Combining Source- and Localized Recovery to Achieve Reliable Multicast in Multi-Hop Ad Hoc Networks

Venkatesh Rajendran[1], Katia Obraczka[1], Yunjung Yi[2], Sung-Ju Lee[3], Ken Tang[4] and Mario Gerla[2]

[1] Computer Engineering Department, University of California, Santa Cruz
[2] Computer Science Department, University of California, Los Angeles
[3] Mobile & Media System Laboratory, Hewlett-Packard Laboratories
[4] Scalable Networks Technologies

Abstract. This paper proposes a novel reliable multicast transport protocol for multi-hop, wireless ad hoc networks (or MANETs). To recover from the different types of losses that may occur in MANETs, our Reliable Adaptive Congestion-controlled Transport protocol, or ReACT, combines source-based congestion- and error control with receiver-initiated localized recovery. While the latter attempts to recover localized losses (e.g., caused by transmission errors), the former is invoked only for losses and congestion that could not be recovered locally (e.g., caused by global congestion). Loss differentiation is an important component of ReACT and uses medium access control (MAC) layer information to distinguish between different types of losses. Through extensive simulations, we evaluate ReACT's performance under a variety of MANET scenarios, including different offered load and mobility conditions, and compare it against a strictly end-to-end (i.e., no localized recovery) scheme. Our results show that ReACT is the best performer in terms of reliability. Our results also showcase the effect of ReACT's local recovery mechanism which quickly corrects error- and path breakage induced losses and thus manages to prevent the source from reducing its rate unnecessarily , thus achieving significant throughput improvement with lower overhead when compared to the strictly end-to-end protocol.

1 Introduction

A multi-hop, wireless ad hoc network (or MANET) [1] operates without any fixed infrastructure. Hosts communicate with each other through wireless packet radios. Because of the limited radio propagation range, routes can often be multi-hop. Hence, every host may act as a packet forwarder as well as source or destination of traffic. Because of their ease of deployment, MANETs are an attractive choice for scenarios where the fixed network infrastructure is non-existent or unusable. Example applications include search and rescue, disaster recovery, digital battlefield, and covert military operations.

Both unicast- and multicast routing in MANETs have been well-studied and, as a result, a number of protocols have been proposed [2, 16]. Several research efforts have also focused on transport-layer approaches to achieve end-to-end reliable point-to-point communication. This includes the work on improving TCP performance in "last-hop" wireless networks and MANETs [3–6].

However, the types of scenarios targeted by MANETs make group-oriented services such as data dissemination and teleconferencing a key application domain. In particular, the mission-critical characteristics of a number of these applications (e.g., emergency response, special civilian or military operations) call for efficient **reliable** multi-point communication protocols for MANETs. Undoubtedly, "network-supported" multicast communication is an efficient means of supporting group-oriented applications. This is especially true in MANETs where nodes are energy- and bandwidth limited. In these resource-constrained environments, reliable point-to-point protocols (that may be viable in wired networks) can get prohibitively expensive: the convergence of multiple requests to a single node typically causes intolerable congestion, violating the reliability and time constraints of a critical mission, and may drain the node's battery, cutting short the network's lifetime. Despite the fact that it is a key enabling technology for mission critical applications in MANETs, surprisingly very few efforts to date focus on reliable multicast transport.

The Reliable Broadcast Protocol [7] addresses the problem of reliable atomic delivery of messages. While this protocol may work well in stable networks with low mobility and low failure rates, its performance will likely degrade in dynamic MANET scenarios where topology changes are frequent. Anonymous Gossip (AG) [8] recovers from losses by having pairs of multicast group participants exchange information on messages they have received or lost. AG uses solely local recovery from nearby members for error control. In our experiments, we compare the performance of ReACT against AG [5]. As expected, mainly due to the fact that AG does not implement congestion control, we observe that its performance deteriorates under heavy load.

