains the leaf set from Z, and the ith row of the routing table from the ith node encountered along the route from A to Z. One can show that with this scheme, X can correctly initialize its state and notify nodes that need to know of its arrival.

Proximity Awareness. Proximity-aware routing in a p2p overlay refers to the property that the network distance traversed by a packet in the overlay is within a small factor of the distance between the source and destination nodes in the underlying IP network. The factor is known as the *delay stretch*. Proximity-aware routing in Pastry is achieved by inclusion of physically nearby nodes in the routing table. Since the Pastry routing table is prefix-based, the upper levels of the routing table allow great freedom in this choice, with lower levels allowing exponentially less choice. As a result, the expected delay of the first hop is very low, it increases exponentially with each hop, and the delay of the final hop dominates. As one can show, this leads to low delay stretch and the *local route convergence* property [7]. Since a particular routing table entry refers to a nearby node with a particular prefix nodeId and since, in general, there are fewer and fewer nodes in a fixed area whose nodeIds have longer and longer prefix matches with that of a (destination) node, when several nearby nodes originate messages going to the same destination node, i.e., with identical keys, the paths taken by these messages tend to converge at a node physically close to the source nodes. Thus, this property allows local clustering to occur naturally when multiple nearby nodes send packets to the same faraway destination.

TABLE 1
A DPSR Routing Table or Leaf Set Entry

Destination	IP Address	Source Route
$< nodeId_x >$	$< IP_x >$	$< S_i....S_x >$

3 DPSR Design

In this section, we describe the design and operation of DPSR. Several changes have been made to the DPSR design originally proposed in our position paper [17]. For completeness, we describe the modified DPSR protocol below.

3.1 Basic DPSR Design

Like DSR, DPSR is proposed as a network layer protocol that runs on all nodes in the MANET. Message destinations and nodes are addressed using IP addresses. DPSR provides an added level of indirection to multihop routing in MANETs by assigning nodeIds from a circular name space to nodes in the MANET. A prefix-based routing scheme similar to Pastry is then employed to route data packets in the name space.

NodeId assignment. DPSR assigns unique nodeIds to nodes. To maintain the routing API as in DSR, nodeIds are generated by hashing the IP addresses of the hosts using collision-resistant hashing functions such as SHA-1 [10], thus obtaining, with high probability, a unique nodeId for each node in the network.

Node state. The routing table and leaf set of a DPSR node are organized in the same way as in Pastry, i.e., the routing table contains $\lceil \log_{2^b} N \rceil$ rows with $(2^b - 1)$ entries each, containing nodes that satisfy certain prefix-match requirement, and the leaf set contains l nodes whose nodeIds are numerically closest to that of the current node. Unlike the Internet, where routing is performed by the IP routing infrastructure, no such infrastructure exists in MANETs. Therefore, each entry in the DPSR leaf set and routing table stores a source route to reach the designated nodeId, as shown in Table 1.

As in Pastry, a routing table entry for any node K is chosen such that it is physically closer (based on the number of routing hops) than the other choices for that routing table entry. This is achieved by making use of the vast amount of implicitly discovered routes.

Node join. Similar to DSR, DPSR does not require any special initialization. This is a departure from Pastry, which has a node joining algorithm to initialize the routing table and leaf set. Since DPSR provides unicast routing as opposed to a DHT, it does not have consistency requirements for its leaf sets, unlike Pastry, which needs to maintain consistent leaf

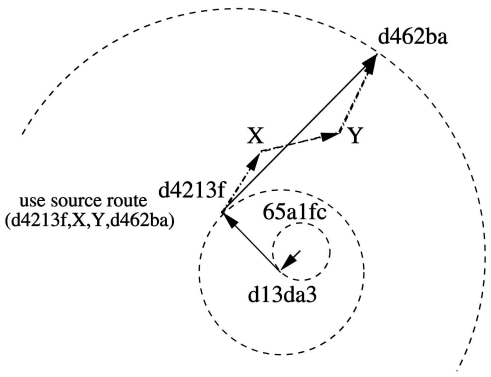

Fig. 3. DPSR routing in MANETs.

sets to provide consistent hashing. Routing in the presence of inconsistent leaf sets is discussed below. The joining node as well as all existing nodes in the network can populate and refresh their leaf sets and routing tables as data packets are forwarded or overheard.

Routing. In DPSR, since both message keys and nodeIds are hashed from IP addresses, an exact match between a message key and the destination node's nodeId is expected. In other words, a message will be delivered to the destination node whose nodeId matches the message key if that destination node is reachable via the wireless links.

Routing in the basic DPSR design is similar to Pastry: A message key is first generated by hashing the destination IP address, and the message is routed using prefix-based routing as described in Section 2.2. If an intermediate node cannot find a node that can make further progress in the name space, for example, due to an inconsistent leaf set, DPSR invokes a route discovery to discover a direct route to the destination node and patches its own leaf set.

Fig. 3 shows that each hop in DPSR routing is a multihop source route. Each data packet travels $\log N$ hops in the nodeId name space, and each of the hops is a multihop source route that travels longer and longer distance from using DPSR's proximity-aware prefix-based routing table construction. For example, the hop from node d4213f is via a source route (d4213f, X, Y, d462ba). This is in contrast to Pastry, where each hop is a multihop Internet route. For clarity, in the rest of the paper, we term each hop in DPSR routing in the nodeId name space a *logical hop*.

3.2 Optimizations

The basic design of DPSR inherits all of the optimizations on route discovery and route maintenance used by the DSR protocol such as caching overheard routing information, caching routes snooped from forwarded packets, replying to route requests using cached routes, caching partial routes from individual routes, and route shortening. In addition, a number of optimizations are unique to the DPSR routing structures and operations.

Short-cut direct routes. In addition to the "prefix-based view" of the routing table or the "neighbor-node view" of the leaf set, the DPSR routing table and leaf set can be viewed as two caches of source routes, similar to the route cache in DSR. This allows the use of *direct source routes* to destinations, as in the DSR protocol. To send a data packet,

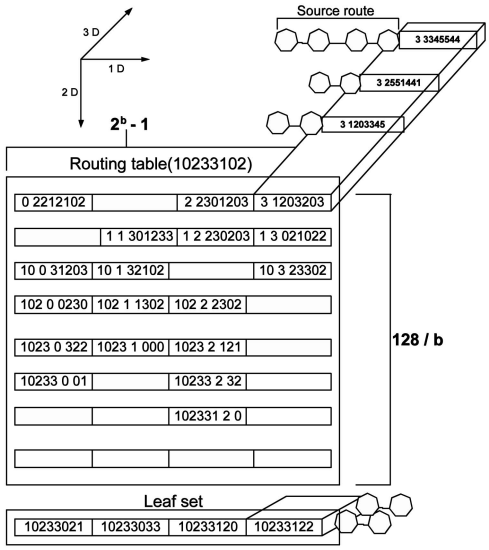

Fig. 4. DPSR's 3D routing table and 2D leaf set. Dimensions 1, 2, and 3 are depicted. Each chain of circles depicted represents a source route.

DPSR first searches for a direct route in the routing table and the leaf set for an exact match with the message key (i.e., destination nodeId). If this initial search returns a source route, DPSR uses it directly. Otherwise, the original DPSR prefix-based routing is executed to return the source route to the next logical hop.

3D routing table and 2D leaf set. To further extend the above idea of using the leaf set and the routing table as route caches, the leaf set can be extended to 2D and the routing table extended to 3D, i.e., each entry in the leaf set or the routing table contains a vector of N_l and N_r routes, where N_l and N_r are configuration parameters. This structure is shown in Fig. 4. The replacement algorithm used in each leaf set entry and each routing table entry is Least Recently Discovered (LRD), disregarding whether a route is discovered explicitly or implicitly. When looking up a route from a leaf set or routing table entry, the freshest among the shortest routes in that 1D entry is returned.

Routing state update using implicitly discovered source routes. In the basic operation of DPSR, a node always chooses the freshest among the shortest routes explicitly discovered (through route discovery) for each entry. As an optimization, a node adds every route implicitly discovered (through snooping and overhearing) as well as partial routes from implicitly and explicitly discovered source routes to the corresponding entry in either the leaf set or the routing table based on matching of the nodeIds. This optimization thus constantly discovers fresh and low proximity routes for the leaf set and the routing table entries.

