# Implicit Source Routes for On-Demand Ad Hoc Network Routing

Yih-Chun Hu
yihchun@cs.cmu.edu

David B. Johnson
dbj@cs.rice.edu

Department of Computer Science
Rice University
Houston, TX 77005-1892 USA

http://www.monarch.cs.rice.edu/

## Abstract

In an ad hoc network, the use of *source routing* has many advantages, including simplicity, correctness, and flexibility. For example, all routing decisions for a packet are made by the sender of the packet, avoiding the need for up-to-date routing information at intermediate nodes and allowing the routes used to be trivially guaranteed loop-free. It is also possible for the sender to use different routes for different packets, without requiring coordination or explicit support by the intermediate nodes. In addition, *on-demand* source routing has performed very strongly when compared against other proposed protocol designs. However, source routing has the disadvantage of increased per-packet overhead due to the source route header that must be present in every packet originated or forwarded.

In this paper, we propose and analyze the use in ad hoc networks of *implicit* source routing, and show that it preserves the advantages of source routing while avoiding the associated per-packet overhead in most cases. We evaluated this technique through detailed simulations of ad hoc networks based on the Dynamic Source Routing protocol (DSR), an on-demand ad hoc network routing protocol based on source routing. Although routing packet overhead increased slightly with implicit source routing, by about 12.3%, the *total* number of bytes of overhead decreased substantially, by between 44 and 86%. On all other metrics evaluated, the performance of DSR either did not change significantly or actually improved somewhat, due to indirect effects of the reduced routing overhead.

## 1. Introduction

In an *ad hoc network*, the hosts (nodes) must cooperate to dynamically establish routing among themselves; a packet sent by one node may be forwarded in turn by a sequence of other nodes, allowing the packet to reach a destination beyond the sender's wireless transmission range. The routing protocol used in the network must be able to find such multi-hop paths, and must be able to repair or find new paths as nodes move or as wireless transmission conditions around the nodes change and existing paths fail. Ad hoc networks do not rely on any traditional network infrastructure such as base stations or access points, and do not typically have any form of centralized administration or control.

A number of routing protocols using a variety of routing techniques have been proposed for use in ad hoc networks. These protocols can be divided into *on-demand* (or reactive) protocols (e.g., DSR [9], AODV [18], and TORA [2]), in which nodes search for or maintain a route only when one is needed, and *periodic* (or proactive) protocols (e.g., DSDV [17]), in which nodes periodically exchange routing information and attempt to always know a current route to each destination. In general, protocols that use on-demand mechanisms have been shown to outperform those based on periodic mechanisms due to their reduced overhead and ability to react quickly as routes change [1, 8, 15, 3], and thus we restrict our focus here to on-demand protocols.

In an ad hoc network, the use of *source routing* can provide many advantages, including simplicity, correctness, and flexibility [9, 10, 11, 12]. For example, since all routing decisions for a packet are made by the sender of the packet, intermediate nodes that forward it need not maintain up-to-date, consistent routing tables for the destination. Forwarding at each hop consists simply of locally transmitting the packet to the next address indicated in the source route in the packet's header; the sequence of hops over which any packet is forwarded can easily be guaranteed to be loop-free by not allowing duplicates in the list of hops. By including the source route in the packet's header, additional routing information is also partially spread around the network without requiring additional packets to be transmitted. In addition, for reasons such as load balancing or differentiated treatment of different types or classes of packets for Quality of Service (QoS), it is possible for the sender to use different routes for different

This work was supported in part by the NASA Cross Enterprise Technology Development Program under Grant Number NAG3-2534. The views and conclusions contained here are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either express or implied, of NASA, Rice University, Carnegie Mellon University, or the U.S. Government.


MobiHOC 2001, Long Beach, CA, USA

packets, without requiring coordination or explicit support by the intermediate nodes.

However, source routing has the disadvantage of increased per-packet overhead. With source routing, the size of each packet is increased in order to carry the source route of hops through which the packet is to be forwarded. Since the source route is always present in the packet, the extra network overhead caused by the presence of the source route is incurred not only when the packet is originated, but also each time it is forwarded to the next hop. This extra network overhead decreases the bandwidth available for transmission of data, increases the transmission latency of each packet, and consumes extra battery power in the network transmitter and receiver hardware.

In this paper, we propose and analyze the use in ad hoc networks of *implicit* source routing. This technique preserves the advantages of source routing while avoiding the associated per-packet overhead. In a manner in part similar to techniques used for MPLS [19] or ATM virtual circuits [21], each packet is tagged with a *flow identifier* when the packet is sent by its original sender. The flow identifier indicates the route to be followed by this and all packets belonging to a logical flow from this sender to the destination of the packets. Intermediate nodes along the route retain *soft state* indicating the next hop to which packets belonging to that flow should be forwarded, avoiding the need to carry the full source route in each packet. This soft state should be retained until the specified timeout but may be discarded earlier, for example due to node failures, without impacting correctness.

We base our design and analysis of implicit source routing on extending the Dynamic Source Routing protocol (DSR) [9, 10, 12, 11, 6], since it is based on source routing and has been shown by a number of groups to perform well when compared to other protocols [1, 8]. DSR allows nodes to dynamically discover, on demand, source routes to nodes to which they send packets, and allows these source routes to be maintained when links between nodes break due to node mobility, wireless propagation changes, or other factors. Our implicit source routing mechanism fits naturally into the existing structure of the DSR protocol [6] and preserves the important fundamental properties of DSR's operation. To evaluate our implicit source routing design, we conducted a set of detailed simulations of DSR, both with and without use of implicit source routing, and we analyze the differences in the behavior of these two protocols in terms of packet delivery ratio, latency, path length optimality, and packet and byte routing overheads.

