Pro-Reactive Route Recovery and Automatic Route Shortening in Wireless Ad Hoc Networks

無線アドホックネットワークにおける
プロ・リアクティブ型ルート回復と自動ルート短縮

by

Zilu LIANG

Supervisor: Yasushi WAKAHARA (Professor)

SUBMITTED TO THE DEPARTMENT OF ELECTRICAL ENGINEERING IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF ENGINEERING IN ELECTRICAL ENGINEERING
AT
THE UNIVERSITY OF TOKYO

August, 2011

To my beloved father in Heaven.

ABSTRACT

This thesis mainly addresses two problems in wireless ad hoc routing: route failure and route redundancy. Route failures occur frequently in ad hoc networks due to their highly dynamic topology as well as their unstable wireless communication medium. Existing route recovery methods lead to long latency and large control overhead. We proposed a novel relay recovery route maintenance protocol RELREC for ad hoc networks to combine the benefits of both proactive and reactive routing protocols and to minimize their drawbacks. In our proposal, one or more substitute routes usually become ready for the recovery of every link in a route before its break, while the route recovery process actually starts only when the upstream node of a link confirms that the link really breaks. Since this scheme does not broadcast any control packet, it can effectively recover a broken link without heavy control overhead traffic. Also, it helps reduce the time delay due to the recovery, since substitute routes are already available when the upstream node initiates the route recovery process. However, the route recovered by RELREC is usually longer than the original one. In fact, route redundancy occurs frequently in ad hoc networks due to highly dynamic topologies. In order to improve the integrated performance of an ad hoc network, we proposed two automatic route shortening strategies to optimize the route length in the network: relay route shortening and active route shortening. While relay shortening can only be initiated by the Relay Nodes in pro-reactive relay recovery processes, active shortening can be launched by any node whenever it overhears a shorter route from its neighbors. Note that the information in Relay Tables is only utilized in route recovery process in RELREC. In order to utilize the Relay Table information to the possible extent to further reduce the control overhead, we extended the usage of Relay Table to route discovery process by incorporating Relay Table information into gossip algorithm, and proposed Relay Gossip Routing (RGR). In RGR, only Relay Nodes are allowed to rebroadcast under a gossiping probability, which differs from existing gossip methods.

The simulation study in ns-2 simulator has demonstrated clearly that our proposed RELREC scheme effectively reduces the average end-to-end time delay by up to 49% and 31%, and the normalized control overhead by up to 17% and 28% compared with the previous algorithms AODV and DRRS respectively in the best cases. According to the simulation results, the data delivery ratio by our proposed scheme is close to DRRS and AODV. The simulation results confirm that our proposal can provide quick and low-cost route recovery without degrading the packet delivery ratio, while it is adaptive to node mobility and traffic load. When it comes to the automatic route shortening algorithms, active shortening works more effectively than relay shortening does, leading to by far the shortest average route length and the smallest end-to-end time delay among all the algorithms in the simulation. However, the slightly larger control overhead traffic and relatively lower packet delivery ratio harms the performance of active route shortening.

In the evaluation of RGR, in order to find the appropriate gossiping value for the Relay Nodes, we firstly studied the relation between packet delivery ratio and gossiping probability *p*. It is shown that the packet delivery ratio equals or is even higher than that of $p = 1$ when the value of p is larger than a certain point lying between 0.35 and 0.45; at the same time the total control overhead is smaller than that of $p = 1$. Accordingly, the gossiping probability is set to 0.5 in the performance evaluation of RGR. Simulation results confirm that RGR successfully reduces the normalized control overhead by up to 17% in the best case compared to RELREC, and ensures a similar performance to RELREC in terms of the packet delivery ratio. Although the basic gossip routing yields the lowest normalized control overhead, it yields the worst performance in terms of the packet delivery ratio. It is conspicuous that RGR strikes a better balance between control overhead and packet delivery ratio than basic gossip routing does. Besides, RGR also outperforms basic gossip routing in terms of average time-to-time end delay and average route length.

ACKNOWLEDGMENT

The research described in this thesis was financially supported by the Global 30 scholarship from the Japanese Ministry of Education, Culture, Sports, Science and Technology from October 2009 through September 2010, and by the Sano Chikara Scholarship from College Women's Association of Japan from April 2011 to September 2010.

I am greatly indebted to my thesis advisor, Professor Yasushi Wakahara, for the continuous support of my master study and research, for his instructions and guidance on how to become a good researcher, for his patience, kindness, enthusiasm, expertise and immense knowledge. He always pushes me to think deep, and to look beyond the surface. Second, I would like to express my heartfelt gratitude to Associate Professor Takefumi Ogawa and Assistant Professor Yuzo Taenaka, for sparing their time to join the weekly discussion on my research progress and providing their valuable suggestions, comments and technical help. My sincere thanks also go to Associate Professor Nakayama and Assistant Professor Miyamoto, for their encouragement, insightful comments and inspiring questions in the lab seminars. I would like to thank Mr. Ni Wei, Mr. Chen Jian, Mr. Yuki Kurebayashi, Mr. Eid Mohamad Samir for their general suggestions on doing research, and I appreciate the technique discussions with Mr. Wenjun Xia, Mr. Miquel Espi and Mr. Raymond Ng. I also want to thank our lab secretaries Ms. Yoshizawa and Ms. Kawasaki, and MEM secretaries Ms. Umeda and Ms. Minegishi for all their help and assistance. I want to thank my lab-mates for creating a pleasant atmosphere in the lab.

Above all, I thank my beloved father, who has passed away on June 18th, 2010, for his selfless love and devoted family education in the first two decades of my life. He encouraged me, believed in my capability, and provided me strategic suggestions in the pivotal time of my life. My father has always been encouraging me to be creative, unique, self-motivated, independent, and willing to face the challenge in my study and life. I have been benefited from the study methodology, thinking style, and living philosophy that shaped under his immense influence, and I believe I will be further benefited from the treasure that my father has granted to me in the future of my life. I thank my mum for her boundless love, for her optimism, for encouraging me to follow my voice from within instead of blind obedience to the mass. Finally, I would like to thank my fiancé, Chapa Martell Mario, for his love, his encouragement, his tolerance, and for sharing his knowledge and expertise in helping me finishing this thesis.

CONTENTS

FIGURE LIST

TABLE LIST

CHAPTER 1

INTRODUCTION

1.1 Problem Formulation

Continuous advances in wireless communication technology coupled with the recent proliferation of portable smart electronic devices have led development efforts for future wireless ad hoc networks, where a group of mobile nodes use radio frequency transceivers to communicate with each other without the support of any centralized administration or established infrastructure. Figure 1 depicts a traditional cellular network whose operation is heavily dependent on the established base stations (left) and an ad hoc network that is completely managed in a distributed manner. The idea of mobile ad-hoc or packet radio networks has been under development since 1970s. Since the mid-90s, when the definition of standards such as IEEE802.11 helped cause commercial wireless technology to emerge, mobile ad-hoc networking has been identified as a challenging evolution in wireless technology. The ad hoc networking technology has been widely used in military field (e.g. DARPA [1, 2]), disaster rescue (e.g. DUMBONET [3]), and in our daily life to provide a ubiquitous communication environment.

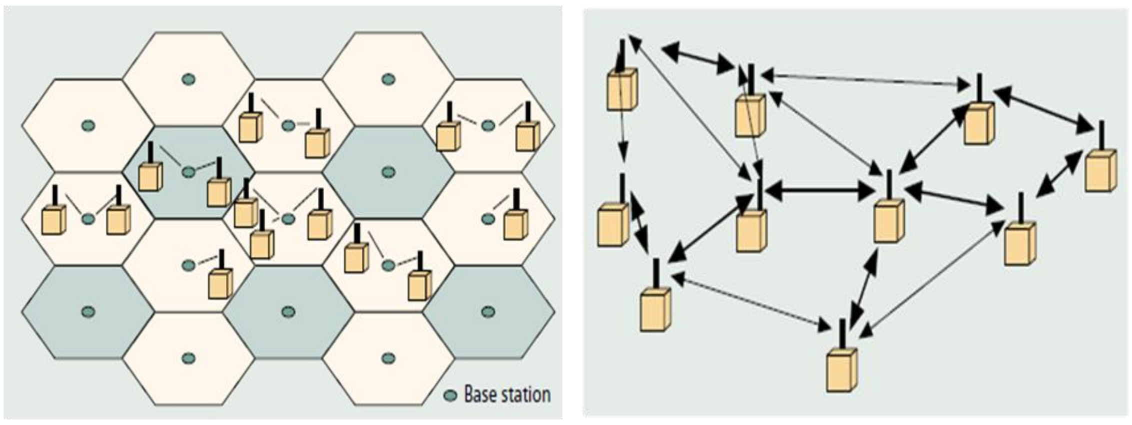


Figure 1. Cellular network based on established infrastructure (left) and distributed ad hoc network (right)

Routing is the process of finding a path over which the packet will be sent across a network from a source to a destination. A routing protocol serves to exchange the route information, find and select a feasible path to a certain destination, gather information of path breaks and repair the broken paths. Due to the high mobility of nodes, together with other reasons such as congestion and power failure, route failures occur frequently in an ad hoc network generally. For example, the rescue teams may move randomly in the disastrous area to search for survivors after disasters, or the conference attendants may walk from time to time in a building, both of which causes route failure in the wireless ad hoc networks, as exemplified in Figure 2 (a). Routing protocols should be able to handle the route failures in a timely manner.

Existing route recovery methods provide recoveries by broadcasting control packets either globally or locally to find a new route, no matter whether it is proactive route recovery [4-6] or reactive route recovery [7-21]. In global broadcasting, the source node of the broken route starts a route discovery by flooding Route Request (RREQ) messages across the whole network to find a new route. This method causes long time delay and heavy control packet traffic in the route recovery process and hence leads to significant degradation of communication performance. In order to reduce time delay and control overhead traffic, the broadcasting of RREQ may be limited in a local area of the broken link. However, due to the inherent characteristic of the broadcasting scheme, the performance of local recovery is still far from satisfying. Heavy traffic caused by control packet broadcasting often increases the time delay and leads to significant degradation of communication performance. A new recovery method that achieves elimination of both control packets and recovery time delay is necessary to avoid this problem.

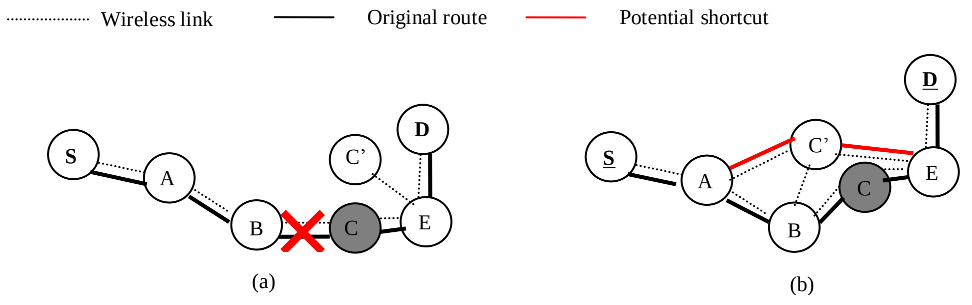


Figure 2. Problems of interest. (a) Route failure; (b) Route redundancy.

On the other hand, route redundancy may occur in ad hoc networks due to route recovery and node mobility, as exemplified in Figure 2 (b). It has been observed that the TCP performance decreases as the length of the routes increases in ad hoc networks [22-24]. In extreme cases, the redundant route may even form a loop and thus significantly degrade the performance of the network. Automatic route shortening is needed to remove the route redundancy and optimize the route length in ad hoc networks, which is important in maintaining the performance from a cross-layer point of view.

This research is devoted to address the problems related to route failure and route redundancy in wireless ad hoc networks by proposing a novel route recovery algorithm and two automatic route shortening schemes.

1.2 Objectives

The main aim of this research is to develop algorithms to provide fast and low-cost route recovery, as well as optimize the route length in wireless ad hoc networks. This research integratedly achieves the following objectives:

- Fast reaction to route failure.
- Instantaneous route recovery.
- No broadcasting of control packet.
- Low packet loss.
- No false detection.
- Optimized route length.

The proposed algorithms are supposed to work in the scenarios where all the users in an ad hoc network move randomly in a certain area. The potential applications include the ad hoc networks used in the disaster rescue after earthquake or tsunami, robot ad hoc networks (RANET), temporarily deployed ad hoc network in a building, to name but a few. The random waypoint model is chosen to simulate the scenarios mentioned above. At the same time, the proposed RERLEC is supposed to work more efficiently in the vehicle-to-vehicle ad hoc networks (VANET) by taking advantage of the more limited moving pattern of vehicles, though simulations based on this model are not studied in this thesis.

1.3 Thesis Outline

The rest of the thesis is organized as follows. Chapter 2 introduces the related works on route recovery in wireless ad hoc networks, the automatic route shortening, and the gossip routing. In Chapter 3 the principles of RELREC and the two automatic route shortening algorithms are presented, and the simulation study is conducted to evaluate the performance of the proposals. In Chapter 4 the RELREC is incorporated into gossip routing to extend the usage of route recovery information in route discovery stage. Conclusions and future works are presented in Chapter 5. Appendix A and Appendix B give a preliminary knowledge on wireless ad hoc networks and the implementation of RELREC in NS-2 simulator respectively.

CHAPTER 2
RELATED WORKS

2.1 Route Recovery in Ad Hoc Networks

In accordance with the routing protocols, existing route recovery methods in wireless ad hoc networks can be divided into two categories: proactive route recovery and reactive route recovery.

2.1.1 Proactive route recovery

Proactive route recovery attempts to initiate the route recovery procedure before the route really breaks [4-6]. The preventive mechanism is realized by judging the received signal power of packets [6]. If the received signal power is lower than a certain threshold, a warning message is sent to the source node indicating the likelihood of the route break. The source can then start route discovery earlier than the actual break, thereby potentially avoiding the effect caused by the route break. However, the problem of this method lies in the detection of received power which is influenced by various factors such as fading, multipath effects and similar random transient phenomena, leading to the so-called false detection problem and thus bringing unnecessary control packet broadcast in the network. Other two examples of the proactive route recovery are the Refinement-Based Route maintenance protocol (RBR) [5] and the Dynamic Link Breaking Avoidance (DLBA) [4]. These two protocols take advantage of the promiscuously overheard information and start a route recovery with pre-emptive local RREQ broadcasting. In this way, the performance can be improved to a certain extent compared to basic pre-emptive methods. However, large control overhead is unavoidable when the upstream node repairs the broken route by sending RREQ packets in a broadcasting manner, which may further increase the average end-to-end time delay. Therefore, the efficiency of RBR and DLBA is still limited by the broadcasting scheme in their route discovery process. Proactive methods are often criticized as power inefficient, since they exchange messages even when there is no traffic. Another typical problem for this category is that the stale route information in the redundant table may lead to false recovery, thus degrading the performance of the network.

2.1.2 Reactive route recovery

Apart from the proactive route recovery, a vast proportion of the route recovery methods belong to the reactive category; that is, they trigger the route recovery after the route really breaks. Based on the broadcast range of control packets, the reactive route recovery can be further divided into global recovery and local recovery.

Global Recovery

In global recovery, the upstream node of broken link generates and broadcasts a Route Error (RERR) message. The RERR will be propagated by the intermediate nodes to the source node of the active route. Upon receiving the RERR, the source node initiates a new route recovery process to find a substitute route. Examples of global recovery include AODV [21], AODV-BR [19] and DSR [12], and all local recovery protocols also use global recovery as the supplement in case local recovery fails.

Local Recovery

In local recovery, the upstream node of the broken link is allowed to recover the broken route via launching route discovery processes in a local area (broadcasting RREQ to k-hop neighborhood). The works in literature are presented according to the value of k as follows.

- Limited to 1-hop neighbor: Quick Local Repair (QLR) [13].
- Limited to 2-hop neighbor: Proximity Approach To Connection Healing algorithm (PATCH) [16], Localized Route Repair (LRR) [17].
- Limited to k-hop neighour (2 <k < network diameter): redundancy based route methods [7, 8, 14], AODV, neighbor aware source routing [10], height-based recovery [11], controlled flooding [15], Witness-Aided Recovery (WAR) [20].

The problem of reactive route recovery is the unavoidable time delay caused by the reactive route discovery procedure and the large control overhead resulted by the broadcasting scheme.

Few proposals that do not use broadcasting in route recovery have already been proposed. A packet retransmission scheme named DRRS and proposed in [10] also uses neighbour nodes of a potential broken route to improve data delivery ratio. However, instead of repairing the broken route, this method only focuses on data retransmission; that is, the relevant nodes on the main route do not update their Route Tables with a new stable route, rather they continue transmitting packets using the bypassed route. The retransmission at neighbour nodes serves to be a compensation for the data delivery ratio. Furthermore, DRRS requires HELLO messages exchanged by all the nodes at a predetermined interval and more than one neighbour node may retransmit the same packet to the downstream node of the broken link, which causes unnecessary extra control overhead. In Implicit Backup Routing-AODV (IBR-AODV) [9], the backup nodes are established by overhearing the transmitted data of the subsequent three nodes that are in the main route. If a backup node detects the retransmission from a node in the main route, it waits for a period of time selected by a random back-off algorithm. If the backup node does not overhear acknowledgement during the back-off time, it should send the Route Change (RC) packet to the node that tries retransmission. The node updates its routing table and sends the acknowledgement to backup node once it receives the RC. The backup node that receives the acknowledgement also updates its Route Table. A back-off algorithm with Contention Window size in reverse proportion to the number of overhearing is used to settle the competition of multiple backup nodes. The backup nodes randomly select a number in CW and wait for the back-off time. If a backup node listens to the RC transmitted by another backup node, then it cancels its transmission. This method does not fully utilize the neighbour nodes, limiting the potential recovery opportunity.

2.2 Automatic Route Shortening in Ad Hoc Networks

Although ad hoc routing has triggered wide research interest and hundreds of thousands of ad hoc routing protocols have been proposed, the research work on route shortening in ad hoc networks is somewhat limited.

The famous on-demand source routing protocol DSR [12] did a pioneer work in optimizing the routes through automatic route shortening. In DSR, all nodes in the network obtain the global topology information using aggressive caching and source routing headers. By keeping the addresses of all the nodes on a particular route in the routing header, it is very easy to discover the route redundancy by checking whether any node appears more than once in the route. However, this method has the limitation that it only works for source routing protocols. Due to the dynamic changes of network topology, source routing is not a reasonable solution in some ad hoc networks. A more general method called proximity-based dynamic path shortening is proposed in [25]. In this scheme, a proximity area is defined for each node based on the received SNR. When a node moves into the proximity area of its one-hop previous node on a route, the node will try to communicate directly its two-hop previous node. If such direct communication succeeds, the node will initiate a route shortening process. The efficiency and correctness of this method depends greatly on the range of the proximity area, and the aggressive control packet exchange may introduce unnecessary control overhead. Simulation results indicate that the proximity based method does not work well for distance vector routing protocols like AODV [21].

Recently a couple of route shortening algorithms based on promiscuous overhearing have been proposed [4, 5, 26, 27]. In SHORT [27] each node keeps a hop comparison array, and the hop count in the header of an overheard or received packet is compared with the corresponding entry in the hop comparison array. A route shortening process is initiated if the comparison result is more than two. This method does not work at source or destination nodes, and it does not work for one-hop shortening. The active probe route redirection (A-PR$_2$) [5] realized the route shortening based on the information in Overhear Table used in local route recovery. If the information in the Overhear Table indicates that two nodes are on the same route and their distance are larger than three hops, a route shortening execution will be triggered. The potential stale information in the Overhear Table may fail to indicate the current network topology, thus leading to incorrect route shortening. Besides, no simulation study is conducted in the paper to verify the performance of the A-PR$_2$. In the path pruning shortening [26], if a node overhears a packet that it once forwarded, it can tell that a shortcut exists. This method is proposed for sensor networks so that it only considers the first transmitted packet, and therefore it is not adaptive to topology change. Besides, it allows extra retransmission, which may account for the improvement of delivery rate. The dynamic path shortening (DPS) [4] is very similar to SHORT. It includes the information of the minimum hop count for this route in the network and the address of the node which has the minimum hop count so that the DPS works for source and destination nodes. However, DPS will introduce extra control overhead in the network.

2.3 Gossip Routing in Ad Hoc Networks

The rational for gossip routing scheme comes from its key characteristic of bimodal behavior which is well known in the percolation theory [28, 29]. Suppose a node has a probability of p to forward a message to its neighbors, then there is a threshold value p_0 such that, in almost all executions in sufficiently large random networks, either almost no node receives the message when $p<p_0$ or almost all nodes receive the message when $p>p_0$. Therefore it is possible for us to set the value of p larger than p_0 so that routing messages propagated can be reduced while still almost all nodes in the network receive the message.

A couple of gossip routing protocols have been proposed in literature. The effort on using gossiping to alleviate broadcast storm problem stretched back to [30] where probabilistic scheme is mentioned as one solution to broadcast storm problem, but no detail is provided in that paper. Z. J. Haas et al. [19] conducted a comprehensive study on the possible heuristics to improve the basic gossiping scheme, including introducing threshold, zones and retry mechanism. A couple of gossiping algorithms set adaptive forwarding probability according to local neighbor density of the forwarding node [31-35]. The problem of this kind of approach is that the counter may not reflect the real node density since some neighbors may suppress their rebroadcasts according to their rebroadcast probability. Besides, the update in forwarding probability may lag behind the topology change. GPS information is used in [34] to localize the gossiping within the ellipse centered at the source and destination. Since we cannot take the GPS service for granted, this method may fail where GPS information is not available.

CHAPTER 3
PRO-REACTIVE RELAY RECOVERY (RELREC) AND AUTOMATIC ROUTE SHORTENING

3.1 Assumptions

Before describing the proposal in detail, we present several assumptions here.

- The antenna is non-directional and the communication channel is bi-directional.
- IEEE 802.11 [42] without RTS/CTS is used as the MAC layer protocol.
- A node receiving a corrupted packet can detect the error, e.g. standard link-layer checksum or Cyclic Redundancy Check (CRC), and discard the packet.
- The network scale is moderate with proper node density, so that there are enough neighbors near the active main routes to ensure the validity of the route recovery procedure.
- The speed with which nodes move is moderate compared with the packet transmission latency and wireless transmission range of the particular underlying network hardware in use.
- All nodes wishing to communicate with other nodes within the ad hoc network are willing to participate fully in the proposed method.
- There is no malicious node in the network and all nodes can trust each other; we do not take security issue into consideration at this stage.

3.2 Pro-Reactive Relay Recovery (RELREC)

3.2.1 Overview of RELREC

In our proposal, we firstly define a Relay Node as a common neighbor of both the upstream node and the downstream node of a link. While all the Relay Nodes participate in an ad hoc network, they keep on overhearing transmission at the upstream and downstream nodes. Since the IEEE 802.11 based MAC protocol conducts frame retransmission to ensure packet delivery over an unstable wireless link, a Relay Node can detect unstableness of the wireless link by overhearing retransmissions. Hence, when a Relay Node overhears the retransmission at the upstream node, which indicates the possibility that the link may have broken, the Relay Node sends a NOTICE packet to the upstream node to inform that the likelihood of a disconnection is recognized by the Relay Node and that the Relay Node can bypass the packet from the upstream node to the downstream node. This is the proactive character of our proposal; that is, to prepare an alternative route before the route really breaks. Even after receiving the NOTICE packet, the upstream node continues the retransmission until it reaches the maximum number of retransmissions (7 times).