Passive routing table proximity maintenance. The original Pastry routing table maintenance algorithm is designed to preserve the local proximity of routing table entries in the presence of network dynamics. However, periodic exchange of routing table rows and subsequent proximity probing is a very high overhead exercise in MANETs, for example, the route to the probed node may need to be discovered. The nature of the shared medium access of MANETs provides an efficient alternative. A node can use

TABLE 2
Pseudocode for the DPSR Routing Algorithm when a Packet p Arrives at a Node

```
SEND_PACKET(p)
(1)  if (direct source route to p.dest exists) // search routing table and leaf set as a cache of source routes.
(2)      forward p to p.dest using source route and return
(3)  // invoke prefix-based routing to find next logical hop entry.
(4)  entry = LOOKUP(hash(p.dest)) // p.dest is hashed to obtain the message key
(5)  if (entry == NULL) // leaf set is inconsistent
(6)      invoke fboding-based route discovery for p.dest and send packet later, patch leaf set L
(7)  else if (entry ∈ R and entry.valid_routes == 0) // routing table entry has no source routes
(8)      invoke prefi x-based route discovery for prefi x and send packet later
(9)  else if (entry ∈ L and entry.valid_routes == 0 ) // leaf set entry has no source routes
(10)     invoke fboding-based route discovery for entry and send packet later
(11) else
(12)     chosen_route = BEST_ROUTE(entry) // selects the freshest among shortest of the source routes in the 1-D array
(13)     forward p to next logical hop using chosen_route

LOOKUP(key)
(1)  if (key isBetween(L_{-l/2}, L_{l/2})) // key is within range of local leaf set
(2)      if there exists L_i such that |key − L_i| = 0, return L_i
(3)      else return NULL
(4)  else // use the routing table
(5)      l = shl(key, own_nodeID)
(6)      if (R_l^{key_l} exists) return R_l^{key_l}
(7)      else return t ∈ L ∪ R, such that shl(t, key) ≥ l ∧ |t − key| < |ownNodeID − key|
```

overhearing of routes and routes from forwarded messages to maintain the locality of its routing table entries. In fact, the nature of the overhearing process guarantees that the routes overheard contain many physically nearby nodes. Regular updates to routing table entries using overheard routes and routes from forwarded messages can make DPSR resilient to degradation of routing table quality due to mobility. Our LRD replacement policy in updating the routing table entries ensures that the cache reflects the most current view of the network.

Prefix-based route discovery. A further optimization is added to DPSR to reduce the cost of route discovery for routing table entries. When a routing table entry has no valid routes, a modified route discovery is performed to discover routes to any nodes whose nodeIds match the prefix for that routing table entry. This is in contrast to performing route discoveries to a particular destination node known to have a prefix match in its nodeId. This approach is more efficient in MANETs because route discovery in DSR and DPSR is flooding-based and, thus, is more efficient at finding some node with a proper prefix match (since there are multiple such nodes) than finding a particular node in DPSR. Prefix-based route discoveries use expanding ring search to localize the discovery. An additional advantage of prefix-based route discoveries is that they allow for low overhead routing table maintenance even if the promiscuous overhearing mode is turned off to conserve energy.

Logical hop salvaging. When a node in DSR encounters a link break while delivering a data packet for another node, it tries to *salvage* the packet by searching in its route cache for an alternative route for the packet. However, if such a route does not exist, that node simply drops the packet. DPSR, in addition to searching for a direct (short-cut) route to the packet's destination, also attempts to send the packet to the next logical hop. Only if valid routes do not exist in both cases, the packet is dropped by DPSR. Thus, the probability of successfully salvaging a data packet is potentially higher than in DSR.

3.3 DPSR Operations: Putting It All Together

In this section, we summarize the operations of DPSR in a mobile network in pseudocode in Table 2. The procedure SEND_PACKET is invoked at node A when a packet p arrives at node A. We begin by defining some notations. $p.dest$ is the IP address of the destination node. The routing table is denoted as R and the leaf set as L. Since L is 2D and R is 3D, each entry of L or R is a 1D array of source routes. R_l^i is the entry in the routing table R at column i and row l. L_i is the ith closest nodeId in the leaf set L, where a negative/positive index indicates counterclockwise/clockwise from the current node in the id space, respectively. $L_{-l/2}$ and $L_{l/2}$ are the nodes at the edges of the current node's leaf set. key_l represents the lth digit in the key. $shl(a, b)$ is the length of the prefix shared among a and b, in digits.

Given a packet p, node A first searches its routing structures for a short-cut direct route to the destination node B. If such a source route is found, the packet is sent to node B directly using its corresponding IP address and source route. If no direct source route is found, prefix-based routing is invoked and the routing structures are looked up for the next logical hop using the LOOKUP function. The LOOKUP function returns an entry (1D array of source routes) from either the routing table or the leaf set. When the entry has valid source routes, the freshest among the shortest is chosen as the route to forward the packet. However, due to the mobility of nodes, it is possible that no

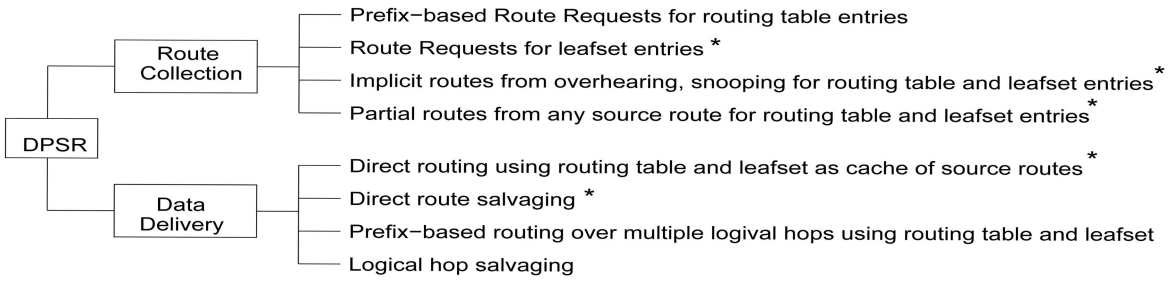

Fig. 5. Summary of DPSR features.

valid source routes currently exist for an entry. In this case, if the entry belongs to the routing table, a prefix-based route discovery is initiated and, if the entry belongs to the leaf set, a flooding-based route discovery is initiated.

Once a packet has a source route, it is forwarded using the source route (without invoking SEND_PACKET) till it reaches the next logical hop (end of the source route). When the packet reaches the logical hop node, if the destination of the packet matches the IP address of the node, the packet is delivered to the transport layer. Otherwise, SEND_PACKET is invoked once again at this logical hop node. Note that, in case a route error happens during forwarding of a packet using a source route, the forwarding node attempts to salvage the packet using short-cut direct routes and logical hop salvaging.

Fig. 5 summarizes all the different features present in DPSR. The features marked with an asterisk are techniques from DSR that have been tailored to and folded into DPSR.

Effect of configuration parameters on DPSR performance. DPSR has two important configuration parameters: b (logical hop control parameter) and l (leaf set size). Parameters l and b control the trade-off between the routing table size and the number of logical hops. In DPSR, each node maintains $\lceil \log_{2^b} N \rceil * (2^b - 1)$ entries in the routing table and, consequently, can route in $\lceil \log_{2^b} N \rceil$) hops. As b is reduced, the number of hops is increased. In a MANET environment, shorter hops are always better since they reduce the delay as well as the probability of error in delivering a packet. Thus, b should not be very small. On the other hand, a large value of b can increase the routing state. l also affects the number of hops traveled by a message although to a smaller extent than b. l denotes the number of destinations for which a node will be used as a clusterhead since a node may potentially be used as a clusterhead for any destination in its leaf set. A small l inhibits route reuse, whereas a large l can potentially create imbalance by causing one node to be a clusterhead for many destinations. Experimentally, we found that routing through two logical hops (going through one clusterhead) gave the best routing performance and to do so we need $\log_{2^b} N \leq 2$ and $\frac{N}{2^b} \leq l$. Thus, $b = 5$ and $l = 32$ works well for a wide range of network sizes up to 1,000 nodes. However, for smaller networks (up to 200 nodes), $b = 4$ and $l = 16$, which result in smaller routing tables, can also be used.

Two other parameters required in DPSR are N_r and N_l, the size of the 1D array of source routes in the routing table and leaf set entry, respectively. These values should be large enough to accommodate enough source routes and also small enough to evict stale entries in time. Experimentally, we found that values of $N_r = 10$ and $N_l = 5$ provide a good trade-off between the availability of routes and their freshness.

Effect of configuration parameters on DPSR memory usage. The memory usage of the DPSR routing table depends on the configuration parameter b since it is a matrix of $2^b - 1$ columns and, on average, $log_{2^b} N$ rows. Each row contains N_r pointers to source routes. Each source route is a vector of 4-byte IP addresses of maximum size M. So, the average size of the routing table is $2^b \cdot log_{2^b} N \cdot N_r \cdot M \cdot 4$ bytes. Thus, assuming each entry of the matrix has all $N_r = 10$ routes of maximum size $M = 15$ each and for a network of 4,096 nodes, the routing table size for $b = 4$ is 28.8 Kbytes. Similarly, the memory usage of the 2D leaf set is given by $l \cdot N_l \cdot M \cdot 4$ bytes. Thus, in the worst case, the leaf set size for $N_l = 5$ and $l = 16$ is 4.8 Kbytes.

