

Cross-layer Inference-based Fast Link Error Recovery for MANETs

Justin Yackoski Chien-Chung Shen
Computer and Information Sciences
University of Delaware
Newark, DE 19716
{yackoski, cshen}@cis.udel.edu

Abstract—In traditional MANET MAC and routing protocols, if any link in a route fails, multiple fruitless attempts are made to use the failed link before reporting failure to the routing layer or attempting local recovery. The high frequency of link errors between mobile nodes requires rapid recovery in order to provide acceptable performance, especially for real-time applications. In this paper, we propose CIFLER, a cross-layer approach which uses enhanced channel reservation messages to allow alternate nodes to immediately elect themselves using only inferred neighbor information. This self-election avoids reliance on individual links and minimizes the impact frequent link errors have on delay, energy usage, and the function of upper layer protocols. We show via simulation that CIFLER provides better results in typical MANET scenarios. Unlike other local recovery schemes, CIFLER does not suffer from duplicated messages, allows new nodes to almost immediately learn the information needed to assist in the recovery of existing routes, and does not require additional hardware, delays, or control messages.

Keywords: Mobile ad hoc networks, local route repair, cross-layer design.

I. INTRODUCTION

In mobile ad hoc networks (MANETs), node mobility, wireless interference, and other concerns challenge the routing of packets between nodes. Routes require frequent repair due to link errors, especially in popular routing protocols which select routes using a shortest distance (or fewest-hop) criteria. Many MANET routing protocols address this issue through on-demand route maintenance and local error recovery, but link errors often are handled using additional control messages or retransmissions, which can result in significant delays. Fast recovery from these common problems is crucial to reducing jitter, out-of-order packets, and errors in higher layers. For these reasons, link errors should be *expected* and handled seamlessly in the MAC layer, allowing the routing protocol to focus its effort on addressing more complicated route failures. In this paper we show that during a link error in a MANET, time and effort can be wasted in protocols which do not differentiate between *link errors* and more severe *route failures* which require advanced repair.

In order to clarify the difference between a link error and a route failure, consider route $S-N_1-N_2-D$. If there is only a

Prepared through collaborative participation in the Communications and Networks Consortium sponsored by the U. S. Army Research Laboratory under the Collaborative Technology Alliance Program, Cooperative Agreement DAAD19-01-2-0011. The U. S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation thereon.

link error in link $S-N_1$, some node N_j with a link to nodes S and N_2 can be used instead of N_1 , or some nodes N_{j1} and N_{j2} can be used such that the repaired route is $S-N_j-N_2-D$ or $S-N_{j1}-N_{j2}-N_2-D$. Thus, routes experiencing a link error can be repaired locally using only neighbors of the nodes with broken links. In contrast, when a route failure occurs no such local recovery is possible. For example if the network topology changes so that the neighborhoods of N_1 and N_2 have no links between them, then a route involving nodes in a geographically different area must be used.

We propose a Cross-layer Inference-based Fast Link Error Recovery mechanism, termed CIFLER, which allows fast detection of and recovery from link errors at the MAC layer, providing a more stable route to higher layers. CIFLER is compatible with both source and link-state routing schemes and any RTS/CTS-based MAC protocol. CIFLER can also be used in conjunction with other recovery techniques designed to handle more severe route failures.

The remainder of this paper is organized as follows: In Section II we summarize related work and the motivation for developing CIFLER. Section III describes CIFLER in detail. In Section IV we examine CIFLER's performance via simulation, and Section V contains the conclusions drawn from our analysis.

II. MOTIVATION AND RELATED WORK

Many route recovery techniques for MANET routing protocols have been proposed. However nearly all existing protocols impose significant additional costs or requirements, and do not fully utilize the local topology information which nodes can infer from overheard information. The main problem faced by recovery schemes is determining suitable alternate nodes quickly and immediately updating local recovery information at each node when a change in topology occurs. This is especially difficult without relying on additional constraints such as GPS, periodic (and thus stale) beacon information, or on-demand broadcasts which are subject to "grey zone" problems [11]. In this section, we summarize several categories of existing recovery protocols and discuss the common problems which CIFLER is designed to avoid.

A. Mode-Based Recovery Protocols

Mode-based recovery protocols are common and have a "recovery mode" behavior which is used only after a link error has been detected. Since recovery from link errors is not part of

the normal protocol operation, significant delays are possible. Examples of these protocols include DSR [2] and AODV-BR [3] which use cached routes as alternatives, REGR [5] and [6] which better focus the search for alternate routes, and SHARP [7] which uses local pro-active routing.

In Bypass Routing [13], a node which has detected a failure sends a request to its neighbors looking for an alternate link to bypass the failure. Error detection occurs at the network layer, meaning the time between a failure and a recovery is several milliseconds or more. Like other network-layer-only recovery schemes, Bypass Routing does not attempt to avoid MAC-layer link errors and assumes that link errors are permanent.

Witness-Aided Routing (WAR) [12] collects and uses local topology information without explicit topology maintenance traffic. Since WAR acts solely at the network layer after MAC-layer failures occur, recovery is still slow. WAR also relies on subsequent hops to arbitrate between *witness hosts* attempting to repair the broken link, which may result in packet duplication when such arbitration is not possible. Additionally, WAR does not directly address *route stretching*, where additional hops must be added to a route as in Figure 2(b).