When the upstream node confirms that the link has really broken, the upstream node sends a CONFIRM packet back to one of the Relay Nodes from which the upstream node has received NOTICE packets and uses the Relay Node as the new downstream node. This is the reactive character of our proposal; that is, the route recovery starts only after the route has really broken. The pro-reactive character without broadcasting control packet of our proposed method ensures shorter average time delay and smaller control overhead, avoiding the false detection. This proposed pro-active route recovery method is called RELREC (RELay RECovery) hereinafter.

3.2.2 Relay Node and Relay Table

In the pro-reactive route recovery scheme, a Relay Node is the common neighbor of the upstream node and downstream node of a link. Nodes that satisfy the definition of Relay Node can be used to effectively recover a broken link via relaying the packet from the upstream node to the downstream node. According to the above mentioned definition, a Relay Node can overhear the packet transmission at both the upstream and downstream nodes. To act as a Relay Node, each node uses a Relay Table to keep information of the links based on overheard packets. An entry in the Relay Table consists of the following four fields:

- Upstream node of the concerned link. This field indicates the sender of the overheard packet.
- Downstream node of the concerned link. This field indicates the receiver of the overheard packet.
- Source and destination nodes. This field records an end-to-end node pair to identify a route, which will be utilized to optimize a route in route shortening mechanism.
- Expiry time. This filed helps to update the Relay Table periodically and remove stale information.

A new entry should be added into the Relay Table in the following situation. When link <B, C> is under good condition, a data packet will be transmitted from B to C and then from C to E. If it is the first time that R overhears the successive transmission of a data packet at B and C separately, R adds an entry for link <B, C> in its Relay Table, indicating that it can serve as a Relay Node for link <B, C>. Each entry will be removed in the following situations:

- Removed immediately after being used by the Relay Node.
- Expires after a fixed time span. In order to remove the stale information in time, the timeout of each entry in the Relay Table should be set properly. We set the *expire time* of a relay entry the same as that of a route entry in the Route Table; that is, Relay_Timeout = Route_Timeout. In the simulation, the lifetime of a route is either determined from RREP or initialized to ACTIVE_ROUTE_TIME_OUT (=3 seconds in NS2-2.34).

Taken as an example, a Relay Node R and its Relay Table are shown in Figure 3. Suppose route {S, A, B, C, E, D} is the concerned route in the network. Node R is a potential Relay Node for link <B, C>. When link <B, C> is under good condition, a data packet will be transmitted from B to C and then from C to E. If it is the first time that R overhears the successive transmission of a data packet at B and C separately, R adds an entry for link <B, C> in its Relay

Table, indicating that it can serve as a relay node for link <B, C>. Each entry will be removed in the following two situations: expired after a fixed time span or removed immediately after it is used.

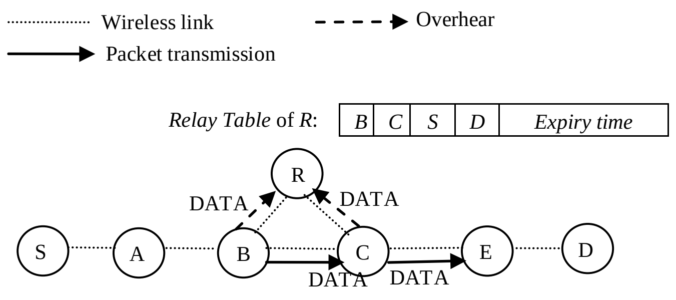


Figure 3. Relay Node *R* for link <*A*, *B*> and its Relay Table.

3.2.3 Design of RELREC

Using the same scenario in Figure 3 where {S, A, B, C, E, D} is the concerned route in the network. When R overhears the data transmission from B to C, R searches its Relay Table for the entry of link <B, C>. In this case, R has an entry for <B, C> in its Relay Table, so that R caches the overheard packet for a short time in case it is needed for retransmission due to the failure of the link. If B sends the DATA packet but receives no acknowledgement from C on MAC layer, which means link <B, C> may have broken. B will retransmit the data packet to C. When overhearing transmission of the same packet at B, R searches its Relay Table for the entry for the link. In this case, R has an entry for link <B, C> in its Relay Table. Then R sends a NOTICE packet to B to inform its potential relay function, as is shown in Figure 4. In this way, an alternative path becomes proactively available before any node confirms the route break. This is the proactive characteristic of RELREC.

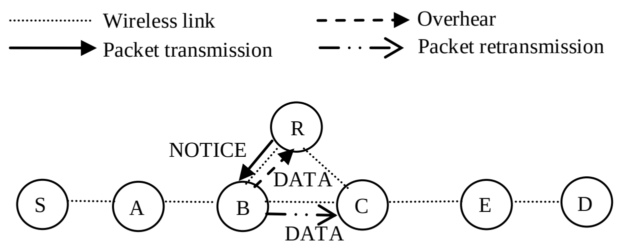


Figure 4. Relay Node *R* sending a *NOTICE* when overhearing DATA retransmission

Upon receiving the NOTICE packet, B buffers the NOTICE and keeps on retransmitting the data packet. Any packet at the MAC layer is retransmitted at most max_retransmission number of retry if an acknowledgement is not received. Providing that C receives the retransmitted data and returns an acknowledgement to B, B just discards the NOTICE received from R without taking any further action. However, if B does not receive an acknowledgement on MAC layer until it reaches the maximum number of retransmission, B believes the link has really broken and searches its buffer for NOTICE packets received from potential Relay Nodes. In this example, B has received the NOTICE from R in practice, so that B returns a CONFIRM packet to R to accept the relay assistance. At the same time, B replaces C with R as its next hop and the recovered route becomes {S-A-B-R-C-E-D}. R retransmits the DATA to C when it receives CONFIRM packet from B and then updates its Route Table. In this way, Relay Node R reactively repairs the route by relaying the packets from B to C. This is the reactive characteristic of RELREC. This process is shown in Figure 5.

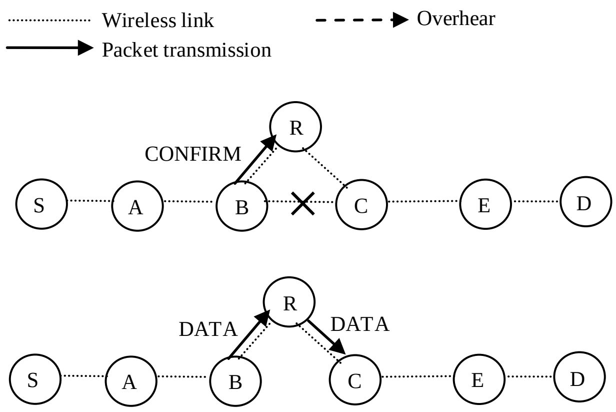


Figure 5. Broken link recovered by Relay Node *R*.

The pseudo-code of RELREC is shown below.

```
Node R overhears a data packet p
{
        extract the following information from the packet header:
        <addr_src, addr_dst, addr_upstrm, addr_dnstrm, pkt_id >;

        if (p has been overheard before)
        {
                if (the entry for <addr_upstrm, addr_dnstrm> exists in Relay Table)
                        send NOTICE to addr_upstrm;
        }
         else if (another p' with <addr_src, addr_dst, addr_upupstrm, addr_upstrm,
        pkt_id> has been heard before)
        {
                add an entry for link <addr_upupstrm, addr_upstrm> in R's Relay Table;
                delete  p' from the cache of overheard packets;
        } else
                insert p into the cache of overheard packets;
}
```

3.2.4 Analysis of Correctness

Existing proactive route recovery methods have two major problems: false detection and false recovery. The former leads to unnecessary route recovery and the latter leads to route recovery failure. False detection refers to the situation where the concerned link is regarded as broken but in fact it has not broken. For signal power based proactive methods; false detection problem is closely related to the inaccuracy and unreliability of signal power detection. In real world, the received power is affected by various factors such as fading, multipath effects and similar random transient phenomena. Therefore the strength of the received power may not accurately indicate the link state. False detection happens when the received signal is in fact dampened below the detection threshold by interference factors even if the link does not break. Unecessary route recovery is started as a result, and the network resource is wasted unnecessarily. False recovery problem exist in redundancy based methods where backup route information is kept in a redundant table. The false recovery happens when the stale route information in the redundant table is used to recover a broken route. In our RELREC, potential Relay Nodes may move away from their neighbour links, which makes the route recovery process more complicated. We will demonstrate that the false detection and false recovery is avoidable in RELREC in a couple of scenarios below.

A) Relay Node R moves out of the transmission range of C but is still within the transmission range of B.

As is shown in Figure 6, if link <B,C> does not break, B will not retransmit the data packet. R does not také any action if it does not overhear the retransmission at B, and the entry for link <B,C> will expire after Relay_Timeout. There is no false detection. If link <B,C> breaks and B retransmits the data packet, R will start a RELREC upon overhearing the retransmission. B and R update their Route Tables and R receives packets from B. However, C cannot receive the packet from R since R has moved away from C. Note that R will retransmit the packet if it does not receive acknowledgement from C. Upon overhearing the retransmission at R, potential Relay Node R' for <R,C> will launch a RELREC to relay the packet from R to C. In this way, the broken link is recovered through the recursive execution of RELREC scheme.

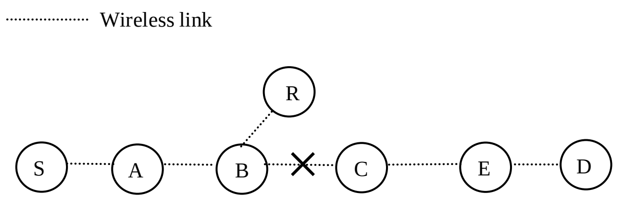


Figure 6. Analysis of correctness: R moves away from C.

In case there is no Relay Node for link <R, C> available, R will start a local recovery or global recovery depending on the situation. Therefore the false recovery can be avoided regardless of the existance of R'. In summary for this scenario, both false detection and false recovery can be avoided.

B) R moves out of the transmission range of B but is still within the transmission range of C.

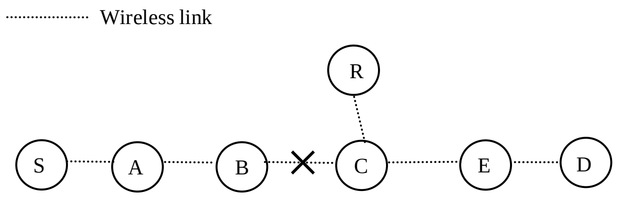


Figure 7. Analysis of correctness: R moves away from B.

In this case shown in Figure 7, R cannot overhear any transmission from B. R takes no action regardless of whether link <B,C> breaks or not. When link <B,C> does not break, we do not need R to launch a RELREC; when link <B,C> breaks, it is reasonable that R does not start a recovery process since it is no longer a qualified Relay Node for link <B,C>. In either situation, the silence of R serves to prevent false detection and false recovery. The entry for link <B,C> in R's Relay Table will expire after Relay_Timeout.

C) R moves out of the transmission range of both B and C.

This scenario is similar with the last scenario. R cannot overhear the transmission from B. R already has no relation to link <B,C>, and the entry for <B,C> in R's Relay Table will expire silently after Relay_Timeout. No false detection and false recovery.

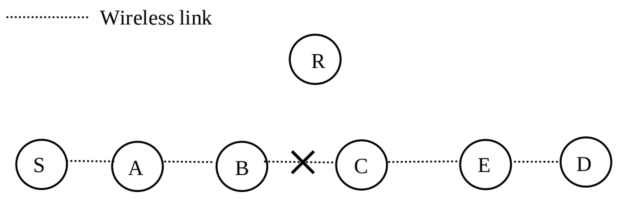


Figure 8. Analysis of correctness: R moves away from both B and C.

D) R is in the transmission range of both B and C, but the retransmission is caused by congestion.

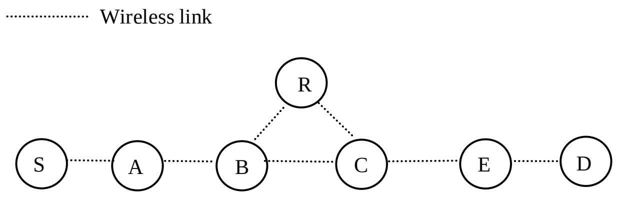


Figure 9. Analysis of correctness: retransmission caused by congestion.

As is shown in Figure 9, the link <B, C> does not break. The retransmission is caused by congestion. In this case, C will receive the packet during the retransmission and return acknowledgement to B. Upon receiving the acknowledgement from C, B just discard the NOTICE received from potential Relay Node and does not take any further reaction. The false detection problem is wisely avoided in RELREC.

3.2.5 Concerned Issues

There may be situations where one Relay Node serves for more than one link (racing among links), or more than one Relay Node available for a certain link (racing among Relay Nodes). We theoretically proposed their solutions for these situations as follows.

A) Racing among links

As is shown in Figure 10, Relay Node R serves for both link <A1, B1> and link <A2, B2>. If both of the two links break, and A1 and A2 transmit CONFIRM to R at the same time, the CONFIRM packets from A1 and A2 respectively may collide. The racing problem of interest here can be divided into two cases: A1 and A2 are located within the transmission range of each other (Figure 10 left), otherwise (Figure 10 right). Considering that it does not matter which one of A1 and A2 transmits to R first, we simply choose the delay time to be a random value between (0, 0.01), as is shown in equation (1) below.

$$T = rand(0,0.01) \tag{1}$$

More complex method can be found in [5].

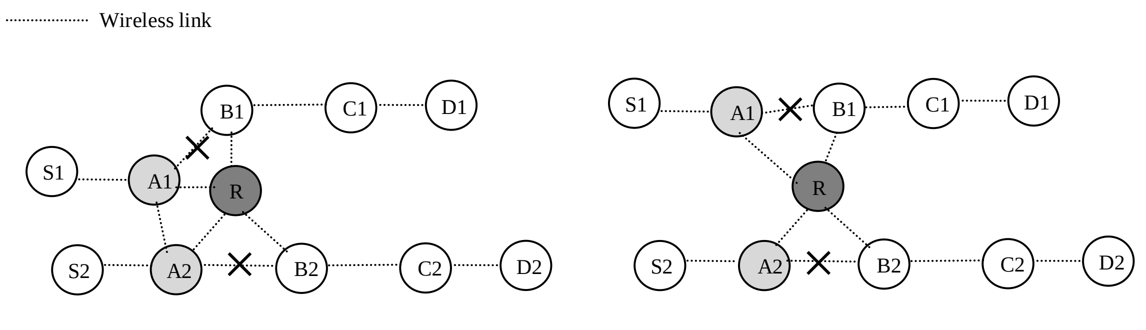


Figure 10. Racing between links: One Relay Node serves for multiple links.

B) Racing among Relay Nodes

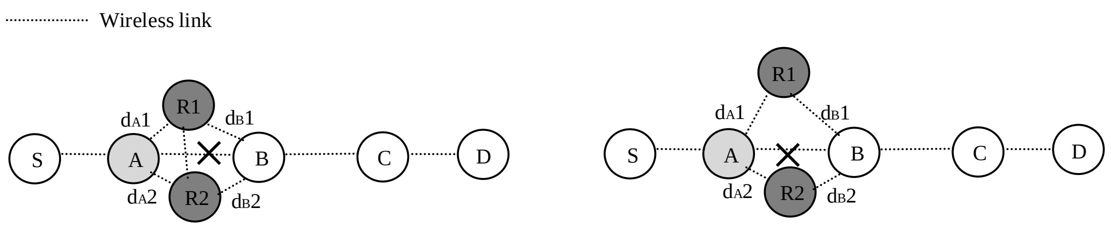


Figure 11. Racing between Relay Nodes: Multiple Relay Nodes serve for one link.

As is shown in Figure 11, link <A,B> breaks; R1 and R2 are two potential Relay Nodes for link <A,B>. Upon overhearing the retransimission at A, R1 and R2 will transmit a NOTICE to A. If the transmission at R1 and R2 are at the same time, the NOTICE packets from R1 and R2 may collide.

Assume that the distance from Relay Node i to A and B are d_{Ai} and d_{Bi} respectively; the distance information can be obtained using GPS. The distance between A and B is d. We prefer the Relay Node which is close to the centre between A and B, that is

- $d_{Ai} + d_{Bi}$ is small
- $|d_{Ai} - d_{Bi}|$ is small
- Expire time ti is large

where $(d_{Ai} + d_{Bi}) \sim [d,2r]$, $|d_{Ai} - d_{Bi}| \sim [0,2r-d]$.

In order to avoid collision, we require that each Relay Node waits for Priority seconds before sending the NOTICE packet, where

$$Priority = \frac{max\{d_A, d_B\} \cdot min\{d_A, d_B\}}{r^n \cdot t} \tag{2}$$

This ensures that the Relay Node which is located closer to the center of A and B, say node R1, will transmit NOTICE earlier than others. In the case where the other Relay Nodes locate within the transmission range of R1 (Figure 11 left), they can overhear the transmission of NOTICE at R1, thus cancelling their own transmission of NOTICE. Otherwise, if other Relay Nodes locate outside the transmission range of R1 (Figure 11 right), they will transmit their NOTICE to A after delaying for priority secondes. In this case, A just discards the NOTICE packetes received from other Relay Nodes.

Accordingly, the entry for link <A,B> in R's Relay Table is extended as follows:

Upstrm	**Dnstrm**	**Src**	**Dst**	**Expire_t**	**|AR|**	**|RB|**	**Priority**
A	B	S	D	t	d1	d2	$\frac{max\{d1, d2\} \cdot min\{d1, d2\}}{r^n \cdot t}$

We did not consider the backoff time for carrier sense in the original RELREC, so we do not consider it either here. But we do need to be careful with the maximum retransmission time, which can be realized by carefully choosing the value of n. According to IEEE 802.11, the backoff time for retransmission is defined as

$$BackoffTime = Random() * aSlotTime \tag{3}$$

where

Random() = pseudo-random integer drawn from a uniform distribution over the interval [0, CW], where CW is an integer within the range of values of the PHY characteristics aCWmin and aCWmax, aCWmin ≤ CW ≤ aCWmax.

aSlotTime = The value of the correspondingly named PHY characteristic.

In the simulation study, aCWmin = 31, aCWmax = 1023, aSlotTime = 0.00002s, the maximum retransmission time is 7. So the total amount of retransmission time is

$$U(0,31) * 0.00002 \leq t\_retrans \leq U(0,1023) * 0.00002 \quad (4)$$

Since the expected value for U(a,b) is (a+b)/2, with variance $(b-a)^2/12$, we can get the expected value for U(0,31) is 15.5 with variance 80. Therefore, the estimated value for t_retrans_min = 3.1×10^{-3} (the simulation result is larger than this minimum value, which is at the order of 10^{-2}). In equation (2), t is usually larger than 1×10^{2} s. If we let $n \geq 4$, we can ensure that the result of equation (2) is at the order of 10^{-4} which is smaller than the t_retrans_min.

3.3 Automatic Route Shortening

In an ad hoc network, there may be some route redundancy for two reasons: route recovery and node mobility. Route recovery may lead to a longer route; this is especially true in our pro-reactive relay recovery where the repaired route is usually one hop longer than the broken one. Node mobility causes topology change that may incur some route redundancy. Suppose the original route is {S-A-B-C-E-D} as is shown in Figure 12 (a). When node C moves into the transmission range of node A, as the position of C' shown in Figure 12 (b), node A can directly transmit data packets to node C without the participation of node B. Correspondingly, the route can be shortened to {S-A-C'-E-D}, shown in Figure 12 (c). Or if there is another route {X-Y-D} and node Y moves close to A, A can redirect the path to Y thus shortening the route to {S-A-Y-D} as is shown in Figure 12 (d).

The importance of route optimization lies in the performance optimization not only on routing layer, but also on transport layer. It has been observed that TCP performance decreases as the route length increases [22-24]. Therefore, automatic route shortening which adaptively optimize the route length is of great importance to ensure the integrated optimization of the network performance in a cross-layer manner.

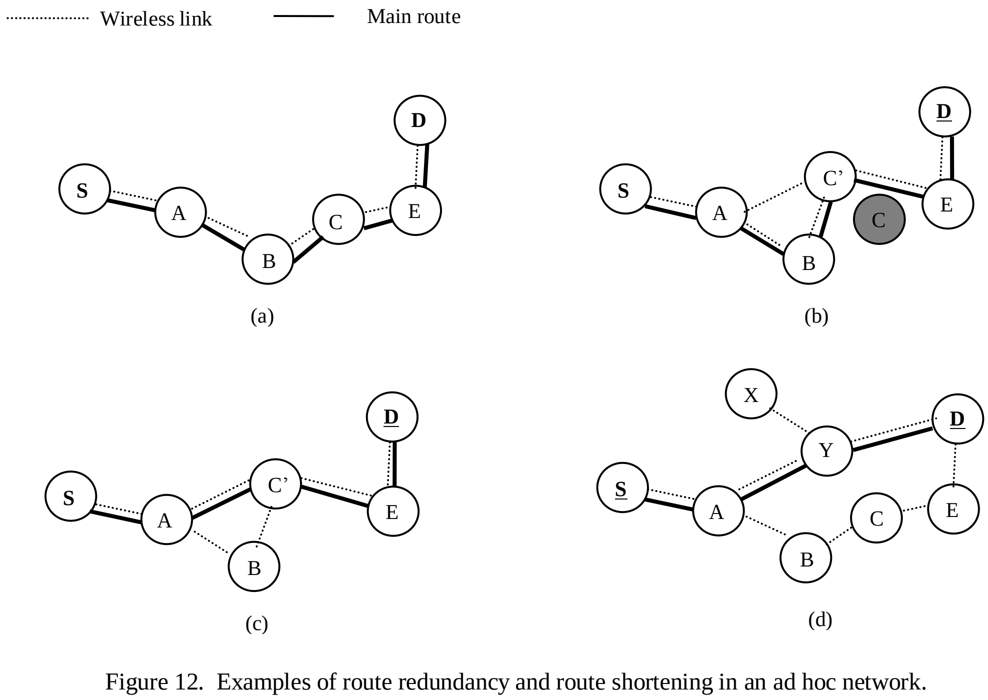
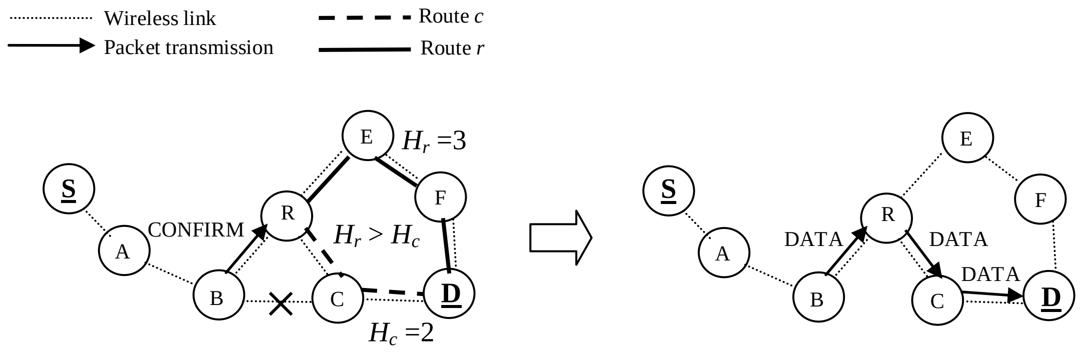


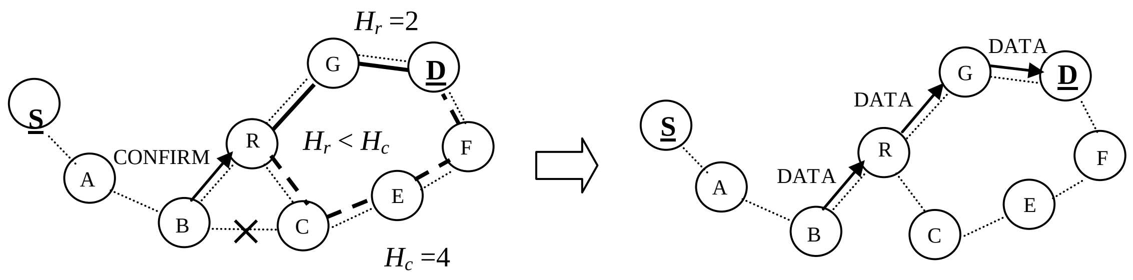
Figure 12. Examples of route redundancy and route shortening in an ad hoc network.