In Section 2 of this paper, we provide a brief overview of the operation of the existing DSR protocol, and we highlight the important properties of DSR that we use as a basis for our design. Section 3 then details the design and operation of implicit source routing mechanism, including a proof of the correctness properties of the design. Our evaluation methodology is described in Section 4, and in Section 5, we present the results of our simulations evaluating implicit source routing. Section 6 discusses related work, and finally, in Section 7, we present conclusions.

## 2. DSR Overview

The Dynamic Source Routing protocol (DSR) [9, 10, 12, 11] is composed of two mechanisms: *Route Discovery* and *Route Maintenance*. Route Discovery is the mechanism by which a node originating a packet to some destination discovers a source route to that destination if it does not currently have a route to that destination cached. Route Maintenance is the mechanism by which a node sending a packet to some destination learns if the route it used for that packet has broken, for example because some node in the route has moved out of wireless transmission range of the previous node in the route. This section gives a brief overview of the operation of Route Discovery and Route Maintenance in DSR, in order to establish a basis for the implicit source routing description that follows in Section 3.

### 2.1. Route Discovery

In Route Discovery, a node **S** wishing to send a packet to some node **D** locally broadcasts a ROUTE REQUEST packet, which is received by nodes within wireless transmission range of **S**. The ROUTE REQUEST identifies the destination node to which a route is needed, and also includes a unique identifier for this Route Discovery, chosen by node **S**. Node **S** is referred to as the *originator* of the Route Discovery, and node **D** is referred to as the *target* of the Discovery.

If a node receives a ROUTE REQUEST for which it is not the target, it locally rebroadcasts the ROUTE REQUEST after adding its own address to a list of nodes in the ROUTE REQUEST that have forwarded this copy of the REQUEST. When the REQUEST reaches the target, this list of hops in the REQUEST will thus indicate the sequence of hops along which this copy of the REQUEST was forwarded in order to reach from the originator to the target of the Route Discovery. The target node then returns a copy of this sequence of hops to the originator in a unicast ROUTE REPLY packet, and the originator remembers this route in its Route Cache for possible use on subsequent packets. The ROUTE REPLY may in general be routed along any path, independent of the route taken by the ROUTE REQUEST packet, thus allowing uni-directional links to be supported (if allowed by the specific MAC protocol in use on that link).

A number of optimizations that improve the performance of this basic Route Discovery mechanism have been defined [12, 5, 10, 11]. For example, ROUTE REQUESTs may be limited by the TTL field of the IP header of the REQUEST packet, allowing non-propagating ROUTE REQUESTs or "expanding ring" searches. If a ROUTE REQUEST reaches a node that has in its cache a route to the target of the Discovery, this node may reply from its cache, setting the route returned in the ROUTE REPLY to the concatenation of the route from the ROUTE REQUEST plus the route from its own Route Cache; by replying from its cache, the new route is returned to the

originator sooner, and the overhead of Route Discovery is reduced since the ROUTE REQUEST need not be rebroadcast. A node may also update its Route Cache based on source routes or other routing information that it forwards or that it may overhear from other nodes by optionally operating its network interface hardware in "promiscuous" receive mode.

### 2.2. Route Maintenance

When a node originates a packet, it lists in a source route in the header of the packet the complete list of hops through which the packet is to be forwarded. The original sender is then responsible for confirming that the packet has been received by the first intermediate hop in the route, retransmitting the packet if necessary until this confirmation is received or until a maximum number of retransmission attempts have been performed. Likewise, in turn, each intermediate node on the route is responsible in the same way for confirming that the packet has been received by the next node in the route; the packet is retransmitted by the intermediate node if necessary, as with the original sender.

The confirmation of receipt by the next hop may be obtained through either a "passive" acknowledgement [13], through the link-level acknowledgement present in many wireless MAC protocols such as IEEE 802.11 [7], or through an explicit DSR acknowledgement packet from the next hop if necessary. If no confirmation is received after a limited number of retransmission attempts for the packet, the link from this node to the next hop is considered to have broken, and a ROUTE ERROR identifying this broken hop is returned to the original sender. The original sender then removes this one broken link from its Route Cache; for subsequent packets to this same destination, the sender may use an alternate route that it may already have in its Route Cache or may reinvoke Route Discovery to discover a new source route to the destination.

As with Route Discovery, a number of optimizations that improve the performance of the protocol have been defined [12, 5, 10, 11]. For example, after a node detects a broken link and returns a ROUTE ERROR to the original sender of a packet, the node may attempt to *salvage* the packet if it has in its own Route Cache a different route to the packet's destination; to do so, the node replaces the original route with the route from its cache and transmits the packet to the new next hop node. As another optimization, the protocol supports *automatic route shortening* to allow source routes in use to be shortened when possible, for example when nodes move close enough together so that one or more intermediate hops are no longer necessary. If a node is able to promiscuously receive a packet not intended for it as the next hop, but for which this node is listed in the unused portion of the packet's source route, then this node returns a "gratuitous" ROUTE REPLY to the original sender of the packet; this REPLY gives the shorter route that does not include the intermediate nodes between the node that transmitted the packet and this node.

### 2.3. Properties of DSR

In wireless networks, communication over a wireless hop between two nodes may at times not work equally well in both directions, resulting in a number of unidirectional links in the network. DSR is capable of routing correctly over networks that contain such unidirectional wireless links, since the path over which the ROUTE REPLY is sent need not be the same as the reverse of the path over which the ROUTE REQUEST was forwarded. Acknowledgements for Route Maintenance may also follow a different (multi-hop) path to the previous node in the source route.

Nodes using DSR may cache multiple routes to any destination, and may use any of these routes at any time for any packet being sent. For example, a node may remember multiple routes, in order to have another route available should a link in the route in use break; a node may also actively use multiple routes at the same time to some destination for load balancing or for Quality of Service control.