4 Methodology

4.1 Simulators

We first implemented DPSR in ns-2 [4]. This implementation is used in the evaluations in Sections 5, 6, and 7. ns-2 simulates a radio model with a nominal bit-rate of 2Mbps and a transmission range of 250m. The link layer used is IEEE 802.11. However, the limited scalability of ns-2 prevented us from using it for performing simulations with large network sizes and traffic volume.

To measure the performance of DPSR and DSR in large networks (discussed in Section 8), we also implemented DPSR in Glomosim [34]. The functionality of Glomosim is similar to that of ns-2. However, Glomosim scales to a larger number of nodes in both memory requirements and simulation execution time. Further, a radio model with a nominal bit-rate of 11Mbps and a transmission range of 250 m is used in the simulations. The link layer used is IEEE 802.11.

4.2 Mobility and Traffic Models

The mobility scenarios used in the simulations are generated using a modified form of random waypoint model [32], [33] that avoids the speed decay problem. In our simulations, nodes move at speeds distributed uniformly between 1 and 19 m/s. The mobility of the nodes is also governed by a *pause time*, which is the time a node rests in between movements. Constant bit rate (CBR) traffic sources are used to accurately evaluate the routing performance. A packet size of 64 bytes is used. We perform simulations for different network sizes. As the network size is increased,

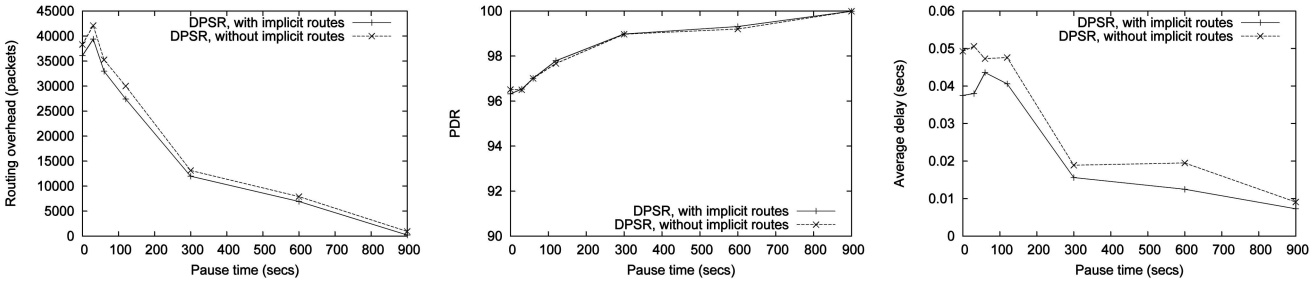

Fig. 6. Routing overhead, PDR, and delay comparison of DPSR with and without direct routes.

the area in which nodes are distributed is scaled such that the node density is kept constant. Both DSR and DPSR maintain a send buffer of 50 packets. The network area and density chosen for all experiments result in the network being connected most of the time.

4.3 Metrics

The following widely used metrics for MANET routing protocols are evaluated: 1) *routing overhead*—the number of control packets transmitted, with each hop-wise transmission of a control packet counted as one transmission, 2) *packet delivery ratio* (PDR)—the ratio of the data packets delivered to the destinations to those generated by the CBR sources, and 3) *average delay*—the end-to-end delay of packet routing which accounts for all possible delays caused by buffering during route discovery process, queuing at the interface queue, retransmissions at the MAC, and propagation and transfer through the channel.

5 Impact of DPSR Design Choices

We first evaluate the performance impact of two design choices: the use of direct routes (Section 3.2) and the choice of the logical hop control parameter b.

5.1 Use of Direct Routes

Since DPSR routes a packet through several logical hops whereas DSR takes the direct paths, if queuing delay is discounted, the routing delay using DPSR is expected to be longer than using DSR. However, the proximity-awareness of Pastry from prefix-based routing ensures that the logical hops taken in routing a message are longer and longer, and the last hop dominates. As a result, the routing paths taken by Pastry routing are about 50-100 percent longer than the direct IP path between two nodes in the Internet [7]. Since DPSR shares the prefix-routing nature with Pastry, the average routing path taken by DPSR is expected to also be within 50-100 percent of the shortest path. The use of direct routes in DPSR which short-cut logical hops further reduces the delay in packet delivery. Fig. 6 compares the performance of DPSR with and without the use of direct routes. We ran simulations for a 50-node network in an area of 1,500 m $\times$ 300 m with 40 CBR sources generating three packets/second each. Each source makes a maximum of two connections. Though the PDRs of DPSR in both cases are similar, the use of direct source routes reduces the average delay as well as the routing overhead for all pause times. Although the gains are not very significant in this scenario, we found that the gains from direct routes increases as the network size increases. Thus, in the rest of the paper, we use DPSR with direct routes unless otherwise stated.

5.2 Logical Hop Control Parameter: b

The value of b (base of Pastry's nodeId digits) controls the trade-off between the number of entries in the routing table and the average number of hops traversed by a message to reach its destination. Table 3 shows how the choice of b affects the performance metrics for a network of 200 nodes, in which 40 sources are sending one packet/second each. A pause time of 300 seconds is used. As b varies from 2 to 4, the average number of logical hops decreases from 2.4 to 1.85. The shortened logical hops in this case increase the PDR and reduce the delay and routing overhead. The use of direct routes, however, significantly diminishes the effects of the choice of b. Table 3 shows that, with direct routes, DPSR with $b = 4$ achieves comparable PDR as DPSR with $b = 2$, although it reduces the routing overhead and delay by about 20 percent. Also note that the gains from direct routes have increased in a 200 node network as compared to a 50 node network in Fig. 6. In the rest of the paper, we use $b = 4$ unless otherwise stated.

6 Initial Experience ("DPSR versus DSR")

We begin our evaluation study with a network size of 50 nodes as this network size was used in all previous studies of DSR [5], [9]. These previous protocol comparison studies also assume a traffic pattern where each node sends packets to one or two other nodes in the network. However, since imposed reuse in DPSR will only exist when each

TABLE 3
Routing Overhead, PDR, and Delay Comparison of DSR and DPSR Varying b

	Logical hops	Path length	Routing overhead	PDR	Delay (sec.)
DSR	—-	5.25	252771	72.07	0.2869
DPSR					
b=2	2.40	6.65	257412	66.35	0.2755
b=3	2.11	6.40	113798	85.23	0.1124
b=4	1.85	6.04	92896	85.20	0.0622
DPSR/DR					
b=2	1.38	5.53	91006	85.03	0.0523
b=3	1.27	5.36	86622	85.33	0.0427
b=4	1.16	5.28	75484	84.57	0.0395

DPSR and DPSR/DR refer to DPSR without and with direct routes, respectively.