B. Election-based Recovery Protocols

Since link errors are both common and unpredictable, election-based protocols constantly adjust the route behind the scenes as needed to both handle and avoid link errors, thus the explicit routing information serves more as “guidelines” than actual rules. Election-based protocols fit into one of two categories. In “forwarder-election” protocols the node which is forwarding the message chooses the best next-hop *before* any transmission. In “self-election” protocols the next-hop nodes elect themselves through some mechanism *after* a transmission.

NSR [8] is a forwarder-election protocol where nodes maintain 2-hop information about their neighborhood to repair routes, which is similar to SHARP [7] and ADV [9]. With NSR, each hop has the freedom to choose any suitable next hop it is aware of. This reduces dependence on individual links, as alternatives are already available before a link error occurs. NSR uses periodic broadcasts to maintain the neighbor information, requiring additional network overhead and causing neighborhood information to become stale between broadcasts, making recovery less effective.

Location-based protocols such as LAR [4] and CBF [10] avoid link errors using GPS to calculate the likelihood their forwarding will be beneficial and using contention or some other means for nodes to perform self-election. However, these schemes require accurate location information and are intended to be used for normal routing in location-based networks, not error recovery on an established route.

ExOR [14] is a hybrid approach where each data message contains a list of forwarder-elected candidate next-hop nodes, and one of these candidates is chosen by self-election. This lack of complete self-election reduces the number of nodes which may respond and increases the time before a new candidate node may participate since it must first be known about by the previous hop. ExOR delays final selection until *after* the data is

transmitted, unlike CIFLER where the decision is made during the RTS/CTS exchange *before* the data is transmitted. This is a significant distinction in that it requires each candidate next hop node to store the entire data message. Control message loss can lead to energy waste as more than one candidate next hop node may forward the data message. Finally, the decision making period where self-election occurs in ExOR has a fixed length to allow all listed candidate nodes to possibly acknowledge the data transmission, resulting in unnecessary delay since the best candidate may receive and successfully forward the data a significant portion of the time but must wait *every* time for the other candidate nodes to acknowledge.

C. Common Design Problems

Existing recovery techniques suffer from several problems which CIFLER is designed to avoid. Many schemes such as [2] and [13] use cached routes as alternatives, but these routes may not be up-to-date and may themselves require repair. In addition, any working cached routes are likely to be part of other active routes which can result in routes tending to use the same path and share bandwidth. If the original route selection criteria is designed to avoid this overlap, the repaired route should be as similar as possible to the original route.

Routing algorithms such as [15] have also been proposed which address the problem of node mobility and link instability by biasing route selection against nodes deemed less stable. While useful in some scenarios, the bias results in more energy usage by the stable nodes, and “stable” links may fail without warning, limiting the effectiveness of these schemes. In addition, unstable nodes tend to be more mobile, and more mobile nodes are often better equipped and have more available energy (i.e., a person vs. a vehicle) and it is beneficial to use the energy of more powerful nodes as much as possible, instead of biasing route selection against them.

Routes with the fewest hops are often preferred by current routing protocols as they are usually the most direct path, but since they have the fewest hops they also contain longer and more unstable links, and the nodes more easily move out of range as in Figure 2(b). Since using the most direct path is desirable and the fewest hop criteria is simple, these routes are useful as long as having some unstable links does not significantly affect performance. CIFLER directly addresses this by allowing a node to perform forwarding between two consecutive but physically distant nodes in a route when necessary.

To achieve the most benefit, routes used most frequently should clearly have the best local recovery information. Nodes which move close to a previously unknown route should also be able to very quickly participate in the repair of that route. Collecting both positive and negative recovery information about neighboring nodes from each (or every Nth) overheard packet allows this. With CIFLER, nodes near active routes are aware of changes in their ability to repair the route after hearing a single packet traverse the nodes which make up the route, allowing the freshness and amount of recovery information for

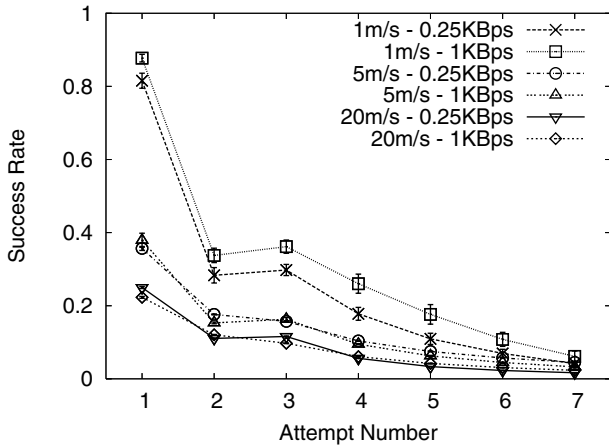


Fig. 1. Individual RTS success rates measured by simulation as described in Section IV for unmodified DSR and IEEE 802.11 DCF (the “Base” protocol) under various average node speeds and traffic per flow.

a particular route to be directly related to the frequency of traffic along the nodes which comprise it.

Finally, we looked at IEEE 802.11 DCF which re-attempts an RTS several times (7 for 802.11b, 3 for 802.11a) upon hearing no response from the intended recipient before returning an error to the network layer. We define a successful RTS as an RTS for which a CTS response is heard, which does not necessarily mean successful data transmission occurred. As shown in Figure 1, we found that the percentage of a node’s first RTS attempts which are successful is below 40% even under minimal traffic and moderate mobility, a significant waste of time and energy. CIFLER directly addresses this by increasing the number of nodes which can respond to a CTS, which increases the RTS success rate. Since link errors are common in MANETs and link quality changes rapidly, especially as mobility increases, recovery should be extremely fast.