All routing decisions for any packet are determined entirely by the packet's sender, allowing the sender to utilize any local algorithm to select the route based on local concerns such as load balancing (as mentioned above), the perceived longevity or reliability of the nodes or links along the route, or the security of the nodes along or within wireless range of the route chosen.

All routing in DSR is done entirely using *soft state*, significantly increasing the potential reliability and robustness of the protocol over other routing protocols that rely on hard state at other nodes in the network. The necessary soft state in DSR is created as needed, and the loss of any soft state does not in any way affect the correctness of the protocol's operation.

The addition of *implicit* source routing to DSR described in this paper preserves the basic operation of DSR's Route Discovery and Route Maintenance mechanisms, including all of these important resulting properties of DSR. Although DSR with implicit source routing may be seen as similar to other on-demand routing protocols that do not use source routing, in practice all routes used in either version of DSR are still discovered and established as source routes, with the complete sequence of hops determined from the source to the destination. For example, AODV [18] borrows features of DSR's on-demand Route Discovery mechanism, but it uses only hop-by-hop routes and is not based on source routing; none of the properties of DSR described in this section hold for AODV.

## 3. Implicit Source Routing Operation

### 3.1. Basic Operation

Conceptually, with implicit source routing, a tuple ⟨source address, destination address, flow identifier⟩ takes the place of the full source route in each packet. The source address and destination address can be placed in the IP header, and the flow identifier is placed in a special header. Each node participating in implicit source routing has a *Flow Table*, with

one entry for each flow forwarded by that node. A Flow Table entry minimally must record the next hop address to which a packet for this flow should be forwarded, in addition to the source address, destination address, and flow identifier for this flow.

A source can establish a new flow by sending a *flow establishment packet*. A flow establishment packet is a packet with two headers (i.e., extension headers or options): one containing the flow identifier, and the other containing a source route and a timeout for the flow. When an intermediate node forwards such a packet, in addition to forwarding it according to the source route information, it creates a Flow Table entry for this flow and inserts the necessary information from the packet. A flow establishment packet is normally sent by including these two headers in an existing packet to be sent along that source route; it is also possible (but not necessary) to send the flow establishment packet as a special control packet along the source route.

A node is required to remove a flow's entry from the Flow Table when that node has not forwarded packets for that flow for a period of time specified by the timeout for the flow. As a result, a Flow Table entry is also required to keep the timeout, as well as the time at which this flow is to expire.

A source that has already sent one or more flow establishment packets for a given flow may decide that each node along that flow has established a Flow Table entry for that flow. This source may then send any subsequent packets routed solely by implicit source routing by adding a flow identifier header in lieu of a source route in each packet. A node forwarding a packet sent using implicit source routing checks its Flow Table for an entry corresponding to the flow identifier in the packet. If the node finds one, it forwards the packet by setting the MAC-layer destination address to the MAC address of the next hop indicated in the matching Flow Table entry. Otherwise, it sends a FLOW UNKNOWN error back to the source of the packet.

A source node receiving a FLOW UNKNOWN error addressed to itself marks its Flow Table entry for this flow to indicate that the flow must be reestablished. For the purposes of our simulation, we send three establishment packets when the flow is first created, and three establishment packets each time a FLOW UNKNOWN error is received; each flow establishment packet is sent only when there is data to be transmitted along the flow. By repeating the flow establishment packet for the first three data packets sent when the flow is established or reestablished, the protocol is able to tolerate loss of some flow establishment packets without triggering the overhead of a FLOW UNKNOWN error and the resulting latency for re-establishment.

### 3.2. Default Flows

When a packet is sent using implicit source routing forwarding, it still requires some small amount of overhead in the packet. These additional header bytes in the packet can be entirely eliminated by the use of *default flows*. Conceptually, a node is most likely to use a flow more recently established. Therefore, our protocol allows the use of the most recently established flow with no per-packet overhead for most packets.

To enable this, each node keeps a *Default Flow Table*. For each ⟨source address, destination address⟩ pair, a node keeps the greatest flow identifier for which it has sent or forwarded packets, as well as the expected TTL value of packets sent along the default flow (alternatively, expected TTL value can be stored in the Flow Table).

In order to allow the same flow to be used for both default flow forwarding as well as basic flow forwarding, the expected TTL in the Default Flow Table must be set only upon hearing a flow establishment packet. Additionally, we constrain a source wishing to use a given flow as a default flow to set the TTL of all flow establishment packets for that flow to the same value, and we disallow the use of default flow routing along paths that do not reduce the TTL value in forwarded packets by exactly one at each hop.

When a source node originates a packet along a route that is the default flow for that ⟨source, destination⟩ pair, and that packet has the same TTL as the flow establishment packets for that flow, it transmits the packet to the next hop specified in the Flow Table.

When a node receives a packet with neither a source route nor a flow identifier header, and it is not the node named in the IP Destination Address field, it checks its Default Flow Table; if it finds a flow for the ⟨source, destination⟩ pair specified in the IP header, and if the expected TTL matches the actual IP header TTL, the packet is processed as if it had a flow header specifying the flow identifier found in the Default Flow Table as the packet's flow identifier. Otherwise, a DEFAULT FLOW UNKNOWN error is returned to the source of the packet. A source node receiving a DEFAULT FLOW UNKNOWN error addressed to itself marks its Flow Table entry for the default flow to indicate that the flow must be reestablished.

A Default Flow Table entry at a node times out when all flows corresponding to the ⟨source, destination⟩ pair time out at that node, although this is necessary for correctness only when nodes may crash and lose their state, or when flow identifiers may wrap around.