In this session, we will explain how our proposal works to automatically eliminate the redundancy in a route. Without the aid of source routing information [12], a route can also be shortened by using overheard information. We propose two automatic route shortening strategies called relay shortening and active shortening respectively to cope with the two kinds of route redundancy mentioned above.

3.3.1 Relay Shortening



(a) The case where route r is longer than route c



(b) The case where route r is shorter than route c

Figure 13. Priciple of relay shortening.

In the pro-reactive route recovery process of RELREC, the repaired routes are expected to be longer than the original broken routes due to the involvement of Relay Nodes. In order to optimize the recovered routes, we propose the relay shortening scheme for RELREC. A Relay Node R can initiate the route shortening as follows. When the R receives CONFIRM packet from the upstream node, R first searches its own Route Table for a route to the destination of the broken route. If R does not have a route to the destination of the broken route, R should update its Route Table according to the RELREC process, which is shown in Figure 5. If there is a route r to that destination, the Relay Node compares the hop count Hr of the route r with the hop count H_c indicated in the CONFIRM packet, and compares the sequence number SEQ_r of the route r with the sequence number SEQ_c indicated in the CONFIRM packet. In order to achieve a shorter route, the Relay Node should obey the following rule: the Relay Node should update its Route Table to use its link to the downstream node of the broken link toward the destination in addition to the recovery of the broken link in the following situations:

- $H_r > H_c$, which means the broken route is shorter than route r;
- $H_r = H_c$ and $SEQ_r < SEQ_c$, which means the length of both routes are the same but the broken route is fresher than route r.

Else the Relay Node does not update its Route Table after the recovery of the broken link; instead of relaying the packet from the upstream node to the downstream node, the Relay Node uses the route stored in its Route Table to recover the broken one.

In Figure 13, when Relay Node R receives CONFIRM packet from the upstream node B, R searches its own Route Table for a route to D. Figure 13 (a) depicts the first case where the broken link is recovered and route c is used to deliver packets after the recovery. Here R has a route r {R,E,F,D} in its Relay Table and it is longer than route c {R,C,D}, that is, $H_r > H_c$ ($H_r = 3$ and $H_c = 2$). Therefore R updates its Route Table and use node C as its next hop on the route to node D. Figure 13 (b) shows the second case where the Relay Node R uses the route r stored in its own Route Table. Here the route r {R,G,D} is shorter than route c {R,C,E,F,D}, that is, $H_r < H_c$ ($H_r = 2$ and $H_c = 4$). Therefore R does not need to update its Route Table; instead, R uses its own route r to deliver packet from the upstream node B to destination D.

3.3.2 Active Shortening

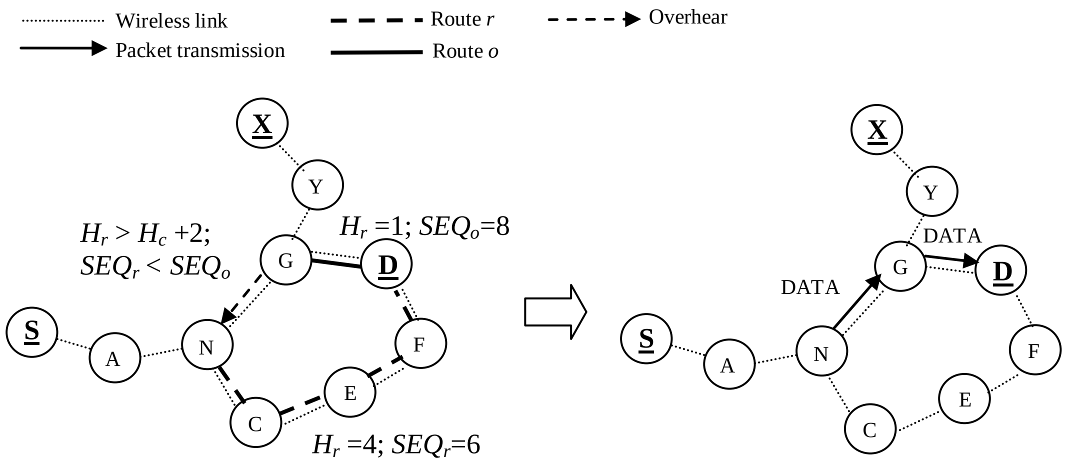


Figure 14. Priciple of active shortening.

In relay shortening, only Relay Nodes are qualified to initiate the shortening process. In order to fully optimize the routes in the network, we propose a more general route shortening strategy where any node in the network can actively initiate the route shortening process anytime it overhears a shorter route from its neighbors. However, such shortening may lead to vulnerable routes that are more likely to break afterwards. To limit the unnecessary route break caused by route shortening, we impose stronger conditions on active shortening to strike a balance between average route length and communication quality.

Assume a node N has a route r whose sequence number is SEQ_r with a length of H_r to a destination D in its Route Table and N overhears a route o to D whose sequence number is SEQ_o with length H_o from one of its neighbors. In our scheme, N can initiate an active shortening to use the route o instead of the route r to D if and only if

- $H_r > H_o+k$ (k is a predetermined integer) and $SEQ_r \leq SEQ_o$.

Note that the second condition actually enables to cover two situations: when $SEQ_r < SEQ_o$, N shortens the route to D by redirecting the remaining part of r to the other fresher route o; when $SEQ_r = SEQ_o$, the satisfaction of this condition means some nodes in the remaining part of route r has moved close to N, and thus N shortens the route r to D by just eliminating the unnecessary hops in the route.

Figure 14 exemplifies the principle of active shortening. Suppose node N has a route r {N,C,E,F,D} to the destination D. N overhears a route o {G,D}from node G. The length of route o is 2-hop shorter than that of route r, and at the same time route o is newer than route r. Therefore node N updates its Route Table and takes G as its next hop towards the destination D.

A brief comparison of relay shortening and active shortening strategies is shown in Table 1.

Table 1. Comparison of two shortening algorithms

Items	**Relay Shortening**	**Active Shortening**
Initiating node	Relay node	Any node
Conditions	$H_r > H_c$ or $\{H_r = H_c$ and $SEQ_r < SEQ_c\}$	$H_r > H_o+k$ and $SEQ_r \leq SEQ_o$

3.4 Performance Evaluation

The pro-reactive RELREC and automatic route shortening algorithms can be implemented into any existing routing protocol to optimize the route recovery process. We have the following concern in choosing the fundamental routing protocol based on which RELREC is implemented. Researchers have proposed numerous routing protocols which are generally divided into three main categories: proactive routing protocols, reactive routing protocols and hybrid routing protocols. Although each category has its own advantages, reactive routing protocols are generally considered out-perform proactive routing protocols in that the former protocols serve to reduce overhead and save resources within the network; while the hybrid protocols are not preferred due to their larger control overhead and complexity in implementation. Compared with proactive protocols and hybrid protocols, reactive protocols potentially require fewer control packets, which is essential to improve communication performance in ad hoc networks. Therefore we choose a most widely used reactive protocol AODV as a base to implement our proposal to reduce the control overhead.

We evaluate the performance in ns-2 simulator [36, 37] with CMU Monarch wireless extension [38, 39]. The performance of AODV and the competitive retransmission scheme DRRS [10] are taken as the baseline with which the comparison is carried out. Each simulation was run 4 times with different seeds and the average value is used in the final results. One simulation lasts for 5000 seconds, and the data in the first 1500 seconds are discarded to remove the effect of the initial status. The 200 nodes are moving under random waypoint model in a 1200m square. The movement scenario is as follows.

- Step 1: Each node is initially placed at a random position within the simulation area.
- Step 2: As the simulation time elapses, each node pauses at its current location for pause time.
- Step 3: Then it randomly chooses a new location and moves to that destination at a constant speed selected from a uniform distribution between the minimum speed and the maximum speed.
- Step 4: Upon reaching the destination, the node pauses again for pause time, and return step2.

The 95% confidence interval is shown in the figures to indicate the statistic reliability. The integer k in active route shortening is set to 3 in the simulations.

3.4.1 Evaluation Metrics

The performance of all these methods is evaluated from four aspects:

Average end-to-end time delay

End-to-end time delay characterizes the required packet transmission time from the source node to the destination node. It is the time between the creation of a packet and its successful arrival at the destination. If the destination fails to receive the retransmitted packet, the latency is considered as infinity and it will not be used to calculate the average end-to-end time delay. The components of the total time delay are D_{total} shown in equation (5), which include the delay in the routing process (route discovery and route recovery) on the network layer of the sender D_{NET_S}, the delay on the link layer of the sender D_{LL_S}, the delay in the outgoing queue of the sender D_{Q_S}, the delay on the mac layer of the sender D_{MAC_S}, the delay on the transmission on the physical channel D_{TRANS} , the delay on the mac layer of the receiver D_{MAC_R}, the delay in the outgoing queue of the receiver D_{MAC_R}, the delay in the ingoing queue of the receiver D_{Q_R}, the delay on the link layer of the receiver D_{LL_R}.

This metric is so sensitive to congestion that it will drastically increase as the congestion limit is reached, due to the packets waiting in large buffers. This metric evaluates the reaction speed of the proposal, and it is regarded to be the lower the better.

$$D_{total} = D_{NET_S} + D_{LL_S} + D_{IFQ_S} + D_{MAC_S} + D_{TRANS} + D_{MAC_R} + D_{IFQ_R} + D_{LL_R} \qquad (5)$$

In NS-2.34, D_{LL} consists of a link_layer_delay (25μs by default) and the time spent in ARP (10ms). The size of the interface queue is 50 packets. D_{MAC} consists of the medium access time including backoff time and the computing overhead (64μs by default but not used). D_{TRANS}

consists channel propagation delay ($4\mu s$ by default) and the time spent in the transmission which equals

$$D_{TRANS} = \frac{Packet\ Size}{Data\ Bit\ Rate} * Hop \tag{6}$$

In the simulation, the packet size is 512byte, the data bit rate is 1M, the average route length is approximately 10, and thus by substituting the parameters in equation (6) with the values we get $\overline{D_{TRANS}} = 0.04s$.

Normalized control overhead

Normalized control overhead is the number of control packets sent in the route recovery process multiplied by the number of hops the control packets traversed and divided by the total number of data packets successfully received by the destination and by the number of hops the data packets traversed, as is shown in equation (7). In order to calculate this metric, we add a counter field in the control message to indicate how many times it has been transmitted. High control overhead in the route recovery process can increase the probability of packet collision and may delay data packets in network interface transmissionn queues, thus further decrease the performance of the strategy. This metric measures the scalability of the proposal and the degree to which it will function in congested or low-bandwidth environments, and it is regarded that the lower the better.

$$O = \frac{N_{RREQ}+N_{RREP}+N_{RREQ}+N_{others}}{N_{p\_received}} \tag{7}$$

Where N_{RREQ}, N_{RREP}, N_{RREQ} stand for the number of Route Request, Route Reply, Route Error packets. $N_{p\_received}$ is the number of packets that has been successfully received by the destination. N_{others} is the number of other control packets caused by the routing protocol, e.g. in RELREC N_{others} is the number of NOTICE and CONFIRM.

Average route length

Average route length is measured by the average hop count in the network, as is shown in equation (8). It is mainly used to judge the efficiency of route shortening algorithms. It has been observed that the performance of TCP decreases as the route length increases [26-28], since the transmission medium has to be acquired at each participating node by sending RTS/CTS frame and this causes more and more delay as hop count increases. Therefore, the mechanism which leads to shorter average route length is preferred.

$$L = \frac{\sum_1^n h_i}{n} \tag{8}$$

Packet delivery ratio

Packet Delivery Ratio is defined as the number of packets successfully received by the destination divided by the number of packets sent by the source, as indicated by equation (9).

$$\text{ratio} = \frac{\mathrm{N_{p\_received}}}{\mathrm{N_{p\_sent}}} \tag{9}$$

This can be evaluated by setting up a number of “test” flows in the network, commonly a number of Constant Bit Rate (CBR) flows with a specified number of packets per second. Thereby we could obtain the packet delivery ratio through calculating the ratio between the number of packets originated by the ”application layer” CBR source and the number of packets received by the CBR sink at the final destination. We use UDP where there is no end-to-end retransmission so that every dropped packet results in a reduction of the delivery ratio. This metric characterizes both the completeness and correctness of the proposal, and it is regarded as the higher the better. This metric is very sensitive to congestion and mobility of nodes. A large enough test load will result in reduced delivery ratio for any protocol due to congestion. According to [40], when using 1024-byte packets, the congestion due to lack of spatial diversity became a problem for all protocols and one or two nodes would drop most of the packets that they received for forwarding. Since the goal of our analysis was to determine if the route recovery method could consistently cope with the broken routes when topology changes, we attempted to factor out congestive effects by setting the packet size to 512 bytes. In ns-2 simulator a traffic generator named cbrgen was developed to simulate CBR sources.

3.4.2 Parameters and Configurations

The parameters and configurations used in the simulation study are shown in Table 2~Table **4**.

Table 2. Parameters used in underlying routing protocol for the evaluation of RELREC and automatic route shortening

Parameters	Value
Route timeout	10s
RREQ retries	3
Time before RREQ retry	6s
RREQ timeout	10s
Local repair wait time	0.15s
RREP wait time	1s
Time before bad link removed	3s

Table 3. Parameters and configuration of traffic pattern for the evaluation of RELREC and automatic route shortening

Parameters	Value
Transport protocol	UDP
Traffic source type	CBR
Number of connections	2
Packet size	512 byte
Packet generation rate	4, 8, 16, 32 packets/s
Traffic load	16, 32, 64, 128 kbps

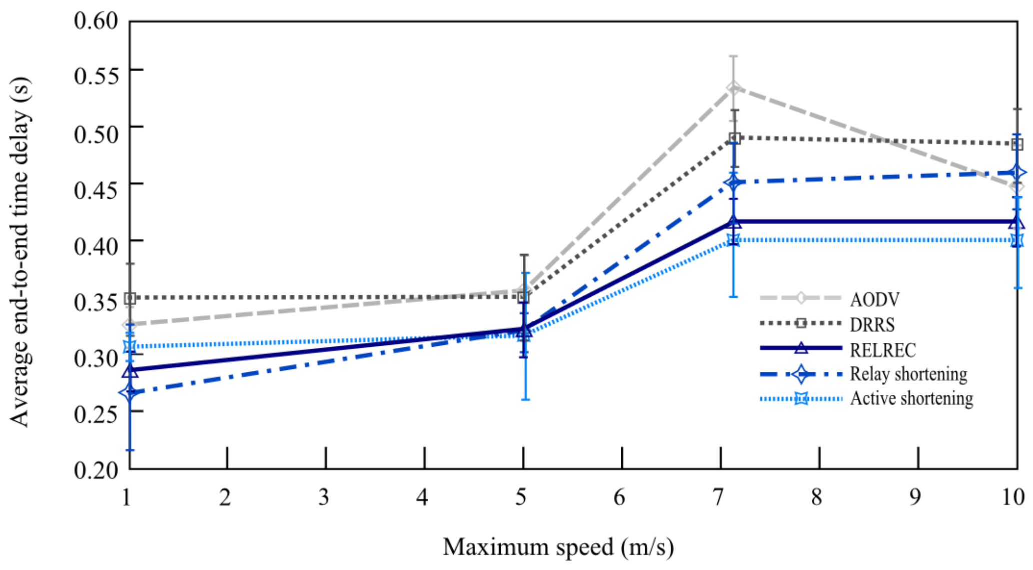
Table 4. Parameters of wireless scenario for the evaluation of RELREC and automatic route shortening

Parameters	Value
Bandwidth	2Mbps
Network scale	1200m x 1200m
Network density	50, 100, 150, 200 nodes/network
Node movement pattern	Random way point
Transmission range	100m
Minimum speed	1m/s
Maximum speed	1, 5, 7, 10m/s
Pause time	20
Simulation time	5000s (0~1500s discarded)

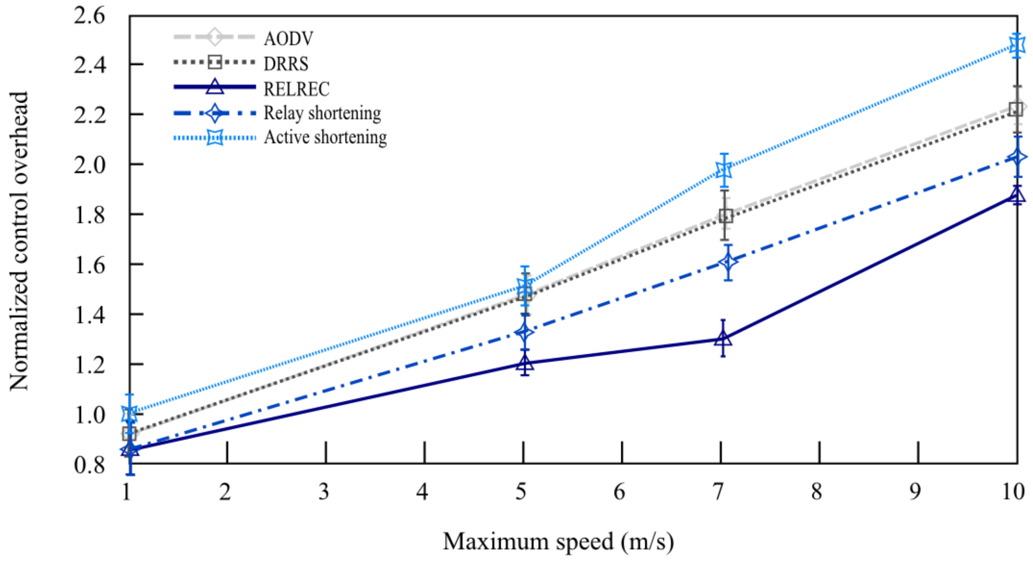
3.4.3 Results and Analysis

3.4.3.1 Impact of Node Mobility

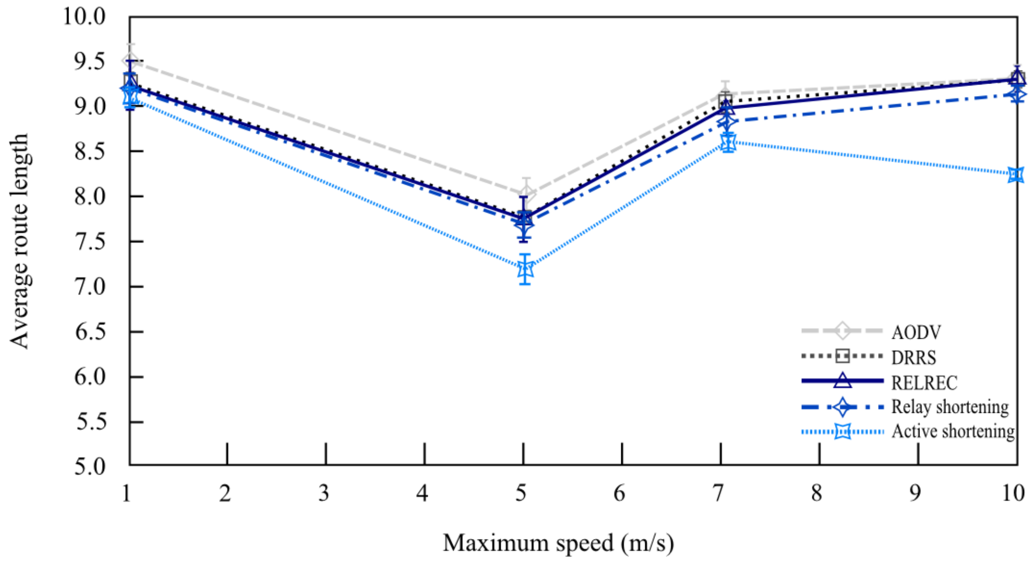
In this set of simulations the maximum speed is taken as a variable and set to 1, 5, 7 or 10m/s. The minimum speed is 1m/s, and the pause time is always set to 20s. The traffic load is 16kbps. The simulation results for this set of scenario are shown in Figure 15.



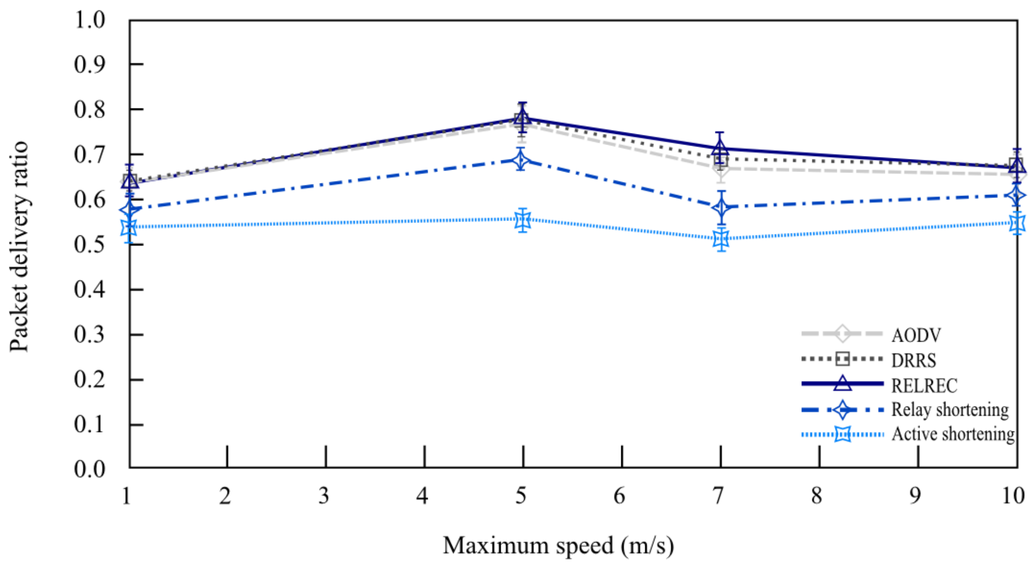
(a) Average end-to-end time delay v.s maximum speed



(b) Normalized control overhead v.s maximum speed



(c) Average route length v.s maximum speed



(d) Packet delivery ratio v.s maximum speed

Figure 15. Impact of node mobility.