In our previous work we demonstrated the importance of congestion control in improving reliability. Congestion-controlled Adaptive Lightweight Multicast (CALM) [10] is a multicast transport protocol that tries to achieve reliable delivery strictly through congestion control. The Reliable Adaptive Lightweight Multicast protocol [11] uses a congestion control scheme similar to that of CALM and recovers from losses using source-based retransmissions. It requires multicast group member information to perform congestion control and error recovery. In an extended version of RALM [12], we do away with the need to maintain group membership information at the source.

Several features unique to MANETs make the design of MANET reliable multicast transport mechanisms quite challenging. Among these features, we highlight: (1) MANET's heterogeneous loss characteristics due to factors such as mobility, node density, time-varying channel conditions, (2) effects of lower layer protocols, e.g., inherent unfairness and unreliability of contention-based medium access control protocols (e.g., IEEE802.11 [13] uses plain CSMA when broadcasting packets and thus do not provide reliable broadcast delivery), and (3) MANET's extreme sensitivity to offered load.

These MANET features render design choices used in reliable multicast protocols for wired networks not at all applicable to MANET environments. Based on observations from our prior work [10], we argue that multicast reliability in MANETs cannot be achieved solely by retransmission of lost packets as is typically done in wired networks with protocols such as Scalable Reliable Multicast (SRM) [14]. Our premise is that, besides error control, effective reliable multicast delivery in MANETs must also

[5] These results that are not presented here due to space limitations can be found in [9]

perform congestion control. As demonstrated in our previous studies [10–12], a simple congestion control scheme results in significant increase in delivery guarantees.

MANETs' complexity also calls for revisiting the layered system design argument which claims that, in a system, the design and implementation of each one of its layers should not be exposed to higher layers. We argue that in MANETs, information obtained from lower layers of the protocol stack is crucial for adequate performance at higher layers.

This motivated us to explore cross-layer mechanisms to achieve efficient reliable multicast transport. More specifically, we use information from lower layer protocols (in particular the MAC layer) to perform loss differentiation addressing MANETs' heterogeneous loss characteristics. Thus, some of the distinguishing features of ReACT are that (1) it combines source-based rate control with local error recovery and (2) uses loss differentiation to trigger either source-based control or local recovery. The goal is to recover from localized losses (e.g. due to node mobility, link quality, channel contention) using nearby group members, while congestion losses are reported to the source, triggering error- as well as congestion recovery.

Through extensive simulations, we evaluate ReACT's performance under a wide range of MANET conditions. In order to demonstrate the benefits of ReACT's loss differentiation and local recovery mechanisms, we also compare its performance against a strictly source-based control scheme (RALM [12]). In our experiments, as the underlying routing mechanism, we use a mesh-based multicast protocol, more specifically the On-Demand Multicast Routing Protocol (ODMRP) [15].

The remainder of this paper is organized as follows. Section 2 presents a detailed description of ReACT's source-based and local recovery mechanisms. Performance evaluation and simulation results follow in Section 3. Section 5 presents our concluding remarks and directions for future work.

2 ReACT

2.1 Overview

Our premise when designing ReACT is that in wireless environments losses may be caused by various factors and should be handled differently. For example losses caused by transmission errors (e.g., due to factors such as noise, interference, etc.) or hidden terminal collisions may be affecting only a small number of nodes in a neighborhood and thus can be recovered locally using a (non-congested) near-by member, i.e., without the involvement of the source. There is no need to trigger congestion control and slow down the source because these losses are not indicative of "global" congestion. Furthermore, by recovering locally, feedback and retransmissions are kept in the affected neighborhood and do not add to traffic destined to the source, hence improving protocol efficiency. On the other hand, congestion losses should be reported to the source triggering reduction of the sending rate as well as error recovery. However, special care should be taken as local recovery can exacerbate congestion if the network neighborhood performing recovery is already congested.

ReACT performs receiver-based loss differentiation to distinguish congestion- from local losses. A multicast receiver samples its MAC queue to detect congestion building up. Receivers also detect congestion building up anywhere on the path from the source by having intermediate nodes set a “congestion” flag in multicast data packets they forward. The congestion flag is set by any intermediate node whose MAC queue grows beyond a certain fraction of the maximum MAC queue size. By detecting incipient congestion (instead of waiting to take action until actual packet drops occur), ReACT tries to avoid persistent congestion conditions.