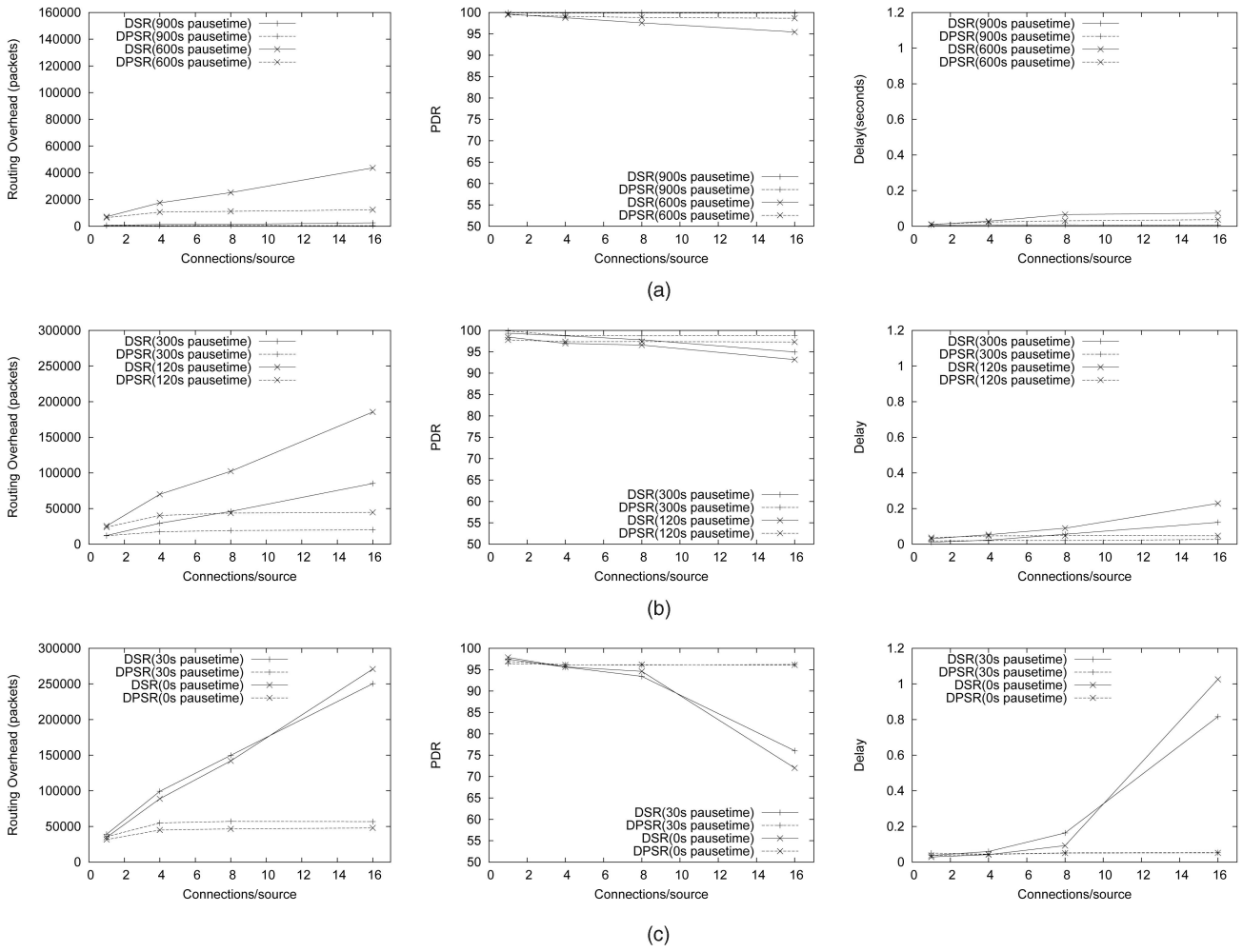

Fig. 7. Routing overhead, PDR, and delay comparison of DPSR and DSR varying the number of data connections. (a) Low mobility, (b) medium mobility, and (c) high mobility.

node communicates with many other nodes, we vary the number of connections maintained by each source node, denoted as a parameter X.

6.1 Experimental Setup

In this section, we use ns-2 [4] to compare the performance of DPSR with DSR. The experiments used a 50-node network with 40 CBR sources and a traffic volume of 120 packets/sec. The nodes are distributed in a rectangular area of dimensions 1,500m $\times$ 300m. The number of connections per source, X, is increased from 1 to 16. Additional higher values of X were simulated, but are not shown as the trend is evident for lower values of X. Also, as X is increased, the packet rate per connection is reduced proportionally so that the traffic volume remains constant. The volume is kept constant to keep the multiaccess interference constant for all scenarios.

6.2 Performance Results

Fig. 7 shows that DPSR performs as well as DSR when $X = 1$. This is slightly unexpected since DPSR routes in multiple logical hops leading to increased transmissions of data packets in comparison to DSR which uses direct routes. However, optimizations such as prefix-requests and direct routes reduce the overhead of discovering routes to logical hop nodes. Additionally, salvaging using the logical hops also helps the packet delivery of DPSR.

Surprisingly, as X is increased from 1 to 16, the overhead of DSR increases almost linearly, while DPSR incurs a constant routing overhead, outperforming DSR by a large margin for large values of X. This is true for all mobilities. As a result, DPSR maintains a PDR close to 100 percent for all mobilities, while the PDR for DSR drops as the number of connections increases and, the higher the mobility, the lower the PDR. Similarly, for each mobility, as the number of connections increases, DPSR maintains a constant delay, while the delay grows significantly in DSR.

6.3 Analysis

The above widening performance gap between DPSR and DSR is unexpected as DPSR is expected to have a higher reuse than DSR only when $X > l + 16 \log_{16} N$ since DSR uses up to $N \cdot X$ distinct source routes and DPSR requires up to $N \cdot (l + 16 \log_{16} N)$ distinct source routes. To understand the reason, we measured the amount of route reuse in both protocols. This is done by modifying ns-2 to include identifiers in routes stored in the cache to identify unique routes, followed by postprocessing of the route usage at all

TABLE 4
Route Reuse Comparison of DPSR and DSR

	DSR	DPSR	
		no direct routes	direct routes
Total routes used	100191	147569	112043
Distinct routes	49913	35972	46593
discovered	35773	4174	3113
overheard	14140	31798	43480

the nodes so as to measure how many distinct routes were used while delivering data packets. Table 4 shows the amount of route reuse for the case of 16 connections per source and a pause time of 120 seconds. The results show that the number of routes that are discovered in DSR are 10 times more than in DPSR with direct routes. Since both protocols use source routing and similar overhearing optimizations, it immediately becomes suspicious that the ratio of overheard routes over discovered routes used in DPSR is over 20 times higher than in DSR. A closer look at the cache structure of DSR reveals why this occurred.

DSR as proposed in [9] and implemented in ns-2 maintains a generational capacity limited cache for discovered and overheard routes. The capacity limitation is present to evict potentially stale routes. When X is increased, ROUTE REQUESTS for one destination result in many ROUTE REPLYS which evict the routes already present in the cache. Thus, for subsequent packets, capacity misses occur in the cache, leading to a high number of discovered routes and a low number of overheard routes, as shown in Table 4. Since DPSR uses a prefix-based routing table which stores routes that match a prefix separately from routes that match another prefix, such capacity misses do not happen, resulting in a much higher usage of overheard routes.

In summary, the observed performance advantage of DPSR over DSR is not because of better route reuse due to structured p2p routing, but rather a decreased route reuse in DSR due to capacity misses in its route cache.

7 Second Experience ("DSR-Graph Strikes Back")

To isolate the limited natural route reuse of DSR due to the cache capacity and measure benefit of imposed route reuse in DPSR due to structured p2p routing, we switched to using the graph-based cache version of DSR proposed in [18] which we term DSR-Graph. Since this version of DSR builds a topological network graph, it does not suffer from capacity misses as does the DSR version using a path cache.

7.1 Experimental Setup

It has been shown in [18] that DSR-Graph outperforms DSR for almost all scenarios due to its ability to use the graph to discover routes that are not explicitly discovered. To enable a fair comparison with DPSR, we implemented a graph-based cache for use in DPSR. We denote this version of DPSR as DPSR-Graph. Specifically, in DPSR-Graph, we construct a graph $G(V, E)$ of the network topology in the physical space. Each vertex is referenced by its nodeId. Additionally, we store the IP address of the node in the vertex so that source routes can be constructed using Dijkstra's single source shortest path algorithm to any node in the network that can be reached in $G(V, E)$. Similar to DSR-Graph, we use a graph-based cache that has an adaptive timeout mechanism for expiring links based on the stability of the endpoint nodes of that link. The stabilities of the endpoint nodes of a link are increased by an additive factor whenever the link is used as part of a source route and are decreased by a multiplicative factor whenever a link breaks. The timeout of any link is chosen as the minimum of the stability values of its endpoints. Routes in the topological graph that are of the shortest lengths and have the highest minimum timeout value of any of their contained links (largest lifetime) are constructed using Dijkstra's algorithm.

In this section, we use the same parameters as in Section 6.1 to perform a four-way comparison among DPSR, DPSR-Graph, DSR, and DSR-Graph.

7.2 Performance Results

Fig. 8 depicts the routing overhead, PDR, and delay comparison between the four protocols. The key observations from the figure are as follows.

First, DSR-Graph significantly outperforms DSR and slightly outperforms DPSR; it has lower overhead and delivers more packets than either of the two protocols. This is because it enjoys more reuse than both DPSR and DSR. DSR-Graph has more route reuse from its cache than DSR as it has no capacity misses, and it has more route reuse than DPSR due to construction of routes from the graph which have not been overheard or discovered.

