Having one's cake and eating it too: Better routes and lower control traffic for AODV

Zeki Bilgin
Department of Computer Science
The Graduate Center, CUNY
Email: zbilgin@gc.cuny.edu

Bilal Khan
Department of Math & Computer Science
John Jay College, CUNY
Email: bkhan@jjay.cuny.edu

***Abstract*—In this research we develop new techniques for optimizing the performance of a reactive routing protocol in operational environments characterized by high node mobility and long-lived connections. The question we seek to answer is whether in such environments, *reactive* routing protocols necessarily exhibit a tradeoff between control traffic and route optimality. More specifically, does a protocol which makes use of *less control traffic* (i.e. better) than standard AODV, necessarily exhibit connection routes that are *longer* (i.e. worse) than those achieved by standard AODV? We show that the commonly assumed tradeoff can be avoided, and that it is possible to "Have one's cake and eat it too". Towards this, we design an extension of the AODV protocol, and show through extensive ns2 simulation experiments that the new protocol both significantly reduces the control traffic overhead, while simultaneously improving the topological optimality of connections. These remarkable conclusions are seen to continue to hold scalably as one varies situational parameters such as network size, number of connections, and node mobility.**

I. Introduction

The problem of *routing* in mobile ad hoc networks (MANETs) has been a very active research topic in the networking community during the past two decades. There have been a tremendous number of routing protocols developed, each suited to particular settings and having its own merits and advantages over others. These routing protocols are commonly classified as either proactive, reactive, or hybrid (that is, a combination of the two types) depending on how and when routing information is obtained and maintained by constituent nodes.

Proactive routing protocols such as OLSR and DSDV follow in the footsteps of wired infrastructure routing protocols like OSPF: in such systems, nodes maintain next-hop routing information, regardless of whether route information is required by connection establishment requests. This route information is updated whenever a change in the underlying network topology occurs. In settings where nodes move frequently and connections are long-lived, these routing protocols may be inefficient due to high levels of routing-related control traffic generated in response to node movement that induces topological changes in the network structure.

On the other hand, reactive routing protocols such as DSR and AODV discover a routing information only when it is required (and cache discovered routes for a very brief interval of time). It is a common knowledge that reactive routing protocols are more appropriate for mobile ad-hoc networks in which node mobility levels are high and connections are long-lived, since these protocols delay systemic response to topology changes until routes are actually requested as part of the connection establishment process. Thus, in such settings, reactive routing protocols incur lower control traffic overhead than their proactive counterparts.

In this research we focus on the performance of reactive routing protocols. In view of the previous paragraphs, we will assume that the operating environment is indeed one which manifests high node mobility and long-lived connections. In a MANET, a reactive routing algorithm serves two functions: (i) to find an initial path between a source and a destination, and (ii) to maintain data forwarding between the two endpoints in the face of node mobility. We restrict our attention to (ii), and seek to reduce the operational cost of long-lived connection, vis-a-vis control traffic. More precisely, the research question we consider is:

> Is there an inherent tradeoff between route optimality and control traffic overhead, or is it possible to improve the performance of a reactive routing protocol with respect to both these metrics *simultaneously*?

Though most reactive routing protocols use similar triggering methods for the route discovery process, they show minor differences in the specific information maintained. For example, AODV keeps only the next hop for each requested destination (at each node) and does not embed any extra information into packet headers. In contrast, DSR uses a source routing strategy and inserts the whole transit list into packet headers. In this paper, for concreteness, we formulate our results as an extension to AODV.

II. Background

In AODV, when a node (source) requires a connection to another node (destination), a global route discovery operation is initiated by the source, resulting in a breadth-first flooding of ROUTE_REQUEST messages within the network. When (at least) one of these messages is received by the intended destination or by a node which has a route to the destination that is "fresh enough", a ROUTE_REPLY message is originated and sent back to the originator of the route discovery process. As the ROUTE_REPLY packet travels back towards to the originator of the request, each node along the route inserts next hop information into its routing table, for the requested destination. Once the source node receives the ROUTE_REPLY message, it can start to forward data towards the destination along the route established. Such a route discovery process is expensive in terms of control traffic incurred and route construction time taken due to requirement of network-wide flooding.

After the initial construction of a connection via the aforementioned process, two distinct types of undesirable events may arise as a consequence of node movement. These are (i) the connection may break because one of its consituent links fails when the link's endpoints moves out of the other's transmission range, or (ii) the connection may be topologically suboptimal because a path with fewer hops may become available as nodes move. These two types of events are illustrated in Figure 1. In this paper, we devise a *local recovery* scheme to address (i) while simultaneously introducing an event-triggered *route optimization* protocol to resolve issue (ii). In the next section, we begin by briefly describing prior work related to each of these two aspects of our proposed system.

