Eastern Illinois University
The Keep

Faculty Research & Creative Activity

Technology, School of

January 2007

# Enhancing Performance by Salvaging Route Reply Messages in On-Demand Routing Protocols for MANETs

Rendong Bai
*Eastern Illinois University*, rbai@eiu.edu

Mukesh Singhal
*University of Kentucky*

Yi Luo
*University of Kentucky*

## Recommended Citation

Bai, Rendong; Singhal, Mukesh; and Luo, Yi, "Enhancing Performance by Salvaging Route Reply Messages in On-Demand Routing Protocols for MANETs" (2007). *Faculty Research & Creative Activity*. 8.
http://thekeep.eiu.edu/tech_fac/8

*Ad Hoc & Sensor Wireless Networks* Vol. 5, pp. 161–188
Reprints available directly from the publisher
Photocopying permitted by license only

# Enhancing Performance by Salvaging Route Reply Messages in On-Demand Routing Protocols for MANETs

RENDONG BAI[1,2], MUKESH SINGHAL[1] AND YI LUO[1]

[1]*Department of Computer Science, University of Kentucky, Lexington, KY 40506*
[2]*Department of Computer Science, Eastern Kentucky University, Richmond, KY 40475*
*E-mail: rendong.bai@eku.edu, singhal@cs.uky.edu, yiluo@cs.uky.edu*

*Received: April 3, 2007. Accepted: August 8, 2007.*

Researchers prefer on-demand routing protocols in mobile ad hoc networks where resources such as energy and bandwidth are constrained. In these protocols, a source discovers a route to a destination typically by flooding the entire or a part of the network with a route request (RREQ) message. The destination responds by sending a route reply (RREP) message to the source. The RREP travels hop by hop on the discovered route in the reverse direction or on another route to the source. Sometimes the RREP can not be sent to the intended next hop by an intermediate node due to node mobility or network congestion. Existing on-demand routing protocols handle the undeliverable RREP as a normal data packet - discard the packet and initiate a route error message. This is highly undesirable because a RREP message has a lot at stake – it is obtained at the cost of a large number of RREQ transmissions, which is an expensive and time-consuming process. In this paper, we propose the idea of salvaging route reply (SRR) to improve the performance of on-demand routing protocols. We present two schemes to salvage an undeliverable RREP. Scheme one actively sends a one-hop salvage request message to find an alternative path to the source, while scheme two passively maintains a backup path to the source. Furthermore, we present the design of two SRR schemes in AODV and prove that routes are loop-free after a salvaging. We conduct extensive simulations to evaluate the performance of SRR, and the simulation results confirm the effectiveness of the SRR approach.

*Keywords:* Mobile ad hoc networks, on-demand routing, AODV, performance enhancement, salvaging.

## I INTRODUCTION

Mobile ad hoc networks (MANETs) have received considerable attention during the past decade, because they can be conveniently deployed without a fixed infrastructure. Despite their desirable features, MANETs have not found many widespread civilian applications. This may raise some doubts about the practicability of MANETs. We address this issue by viewing MANETs as a prototype of future wireless networks that operate in a peer-to-peer manner. MANETs inherently have the property of polymorphism. By adding or removing certain constraints such as mobility and power supply, we can transform a MANET into different variations, for example, a mesh network, a delay tolerant network or a sensor network. The latter three types of wireless networks have been deployed, e.g., Roofnet [1], NetEquality [2], Interplanetary Internet [3] and Smart Surrogates [4]. Sometimes these networks are even irreplaceable because of cost or deployment environment. In order to gain insights into these peer-to-peer wireless networks and improve their applicability and practicability, it is important to identify and address fundamental problems and shortcomings in MANETs. Such a knowledge or an improvement to MANETs will potentially benefit all the peer-to-peer wireless networks.

Routing in MANETs is a challenging task due to random topology changes and frequent route breakages. Researchers have proposed many routing protocols for MANETs (see [5] for a review). We can classify these protocols into different categories according to different criteria. If based on whether they require nodes to have positioning capability, we can classify routing protocols into topology-based (e.g., [6][7]) and position-based protocols (e.g., [8][9]). If based on the way they establish and maintain routes, we can classify routing protocols into proactive (or table-driven, e.g., [10][11]) and reactive (or on-demand) protocols (e.g., [6][7]) (though hybrid protocols also exist, e.g., [12]).

Proactive routing protocols require nodes to exchange routing information (e.g., the information of one-hop neighbors of a node) periodically and compute routes continuously between any nodes in the network, regardless of if the routes will be used or not. As a result, a lot of network resources such as energy and bandwidth may be wasted. This is not desirable in MANETs where resources are constrained. On the other hand, on-demand routing protocols don't exchange routing information periodically. Instead, they discover a route only when it is needed for the communication between two nodes. Previous work [13][14][15] shows that on-demand routing protocols perform better than proactive routing protocols. We focus on on-demand routing protocols in this paper.

An on-demand routing protocol typically consists of two components: route discovery and route maintenance. Route discovery happens when a source node (say $S$) has data packets to send to a destination node (say $D$) but $S$ doesn't have a route to $D$ in its routing table. To establish a route to $D$, $S$ broadcasts a *route request (RREQ)* message searching for $D$. The RREQ message is propagated throughout the entire network or a limited scope, based on the TTL (time to live, generally using hop count) of the RREQ. When the RREQ reaches the destination $D$, $D$ sends a *route reply (RREP)* message to $S$.

Other intermediate nodes that have a route to $D$ in their routing tables may send a *cached* RREP to $S$. Nodes transmit RREP using unicast. When $S$ receives the RREP, it discovers a route to $D$ and uses this route to send data packets to $D$. Route maintenance deals with routing information at nodes, typically involving three operations: handling route errors, deleting stale route entries, and learning new routes from the traffic.

Due to the dynamic nature of MANETs, links between nodes tend to be ephemeral – new connections occur often but only exist for a short time. Link failure causes packet losses and the losses inevitably include route reply packets. Among existing on-demand routing protocols, to the best of our knowledge, no protocol takes special care to prevent a route reply message from being lost. When a route reply message can not be delivered to the intended next hop node, the common response is to discard the message and send a route error message to the destination (initiator of the RREP). In our opinion, simply discarding undeliverable RREP messages is very wasteful. This is because a RREP message has a lot at stake, i.e., a RREP message costs a considerable amount of route discovery overhead. If undeliverable RREP messages can be salvaged and delivered (possibly with a repair to the route), we can save a large amount of route discovery overhead and achieve a noticeable performance improvement.

In this paper, we propose the idea of salvaging route reply (SRR) for on-demand routing protocols. Although DSR (dynamic source routing) [7] uses salvaging, the subject of salvaging is data packets rather than route reply packets. We compare SRR with salvaging in DSR in detail in Section E. In SRR, when an intermediate node can not deliver a RREP to the intended next hop node, it tries to salvage the RREP. We call this node the *salvor*.

We present two SRR schemes. In scheme one, the salvor actively broadcasts a salvage request message to its one-hop neighbors, asking for a cached path to the source. We propose scheme two as an improvement to scheme one. In scheme two, intermediate nodes passively maintain a backup path to the source utilizing duplicate RREQ packets. When the RREP can not be relayed using the original path to the source, the salvor directly switches to the backup path. Therefore, SRR introduces very little extra overhead (in scheme one) or no extra overhead (in scheme two) to the routing protocol, but it has potential to substantially reduce the overhead of route discovery and significantly improve other performance metrics such as packet delivery ratio and end-to-end delay.

We present the implementation of both SRR schemes in AODV (ad hoc on-demand distance vector) [6] routing protocol[1]. Additionally, we prove that in the implementation, routes are loop-free after a salvaging. We conduct extensive simulations to evaluate the performance of SRR in conjunction with

[1] We also point out guidelines for the implementation of SRR in other on-demand routing protocols.

AODV. The results show that SRR improves the routing performance significantly in a wide range of system parameter values and in all critical metrics, including packet delivery ratio, control overhead and end-to-end delay.

The rest of the paper is organized as follows. In the next section, we discuss the motivation for SRR. In Section III, we present two SRR schemes. The design of SRR in AODV is described in Section IV. Section V presents simulation setup and results. We review related work in Section VI and draw conclusions in Section VII.

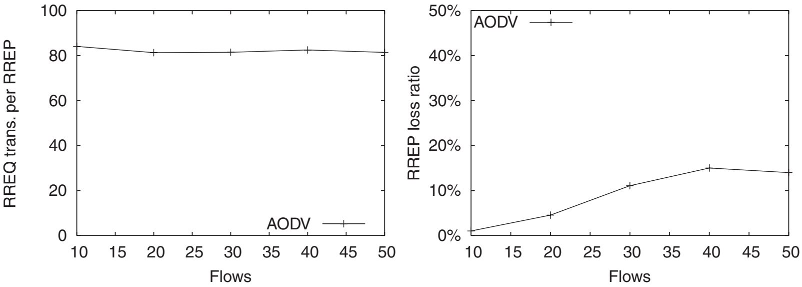
## II MOTIVATIONS

In MANETs, a node may move from one location to another, turn its power on and off, join and leave the network, and the hardware of a node may fail. As a result, the network topology changes constantly and unpredictably. A link between two neighboring nodes may last for only a short period of time, or temporarily become congested. It is a common incidence that a node can not successfully send a packet to its intended next hop node. When a node can not access the shared medium because the medium is continuously busy, or when a node can not receive a link-layer or network-layer acknowledgment after transmitting the packet, the node concludes that it failed to send the packet. In most cases, the node discards the undeliverable packet. The packet loss causes certain impairment to the routing performance.