### 3.3. Automatic Route Shortening

As described in Section 2.2, in the base version of DSR [12], a mechanism exists by which routes actively being used can be shortened in certain ways. Specifically, when the transmission of a packet is promiscuously overheard by a node in the source route, that node determines if the packet is downstream of it (that is, has already been forwarded by it) or upstream of it (that is, has yet to be forwarded by it). If the packet is upstream of it, the route can be shortened by removing the intervening hops not yet traversed leading to this node's receipt of the packet. In this case, the node generates a DSR "gratuitous" ROUTE REPLY to the source of the packet, indicating the shortened route. For example, suppose node **S** is using the route $\mathbf{S} \rightarrow \mathbf{A} \rightarrow \mathbf{B} \rightarrow \mathbf{C} \rightarrow \mathbf{D}$ to send packets to destination node **D**; if node **C** overhears a packet from **S** being

forwarded by node **A**, node **C** can return a gratuitous ROUTE REPLY to **S** providing the shorter route $\mathbf{S} \rightarrow \mathbf{A} \rightarrow \mathbf{C} \rightarrow \mathbf{D}$ for use with subsequent packets to **D**.

This mechanism, called *automatic route shortening*, takes advantage of the source route in each packet to determine whether or not a packet is upstream of a node that promiscuously overheard the packet. In order to enable automatic route shortening when no source route is present, a Hop Count field is added to the flow identifier header. A source node originating a packet initializes the Hop Count field to 0, and each node forwarding this packet using implicit source routing increments the Hop Count by 1. At each node, the expected Hop Count value, as well as the complete source route, is stored in the node's Flow Table during flow establishment. Packets overheard upstream are determined to be those that have a Hop Count *less than* the Hop Count in the corresponding Flow Table entry.

Similarly, automatic route shortening is possible for packets sent along default flows by examining the TTL; if the TTL value is *greater than* expected, it is considered to be upstream. Using the IP TTL for automatic route shortening for packets sent along a default flow avoids the need to carry *any* header information in the packet not normally already present in an IP header; we use an explicit Hop Count rather than the IP TTL in non-default flow routed packets to avoid placing any restriction on how the IP TTL field is used in forwarding the packet and to handle the case in which the IP TTL is used in some non-standard way by some hop along the flow (e.g., for nodes that decrement the TTL by more than 1 when forwarding the packet).

When a node promiscuously overhears a packet, it searches the packet for a flow identifier by looking in the flow header, if present, or by searching its Default Flow Table. If the packet is determined to be upstream of this node, the node stores a record in its limited-size *Automatic Route Shortening Table*, with the source and destination addresses, the flow identifier, the packet, and the number of hops by which the route could have been shortened. In order to reduce the possibility of polluting the source's Route Cache, a gratuitous ROUTE REPLY is sent only when the packet is forwarded by the node that previously overheard the packet.

### 3.4. Additional Interactions with DSR

DSR takes advantage of aggressively caching overheard routes in order to maintain high packet delivery ratio and low overhead. However, with implicit source routing, source routes do not appear in the majority of packets, lessening the opportunity for these routes to be overheard and cached by other nodes. As a heuristic to give DSR with implicit source routing a better ability to utilize these optimizations, we chose to send any data packet sent along a flow as a flow establishment packet (containing both a flow identifier header and a source route header) if an establishment packet has not been sent along that flow within the last 5 seconds.

Implicit source routing also has an effect on DSR's salvaging optimization, as described in Section 2.2. Since packets sent with both a flow identifier header and a source route header are considered to be establishment packets, a salvaging node must remove any existing flow identifier header. Furthermore, since flow identifiers may only be assigned at the source, intermediate nodes may not salvage by using a flow header and must instead use a full source route header for salvaging.

### 3.5. Correctness

In this section, we give a proof of the correctness properties of the implicit source routing mechanism.

**Claim 1.** No packet will be received by a single node twice while being forwarded using a flow identifier header.

**Proof 1.** Proof is by contradiction. Consider the first node that received a packet twice; call it **A**. Node **A** must have been sent the packet the first time by some node **B** and some second time by some node **C**. Now **B** and **C** are distinct nodes, since otherwise that node would have received the packet twice, prior to **A** receiving the packet twice. Since the next hop for an explicitly specified flow is the next hop in the source route of the flow establishment packet, then in the source route of the flow establishment packet, the address of **A** occurred immediately after the address of node **B**, and occurred immediately after the address of node **C**. Since we require source routes to be loop-free, this is impossible, which completes the proof.

**Claim 2.** After a packet $p$ has been received by the same node **A** twice since the most recent default route change for that packet's $\langle$source, destination$\rangle$ pair at **A**, packet $p$ will no longer be forwarded using default flow forwarding.

**Proof 2.** Proof is by contradiction. Assume that there is a packet $p$ that has been received by **A** twice since the time of the most recent default route change at **A**. We define $t_s$ to be the time of the most recent default route change for this $\langle$source, destination$\rangle$ pair at **A**.

Both times packet $p$ was received, **A**'s default flow had the same expected TTL, since no default flow changes occurred since $t_s$. Let this expected TTL be $\text{TTL}_e$.

The first time $p$ was received after $t_s$, the TTL must have been equal to $\text{TTL}_e$, since otherwise **A** would not have forwarded the packet using the default flow mechanism. Also, since TTL is strictly monotone decreasing, the second time **A** receives $p$, its TTL must be some value less than $\text{TTL}_e$. Node **A** would see that $p$ does not have the correct TTL for this default flow and would stop forwarding the packet using default flow forwarding.

**Claim 3.** A routing loop in the default flow forwarding mechanism cannot persist indefinitely after the last default route change at any point in the network.

**Proof 3.** Whenever a transient routing loop is stopped in the way described in Proof 2, a DEFAULT FLOW UNKNOWN error is sent to the source of the packet. After one of those errors reaches the source, the source will attempt to reestablish the default flow. When the default flow has been reestab-

lished across the entire source route, packets sent along it will not loop until there are further routing changes.

Protocols that rely entirely on hop-by-hop, per-flow state to forward packets are unable to detect routing loops if routing state continues to change in certain ways *while* the packet is in transit. In the implicit source routing extensions described above, a source particularly concerned about looping in such a fashion can send all packets with a flow identifier header for 4 bytes of overhead per packet for the flow identifier header.