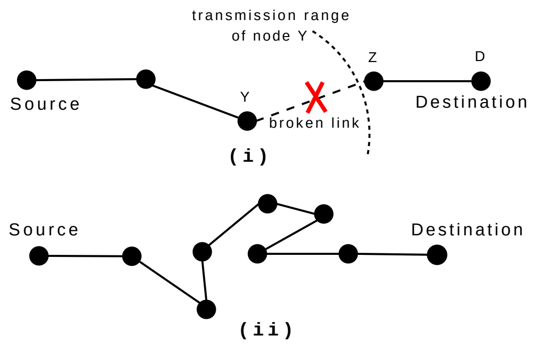

Fig. 1. A connection may (i) break or (ii) become topologically suboptimal.

III. Related Work

A. Local Recovery

Many researchers [1], [2], [3], [4], [5], [6] have proposed local recovery protocols, however, none of them presents a viable solution to the degradation in path optimality which necessarily occurs over time as successive local recovery operations take place along a route. We discuss he last three of these local recovery schemes in some detail, as they are representative of the types of approaches that have been considered to date.

Liu et al. propose a local recovery protocol, called PATCH [4] that is specifically developed for DSR. In their scheme, when a link on an active route becomes disconnected, the upper endpoint of the broken link initiates the local recovery operation by broadcasting a ROUTE_REQUEST packet with low TTL (e.g. TTL=2). Upon receiving such a ROUTE_REQUEST packet, the nodes located on the downstream side of the broken link on the route send back a ROUTE_REPLY packet to the initiator node in the opposite direction of the route that ROUTE_REQUEST packet followed. Once the ROUTE_REPLY packet arrives to the initiator of the local recovery operation, the route is patched. The main shortcoming of this approach is that path optimality degrades over time, because path length increases.

In a more recent study, Alshanyour and Baroudi [5] present another local recovery protocol called Bypass-AODV. In this scheme, whenever a link breaks at the MAC layer, the upstream endpoint of the broken link initiates the proposed local recovery protocol by broadcasting a bypass-RREQ message with TTL of 2. This bypass-RREQ is a modified route request message, that is sent *regardless of the position where the break occurred along the route* (in contrast to AODV's local recovery logic). In a response to a received bypass-RREQ, only the downstream endpoint of the broken link is allowed to send back a bypass-RREP (which is a modified route reply message). When the initiator node of the local recovery operation receives the bypass-RREP message, it updates its routing table, and starts transmitting buffered packets on the updated route. Like PATCH, the major drawback of Bypass-AODV is that it produces unnecessarily long routes, especially after several local recovery attempts have occurred.

Another recent paper by Shi and Deng [6] also proposes an improvement to AODV's default local repair protocol. In this work, the authors use "Hello" messages between neighbors (along an active connection) as a means for detecting possible link failure. If a node does not receive a Hello message from its neighbors for some time, it concludes that the corresponding link between them has failed, and a proposed local recovery protocol is initiated. The protocol is, however, based on the assumption that each node on a connection has knowledge of its upstream neighbor; regretably, this is not the case in AODV. Nonetheless, making this assumption, the authors propose that upon detecting link disconnection, the downstream endpoint of the broken link broadcasts an unsolicited Route Reply message (without waiting for any Route Request message) with the objective of reaching the upstream endpoint of the broken link through broadcasting via intermediate nodes. This approach is problematic because the overhead of continuous periodic "Hello" traffic is onerous in lower mobility settings (unless nodes can sense their position, though this introduces additional capabilities for the nodes).

B. Route Optimization

Several researchers have attempted to address the problem of dynamic connection optimization in MANETS which use a reactive routing protocol. Indeed, recent papers by Jia et al. [7], [8] present a model of the path compression problem, compares several existing path compression techniques, while presenting new approaches.

One of the seminal in this field is the work of Wu et al. [9] which considers several route optimization schemes for the routing protocols DSR, SSA, AODV, and ZRP. The authors exploit the promiscuous receive mode of wireless devices to collect fresh routing information. Their study aims to optimize active routes, and requires some modifications to packet headers and routing tables of the original routing protocols. Similar to the work of Wu et al, there are other studies [10], [11], [7] which exploit the promiscuous mode of the wireless cards to search for possible short-cuts on a route. These authors' strategies are frequently criticized because of their use of promiscuous mode, which may consume excessive power and thereby degrades the performance of the wireless cards.