ReACT ensures that only multicast members that are situated in a non-congested area will be used to perform local recovery. This avoids contributing to congestion in an already congested neighborhood. The remainder of this section describes ReACT in detail by presenting its two main components, namely source-based (error and congestion) control and receiver-based error recovery.

2.2 Source-Based Control

ReACT employs a rate-based congestion control scheme that has two main modes of operation: initial rate set-up (i.e., determining the initial sending rate) [6], and congestion control. ReACT tries to determine the appropriate sending rate in order to avoid (1) initial bandwidth under utilization by starting too low, and (2) congestion by starting at too high of a rate.

One approach at setting up the initial rate is to probe the entire network and then decide on the rate based on the aggregate network condition. Though this approach can provide information on the overall state of the network, it is not scalable. Alternatively, in ReACT we establish the initial rate based on the set of members that are directly connected to the source. This provides the source with an estimate of its neighborhood’s current conditions. The rate is decided so as to satisfy the worst receiver in this neighborhood.

The first data packet sent by the multicast source serves as a probe packet and each directly connected member replies with a *PROBE_REPLY* packet.After sending the first packet, the source waits for *PROBE_WAIT_TIME* to receive replies to its probe. *PROBE_WAIT_TIME* is set based on the network diameter (*NET_DIAMETER*) and an estimate of average time to traverse one hop (*NODE_TRAVERSAL_TIME*) accounting for queuing and transmission delays (similar to the route reply timeout of AODV [17]). If the source does not hear from any receiver in response to the probe packet, it will continue to send (probe) packets every *PROBE_WAIT_TIME* interval. The source then computes the inverse of the largest round-trip time reported during the initial probing period and uses that as its initial sending rate.

The rate is periodically updated once every *PROBE_INTERVAL* by directly connected neighbors. If no feedback has been sent to the source for the last *PROBE_INTERVAL* seconds, the receiver generates an explicit *PROBE_REPLY* packet. Receivers only send an update to the source if they detect significant changes to the time it takes them to get packets from the source. *PROBE_INTERVAL* is set sufficiently large to prevent oscillations in the source sending rate and also to reduce feedback overhead due probing. The

[6] In our previous approaches [10–12], we start at the application sending rate and then react to congestion based on receiver feedback. Our experiments indicate that setting an initial rate too high may lead to extreme (sometimes unrecoverable) congestion and thus numerous packet losses (e.g., if the feedback path from the receivers to the source gets blocked.)

source continues to send at this rate, until it hears a negative acknowledgment (NACK) from any receiver experiencing congestion. In that case, it reverts to congestion control.

ReACT's congestion control works as follows. The source initially multicasts data packets at the rate decided using initial probing as described above. Upon reception of a NACK, the source adds the NACK sender to its *Receiver List* and enters loss recovery. The missing sequence numbers reported by the NACK are added to a global retransmission list, which is an aggregate of lost sequence numbers from all reporting receivers. This list is updated whenever the source retransmits a packet to prevent duplicate retransmissions. In addition, the source keeps track of the end-to-end latency between itself and each receiver that sent NACKs.

The source initiates loss recovery by selecting a receiver from the *Receiver List*, which we call *Feedback Receiver*. The source then retransmits a lost packet requested by the *Feedback Receiver* or multicasts a new packet (e.g., if all lost packets requested by that receiver had already been retransmitted). The packet header includes information instructing the *Feedback Receiver* to reply via "unicast" with a (positive) acknowledgment (ACK) indicating that all packets have been successfully received or specifying the sequence number(s) of packets that are still missing. All other receivers process the packet without replying to the source.

The source then responds by retransmitting the requested packets one at a time until the *Feedback Receiver* receives all packets (i.e., send-and-wait). The design philosophy behind retransmitting one packet at a time is to slow down the source when congestion is detected. Since only the *Feedback Receiver* replies to the source, the ACK/NACK implosion problem is avoided. NACKs are rate-limited to prevent excessive feedback overhead. The mechanism for controlling NACK generation is described in Section 2.3.