Figure 15 (a) shows the average end-to-end time delay which increases as the maximum speed goes up. Not surprisingly, our proposed RELREC, shows a satisfying performance by reducing the time delay up to 25% and 15% in the best case when compared with AODV and DRRS respectively. This suggests that RELREC does successfully reduce the average time delay in route recovery process. As a benefit from the shortened routes, active shortening also ensures short time delay. When the maximum speed is larger than 5m/s, active shortening further reduces the time delay of RELREC by 5% on average. As for relay shortening, it unexpectedly leads to a longer delay than RELREC, though still shorter than AODV and DRRS. This may due to the extra operation time at the relay node in the relay recovery process. Another possible reason is that relay shortening does not shorten the average route as effectively as active shortening does so that it cannot benefit much from the shortened route.

As shown in Figure 15 (b), the normalized control overhead increases monotonically as the maximum speed goes up. The higher the node speed, the faster the topology changes. Therefore more route discovery processes will be triggered, which contributes to a higher control overhead. Active shortening peaks among all the algorithms, leading to the largest control overhead regardless of node mobility. There are three possible reasons for this phenomenon. First, even though we have implemented relay recovery scheme RELREC in active shortening, in practice no relay recovery was carried out in the simulation; that is, the broken routes were all recovered by global recovery or local recovery which causes large control overhead. Second, the shortened routes are not as stable as the routes discovered in a wholesome route discovery process by global recovery, causing more route break in the network. Hence more route recoveries were performed to ensure the operation of the network. That is why active shortening leads to even larger control overhead than AODV and DRRS, increasing up to 10% in the worst case. Third, because of the relatively unstable routes, more packets are lost before they successfully arrive at the destination.

By the definition of the normalized control overhead, if the number of data packets arrived at the destination decreases, the normalized control overhead will increase accordingly. As expected, RELREC generally ensures the smallest normalized control overhead among all the algorithms and it reduces the overhead by up to 18% in comparison with AODV and DRRS.

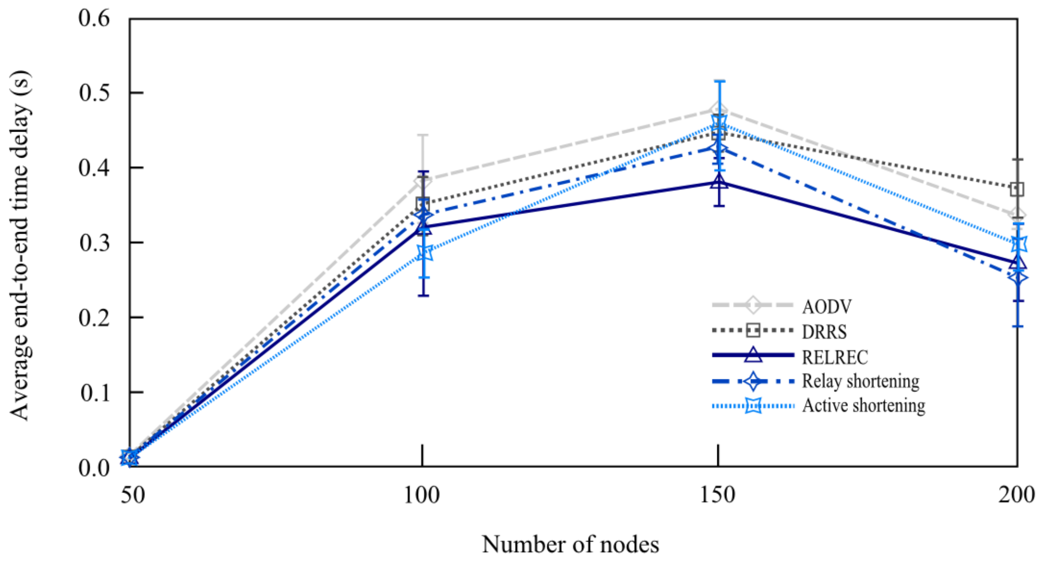
Figure 15 (c) shows the average route length measured by hop count, where active route shortening is clearly the best among all the algorithms. Active shortening wins its largest margins ahead when the maximum speed is 5m/s and 10m/s, shortening the route by up to 9% when compared with that of AODV. Relay shortening gives the second best performance in terms of route length, shortening 3% of the average route length in AODV. In all cases AODV leads to the longest route length among all the algorithms, closely followed by RELREC. One thing worth mentioning is that in the simulation, the average route length is directly determined by the distance between the source node S and the destination node D, which is independent among each scenario. In other words, the average route length is affected not only by the moving speed, but also the initial position, the moving direction, ect..

Figure 15 (d) shows the packet delivery ratio where DRRS and RELREC are very close to each other and slightly higher than AODV, which means the recovered routes in RELREC are generally stable enough to ensure the operation of the network and the communication quality. Considering AODV as the baseline, the shortening strategies lead to lower packet delivery ratio, which may be resulted by the unstable shortened routes as well as the congestion caused by the high control packet traffic. Note that active shortening is not as sensitive to topology change as other methods are. This is because in active shortening nodes frequently redirect the routes, which mitigates the effect of topology change caused by node mobility. The slight increase in the packet delivery ratio for AODV, DRRS, and RELREC at 5m/s is due to the relatively short route length, as is indicated by Figure 15 (c).

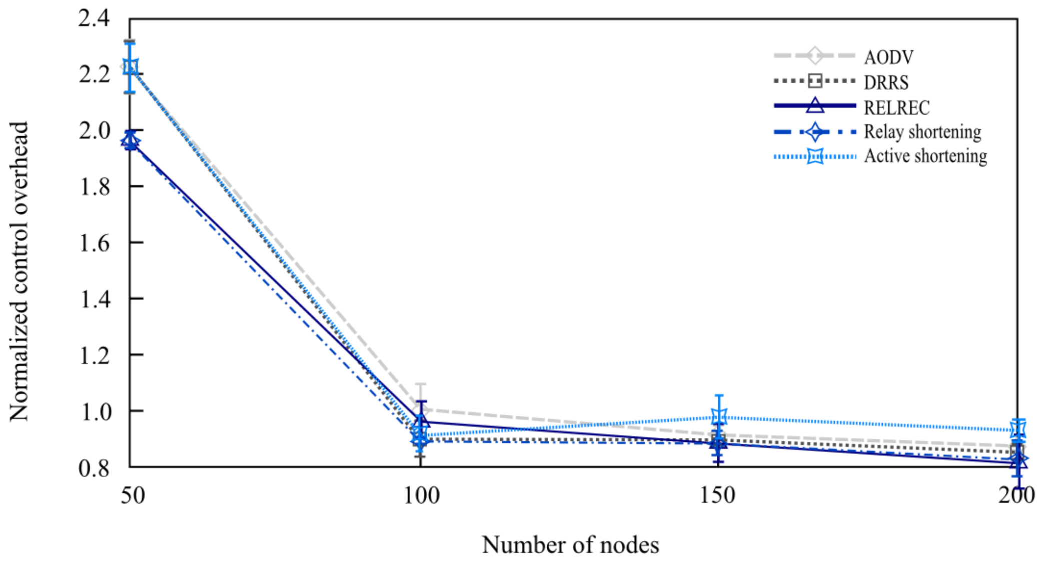
3.4.3.2 Impact of Network Density

In this set of simulations the number of nodes in the network is taken as a variable and set to 50, 100, 150, 200 nodes respectively. Minimum speed is always set to 1m/s and maximum speed 5m/s. Pause time is set to 20s. The traffic load is 16kbps. The simulation results for this set of scenario are shown in Figure 16.

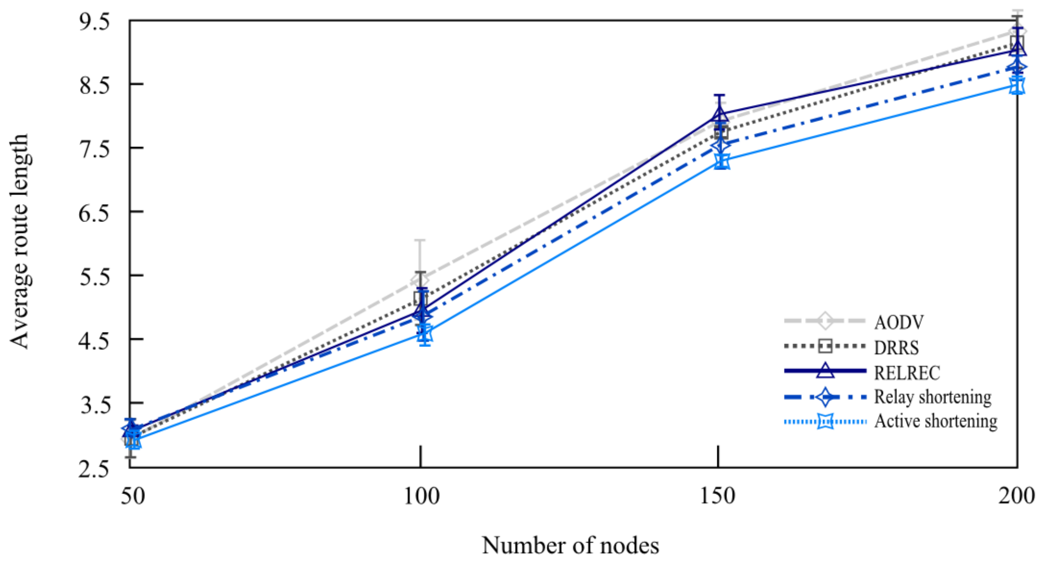
Figure 16 (a) indicates the average end-to-end time delay, which increases with the node density until the network density reaches 150. As the node density further increases, the average end-to-end time delay decreases. Note that as node density increases, the success ratio of route discovery and recovery increases and the time delay on the network layer D_{NET_S} decreases. The packets can be transmitted more smoothly, thus the waiting time in the interface queue D_{Q_S} decreases. However, since there are more nodes on the channel, the delay on MAC layer D_{MAC_S} increases; at the same time, the transmission time D_{TRANS} increases due to the increased route length, as is indicated in Figure 16 (c). Before the number of nodes reaches 150, the route length increases considerably; the increase in D_{TRANS} cast the main impact on the total delay, making the total time delay increases. When the number of nodes further increases from 150 to 200, the average route length does not increase too much; the decrease of D_{NET_S} and D_{Q_S} become the main factors that impact the total delay. Therefore, the total time delay decreases as the number of nodes increases from 150 to 200. Active shortening performs the best at the scenario of 100 nodes, which equals 27% reduction of the time delay of AODV and %13 of RELREC. As the node number increases, the degree of route shortening decreases; correspondently, the gap of time delay between active shortening and AODV shrinks. Therefore, the good performance of active shortening at 100 nodes may be resulted from the shortest average route length.



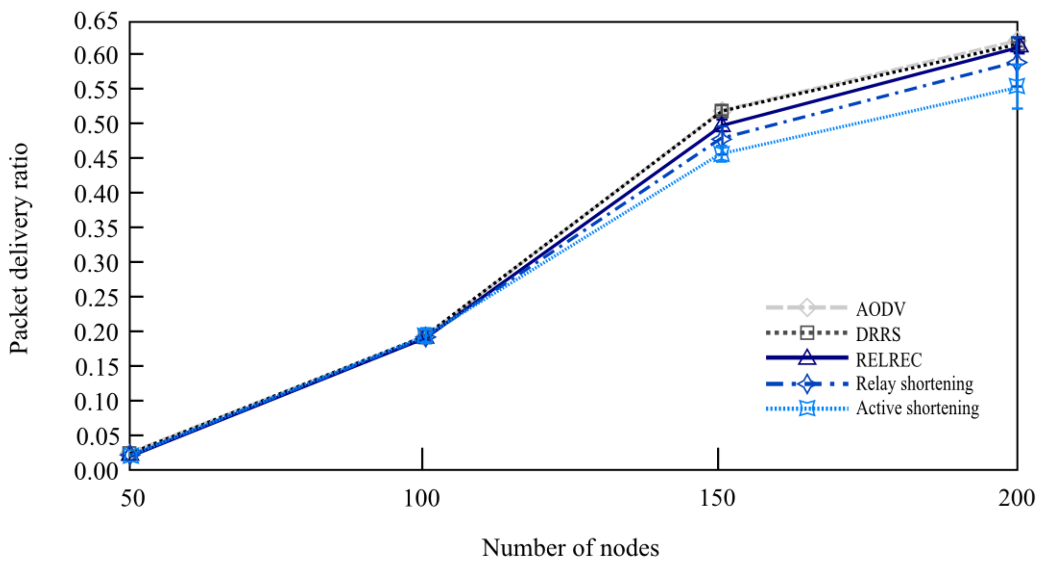
(a) Average end-to-end time delay v.s the number of nodes



(b) Normalized control overhead v.s the number of nodes



(c) Average route length v.s the number of nodes



(d) Packet delivery ratio v.s the number of nodes

Figure 16. Impact of network density.

As shown in Figure 16 (b), the normalized control overhead decreases as the node density increases. When the node density is low, it takes more attempts to find a route in the route discovery process, and the packet delivery ratio is low. It means that the total number of control packet generated is high, whereas only few packets are successfully received by the destination.

Therefore, the normalized control overhead, which is calculated as the total number of control packets divided by the total number of received packets, is much higher when the node density is low. As the node density increases, the packet delivery ratio increases significantly, which means more and more packets can be received by the destination. Hereby the normalized control overhead decrease as the node density goes up. Again, RELREC successfully reduces the control overhead, whereas the active shortening leads to the highest value. The possible reason has been discussed in 3.4.3.1.

As for the average route length which is shown in Figure 16 (c), it increases sharply as the node density increases. This is because, when the node density is too sparse and there are not enough intermediate relay node on the routes, nodes can only successfully receive the packets from the neighbors that are not far away. As the node density increases, it becomes more possible to establish longer routes which involve more intermediate relay nodes. Generally, relay shortening and active shortening shortened the route up to %6 and %7 in the best case compared with AODV. We believe that in a larger scenario the shortening effect may be more conspicuous.

In terms of the packet delivery ratio shown in Figure 16 (d), it generally increases with node density. This is reasonably because only when there are enough intermediate nodes on the route can the packets be successfully relayed to the destination. Whereas AODV, DRRS and RELREC almost have neck-to-neck performances, the performances of automatic route shortening strategies are far from satisfying. This is especially true when the network density is larger than 100 nodes, the packet delivery ratio of active route shortening decreases %11 from that of AODV in the worst case. As have mentioned in 3.4.3.1, automatic route shortening may lead to less stable routes, thus degrading the packet delivery ratio.

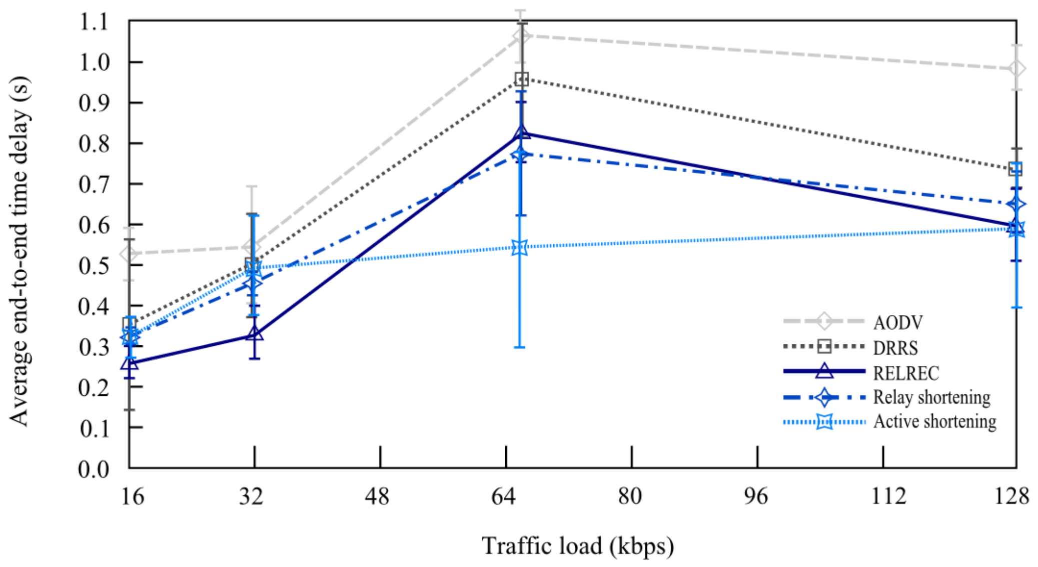
3.4.3.3 Impact of Traffic Load

In this set of simulations the traffic load is taken as a variable and set to 16, 32, 64, 128kbps. There are 200 nodes moving in the network at a minimum speed of 1m/s and a maximum speed of 5m/s. Pause time is 20s. The simulation results of this set of scenarios are shown in Figure 17.

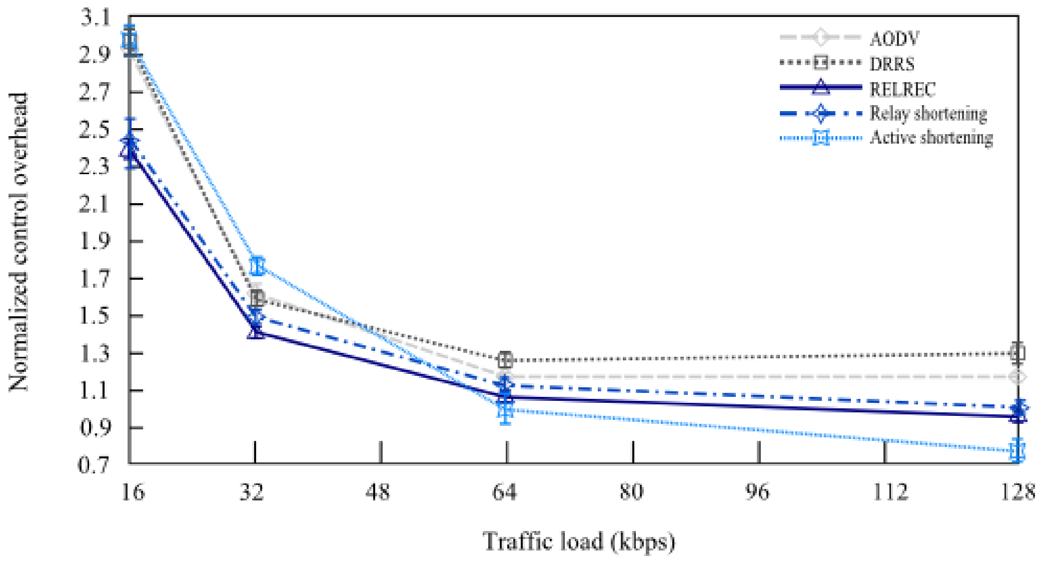
The average end-to-end time delay is shown in Figure 17 (a). RELREC effectively reduces the time delay by up to 49% and 31% respectively compared with AODV and DRRS. Benefited from the shortened route, route shortening strategies also successfully reduce the time delay. Active route shortening reduces up to 51% and 45% respectively compared with AODV and DRRS in the best case, and it ensures the shortest time delay among all the algorithms when the traffic load is larger than 48kbps. Relay shortening also reduces the time delay from the baseline, though not as significantly as active shortening does. As mentioned before, there are possibly two reasons: extra operation time at the relay nodes and the less shortened routes. When the traffic load is larger than 64kbps, relay shortening yields almost the same performance as that of RELREC. The waiting time in the interface queue (IFQ) and the average route length are two important factors that affect the time delay in this set of scenario. The traffic load of 64kbps is the turning point in terms of time delay. Before the traffic load reaches 64kbps, the time delay increases as the traffic load goes up. This is due to the longer waiting time in the interface queue (IFQ) D_{Q_S}, and the benefit from slightly shorter route length does not overtake the increase of waiting time in IFQ. Conversely, when the traffic load is above 64kbps, the time delay slightly decreases as the traffic load increases. This may due to the obvious shorter route length whose effect overtakes the increase of waiting time in IFQ. Different from other algorithms, active shortening is less sensitive to the increase of traffic load, which indicates that the route redirection may serve to redirection traffic load, thus alleviating the potential congestion in the network.

As shown in Figure 17 (b), the normalized control overhead decreases as the traffic load increases. According to the simulation data, the absolute number of control packet actually slightly increases as the traffic load goes up. However, the number of received packet significantly increases to a much higher degree. As a result, the normalized control overhead turns out to be dropping. Just as RELREC does, relay shortening reduced the control overhead by up to 17% and 28% respectively compared with AODV and DRRS in the best case. Take the traffic load of 64kbps as the threshold, active shortening causes the highest control overhead when the traffic load is below 64kbps due to the unstable routes. When the traffic load is above 64kbps, active shortening effectively alleviates the potential congestion through redirecting the data traffic, thus reducing the normalized control overhead.

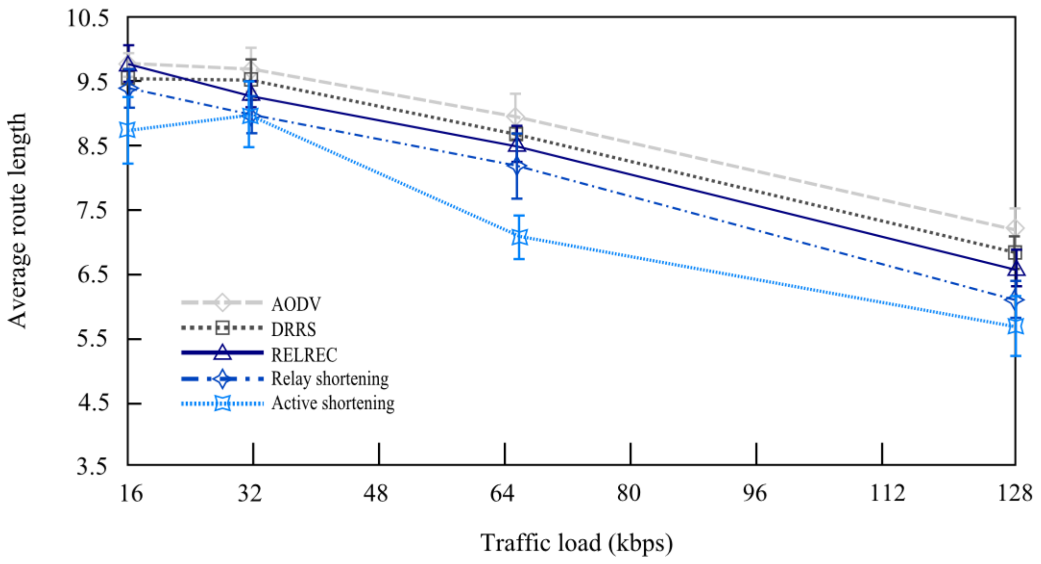
The average route length decreases as the traffic load increases, as is shown in Figure 17 (c). Notice that only the packets that are successfully received by the destination will be used to count the route length. As the traffic load increases, more route failure happen due to congestion. Longer routes have higher probability to break and the packets transmitted on the longer routes thus cannot arrive at the destination, whereas packets transmitted on the shorter routes have higher probability to be successfully received by the destination. In the scenario of heavy traffic load, only packets that are transmitted along short route can be received. Only these packets are used to count the route length, so that is why the average route length decreases as the traffic load increases. As for the two route shortening strategies, generally active route shortening works more effectively than relay route shortening. Active shortening initially shortens the route by 5% at the lowest traffic load, and the efficiency increases to 23% at the highest traffic load in this set of scenarios. Relay shortening does not work as effectively as active shortening, but it still shortens the route to 14% compared with that of AODV in the best case.