For these reasons, we propose CIFLER which can detect and repair link errors by treating each MAC-layer RTS as an implicit recovery request, allowing recovery to often occur *before* a second RTS is transmitted. CIFLER does this while avoiding the problems mentioned above and without significant additional constraints or costs.

III. CIFLER

We present the general CIFLER concept and the adjustments needed for our implementation based on IEEE 802.11 DCF and DSR. Any MAC protocol which uses an RTS/CTS mechanism and any source-routed or link-state routing protocol can be adapted to use CIFLER with few adjustments.

The main concept of CIFLER is the lack of a strong reliance on individual links. As an example, in Figure 2(a), packets from S to R are routed through node A, using links S–A and A–R. Although node A is chosen for the explicit route, any node in Group S-R would be suitable as an alternative for A.

Ideally, every time S has a packet to send to R, the current best node in Group S-R should be selected to forward through.

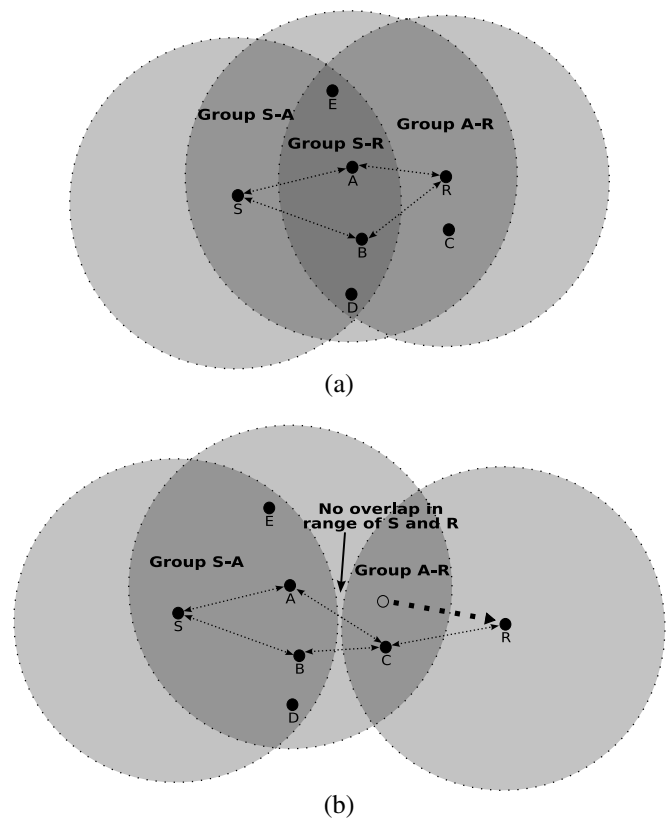


Fig. 2. (a) Initially, nodes A and B are in Group S-R, which includes all nodes within range of S and R. Group A-R contains node C, among others. (b) Later, node R has moved and Group S-R no longer exists.

However, determining the exact answer to this isn’t feasible on a per-packet basis, and routing protocols typically select a suitable node at each hop and then re-select new nodes only when needed. CIFLER works by treating the nodes selected for the route as “preferred” nodes (node A in Figure 2(a)). Since a link error is expected, CIFLER prepares *in advance* all usable alternate nodes to “stand-in”, or perform forwarding along the route in place of broken links to preferred nodes.

At each hop, CIFLER uses the initial route to determine where suitable next-hop nodes exist to forward packets. The routing protocol provides the MAC layer with “next-next-hop” information to allow candidate nodes to determine if they are a suitable next-hop. In Figure 2(a), from node S’s perspective, the next-hop is A, and the next-next-hop is R. All nodes in Group S-R are suitable alternates for node A due to being within the transmission range of node S and node R.

Consider node B which is not aware of route S–A–R and hears node S’s RTS request to node A to forward a packet to next-next-hop node R. Node B must determine whether it is in Group S-R to know if it can respond to the RTS in A’s place. Since node B heard S’s request, clearly node B is within range of node S, which meets the first criterion for being in Group S-R. Assuming that node B has also heard node R transmit recently (i.e., when the previous packet was sent along route S–A–R), then node B has met the second criterion, and hence

node B can infer its membership in Group S-R.

Once aware of its membership in Group S-R, if node B infers the failure of link S-A, node B can stand-in for the middle node, node A, in the route segment S-A-R and forward the packet from node S to node R. In fact, if node B has heard recently from node R, node B has sufficient information to stand-in for the middle node Y in the route segment $X-Y-R$, where X and Y are any nodes. Node B can then respond to S's RTS requests after waiting a very short delay to give the preferred next-hop node mentioned in the RTS a chance to respond before inferring its failure. In this way, alternatives are prepared for link errors before they occur and can detect them in time to respond to the first RTS sent along the failed link.

Another important situation to handle in MANETs is "route stretching," such as in Figure 2(b) where node R moves out of range of node A, such that Group S-R disappears. When this occurs, CIFLER acts as though the original route was S-A- X_1 -R, and node C can stand-in for imaginary node X_1 in the route segment A- X_1 -R by identifying itself as a member of Group A-R. Note that node C is already in Group A-R in Figure 2(a) well before link A-R breaks. Node C is therefore immediately able to stand-in when needed in Figure 2(b).

In some cases, node A may not be responding due to channel congestion. Since alternatives for A are in the same physical area as A, it is more likely that the channel is congested from the alternatives' perspective as well. However, since the alternatives wait a short time after hearing the RTS to sense the channel and attempt to send a CTS, the channel may be clear. We expect CIFLER to behave well under congestion, but not necessarily to improve performance in this situation.