Once the *Feedback Receiver* obtains all packets, it unicasts an ACK to the source indicating successful reception of all packets. Upon reception of the ACK, the source removes the node from the *Receiver List*, chooses a new *Feedback Receiver* in a round robin fashion, and repeats this process until the *Receiver List* is empty.

When the *Receiver List* is empty, the source reverts to the latest sending rate decided based on periodic probe packets. If, however, the source does not receive a NACK or ACK from the *Feedback Receiver* within the time interval given by the measured round-trip time from the source to the *Feedback Receiver*, the source backs off and tries again up to a maximum number of times (which is three in our simulations) before removing it from the *Receiver List*. The removed receiver may later re-synchronize with the source through the normal NACK mechanism.

The round-robin send-and-wait approach does not require retransmissions of the same lost packets multiple times to each receiver. In the best-case scenario, lost packets are retransmitted only once by the source since retransmissions are multicast. For instance, if a set of receivers lost the same packet, it is retransmitted only once assuming the retransmitted packet is received by all the receivers. In the worse-case scenario, each receiver experiences different packet losses. In this case, all lost packets must be retransmitted to each receiver.

2.3 Receiver-Based Error Recovery

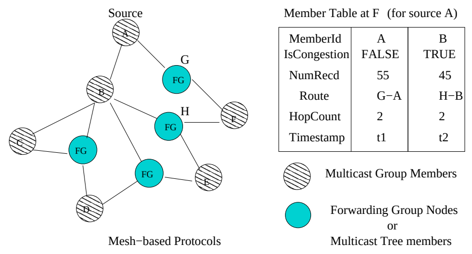
The main goal of ReACT's receiver-based recovery mechanism is to detect losses that can be recovered locally avoiding source involvement (and hence avoid triggering congestion control). Congestion losses, however, should be reported to the source so that it knows to slow down.

In order to recover from losses "locally", nodes must obtain information about other group members as potential *Recovery Nodes*. Our scheme gathers member information using multicast data packets as they get forwarded over the multicast tree or mesh. Hence it is independent of the underlying multicast routing protocol. We are only interested in *Recovery Nodes* that are in the forwarding path from the source. More specifically, ReACT only uses immediate upstream member node(s) for recovery.

The way recovery requests and replies (or retransmissions) are routed has significant impact on the overall performance of the reliable multicast mechanism. If the underlying unicast routing protocol does not have a valid path for the recovery request and performs flooding for route discovery, significant additional load may result. Our simulation study indicates that local recovery based protocols that do not address this problem (e.g., AG) suffer from congestion even at moderate loads. ReACT employs a source routing approach that makes use of valid cached paths. The main advantage of this approach is that it makes ReACT independent of the underlying unicast routing protocol. The tradeoff is the overhead involved in maintaining source routes, especially in highly mobile environments. ReACT restricts the maximum distance between a member and its *Recovery Node* to *LR_ROUTE_LEN* hops to reduce the failure probability (e.g., due to node mobility) of source route.

Every node maintains a *Member Table* that stores information about current *Recovery Nodes*. To account for route volatility, *Member Table* entries are assigned an expiration time (*LR_VALID_TIME*). Additionally, each node maintains a metric of *reliability*, i.e., the rate at which it receives multicast data packets from *Recovery Nodes*. This information is used in selecting a *Recovery Node* if multiple ones exist. Each entry also has a flag to indicate if the path to the *Recovery Node* is congested. This flag is set if any intermediate node on the path to the *Recovery Node* has MAC queue size beyond the *CONGESTION_THRESHOLD*.

The IP option fields in the multicast data packet is used to carry route, hop count and congestion information. These fields are updated as the packet is forwarded to the group. The *route* field contains the path traversed by the packet from the upstream multicast group member. The *hopCount* field carries the length of the path. The *isCongestion* field denotes if any of the node in the path is congested. Whenever a node decides to perform local recovery, it selects a non-congested member that has the highest receive rate, lowest hop-count, and latest timestamp. Figure 1 shows a sample *Member Table* maintained by member node F when using either a tree- or mesh-based protocol. Mesh-based protocols may yield more than one upstream member because of path redundancy. As tree-based protocols also use broadcast for delivery, it is possible that a receiver might receive a packet from a node other than its parent node. Selecting an upstream *Recovery Node* based on its reliability and proximity increases the likelihood of successful local packet recovery.