In contrast, Zapata [12] presents a shortcut detection mechanism for active routes that does not rely on the promiscuous mode of the devices. In his proposed work, a special packet (Shortcut Request–SREQ) is broadcasted with TTL of 0 at each node along the route starting from the source node to the destination node. The SREQ packet includes IP addresses of the source, the destination, and the next hop along to route, in addition to corresponding distance in hops to both endpoint of the path. Since all 1-hop neighbors can receive a broadcasted SREQ packet, upon receiving such a packet, they check if there is a short-cut between the sender of SREQ packet and themselves by comparing the hop count metric for the same route if defined in their routing table. If so, the corresponding node sends back a SREP packet to inform sender of SREQ node about the possible shortcut. Upon receiving a SREP packet, a node modifies its routing table such that the next hop for the corresponding destination is specified as the sender of SREP packet. Regretably, the author's scheme has not been tested in extensive experiments, as is evidenced by the following oversignt in the protocol's design: A node may have a valid routing table entry for the destination and yet might no longer lie on the connection itself. If such a node responds to an SREQ by sending an SREP, it will cause the formation of an invalid route. Our own work in this paper uses a similar path optimization mechanism, but our scheme uses a protocol which is designed to sidestep the aforementioned pitfalls. Although our proposed route optimization protocol is similar to Zapata's work in some respects, there are several important differences: (i) our proposed route optimization protocol is event-triggered, in contrast to the periodic protocol proposed in [12], (ii) our protocol searches for the shortcuts to upstream nodes and thus sidesteps the aforementioned protocol design pitfall, and (iii) our protocol is validated by experimental studies conducted using packet-level simulations and ns2.

IV. System Design

The contributions of this paper are two-fold. First we describe a new *local recovery* scheme which responds to the event that a connection has broken because one of its consituent links failed when the link's endpoints moved out of the other's transmission range. This is presented in Section V. Second, we describe a novel *route optimization* protocol which resolves the issue that a connection may become topologically suboptimal over time, since a path with fewer hops may become available as nodes move. This is the subject of Section VI. Any system which implements *both* the local recovery scheme and the route optimization scheme is referred to as a **LocOpt** system. In Section VII, we design experiments which will enable us to compare the performance of **LocOpt** systems against those that adopt standard AODV. The results of these experiments are presented and analyzed in Section VIII.

V. Local Recovery Protocol

In this section, we describe an new alternative local recovery protocol for AODV. The objective of the proposed local recovery protocol is to patch a broken link on a route by splicing a *single* extra relay node into the route between the two endpoints of a broken link. The process is described with reference to the illustration in Figure 1. The local recovery process is initiated by the upstream endpoint of the broken link (i.e. by node Y) whenever the MAC layer reports a failure of link that is known to have been actively carrying outbound traffic towards some destination. Such a link failure report occurs after several unsuccessful MAC layer attempts to reach the next hop during a packet transmission. When this occurs, the initiator node (i.e. node Y) begins buffering data packets it has for the corresponding destination, and prepares a RREQUEST packet which includes the IP addresses of both the previously declared next-hop (i.e. node Z) and the final destination (node D). Notice that including the previously declared next-hop in a RREQUEST packet is a modification of standard AODV packet formats, introduced by us to support the proposed local recovery protocol. Then, node Y broadcasts prepared RREQUEST packet with TTL of 2, and waits for corresponding RREPLY packets while buffering incoming data packets for the destination under repair. Because we seek to reduce control traffic overhead (relative to what is incurred by standard AODV), in our protocol only two nodes are allowed to send back a RREPLY packet: These are the previously declared next-hop (i.e. node Z) and the final destination (node D). Because of this, the initiating node of the local recovery protocol will receive at most two RREPLY messages. If it receives both of them, the RREPLY coming from the final destination node is given priority since acting on it would result in a shorter connection route. Upon receiving a RREPLY packet, the initiator node updates its routing table and starts to send buffered data packets, in order, and thus the connection is repaired by splicing in an intermediate node.

The above local recovery operation is attempted at most twice. If the second attempt does not succeed either, then the initiator node drops all data packets that it has buffered for the corresponding destination, and sends an RERROR packet to the source node of the connection (i.e. node S). Upon receiving an RERROR packet along to route towards the source node, all nodes invalidate the corresponding routing table entries in their routing table. The source, upon receiving the RERROR, initiates a global route discovery process.

The reader might object such that successive local recovery operations can produce unnecessarily long paths, over time. Indeed, although the proposed local recovery protocol can resolve route disconnections in a more efficient, faster and cheaper way, it would certainly result in paths that were far from topologically optimal, as a side effect of continuously inserting new nodes into a route. To eliminate such a side effect, we will simultaneously introduce a *route optimization* protocol. However, before we can describe this protocol, we need to fix some of the inadvertent side effects of local recovery.