## 4. Evaluation Methodology

To evaluate our implicit source routing mechanism, we utilized the *ns-2* network simulator [4], together with our Monarch Project wireless and mobile *ns-2* extensions [1, 16], to compare the behavior and performance of DSR with implicit source routing against the original operation of DSR without it. *ns-2* is a discrete event simulator developed by the VINT Project and widely used in network protocol research. The Monarch Project wireless and mobile simulation extensions to *ns-2* provide arbitrary physical node position and mobility, with a realistic radio propagation model including effects such as attenuation, propagation delay, carrier sense, collision, and capture effect. The network interface is modeled after the Lucent/Agere WaveLAN/ORiNOCO IEEE 802.11 product, which has a nominal transmission range of 250 m depending on capture effect and colliding packets, and provides a transmission rate of 2 Mbps; the network interface uses the IEEE 802.11 Distributed Coordination Function (DCF) [7] MAC protocol, which employs physical and virtual carrier sensing for collision avoidance.

For the base version of DSR without implicit source routing, we used the latest simulation code for DSR available from the Monarch Project [5]. The Route Cache we used is the "Link-MaxLife" cache, which is a link state cache in which link timeouts are chosen dynamically based on observed usage and errors. The cache also attempts to choose the longest-lived shortest path when searching the cache for a path to the destination. We have found the base version of DSR with this Route Cache to perform very well [5]; for example, in ad hoc networks of 50 mobile nodes moving continuously at maximum speeds of 20 m/s (average 10 m/s), over 98% of originated data packets are delivered.

The simulated implicit source routing protocol is based on the same version of the DSR simulation code, using the same Link-MaxLife cache. However, in this case, the route selection made by the cache does not override the default route unless the selection is shorter than the current default route. All packets are sent using implicit source routing, and when a flow has not had an establishment packet sent for 5 or more seconds, the following packet originated along that flow includes the extra header to make it an establishment packet.

The simulation results presented in this paper are based on 40 randomly generated scenarios, each involving 50 mobile nodes moving about in an area 1500 m $\times$ 300 m for 900 seconds. Nodes in our simulations move according to the Random Waypoint model [1, 10], in which each node begins at a randomly chosen position, picks a new random position to which to move, and moves there in a straight line at a randomly chosen speed. Each node independently repeats this behavior for the duration of the simulation run. The average degree of mobility is varied by causing each node to remain stationary for a period called the *pause time* each time before it begins moving to its next chosen position. In our simulations, the movement speed of each node is uniformly chosen with a maximum speed of 20 m/s, and pause time is varied between 0 s (a continuously moving network) and 900 s (a stationary network). Specifically, the following seven pause time values were used in our simulations: 0, 30, 60, 120, 300, 600, and 900 s. Each of the 40 scenarios used in our simulations was generated in advance, allowing identical scenarios to be used in the simulations of each version of the protocol.

The communication model for each flow between nodes used in our simulations is constant bit rate (CBR) traffic. For each flow, a random source and destination node was chosen, and each flow sends 4 packets per second; each simulation shows the results for 20 such flows. Each packet carries 512 bytes of data payload, making the basic packet size including an IP header 532 bytes.

We evaluated the implicit source routing mechanism using the simulation methodology described in Section 4, and we report our results using the following metrics:

- *Packet delivery ratio*: The fraction of originated data packets that are successfully delivered to their intended destination nodes. This specifically counts packets that are sent by the "application layer" on a source node that are received by the "application layer" on the corresponding destination node.
- *Packet delivery latency*: The average latency required between originating a data packet until the packet is delivered to the intended destination node. The packet delivery latency can only be measured for packets that are successfully delivered to their destination.
- *Path length optimality*: The difference between the number of hops over which a packet was routed and the number of hops in the shortest route that physically existed when the packet was sent. The simulator is able to determine this theoretical shortest route at all times, based on the nominal wireless transmission range of each node of 250 m.
- *Routing packet overhead*: The total number of separate overhead packets used by the routing protocol. Each transmission (whether original or forwarding) of an overhead packet is counted.
- *Total bytes of overhead*: The total number of bytes of routing overhead, including the size of all routing overhead packets and the size of any routing headers added

to data packets. The bytes are counted on transmission (whether original or forwarding) for each packet.

## 5. Results

Packet delivery ratio is an important measure of the overall operation of any routing protocol. We measured the packet delivery ratio averaged over the 40 randomly generated scenarios for each pause time. These results are shown in Figure 1, with the error bars in the graph representing the 99% confidence interval of the average shown.

The use of implicit source routing marginally *improves* the packet delivery ratio. Two competing factors cause the difference in packet delivery ratio between the base version of DSR and DSR with implicit source routing. Since implicit source routing sends most packets without a source route, less-complete routing information is propagated through the network, reducing the success ratio of salvaging; across all 280 runs of each protocol, implicit source routing drops a packet due to the inability to find a route 70% more often (although both protocols drop very few packets). The other factor is congestion: since the packets transmitted by implicit source routing are smaller on average, each node is able to drain its network interface transmit queue more quickly, resulting in fewer drops from full interface queues; across a random sample of 140 runs of each protocol, implicit source routing drops 17% fewer packets as a result of full interface queues. Since drops resulting from inability to find a route are much less common, DSR with implicit source routing generally has slightly higher packet delivery ratio.

Another measure of the overall operation of the routing protocol is the packet delivery latency. Our simulation results for packet delivery latency are shown in Figure 2. In general, DSR with implicit source routing has slightly *better* latency than does base DSR, due to the smaller average size of each packet without the source routing header present. The reduced packet size directly decreases the transmission time for the packet and also indirectly improves latency due to reduced contention for transmission bandwidth from other packets.