Among different packet losses, the loss of RREP packets causes the most serious impairment to the routing protocol. Usually, a RREP packet is obtained at the cost of flooding the entire network or a limited scope. In other words, the knowledge of a RREP message is obtained through tens, may be hundreds transmissions of RREQ messages, which is an expensive and time-consuming process. Simply discarding an undeliverable RREP wastes a lot of route discovery effort. Furthermore, because the RREP is lost, the source node may have to initiate another round of route discovery, degrading the routing performance, for example, increasing routing overhead and end-to-end delay. Therefore, in on-demand routing protocols, RREP packets should be given a higher priority than other packets. Unfortunately, to the best of our knowledge, no existing routing protocol attempts to salvage a RREP packet when it can not be delivered to the intended next hop node.

In order to illustrate the importance of RREP packets, we conducted a set of simulations, in which the routing protocol was AODV and the traffic was CBR (constant bit rate). The traffic load varied from 10 to 50 flows. Other setups are as described in Section V. We only considered RREPs initiated by destinations[2]. Figure 1(a) shows the number of RREQ transmissions needed

[2] Intermediate nodes that have a route to the destination may answer a route request with a RREP message. We only salvage RREPs generated by destinations.



(a) RREQ transmissions per RREP.

(b) Loss Ratio of RREPs

FIGURE 1
The cost and loss ratio of RREP packets in AODV. Network area 1400 × $1400m^2$, simulation time $10min$, 100 nodes, 10–50 flows, maximum speed $20m/s$ and pause time $30s$.

for destinations to initiate one RREP packet. On the average, each RREP is generated at the cost of more than 80 transmissions of the route request message. Figure 1(b) shows the loss ratio of RREP packets. In general, the loss ratio increases as the number of flows increases because the network becomes more congested. When the number of flows is 50, 14% of total RREP packets, or as many as 2053 RREP packets, are lost. This means more than $1.6 \times 10^5$ ($80 \times 2053$) transmissions of route request messages are wasted. These results justify that it is necessary to salvage undeliverable RREP packets.

We propose the idea of salvaging route reply (SRR) messages to minimize the loss of RREP packets and thus reduce the number of route discoveries. When a RREP is undeliverable due to link break or channel congestion, SRR tries to find an alternative path to relay the RREP to the source node, which initiated the route discovery and is waiting for a reply. SRR is practical and applicable to all on-demand routing protocols. The direct advantage of SRR is a substantial saving of control overhead. Less overhead means less congestion. Less congestion results in shorter end-to-end delay and higher packet delivery ratio because data packets travel faster in the network and experience less “overflow” (buffer is full) and “outdated” (packets wait too long) drops.

## III SRR: SALVAGING ROUTE REPLY

In a route discovery, the destination sends a RREP message to the source after receiving the RREQ message. The RREP travels hop by hop towards the source, on the discovered route in reverse direction or on another route. During the relaying of the RREP, when an intermediate node can not deliver the message to the intended next hop node because the next hop is unreachable, the intermediate node becomes a salvor and starts the SRR. In this section, we first present two schemes of SRR, then discuss several guidelines for SRR design. Moreover, we discuss MAC layer considerations for SRR and compare SRR with data packet salvaging in DSR.

## A SRR Scheme One (SRR1)

In SRR scheme one, the salvor runs a local route discovery to find an alternative path to the source. The salvor starts the discovery by broadcasting a salvage request message (*SREQ*). The SREQ has a limited propagation scope. We use one hop, which means only one transmission for a SREQ message. This local SRR route discovery can be viewed as being "embedded" into the original route discovery whose RREP is being salvaged. Upon receiving the SREQ, if a neighboring node of the salvor has a route to the source or it is the source itself, it sends a salvage reply message (*SREP*) to the salvor. On receiving the SREP, the salvor sends the undeliverable RREP to the source using the alternative path returned by the SREP. The salvor sets a timer when starting the SRR route discovery. If no SREP returns when the timer expires, the SRR route discovery fails and the salvor discards the RREP being salvaged.

A one-hop SRR route discovery will find an alternative path to the source most of time due to the following reasons. If the source is a one-hop neighbor of the salvor, it can send a SREP to the salvor directly. Otherwise, the one-hop neighbors of the salvor should have a route to the source in their routing tables. This is because when the original RREQ message was propagated outward from the source, intermediate nodes that relayed the RREQ learned a route to the source from this message[3]. For example, in AODV, the last node from which the current node received the RREQ will be the next hop from current node to the source. In DSR, the reversal of the recorded path in a RREQ will be the route from the current node to the source. Therefore, although the salvor does not have a path to the source since the old path is broken, it is highly likely that some of its neighbors have such a path and are able to answer the salvor's salvage request.

## B An Improvement: SRR Scheme Two (SRR2)

The SRR scheme one is an elegant approach because a salvor broadcasts a salvage request only once and it is very likely that it can find an alternative path to relay the undeliverable RREP message. However, the SRR1 introduces extra control messages, i.e., SREQ and SREP. This has two shortcomings. First, the extra messages increase the complexity of a routing protocol. Second, the salvaging process adds a bit of delay to the route discovery process, because the salvor needs to send a SREQ and wait for a SREP (of course, compared to the payoff, these shortcomings are very nominal).

[3] Bidirectional links are assumed. We discuss the situation of unidirectional links in Section D

Therefore, we are motivated to further improve the SRR and develop the SRR scheme two, which requires no extra control message when salvaging an undeliverable RREP. Now the challenge is how we can find an alternative path from the salvor to the source without sending any salvage request. We solve this problem by focusing on and utilizing duplicate RREQ packets. In existing on-demand routing protocols, upon receiving RREQ packets for a route discovery, intermediate nodes typically only handle the first received RREQ and discard duplicate RREQs they receive later. The first or a duplicate RREQ can be distinguished because each RREQ message is identified, typically by the combination of a source id and a sequence number.

In SRR2, intermediate nodes do not relay (broadcast) duplicate RREQs either. However, intermediate nodes examine duplicate RREQs because a duplicate RREQ typically provides another path to the source. This path can be stored at nodes as an alternative path to the source. Therefore, in addition to the primary path learned from the first RREQ, intermediate nodes also maintain a secondary path learned from duplicate RREQs. The secondary path serves as a backup route to the source in case the primary path is broken.

When relaying a RREP message, intermediate nodes first use the primary path. When the next hop on the primary path is unreachable, the intermediate node switches to the secondary path. Thus, the RREP that otherwise would be dropped is very likely to be salvaged.

One issue is how to choose the secondary path, because an intermediate node may receive multiple duplicate RREQs that represent for multiple paths to the source. A node can select the secondary path based on different criterion. For example, the node may choose the path obtained from the second RREQ since it is the second fastest, or choose the path according to other metrics, e.g., minimal hop count.

*Single-Path and Multiple-Path Routing:* Protocols such as AODV [6] are inherently single-path-based. Thus, it is necessary for SRR2 to check duplicate RREQs and learn a backup path to the source. Some protocols, e.g., DSR, are able to find multiple paths. So it may seem unnecessary to learn a backup path from duplicate RREQs, because a salvor may already have multiple paths to the source. However, when used without significant modifications, most of the paths discovered by DSR share a large number of common nodes–only the last few hops are different [16]. Therefore, we believe that SRR2 is useful for protocols such as DSR as well.

*Maintaining a Backup Path:* In on-demand routing protocols, nodes maintain a *request_seen* table to record route requests they have received. This recording prevents nodes from forwarding duplicate requests received later for the same route discovery. We use the *trequest_seen* table to maintain backup paths. We make no change to the routing table in the original routing protocol. Thus, when applying SRR2 to an existing routing protocol, changes to the original protocol are minimal.

## C Guidelines for the SRR Design

Because SRR is an improvement to existing on-demand routing protocols, its operation should be simple and its overhead should be little. We give some guidelines for SRR design:

- One RREP packet can be salvaged at most once. After a RREP is salvaged, perhaps the quality of the salvaged route from the destination to the source is not good (say, not optimal or not stable). Additional link breakages on the route further indicate its bad quality. More salvaging efforts for such a route may not be worthwhile. Moreover, salvaging an undeliverable RREP packet only once helps prevent the RREP from entering a routing loop because different nodes may salvage the packet and send it to each other.