A. *Distance Update Mechanism*

In AODV, there is a dedicated field in the routing tables of the nodes which maintains the hop-count to destination. The values kept in this field must be strictly increasing along a route from destination to source in order to prevent loop formation during the route discovery process [13]. However, when an extra relay node is inserted into an active route (as is the case in the proposed local recovery operation above), this monotonicity may be violated because the field values at nodes that are upstream of the broken link are no longer correct and need to be increased by 1. To fix this problem, all nodes upstream of the broken link need to be informed about the new correct hop-count to the destination. The challenge is that nodes on the route are generally not aware of their upstream nodes, but rather only know the next (downstream) hop towards a specific destination. To address this difficulty, we developed the following mechanism:

When a local recovery operation is successfully completed, the initiator node of the local recovery operation prepares a special packet, called DISTANCE, which includes: the IP addresses of the source node (i.e. node S), the destination of the path (i.e. node D), and itself (i.e. node Y). The initiator (i.e. node Y) *broadcasts* this DISTANCE packet with TTL of 1. When a node receives DISTANCE packet, it infers that distance updating is to take place. To do this, the node checks its routing table to determine if the sender of the DISTANCE packet is the next hop for the specified destination in the packet. If it is, then it updates the hop-count to destination field in its routing table, and then broadcasts a similar DISTANCE packet in which its address is written into the sender field. Thus, the update process progresses towards source node using DISTANCE packets; the distance update process stops when it reaches the source node.

VI. Route Optimization Scheme

Our proposed route optimization scheme is called Multihop Shrinking. The objective of the scheme is to shorten unnecessarily long connections by eliminating inessential hops as illustrated in Figure 2. In order to do this, the mechanism checks if there is any direct shortcut between non-adjacent pairs of the nodes on the connection. More precisely, for each node along the connection, the mechanism checks whether there is any upstream node within the direct transmission range of the node of interest. If there is such a shortcut between two distant nodes on the same route, and the channel quality of this shortcut is "good enough", then the Multihop Shrink mechanism modifies the connection topology so that two end-point nodes of the short-cut connect to each other directly, thereby eliminating the inessential intermediary *node(s)* between them (see Figure 2). Such an operation has many potential advantages, including (i) relaxing path optimality issues in a local recovery protocol, (ii) reducing the end-to-end delay experienced in packet delivery by decreasing the number of hops on the path, (iii) increasing spatial reuse and network capacity by eliminating unnecessary transmissions. The protocol that achieves this will now be described, with the aid Figure 3.

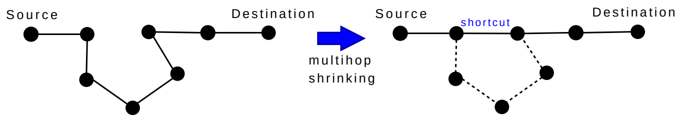

Fig. 2. The main idea of *Multihop Shrink* mechanism.

Initiation of the process: Multihop Shrinking is activated only *after* completion of "Local Recovery" (see Section V) and "Distance Update" (see Section V-A), as described earlier. More precisely, Multihop Shrinking is initiated when the source node (e.g. node S) of a path receives a DISTANCE packet, which happens because of the distance update mechanism that is initiated post local recovery. Once the source node receives a DISTANCE packet, it initiates the shrinking mechanism by sending a special SHRINK-0[1] packet to the next hop (e.g. node A) on the route.

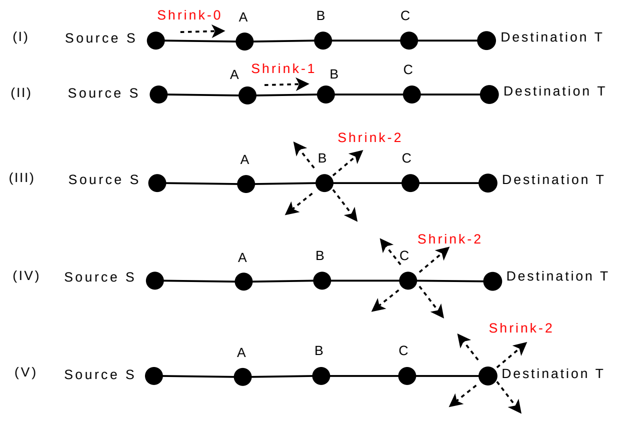

Fig. 3. Illustration of how multihop shrinking works

A Shrink packet contains the following fields:

- **sender**: The IP address of the sender of the Shrink packet (i.e. node S in the example),
- **next-hop**: The IP address of the next hop (i.e. node A in the example),
- **final-destination**: The IP address of the final destination (i.e. node T).
- **hops-to-destination**: The number of hops to the final destination (available in the routing table)
- **flag**: It can be 0, 1, or 2.