Finally, a third measure of the routing protocol's overall operation is the path length optimality. As shown in Figure 3, however, the introduction of implicit source routing into DSR did not significantly alter DSR's path length optimality.

Beyond these three overall measures, the routing packet overhead and total bytes of overhead provide an internal measure of the operation of the protocol. These metrics indirectly affect the three overall metrics discussed above and also contribute to other measures of the protocol such as CPU efficiency and battery power consumption. Figure 4 shows the routing packet overhead for base DSR and for DSR with implicit source routing, and Figure 5 shows this comparison for total bytes of overhead.

For routing packet overhead, we did not count flow establishment packets as overhead since they also contain data. In comparing DSR with and without implicit source routing, we found that implicit source routing incurs 12.3% more routing packet overhead. These overhead packets came from three sources: FLOW UNKNOWN errors, DEFAULT FLOW UNKNOWN errors, and additional Route Discoveries. The number of FLOW UNKNOWN errors and DEFAULT FLOW UNKNOWN errors, however, is quite small, since we do not model a limit on the Flow Table size, nor do we model nodes crashing and restarting. Most of the additional overhead packets thus are the result of additional Route Discoveries, which are required because fewer packets are sent with full source routes, lessening the ability of other nodes to cache overheard routes, as described in Section 3.4.

For counting total bytes of overhead, we modeled packet sizes in our simulations according to the packet formats defined in our IETF Internet-Draft specifications for DSR [12,