- The RREPs generated by intermediate nodes (denoted as $\text{RREP}_{in}$s) should not be salvaged. This is for two reasons. First, an intermediate node generates a $\text{RREP}_{in}$ because it has a route to the destination. However, the route known by this node may be stale. Salvaging such a $\text{RREP}_{in}$ will only incur extra overhead. Second, for a RREQ, the number of $\text{RREP}_{in}$s generated by intermediate nodes may be large, which indicates the number of undeliverable $\text{RREP}_{in}$s is increasing as well. If counting $\text{RREP}_{in}$s for salvaging, SRR may introduce a large overhead to the routing protocol. Therefore, we recommend that SRR should only salvage the RREPs generated by the destinations of route discoveries.

- We need to consider the issue of loop freedom in SRR. Loop freedom is a key requirement for ad hoc routing protocols because routing loops waste resources and may degrade the routing performance seriously. After SRR salvages an undeliverable RREP, we must ensure that the route after the salvaging is loop-free. This issue is less serious in protocols that use source routing, in which packet headers carry routes and thus routing protocols can detect a loop easily. However, when used in some other protocols, e.g., protocols using distance vector, we should carefully design the operation of SRR to prevent routing loops from being formed.

- In scheme one, a node should salvage at most one RREP packet at a time. SRR route discovery transmits SREQ and SREP, which increase the traffic around the salvor. Multiple SRRs that run simultaneously at a salvor may cause congestion around the salvor and its neighbors.

- In scheme one, the scope of SRR route discovery should be as small as possible. We recommend that it should only cover one-hop neighbors of the salvor, i.e., the initial TTL value of a SREQ should be set to one. This means a SREQ message needs only one transmission.

## D MAC Layer Considerations

Majority of the routing protocols for MANETs tend to assume bidirectional links, for example, AODV, TORA [17] and proactive protocols based on distance vector routing. With bidirectional links assumption, the implementation of SRR becomes easier. Otherwise, intermediate nodes may not be able to obtain a reverse path to the source from route request messages. Routing protocols require additional mechanisms to handle unidirectional links in the network.

Marina [18] shows the advantage of using unidirectional links is almost non-existent and proposes a *ReversePathSearch* technique to eliminate unidirectional links from route computations. Similar techniques include BlackListing [19] and Hello [20]. When using such techniques, a reverse path from an intermediate node to a source still holds because a route uses only bidirectional links.

If a routing protocol indeed needs to use unidirectional links, Duros *et al.* [21] propose a *tunneling* mechanism to emulate bidirectional connectivity upon unidirectional links. Thus the network layer protocols can still assume bidirectional links.

Another issue we need to consider in SRR is the saving of RREP packets being salvaged. When the MAC layer gives up the transmitting of a unicast packet, it notifies the routing protocol at the network layer. Some MAC layer protocols (or the implementations) return the undeliverable packet as well. For example, the implementations of IEEE 802.11 MAC protocol [22] in simulators including GloMoSim [23] and ns-2 [24] hand the undeliverable packet back up to the network layer. In such cases, SRR can automatically know the information about an undeliverable RREP packet. Otherwise, SRR needs to save the RREP packet before the routing protocol hands it down to the MAC layer.

## E A Comparison to Packet Salvaging in DSR

In DSR, when a node fails to send a data packet to the intended next hop node, in addition to sending a route error message to the source of the packet, it attempts to salvage the packet by looking up its own route cache for an alternate route to the destination. If such a route exists, the node makes necessary changes to the packet header and sends the packet using this route. Once DSR salvages a data packet, it marks the packet as *salvaged*. This is to prevent the packet from being salvaged multiple times. Otherwise, the packet may enter a routing loop because different nodes may salvage the packet and send it to each other.

SRR has two important differences compared to packet salvaging in DSR. First, the salvaging subject is different. SRR salvages undeliverable RREP packets which are control packets, while DSR salvages undeliverable data packets. RREP packets are more important than data packets for the performance of on-demand routing protocols. Second, the salvaging approach is different. SRR1 conducts a one-hop route discovery, and SRR2 utilizes duplicate RREQs to prepare a backup path to the source in advance. While the salvaging in DSR only looks up the node's route cache. It neither sends out extra routing messages nor looks into duplicate RREQs.

The differences between SRR and packet salvaging in DSR do not imply that SRR can not be applied to certain on-demand routing protocols. In fact, SRR can be incorporated into all on-demand protocols in which the route discovery works through broadcasting a RREQ message and waiting for a RREP message.

## IV SALVAGING ROUTE REPLY IN AODV

We now present the implementation of SRR scheme one and scheme two in AODV [6]. We choose AODV because it is a prominent and widely used routing protocol. AODV is based on traditional distance vector routing scheme, and it makes routing decision on a hop-by-hop basis. AODV discovers a route by adding a backward routing entry pointing to the source at intermediate nodes when they propagate the RREQ message, and by adding a forward routing entry pointing to the destination at intermediate nodes when they relay the RREP message to the source. In the following discussion, when we mention the *original* route discovery, we mean the route discovery whose RREP SRR is salvaging. When we mention the *source*, we mean the source of the original route discovery.

During a route discovery, after the destination receives the RREQ message, it sends a RREP message to the source via intermediate nodes. If an intermediate node can not deliver the RREP because the intended next hop neighbor is unreachable, the node tries to salvage the RREP. First, the node checks two conditions. (i) The RREP is not generated by an intermediate node. (ii) The RREP has not been salvaged before (it is not marked as *salvaged*). If the RREP satisfies both conditions, the node tries to salvage the RREP using either SRR1 or SRR2.

Next, we discuss how to prevent a loop from being formed on the route after salvaging a RREP. Then, we present the design details of SRR1 and SRR2 in AODV. At the end of this section, we discuss the use of a new message type called route update message. We discuss the loop prevention first because both scheme one and scheme two use it.

### A Loop Freedom

The salvor tries to find a new next hop to the source, either by broadcasting a SREQ message or by maintaining a backup path to the source when the salvor receives duplicate RREQs. In this subsection, we discuss the issue of loop freedom when using SRR in AODV. We use the following notations in the discussion:

| | |
|---|---|
| $Path(A, B), Path_{srr}(A, B)$ : | Path from node $A$ to node $B$ before and after SRR, respectively. |
| $NextHop(A, B, \mathcal{P})$ : | The next hop from node $A$ to node $B$ on path $\mathcal{P}$. |
| $HopCount(\mathcal{P})$ : | The number of hops on path $\mathcal{P}$. |
| $S, D, X$ : | Source, destination and salvor nodes, respectively. |

We define two conditions for preventing a loop from being formed on the new path after using SRR:

Condition 1: $NextHop(NextHop(X, S, Path_{srr}(S, A)), S, Path_{srr}(S, A)) \neq X$

Condition 2: $HopCount(Path_{srr}(X, S)) \leq HopCount(Path(X, S)) + 1$

Condition 1 means the new next hop node does not use the salvor as the next hop to the source. Condition 2 means the new path has at most one more hop than the broken path. We now prove that when the above two conditions hold, the route is loop-free after SRR salvages the RREP.

*Theorem 1. When both Conditions 1 and 2 hold, the route is loop-free after SRR salvages the RREP.*

*proof*: The proof is by contradiction. Assume after the salvaging, the route contains a loop. We call the node that the route passes through twice the *loop node*. The order of the salvor ($X$) and the loop node on the route have three possibilities: (1) $X$ is before the loop node; (2) $X$ is after the loop node; (3) $X$ is the loop node. Figure 2 (a), (b) and (c) show the three cases, respectively.

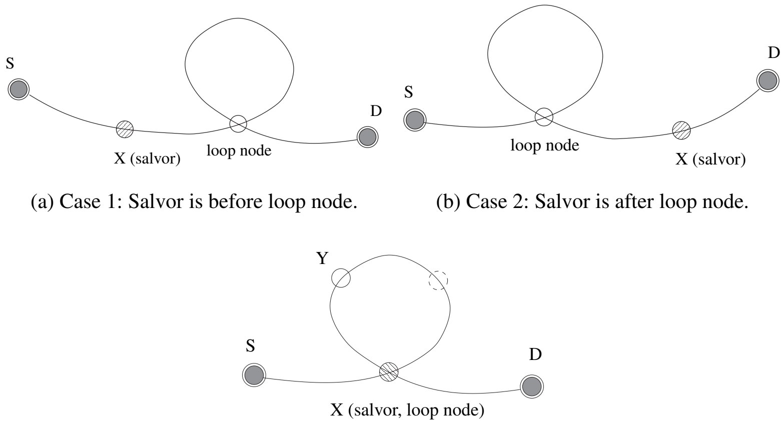
Cases (1) and (2) mean the loop node forwarded the RREQ message twice, which is not possible because in AODV nodes forward the RREQ message for a route discovery at most once.