Once the second hop (e.g. node A) receives SHRINK-0 packet, it prepares a similar packet, SHRINK-1, and then sends it to the next hop (i.e. the third node) along the route.

[1]Zero represents a flag's value in the packet.

When the third node (e.g. node B) receives SHRINK-1 packet, it prepares a similar packet, SHRINK-2, and this time it *broadcasts* the SHRINK-2 packet with TTL of 1 (Notice that SHRINK-0 and SHRINK-1 were unicast packets). Likewise, the process progresses similarly towards the destination node.

From the vantage point of other nodes: A node which receives a Shrink-2 packet performs actions depending on its relative placement on the connection with respect to sender of the Shrink packet. A node, upon receiving the Shrink-2 packet, determines its relationship to the Shrink-2 packet, selected from the following set of five mutually exclusive classes:

- *The next hop:* A node identifies itself to be the *next hop* by noting that its own IP is the one specified in the next-hop field of Shrink-2 packet it received. In that case, it modifies the received Shrink-2 packet by updating the related fields using the information in its routing table, and then broadcasts the updated Shrink-2 packet.
- *Further downstream hops:* A node recognizes itself to be in this class if its routing table indicates that the hop count to the destination is smaller than the value in the hops-to-destination field of the received Shrink-2 packet. Nodes in this category discard the received Shrink-2 packet.
- *The previous hop:* A node identifies itself to be the *previous hop* when it realizes that in its routing table, the sender of the Shrink-2 packet is listed as the next hop to the final-destination. This node just discards the received Shrink-2 packet.
- *Further upstream nodes:* A node recognizes itself to be in this class if its routing table indicates that the hop count to the destination is greater than the value in the hops-to-destination field of the received Shrink-2 packet. If this is the case, then it can be concluded that there may be a short cut available between this node and the sender of Shrink-2 message. However, the quality of this new hypothesized link may not be good enough to warrant changing the routing table. Therefore, before doing any update to the routing table, the quality of the prospective new link is checked by looking at the signal strength at which the Shrink-2 packet was received. If the received signal strength is greater than a predefined threshold level, then the node updates its routing table in such a way that the next hop for the final-destination (as specified in the Shrink-2 packet) is replaced with the address of the sender the Shrink-2 packet.
- *Irrelevant nodes:* When a node receives Shrink-2 packet, but there is no next hop in its routing table for the final-destination specified in the Shrink-2 packet, then the node classifies itself as irrelevant to the Shrink-2 packet, which is simply ignored.

Every node receiving a Shrink-2 packet falls into precisely one of the above five classes, and behaves as specified. In practice, there can be many data flows (i.e. many connections) between different source/destination pairs, all active simultaneously. Such a situation does not alter the operation of proposed mechanism, since each node responds to different Shrink messages arising from different transient connections independently. No further state needs to be maintained at each node than what is mandated in AODVs routing table format.

VII. Experimental Setup

We call the whole proposed protocols LocOpt. The LocOpt is implemented as an extension to the standard implementation of AODV in ns-2.33. The performance of the original AODV and LocOpt are compared for the following network size, mobility models, and traffic/connection pattern as follows:

Networks: Several network sizes are investigated, comprised of between 50 and 150 nodes. Node density is kept fixed at 50 nodes per 700 m x 700 m. Initial placement of nodes is uniformly random.

Traffic Patterns: The traffic connections are initiated between randomly chosen source and destination pairs chosen uniformly from the nodes. The number of connections varied between 1 and 50, depending on the type of experiment. Traffic sources generate constant bit rate (CBR) traffic consisting of packets of size 512 bytes, at a rate of 4 packets per second.

Mobility Model: To investigate the performance of the proposed mechanism under different mobility levels, we modified random waypoint (RWP) mobility model as follows: First we set the pause time to zero (nodes move without stopping between subsequent movements). Then we generated a movement plan with maximum speed of 5 m/s. To obtain higher mobility scenarios, we multiplied the velocity of each node by $\beta > 1$. The advantage of this modified RWP is that as $beta$ is increased, the same sequence of (link level) networks arise, but evolved at a faster rate. We considered $\beta = 1, 2, 3, 4, 5$, with larger values of β signifying higher mobility scenarios. Note that under this mobility model, the simulation duration changes inversely with β. For example, when maximum speed is 5 m/s the simulation duration is 1200 seconds, while it is 600 seconds when the maximum speed is 10 m/s. Our performance metrics (see below) are robust to this fact.

Performance metrics: The following three metrics are evaluated:

- *Normalized Routing load* (NRL)–the number of routing-related control packets transmitted *per data packet delivered to its destination*.
- *Packet delivery fraction* (PDF)– The percentage of data packets submitted at a source, that were delivered to their intended destination.
- *Path Optimality* (PO)–For each data packet delivered to its destination, we note both the number of hops that packet traveled and the length of the optimal source-destination path at the time of packet delivery. Then we calculate the discrepancy, both as ratio and as a difference.