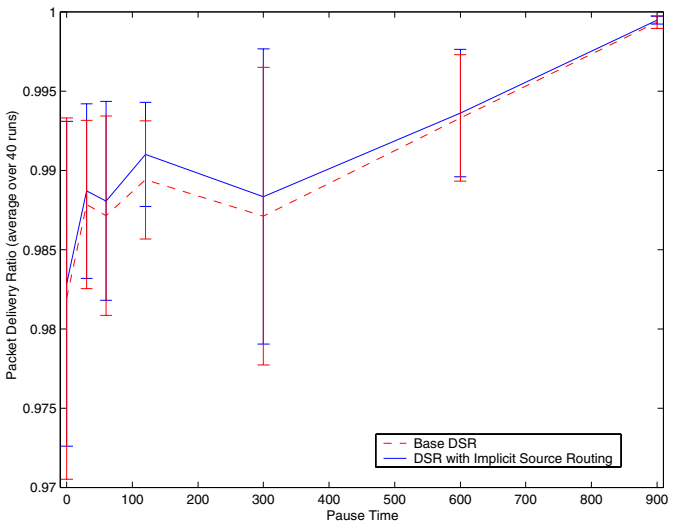


**Figure 1** Effect of Implicit Source Routing on Packet Delivery Ratio

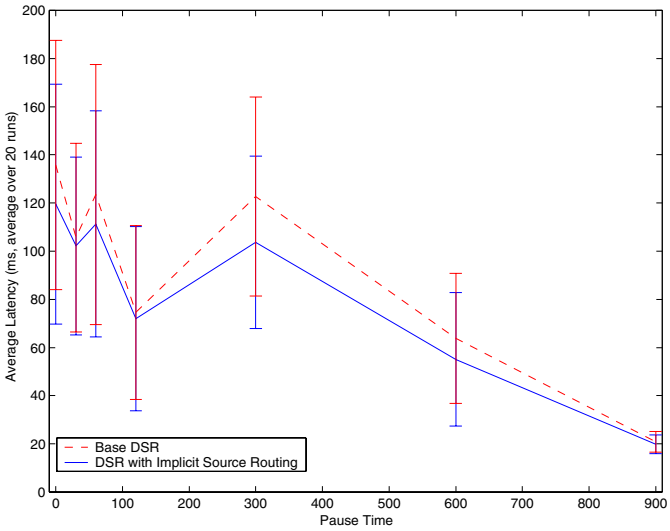


**Figure 2** Effect of Implicit Source Routing on Packet Delivery Latency

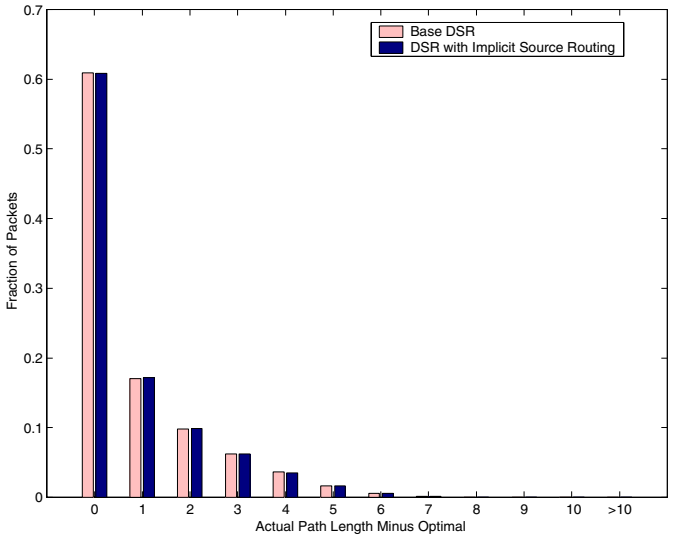


**Figure 3** Effect of Implicit Source Routing on Path Length Optimality

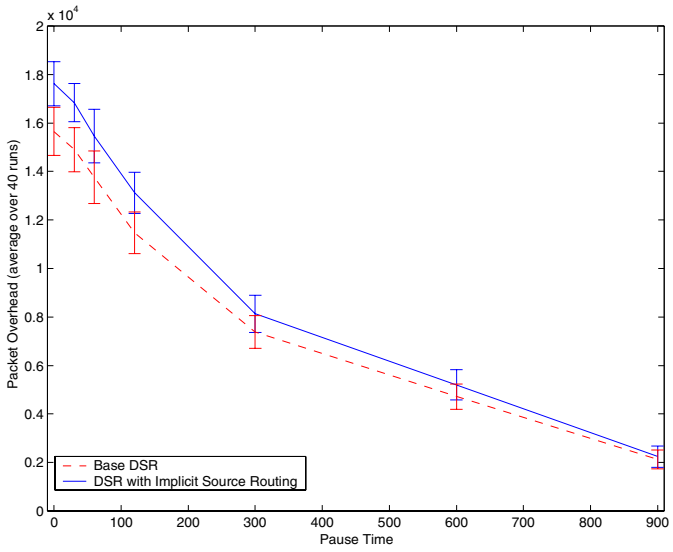


**Figure 4** Effect of Implicit Source Routing on Routing Packet Overhead

6]. The total bytes of overhead includes *all* bytes in overhead packets, *plus* the overhead such as any source routes and flow identifiers carried in data packets. In contrast to the routing packet overhead described above, the total bytes of overhead with implicit source routing decreased substantially over the base version of DSR. For example, with continuous mobility of all nodes, the total byte overhead for DSR decreased by 44% with implicit source routing, and in a stationary network, total byte overhead decreased by 86%.

Furthermore, by avoiding the need to include a source route header in every data packet, the savings in total bytes of overhead becomes proportional to the offered data packet load; in our simulations, each node originated only 4 packets per second in order to test the routing protocol's ability to find and maintain routes in a moving network, but many real applications would generate greater rates of data packets, further increasing the savings in total byte overhead with implicit source routing.

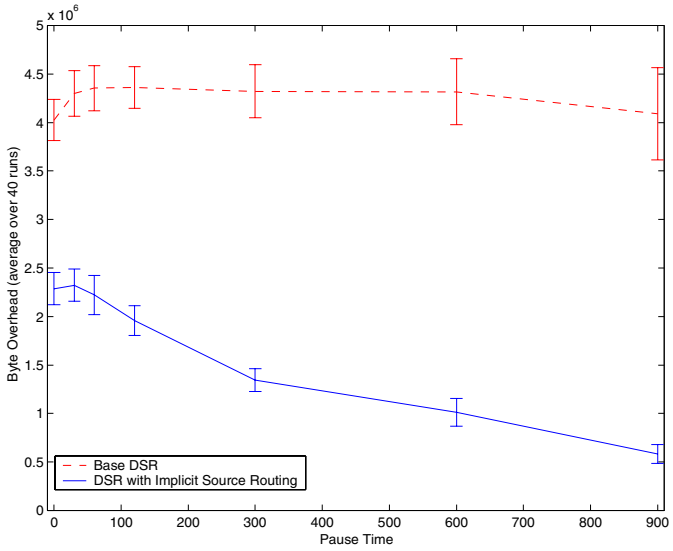


**Figure 5** Effect of Implicit Source Routing on Total Bytes of Overhead

## 6. Related Work

The concept of routing flows using per-hop state is a part of Multi-Protocol Label Switching (MPLS) [19] and ATM [21]. However, unlike such protocols, our implicit source routing technique is designed for use in an ad hoc network, where nodes may move and the network topology may change often (or continuously) and nodes generally forward packets over the same wireless network interface on which they received them. In addition, in MPLS and ATM, the flow identifier changes at each hop. This change prevents the use of default flows as defined in implicit source routing.

On-demand routing based on per-hop state is also present in AODV [18]; however, nodes using AODV are unable to take advantage of multiple paths to the same destination and cannot choose which sequence of hops a packet will take, though it may be desirable to do so as different hops have different characteristics, such as security, cost, and available bandwidth. AODV also relies on the hard state of a sequence number at each node to ensure loop-freedom of its routing, while all of the state in DSR with implicit source routing is soft state. In addition, AODV cannot utilize unidirectional links in the network for communication between nodes, whereas DSR, with or without implicit source routing, can fully support unidirectional links.

Many additional optimizations have been proposed for various portions of DSR, such as Location Aided Routing (LAR) [14] and core routing [20]. All of these modifications can be used with DSR with implicit source routing, although any optimizations that attempt to achieve better performance

through improved route selection may conflict with optimal route selection for minimizing overhead.

## 7. Conclusion

The use of source routing in an ad hoc network has many advantages, yet these advantages come at the cost of increased packet header size and thus increased routing overhead bytes. In this paper, we have presented the design and evaluation of *implicit source routing*, a technique that preserves the advantages of source routing while avoiding the associated per-packet overhead in most cases. In a manner in part similar to techniques used for MPLS [19] or ATM virtual circuits [21], the originator of a packet tags the packet a flow identifier implicitly indicating the sequence of hops through which the packet is to be forwarded on its way to its intended destination. All per-hop forwarding state is dynamically established when forwarding the first packet along this route and is maintained by each node along the route only as *soft state*; the soft state is automatically established as needed, and loss of any portion of this state has no effect on the correct operation of the protocol.

In addition, our implicit source routing mechanism includes support for DSR's *automatic route shortening* mechanism, allowing routes in use to be automatically shortened if one or more intermediate hops in the route become unnecessary, and DSR's *salvaging* mechanism, allowing packets to be forwarded along alternate routes if the original route for the packet encounters a broken link at some intermediate node. Implicit source routing also supports use of *default flows*, avoiding the need for a flow identifier header in the packet and thus avoiding *all* routing overhead in that packet.

We have evaluated this technique by extending the Dynamic Source Routing protocol (DSR) to include use of implicit source routing. Routing in DSR is based on source routing that is dynamically established on-demand when a sender needs a new route to some destination. The DSR protocol is simple and has been shown by a number of groups to perform well when compared to other protocols [1, 8]. Our implicit source routing mechanism fits naturally into the existing structure of the DSR protocol [6] and preserves the important fundamental properties of DSR's operation, including sender-selected routes, allowing the use of multiple routes to any destination, providing guarantees of loop-freedom even for packets sent with minimal overhead, routing based entirely on soft state, and the ability to use unidirectional links.