For case (3), besides the salvor $X$, there should be one or more nodes in the loop. When there are two nodes in the loop, the two nodes would be $X$ and another node $Y$. Then, $Y$ would use $X$ as the next hop to the source, i.e., $NextHop(NextHop(X, S, Path_{srr}(S, A)), S, Path_{srr}(S, A)) = X$. This contradicts condition (i). When there are more than two nodes in the loop, the new path would have at least two more hops than the broken path, i.e., $HopCount(Path_{srr}(X, S)) > HopCount(Path(X, S)) + 1$. This contradicts condition (ii).

Therefore, the assumption that the route contains a loop is incorrect. The two conditions guarantee loop freedom for the SRR approach in AODV. □

## B SRR Scheme One for AODV (AODV-SRR1)

In scheme one, if the node is not salvaging any other RREPs, it acts as a salvor and starts the SRR procedure. The salvor first saves the undeliverable RREP



(a) Case 1: Salvor is before loop node.

(b) Case 2: Salvor is after loop node.

(c) Case 3: Salvor is loop node.

FIGURE 2
Three possible orders of salvor and loop node if assuming a loop occurs after salvaging a RREP.

message. This will also prevent the salvor from salvaging other RREPs at the same time. Then, the salvor initiates a SRR route discovery by broadcasting a SREQ message. In the message, the *initiator* field is set to the salvor, the *target* field is set to the source (of the original route discovery), and the *TTL* field is set to one. The message also carries the hop count of the broken path (from the salvor to the source).

If the source receives the SREQ, the source sends a SREP to the salvor directly. Otherwise, the node that receives the SREQ looks up its routing table for a route to the source. As discussed in Section A, one-hop neighbors of the salvor should have a route to the source because they recently propagated the original route request message and learned a reverse path to the source. If such a route exists, the node checks the two loop-prevention conditions discussed in Section A. If both conditions are satisfied, the node responds to the salvor with a SREP message.

The salvor waits a short period of time (e.g., 2 *NODE_TRAVERSAL_TIME [19]) for the SREPs from its neighbors. Then it selects the best route from them according to the standard route update rules in AODV: use the route with newer sequence number; when sequence numbers are same, use the route with fewer hop count. Next, the salvor salvages the saved RREP by sending it to the source using the path just discovered. The salvor marks the RREP as *salvaged* to prevent other intermediate nodes from salvaging it again.

We give an example in Figure 3 to explain how AODV-SRR1 works. Node $S$ is discovering a route to node $D$. $D$ sends a RREP to $S$ and the original return path is $D \rightarrow e \rightarrow c \rightarrow b \rightarrow a \rightarrow S$. Node $c$ can not send the RREP to node $b$ because $b$ moves away. Node $c$ becomes the salvor and broadcasts a SREQ. Node $v$ receives the SREQ and has a route to $S$ in its routing table,

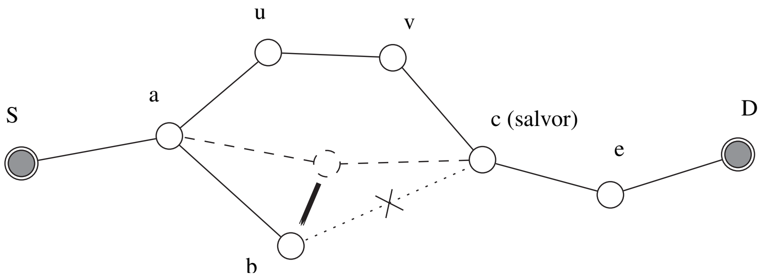


FIGURE 3
An example of SRR. Link $b - c$ is broken. Salvor node is c. Intended RREP return path is $D \rightarrow e \rightarrow c \rightarrow b \rightarrow a \rightarrow S$. Actual return path after SRR is $D \rightarrow e \rightarrow c \rightarrow v \rightarrow u \rightarrow a \rightarrow S$.

so $v$ responds a SREP to $c$. $c$ receives the SREP and successfully salvages the RREP by relaying it on the path discovered by SRR. Thus the return path after SRR is $D \rightarrow e \rightarrow c \rightarrow v \rightarrow u \rightarrow a \rightarrow S$.

### C SRR Scheme Two for AODV (AODV-SRR2)

In scheme two, during a route discovery process, intermediate nodes handle the first received RREQ packet as in the original AODV. When intermediate nodes receive a duplicate RREQ packet, which represents a new path to the source, they decide whether the new path can be used as a secondary path to the source. If a node does not have a secondary path yet, the node uses the new path as the secondary path if the new path satisfies the two loop-prevention conditions in Section A. If the node already has a secondary path, it uses the new path if the new path has fewer hop count than the old path.

Later on, when an intermediate node can not send the RREP to the next hop on the primary path, if the node has cached a secondary path to the source, the node marks the RREP as *salvaged* and relay it using the secondary path. We illustrate how AODV-SRR2 works using Figure 3. For node $c$, the primary path to source $S$ is $c \rightarrow b \rightarrow a \rightarrow S$ and the secondary path is $c \rightarrow v \rightarrow u \rightarrow a \rightarrow S$. When the primary path is broken, $c$ salvages the RREP by relaying it using the secondary path.

### D Route Update Message (RUPD)

After the salvor salvages the undeliverable RREP using either AODV-SRR1 or AODV-SRR2, downstream nodes on the route (e.g., nodes $e$ and $D$ in Figure 3) may have incorrect route information to the source, for example, incorrect hop count to the source or incorrect sequence number for the source. This situation happens when the new path has different hop count from the broken path, or when downstream nodes have an outdated sequence number for the source. For the example in Figure 3, originally nodes $e$ and $D$ are 4 and 5 hops away from the source, respectively. But after the SRR, the distances become 5 hops and 6 hops. Therefore, SRR needs to notify downstream nodes the correct route information to the source.

We define a new message type for SRR, *route update (RUPD)*, to handle this situation. A RUPD is a unicast message that SRR uses to inform downstream nodes on the salvaged route about the source information: First, the sequence number (*SN*) of the source known by the salvor; Second, the hop count to the source from the node that receives the RUPD. The following is the format of RUPD message:

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Type (RUPD)  |           Reserved            |   Hop Count   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Source Address                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Source Sequence Number                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

We illustrate how RUPD message works using the example in Figure 3. After salvaging the RREP, the salvor *c* sends a RUPD message to node *e*. The RUPD contains the *SN* of the source known by *c* and a hop count of 5. When *e* receives the RUPD, it updates its routing table accordingly. Then, node *e* changes the hop count value in RUPD to 6 and sends it to node *D*. *D* processes the RUPD similarly but no longer relays it because it is the destination.

The RUPD message works well with AODV-SRR1, which has already introduced the SREQ and the SREP messages. However, for AODV-SRR2, the use of RUPD message is undesirable, because a major design objective of SRR2 is to eliminate extra control messages. We solve this problem by using a RUPD IP option in the header of a data packet. The following is the format of RUPD option:

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |    Length     |   Reserved    |   Hop Count   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Source Address                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Source Sequence Number                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The `Type` octet has 3 fields: `copy flag` (1 bit), `option class` (2 bits) and `option number` (5 bits). RFC 791 [25] defines these fields and gives the descriptions of them. RFC 3692 [26] and RFC 4727 [27] give recommendations for assigning experimental and testing numbers when adding new functions using IP option. In our simulations, we set `copy flag` to 1, `option class` to 0, and `option number` to 30. Thus, the value of `Type` is 158. Because the RUPD option uses 12 bytes, the value of `Length` octet is 12.

When adding a RUPD option to a data packet, we increase the IP header length (`IHL`, in 32-bit words) of the packet by 3, and append the RUPD option to the end of the IP header. To determine if a RUPD option exists in the IP header of a data packet, we check two conditions. First, the header length should be greater than 5, meaning the header carries one or more options. Second, the `Type` value of some option should be equal to the `Type` value of RUPD (158).

After the source receives a RREP marked as $salvaged$ and before the source uses the discovered route to send buffered data packets, the source inserts a RUPD option to the header of the first data packet to be sent. Upon receiving a data packet that contains a RUPD option in its header, intermediate nodes update their routing tables according to the RUPD option.

## V PERFORMANCE EVALUATION

We conducted extensive simulations to evaluate the performance of SRR. We simulated AODV-SRR1 and AODV-SRR2 and compared it with AODV. In this section, we first introduce the simulation setup and then present the results and analysis.

### A Simulation Setup

We conducted simulations using GloMoSim 2.03 [23], a scalable simulation environment for wireless network systems. GloMoSim uses the parallel discrete-event simulation capability provided by PARSEC [28]. The MAC layer protocol was the Distributed Coordination Function (DCF) of IEEE 802.11 [22]. DCF uses Request-To-Send (RTS) and Clear-To-Send (CTS) control packets for unicast transmissions. The MAC protocol sends broadcast packets using the unslotted Carrier Sense Multiple Access protocol with Collision Avoidance (CSMA/CA) [22]. The propagation model was the two-ray ground reflection model [29]. The radio bandwidth was $2Mb/s$ and the radio range was $250m$.