Trials: To increase our confidence in conclusions drawn from the analysis of simulations, we repeated up to 40 trial experiments for each setting of the independent variable (with mobility plans and traffic patterns generated as above). We then computed the mean and standard deviations of the performance metrics (dependent variables), and examined how these varied as the independent variable was changed.

VIII. Results

Normalized Routing Load (NRL):

We begin by considering normalized routing load (NRL), that is, the number of routing-related control packets transmitted *per data packet delivered to its destination*. We investigated NRL of AODV and LocOpt under different assumptions of node mobility level, network size, and traffic load. Figure 4 depicts the relationship between NRL and mobility levels It is clear from the figure that the proposed scheme reduces NRL of the network regarless of node mobility level. For example, when nodes are moving at 25m/s, AODV utilizes 3.25 control packets per data packet delivered, while LocOpt uses only 2.4. This is because the proposed mechanism decreases the number of global route discovery attempts performed in the network. More importantly, as node mobility increases, the advantage of the proposed scheme over AODV *increases*, implying that the proposed solution scales as networks become increasingly more mobile.

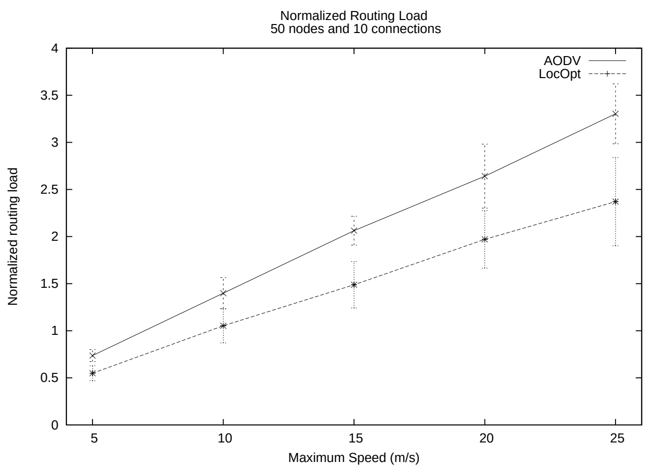

Fig. 4. NRL vs. Mobility

In the second set of NRL experiments, we consider the effect of network size. For these experiments, we normalized AODV's NRL with respect to the LocOpt scheme's NRL. As seen in Figure 5, the relative advantage of the LocOpt scheme is significant and *increases* as the network size gets larger. We note that because we keep node density fixed in our experiments, increasing network size implicitly increases the geometric extent of the network. When the network has 50 nodes, AODV uses 1.5 times as many control packets (per delivered data packet) as LocOpt does; but when the network has 1500 nodes, AODV uses more than 2.5 times as many control packets (per delivered data packet) as LocOpt does. The relative advantage gained by the LocOpt scheme increases for larger networks because the importance of local recovery is more critical in large networks where the cost of global route discovery are much more expensive.

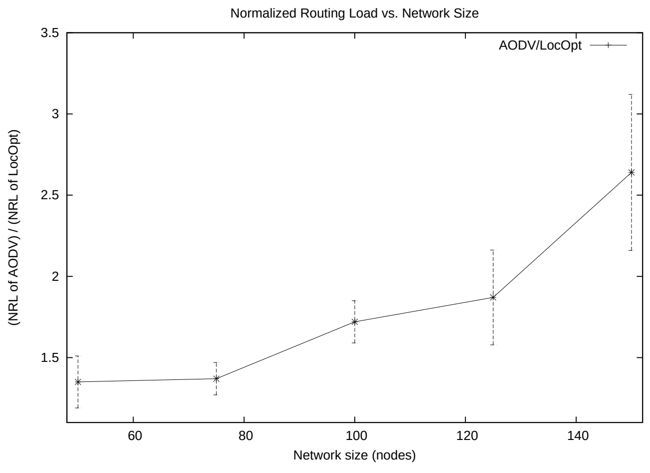

Fig. 5. NRL vs. Network size

In the third set of NRL experiments, we consider the effect of data traffic load. For these experiments, we normalized the AODV's NRL with respect to LocOpt scheme's NRL. As seen in Figure 6, there is a tradeoff between the relative advantage of the LocOpt scheme and traffic load. The curve decreases until the network becomes saturated, and after this point, it begins to increase for higher traffic loads. The reason of the decrease in the graph is that the number of successful local recovery operations decreases as data traffic crowds out the control traffic required to execute local recovery. On the other hand, the increase in the graph is due to an increase in packet delivery fraction (recall that NRL is a normalized by PDF), as seen in Figure 7.