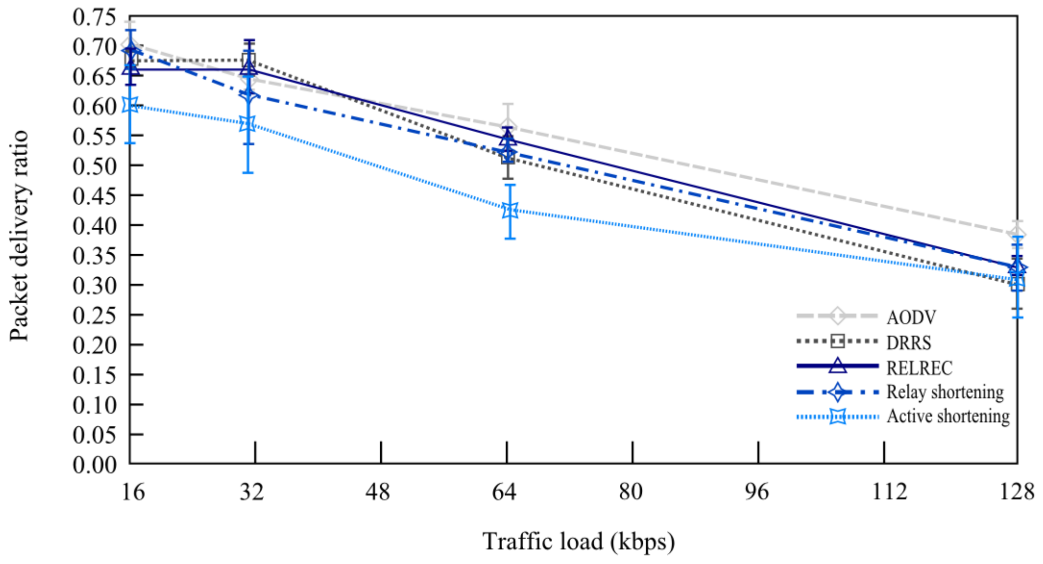
(a) Average end-to-end time delay v.s traffic load



(b) Normalized control overhead v.s traffic load



(c) Average route length v.s traffic load



(d) Packet delivery ratio v.s traffic load

Figure 17. Impact of traffic load.

Figure 17 (d) reveals that the packet delivery ratios of all the methods decrease dramatically as the traffic load increases. The reason is that heavy traffic load causes congestion, thus degrading the packet delivery ratio. Relay route shortening has similar performance as RELREC, which is slightly lower than that of AODV, but slightly higher than that of DRRS. Active route shortening leads to the lowest packet delivery ratio, which may be resulted by the shortened unstable routes.

3.5 Summary

In this chapter the pro-reactive route recovery algorithm RELREC and two automatic route shortening algorithms have been proposed. RELREC dedicates to reduce the time delay and control overhead in the route recovery process. In this algorithm, a common neighbor of the upstream node and downstream node of a broken link is defined as relay node and is utilized to repair the broken route in an instantaneous manner. Relay nodes monitor the link state through promiscuous overhearing. One or more substitute routes via the relay nodes are usually ready before the route break is confirmed, which is the proactive characteristic of this proposal; while the substitute routes are used to recover the broken route only after the confirmation by the upstream node that the route has really broken, which is the reactive characteristic. Since a substitute route is immediately available after the route break is confirmed, this scheme causes no extra time delay in the route recovery process. There is no broadcast of any control packet in the network, thus the control overhead being reduced. In order to optimize the route length in the network, we also implemented two automatic route shortening strategies: relay route shortening and active route shortening. While relay shortening can only be initiated by the relay nodes in pro-reactive relay recovery processes, the active shortening can be launched by any node whenever they overhear a shorter route from their neighbors.

Simulation study by ns-2 simulator has demonstrated clearly that our proposed RELREC effectively reduces the average end-to-end time delay by up to 49% and 31%, and the normalized control overhead by up to 17% and 28% compared with the previous algorithms AODV and DRRS respectively in the best cases. According to the simulation results, the data delivery ratio by our proposed scheme is close to DRRS and AODV. The simulation results confirm that our proposal can provide quick and low-cost route recovery without degrading the packet delivery ratio, while it is adaptive to node mobility and traffic load. As for the two route shortening strategies, active shortening works more effectively than relay shortening, leading to by far the shortest average route length and the smallest end-to-end time delay among all the algorithms in the simulation. Meanwhile, the slightly larger control overhead traffic and relatively lower packet delivery ratio harms the performance of active route shortening. The improvement of the control overhead traffic and the packet delivery ratio by active route shortening will be included into our future work.

CHAPTER 4

RELREC GOSSIP ROUTING (RGR)

4.1 Introduction

Routing protocol generally has two functions: to find a route in route discovery stage and to maintain the route in route recovery stage if route breaks. A novel route recovery strategy RELREC has been proposed in Chapter 3 to combine the benefits of both proactive and reactive routing protocols and to minimize their drawbacks. Different from traditional route recovery methods where control messages are broadcast either globally [12, 21] or locally [4-6, 9-20] to find a substitute route, there is no broadcast of any control message in RELREC; instead, relay nodes continuously monitor the link states through promiscuous overhearing and only NOTICE/CONFIRM packet pairs are exchanged between the relay node and the upstream node of the broken link. To be more clear, one or more substitute routes usually become ready when relay nodes send NOTICE packet to the upstream node of a potential broken link, while the route recovery process actually starts only when the upstream node confirms the link break and replies a CONFIRM packet to one of the relay nodes. Since this scheme does not broadcast any control packet, it can effectively recover a broken link without heavy control overhead traffic. Also, it helps reduce the time delay due to the recovery, since substitute routes are already available when the upstream node initiates the route recovery process. Simulation results confirm that the RELREC scheme efficiently reduces the end-to-end time delay and control overhead compared with a most popular protocol AODV [21] and a similar retransmission scheme DRRS [10], while ensuring a satisfying packet delivery ratio which is as high as that of AODV.

Whereas RELREC works efficiently in route recovery stage with the aid of Relay Tables maintained at each potential relay node, we expect to explore its full potential by extending the usage of Relay Table information in route discovery stage to help further reduce the unnecessary redundant rebroadcast of control messages. Based on the straightforward idea that potential relay nodes rebroadcast with probability p ($0<p<1$) after receiving route request, we incorporate the Relay Table information into gossip routing scheme, which we call Relay Gossip (RGR), to strike a balance between packet delivery ratio and control overhead reduction. In this chapter, it is shown that RGR– essentially, relay nodes tossing a coin to randomly decide whether to forward a message or not– can be used to significantly reduce the number or messages transmitted in the routing discovery process without degrading the packet delivery ratio.

Compared with existing methods, the characteristic and advantage of RGR is that it is based on the RELREC scheme which already effectively improves the network performance in route recovery stage; therefore the extended usage of Relay Table in route recovery stage is expected to further reduce control overhead and time delay without introducing extra cost. Extensive simulations are conducted at the end of this chapter to analysis the effect of RGR in detail compared with basic gossip routing and vanilla RELREC. Since RGR is based on RELREC, the assumptions in RELREC which have been presented in session 3.1 also stand here.

4.2 Rational for RGR

It is widely known that flooding-based broadcast scheme causes severe redundant rebroadcast [41] that degrades the performance of the network. Here we briefly rephrase the analysis provided in [41]. Consider a scenario shown in Figure 18, B receives a broadcasting message from A and decides to rebroadcast the message. Let S_A and S_B denote the transmission of node A and B respectively. The additional area that can benefit from B's rebroadcast is the gray shadow area, denoted as S_{A-B}. Suppose the radii of S_A and S_B is r. We can derive that $|S_{B-A}| = |S_B| - |S_{AB}|$. In other words, the intersection area of the two circles cannot benefit from B's rebroadcast. Recall that relay nodes for link <A, B> are the node which is the nodes which are located within the transmission range of both A and B; that is, relay nodes are exactly located in the $|S_{AB}|$ area. Suppose R has the same transmission range as A and B, the additional area that can benefit from R's rebroadcast is the black shadow area, which is only a trivial margin. Therefore the rebroadcast of relay nodes which are located in the common area of S_A and S_B is highly redundant and unnecessary. In order to reduce redundant rebroadcast and to alleviate the potential broadcast storm problem in the network, the gossip scheme is implemented at Relay Nodes and allow them to decide whether to rebroadcast or not by tossing the coin, which is expected to help further reduce the broadcasting of control packets in the route discovery stage.

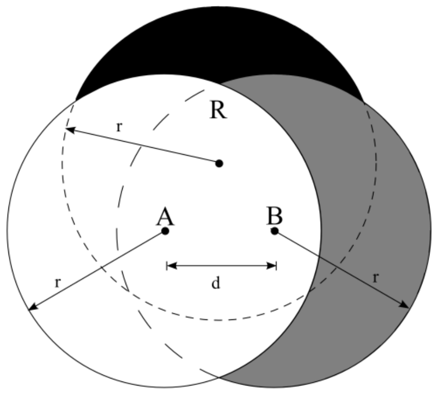


Figure 18. The rebroadcast of Relay Node *R* is highly redundant.

4.3 Design of RGR

Based on the rationality analyzed above, the Relay Gossip Routing (RGR) is proposed in a quite straight-forward way. Since RGR is based on the RELREC, the assumptions in RELREC also stand in RGR. Suppose a source node sends the Route Request (RREQ) with probability 1. When a node N receives a RREQ from the upstream node U for the first time, it checks whether there is an entry for that upstream node U in its Relay Table. If there is an entry for node U in N's Relay Table, which means N is a relay node and its rebroadcasting may be highly redundant, then N broadcasts the RREQ with probability p and with probability 1-p it discards the RREQ. If there is no entry for U in N's Relay Table, N has no knowledge of the redundancy in local area; in order to secure the performance of the network N should rebroadcast the RREQ with probability 1.

The RGR scheme has two major advantages. On one hand, it helps to address the premature gossip death [30] problem. The existing gossip approaches allow all nodes in the network to gossip the message, and in some cases the message may "die out" in a certain fraction of the executions, which is called the premature gossip death. As a comparison, in RGR the gossip scheme is limited to Relay Nodes since their rebroadcast has a high probability to be redundant than that at non-relay node. By carefully choosing the value of p, RGR is expected to eliminate the potential premature gossip death problem as well as reduce the rebroadcast redundancy, thus improving the overall performance of the network. On the other hand, RGR can easily be implemented upon the pro-reactive RELREC route relay method without causing further cost. The RELREC method based on Relay Table maintenance already significantly improve the performance of the network in route recovery process, and the sustainable usage extension of Relay Table information to route discovery process yields further benefit from original RELREC in terms of reducing control overhead and time delay

4.4 Performance Evaluation

In this section the performance of the proposed Relay Gossip Routing (RGR), implemented based on RELREC, is evaluated through simulation study in ns-2 simulator [36, 37] with CMU Monarch wireless extension [38, 39]. The performance of basic gossip routing and RELREC are taken as the baseline with which the comparison is carried out. Each simulation was run 4 times with different seeds and the average value is used in the final results. We have confirmed statistically that these results are reliable enough from the viewpoint of their small confidence interval, though the intervals are not shown in the figures below to avoid their complexity in observation.

4.4.1 Evaluation Metrics

The performance of RGR is evaluated from four aspects:

Normalized Control Overhead

The Normalized control overhead is the number of all control packets divided by the total number of data packets successfully received. It implies the degree to which RGR reduces the broadcasting of control packets.

Packet Delivery Ratio

Packet delivery ratio is calculated as the number of packets successfully received by the destination divided by the number of packets sent by the source. The premature gossip death will result in low packet delivery ratio, therefore it is possible to tell whether gossip death occurs or not from the packet delivery ratio. This metric is used to judge the correctness of the RGR algorithms. The effect of gossiping probability at Relay Nodes on packet delivery ratio is studied before the performance comparison carried out.

Average End-to-End Time Delay

The average end-to-end time delay is the average value of the time between creation of packets and their successful arrival at the destination, and it characterizes the reaction speed of the mechanisms.

Average Route Length

The average route length is measured by the average hop count in the network, and this metric is used to check whether RGR leads to longer route in the network.

4.4.2 Parameters and Configurations

The configuration of the network and corresponding parameters are shown in the Table 5~Table 7.

Table 5. Parameters used in underlying routing protocol for the evaluation of RGR

Parameters	Value
Route timeout	10s
RREQ retries	3

Time before RREQ retry	6s
RREQ timeout	10s
Local repair wait time	0.15s
RREP wait time	1s
Time before bad link removed	3s

Table 6. Parameters and configurations of the traffic pattern for the evaluation of RGR

Parameters	Value
Transport protocol	UDP
Traffic source type	CBR
Number of connections	1
Packet size	512 byte
Packet generation rate	4, 6, 8, 10, 12, 14, 16, 18, 20 packets/s
Traffic load	16, 24, 32, 40, 48, 56, 64, 72, 80 kbps

Table 7. Parameters and configurations of the wireless scenario for the evaluation of RGR

Parameters	Value
Bandwidth	2Mbps
Network scale	1200m x 1200m
Network density	200 nodes/network
Node movement pattern	Random way point
Transmission range	100m
Minimum speed	1m/s
Maximum speed	5m/s
Pause time	20s
Simulation time	5000s (0~1500s discarded)

4.4.3 Effect of Gossiping Probability on Packet Delivery Ratio

According to [33], the authors concluded that the threshold probability is in the range [0.65, 0.75]. However, in the proposed RGR the gossiping is limited to relay nodes only, thus the conclusion from global gossiping scheme cannot be directly adopted. Instead, the packet delivery ratio under varied gossip probability at only relay nodes is studied to help decide the optimum value of p.

In this set of simulations the traffic load is 16kbps. There are 200 nodes moving in the network at a minimum speed of 1m/s and a maximum speed of 5m/s. Pause time is 20s. A simulation lasts 5000s, and data of the first 1500s are discarded to remove the effect of transit status. Each simulation runs 4 times with different seeds, and the average value is used as the final results in each set. We have confirmed statistically that these results are reliable enough from the viewpoint of their small confidence interval, though the intervals are not shown in the figures below.

The relation between packet delivery ratio and gossiping probability is shown in Figure 19. The packet delivery ratio equals or even goes higher than that of $p = 1$ when the value of p is larger than a certain point lies between (0.40, 0.50); at the same time the total control overhead is smaller than that of $p = 1$. This means if we choose the gossiping probability above the certain value lies between (0.40, 0.50), it is possible to reduce control overhead without harming the packet delivery ratio.

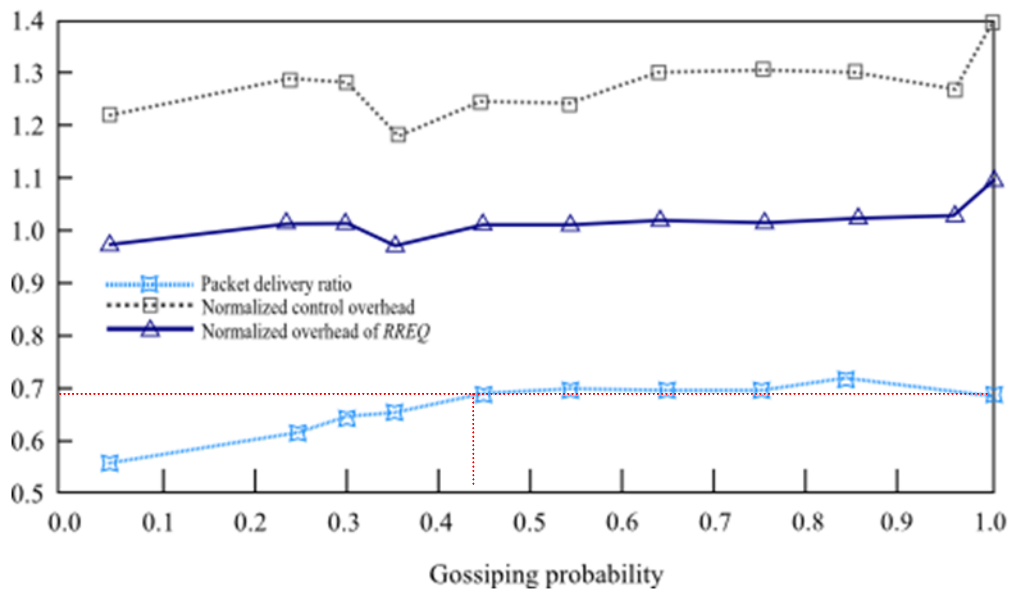


Figure 19. Packet delivery ratio and normalized control overhead under different gossiping probability.

4.4.4 Results and Analysis

In this session, the performance of RGR is evaluated. The simulation setting is the same as that of 4.3.3, and the gossiping probability p = 0.5. The simulation results are shown in Figure 20~Figure 23.

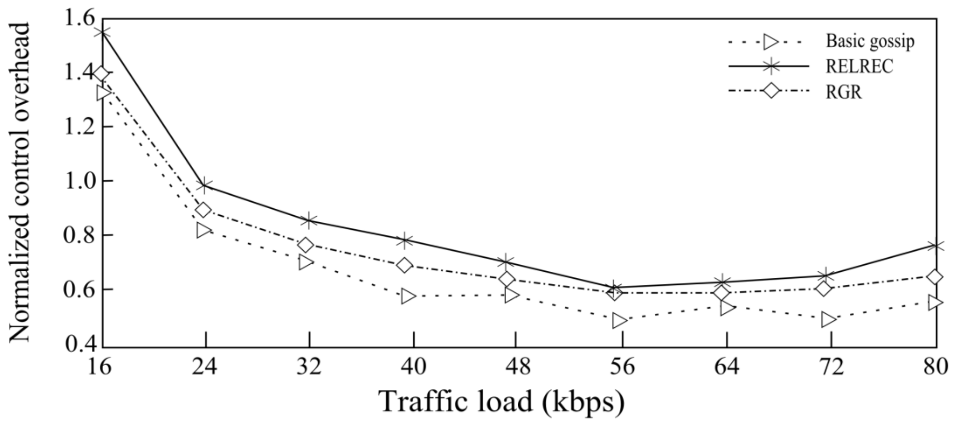


Figure 20. Normalized control overhead v.s. traffic load (kbps).

As indicated in Figure 20, the normalized control overhead decreases as the traffic load goes up. Notice that the normalized control overhead is calculated as the transmission of control packets sent in the route recovery process divided by the transmission of data packets successfully received by the destination. The fact that the number of received packets increases much faster than that of control packets leads to the decreasing normalized control overhead. On the one hand, the network topology is relatively stable in this simulation setting. All nodes move at a low speed and the CBR/UDP connection is fixed throughout the whole simulation. After the routes between the communication pair are established, traffic load only has marginal impact on the number of route discovery executions; though higher traffic load may cause congestion in the network, thus still triggering route discovery process and introducing more control packets. According to the simulation data, the total number of control packet only increases slightly as traffic load goes up. On the other hand, the number of successfully received packets increases sharply with the traffic load. The combined impact of the two aspects yields the decreasing normalized control overhead. RGR successfully reduces the normalized control overhead by up to 17% in the best case compared with RELREC. The basic gossip yields the lowest normalized control overhead by further reducing 23% of that of RGR in the best case. This is because all nodes toss the coin to decide whether to forward a message or not in the basic gossip, whereas only Relay Nodes are allowed to do so in RGR.

As shown in Figure 21, the packet delivery ratio decreases slightly as the traffic load goes up. The performance of RGR is almost the same as that of RELREC. Basic gossip routing yields the worst performance by degrading the packet delivery ratio up to 17% compared with RELREC in the worst case. Since the rebroadcast of RREQ messages are completely random in basic gossip, it cannot assure that all nodes in the network receive the RREQ messages. That’s why the packet delivery ratio decreased in basic gossip; it achieves low control overhead at the sacrifice of packet delivery ratio. As a contrast, in RGR the gossiping is only limited to relay nodes whose rebroadcast may be highly redundant. Therefore RGR achieves the same packet delivery ratio as RELREC does. It is conspicuous that the basic gossip achieves the reduction in the control overhead at the sacrifice of packet delivery ratio.

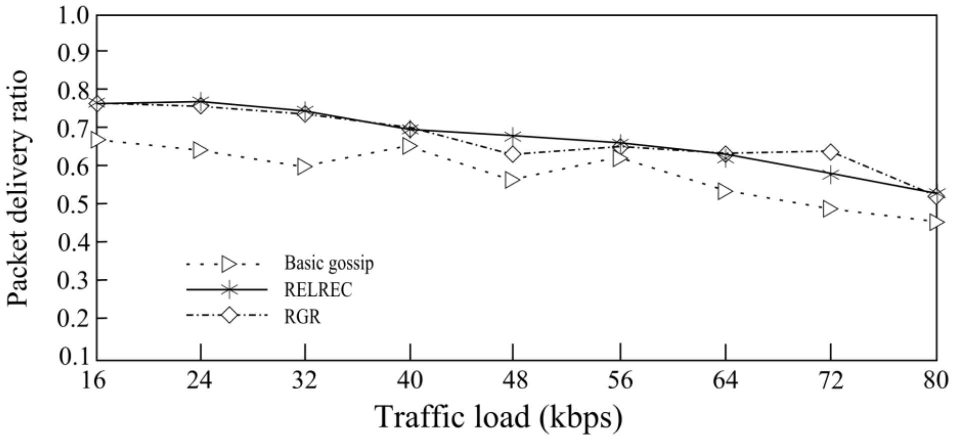


Figure 21. Packet delivery ratio v.s. traffic load (kbps).

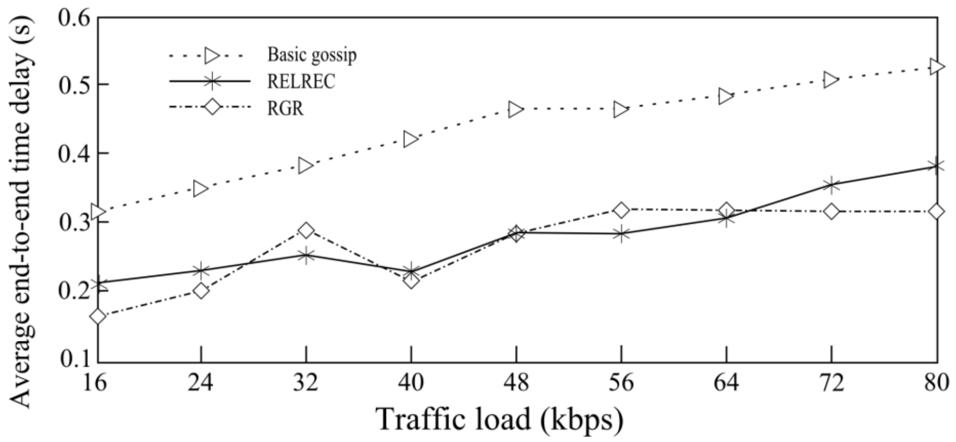


Figure 22. Average end-to-end delay (s) v.s. traffic load (kbps).