Second, DSR-Graph and DPSR-Graph have similar performance for all values of X. Both protocols exhibit low overhead and high delivery ratios with acceptable delay performance. This shows conclusively that the gain of

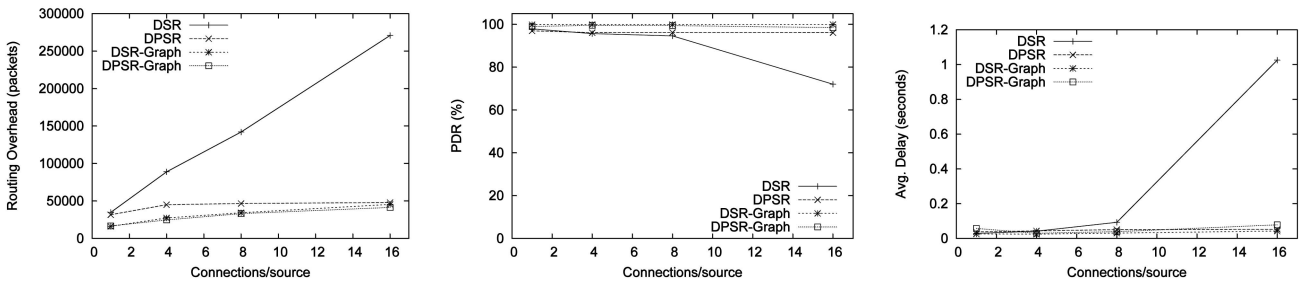

Fig. 8. Routing overhead, PDR, and delay comparison of DPSR and DSR varying the number of data connections for high mobility (pause time 0s).

TABLE 5
Comparison of DPSR-Link and DSR-Link for All Pairs Communication for a 50-Node Network

	DSR-Link	DPSR-Link
Routing Overhead	62449	50225
PDR	97.29	94.21
Average Delay	0.08	0.1

DPSR when compared to DSR was solely due to the design of DSR's route cache which caused capacity misses.

To simulate the scenario where each node sends packets to every other node, we simulated $X = 50$. The results, shown in Table 5, show that, in this scenario, DPSR-Graph performs better than DSR-Graph. It delivers almost the same amount of packets, but with 20 percent lower overhead and similar delay.

7.3 Analysis

The above outcome is explained by the many factors that are involved in the comparison. We first analyze the route usage in DPSR-Graph and DSR-Graph in the worst-case scenario when no natural reuse occurs.

In DPSR, when N nodes send packets to N nodes, the exact number of routes used is $N \cdot (16 \cdot \log_{16} N + l)$, where l is the size of the leaf set. For the leaf set size 16, the number of routes used becomes $50 \cdot (16 \cdot \log_{16} 50 + 16)$, which is approximately $50 \cdot 48$. This is very close to N^2, which is $50 \cdot 50$.

When natural reuse occurs, it benefits both DSR-Graph and DPSR-Graph and, thus, the number of routes discovered is actually much less than the theoretical numbers above, i.e., 2,500 multiplied by the number of packets sent by each node. For example, DPSR-Graph performs only 760 discoveries and DSR-Graph performs 3,255 route discoveries during the entire simulation. The extent of natural reuse is further dictated by other parameters such as the hearing range of nodes and the density of nodes considered.

In summary, because of the effects of constant factors in the analysis and the effect of natural reuse, we need to simulate a larger network when the imposed route reuse in DPSR-Graph is not overshadowed by the natural reuse common to both protocols.

8 Third Experience ("Return of DPSR-Graph")

In this section, we compare DPSR-Graph and DSR-Graph in large networks using the Glomosim simulator.

8.1 The True Potential of DPSR

Before proceeding with experiments with large networks, we summarize the different ways the potential benefits of imposed route reuse of DPSR can be affected.

- **Traffic pattern.** As discussed in Section 1, for imposed route reuse to be maximum, the communication pattern in the network needs to be many-to-many, similarly to Pastry.

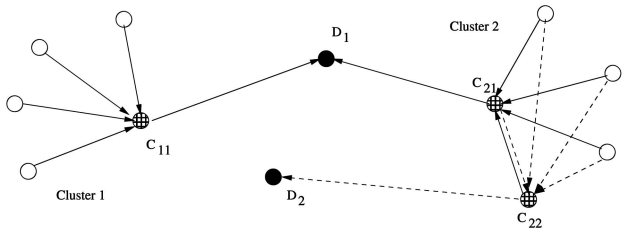

Fig. 9. Local route convergence leads to reuse in DPSR. For simplicity, we assume routing takes two logical hops. C_{ij} denotes the clusterhead for destination j in cluster i.

- **Network size.** The network size directly affects the extent of imposed reuse in DPSR. We analyze this factor in the worst-case scenario when no natural reuse occurs. In DPSR, when N nodes send packets to N nodes, the exact number of routes used is $N \cdot (2^b \cdot \log_{2^b} N + l)$, where l is the size of the leaf set. Since DSR can potentially use up to N^2 distinct routes, the benefit of imposed reuse in DPSR over DSR increases with the network size N as $N/(2^b \cdot \log_{2^b} N + l)$.
- **Packet arrival timing.** The timing of packet arrival also has a direct impact on imposed route reuse. Fig. 9 depicts an example of route convergence and reuse in DPSR routing. It shows two physically nearby groups of nodes which form two clusters. Cluster 1 has one clusterhead C_{11} for destination D_1. Cluster 2 has two clusterheads, one each for destinations D_1 and D_2. Consider the case when all the nodes in Cluster 1 communicate with D_1. Since C_{11} matches the prefix with D_1, it becomes the clusterhead for destination node D_1. Thus, once C_{11} discovers a route for D_1, it can potentially reuse this route to deliver the packets destined for D_1 from any node in Cluster 1. However, this reuse is only possible if all the packets from other nodes in Cluster 1 arrive at C_{11} before route $C_{11} \rightarrow D_1$ breaks.
- **Interference of natural reuse.** The benefits of imposed reuse, i.e., the potential reduction in route discoveries, can potentially be shadowed by various forms of natural route reuse that already exists in DSR.
- **Cost of detour.** The cost of reaching the clusterhead, e.g., during the first $(\log N - 1)$ logical hops before reaching the last hop clusterhead, can potentially add to the routing overhead and increase the routing delay.

In summary, to demonstrate the performance benefit of imposed route reuse in DPSR requires the presence of many-to-many traffic patterns in a large network, and the packets going through the same "clusterheads" need to arrive before the route from that clusterhead to the destination breaks. However, supporting many-to-many traffic patterns in a large network places a high demand on the capacity of the wireless network which is beyond the capability of the current physical layer interference model [14], [21]. To restrict the traffic volume while demonstrating the potential of imposed route reuse in DPSR, we chose a traffic pattern where each node communicates with C ($C < N$) other nodes simultaneously.

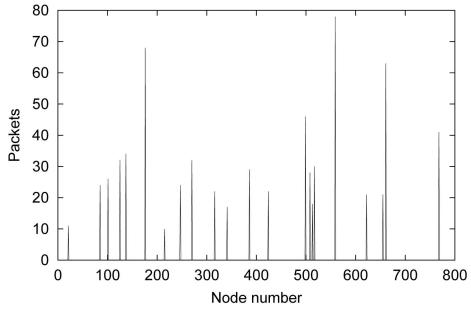

Fig. 10. Number of packets going through each logical hop, i.e., the clusterhead node, for DPSR-Graph in a network of 800 nodes.

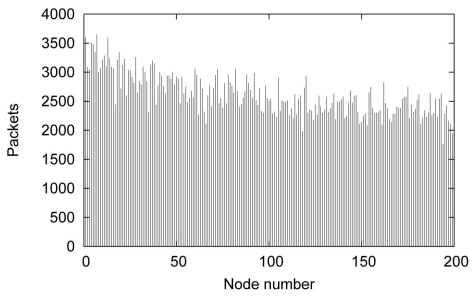

Fig. 11. Number of total packets transmitted by each node for DPSR-Graph in a network of 200 nodes.

8.2 Experimental Setup

To enable simulations of large networks, we implemented DPSR-Graph and DSR-Graph in Glomosim, which originally only supported the basic DSR. The traffic pattern where every node initiates connections to a subset of C nodes is used. The network size is increased from 200 to 800 nodes and a value of $b = 5$ is used because of the larger network sizes. A square area is chosen for each network size such that, on average, there are 10 neighboring nodes within the broadcast range of each node. For all simulation scenarios, a pause time of 0s was used. The maximum speed used was 19 m/s, but, for scenarios when the network capacity is not sufficient to support the traffic volumes reached, a maximum speed of 9 m/s was used. The simulation ran for 100 seconds with each node initiating one data packet to each of C nodes within every 2 seconds.

8.3 Reuse Analysis

To demonstrate the benefit of imposed reuse and evaluate the cost of prefix requests, we simulated DPSR-Graph for a network of 800 nodes with the traffic pattern described above, but with $C = 1$. All the remaining parameters are as described above. We also disabled the use of direct routes in DPSR-Graph since direct routes pose an opportunity of natural reuse to DPSR-Graph which obscures its ability to impose route reuse.