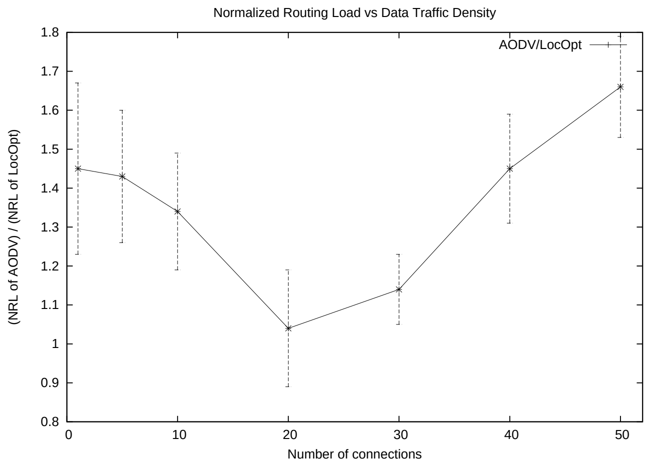

Fig. 6. NRL vs. Traffic load

Packet Delivery Fraction (PDF): Figure 7 shows PDF for pure AODV and LocOpt. The graph shows that the proposed mechanism improves the packet delivery fraction at all velocity levels. For instance, while PDF is about 97% for pure AODV at 15m/s case, it rises to up to 98% when the LocOpt scheme is applied. This increase in PDF, while not very significant, arises from the fact that the proposed mechanisms decrease packet losses which may occur due to connection failures, by applying a local recovery operation to every link failure, regardless of its relative location on the route (this is in stark contrast to AODV's standard local recovery logic).

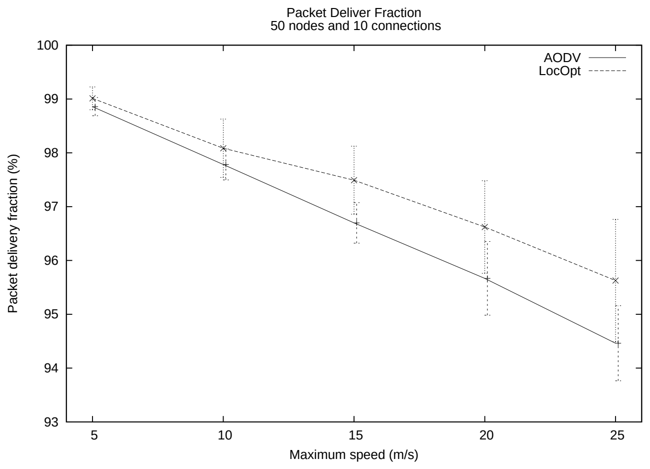

Fig. 7. Packet Delivery Fraction vs. Mobility

Path Optimality: We see from the preceding analysis that the LocOpt scheme is capable of significantly reducing normalized traffic overhead without compromising on packet delivery fraction. We can have our cake apparently, but can we eat it too?

Another important performance metric for a local recovery protocol is how it affects the path optimality. Most if not all previous attempts at crafting a local recovery mechanism have failed to address the issue of route degradation. This is a well-known side effect of local recovery [5], [4]. To investigate this issue, in our final set of experiments, we calculated for each data packet delivered to its destination, the number of hops that the packet traveled. We compared this with the number of hops on the optimal source-destination path (at the time of packet delivery). The ratio of the first quantity to the second was taken as the normalized path length for the packet. We computed the normalized path length for all delivered packets, and showed the mean and standard deviations of these values. Figure 8 below shows that the normalized path lengths of AODV are higher than those provided by LocOpt. Thus, in contrast to previously proposed local recovery protocols, LocOpt does not harm path optimality, but rather, outperforms even standard AODV. Obviously, this is because of we have incorporated route optimization in addition to a local recovery mechanism.

IX. Conclusion

We presented new techniques for optimizing the performance of a reactive routing protocol in operational environments characterized by high node mobility and long-lived connections. We designed an extension of the AODV protocol, and showed through extensive ns2 simulation experiments that this new protocol both significantly reduces the control traffic overhead, while simultaneously improving the topological