MemberId	A	B
IsCongestion	FALSE	TRUE
NumRecd	55	45
Route	G−A	H−B
HopCount	2	2
Timestamp	t1	t2

Fig. 1. Member table maintained for local recovery

Feedback generation is rate-limited to once every $MIN\_FEEDBACK\_INTERVAL$ seconds to prevent excessive feedback overhead. Thus, every $MIN\_FEEDBACK\_INTERVAL$, receivers check if they need to perform error recovery by sending a NACK to a nearby member. There is a tradeoff in setting $MIN\_FEEDBACK\_INTEVAL$. When smaller intervals are used, we observe higher packet delivery ratios at the expense of higher overhead and lower throughput.

A NACK packet consists of an request array that is filled with the node's missing sequence numbers. The NACK is then sent to the selected *Recovery Node* if losses are found to be localized. Nodes use cached source routes to communicate with the *Recovery Node*. On the other hand, if losses are due to "global congestion" or if the node finds that it is experiencing congestion, then it sends the NACK to the source using the underlying unicast routing protocol.

A node checks if losses are due to "global congestion" by examining the paths to *Recovery Nodes*. If all paths are congested, then all valid *Recovery Nodes* in the *Member Table* will have the *isCongestion* flag set. Additionally, the node also examines its queue to check if it is congested. If any of the above conditions is true, then losses are classified as due to "global" congestion and feedback is sent to the source directly triggering congestion control.

Besides missing sequence numbers, a NACK packet destined to the source also includes the average delay multicast packets take to reach the node from the source. Receivers update the average delay to a multicast source every time they receive a data packet from that source. The average delay is computed as an exponential average with more weight to recent measurements. Receivers sending NACKs to the source are placed in the *Receiver List*. The source then transmits to each *Receiver List* receiver based on its reported delay using a send-and-wait approach as described in Section 2.2.

3 Experimental Setup

As our simulation platform, we use the QualNet network simulator [19]. ODMRP [18] and AODV [17] are used as the underlying multicast and unicast routing protocols, respectively. The transmission range for the radio is $447.807m$ with a data rate of $2Mbps$. The MAC protocol used is IEEE 802.11 DCF [13].

We evaluate the performance of ReACT in comparison with plain Reliable Adaptive Lightweight Multicast (RALM) [12], a strictly source-based control scheme. We

Table 1. Simulation parameters

ReACT Parameters	Value
LR_ROUTE_LEN	3
LR_VALID_TIME	3 s
MIN_FEEDBACK_INTERVAL	5 s
NODE_TRAVERSAL_TIME	50 ms
NET_DIAMETER	35
PROBE_INTERVAL	50 s

evaluate ReACT's performance subject to a wide range of network conditions. We are particularly interested in how ReACT performs under various offered loads, and what is the impact of node mobility. Table 1 shows the values of the parameters used by ReACT.

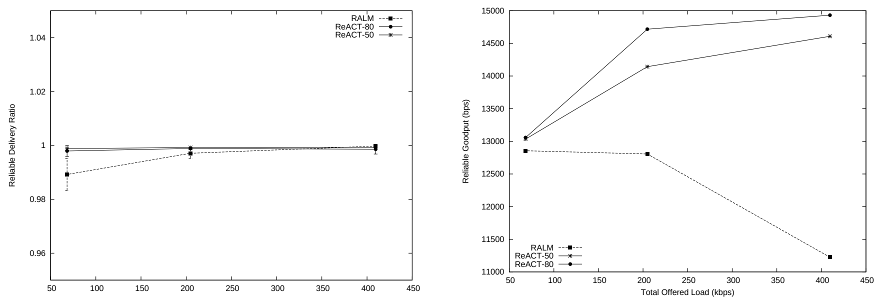
As we target applications that require the highest possible delivery guarantees, protocol reliability is a critical performance metric. We define *Reliable Delivery Ratio* as the fraction of packets successfully (or reliably) delivered to ALL receivers over the total number of packets sent. We also measure *Reliable Goodput* defined as the throughput of packets reliably delivered, i.e., packets that are received by all members. Finally, we measure the overhead incurred by the protocols. To account for control packets sent by underlying unicast/multicast protocols, we measure the total number of packets sent by each node at the MAC layer. *Normalized Overhead* is thus computed as the ratio of total packets sent at the MAC layer to total data packets delivered to all members. This measures the total number of packets transmitted to successfully deliver one data packet to all members.