Since $C = 1$, let us consider how packets from different nodes are routed to the single chosen destination. The number of nodes starting with a unique digit is, on average, $N/32$ ($N/2^b$) as the nodeIds are uniformly hashed. Thus, each node, while attempting to reach a destination, will route the packet to one of these $N/32$ nodes which match the first digit with the destination. Moreover, due to the proximity-aware routing table construction, each node picks the node that matches the prefix from its neighborhood. This creates the scenario as depicted in Fig. 9, where each of $N/32$ nodes acts as a clusterhead for all the packets destined to the same destination initiated by nearby nodes. Each of these $N/32$ nodes now finds a route to the final destination and tunnels all the packets toward the destination. Thus, the routes found by the $N/32$ clustering nodes are reused by all the other nodes in the network. This imposed reuse is measured and depicted in Fig. 10. The figure shows the number of packets that are received by each logical hop node to be forwarded to the destination (the next logical hop). As seen in the figure, a fraction of the nodes inherently act as clusterheads due to the prefix-based routing. Also, the figure shows that the load (number of packets arriving at each clusterhead) is distributed fairly evenly among the clusterhead nodes.

An interesting observation from Fig. 10 is that, although the load among the clusterheads is fairly evenly balanced, the clusterheads themselves appear to be at a disadvantage (transmitting more packets) compared to other noncluster-head nodes. However, this occurs because the traffic is many-to-one in nature. We conjecture that the load will be better balanced in a more general many-to-many communication scenario. To study the load distribution further for this general pattern, we simulated a 200-node network in which every node sends a data packet to every other node in the network. The network size is reduced to support the increased traffic from many-to-many communications used in this experiment. Fig. 11 shows the number of packets (control and data) transmitted by each node. As seen from the figure, the load is fairly evenly distributed for this general traffic pattern. This is because: 1) for a particular destination, different regions in the network use different clusterheads and 2) for different destinations, the same region has different nodes as clusterheads. Thus, DPSR provides an even load distribution for general traffic patterns.

We now analyze the routing overhead in this scenario. Since prefix-based routing is directionless, i.e., the first $(\log N - 1)$ logical hops could be along any direction and, thus, on average, not making any progress toward the final destination, the last hop from the clustering node to the final destination usually is as long as any average route in the network. Thus, the overhead of DPSR-Graph in this scenario is $\frac{N \cdot (Cost_{RREQ})}{32} + RO_{1hop}$, where $Cost_{RREQ}$ is the overhead incurred per route request (RREQ), and RO_{1hop} represents the overhead incurred by all the nodes to reach the clustering nodes, which, on average, takes one hop in this scenario. In DPSR-Graph, the routes to the neighboring clusterheads can be obtained with no overhead via overhearing or snooping on the forwarded routes. If a route to the clusterhead or the clusterhead itself is not known, the node initiates a prefix route request (PRREQ).

A prefix route request essentially returns the closest node which is a valid first logical hop toward the destination. The prefix route request is a variable TTL search. A node first initiates a one-hop broadcast and retries with a maximum TTL if no reply is received. Each intermediate node that receives the prefix route request searches through its cache to return a valid first logical hop.

TABLE 6
Breakdown of the Overhead Incurred in the First Logical Hop

	Initiated	Transmissions
PRREQs with TTL =1	30	30
PRREQs with MAX TTL	12	1092
PREPS	469	2918

PREP stands for prefix route replies.

TABLE 7
The Effect of Prefix Route Requests

	PRREQ/RREQ Initiated	Transmissions
With PRREQs	12 / 16	23,824
Without PRREQs	0 / 112	69,773

Only PRREQs and RREQs with MAX TTL are counted.

Since there are $N/32$ possibilities for each prefix, the probability that a prefix match is found in the first broadcast is high. Table 6 shows the breakdown of the cost to discover the first hop clusterheads. The results in Table 6 confirm that the majority of the prefix requests (30 out of 42) are limited to one hop. Thus, the overhead of DPSR-Graph is dominated by the route discoveries from the clustering nodes to the final destination. In fact, RO_{1hop} is about 4,040, whereas the total overhead for DPSR-Graph is 23,824. The low overhead of RO_{1hop} is an important factor for the reduced routing overhead due to imposed reuse.

Table 7 further highlights the importance of prefix route request by comparing the routing overhead of DPSR-Graph with and without the prefix route request feature. It shows that, in this scenario, DPSR-Graph without the feature incurs an overhead even higher than that of DSR-Graph (50,760) as it has to perform the normal route discoveries to learn about a valid first hop. The reason for the larger number of RREQs when not using prefix route request compared to when using prefix route request is that, in the latter case, many more (prefix) route replies are received which leads to more natural reuse.

8.4 Performance Results

Many-to-One Traffic. In this experiment, we compare DPSR-Graph and DSR-Graph using the many-to-one traffic pattern, i.e., $C = 1$. Since the isolation of natural reuse from imposed reuse is practically infeasible, Fig. 12 depicts the improvement of DPSR-Graph over DSR-Graph when all forms of reuse are in place, as the network size is successively increased from 200 to 800 nodes. Fig. 12a shows that the routing overhead of DPSR-Graph is consistently lower than that of DSR-Graph. Fig. 12b shows that the PDR of DPSR-Graph is comparable to that of DSR-Graph when the maximum speed of nodes is 9 m/s. Further, we find that DPSR-Graph outperforms DSR-Graph even for high mobility with a maximum speed of 19 m/s. This suggests that high mobility does not affect the clustering mechanisms of DPSR significantly since the overhead increases only slightly. Finally, Fig. 12c shows the imposed reuse for DPSR-Graph in the presence of direct routes for the same 800-node network as in Fig. 10. As expected, natural reuse reduces the amount of imposed reuse.

If there were no natural reuse and caching, the ratio between the overhead of DSR-Graph and DPSR-Graph would have been about 32 (for large N) since the overhead of DSR-Graph would be $N \cdot (Cost_{RREQ})$ and the overhead of DPSR-Graph is $\frac{N \cdot (Cost_{RREQ})}{32} + RO_{1hop}$ and the RO_{1hop} term is negligible. However, the natural reuse due to routes obtained from forwarding, intermediate replies, and overhearing in both DPSR-Graph and DSR-Graph reduces the routing overhead difference to a factor of 3.46 for $N = 800$.

In some energy-constrained applications, it may be required to not operate the wireless interface in promiscuous mode. In this scenario, DPSR and DSR cannot benefit from overhearing routes to refresh their routing state. To verify whether DPSR still provides performance improvement with overhearing turned off, we ran simulations for the above scenario of 800 nodes with overhearing turned off. The results show that, without overhearing, the overhead of DSR increased by 124 percent while that of DPSR increased by 89.3 percent. Consequently, without overhearing, DPSR still has much lower overhead than DSR while maintaining a comparable packet delivery ratio.

Many-to-Many Traffic. In this experiment, we increase the value of C from 1 to 8. We evaluate with a maximum speed of 9 m/s since the capacity of the network running DSR or DPSR is not sufficient to support many-to-many traffic at high network sizes and high speeds. Fig. 13 depicts the routing overhead and PDR of DPSR-Graph and DSR-Graph as the network size is increased from 200 to 800 nodes. It can be seen that the routing overhead of DPSR-Graph is consistently lower than that of DSR-Graph. In fact, the overhead of DSR-Graph is almost twice that of DPSR-Graph.

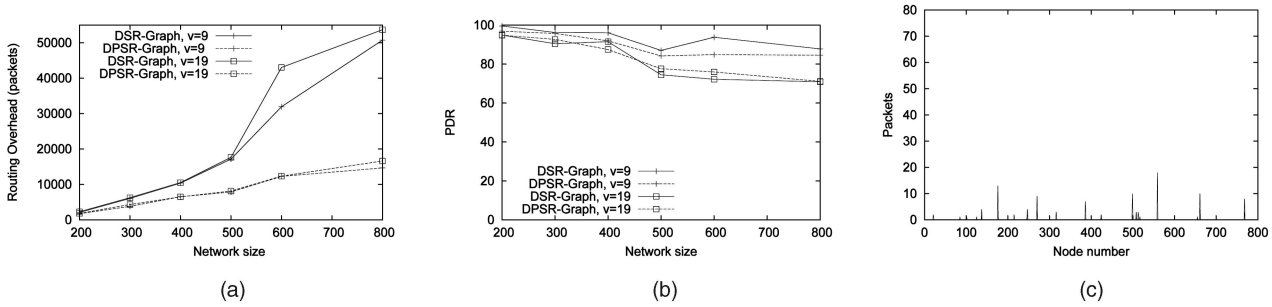

Fig. 12. Routing overhead and PDR comparison of DPSR-Graph and DSR-Graph varying the network size for high mobility (pause time 0s) and $C = 1$. Imposed reuse in DPSR for a network of 800 nodes is also depicted. (a) Running overhead, (b) PDR, and (c) reuse histogram.