A. Routing Protocol Details

The routing protocol must provide additional information to the MAC layer but requires no functional changes. First, in addition to providing the next hop, the routing protocol must provide the MAC layer with the address of the "next-next-hop". For source-routed protocols, the next-next-hop is the ID of the node in the route immediately after the next-hop specified in the RTS, and a protocol such as DSR need only provide it to the MAC layer with each data message. For link-state protocols, there are several implementation possibilities which we do not discuss at length due to space constraints. One approach is to add a next-next-hop field to routing table entries so that the information can be given to the MAC layer.

Route stretching is a key part of CIFLER, and the routing protocol must regulate when stretching is and is not allowed. At the source node, the time-to-live (TTL) of the packet is set to $2n + 2$ where n is the number of hops in the source route. This limits the amount of stretching done on a given packet, since the MAC-layer CIFLER behavior does not detect loops and other problems.

At each hop, the routing protocol informs the MAC layer of whether or not stretching is allowed at the current hop for each message by checking the TTL in the IP header. To prevent a disproportional amount of effort near the start of the route, stretching is only allowed if the TTL of the packet is greater

than twice the number of remaining hops between the current node and the destination. This constraint allows a route which has been stretched little during the first part of the route to stretch considerably more near the destination.

This behavior near the end of the route is desirable because considerable effort has already been spent getting the packet to its current position, and the global destination node may have moved slightly away from the rest of the route. Although such a route may be non-optimal in a normal situation, this behavior very near the destination helps increase the delivery rate. In the rest of the route, other nodes can stand-in for the preferred nodes in the route, but the last node *must* be the destination node. Therefore, only stretching can be used if the destination does not respond to an RTS.

B. MAC Layer Details

CIFLER requires one additional node address on RTS and CTS messages. RTSs normally include the local sender ID and the local destination (next-hop) ID. CIFLER adds a *next-next-hop* ID field to the RTS which carries the next-next-hop address provided by the routing protocol. The next-next-hop field allows nodes other than the next-hop mentioned in the RTS to respond. CTSs normally include only the ID of the intended recipient of the CTS. CIFLER adds a source field for the ID of the sender of the CTS. Since a node other than the intended recipient of an RTS may respond to it, this field is needed to allow the sender of the RTS to direct the subsequent DATA message to the correct node.

While *not* required on every RTS and CTS, our implementation includes these extra fields on all RTS and CTS messages. An alternate implementation may normally use standard RTS and CTS messages, and only include the extra fields when the standard RTS attempt fails or is likely to fail. This would slightly reduce overhead at the expense of slightly slower error recovery, making it useful when link errors are much less likely to occur.

C. Neighbor Lists

The MAC layer at each node maintains a whitelist and a blacklist of neighbors. In cases where a neighbor is listed in both the whitelist and the blacklist, the blacklist takes priority. Each list is limited to n_{max} unique node IDs and entries expire upon reaching the expiration time most recently specified for the entry. The whitelist contains the IDs of neighbors which have been heard from recently, and thus assumed to be within transmission range. Whenever a node successfully receives any MAC layer unicast control or data message, the node adds the sender's ID to the whitelist for t_w seconds.

The blacklist contains the IDs of neighbors which are known to be unreachable, and allows CIFLER to handle unidirectional links, dead-end routes, and other situations where a node may be whitelisted but should not be forwarded to. Whenever a node attempts to transmit data to a neighboring node and fails due to exceeding the RTS retransmit limit, the destination node is removed from the sender's whitelist and added to the sender's blacklist for t_f seconds. The link is likely broken or temporarily

unstable in this case, hence the sender node should not attempt to contact the node until it is heard from again.

In several other situations, assumptions about the state of a link can be made using overheard transmissions. Since the whitelist entry expiration time is t_w more than the time each neighbor was last heard from, when node A hears a CTS destined for node S, the whitelist entry can be checked to see whether node A has heard from node S within the time period that the corresponding RTS was likely sent. If node A has not heard from node S recently, then node A did not hear the RTS. Thus, node A can infer that node S is no longer a neighbor, and node A removes node S from the whitelist and adds node S to the blacklist for t_b seconds. A similar method is used to remove neighbors when a node overhears DATA and ACK messages. By making these inferences, each node is rapidly aware of any changes in the status of links to whitelisted nodes on active routes.

The neighbor list size in CIFLER will be much smaller than other schemes since broadcasts are not used to populate them. Since nodes elect themselves, nodes in an active route do not need to be aware of candidate stand-in nodes. Instead, nodes which are willing to assist in link error recovery need only know about neighboring nodes which are actively transmitting messages. If node density and the number of routes are both extremely high, a random deletion scheme can be used to keep the whitelist small while still allowing a subset of the capable nodes to be ready to stand-in for each route. The blacklist size also depends on the number of active neighbors and will generally be small, but can be further decreased if needed by shortening t_f and t_b , or by not using the optimizations mentioned in the previous paragraph.

D. Next-Hop Selection

When sending an RTS, the sender, hereafter N_s , includes on the RTS the next-next-hop address provided by the routing protocol. If the next-hop is the destination of the packet then the normal stand-in behavior is not possible, and the next-next-hop is set to the broadcast address, which is never present in the neighbor lists, preventing a stand-in from occurring.