The relation between average end-to-end time delay and traffic load is shown in Figure 22. The average end-to-end time delay increases as the traffic load goes up, which may be the result of the increasing waiting time in the Interface Queue. Basic gossip causes approximate twice the time delay as that of RELREC and RGR. In basic gossip, nodes have to initiate a second route discovery process if the premature gossip death happens; as a result, it takes more time to find a route in the route discovery stage. RGR yields similar performance to RELREC, which indicates that the premature gossip death is effectively avoided in RGR so that the route discovery in RGR is as efficient as that in RELREC.

As for average route length indicated in Figure 23, it is insensitive to the increase of traffic load. This is reasonable in that average route length is mainly determined by the network scale. However, when the traffic load is larger than 64kbps, the average route length in basic gossip is approximately 1-hop longer than that of RGR and RELREC. This means that when the traffic load increases, the route discovery in basic gossip may include certain route redundancy. As have mentioned before, the performance of TCP is affected by the average route length [22-24]. From this point of view, RGR may also help improve the TCP performance as it reduces the average route length compared to basic gossip routing, though this is not included in the scope of this thesis.

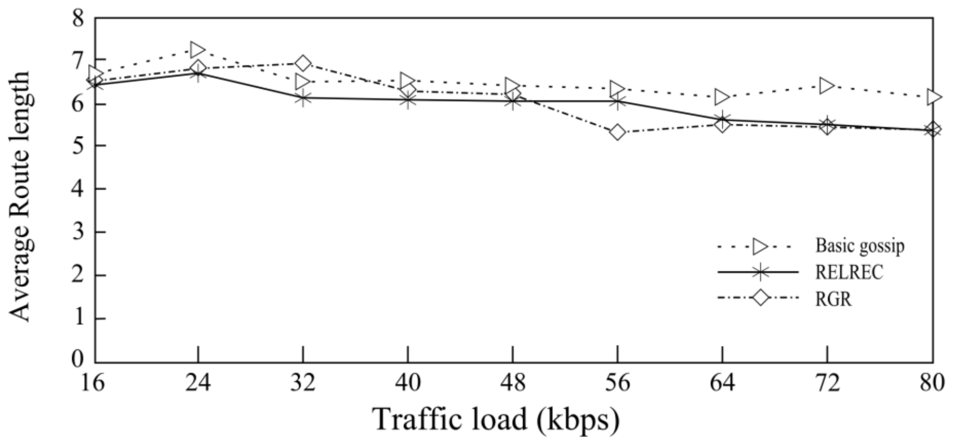


Figure 23. Average route length v.s. traffic load (kbps).

4.5 Summary

In the pro-reactive RELREC algorithm presented in Chapter 3, Relay Nodes in the network are utilized to effectively recover broken routes. Broadcast of control messages is avoided so that time delay and control overhead can be reduced. In order to become a Relay Node in RELREC scheme, each node in the network operates in promiscuous mode and uses a Relay Table to keep information of the links based on overheard packets. However, the information in Relay Tables is only utilized in route recovery process in RELREC. In this chapter, we extended the usage of Relay Table to route discovery process by incorporating Relay Table information into gossip algorithm, and propose what we call Relay Gossip Routing (RGR). Only Relay Nodes are allowed to rebroadcast under a gossiping probability in RGR, which differs from existing gossip methods. The main purpose of RGR is to reduce the redundant rebroadcast in route discovery process so that the potential broadcast storm problem can be alleviated, while at the same time the Relay Table information can be fully utilized to yield further benefit from vanilla RELREC.

The relation between packet delivery ratio and gossiping probability is firstly studied. It is shown that the packet delivery ratio equals or even becomes higher than that of p = 1 when the value of p is larger than a certain point between [0.40, 0.50], while the total control overhead is smaller than that of p = 1. Accordingly, the gossiping probability is set to 0.5 and the performance of RGR is studied. Simulation results show that RGR successfully reduces the normalized control overhead by up to 17% in the best case compared with RELREC, and the basic gossip yields the lowest normalized control overhead by further reducing 23% of that of RGR in the best case. This is because all nodes toss the coin to decide whether to forward a message or not in the basic gossip, whereas only Relay Nodes are allowed to do so in RGR. However, when it comes to the packet delivery ratio, basic gossip routing yields the worst performance by degrading the packet delivery ratio up to 17% compared with RELREC in the worst case, whereas RGR ensures a similar performance to that of RELREC. It is conspicuous that the basic gossip achieves the reduction in the control overhead at the sacrifice of packet delivery ratio, whereas RGR reduces the control overhead to some degree, and at the same time ensures a satisfying packet delivery ratio. Besides, RGR also outperforms basic gossip routing in terms of average time-to-time end delay and average route length.

CHAPTER 5
CONCLUSIONS AND FUTURE WORKS

5.1 Conclusions

In this thesis, the pro-reactive route recovery algorithm RELREC in ad hoc networks has been proposed to reduce the time delay and control overhead in route recovery processes. We firstly defined the concept of *Relay Node,* which is the common neighbor of the upstream node and the downstream node of a link. In existing works, the upstream node of a link is responsible for detecting the potential link failure. As a contract, the *Relay Node*s are used to detect the potential link failure in RELREC by overhearing the transmission at the neighboring links. The judgment based on overhearing of retransmission from the upstream node helps to avoid false detection; therefore RELREC out-performs the proactive recovery methods where the detection of link failure is based on received signal strength. In RELREC, one or more substitute routes via the *Relay Node*s are usually ready before the link break is confirmed, which is the proactive characteristic of this proposal. The upstream node of the potential broken link continues the retransmission even upon receiving the *NOTICE* packet from the *Relay Node*. The substitute routes are used to recover the broken route only after the confirmation by the upstream node that the link has really broken, which is the reactive characteristic. Since a substitute route is immediately available after the link failure is confirmed, this scheme causes no extra time delay in the route recovery process. There is no broadcast of any control packet in the network, thus the control overhead being reduced. The simulation study in ns-2 simulator has demonstrated clearly that our proposed RELREC scheme effectively reduces the average end-to-end time delay by up to 49% and 31%, and the normalized control overhead by up to 17% and 28% compared with the previous algorithms AODV and DRRS respectively in the best cases. According to the simulation results, the data delivery ratio by our proposed scheme is close to AODV and DRRS. The simulation results confirmed that our proposal can provide quick and low-cost route recovery without degrading the packet delivery ratio, while it is adaptive to node mobility and traffic load.

However, the route recovered by RELREC is usually longer than the original one. In fact, route redundancy occurs frequently in ad hoc networks due to the highly dynamic topologies. In order to optimize the route length in ad hoc networks, we proposed two automatic route shortening algorithms: relay shortening and active shortening. While relay shortening can only be initiated by the *Relay Node*s in pro-reactive relay recovery processes, the active shortening can be launched by any node whenever it overhears a shorter route from their neighbors. The simulation results indicated that active shortening works more effectively than relay shortening, leading to by far the shortest average route length and the smallest end-to-end time delay among all the algorithms in the simulation. However, the slightly larger control overhead traffic and relatively lower packet delivery ratio harms the performance of active route shortening.

Note that the information in *Relay Table*s is only utilized in route recovery processes in RELREC. In order to utilize the *Relay Table* information to the possible extent to further reduce control overhead, we extended the usage of *Relay Table* to route discovery process by incorporating RELREC into gossip algorithm, and proposed the Relay Gossip Routing (RGR). In RGR, only *Relay Node*s are allowed to rebroadcast under a gossiping probability, which differs from existing gossip methods. In order to find the appropriate gossiping probability for the *Relay Node*s, we firstly studied the relation between packet delivery ratio and gossiping probability. It is shown that the packet delivery ratio equals or even exceeds that of p = 1 when the value of p is larger than a certain point between [0.40, 0.50]; at the same time the total control overhead is smaller than that of p = 1. Accordingly, the gossiping probability is set to 0.5 in the performance evaluation of RGR. Simulation results confirmed that RGR successfully reduces the normalized control overhead by up to 17% in the best case compared to RELREC, and ensures a similar performance to RELREC in terms of the packet delivery ratio. Although the basic gossip routing yields the lowest normalized control overhead, it yields the worst performance in terms of the packet delivery ratio. It is conspicuous that RGR strikes a better balance between control overhead and packet delivery ratio than basic gossip routing does. Besides, RGR also outperforms basic gossip routing in terms of average time-to-time end delay and average route length.

In conclusion, the research described in this thesis effectively addresses the route failure and route redundancy problems in wireless ad hoc networks. The potential applications of the proposals described in this thesis include robot ad hoc networks, disaster rescue ad hoc networks, vehicular ad hoc networks, and so on.

5.2 Limitations of Current Works

The work presented in this thesis possesses a number of limitations that should be addressed in future studies.

- Energy consumption and security issues not considered.

  The proposed RELREC, two automatic route shortening algorithms, and RGR all require that nodes in the network operate under promiscuous overhearing mode. Although this prerequisite is indispensable for the execution of these proposals, it is somewhat energy consuming. Therefore, the proposals are not suitable for energy constraint environment such as sensor networks. Moreover, we assume that there is no selfish or malicious node in the network in the design of the proposals. In reality, however, selfish or misbehavior nodes often exist in ad hoc networks, and they may prevent the operation of many networking technologies and cause severe problems.

- Performance of automatic route shortening not satisfying.

  The simulation study on automatic route shortening indicates that the shortening routes may not be stable enough to ensure a satisfying packet delivery ratio. This is especially true in active route shortening, even though it significantly reduces the time delay, which may be a benefit from the shorter average route length.

- Intuitive design of RGR.

  The current proposal of RGR is simply a combination of RELREC and gossip routing. It is preferable to improve RGR with more original and complex algorithms that are adaptive to the dynamic topology in wireless ad hoc networks.

- Simulation based performance evaluation.

  The performance of the proposed algorithms and methods are examined based on simulation study in NS-2 simulator. Simulation based tests are fast, repeatable, easy to configure and customize, and more economical than emulation or physical implementation. However, the creditability of the simulation depends heavily on the quality and accuracy of the simulation model used. Deficiencies have been observed in the usage of NS-2 simulator, e.g. the random number generation. Consequently, simulation result may not entirely reflect the performance of the developed algorithms in reality.

- Extension and evaluation on other mobility models not performed.

  It is expected that the proposed RELREC may works more efficiently in VANET. However, we did not take action to extend the current RELREC to VANET, though we did some initial work trying to use the high-way model in NS-3 to evaluate the performance of RELREC in VANET system on high ways.

- Investigation into the integration of the proposals with applications/services not performed.

  In this thesis, the performance of the proposals developed is validated with network level metrics (e.g. packet delivery ratio, end-to-end time delay, normalized control overhead, and route length). An investigation into the performance of the proposed algorithms when they are integrated with specific applications and services remains.

5.3 Suggestions for Future Works

There are mainly three aspects for the improvement and extension of this research and they are considered as the future work.

- Improving the packet delivery ratio in the automatic route shortening.

  We have been working on improving the packet delivery ratio in automatic route shortening based on geographic information, which we call Geographic Automatic Route Shortening (GARS). It has been proven in simulation study that GARS successfully improves the packet delivery ratio, as well as reduce the control overhead and end-to-end time delay. Currently we are working on improving the accuracy of the shortening executions in GARS. The basic idea is to classify the potential shortening into preferred ones and non-preferred ones using SVM (Supportive Vector Machine), and to execute only the preferred shortening.

- Improving the rational of RGR.

  Being intuitive and straightforward, the current RGR is merely a combination of the proposed RELREC and existing gossip routing, though it has been proven to be able to further improve the performance compared with RELREC in terms of control overhead. The gossiping probability in RGR is a constant defined beforehand. In the future work, parameters reflecting the network status and topology should be used to decide the gossiping probability dynamically.

- Improving the integration of the work presented in this thesis.

  All the proposed algorithms in this thesis need to be integrated into one wholesome routing protocol, which is able to realize all the functions required for a routing protocol.

APPENDIX

A. PRELIMINARIES OF AD HOC NETWORKS

A.1 Fundamental of Ad Hoc Networks

A Wireless ad hoc network is a self-organized wireless network without the aid of any fixed infrastructure. Neighbor nodes communicate directly with each other over wireless channels. Each mobile node in the network also functions as a router to deliver the packets for other nodes to their destinations, thereby extending the diameter of the network through multi-hop relay mechanism. In situations where networks are constructed and destructed in ad-hoc manner, mobile ad-hoc networking is an excellent choice.

The initial development of ad hoc networks was primarily driven by military applications, where rapid network formation and survivability are key requirements. Relying on a system centralized around base stations is simply not an option because the base stations must first be deployed in the correct location (almost impossible in a hostile environment) and the network is subject to failure if one or several base stations are destroyed. On the other hand, the distributed network architecture of ad hoc networks, with all nodes having equal responsibility and using broadcast radio, is ideally suited to the military requirements for quickly establishing temporary communication among a group of soldiers in enemy territories or inhospitable terrains. To overcome the limited radio transmission ranges (i.e. not all nodes are within the range of every other node), nodes are equipped with the ability to forward information on behalf of others, i.e. multi-hop communications. The US Department of Defense, in particular DARPA [1], played a key role in the development of, and hence fostering research in ad hoc networks, with the Packet Radio Network (PRnet) [43] being deployed in 1972, followed by an updated network, Survivable Radio Network (SURAN), developed in 1983 and several ad hoc networks developed under the Global Mobile (GloMo) Information Systems program in 1994. Recent demonstration and production networks include the US Army Tactical Internet (TI) in 1997 and the Extending the Littoral Battlespace Advanced Concept Technology Demonstration (ELB ACTD) used by the US Marines in 1999. In Australia, DSTO [44] has led major research and development projects, in conjunction with local universities, on ad hoc networks, including the Packet Structures Research for Radios project in 1991, Self-Organising and Adaptive Links and Networks in 1993 and, more recently, Military Ad Hoc Wireless Networks in 2001 and Routing in Military Ad Hoc Networks 2003.

On commercial level, the need for ubiquitous networking is rising as the capacity of mobile computers increases steadily and new portable devices are commercialized. In daily scenario, ad hoc networks between laptop or palmtop or iPod could be used to spread and share information among people at a certain place, e.g. a conference or lecture. Another example is the rescue operation in remote areas where deploy an infrastructure communication system becomes impossible. In this case, self-configured ad hoc networks overcome the shortcoming of traditional communication system by eliminating the tedious need for base station and cables and facilitate to deploy a temporary wireless communication network quickly. Companies such as Motorola [45], Green Packet [46], PacketHop [47] and Firetide [48] are offering products and solutions based on ad hoc networking technology, with applications such as Law Enforcement, Intelligent Transport Systems, Community Networking and Home Networks in mind. However, the commercial technology available today is still a long way from the full potential of ad hoc networks. Fundamental problems must still be solved before ad hoc networks can fully enable a ubiquitous computing and communications environment. The main characteristics of ad hoc networks [49] include:

- Dynamic topologies. Nodes in mobile ad-hoc network are highly mobile which causes network topology to change rapidly and unpredictably, leading to unstable connectivity among the hosts. So some theories that are used in fixed network cannot be extended directly to ad hoc networks.
- Bandwidth-constrained and variable capacity links.
- Limited physical security.
- Distributed management. The control and management of mobile ad hoc network is distributed among the participating nodes. Each node is responsible to forward packet to other nodes in the networks. The nodes are also collaborate themselves to implement network routine functions such as security.

A key research challenge in ad hoc networks is to increase the efficiency of information transfer, while handling the harsh environmental conditions such as energy constrained and highly mobile devices. Advances in wireless communications technology are required to overcome the limitations inherent of broadcast radio networks. In addition, routing and transport protocols (e.g. TCP/IP) must be made more intelligent such that communication paths avoid nodes low on resources (e.g. low battery power).

A second challenge is enhancing the usability of ad hoc networks to support future commercial applications. With no prior configuration of network services, nor any central authority, basic tasks expected of a computer communications network become more complicated. Securing the network is perhaps the most difficult task. Supporting interactive voice and video applications will only be possible if some control of service quality is available. Finally, it is necessary to develop middleware services that hide the complexities of the ad hoc network from application programmers.

A.2 Routing in Ad Hoc Networks

Routing is the process of finding a path over which the packet will be sent across a network from a source to a destination. A routing protocol serves to exchange the route information; find feasible paths to a certain destination and select a path based on criteria such as length of the route, power consumption and lifetime of the wireless link; gather information about path breaks and mend the broke links. It has been clear that routing in mobile ad-hoc network is different than traditional routing in a fixed network. Routing in mobile ad hoc networking depends on many factors which include topology, selection of routers, initiation of request and available bandwidth. The characteristics of ad hoc network such as mobility and resource constraints impose great challenges to a routing protocol. The major requirements [50] of a fundamental routing protocol without QoS concern in ad hoc networks are as follows:

- Minimum route acquisition delay to transmit packets to a particular destination.
- Quick route reconfiguration to handle path break and subsequent packet losses.
- Loop-free to avoid unnecessary wastage of network bandwidth.
- Minimum control overhead to save precious bandwidth and avoid collision with data packets.

- High scalability to adapt the routing protocol to the network size.

An ideal routing protocol for ad hoc wireless networks should be able to address the challenges described above.

Routing protocols for ad hoc wireless networks can be classified into several types based on different criteria. Based on routing information update mechanism, all the research work in multi-hop routing protocols falls into three main categories: proactive routing, reactive routing, and hybrid routing that is a combination of the first two categories.

Proactive routing is also named table-driven routing. In this type of protocols, every node maintains the global network topology information with the form of a routing table by periodically updating routing information. Proactive routing can be subdivided into two further categories: flat and hierarchical. The Wireless Routing Protocol (WRP) [51] and Destination Sequenced Distance-Vector routing protocol (DSDV) [52] belong to the flat proactive routing. Dynamic Address Routing (DART) [53], L+ [54], Fisheye State Routing [55], LANMAR [56], Hierarchical State Routing (HSR) [57], Source-Tree Adaptive Routing Protocol (STAR) [58], Landmark [59], MMWN [60] and Cluster-head Gateway Switch Routing Protocol (CGSR) [61] are the protocols that belong to hierarchical proactive routing. Proactive routing is often criticized as power inefficient. Though it causes large control overhead in small scale ad hoc networks of only a few hundred nodes, it is expected to yields better performance over reactive routing in very large scale networks with hundreds of thousands of wireless nodes.

Reactive routing is also called on-demand routing. Instead of storing the global routing information of the whole network and update it periodically, protocols that falls into this category obtain the necessary path only when it is required. Some of the reactive routing protocols includes Dynamic Source Routing Protocol (DSR)[12], Ad Hoc On-Demand Distance Vector Routing Protocol (AODV)[21, 62], Flow-Oriented Routing Protocol (FORP)[63], Temporally Ordered Routing Algorithm (TORA)[64], Location aided Routing Protocol (LAR)[65], Associativity-Based Routing (ABR) [66], Signal Stability-Based Adaptive Routing Protocol (SSA)[67], and Lightweight Mobile Routing (LMR) [68]. Reactive routing is widely regarded as the technique choice for ad hoc networks, but the protocols all rely on some form of flooding to identify paths on demand, which may cause large control overhead in very large scale networks.

The hybrid routing protocol, which is a combination of proactive and reactive method, utilizes proactive routing within certain zone and reactive routing outside the zone. Thereby take advantages of the best features of the above two categories. Typical protocols in this categories are Core Extraction Distributed Ad Hoc Routing (CEDAR) [69], Zone-based Hierarchical Link States (ZHLS) routing protocol [70] and Zone Routing Protocol [71]. Although each category has its own advantage, reactive routing protocols are generally considered out-perform proactive routing protocols in that they serve to reduce overhead and save resources within the networks; while the hybrid protocols are not preferred due to their complexity in implementation. For these reasons, reactive routing protocols are often regarded as the first choice to operate in an ad hoc network.

B. IMPLEMENTATION OF RELREC IN NS-2

B.1 Basics of NS-2 simulator

The Network Simulator 2 (NS-2) [36-39, 57] is an object oriented simulator, written in C++, with an OTcl interpreter as a frontend. NS-2 adopted two languages, C++ and OTcl, to strike a trade-off between run-time speed and iteration time. In general, all network components are created, plugged and configured from Tcl. The wireless extension, derived from CMU Monarch Project [38], has two assumptions simplifying the physical world:

(1) Nodes do not move significantly over the length of the time they transmit or receive a packet. This assumption holds only for mobile nodes of high-rate and low-speed.

(2) Node velocity is insignificant compared to the speed of light, and the Doppler effects are not considered.

Figure shows the network components in the mobile node and the data path of sending and receiving packets.

In this appendix, we present the procedure of implementing RELREC in NS-2.34 wireless extension on Ubuntu 10.10. We implemented the proposed RELREC based on the underlying routing protocol AODV [13].

B.2 Data Structures

Besides the data structures inherited from AODV, we also defined a couple of new data structures works exclusively in RELREC. They include caches like Tap Cache, Relay Cache, Notice Cache, and Relay Table. All nodes in the network are required to maintain a Tap Cache, a Relay Cache, a Notice Cache, and a Relay Table to ensure the operation of RELREC. The two new-defined packets are NOTICE and CONFIRM. The former is used by the Relay Node to inform the upstream node of the potential link failure; the latter is used by the upstream node to trigger the recovery process after confirming the link failure.

B.2.1 Caches

The new caches defined in RELREC include Tap Cache, Relay Cache, Notice Cache, and Relay Table. All nodes in the network are required to maintain a Tap Cache, a Relay Cache, a Notice Cache, and a Relay Table to ensure the operation of RELREC.

Tap Cache

The Tap Cache is used by the tap() function to keep the overheard packets for a certain time duration in the overhearing node. When the overhearing node overhears a packet, it firstly compares the current overheard packet with the packets stored in Tap Cache. If the same packet is found, it means the sender is retransmitting the packet, indicating a potential link failure.

Each entry in the Tap Cache includes the following fields:

- prev_hop: the address of the previous hop of the packet.
- next_hop: the address of the next hop of the packet.
- src: the address of the source included in the packet.
- dst: the address of the destination included in the packet.

• index: the ID of the nodes who overhears this packet.

• uid: packet ID.

• tap_expire: the expire time of this entry. The maximum life time of each entry is 1s. A new kind of timer TapCacheTimer is defined to update a relay table every 1s.