The traffic was constant bit rate (CBR). We randomly selected the source and the destination of each CBR flow but they were not identical (sources and destinations of different flows might coincide). Each simulation lasted for 1200 seconds. We did not change the source and the destination of a flow during the lifetime of a simulation run. After a simulation started, each source waited a *warm-up* time, which we randomly selected from 60 to 100 seconds, before sending data packets. The size of data packets was 512 bytes. Each source stopped sending data packets 20 seconds before the simulation ended. We call this period the *cool-down* time,

The mobility model was random waypoint [13], which randomly chose the speed of a node between a minimum and a maximum value. A node stayed for a pause time after reaching a waypoint. A zero pause time means nodes move continuously. Unless specified otherwise, the following configurations for the traffic and the mobility were common among many of our simulations: CBR sending rate $4packets/s$, minimum node speed $0.1m/s$, maximum node speed $20m/s$ and pause time $30s$.

In the performance evaluation, we first studied AODV-SRR1, AODV-SRR2 and AODV in networks containing 100 nodes in an area of $1400 \times 1400m^2$. We organized the 100-node simulations into four groups by varying four network parameters, namely, the number of CBR flows, packets sending rate at sources, maximum node moving speed and node pause time, respectively. Table 1 summaries the setups for these four groups of simulations.

We then studied AODV-SRR1, AODV-SRR2 and AODV in larger networks. The number of nodes was varied from 100 to 300, and the number of flows was 20% of total number of nodes. For example, in a 300-node network, the number of flows was 60. To scale the simulations from small networks to larger networks, we maintained constant node density by increasing the network area accordingly. Table 2 gives the setups for this set of simulations.

In the performance study, we focused on three key metrics that researchers widely used in evaluating the performance of routing protocols: *packet delivery ratio (PDR)*, *control overhead* and *end-to-end delay*. Moreover, we collected the number of SRR control packets, including SREQ, SREP and RUPD, for AODV-SRR1 in 100-node simulations with varied number of flows. This would reflect the advantage of SRR scheme two because compared to AODV-SRR1, AODV-SRR2 requires no extra control packets. Each

| Parameter | Flows | Load (pkts/s) | Max Speed(m/s) | Pause Time (s) |
|---|---|---|---|---|
| Flows | 10–50 | 4 | 20 | 30 |
| Load | 20 | 4–14 | 20 | 30 |
| Max Speed | 30 | 4 | 5–30 | 0 |
| Pause Time | 30 | 4 | 20 | 0–50 |

TABLE 1
Setup summary of 100-node simulations.

| Nodes | Flows | Area (m$^2$) |
|---|---|---|
| 100 | 20 | 1400 × 1400 |
| 150 | 30 | 1700 × 1700 |
| 200 | 40 | 2000 × 2000 |
| 250 | 50 | 2200 × 2200 |
| 300 | 60 | 2450 × 2450 |

TABLE 2
Setup summary of simulations with varied network size.

data point in the graphs was averaged over 40 simulation runs, each with a different seed.

## B Results and Analysis

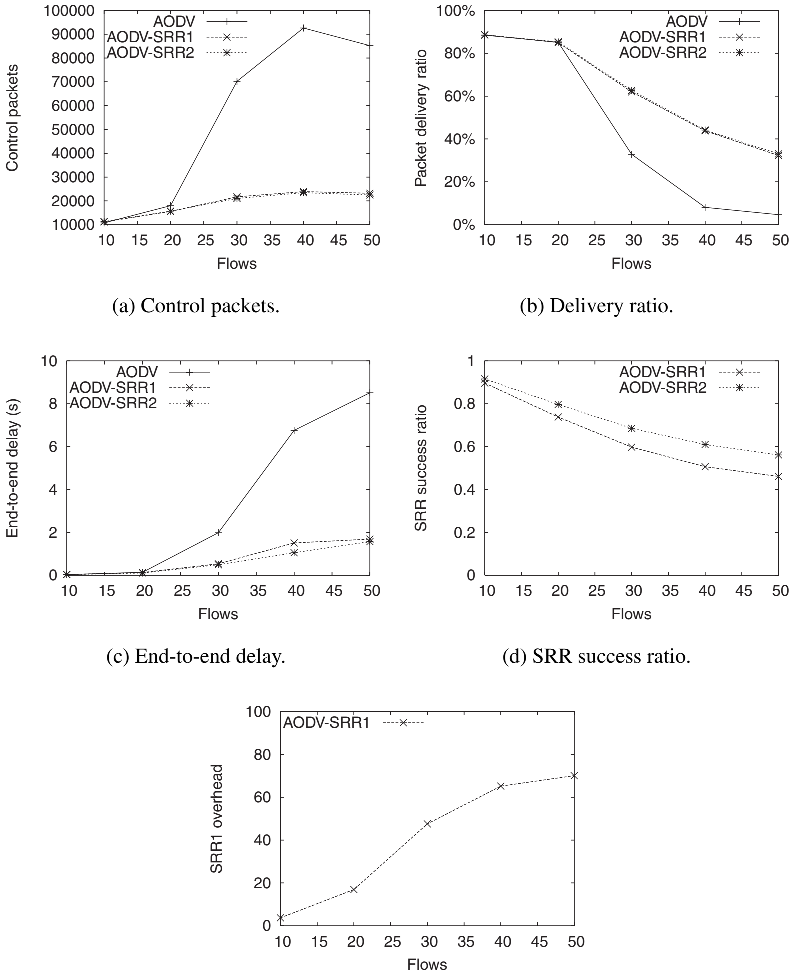
### B.1 Varied CBR Flows

Figures 4(a), 4(b) and 4(c) show the number of control packets per flow, the PDR and the end-to-end delay, respectively, when the number of flows varies from 10 to 50. When the number of flows is less than 20, AODV-SRR1 and AODV-SRR2 do not show much improvement over AODV. However, when the number of flows increases, the improvement becomes significant. For example, when there are 40 flows, compared to AODV, AODV-SRR1 and AODV-SRR2 save the control packets by 74% (Figure 4(a)), improve the PDR from 8% to 44% (Figure 4(b)), and reduce the end-to-end delay by 78% and 84% (Figure 4(c)), respectively. Thus, SRR significantly improves the performance of AODV in all aspects, instead of trading off one for another. The improvement becomes even more significant when considering the simplicity of SRR – in SRR1, a one-hop route discovery salvages an undeliverable RREP packet; in SRR2, nodes utilize duplicate RREQ packets to maintain a backup path to the source and require no extra control packets when salvaging a undeliverable RREP packet. SRR2 has a slightly better performance than SRR1.

The improvement verifies the motivation discussed in Section II. RREP packets in on-demand routing protocols are important and a routing protocol should take special care to prevent them from being lost. SRR may not succeed in every attempt, but as long as it salvages a majority of RREPs, it can save a large number of RREQ transmissions. SRR reduces the end-to-end delay as well because packets spend less time in buffer waiting for route replies. As a result, nodes discard less packets from their buffers due to overflow or timeout and thus the packet delivery ratio is also improved.

When the number of flows increases, the network carries more traffic, and more RREP packets can not reach their intended next hop nodes due to link breakages or channel congestion. Therefore, SRR salvages more RREP packets and improves the performance of AODV more significantly when the number of flows is higher. AODV-SRR2 performs slightly better than AODV-SRR1. One advantage of SRR2 is that it does not use extra control messages. But compared to the overall control overhead caused by the routing protocol, the control overhead saved by SRR2 may be a small proportion. Therefore, the performances of AODV-SRR1 and AODV-SRR2 are close to each other. However, compared to SRR1, in addition to no extra control messages, SRR2 requires fewer changes to the routing protocol and thus simplifies the protocol design. In SRR2, we do not need to design the formats of SREQ and SREP messages, and their respective message handlers.

Figure 4(d) shows the success ratios of AODV-SRR1 and AODV-SRR2, and Figure 4(e) shows the number of SRR control packets (including SREQ,



(a) Control packets.

(b) Delivery ratio.

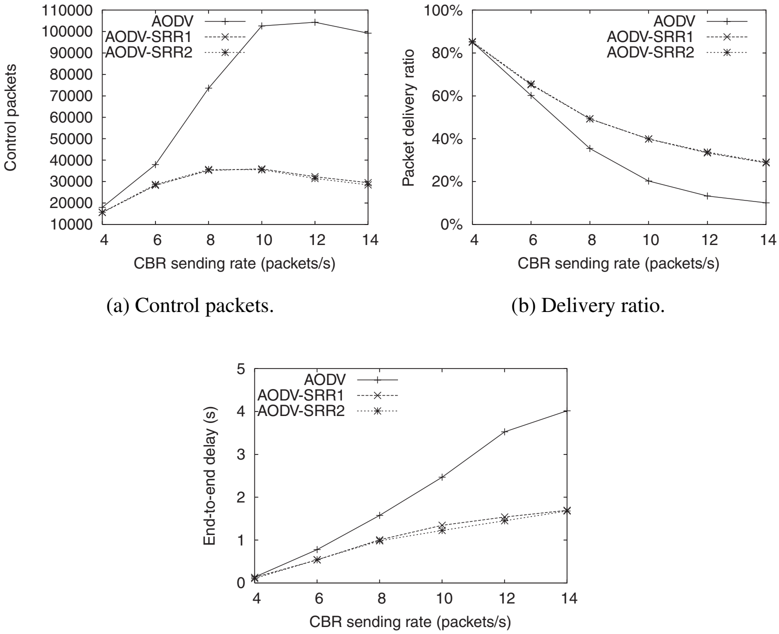
(c) End-to-end delay.