Upon hearing an RTS, each node, hereafter referred to as N_r , first checks if it is the intended recipient specified in the next-hop field of the RTS, hereafter N_p . N_p responds immediately if the channel is clear, as in standard 802.11 DCF. Otherwise, if N_r is not already busy trying to send or receive a packet, N_r checks to ensure the next-next-hop on the RTS is present in N_r 's whitelist and not present in N_r 's blacklist. If so, N_r may stand-in for the next-hop as shown in Figure 3.

To avoid congestion, N_r estimates the number of other nodes likely to stand-in, ψ , by dividing the number of neighbors in N_r 's whitelist by a factor of f and adding 1 for itself. Experimentally, we have found that $f = 8$ works well for common scenarios. N_r then stands-in with a probability of $1/\psi$. To give first priority to N_p , N_r first waits one slot to allow time to hear the beginning of the CTS that N_p may send. N_r then uses a Random Wait Period between 0 and n_s slots to reduce contention between multiple N_r nodes. Since N_s waits between

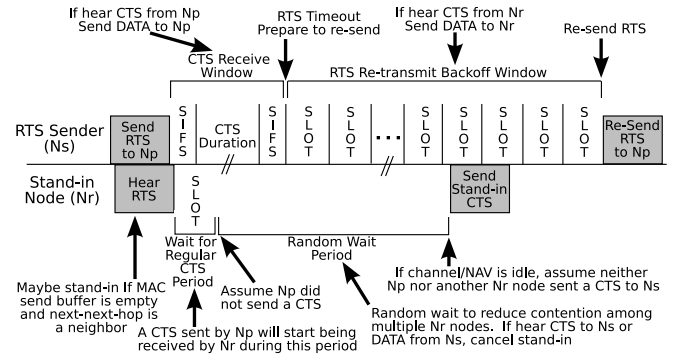


Fig. 3. Top - behavior of N_s sending an unanswered RTS to N_p . Bottom - behavior of potential stand-in node N_r upon hearing an RTS from N_s and no CTS from N_p . Points where inferences can be made and actions may be taken are indicated with arrows.

RTS re-attempts, CIFLER utilizes this time for possible stand-in CTSs to be received.

If N_r 's NAV indicates the channel is currently busy, the Random Wait Period starts after the NAV timer expires. Note that since N_r may be the node which responds to the RTS, N_r 's NAV is not updated by an overheard RTS if N_r has decided to attempt to respond to it with a stand-in CTS. If this were not done, when the Random Wait Period expires N_r 's NAV would always indicate the channel is busy on the assumption that N_p will respond.

If, during either the Wait for Regular CTS Period or the Random Wait Period, N_r hears any CTS messages to N_s or DATA messages from N_s , N_r infers that the RTS/CTS exchange was successful and the stand-in is canceled. Once the Random Wait Period has passed, if N_r sees that the channel is busy (either by checking the physical channel or via NAV) or the node itself is busy sending/receiving something else, the stand-in is canceled. This is done since the traffic in the channel is either from node N_s or N_p indicating the RTS/CTS exchange was successful, or is from another node indicating the RTS/CTS exchange is currently not possible due to congestion. Otherwise, when none of these conditions are met a link error between N_s and N_p is inferred and a stand-in CTS is transmitted to N_s . N_r includes its ID in the source field of the CTS.

Upon hearing the stand-in CTS, N_s sends a DATA message addressed to the source ID included on the CTS, which may not be N_p . Specific addressing of the DATA message prevents packet duplication and requires only one node, the source of the CTS, to buffer, process, and forward the DATA. An ACK is then also sent as usual upon receipt of the DATA.

Due to the waiting periods before a stand-in CTS is sent, N_s often receives the stand-in CTSs after the original RTS has timed out and N_s is in the RTS Re-transmit Backoff Window, as shown in Figure 3. Normally in 802.11 DCF, N_s would neither expect nor respond to a CTS during this period. In CIFLER, however, N_s cancels the RTS re-transmission timer and responds to the stand-in CTS immediately.

E. Route Stretching and Compression

Due to the low RTS success rates for IEEE 802.11 DCF in Figure 1, CIFLER attempts to increase the RTS success rate by allowing route stretching as a last resort before reporting an error to the network layer. Unlike the regular stand-in behavior, stretching *is* allowed when the next-hop is the global destination.

If allowed by the routing protocol based on the TTL, CIFLER sends a “stretching” RTS on the 3rd and subsequent retry attempts. In a stretching RTS, the *next-hop* address is copied to the *next-next-hop* field on the RTS. This allows one hop to be added to the route since nodes can then perform a stand-in if the next-hop is in their neighbor list. The node performing the stand-in then attempts to forward the packet to the next-hop and continue the normal route. Stretching only changes N_s 's behavior, the N_p and N_r nodes are not aware that the next-hop and next-next-hop fields on the RTS are equal and act as in the regular stand-in case.

N_p is almost certainly closer to N_s than the next-next-hop is, and the stretching RTS targets nodes between N_p and N_s . This is desirable because N_s and N_p may have separated or have a low-quality link, requiring two shorter links to be substituted in its place. Although regular CIFLER RTSs address this somewhat by allowing other nodes in contact with both N_s and the next-next-hop to respond, the route may still be unreparable as in Figure 2(b) where stretching the route by adding node C is needed to reach node R.

Route compression is also allowed by CIFLER, though it had little impact on performance in our experiments. If a node hears an RTS where the next-next-hop is its own ID, the node may send a stand-in CTS as described above.