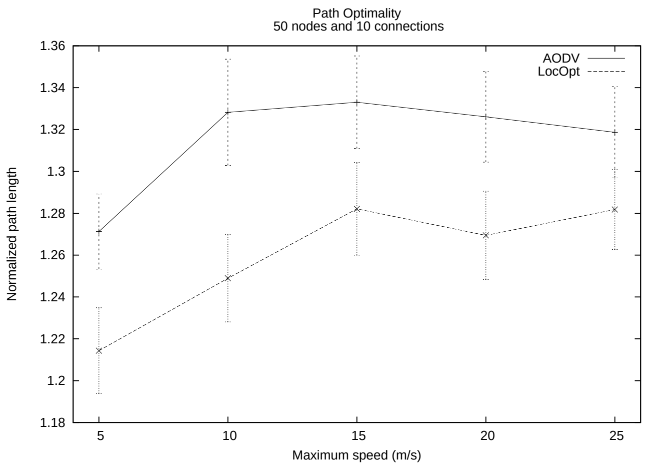

Fig. 8. Distribution of path length discrepancy from optimal.

optimality of connections. The proposed protocol makes use of *less control traffic* than standard AODV while exhibiting connection routes that are *shorter* than those achieved by standard AODV. Thus, we showed that reactive routing protocols do not contain an inherent tradeoff between control traffic and route optimality. These conclusions were shown to hold scalably as one varies situational parameters such as network size, number of connections, and node mobility.

Acknowledgments

The first author thanks the Turkish National Police for supporting his graduate studies at City University of New York, of which this research is a part.

References

[1] S. Medidi and J. Wang, "A fault resilient routing protocol for mobile ad-hoc networks," in *WIMOB '07: Proceedings of the Third IEEE International Conference on Wireless and Mobile Computing, Networking and Communications*. Washington, DC, USA: IEEE Computer Society, 2007, p. 41.

[2] S. M. Lee and K. Kim, *An Effective Path Recovery Mechanism for AODV Using Candidate Node*. ISPA Workshops: LNCS, 2006, ch. 4331, pp. 1117–1125.

[3] M. Pan, S.-Y. Chuang, and S.-D. Wang, "Local repair mechanisms for on-demand routing in mobile ad hoc networks," in *PRDC '05: Proceedings of the 11th Pacific Rim International Symposium on Dependable Computing*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 317–324.

[4] G. Liu, K.-J. Wong, B. sung Lee, B. chong Seet, C. heng Foh, and L. Zhu, "Patch: A novel local recovery mechanism for mobile ad-hoc networks," in *Proc. of IEEE Vehicular Technology Conference*, 2003, pp. 2995–2999.

[5] A. M. Alshanyour and U. Baroudi, "Bypass aodv: improving performance of ad hoc on-demand distance vector (aodv) routing protocol in wireless ad hoc networks," in *Ambi-Sys '08: Proceedings of the 1st international conference on Ambient media and systems*. ICST, Brussels, Belgium, Belgium: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008, pp. 1–8.

[6] R. Shi and Y. Deng, "An improved scheme for reducing the latency of aodv in ad hoc networks," in *ICYCS '08: Proceedings of the 2008 The 9th International Conference for Young Computer Scientists*. Washington, DC, USA: IEEE Computer Society, 2008, pp. 594–598.

[7] Y.-S. Yen, H.-C. Chang, R.-S. Chang, and H.-C. Chao, “Routing with adaptive path and limited flooding for mobile ad hoc networks,” *Computers and Electrical Engineering*, vol. 36, no. 2, pp. 280 – 290, 2010, wireless ad hoc, Sensor and Mesh Networks. [Online]. Available: http://www.sciencedirect.com/science/article/B6V25-4W5M06W-1/2/1e001d0aa105c95138a5da479322829c

[8] X. Jia, L. Qian-mu, Z. Hong, and L. Feng-yu, “Model and analysis of path compression for mobile ad hoc networks,” *Computers and Electrical Engineering*, vol. In Press, Corrected Proof, pp. –, 2009. [Online]. Available: http://www.sciencedirect.com/science/article/B6V25-4Y0TDNG-1/2/4f2123dd3d3af8581723a8958225a936

[9] S. Wu, S. Ni, Y. Tseng, and J. Sheu, “Route Maintenance in a Wireless Mobile Ad Hoc Network.” In 33rd Hawaii International Conference on System Sciences, Maui, 2000.

[10] C. Gui and P. Mohapatra, “Short: self-healing and optimizing routing techniques for mobile ad hoc networks,” in *MobiHoc ’03: Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing*. New York, NY, USA: ACM, 2003, pp. 279–290.

[11] V. C. Giruka, M. Singhal, and S. P. Yarravarapu, “A path compression technique for on-demand ad-hoc routing protocols,” in *In Mobile Ad-hoc and Sensor Systems (MASS) 2004*, 2004.

[12] M. G. Zapata, “Shortcut detection and route repair in ad hoc networks,” *Pervasive Computing and Communications Workshops, IEEE International Conference on*, vol. 0, pp. 237–242, 2005.

[13] C.Perkins, E.B-Royer, and S.Das, “Ad hoc on-demand distance vector (aodv) routing.” Internet Engineering Task Force, 2003.