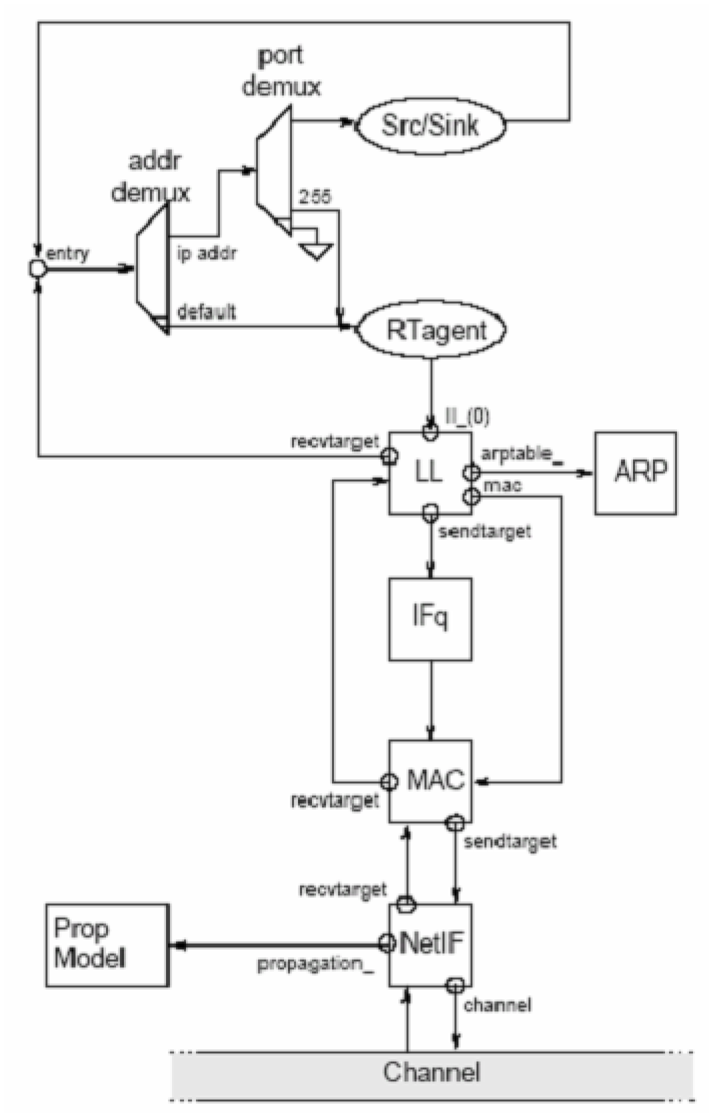


Figure B. 1 Schematic of a mobile node under the CMU Monarch wireless extension to NS-2.

Relay Cache

The Relay Cache is usedto store the information of the Relay Node. Each entry in the Relay Cache includes the following fields:

• prev_hop: the address of the upstream node of the link.

• next_hop: the address of the downstream node of the link.

• rel: the address of the possible Relay Node for the link.

• r_expire: the expire time of this entry. The maximum life time of each entry is 0.5s. A new kind of timer RelayCacheTimer is defined to update a relay table every 0.5s.

Notice Cache

When the upstream node of a potential broken link receives a NOTICE packet from Relay Node, it keeps the NOTICE in its Notice Caches for a certain time span. Each entry in the Notice Cache includes the following fields:

• upstrm: the address of the upstream node of the link.

• dnstrm: the address of the downstream node of the link.

• rel: the potential relay node.

• nt_expire: the expire time of this entry. The maximum life time of each entry is 10s. A new kind of timer NoticeCacheTimer is defined to update a relay table every 10s.

Relay Table(20100518)

Relay Table is used to cache the links for which this node can serve as Relay Node. Each entry in the Relay Table includes the following fields:

• upstrm: the address of the upstream node of the link.

• dnstrm: the address of the downstream node of the link.

• rl_expire: the expire time of this entry. The maximum life time of each entry is 10s. A new kind of timer RelayTableTimer is defined to update a relay table every 10s. (In the simulation, the lifetime of a route is either determined from RREP or initialized to ACTIVE_ROUTE_TIME_OUT(=3 seconds in NS2-2.34).)

B.2.2 Packets

NOTICE

The NOTICE packet is sent by the Relay Node when it overhears the retransmission from the upstream node of the broken link. A NOTICE packet includes the following fields:

• rel_next_hop_: the address of the downstream node of the broken link.

• pkt_uid_: the ID of the overheard packet which has been retransmitted.

• dst_: the destination of the packet.

CONFIRM

The CONFIRM packet is sent by the upstream node to trigger the route recovery. A CONFIRM packet includes the following fields:

• rel_next_hop_: the address of the downstream node of the broken link.

• *rt: the relay entry of the broken link.

B.3 Necessary Changes in NS-2.34

Following the instructions provided in [37], it is necessary to make some changes in order to implement the proposed RELREC protocol into NS-2.34. To allocate the code I will firstly create a new directory called relrec inside the NS-2.34 base directory. I created ten files there:

relrec.h This is the header file where all necessary timers and routing agents are defined

relrec.cc In this file all timers, routing agents, and Tcl hooks are implemented.

relrec_packet.h In this file all packets that needs to be exchanged among nodes are declared.

relrec_rqueue.h This is the header file where the packet caching queue is defined.

relrec_rqueue.cc In this file the packet caching queue is implemented.

relrec_rtable.h This is the header file where the routing table is declared.

relrec_rtable.cc In this file the route table is implemented.

relrec_tqueue.h This is the header file where the overheard packet caching queue is defined.

relrec_tqueue.cc In this file the overheard packet caching queue is implemented.

rl_table.h This is the header file where the Relay Table is defined.

rl_table.cc In this file the Relay Table is implemented

Now the “physical” structure is ready, and next step is to create a new protocol agent by inheriting from **Agent** class, implement the timers inheriting from **Timer** class, implement corresponding trace function inhering from **Trace** class, and so on. After the implementation, we need to make some changes in NS-2.34 in order to integrate the code inside the simulator.

B.3.1 Packet Type Declaration

We have to use a constant to indicate the new packet type PT_RELREC, which is defined inside file ns-2.34/common/packet.h. Find the packet_t enumeration, where all packet types are listed. We add PT_RELREC to this list as is shown in the following piece of code (line 6).

```
/* ns-2.34/Common/packet.h */

1:  enum packet_t {
2:          PT_TCP,
3:          PT_UDP,
4:          PT_CBR,
5:          /* … much more packet types… */
6:          PT_RELREC,
7:          PT_NTYPE // This MUST be the LAST one
8:  };
```

Just below in the same file there is definition of p_info **c**lass. Inside constructor we will provide a textual name for our packet type (line 6).

```
/* ns-2.34/Common/packet.h */

1:  p_info() {
2:          name_[PT_TCP] = “tcp”;
3:          name_[PT_UDP] = “udp”;
4:          name_[PT_CBR] = “cbr”;
5:      /* … much more names… */
6:          name_[PT_RELREC] = “RELREC”;
7:  }
```

B.3.2 Tracing Support

The trace files describe all the event happened during execution. The explanation of the structure of trace files can be found in chapter 23 [72]. To log information regarding the RELREC packet type we implement the format_relrec() function inside the CMUTrace class which is described in chapter 16 [72]. We first add the following code into ns-2.34/trace/cmu-trace.h:

```
/* ns-2.34/trace/cmu-trace.h */

class CMUTrace : public Trace {
  /* … definitions … */
private :
  /* … */
     void format_aodv(Packet *p, int offset);
     void format_relrec(Packet *p, int offset);
};
```

Add the following code into ns-2.34/trace/cmu-trace.cc.

```
/* ns-2.34/trace/cmu-trace.cc */

#include <relrec/relrec_pkt.h>

 /* … */

void
CMUTrace :: format_relrec (Packet *p, int offset)
{
  /* implementations */
};
```

In order to call this recently created function, we must change the format() in ns-2.34/trace/cmu-trace.cc.

```
/* ns-2.34/trace/cmu-trace.cc */

void
CMUTrace :: format(Packet *p, const char *why)
{
      /* … */
      case PT_PING:
              break;

      case PT_RELREC:
              format_relrec(p, offset);
              break;

      default:
      /* … */
 }
```

B.3.3 Tcl Library

We also need to do some changes in Tcl files, including adding the RELREC packet type, giving default values for binded attributes and providing the needed infrastructure to create wireless nodes running our RELREC.

RELREC has to be added into the list shown below in ns-2.34/tcl/lib/ns-packet.tcl (line 2).

```
/* ns-2.34/tcl/lib/ns-packet.tcl */

foreach prot {
       RELREC
       AODV
       ARP
     # …
       NV
} {
       add-packet-header $prot
}
```

Default values for binded attributes have to be given inside ns-2.34/tcl/lib/ns-default.tcl. The following code must be put at the end of the file.

```
/* ns-2.34/tcl/lib/ns-default.tcl */

# …
# Defaults defined for RELREC
Agent/RELREC set accessible_var_ true
```

We also need to add procedures in ns-2.34/tcl/lib/ns-lib.tcl for creating a node (instance) with RELREC as routing protocol.

```
/* ns-2.34/tcl/lib/ns-lib.tcl */

Simulator instproc create-wireless-node args {
     # …
       switch -exact $routingAgent_ {
                   relrec {
                               set  ragent  [$self  create-relrec-agent  $node]
                   }
                   # …
     }
       # …
  }
```

Then create-relrec-agent will be coded as shown below.

```
/* ns-2.34/tcl/lib/ns-lib.tcl */

1:  Simulator instproc create-relrec-agent  {  node  }  {
2:         # Create relrec routing agent
3:         set ragent [new Agent/t [$node  node-addr] ]
4:         $self at 0.0 “$ragent start”
5:         $node set ragent_ $ragent
6:       return $ragent
7:  }
```

Line 3 creates a new RELREC agent with the node’s address, which is scheduled to start at the beginning of the simulation (line 4), and is assigned as the node’s routing agent in line 5.

B.3.4 Priority Queue

We need to tell the PriQueue class that relrec packets are routing packets and therefore treated as high priority, so we must modify the recv() function in ns-2.34/queue/priqueue.cc file as follows:

```
/* ns-2.34/queue/priqueue.cc*/

void
PriQueue :: recv (Packet *p, Handler *h)p
{
        struct hdr_cmn *ch = HDR_CMN();

        if (Prefer_Routing_Protocols)  {

                switch (ch->ptype()) {
                        case PT_DSR:
                        case PT_MESSAGE:
                        case PT_TORA:
                        case PT_AODV:
                        case PT_RELREC:
                                recvHighPriority(p,h);
                                break;

                        default:
                                Queue:: recv(p, h);
                }
         }
        else {
                Queue:: recv (p, h);
        }
 }
```

B.3.5 Makefile

We edit the Makefile file by adding our object files inside OBJ_CC variable as in the following code.

```
/* ns-2.34/Makefile */

 OBJ_CC = ¥
tools/random.o tools/rng.o tools/ranvar.o common/misc.o common/timer-handler.o ¥
        …
relrec/relrec.o relrec/relrec_rtable.o relrec/relrec_logs.o ¥
relrec/relrec_rqueue.o relrec/rl_table.o ¥
        …
        $(OBJ_STL)
```

After modify all the files, use the following commands to compile NS-2:

```
$ touch common/packet.cc
$ make clean all
$ sudo make install
```

B.4 Node Configuration

The components on a single nodes from bottom to up are [73]:

- Channel: class WirelessChannel, file ns-2.34/mac/channel.{h,cc}, inheriting from class Channel in the same folder.
- Network Interface (netif): class WirelessPhy, files ns-2.34/mac/wireless-phyf.{h,.cc}, inheriting from class Phy in files ns2/mac/phyf.{h,.cc}.
- Propagation Model: a composed component of network interface, class Propagation and MobileNode.
- Media Access Control (MAC): class Mac802_11 inheriting from the abstract class MAC.
- Outgoing Queue: queue has only one target (down-target), mac; it is not the up-target of mac. Class Queue has many variations such as droptail, priqueue, etc.
- Link layer: class LL(inherits from LinkDelay).It has a composed component ARP which works as ARP procedure, mapping the protocol address (such as IP address) to Hardware address (such as MAC address).
- Network layer: routing agents with many variations such as DSDV, DSR, AODV, etc.

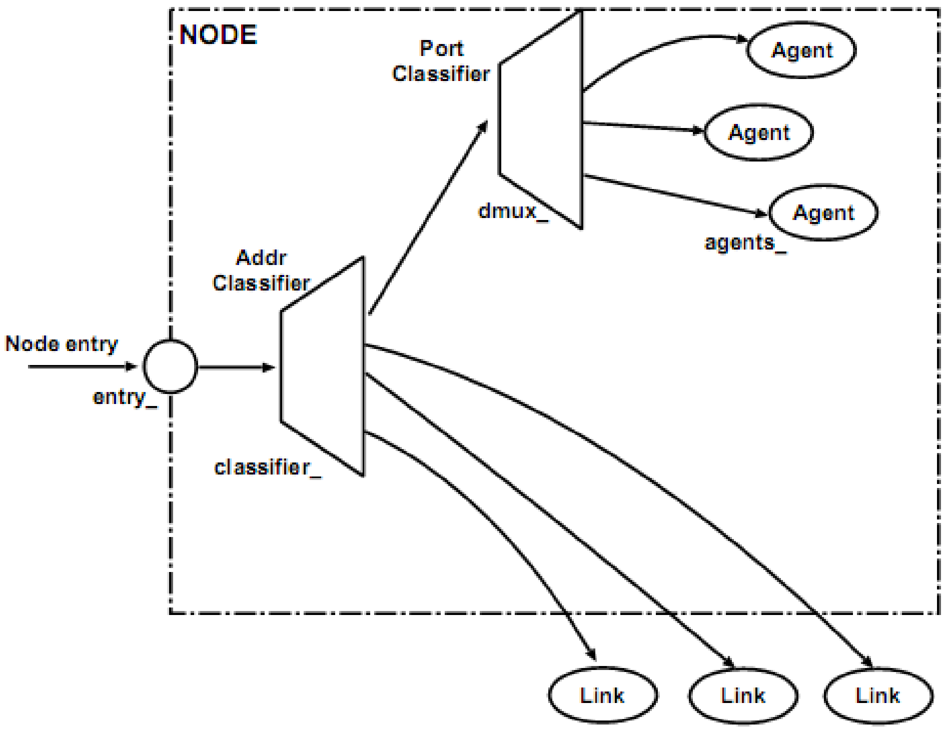


Figure B. 2 Structure of a unicast node in NS-2 [57].

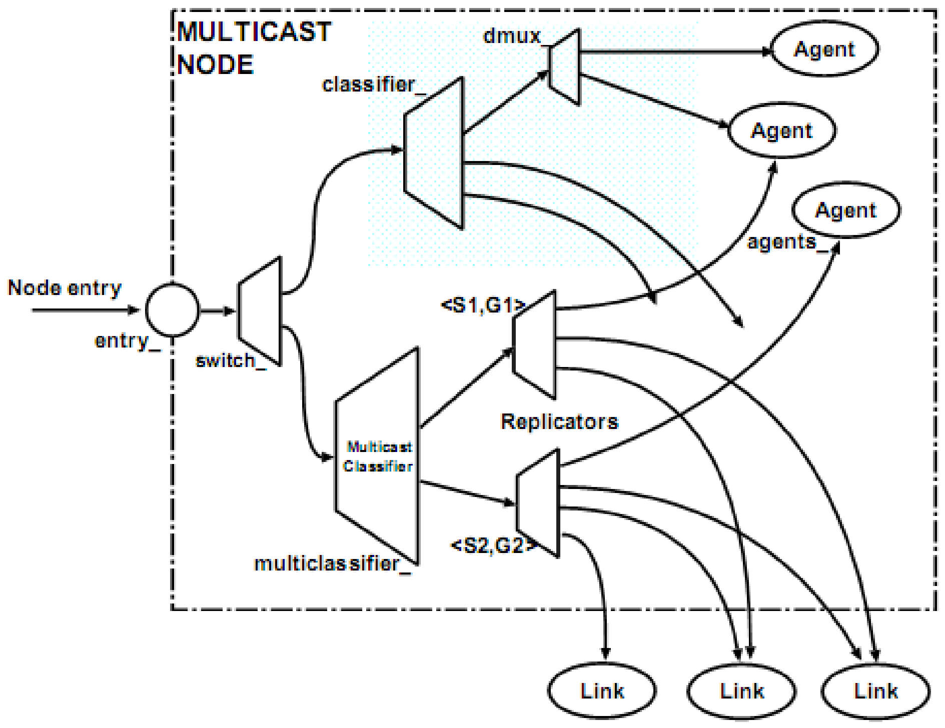


Figure B. 3 Structure of a multicast node in NS-2 [72].

Node configuration essentially consists of defining node characteristics before creating them, including type of addressing structure, the network components for mobile nodes, turning on or off the trace at Agent/Router/MAC levels, selecting the type of ad hoc routing protocol or defining their energy model.

```
$ns node-config     -addressType    flat     //Type of addressing structure
                    -adhocRouting     RELREC     //Ad hoc routing protocol
                    -llType      LL     //Link layer type
                    -macType     Mac/802 11 //MAC layer protocol
                    -propType    Propagation/FreeSpace  //Radio-propagationmodel
                     -ifqType Queue       /DropTail/PriQueue    //Interface queue type
                    -ifqLen      50     //Interface queue length
                    -antType     Antenna/OmniAntenna    //Antenna type
                    -channel     Channel/WirelessChannel               //Channel type
                    -phyType     Phy/WirelessPhy   //Network interface type
                    -topoInstance      [new Topography]         //New topology
                    -agentTrace        ON    //Trace on agent level
                    -routerTrace       ON    //Trace on router level
                    -macTrace          OFF   //No trace on MAC level
                    -movementTrace     OFF   //No movement trace
```

The address of an agent in a node is 16 bits: the higher 8 bits define the node id_; the lower 8 bits identify the individual agent at the node. This limits the number of nodes in a simulation to 256 nodes. If we need to create a topology with more than 256 nodes, we should first expand the address space before creating any nodes:

```
Node expandaddr
```

This expands the address space to 30 bits, and the higher 22 bits are used to assign node numbers. Also, nodes in NS-2 are constructed for unicast simulations by default. In order to create nodes for multicast simulation, we should use the following command to set EnableMcast_ to 1:

```
Simulator set EnableMcast_ 1
```

The structure of a unicast node and a multicast node are shown in Figure B. 2 and Figure B. 3 respectively.

After node configuration we need to create mobile nodes in the network:

```
1:  for {set i 0} {$i <50} {incr i}{ set node ($j) [$ns node]
2:     $node ($i) random-motion 0 }              //Create 50 nodes without random motion.
```

B.5 Physical Layer Configuration

B.5.1 Radio Propagation Model

There are three propagation models in ns-2: the free space model, the two ray ground model, and the shadowing model, all of which are introduced in this session.

Free space model

This model assumes the ideal propagation condition where only one clear line-of-sight path exists between the transceiver pair. It basically represents the communication range as a circle around the transmitter. If a receiver is within the circle, it receives all packets; otherwise, it loses all. The transmitted signal power can be calculated as follows [74]:

$$P_r(d) = \frac{P_t G_t G_r \lambda^2}{(4\pi)^2 d^2 L} \qquad (7)$$

where Gt and Gr are the antenna gains of the transmitter and the receiver respectively. λ is the wavelength, and L ($L \geq 1$) is the system loss. It is common to select Gt =Gr =1 and L=1in ns-2 simulator. The OTcl interface for utilizing a propagation model is the nodeconfig command. In order to use free space model, there are two ways:

```
$ns node-config -propType Propagation/FreeSpace
```

or

```
set prop [new Propagation/FreeSpace]
$ns node-config -propInstance $prop
```

The corresponding source code file is ns-2.34/mobile/propagation.cc.

Two-ray ground model

This model considers both the direct path and a ground reflection path. It also represents the communication range as an ideal circle. It gives more accurate prediction at a long distance than the free space model. The received power at distance d is predicted [75] by

$$P_r(d) = \frac{P_t G_t G_r h_r^2 h_t^2}{d^4 L} \qquad (8)$$

where $h_t$and h_r are the heights of the transmitting and receiving antennas. The above equation shows a faster power loss than the equation of free space model as distance increases. However, it does not give a good result for a short distance due to the oscillation caused by the constructive and destructive combination of the two rays.

In order to use two-ray ground reflection model, there are two ways:

```
$ns node-config -propType Propagation/TwoRayGround
```

or

```
set prop [new Propagation/TwoRayGround]
$ns node-config -propInstance $prop
```

The corresponding source code file is ns-2.34/mobile/tworayground.cc.

The above two models predict the mean received power at distance d. In real situation, the received power at certain distance is a random variable due to multipath propagation effect (or fading effect). Correspondently, a more complex model, the shadowing model was proposed.

Shadowing model

This model consists of two parts. The first one is known as path loss model, which also predicts the mean received power at distance d, as is shown in the first equation below; the second part is a log-normal random variable which reflects the variation of the received power at certain distance. The overall shadowing model is represented by the equation (10).

$$[\frac{P_r(d)}{P_r(d_0)}]_{dB} = -10\beta \log\left(\frac{d}{d_0}\right) \quad (9)$$

$$[\frac{P_r(d)}{P_r(d_0)}]_{dB} = -10\beta \log\left(\frac{d}{d_0}\right) + X_{dB} \quad (10)$$

where d_0 is a reference close-in distance. β is called the path loss exponent and is usually empirically determined by field measurement. $X_{dB}$is a Gaussian random variable with zero mean and standard deviation σ_{dB} which is called shadowing deviation and is also obtained by measurement.

In order to use free space model, there are two ways:

```
#first set values of shadowing model
Propagation/Shadowing set pathlossExp_ 2.0
Propagation/Shadowing set std_db_ 4.0
Propagation/Shadowing set dist0_ 1.0
Propagation/Shadowing set seed_ 0
$ns node-config -propType Propagation/Shadowing
```

or

```
set prop [new Propagation/Shadowing]
$prop set pathlossExp_ 2.0
$prop set std_db_ 4.0
$prop set dist0_ 1.0
$prop set seed-type 0
$ns node-config -propInstance $prop
```

The seed-type above can be raw, predef or heuristic. The corresponding source code file is ns-2.34/mobile/shadowing.cc.

In our simulation, we use the two-ray ground model to achieve satisfying accuracy without including two complicated parameters.

B.5.2 Transmission Range

In NS-2 simulator the transmission range of a node is decided by two parameters: transmit power P_t and receiving threshold RXThresh_. The default value of transmission range in NS-2 is 250m with P_t = 0.28183815 and RXThresh_ = 3.652e-10, which is considered too long in terms of the moving area of our simulation. Therefore we changed the transmission range to 100m according to the following steps.

Step 1: Compile ns-2.34/indep-utils/propagation/threshold.cc to get the value of RXThresh_.

```
$ g++ threshold.cc -o threshold
$ ./threshold -m TwoRayGround 100
```

We get the following information:

```
distance = 100
propagation model: TwoRayGround

Selected parameters:
transmit power: 0.281838
frequency: 9.14e+08
transmit antenna gain: 1
receive antenna gain: 1
system loss: 1
transmit antenna height: 1.5
receive antenna height: 1.5

Receiving threshold RXThresh_ is: 1.42681e-08
```

This means if we keep the P_t to be the default value 0.28183815, we should set RXThresh_ to 1.42681-08 to achieve 100m transmission range. Another way is to keep RXThresh_ to the default value 3.652e-10 while changing P_t to 7.214e-3.

Step 2: Necessary change in ns-2.34/tcl/lib/ns-default.tcl.

```
Phy/WirelessPhy set RXThresh 1.42681-08
```

B.6 MAC Layer Configuration

B.6.1 Promiscuous Overhearing Mode

In default the promiscuous overhearing mode is not implemented in NS-2. We can enable the overhearing mode in NS-2 by adding a tap() function in ns-2.34/mac/mac-802_11.cc. In default, tap() function only overhearing data packet. Usually we can get the address information of previous hop in the common header [ch → prev_hop] of a packet; however, the debugging information shows that the previous hop in the common header of overheard packets are always 0. Instead of using common header, we use MAC header to get address information of previous hop [ETHER_ADDR(mh→dh_ta)] and next hop [ETHER_ADDR(mh→dh_ra)] of a packet. Debugging information shows the address information in MAC header is correct and overhearing of data packet works well.