First, we study the importance of congestion control by simulating a scenario with multiple sources generating different traffic loads and then we analyze the impact of node mobility. For these sets of experiments, 50 nodes are placed randomly in a $1500m \times 1500m$ field and 10 randomly chosen nodes join the multicast group. These group nodes join at the start of the simulation and stay subscribed to the group till the end of the simulation. Five randomly selected members continuously send CBR traffic throughout the whole duration of the simulation with payload size of 512 bytes. The results are averaged over several runs and presented with 95% confidence.

4 Simulation Results

4.1 Effect of Congestion

Figure 2(a) shows reliable delivery ratio under different loads. The error bars correspond to reliable delivery ratio's 95% confidence intervals. As the load increases, so does packet loss due to congestion and hidden terminal collisions. Both RALM and ReACT perform error recovery and congestion control and hence they achieve very high reliability. RALM employs strictly source-based error recovery using NACKs. NACKs are generated when lost sequence numbers are detected upon arrival of a new data packet. Hence, if a node stops receiving data from a particular source, it will never generate NACK to recover lost packets. This leads to the reduced reliability of RALM at low loads and also higher reliability variance when compared to ReACT. On the other hand, ReACT achieves perfect reliability under various loads due to its robust error recovery mechanism.



(a) Reliable delivery ratio (b) Average reliable goodput

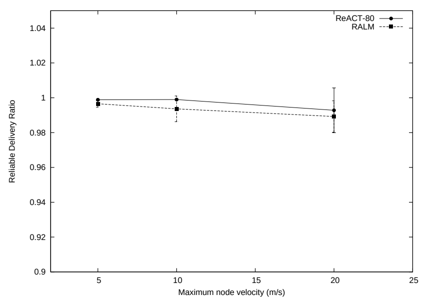
Fig. 2. Effect of congestion

We show results for two different versions of ReACT: one that detects congestion when queues grow above 80% of their maximum size, while the other version uses 50% as the queue threshold indicating congestion. As observed from Figure 2(a), both versions deliver almost perfect reliability.

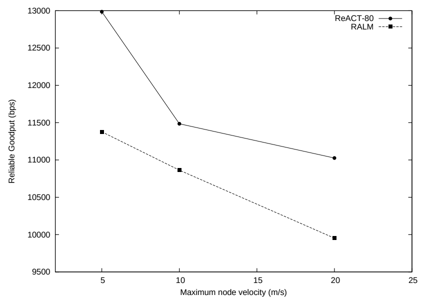
Figure 2(b) illustrates the impact of ReACT's combined local- and source-based recovery mechanisms on goodput. We observe that ReACT achieves considerably higher (reliable) goodput when compared with RALM. Furthermore, ReACT is able to keep its goodput steady even at higher loads, while RALM suffers severe degradation at higher traffic rates. This is mainly because local recovery prevents the source from backing off its rate when packet losses are recovered locally. It should also be noted that RALM starts sending at the application rate and then performs congestion control when it receives feedback from the receivers. This aggressive behavior can potentially lead to severe congestion preventing NACKs from receivers to reach the source. This is one of the reasons for RALM's reliable goodput degradation as we increase the load. On the other hand, ReACT's initial setting of the sending rate also contributes to its high reliable goodput.

Figure 2(b) also illustrates the effect of *CONGESTION_THRESHOLD*, which is set to 80% and 50% of the maximum MAC queue size. As expected, goodput is lower at 50% as ReACT becomes more conservative, generates feedback sooner and thus causes more frequent rate decreases.