IV. EXPERIMENTAL RESULTS

In this section, we compare the experimental performance of traditional DSR and 802.11 DCF with versions enhanced by CIFLER. All simulations are run for 600s using QualNet 3.7 [16] with 10 CBR flows which randomly start in the first 5 seconds and continue for 590 seconds. The 600s run time allows nodes to travel significantly during the simulation to ensure many link failures occur. We use a transmission range of 250m and a data rate of 2Mbps. For the “Base” protocol, we use unmodified 802.11 DCF and DSR, including route caching, salvaging, and other DSR repair techniques. For the “CIFLER” protocol, we add the stand-in and stretching behaviors of CIFLER implemented as described in Section III. We use the following values for parameters: $n_{max} = 100$, $f = 8$, $t_w = 10$, $t_b = 2$, $t_f = 2$, and $n_s = 10$. Data presented is the average of 25 seeds with 95% confidence intervals.

Due to the issues discussed in [17], we use random waypoint mobility with no pause time and equal minimum and maximum speeds to ensure the speeds presented are in fact the exact average speed of the nodes. Since CIFLER is intended for MANETs where link errors are common and node mobility directly affects the frequency of link errors, it is critical to know the exact average node speed to understand its relationship with the measured data.

We randomly place 40 nodes in a square $774.6m \times 774.6m$ terrain, giving a node density of 13 nodes per $\pi 250^2$ coverage area. We use 512-byte packets with 0.25KBps (Kilo Bytes per second) per CBR flow for low traffic, 1KBps for moderate traffic, and 5KBps for high traffic. At 5KBps, the network is congested to the point that even under negligible mobility the delivery ratio is below 50% as shown in Figure 5. Certainly higher rates are desirable, but are not sustainable with 802.11 DCF in this configuration.

We study several metrics in this section, defined as follows. We consider RTS success rate as defined in Section II. Delivery ratio is defined as the percentage of CBR packets sent which are successfully received (i.e., unchanged and in order) at the destination. Route requests initiated measures the total number of times that no working route is available or cached, triggering DSR to initiate a RREQ. We use QualNet's default 802.11b energy model and only consider energy used for transmissions. Since we use a fixed 2Mbps data rate, energy consumption is directly related to the number of bits transmitted. We divide total energy by the number of successfully received CBR packets. Finally, to determine the end-to-end delay we calculate the time between when a CBR packet is first given to DSR at the source and when the packet is received by DSR at the destination.

A. RTS Success Rate

In Figure 4, we see that the increased number of nodes responding to an RTS greatly improves the RTS success rate over the Base protocol, shown previously in Figure 1. Under very low mobility, link errors are not very common, thus the success rate for the first RTS attempt is slightly worse than the Base protocol due to some nodes incorrectly inferring a link error and causing interference with unnecessary stand-in CTSs. However, with increasing mobility CIFLER's first RTS attempts are much more successful than the Base protocol due to CIFLER's ability to handle the increasing link errors. On the second RTS attempt, CIFLER's success rate drops as with the Base protocol because an RTS which was not successful on the first attempt is likely to be unsuccessful on the second attempt as well. On the third and subsequent attempts CIFLER's stretching behavior is used to allow a larger group of nodes to respond to the RTS, leading to a significant increase in the RTS success rate.

B. Delivery Ratio

Figure 5 shows the effect of node speed on delivery ratio. Under low traffic, CIFLER initially performs slightly worse but is less affected by increased mobility. When there is little mobility and thus few broken links, the slight uncertainty in CIFLER's inferred information is not completely offset by gains from repairing link errors. With little traffic in the network, the Base protocol can re-discover routes frequently and maintain high delivery rates without causing congestion.

Under moderate traffic, the Base protocol's performance is initially similar to the low traffic scenario, but as mobility increases and route discoveries and RTS re-transmissions congest

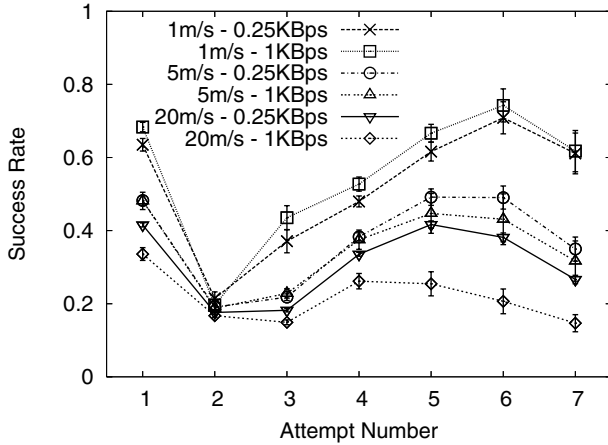


Fig. 4. Individual RTS success rates for CIFLER.

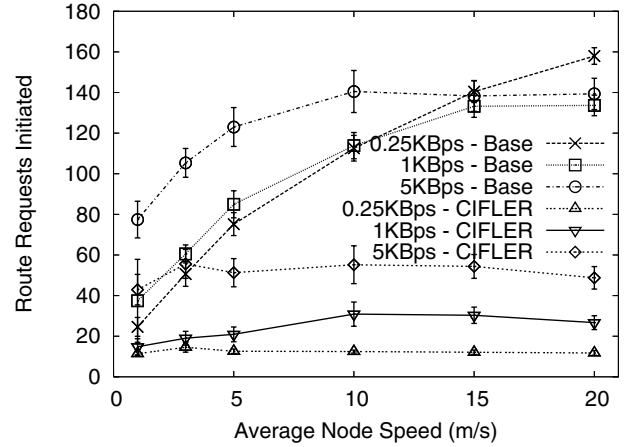


Fig. 6. Route requests initiated by DSR.