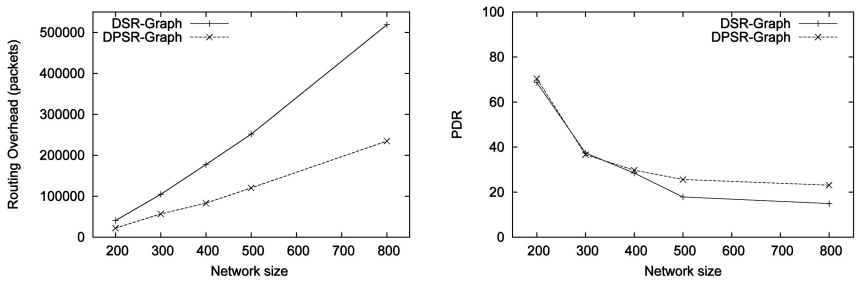

Fig. 13. Routing overhead and PDR comparison of DPSR-Graph and DSR-Graph varying the network size for pause time 0s and $C = 8$.

Fig. 13 also shows that DPSR-Graph's PDR is comparable to that of DSR-Graph for small network sizes, but is up to 50 percent higher than that of DSR-Graph as the network size is increased to 800 nodes. The PDRs of both protocols are low in this scenario. This low PDR is caused by the increased congestion when C is increased to 8. As C is increased, the number of ROUTE REQUEST packets increases. This increase in broadcast packets results in the dropping of data packets due to increased collisions, suggesting that irrespective of the protocol used, the capacity of the network is not enough to support the delivery of data packets. Despite of the low PDRs, the performance benefit of DPSR is identified by the factor of two lower overhead as compared to DSR-Graph.

In this scenario, the overhead of DSR-Graph can be approximated as $N \cdot C \cdot (Cost_{RREQ})$. Since the clustering effect happens for each of the C destinations. The overhead of DPSR-Graph will be $\frac{N \cdot C \cdot (Cost_{RREQ})}{32} + RO_{1hop,Cdests}$. Effectively, DPSR can be viewed as a protocol that imposes reuse by per-destination clustering. As C increases, DPSR-Graph will start reusing the routes discovered to one clusterhead to reach one destination in sending packets to a different destination, e.g., when the two destinations share the same first digit. Thus, the first-hop overhead for C destinations, $RO_{1hop,Cdests}$, will grow slower than $C \cdot RO_{1hop}$. This suggests that, as C increases, the gap between the overhead of DSR-Graph and DPSR-Graph should widen for large networks. However, as the PDR at this network size is low, this effect is not evident.

In summary, the limited capacity of current ad hoc networking technology limits the usability of any ad hoc networking protocol for many-to-many traffic patterns. In the meantime, many-to-one/many-to-some traffic patterns are common in many application scenarios for ad hoc networks (Section 9). In such scenarios, DPSR will provide better performance than DSR.

9 Application Scenarios

The many-to-one traffic scenarios in which DPSR exhibits high imposed reuse can potentially occur in many application scenarios.

Wireless access networks. Consider an ad hoc network being deployed as a wireless extension to the Internet similar to the scenario proposed in [27]. Every node in the network that requires Internet connectivity will initiate multihop connections to the set of gateways. This will effectively result in bursty flows (Web traffic, email, DNS) from many nodes to the gateway nodes. DPSR will be useful in reducing the overhead in such scenarios. Similarly, DPSR will also be useful when many mobile nodes use Transit Access Points [11] as gateways to mesh networks which provide Internet access and community services.

Mobile robotics. Consider an ad hoc network formed by a team of mobile robots. Such kind of networks are useful for military applications, search and rescue, and exploration. Communication to the operator (base) is likely to be frequent in such applications and many-to-one in nature. Unlike sensor networks, the data in these networks is not compressible by aggregation, leading to capacity issues from many-to-one communication [22]. DPSR could be used to reduce both bandwidth consumption and energy consumption in this and other similar scenarios in which the communication is many-to-one and not compressible by aggregation.

Distributed storage systems. Applications that employ parallel download form another type of application that generates the traffic pattern where many nodes communicate with a subset of the nodes. One such example is a p2p-based distributed storage system, CFS [8], in which blocks of a file are distributed over many nodes.

10 Related Work

In addition to DSR, AODV [26], DSDV [25], and TORA [23] are also topology-based MANET routing protocols which assume no knowledge of the mobile nodes' positions. AODV is a hop-by-hop on-demand routing protocol; it uses a similar route discovery mechanism as in DSR, but the routes discovered are set up hop-by-hop along the intermediate nodes. Compared to DSR, AODV emphasizes the freshness of routes more than their reuse. Consequently, it incurs higher routing overhead [5], [9]. AODV leverages implicit reuse when an intermediate node that already has a route set up to a destination replies to a route request. Due to its hop-by-hop nature, the original AODV does not have reuse of partial routes. DSDV is a distance vector protocol that guarantees loop-freedom and performs hop-by-hop routing. It needs periodic routing updates for route maintenance. DSDV does not reuse routes for one destination for another destination due to its inherent proactive nature. TORA is a distributed routing protocol that trades off route optimality with the overhead of route discovery. It also has no specific mechanism for route reuse.

Hierarchical routing protocols using landmarks [2], [3], [16], [24], [31] share the essence of imposing reuse with DPSR. In such systems, the routes between landmarks or their vicinities are reused in delivering data packets. However, hierarchical routing requires an explicit maintenance protocol, whereas "hierarchies" exist implicitly in the DPSR network. Although DPSR imposes reuse in routing, it is still a flat topology-based routing protocol.

DPSR organizes groups of *physically nearby* nodes into clusters so that, for each unique destination, a different member of the group is likely to be chosen as the clusterhead leading to natural load balance. This is different from traditional clustering schemes (for example, [1], [12]) in which the same clusterhead is chosen irrespective of the destination being communicated with. Similarly to traditional clustering schemes, the clusterheads in DPSR in different spatial regions are different.

DPSR is intrinsically effective in reducing the routing overhead in scenarios where many-to-one or many-to-some communication occurs in *mobile* networks. It is fundamentally different from sensor routing protocols which also deal with many-to-one communication patterns for communication from sensor nodes to the sink, but exploit the static deployment of sensor nodes to deal with the many-to-one communication patterns and often perform data aggregation.

11 Conclusions

In this paper, we have presented an on-demand multihop routing protocol for mobile ad hoc networks, DPSR, that exploits the synergy between structured p2p overlay networks and MANETs for imposing route reuse. By imposing reuse of routes to a faraway common destination node among a set of nearby source nodes, DPSR limits the number of the source routes that each node has to discover and rediscover to $O(logN)$, in contrast to the maximum of N source routes each node has to maintain in DSR, and, thus, has the potential to significantly reduce the routing overhead. DPSR limits the number of the source routes that each node has to discover and rediscover to $O(logN)$, while retaining all the attributes of DSR for dealing with the specifics of ad hoc networks.

Our study demonstrates the challenges faced when adapting a p2p routing protocol from wired networks to wireless networks. The design of DPSR illustrates the importance of mobility adaptation and topology awareness in adapting wired Internet-based protocols to efficiently operate in mobile ad hoc networks.

Simulation results show that, for mobile networks with many-to-one or many-to-some traffic patterns, DPSR significantly reduces the routing overhead while providing comparable packet delivery performance to DSR. The benefit from DPSR increases with the network size. In addition, for the any-to-any traffic patterns widely used previously, DPSR is comparable in performance to DSR.

Although DPSR provides significant gains for the many-to-one traffic patterns, the maximum performance benefit from DPSR would occur under the many-to-many traffic patterns. However, our study shows that sustaining such traffic patterns needed to obtain maximum benefit from DPSR is beyond the capacity of current ad hoc networks. Given advances in physical layer technology, it may be possible to support many-to-many traffic patterns in large ad hoc networks. For example, the UWB-based IEEE 802.15.3.a standard [36] is expected to support significantly higher traffic rates ranging from 100 to 500 Mbps. Improvements in modulation schemes are resulting in higher capacity. Techniques like multiuser detection and interference cancellation as well as hardware innovations such as smart antennas, directional antennas, multiple radios, and multichannel radios also have the potential to significantly improve the capacity of ad hoc networks. We believe that the true potential of DPSR will become apparent as the capacity of ad hoc networks increases in the future.