Normalized overhead for RALM and ReACT are depicted in Figure 4(a). ReACT incurs significantly lesser overhead than RALM due to its local recovery mechanism at higher loads. It prevents unnecessary source retransmissions (which are multicast) for errors that are recoverable locally. Source route caching used by local recovery also helps to reduce the overhead incurred by local recovery. Otherwise, route discovery flooding by the underlying unicast routing protocols can significantly increase the total overhead. However, at low loads, probe replies sent by receivers for updating the source sending rate and the corresponding route discovery initiated by AODV slightly increases the overall normalized overhead.



(a) Reliable delivery ratio



(b) Average reliable goodput

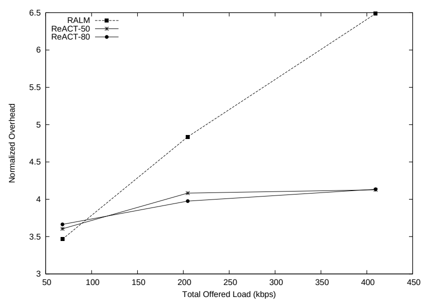
Fig. 3. Effect of mobility

4.2 Effect of Node Mobility

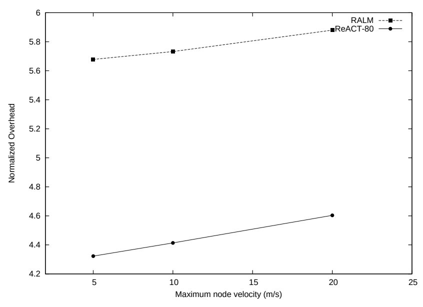
In these experiments, we use the random-way-point mobility model with no pause time and $0m/s$ minimum speed. We vary maximum speed from $5m/s$ to $20m/s$. The total network load injected in these mobile scenarios is $200Kbps$.

Figure 3(a) shows the reliable delivery ratio achieved by RALM and ReACT for different node velocities. Both RALM and ReACT are able to achieve perfect reliability even at high mobility. As previously discussed, the slight variation in RALM's reliable delivery ratio is due to the NACK generation mechanism driven by received data. As expected, Figure 3(b) shows that both protocols exhibit degradation in goodput as we increase node mobility. In ReACT, the sending rate is updated based on the measured delay reported by the probed set of receivers. As we increase node mobility, the delay experienced by nodes becomes highly variable. ReACT uses the highest delay reported to update the sending rate, which can substantially reduce throughput. Thus, probing more frequently can improve the goodput with increased node mobility, at the expense of increased overhead due to probe replies. ReACT's local recovery mechanism is able to recover locally some mobility-induced losses, and thus achieves higher goodput than RALM.

As shown in Figure 4(b), ReACT's overhead is significantly lesser than RALM for all mobility conditions. This is mainly due to ReACT's local recovery mechanism which recovers mobility losses locally. As we increase mobility, ReACT's overhead increases as its local recovery effectiveness decreases: mobility leads to frequent timeouts of source routes maintained in the *Member Table*. This invalidates potential *Recovery Nodes* and forces receivers to resort to the source for error recovery. As previously discussed, this effect also contributes to the reduction in goodput with increased mobility. In our future work, we plan to overcome this problem by expanding cross-layer interaction and using more information from the lower layers (e.g., unicast routing).



(a) Effect of congestion



(b) Effect of mobility

Fig. 4. Normalized overhead

5 Conclusion and Future Work

In this paper, we presented ReACT, an adaptive, congestion-controlled multicast transport protocol for reliable and timely multicast delivery in MANETs. One of ReACT's main distinguishing feature is its combination of source-based control and local recovery. ReACT's source-based control includes initial rate setup and congestion recovery which adjusts the sending rate using a simple stop-and-wait mechanism based on receivers' feedback.