(d) SRR success ratio.

(e) SRR1 overhead.

FIGURE 4
Performance when number of flows varies. Network area 1400 × 1400$m^2$, 100 nodes, speed [0.1–20]$m/s$ and pause time 30$s$.

SREP and RUPD) per flow generated by AODV-SRR1. We observe that around 50% to 90% salvage attempts succeed. AODV-SRR2 has higher success ratio than AODV-SRR1 because it does not incur extra control packets. The success ratios decrease with the number of flows because when the network becomes more congested, even after being salvaged, RREP packets are likely to become undeliverable again. Note that SRR salvages a RREP at most once (Section C). When the number of flow increases, AODV-SRR1 generates more



(a) Control packets.

(b) Delivery ratio.

(c) End-to-end delay.

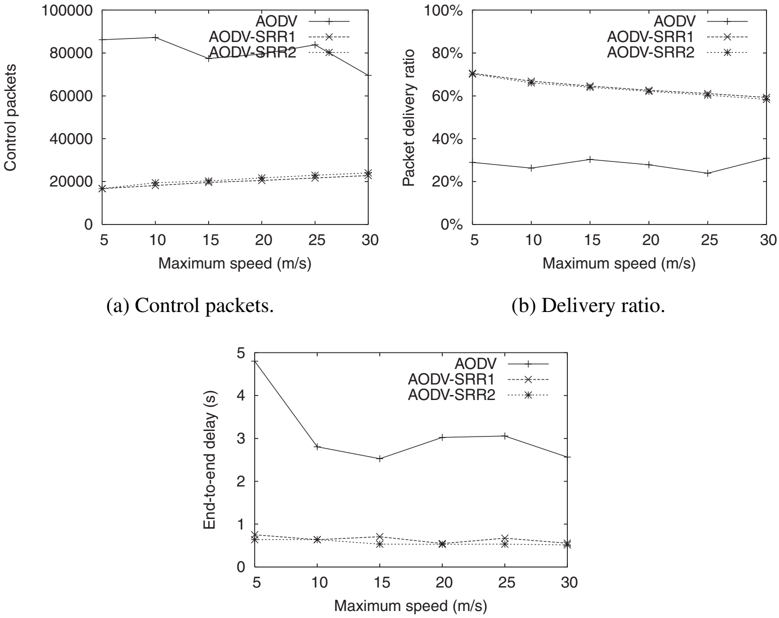
FIGURE 5
Performance when CBR sending rate changes. Network area 1400 × $1400m^2$, 100 nodes, 20 flows, maximum speed $20m/s$, pause time $30s$.

SRR control packets, while AODV-SRR2 does not generate any extra control packets at all. Therefore, SRR scheme two is a noticeable improvement over SRR scheme one.

## B.2 Varied CBR Load

Figures 5(a), 5(b) and 5(c) show the number of control packets per flow, the PDR and the end-to-end delay, respectively, when the CBR sending rate varies from 4 $packets/s$ to 14 $packets/s$. Like previous simulations, AODV-SRR1 and AODV-SRR2 have significantly better results than AODV for all three performance metrics. For example, when data packet sending rate is 10 packets per second, compared to AODV, AODV-SRR1 and AODV-SRR2 save the control overhead by more than 60%, improve the PDR by 97%, and reduce the end-to-end delay by about 45%.

When the CBR sources inject more data packets to the network, the network becomes more congested and more packets can not be forwarded to their intended next hop nodes. Thus, SRR is able to salvage more RREP packets and improve the performance of AODV more significantly.



(a) Control packets.

(b) Delivery ratio.

(c) End-to-end delay.

FIGURE 6
Performance when maximum node speed changes. Network area 1400 × $1400m^2$, 100 nodes, 30 flows, pause time $0s$.

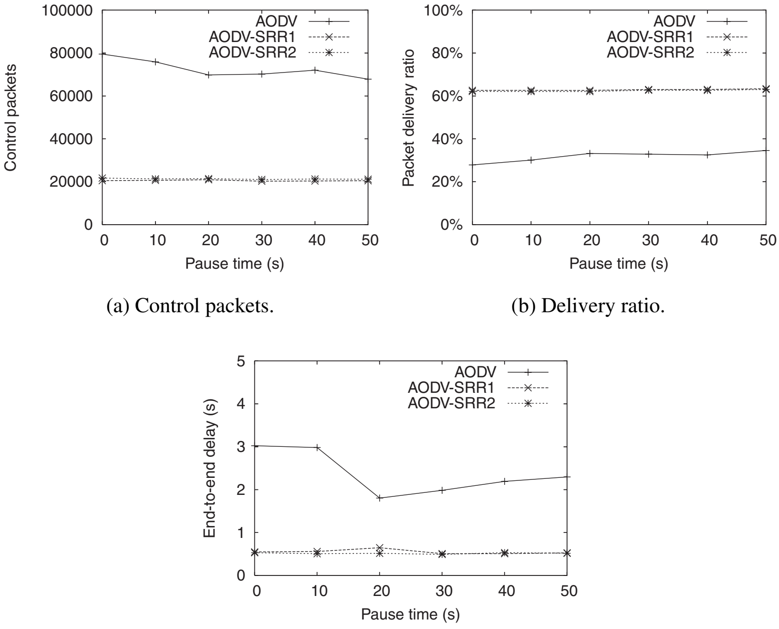
### B.3 Varied Maximum Node Speed

Figures 6(a), 6(b) and 6(c) show the number of control packets per flow, the PDR and the end-to-end delay, respectively, when the maximum node speed varies from $5m/s$ to $30m/s$. The number of flows is 30 and the pause time is $0s$. From the results, we observe that AODV-SRR1 and AODV-SRR2 perform significantly better than AODV in all three performance metrics. For example, when maximum node speed is $20m/s$, AODV-SRR1 and AODV-SRR2 approximately save the number of control packets by 73%, improve the PDR from 28% to 63%, and reduce the end-to-end delay by 82%.

When nodes in the network move faster, links break more often and more packets are not deliverable. Thus, SRR salvages more RREP packets and improves the performance significantly. However, since the network topology changes more dramatically as the node speed increases, the routes established by the salvaged RREPs may break soon. Thus, the performance improvement is more significant at lower speeds.

### B.4 Varied Pause Time

Figures 7(a), 7(b) and 7(c) show the number of control packets per flow, the PDR and the end-to-end delay, respectively, when the node pause time varies



(a) Control packets.

(b) Delivery ratio.

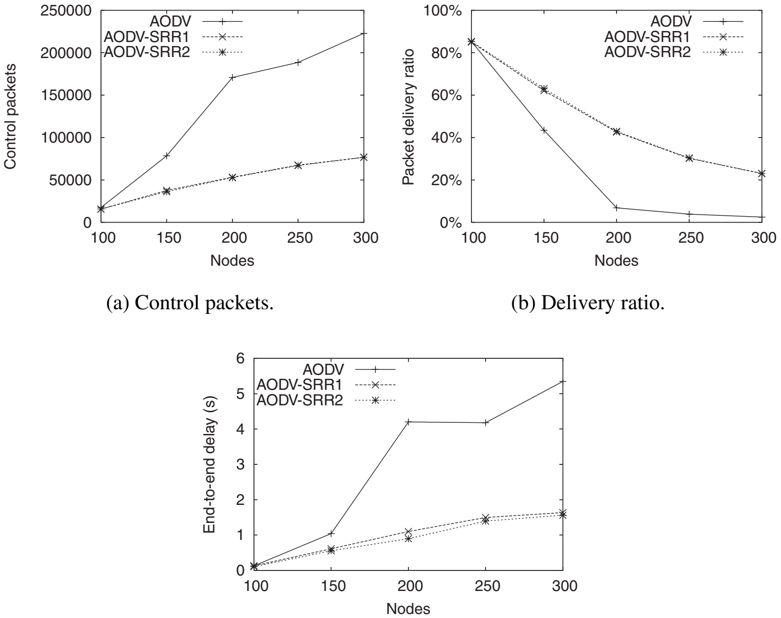
(c) End-to-end delay.

FIGURE 7
Performance when pause time changes. Network area 1400 × $1400m^2$, 100 nodes, 30 flows, maximum speed $20m/s$.

from $0s$ to $50s$. The number of flows is 30 and the maximum node speed is $20m/s$. We observe that AODV-SRR1 and AODV-SRR2 perform considerably better than AODV in all three performance metrics. For example, when the node pause time is $20s$, AODV-SRR1 and AODV-SRR2 approximately save the control packets by 70%, improve the PDR from 30% to 60%, and reduce the end-to-end delay by 60%.