B.6.2 Disabling RTS/CTS

In our current implementation we use IEEE 802.11 without RTS/CTS exchange on the MAC layer, though the proposal can be easily extended to situations with RTS/CTS through minor adjustment. The switch of RTS/CTS exchange is controlled by the parameter RTSThreshold_ in ns-default.tcl. Simply Use the following command:

```
Mac/802_11 set RTSThresholdd_ 3000
```

This means that an RTS will only be sent for packets that are bigger than 3000 bytes, which should be never. The RTS/CTS is enabled if the value of RTSThreshold_ is zero.

B.7 Generating Wireless Scenario

In order to generate the wireless scenario, we need to create two files: the traffic pattern file and the node movement file.

B.7.1 Traffic Pattern File

The traffic pattern file defines the Random TCP or CBR traffic connections between mobile nodes in the network. It can be setup using a traffic-scenario generator script ns-2.34/indep-utils/cmu-scen-gen/cbrgen.tcl. In order to create a traffic pattern file, we need to define the type of traffic connection (CBR or TCP), the number of nodes and maximum number of connections to be setup between them, a random seed and incase of CBR connections, a rate whose inverse value is used to compute the interval time between the CBR packets. So the command line looks like the following:

```
ns cbrgen.tcl [-type cbr|tcp] [-nn nodes] [-seed seed] [-mc connections][-rate rate] >cbr-nodes-connections-seed-rate
```

In our simulation, we use CBR connections among 200 mobile nodes. Suppose we need a traffic pattern file with 2 random CBR connections among 200 nodes and 4 packets are generated per second, we can generate the file by using the command below:

```
ns cbrgen.tcl –type cbr –nn 200 –seed 1 –mc 2 –rate 4.0 > cbr-150-2-1-4.0
```

A typical generated CBR traffic file looks like:

```
#
# 2 connecting to 3 at time 82.557023746220864
#
set udp_(0) [new Agent/UDP]
$ns_ attach-agent $node_(2) $udp_(0)
set null_(0) [new Agent/Null]
$ns_ attach-agent $node_(3) $null_(0)
set cbr_(0) [new Application/Traffic/CBR]
$cbr_(0) set packetSize_ 512
$cbr_(0) set interval_ 0.25
$cbr_(0) set random_ 1
$cbr_(0) set maxpkts_ 10000
$cbr_(0) attach-agent $udp_(0)
$ns_ connect $udp_(0) $null_(0)
$ns_ at 82.557023746220864 "$cbr_(0) start"
......
```

```
#
#Total sources/connections: 4/6
#
```

B.7.2 Node Movement File

The node movement file specifies the positions of nodes and their moving speed and moving directions under random waypoint model. We can generate the movement file with large number of nodes using the tool ns-2.34/indep-utils/cmu-scen-gen/setdest. The syntax is:

```
./setdest -v 2 -n $numnodes –m $minspeed -M $maxspeed  -t $simtime -p $pausetime -x
$maxx -y $maxy > scen-maxx*maxy-numnodes-minspeed-maxspeed-pausetime-simtime
```

For example, we need to generate a network with 200 nodes moving in a square of 1200m×1200m. The minimum speed is 1m/s and the maximum speed is 5m/s. The pause time for all nodes are 20s, and the simulation last 1000s. We can generate the file using the command below:

```
./setdest -v 2 -n 200 –m 1 -M 20 -t 1000 -p 20 -x 1200 -y 1200 > scen-1200x1200-200-1-
5-20-1000
```

The file scen-1200x1200-200-1-5-20-1000 specifies a 1200x1200 topology with 200 nodes random distributed labeled by XY-coordinates. After the initial position information, the nodes are specified with their movement destination and speed. The initial distance (hop counts) information is counted by a GOD. Currently, the god object is used only to store an array of the shortest number of hops required to reach from one node to another. The god object does not calculate this on the fly during simulation runs, since it can be quite time consuming. The information is loaded into the god object from the movement file.

The nodes are moving during this 1000-second simulation scenario, and the distance (hop-counts) information changing accordingly. The god information should not be available to any of the node. Therefore a routing protocol has to discover the distance by itself with some mechanism.

It is possible that a node reaches its destination before the simulation timer ends. Thus, it needs to re-specify a new direction and speed for it. Also, the average pause time is a parameter to allow a node stop to move in a destination before moving again. A segment of the generated node movement file is shown below.

```
#
# nodes: 200, speed type 1, min speed 1.00, max speed: 5.00
# avg speed: 2.30, pause type: 1, pause: 20.00, max x: 1200.00, max y: 1200.00
#
$node_(0) set X_ 1472.239335818631
$node_(0) set Y_ 218.102475038104
$node_(0) set Z_ 0.000000000000
$node_(1) set X_ 266.329091949493
$node_(1) set Y_ 188.988756134562
$node_(1) set Z_ 0.000000000000
.....
$ns_ at 0.000000000000 "$node_(0) setdest 298.258250355189 34.553480936942
5.763540789209"
$ns_ at 0.000000000000 "$node_(1) setdest 783.793876303622 280.181784294036
16.832727003192"
$ns_ at 0.000000000000 "$node_(2) setdest 978.836090908830 151.46706034549
19.354908025316"
```

```
$ns_ at 0.000000000000 "$node_(3) setdest 376.126888262665 139.747808400435
11.497476206039"
$ns_ at 0.000000000000 "$node_(4) setdest 370.541780738321 185.032792269909
8.002013555825"
$ns_ at 0.000000000000 "$node_(5) setdest 1463.652810766312 132.618506062927
17.607998988233"
$ns_ at 0.000000000000 "$node_(6) setdest 1020.046295404814 40.982731716802
15.434148765705"
......
$god_ set-dist 0 1 6
$god_ set-dist 0 2 1
$god_ set-dist 0 3 4
$god_ set-dist 0 4 2
$god_ set-dist 0 5 4
$god_ set-dist 0 6 8
$god_ set-dist 0 7 6
......
$ns_ at 0.141920608238 "$god_ set-dist 17 24 2"
$ns_ at 0.165720133690 "$god_ set-dist 3 15 1"
$ns_ at 0.263269271549 "$god_ set-dist 1 46 2"
$ns_ at 0.263269271549 "$god_ set-dist 7 46 2"
$ns_ at 0.263269271549 "$god_ set-dist 12 46 1"
$ns_ at 0.263269271549 "$god_ set-dist 25 46 3"
$ns_ at 0.263269271549 "$god_ set-dist 46 47 3"
$ns_ at 0.314230786054 "$god_ set-dist 5 42 5"
......
$ns_ at 12.602131706040 "$god_ set-dist 16 36 2"
$ns_ at 12.623221733243 "$node_(12) setdest 424.679916546080 186.581540202946
5.803287072825"
$ns_ at 12.682532746855 "$god_ set-dist 18 40 3"
....
$ns_ at 899.907548331028 "$god_ set-dist 32 40 1"
$ns_ at 899.932142983319 "$god_ set-dist 22 40 1"
$ns_ at 899.991403308054 "$god_ set-dist 3 29 1"
#
# Destination Unreachables: 8228
#
# Route Changes: 225817
#
# Link Changes: 15494
#
# Node | Route Changes | Link Changes
#     0 |          2709 |          433
#     1 |          3098 |          467
#     2 |          2651 |          496
#     3 |          3398 |   642Setdest
#     4 |          3036 |          548
#     5 |          2755 |          252
#     6 |          3471 |          601
```

REFERENCES

[1] Defense Advanced Research Projects Agency. http://www.darpa.mil/

[2] “Raytheon to develop network centric radio systems for DARPA,” Defense Industry Daily, July 20, 2009

[3] K. Kanchanasut, A. Tunpan, M. A. Awal, Dwjendra Kumar Das, Thirapon Wongaardsakul and Yasuo Tsuchimoto, "DUMBONET: A Multimedia Communication System for Collaborative Emergency Response Operation in Disaster-affected Areas", International Journal of Emergency Management (IJEM), vol.4, no.4, pp.670-681, 2007.

[4] R.-H. Cheng, T.-K Wu and C. W. Yu, "A highly topology adaptable ad hoc routing protocol with complementary preemptive link breaking avoidance and path shortening mechanisms," Wireless Network, vol. 16, pp. 1289-1311, 2010.

[5] J.-S. Liu and C.-H. R. Lin, “RBR: refinement-based route maintenance protocol in wireless ad hoc network,” Computer Communications, vol.28, pp. 908-920, 2005.

[6] T. Goff, N. A.-Ghazaleh, D. Phatak, and R. Kahvecioglu, “Pre-emptive routing in ad hoc networks,” Journal of Parallel and Distributed Computing, vol. 63, pp. 123–140, 2003.

[7] M. H. Jiang and R. H. Jan, "An efficient multiple paths routing protocol for ad hoc networks," in Proc. of 15th IEEE ICIN, 2001, p.544-549.

[8] Y. H. Wang, C. M. Chung and C. C. Chuang, "Ad hoc on-demand backup node setup routing protocol," in Proc. of 15th IEEE ICIN, 2001, p. 933-937.

[9] J. Jeon, K. Lee, and C. Kim, “Fast route recovery scheme for mobile ad hoc networks,” ICOIN 2011, p.419-423, Malaysia, January 2011.

[10] R. Yamamoto and T. Miyoshi, “Distributed retransmission method using neighbour terminals for ad hoc networks,” in Proc. APCC2008, 2008, SB 0083.

[11] T.-K. Wu, C. W. Yu and R. H. Cheng, “A low overhead dynamic route repairing mechanism for mobile ad hoc networks,” Computer Communication, vol. 30, pp. 1152-1163, 2007.

[12] RFC 4728–The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4. February 2007.

[13] J.-S. Youn, J.-H. Lee, D.-H. Sung and C.-H. Kang, “Quick local repair scheme using adaptive promiscuous mode in mobile ad hoc networks,” Journal of Networks, vol.1, no.1, pp.1-11, May 2006.

[14] S. Kim, and S. An, “A new routing protocol using route redundancy in ad hoc networks,” IEICE Transactions on Communication, vol. E88-B, no.3, p.1000-1008, March 2005.

[15] L. H. M. K. Costa and M. D. D. Amorim and S. Fdida, "Reducing latency and overhead of route repair with controlled flooding," Wireless Network, vol. 10, pp. 347-358, 2004.

[16] G. Liu, K. J. Wong, B. S. Lee, B. C. Seet, C. H. Foh, and L. Zhu, “PATCH: a novel recovery mechanism for mobile ad-hoc networks,” IEEE Vehicular Technology Conference, vol.5, pp.2995-2999, October 2003.

[17] R. Duggirala, R. Gupta, Q. A. Zeng, and D. P. Agrawal, “Performance enhancements of ad hoc networks with localized route repair, ” IEEE Transactions on Computers, vol.52, no.7, pp.854-861, July 2003.

[18] M. Sphohn and J. J. G.-L.-Aceves, "Neighbour aware source routing," in Proc. MobiHOC 2001, 2001, p. 11-21.

[19] S.-J. Lee and M. Gerla, “AODV-BR: Backup routing in ad hoc networks,” in Proc. of Wireless Communications and Networking Conference (WCNC), p.1311-1316, Chicago, IL, US, September 2000.

[20] Aron and S. Gupta., "A witness-aided routing protocol for mobile ad-hoc networks with unidirectional links," in Proc. 1st International Conference on Mobile Data Access, 1999, p. 24-33.

[21] C. E. Perkins and E. M. Royer, “Ad Hoc On-Demand Distance Vector Routing,” in Proc. of IEEE Workshop on Mobile Computing Systems and Applications, pp.90-100, 1999.

[22] H. Xiao, Y. Zhang, J. Malcolm, B. Christianson, and K. C. Chua, “Modelling and analysis of TCP performance in wireless multihop networks,” Wireless Sensor Network, vol.2, p. 493-503, 2010.

[23] V. Kawadia, and P. R. Kumar, “Experimental investigations into TCP performance over wireless multihop networks,” in Proc. of the 2005 ACM SIGCOMM workshop on experimental approaches to wireless network design and analysis (E-WIND ’05), Philadelphia, PA, USA, pp.22-25, 2005.

[24] M. Gerla, K. Tang, and R. Bagrodia, “TCP performance in wireless multi-hop networks,” in Proc. of the 2nd IEEE Workshop on Mobile Computer Systems and Applications (WMCSA ’99), Washington D.C., USA, pp. 41-50, Feb. 1999.

[25] M. Saito, H. Aida, Y. Tobe, and H. Tokuda, “Proximity-Based Dynamic Path Shortening Scheme for Ubiquitous Ad Hoc Networks,” in Proc. of the 24th International Conference on Distributed Computing Systems, IEEE Computer Society, 2004.

[26] X. Liu, G. Zhao, X. Ma, and M. Sun, “Path Shortening for Delivery Rate Enhancement in Geographical Routing via Channel Listening,” in Proc. of IEEE GLOBECOM 2006, pp.1-5, San Francisco, USA, 2006.

[27] C. Gui and P. Mohapatra, “SHORT: Self-Healing and Optimizing Routing Techniques for Mobile Ad Hoc Networks,” in Proc. of MobiHoc’03, pp.279-290. Maryland, USA, 2003.

[28] R. Meester and R. Roy, Continuum Percolation, Cambridge University Press, 1996.

[29] P. Hall, “On continuum percolation,” The Annals of Probability, vol.13, no.4, 1985.

[30] S. Y. Ni, Y. C. Tseng, Y. S. Chen and J. P. Sheu, “The broadcast storm problem in a mobile ad hoc network,” in Proc. of MobiCom'99, Seattle, Washington, USA, pp.151-162.

[31] M. B. Yassein and M. B. Khalaf, “A performance comparison of smart probabilistic broadcasting of ad hoc distance vector (AODV),” Journal of Supercomputing, March 2010, ISSN 0920-8542.

[32] J-D Abdulai, O-K. Mohamed,and L. M. Mackenzie, “Adjusted probabilistic route discovery in mobile ad hoc networks,” Computers & Electrical Engineering, vol.35, no.1, pp.168-182, Jan. 2009.

[33] Z. J. Haas, J. Y. Halpern and L. Li, “Gossip-based ad hoc routing,” IEEE Trans. on Networking, vol.14, no.3, pp.479-491, June 2006.

[34] X. Y. Li, K. Moaveninejad and O. Frieder, "Regional gossip routing for wireless ad hoc network," Mobile Networks and Applications (MONET), vol.10, no.1-2, pp.61-77, 2005.

[35] Q. Zhang and D. P. Agrawal, "Dynamic probabilistic broadcasting in MANETs," J. Parallel Distrib. Comput., vol.65, pp.220-233, 2005.

[36] Ns notes and documentation. [Online]. Available: http://www.isi.edu/nsnam/ns/.

[37] Implementing a new manet unicast routing protocol in NS2. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.121.4215.

[38] MANET extension to ns2. [Online]. Available: http://andres.lagarcavilla.com/publications/MANET_extensions.pdf.

[39] The CMU Monarch project's wireless and mobility extension to ns. [Online]. Available: http://www.ietf.org/proceedings/42/slides/manet-dave-98aug.pdf.

[40] 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 the Proc. of MobiCom '98, pp. 85-97, New York, NY, USA, 1998.

[41] S. Y. Ni, Y. C. Tseng, Y. S. Chen and J. P. Sheu, "The broadcast storm problem in a mobile ad hoc network," in Proc. of MobiCom'99, Seattle, Washington, USA, pp.151-162.

[42] IEEE 802.11, Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, IEEE Std, June 2007.

[43] J. Jubin and J. D. Tornow, "The DARPA Packet Radio Network Protocols," in Proc. of the IEEE, vol.75, no.1, January 1987.

[44] Defense Science and Technology Organization, Department of Defense, Australian Government. http://www.dsto.defence.gov.au/.

[45] Motorola's Mesh Wide Area Networks. http://www.motorola.com/Business/US-EN/Business+Product+and+Services/Wireless+Broadband+Networks/Mesh+Networks/.

[46] Greenpacket. http://www.greenpacket.com/.

[47] DynaMesh. https://direct.sri.com/index.php/sri/page/products_aware_dynamesh_technology.

[48] Firetide. http://www.firetide.com/.

[49] RFC 2501-Mobile Ad Hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations. Jan. 1999.

[50] C. Siva Ram Murthy and B. S. Manoj. Ad Hoc Wireless Networks Architectures and Protocols. Prentice Hall, 2004.

[51] S. Murthy and J. J. Garcia-Luna-Aceves, "A routing protocol for packet radio networks," in Proc. of the 1st International Conference on Mobile Computing and Networking, ACM MOBICOM'95, November 13-15, 1995, Berkeley, California, USA, pages 86{95. ACM, ACM, November 1995.

[52] C. E. Perkins and P. Bhagwat, "Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers." ACM SIGCOMM Computer Communication Review, vol.24, no.4, pp.234-244, October 1994.

[53] J. Eriksson, M. Faloutsos, S. Krishnamurthy, "DART: Dynamic address routing for scalable ad hoc and mesh networks," IEEE/ACM Trans. on Networking, vol.15, no.1, pp. 119-132, 2007.

[54] B. Chen and R. Morris, "L+: scalable landmark routing and adress lookup for multi-hop wireless networks," 2002.

[55] G. Pei, M. Gerla, and T.-W. Chen, “Fisheye state routing: A routing scheme for ad hoc networks,” in Proc. of ICC’00, no.1, pp.70-74, 2000.

[56] G. Pei, M. Gerla, and X. Hong, "Lanmar: Landmark routing for large scale wireless ad hoc networks with group mobility," in Proc. of ACM MobiHOC’00, 2000.

[57] G. Pei, M. Gerla, . Hong, and C.-C Chiang, "A wireless hierarchical routing protocol with group mobility," in Proc. of ACM MobiHOC"00, 2000.

[58] J. J. Garcia-Luna-Aceves and Marcelo Spohn, “Source-tree routing in wireless networks,” in Proc. of the 7th Annual International Conference on Network Protocols (ICNP'99), p.273, Washington, DC, USA, 1999. IEEE Computer Society.

[59] P. F. Tsuchiya, "The landmark hierarchy: A new hierarchy for routing in very large networks," in SIGCOMM’88, ACM, 1998.

[60] R. Ramanathan and M. Steenstrup, "Hierarchically-organized, multihop mobile wireless networks for quality-of-service support,” Mobile Networks and Applications, vol.3, no.1, pp. 101-119, 1998.

[61] W. Liu C. C. Chiang, H. K. Wu and M. Gerla, “Routing in clustered multi-hop mobile wireless networks with fading channel,” in Proc. of IEEE SICON, p.197-211, April 1997.

[62] RFC 3561–Ad Hoc On-Demand Distance Vector (AODV) Routing. July 2003.

[63] J. J. Garcia-Luna-Aceves, “Flow-oriented protocols for scalable wireless networks,” in proc. of MSWiM ’02, Atlanta, Georgia, USA, Sept. 2002.

[64] V. Park and S. Corson, “Temporally-ordered routing algorithm (TORA) version 1 functional specification,” draft-ietf-manet-tora-spec-04.txt, July 2001.

[65] Y. Ko and N. H. Vaidya, “Location-aided routing (LAR) in mobile ad hoc networks,” in Proc. of ACM MOBICOM, p.66-75, October 1998.

[66] C. K. Toh, “Associativity-based routing for ad hoc mobile networks,” Wireless Personal Communications, vol.4, pp.1-36, March 1997.

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

[68] M. S. Corson and A. Ephremides, "A destributed routing algorithm for mobile wireless networks," ACM-Baltzer Journal of Wireless Networks, vol.1, pp.61-81, January 1995.

[69] R. Sivakumar P. Sinha and V. Bharghavan, “Cedar: A core extraction distributed ad hoc routing algorithm,” IEEE Journal on Selected Areas in Communications, vol.17, no.8, pp.1454-1466, August 1999.

[70] M. J.-Ng and I. T. Lu, “A peer-to-peer zone-based two-level link state routing for mobile ad hoc networks,” Journal on Selected Areas in Communications, vol.17, no.8, pp.1415-1425, August 1999.

[71] Z. J. Haas, “The routing algorithm for the reconfigurable wireless networks,” in Proc.of ICUPC’97, vol.2, p.562-566, October 1997.

[72] The NS Manual, December 2003. [Online]. Available: http://www.isi.edu/nsnam/ns/ns-documentation.html.

[73] Ke Liu, “Understanding the implementation of IEEE MAC 802.11 standard in NS-2”. http://www.cs.binghamton.edu/~kliu/research/ns2code/note.pdf.

[74] H. T. Friis, “A note on simple transmission formula,” in the Proc. IRE, vol.34, 1946.

[75] T. S. Rappaport. Wireless Communication: Principles and Practice. Prentice Hall, Upper Saddle River, NJ, October 1995.

PUBLICATIONS

[1] Z. Liang, Y. Taenaka, T. Ogawa, and Y. Wakahara, "Automatic Route Shortening Based on Supportive Vector Machine in Wireless Ad Hoc Networks," IEICE Society Conference, Sapporo, Hokkaido, Japan, September 2011. BS-8-1. (to appear)

[2] Z. Liang, Y. Taenaka, T. Ogawa, and Y. Wakahara, "Automatic Route Shortening for Performance Enhancement in Ad Hoc Wireless Networks," The 13th Network Software (NWS) Conference, Kushiro, Hokkaido, Japan, June 2011.

[3] Z. Liang, Y. Taenaka, T. Ogawa and Y. Wakahara, “Pro-reactive route recovery with automatic route shortening in wireless ad hoc networks,” The 10th International Symposium on Autonomous Decentralized System (ISADS 2011), June 2011, Kobe, Japan.

[4] Z. Liang, Y. Taenaka, T. Ogawa and Y. Wakahara, “Incorporating route recovery information into gossip routing,” IEICE General Conference, March 2011, Tokyo, Japan. BS-4-16.

[5] Z. Liang, Y. Taenaka, T. Ogawa and Y. Wakahara, “A pro-reactive route recovery scheme in wireless ad hoc networks,” The 1st International Symposium on Multidisciplinary Research, March 2011, Tokyo, Japan.

[6] Z. Liang, Y. Taenaka, T. Ogawa and Y. Wakahara, “Fast and low-overhead local Route recovery in ad hoc wireless networks,” IEICE Technical Report, NS2010-146, pp. 25-30, January 2011.

[7] Z. Liang, Y. Taenaka, T. Ogawa and Y. Wakahara, “Automatic route recovery based on relay nodes in wireless ad hoc networks,” IEICE Society Conference, September 2010, Osaka, Japan. BS-7-24.

[8] Z. Liang, T. Ogawa and Y. Wakahara, “An automatic route recovery method based on reserved virtual local paths for wireless ad hoc network,” IEICE General Conference, March 2010, Sendai, Japan. ISS-P-134.

Award

[1] Z. Liang, “Harajima Award” for “Routing in Ad Hoc Wireless Networks”, 財団法人 電気・電子情報学術振興財団, 9th June 2011.