Through simulations, we evaluated ReACT's performance and compared it with RALM, a strictly source-based protocol. Our results show that ReACT significantly improves both goodput and packet delivery with lower overhead. By way of its congestion control mechanism, ReACT is able to deliver perfect reliability under a wide range of conditions. We also demonstrate the benefits of ReACT's local recovery mechanism which prevents the source from reducing its rate unnecessarily and restricts the scope of receiver feedback yielding reduced protocol overhead.

In our future work, we will focus on improving the efficiency of ReACT at higher node mobility scenarios by extending the interaction with the underlying unicast routing layer. As a follow-on to our initial study on layer interaction, we will also investigate the interactions between the transport- and the MAC layer. In particular, we will investigate the synergy between ReACT and MAC protocols that provide link-level reliability for broadcast/multicast data.

References

1. Perkins, C.E.: Ad Hoc Networking. Addison Wesley (2001)
2. Broch, J., Maltz, D.A., Johnson, D.B., Hu, Y.C., Jetcheva, J.: A performance comparison of multi-hop wireless ad hoc network routing protocols. In: Proceedings of the Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking. (1998)
3. Kim, D., Toh, C.K., Choi, Y.: TCP-BuS: Improving TCP performance in wireless ad hoc networks. Journal of Communications and Networks **3** (2001)

4. Sun, D., Man, H.: ENIC - an improved reliable transport scheme for mobile ad hoc networks. In: Proceedings of IEEE GLOBECOM 2001, San Antonio, TX (2001)
5. Liu, J., Singh, S.: ATCP: TCP for mobile ad hoc networks. IEEE Journal on Selected Areas in Communications **19** (2001) 1300–1315
6. Sundaresan, K., Anantharaman, V., Hsieh, H.Y., Sivakumar, R.: ATP: A reliable transport protocol for ad-hoc networks. In: Proceedings of ACM MobiHoc, Annapolis, MD (2003)
7. Pagani, E., Rossi, G.P.: Reliable broadcast in mobile multihop packet networks. In: Proceedings of the third annual ACM/IEEE international conference on Mobile computing and networking, ACM Press (1997) 34–42
8. Chandra, R., Ramasubramanian, V., Birman, K.P.: Anonymous Gossip: Improving multicast reliability in mobile ad-hoc networks. International Conference on Distributed Computing Systems (2001) 275–283
9. Rajendran, V.: Reliable multicasting in ad hoc networks. Master's thesis, University of California (2003)
10. Tang, K., Obraczka, K., Lee, S.J., Gerla, M.: Congestion controlled adaptive lightweight multicast in wireless mobile ad hoc networks. Proceedings of IEEE ISCC (July 2002)
11. Tang, K., Obraczka, K., Lee, S.J., Gerla, M.: A reliable, congestion-controlled multicast transport protocol in multimedia multi-hop networks. In: Proceedings of IEEE WPMC 2002. (2002)
12. Tang, K., Obraczka, K., Lee, S.J., Gerla, M.: Reliable adaptive lightweight multicast protocol. In: Proceedings of IEEE ICC 2003. (2003)
13. IEEE: Wireless LAN medium access control (MAC) and physical layer specifications. ANSI/IEEE Standard 802.11, 1999 Edition (1999)
14. Floyd, S., Jacobson, V., Liu, C.G., McCanne, S., Zhang, L.: A reliable multicast framework for light-weight sessions and application level framing. IEEE/ACM Transactions on Networking **5** (1997) 784–803
15. Lee, S.J., Su, W., Gerla, M.: On-demand multicast routing protocol for multihop wireless mobile networks. ACM/Kluwer Mobile Networks and Applications **7** (2002) 441–453
16. Lee, S.J., Su, W., Hsu, J., Gerla, M., Bagrodia, R.: A performance comparison study of ad hoc wireless multicast protocols. In: INFOCOM (2). (2000) 565–574
17. Perkins, C.E., Royer, E.M.: Ad-hoc on-demand distance vector routing. In: Proceedings of IEEE WMCSA, New Orleans, LA (1999) 90–100
18. Lee, S.J., Gerla, M., Chiang, C.C.: On-demand multicast routing protocol. Proceedings of IEEE WCNC (1999)
19. Scalable Networks: http://www.scalble-networks.com.