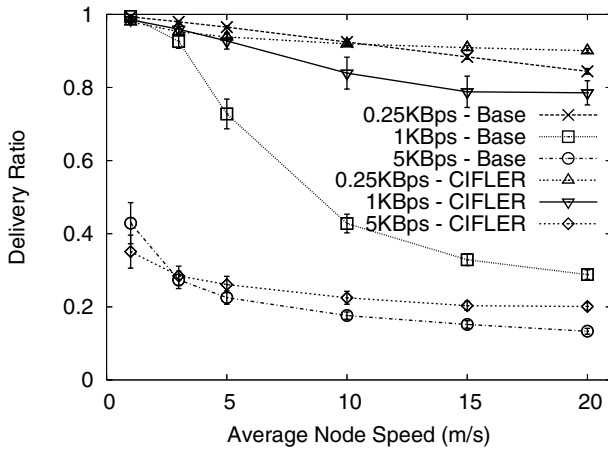


Fig. 5. Delivery ratio of CBR packets.

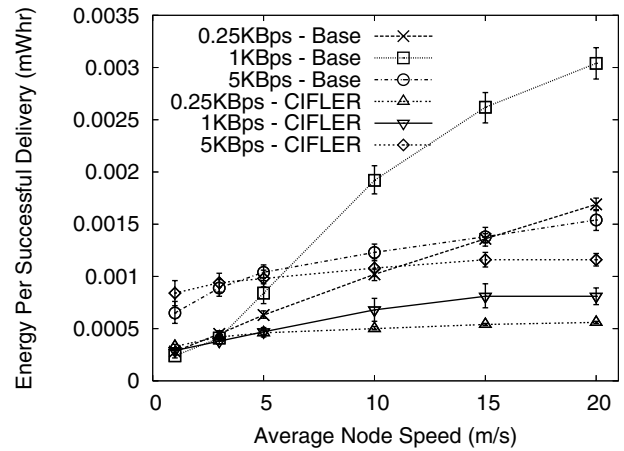


Fig. 7. Energy usage per successful delivery.

the network, performance degrades drastically. CIFLER, on the other hand, is able to handle moderate traffic well even under high mobility without additional control traffic. Under high traffic, CIFLER's performance is statistically similar to the Base protocol for low mobility, and as node speed increases, CIFLER is again less affected than the Base protocol.

C. Route Rediscovery

We expect that CIFLER will increase route lifetime over the base protocol under mobility, resulting in fewer route re-discoveries (RREQs). As figure 6 shows, CIFLER experiences significantly fewer RREQs than the Base protocol. This is especially true for the low traffic case, where the Base protocol is working hard to maintain a delivery ratio similar to CIFLER. Under moderate to high traffic, the Base protocol's retransmissions are significantly delayed by the added congestion, thus routes failures are not detected as quickly or as often, causing fewer RREQs than in the low traffic case.

D. Energy Consumption

Although CIFLER uses extra energy to send additional stand-in CTSs and one additional node address on each RTS and CTS, we expect a net energy savings due to fewer RTSs and route re-discoveries. Figure 7 shows that CIFLER's energy usage per delivery is lower in all cases above 5m/s. Under low traffic, CIFLER and the Base protocol maintain similar high delivery ratios as mobility increases, but CIFLER's energy usage increases by a factor of 1.6 between 1m/s and 20m/s, while the Base protocol's energy increases by a factor of 5 due to additional broadcasts and RTS re-transmissions. Thus, the two protocols are able to deliver a similar number of the packets, but CIFLER does so in a much more efficient manner.

E. Delay

A key motivation of CIFLER is to reduce latencies due to link failures and wasted RTSs. Figure 8 shows that for low to moderate traffic, CIFLER's delay is relatively unaffected by increased mobility. In the low traffic case, CIFLER's delay between 1m/s and 20m/s increases by a factor of 2.5 from 0.033s to 0.085s, while the Base protocol's delay increases from

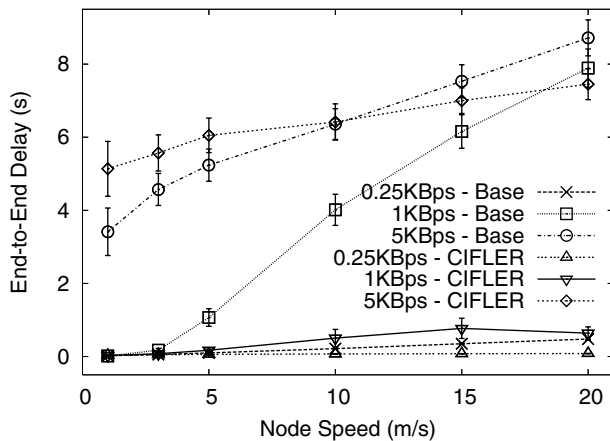


Fig. 8. Average end-to-end delay of delivered packets.

0.029s to 0.51s, a factor of 17.5. As with other metrics, the Base protocol's delay increases significantly under moderate to high traffic due to the increased congestion from RTS re-transmissions and route discoveries.