## B.5 Varied Network Size

In this set of simulations, we study the performance of AODV-SRR1, AODV-SRR2 and AODV when the size of networks increases from 100 nodes to 300 nodes. Figures 8(a), 8(b) and 8(c) show the number of control packets per flow, the PDR and the end-to-end delay, respectively. The performance of AODV degrades rapidly with increasing network size. For example, in networks with 200 nodes and 40 flows, AODV delivers only 7% of data packets. AODV-SRR1 and AODV-SRR2 improve the performance significantly. For example, when the network size is 200, AODV-SRR improves the delivery ratio from 7% to 43%, saves the control packets by 69%, and reduces the end-to-end delay by 76%.



(a) Control packets.

(b) Delivery ratio.

(c) End-to-end delay.

FIGURE 8
Performance when network size increases. Constant node density, flows 20% of nodes number, maximum speed $20m/s$, pause time $30s$.

As the network size grows, the route between a source and a destination becomes longer. A RREP packet has to travel farther before reaching the source node. It is more likely that the RREP can not reach its intended next hop. Therefore, AODV-SRR1 and AODV-SRR2 have more chances to salvage undeliverable RREPs and play a more important role in improving the routing performance.

## VI RELATED WORK

Researchers have proposed many approaches to improve the performance of on-demand routing protocols. Some approaches may benefit most protocols (e.g., expanding ring search [19]), while others are suitable for a particular category of protocols (e.g., route cache strategy for DSR [30]). Different approaches may attack different aspects of on-demand routing, like, reducing the overhead of route request messages (e.g., query localization [31]), or making an active route last longer (e.g., local repair [19]).

Expanding ring search (ERS) [19] tries to avoid flooding the entire network when discovering a route and reduces the overhead of RREQ messages. In

ERS, a source node broadcasts a RREQ message within successively larger areas, centered at the source, until the source receives a RREP message. Initially the source uses a small TTL value for a RREQ, for example, 2 hops. If the source has not received a RREP by the end of the discovery time, it broadcasts another RREQ with an incremented TTL value. This process continues until the TTL reaches a threshold value. At this point, the source floods the entire network to find a route. Chang [32] presents a dynamic programming formulation to minimize the expected cost of TTL-based flooding search.

In bilateral route discovery (BRD) [33], both source and destination actively participate in a route discovery process. BRD uses gratuitous route error reporting (GRER) to notify the destination of a broken route. The destination can thus play an active role in the upcoming route re-discovery. Bai and Singhal [34] apply the idea of carpooling to MANETs and propose multiple-target route discovery (MTRD). MTRD aggregates multiple route requests into one RREQ message and discovers multiple targets simultaneously. The MTRD approach improves the routing performance because it reduces the number of regular route discoveries.

Query localization [31] also reduces the overhead of RREQ messages. This approach limits the propagation of a RREQ to the area around the previously known route to the destination node. Therefore, route discovery in query localization does not flood the entire network. One method to perform the query localization is *exploiting node locality*. This method assumes that the destination has not moved too far from its previous location, and hence can be found within a few hops from the most recently used route to it. A RREQ packet contains a counter to control how far intermediate nodes can propagate it from the previous route. When a node receives the RREQ, if the node was on the previous route, it resets the counter to zero and relays the RREQ. Otherwise, the node increments the counter and relays the RREQ if the counter is not greater than a threshold value.

Local repair [19] tries to fix a broken active route locally, and thus extends the lifetime of the route and reduces the number of global route discoveries. Usually, when a link break on an active route happens, the upstream node of the break sends a RERR message to the source. Then the source may start another route discovery. In local repair approach, instead of sending a RERR message immediately, the upstream node first attempts to repair the broken route locally. It does so by broadcasting a RREQ with a limited TTL to discover a path to the destination. If the node receives a RREP message, it can repair the route successfully. Otherwise, it sends a RERR message. Way point routing (WPR) [16] selects a number of nodes on a route as waypoints and divides the route into segments at the waypoints. When an intermediate node moves out or fails, WPR only requires the two waypoints of the broken segment to repair that segment.

Another approach to reduce the overhead of route discovery is using route caches. Hu and Johnson [35] studied the strategies for cache structure, cache capacity and cache timeout. They proposed a link cache structure and several link timeout algorithms. Lou and Fang [36] proposed an adaptive link timeout mechanism that adjusts the link lifetime according to the real link lifetime statistics. Marina and Das [37] proposed three techniques to improve cache correctness in DSR: wider error notification, route expiry mechanism with adaptive timeout selection and negative caches. Yu and Kekem [38] proposed a cache update algorithm to make route caches adapt to topology changes without using ad hoc parameters.

Path optimization [39][40] is based on the following observation: routes are optimal (e.g., number of hops) during the establishment phase, but they may become sub-optimal over time due to node mobility. Path optimizing approaches typically require nodes to work in promiscuous mode to find an optimization opportunity. In SHORT [39], each data packet carries two additional information: *hop_count* (to the destination) and *sending_node* (that transmits the packet). Each node maintains a *comparison array* to collect information from data packets that the node receives or overhears. When a node receives or overhears a data packet, it compares the packet to the information in its comparison array. If an optimization is possible, the node sends a message to notify the node from which the packet is received. PCA [40] improves SHORT by adding another information to the header of each data packet: hop count to the source. Thus when a node receives or overhears a data packet, it attempts to optimize the route in both directions: to the destination and to the source.

Path stability is an important issue for MANETs because a stable path lasts longer and thus reduces the number of route discoveries. A number of routing protocols select paths based on path stability: associativity based routing (ABR) [41], signal stability adaptive routing (SSA) [42], route lifetime assessment based routing (RABR) [43], and flow oriented routing protocol (FORP) [44]. The estimation of path stability either uses past topology changes (e.g., ABR and SSA), or uses prediction of future topology changes (e.g., RABR and FORP). Strategies that use prediction of future topology changes perform better [45]. However, these strategies require nodes to know geographic information such as location, speed and direction.

Han *et al.* [46] study the distribution of path duration. First, they prove that, under a number of mild conditions, when the hop count of a path is large, the distribution of path duration can be approximated by an exponential distribution. Then, they prove that the parameter of the exponential distribution is related to the link durations only through their means and they gave the parameter by the sum of the inverses of the expected link durations. Finally, they propose a scheme for existing routing protocols to select the paths with the largest expected durations.

Routing protocols use failure notification from MAC layer or a HELLO protocol to detect failed links. However, existing routing protocols or MAC layer protocols do not distinguish a link failure caused by mobility between a link failure caused by congestion. If a link failure is caused by congestion, we may solve this problem by letting the transport protocol reduce its rate, instead of letting the routing protocol run an expensive route discovery. Pandey *et al.* [47] propose mobility detection algorithm (MDA), which uses MAC-layer statistics to distinguish between mobility and congestion-based failures. With MDA, routing protocols only react to link failures due to mobility.

In a previous work [48], we focused on the loss of RREP packets and proposed the idea of salvaging route reply. This paper significantly enhances the previous work and the enhancements lie in the following aspects. First, we design a new salvaging scheme, SRR scheme two, which does not incur extra control messages. Second, we consider the issue of routing loops and prove that the route is loop-free after SRR salvages a RREP. Finally, we have added many new insights and discussions.

## VII CONCLUSION AND FUTURE WORK

Route reply (RREP) messages are important in on-demand routing protocols for ad hoc networks. The loss of RREPs causes serious impairment to the routing performance because the cost of a RREP is very high. Typically, an on-demand routing protocol obtains a RREP after flooding the entire or a part of the network with route request (RREQ) messages. This process is expensive and time-consuming - a route discovery requires tens, may be hundreds of RREQ transmissions. If the RREP is lost, a large amount of route discovery effort will be wasted. Furthermore, the source node may have to initiate another round of route discovery to establish a route to the destination.

We proposed the idea of salvaging route reply (SRR), which attempts to salvage an undeliverable RREP in two schemes. In SRR scheme one (SRR1), the salvor runs a one-hop SRR route discovery to find an alternative path to the source. The SRR route discovery succeeds most of the time because neighboring nodes recently propagated the RREQ message originated from the source and learned a reverse path to the source. In SRR scheme two (SRR2), intermediate nodes utilize duplicate RREQ packets and maintain a backup path to the source. When an intermediate node can not relay a RREP message using the original path to the source, it directly switches to the backup path. SRR2 is an improvement to SRR1 because SRR2 requires no extra control message when salvaging an undeliverable RREP.

We presented the implementation of both SRR schemes in AODV. We proved that in the implementation, routes are loop-free after a salvaging. We conducted extensive simulations to evaluate the performance of two SRR schemes in conjunction with AODV. The results show that SRR significantly improves the routing performance in a wide range of system parameter values and in all critical metrics, including packet delivery ratio, control overhead and end-to-end delay.