Acknowledgments

The authors would like to thank the anonymous *IEEE Transactions on Parallel and Distributed Systems* (*TPDS*) reviewers whose comments have helped to improve the presentation of this paper. This work was supported in part by the US National Science Foundation (NSF) under Grant No. ANI-0338856. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

References

[1] A.D. Amis, R. Prakash, T.H. Vuong, and D.T. Huynh, "Max-Min D-Cluster Formation in Wireless Ad Hoc Networks," *Proc. IEEE INFOCOM,* Mar. 2000.

[2] S. Banerjee and S. Khuller, "A Clustering Scheme for Hierarchical Control in Multi-Hop Wireless Networks." *Proc. IEEE INFOCOM,* Apr. 2001.

[3] S. Basagni, "Distributed and Mobility-Adaptive Clustering for Multimedia Support in Multi-Hop Wireless Networks," *Proc. IEEE Vehicular Technology Conf.,* Fall 1999.

[4] L. Breslau et al., "Advances in Network Simulation" *Computer,* vol. 33, no. 5, pp. 59-67, May 2000.

[5] 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." *Proc. ACM MobiCom,* Oct. 1998.

[6] M. Castro, P. Druschel, C. Hu, and A. Rowstron, "Exploiting Network Proximity in Distributed Hash Tables," *Proc. Int'l Workshop Future Directions in Distributed Computing (FuDiCo),* June 2002.

[7] M. Castro, P. Druschel, Y.C. Hu, and A. Rowstron, "Exploiting Network Proximity in Peer-to-Peer Overlay Networks," Technical Report MSR-TR-2002-82, 2002.

[8] F. Dabek, M.F. Kaashoek, D. Karger, R. Morris, and I. Stoica, "Wide-Area Cooperative Storage with CFS," *Proc. ACM Symp. Operating Systems Principles (SOSP),* Oct. 2001.

[9] S.R. Das, C.E. Perkins, and E.M. Royer, "Performance Comparison of Two On-Demand Routing Protocols for Ad Hoc Networks," *Proc. IEEE INFOCOM,* Mar. 2000.

[10] *FIPS 180-1 Secure Hash Standard, Technical Report Publication 180-1, Federal Information Processing Standard (FIPS),* NIST, US Dept. of Commerce, Washington D.C., Apr. 1995.

[11] V. Gambiroza, B. Sadeghi, and E.W. Knightly, "End-to-End Performance and Fairness in Multihop Wireless Backhaul Networks," *Proc. ACM MobiCom,* 2004.

[12] M. Gerla and J.T.-C. Tsai, "Multicluster, Mobile, Multimedia Radio Network," *Wireless Networks,* vol. 1, no. 3, pp. 255-265, 1995.

[13] K. Gummadi, R. Gummadi, S. Gribble, S. Ratnasamy, S. Shenker, and I. Stoica, "The Impact of DHT Routing Geometry on Resilience and Proximity." *Proc. ACM SIGCOMM,* Aug. 2003.

[14] P. Gupta and P.R. Kumar, "The Capacity of Wireless Networks," *IEEE Trans. Information Theory,* vol. 46, no. 2, pp. 388-404, Mar. 2000.

[15] S. Gwalani, E.M. Belding-Royer, and C.E. Perkins, "AODV-PA: AODV with Path Accumulation," *Proc. Next Generation Internet Symp.,* May 2003.
[16] Z.J. Haas and M.R. Pearlman, "The Performance of Query Control Schemes for the Zone Routing Protocol," *Proc. ACM SIGCOMM,* Aug. 1998.
[17] Y.C. Hu, S.M. Das, and H. Pucha, "Exploiting the Synergy between Peer-to-Peer and Mobile Ad Hoc Networks," *Proc. Workshop Hot Topics in Operating Systems (HotOS-IX),* May 2003.
[18] Y.-C. Hu and D.B. Johnson, "Caching Strategies in On-Demand Routing Protocols for Wireless Ad Hoc Networks," *Proc. ACM MobiCom,* Aug. 2000.
[19] D.B. Johnson and D.A. Maltz, *Dynamic Source Routing in Ad Hoc Wireless Networks.* Kluwer Academic, 1996.
[20] S.-J. Lee and M. Gerla, "AODV-BR: Backup Routing in Ad Hoc Networks," *Proc. IEEE Wireless Comm. and Networking Conf. (WCNC),* Sept. 2000.
[21] J. Li, C. Blake, D.S.D. Couto, H.I. Lee, and R. Morris, "Capacity of Ad Hoc Wireless Networks." *Proc. ACM MobiCom,* Mar. 2001.
[22] D. Marco, E. Duarte-Melo, M. Liu, and D.L. Neuhoff, "On the Many-to-One Transport Capacity of a Dense Wireless Sensor Network and the Compressibility of its Data," *Proc. Int'l Workshop Information Processing in Sensor Networks (IPSN),* Apr. 2003.
[23] V.D. Park and M.S. Corson, "A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks," *Proc. IEEE INFOCOM,* Apr. 1997.
[24] G. Pei, M. Gerla, and X. Hong, "LANMAR: Landmark Routing for Large Scale Wireless Ad Hoc Networks with Group Mobility," *Proc. IEEE/ACM MobiHoc,* Aug. 2000.
[25] C.E. Perkins and P. Bhagwat, "Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers," *Proc. ACM SIGCOMM,* Aug. 1994.
[26] C.E. Perkins and E.M. Royer, "Ad Hoc On-Demand Distance Vector Routing," *Proc. IEEE Workshop Mobile Computing Systems and Applications (WMCSA),* Feb. 1999.
[27] J. Raju and J. Garcia-Luna-Aceves, "Scenario-Based Comparison of Source-Tracing and Dynamic Source Routing Protocols for Ad Hoc Networks," *ACM Computer Comm. Rev.,* Oct. 2001.
[28] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker, "A Scalable Content-Addressable Network," *Proc. ACM SIGCOMM,* Aug. 2001.
[29] A. Rowstron and P. Druschel, "Pastry: Scalable, Distributed Object Location and Routing for Large-Scale Peer-to-Peer Systems," *Proc. Middleware Conf.,* Nov. 2001.
[30] I. Stoica, R. Morris, D. Karger, M.F. Kaashoek, and H. Balakrishnan, "Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications." *Proc. ACM SIGCOMM,* Aug. 2001.
[31] P.F. Tsuchiya, "The Landmark Hierarchy: A New Hierarchy for Routing in Very Large Networks," *Proc. ACM SIGCOMM,* Aug. 1988.
[32] J. Yoon, M. Liu, and B. Noble, "Random Waypoint Considered Harmful," *Proc. IEEE INFOCOM,* Apr. 2003.
[33] J. Yoon, M. Liu, and B. Noble, "Sound Mobility Models." *Proc. ACM MobiCom,* Sept. 2003.
[34] X. Zeng, R. Bagrodia, and M. Gerla, "Glomosim: A Library for Parallel Simulation of Large-Scale Wireless Networks," *Proc. Parallel and Distributed Simulations Workshop (PADS),* May 1998.
[35] B.Y. Zhao, J.D. Kubiatowicz, and A.D. Joseph, "Tapestry: An Infrastructure for Fault-Resilient Wide-Area Location and Routing," Technical Report UCB//CSD-01-1141, Univ. of California Berkeley, Apr. 2001.
[36] IEEE 802.15 WPAN High Rate Alternative PHY Task Group 3a, http://www.ieee802.org/15/pub/TG3a.html, 2006.

Himabindu Pucha received the MSEE degree from Purdue University and the BEng degree from the University of Bombay, India. She is currently a PhD candidate in the School of Electrical and Computer Engineering at Purdue University. Her research interests include Internet routing and overlay networks, peer-to-peer systems and applications, and mobile computing. She is a student member of the IEEE.

Saumitra M. Das received the MS degree from Carnegie Mellon University and the BEng degree from the University of Bombay, India. He is currently a PhD candidate in the School of Electrical and Computer Engineering at Purdue University. His research interests include cross-layer system design for multihop wireless networks, scalable routing strategies in wireless ad hoc networks, and mobile robotics. He is a student member of the IEEE.

Y. Charlie Hu received the MS and MPhil degrees from Yale University in 1992 and the PhD degree in computer science from Harvard University in 1997. He is an assistant professor of electrical and computer engineering and computer science at Purdue University. From 1997 to 2001, he was a research scientist at Rice University. Dr. Hu's research interests include operating systems, distributed systems, networking, and parallel computing. He has published more than 80 papers in these areas. Dr. Hu received the US National Science Foundation (NSF) CAREER Award in 2003. He served as a TPC vice chair for the 2004 International Conference on Parallel Processing, and a cofounder and TPC cochair for the International Workshop on Mobile Peer-to-Peer Computing. Dr. Hu is a member of USENIX, the ACM, and the IEEE.