For the high traffic case, CIFLER has a slightly higher delay than the Base protocol under low mobility but is less affected by increased mobility than the Base protocol. In a highly congested network with little mobility, the low RTS success rate and backoff period of traditional IEEE 802.11 DCF act as throttling mechanisms. CIFLER uses slightly larger RTSs and CTSs and is designed to more aggressively send CTSs during the backoff periods, leading to poorer performance when there are few mobility-related link errors, the network is fully saturated with traffic, and no congestion control is used.

V. CONCLUSIONS

We have shown that through very simple cooperation between the routing protocol and the MAC layer to make RTS transmissions more than just channel reservation messages, link errors can be better handled and the RTS success rate can be significantly improved. Through these foundational improvements, CIFLER can provide faster and more effective recovery in mobile environments than existing protocols which do not directly address such link errors.

In addition, having available nodes elect themselves using inferences avoids the problem of maintaining and distributing neighbor information. We have shown that inferred neighbor information is very useful to make decisions when a link error occurs. Although stand-in nodes do not know for certain that they will be able to forward to the next-next-hop, we have shown that inferred information is reliable enough to make effective recovery decisions. In situations where the delivery ratio would otherwise be relatively low, the small percentage of packets which are lost or delayed due to incorrect inferences is minimal. MAC protocols like IEEE 802.11 DCF must already handle the uncertainty of the wireless medium via retransmissions, thus CIFLER's inferences are unobtrusive.

Currently, we have focused on using CIFLER to extend route lifetime by using temporary alternate links to repair link errors. However, such errors may be permanent and CIFLER could use the MAC layer information to instruct the routing protocol to make and propagate permanent route changes. Although more complicated due to the possible loops and other problems such updates may introduce, permanent route changes avoid the random backoff and contention for broken links which are repeatedly corrected. In addition, without permanent updates, if two consecutive nodes in a route fail, eventually the route will be unreparable by CIFLER. Our research is in progress to address these issues.

REFERENCES

- [1] C. E. Perkins, E. M. Royer, and S. R. Das, "Ad hoc on-demand distance vector (aodv) routing," in *2nd IEEE Workshop on Mobile Computing Systems and Applications*, New Orleans, LA, February 1999.
- [2] D. B. Johnson and D. A. Maltz, "Dynamic source routing in ad hoc wireless networks," in *Mobile Computing*, Imielinski and Korth, Eds. Kluwer Academic Publishers, 1996, vol. 353.
- [3] S.-J. Lee and M. Gerla, "Aodv-br: Backup routing in ad hoc networks," in *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC 2000)*, Chicago, IL, September 2000.
- [4] Y.-B. Ko and N. H. Vaidya, "Location-aided routing (lar) in mobile ad hoc networks," *Wireless Networks*, vol. 6, no. 4, pp. 307–321, 2000.
- [5] Y. Liu, X. Hu, M. J. Lee, and T. N. Saadawi, "A region-based routing protocol for wireless mobile ad hoc networks," *IEEE Networks Magazine*, pp. 12–17, July/August 2004.
- [6] R. Castañeda, S. R. Das, and M. K. Marina, "Query localization techniques for on-demand routing protocols in ad hoc networks," *Wireless Networks*, vol. 8, no. 2/3, pp. 137–151, 2002.
- [7] V. Ramasubramanian, Z. J. Haas, and E. G. Sirer, "Sharp: a hybrid adaptive routing protocol for mobile ad hoc networks," in *MobiHoc*, 2003, pp. 303–314.
- [8] M. Spohn and J. J. Garcia-Luna-Aceves, "Neighborhood aware source routing," in *Proceedings of the 2nd ACM International Symposium on Mobile Ad hoc Networking & Computing (MobiHoc)*. New York, NY, USA: ACM Press, 2001, pp. 11–21.
- [9] R. V. Boppana and S. Konduru, "An adaptive distance vector routing algorithm for mobile, ad hoc networks," in *Proceedings of the Twentieth Annual Joint Conference of the IEEE Computer and communications Societies*, 2001, pp. 1753–1762.
- [10] H. Füller, J. Widmer, M. Käsemann, M. Mauve, and H. Hartenstein, "Beaconless position-based routing for mobile ad-hoc networks," Department of Computer Science, University of Mannheim, Tech. Rep. TR-03-001, February 2003.
- [11] H. Lundgren, E. Nordström, and C. Tschudin, "Coping with communication gray zones in ieee 802.11b based ad hoc networks," in *WOWMOM '02: Proceedings of the 5th ACM international workshop on Wireless mobile multimedia*. New York, NY, USA: ACM Press, 2002, pp. 49–55.
- [12] I. D. Aron and S. K. S. Gupta, "A witness-aided routing protocol for mobile ad-hoc networks with unidirectional links," *Lecture Notes in Computer Science*, vol. 1748, pp. 24–33, 1999.
- [13] C. Sengul and R. Kravets, "Bypass routing: An on demand local recovery protocol for ad hoc networks," in *Med-Hoc-Net Workshop*, 2004.
- [14] S. Biswas and R. Morris, "Opportunistic routing in multi-hop wireless networks," *SIGCOMM Comput. Commun. Rev.*, vol. 34, no. 1, pp. 69–74, 2004.
- [15] C.-K. Toh, "Associativity-based routing for ad hoc mobile networks," *Wireless Personal Communications*, vol. 4, no. 2, pp. 103–139, 1997.
- [16] Scalable Network Technologies, Inc., "Qualnet simulator," Scalable Network Technologies, Inc., <http://www.scalable-networks.com>.
- [17] J. Yoon, M. Liu, and B. Noble, "Random waypoint considered harmful," in *INFOCOM. IEEE*, 2003.

*The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U. S. Government.