Although DSR makes extensive use of overheard routes for a number of important optimizations, applying implicit source routing to DSR improved both packet delivery ratio and latency; although routing packet overhead increased slightly, total bytes of routing overhead was reduced substantially. In particular, although routing packet overhead increased by about 12.3% with implicit source routing, *total* bytes of overhead decreased by between 44 and 86%; on all other metrics evaluated, the performance of DSR either did not change significantly or actually improved somewhat, due to indirect effects of the reduced routing overhead.

## Acknowledgements

We would like to thank David Maltz, who helped in designing an earlier version of the implicit source routing technique, and Amit Saha, who helped with the validation of our simulation model of the protocol and gave numerous comments and suggestions on the paper's presentation. We would also like to thank the other members of the Monarch Project for their valuable feedback on the design and presentation in this paper. Finally, we thank the anonymous reviewers for their comments that helped to improved the clarity of the presentation.

## References

[1] Josh Broch, David A. Maltz, David B. Johnson, Yih-Chun Hu, and Jorjeta G. Jetcheva. A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols. In *Proceedings of the Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking* (MobiCom'98), pages 85–97, October 1998.

[2] M. Scott Corson and Anthony Ephremides. A Distributed Routing Algorithm for Mobile Wireless Networks. *Wireless Networks*, 1(1):61–81, February 1995.

[3] Samir R. Das, Charles E. Perkins, and Elizabeth M. Royer. Performance Comparison of Two On-demand Routing Protocols for Ad Hoc Networks. In *Proceedings of the Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies* (INFOCOM 2000), pages 3–12, 2000.

[4] Kevin Fall and Kannan Varadhan, editors. *ns* Notes and Documentation. The VINT Project, UC Berkeley, LBL, USC/ISI, and Xerox PARC, November 1997. Available from http://www-mash.cs.berkeley.edu/ns/.

[5] Yih-Chun Hu and David B. Johnson. Caching Strategies in On-Demand Routing Protocols for Wireless Ad Hoc Networks. In *Proceedings of the Sixth Annual IEEE/ACM International Conference on Mobile Computing and Networking* (MobiCom 2000), pages 231–242, August 2000.

[6] Yih-Chun Hu, David B. Johnson, and David A. Maltz. Flow State in the Dynamic Source Routing Protocol for Mobile Ad Hoc Networks. Internet-Draft, draft-ietf-manet-dsrflow-00.txt, February 2001. Work in progress.

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

[8] Per Johansson, Tony Larsson, Nicklas Hedman, Bartosz Mielczarek, and Mikael Degermark. Scenario-based Performance Analysis of Routing Protocols for Mobile Ad-hoc Networks. In *Proceedings of the Fifth Annual ACM/IEEE International Conference on Mobile Computing and Networking* (MobiCom'99), pages 195–206, August 1999.

[9] David B. Johnson. Routing in Ad Hoc Networks of Mobile Hosts. In *Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications* (WMCSA'94), pages 158–163, December 1994.

[10] David B. Johnson and David A. Maltz. Dynamic Source Routing in Ad Hoc Wireless Networks. In *Mobile Computing*, edited by Tomasz Imielinski and Hank Korth, chapter 5, pages 153–181. Kluwer Academic Publishers, 1996.

[11] David B. Johnson, David A. Maltz, and Josh Broch. The Dynamic Source Routing Protocol for Multihop Wireless Ad Hoc Networks. In *Ad Hoc Networking*, edited by Charles E. Perkins, chapter 5, pages 139–172. Addison-Wesley, 2001.

[12] David B. Johnson, David A. Maltz, Yih-Chun Hu, and Jorjeta G. Jetcheva. The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks. Internet-Draft, draft-ietf-manet-dsr-05.txt, March 2001. Work in progress.

[13] John Jubin and Janet D. Tornow. The DARPA Packet Radio Network Protocols. *Proceedings of the IEEE*, 75(1):21–32, January 1987.

[14] Young-Bae Ko and Nitin Vaidya. Location-Aided Routing (LAR) in Mobile Ad Hoc Networks. In *Proceedings of the Fourth International Conference on Mobile Computing and Networking* (MobiCom'98), pages 66–75, October 1998.

[15] David A. Maltz, Josh Broch, Jorjeta Jetcheva, and David B. Johnson. The Effects of On-Demand Behavior in Routing Protocols for Multi-Hop Wireless Ad Hoc Networks. *IEEE Journal on Selected Areas in Communications*, Special Issue on "Wireless Ad Hoc Networks," 17(8):1439–1453, August 1999.

[16] The Monarch Project. Rice Monarch Project: Mobile Networking Architectures, project home page. Available at http://www.monarch.cs.rice.edu/.

[17] Charles E. Perkins and Pravin Bhagwat. Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers. In *Proceedings of the SIGCOMM '94 Conference on Communications Architectures, Protocols and Applications*, pages 234–244, August 1994.

[18] Charles E. Perkins and Elizabeth M. Royer. Ad-Hoc On-Demand Distance Vector Routing. In *Second IEEE Workshop on Mobile Computing Systems and Applications* (WMCSA'99), pages 90–100, February 1999.

[19] E. Rosen, A. Viswanathan, and R. Callon. Multiprotocol Label Switching Architecture. RFC 3031, January 2001.

[20] Prasun Sinha, Raghupathy Sivakumar, and Bharghavan Vaduvur. Enhancing Ad Hoc Routing with Dynamic Virtual Infrastructures. In *Proceedings of the Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies* (INFOCOM 2001), April 2001.

[21] Andrew S. Tanenbaum. *Computer Networks*. Prentice Hall, New Jersey, 1996.