We are the first research group that focuses on the loss of route reply messages and proposes the idea of salvaging route reply messages to enhance the routing performance in MANETs. Our approach is practical and applicable to all on-demand routing protocols, and does not conflict with existing optimization techniques reviewed in the previous section.

## REFERENCES

[1] Roofnet Project, http://pdos.csail.mit.edu/roofnet/.

[2] NetEquality, http://www.netequality.org/.

[3] Interplanetary Internet Project, http://www.ipnsig.org/.

[4] T. Knott, "Smart surrogates," *BP Frontiers magazine*, April 2004.

[5] E. Royer and C. Toh, "A review of current routing protocols for ad hoc mobile wireless networks," *IEEE Personal Communications*, pp. 46–55, Apr. 1999.

[6] C. E. Perkins and E. M. Royer, "Ad-hoc on-demand distance vector routing," in *WMCSA '99*, Feb 1999.

[7] D. B. Johnson and D. A. Maltz, "Dynamic source routing in ad hoc wireless networks," in *Mobile Computing*. Kluwer Academic Publishers, 1996, vol. 353.

[8] Y.-B. Ko and N. H. Vaidya, "Location-aided routing (lar) in mobile ad hoc networks," in *MobiCom '98*. New York, NY, USA: ACM Press, 1998, pp. 66–75.

[9] R. Jain, A. Puri, and R. Sengupta, "Geographical routing using partial information for wireless ad hoc networks," *IEEE Personal Communication*, pp. 48–57, Feb. 2001.

[10] C. E. Perkins and P. Bhagwat, "Highly dynamic destination-sequenced distance-vector routing (dsdv) for mobile computers," in *SIGCOMM '94*. New York, NY, USA: ACM Press, 1994, pp. 234–244.

[11] G. Pei, M. Gerla, and T.-W. Chen, "Fisheye state routing in mobile ad hoc networks." in *ICDCS Workshop on Wireless Networks and Mobile Computing*, 2000.

[12] Z. J. Haas and M. R. Pearlman, "The zone routing protocol (zrp) for ad-hoc networks," IETF MANET working group, Internet Draft, June 1999.

[13] 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," in *MobiCom*, 1998, pp. 85–97.

[14] P. Johansson, T. Larsson, N. Hedman, B. Mielczarek, and M. Degermark, "Scenario-based performance analysis of routing protocols for mobile ad-hoc networks," in *MobiCom '99*, New York, NY, USA, 1999, pp. 195–206.

[15] J.-M. Choi and Y.-B. Ko, "A performance evaluation for ad hoc routing protocols in realistic military scenarios," in *the 9th International Conference on Cellular and Intelligent Communications (CIC '04)*, 2004.

[16] R. Bai and M. Singhal, "Doa: Dsr over aodv routing for mobile ad hoc networks," *IEEE Transactions on Mobile Computing*, vol. 5, no. 10, pp. 1403–1416, 2006.

[17] V. D. Park and M. S. Corson, "A highly adaptive distributed routing algorithm for mobile wireless networks," in *INFOCOM'97*, vol. 3, Kobe, Japan, April 1997, pp. 1405–1413.

[18] M. K. Marina and S. R. Das, "Routing performance in the presence of unidirectional links in multihop wireless networks," in *MobiHoc '02: Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking & computing*. New York, NY, USA: ACM Press, 2002, pp. 12–23.

[19] C. E. Perkins, E. M. Belding-Royer, and I. D. Chakeres, "Ad hoc on demand distance vector (aodv) routing," IETF Internet draft, Oct. 2003.

[20] P. Jacquet, P. Mühlethaler, T. Clausen, A. Laouiti, A. Qayyum, and L. Viennot, "Optimized link state routing protocol," in *IEEE INMIC'01*, Lahore, Pakistan, December 2001.

[21] E. Duros, W. Dabbous, H. Izumiyama, N. Fujii, and Y. Zhang, “A link-layer tunneling mechanism for unidirectional links,” RFC 3077, Mar. 2001.

[22] IEEE 802.11. Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, Aug. 1999.

[23] L. Bajaj, M. Takai, R. Ahuja, R. Bagrodia, and M. Gerla, “Glomosim: A scalable network simulation environment, Tech. Rep. 990027, 13, 1999.

[24] K. Fall and K. Varadhan, The ns Manual. http://www.isi.edu/nsnam/ns/nsdocumentation.html.

[25] J. Postel, “Internet protocol,” RFC 791, September 1981.

[26] T. Narten, “Assigning experimental and testing numbers considered useful,” RFC 3692, January 2004.

[27] B. Fenner, “Experimental values in ipv4, ipv6, icmpv4, icmpv6, udp, and tcp headers,” RFC 4727, November 2006.

[28] R. Bagrodia, R. Meyer, M. Takai, Y. an Chen, X. Zeng, J. Martin, and H. Y. Song, “Parsec: A parallel simulation environment for complex systems,” *Computer*, vol. 31, no. 10, pp. 77–85, 1998.

[29] T. Rappaport, *Wireless Communications: Principles and Practice*. Prentice Hall PTR, 2001.

[30] D. B. Johnson, D. A. Maltz, and Y.-C. Hu, “The dynamic source routing protocol for mobile ad hoc networks (dsr),” InternetDraft, draft-ietf-manet-dsr-09.txt, Apr. 2003.

[31] R. Castaneda and S. R. Das, “Query localization techniques for on-demand routing protocols in ad hoc networks,” in *MobiCom ’99*. New York, NY, USA: ACM Press, 1999, pp. 186–194.

[32] N. Chang and M. Liu, “Revisiting the ttl-based controlled flooding search: Optimality and randomization,” in *MobiCom ’04*. ACM Press, 2004, pp. 85–99.

[33] R. Bai and M. Singhal, “Brd: Bilateral route discovery in mobile ad hoc networks,” in *IFIP Networking 2007: the sixth International Conferences on Networking*, Atlanta, Georgia, May 2007.

[34] R. Bai and M. Singhal, “Carpooling in mobile ad hoc networks: the case of multiple-target route discovery,” in *WiOpt 2007: 5th Intl. Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks*, Limassol, Cyprus, April 2007.

[35] Y.-C. Hu and D. B. Johnson, “Caching strategies in on-demand routing protocols for wireless ad hoc networks,” in *MobiCom ’00*. ACM Press, 2000, pp. 231–242.

[36] W. Lou and Y. Fang, “Predictive caching strategy for on-demand routing protocols in wireless ad hoc networks,” *Wireless Networks*, vol. 8, no. 6, pp. 671–679, 2002.

[37] M. K. Marina and S. R. Das, “Performance of route caching strategies in dynamic source routing,” in *ICDCSW ’01*. IEEE Computer Society, 2001, p. 425.

[38] X. Yu and Z. M. Kedem, “A distributed adaptive cache update algorithm for the dynamic source routing protocol,” in *INFOCOM’05*, Miami, FL, USA, March 2005.

[39] C. Gui and P. Mohapatra, “Short: Self-healing and optimizing routing techniques for mobile ad hoc networks,” in *MobiHoc ’03*. New York, NY, USA: ACM Press, 2003, pp. 279–290.

[40] V. C. Giruka, M. Singhal, and S. P. Yarravarapu, “A path compression technique for on-demand ad-hoc routing protocols,” in *IEEE MASS ’04*.

[41] C.-K. Toh, “A novel distributed routing protocol to support ad hoc mobile computing,” in *IEEE 15th IPCCC*. Phoenix, AZ, USA: IEEE, March 1996, pp. 480–486.

[42] R. Dube, C. D. Rais, K.-Y. Wang, and S. K. Tripathi, “Signal stability-based adaptive routing (ssa) for ad hoc mobile networks,” in *IEEE Personal Communications Magazine*. IEEE, February 1997, pp. 36–45.

[43] S. Agarwal, A. Ahuja, J. Singh, and R. Shorey, “Route-lifetime assessment based routing (rabr) protocol for mobilead-hoc networks,” in *IEEE ICC*, June 2000.

[44] W. Su, S.-J. Lee, and M. Gerla, “Mobility prediction and routing in ad hoc wireless networks,” *Int. J. Netw. Manag.*, vol. 11, no. 1, pp. 3–30, 2001.

[45] N. Meghanathan, "Comparison of stable path selection strategies for mobile ad hoc networks," in *ICN/ICONS/MCL*, 2006.

[46] Y. Han, R. J. La, and H. Zhang, "Path selection in mobile ad-hoc networks and distribution of path duration," in *IEEE Infocom*, 2006.

[47] M. Pandey, R. Pack, L. Wang, Q. Duan, and D. Zappala, "To repair or not to repair: Helping routing protocols to distinguish mobility from congestion," in *IEEE Infocom MiniSymposia' 2007*, Anchorage, AK, USA, May 2007.

[48] R. Bai and M. Singhal, "Salvaging route reply for on-demand routing protocols in mobile ad-hoc networks," in *MSWiM 2005: Proceedings of The 8th ACM/IEEE International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, October 2005.