The Case for FEC-based Reliable Multicast in Wireless Mesh Networks

Dimitrios Koutsonikolas Y. Charlie Hu
School of Electrical and Computer Engineering
Center for Wireless Systems and Applications
Purdue University, West Lafayette, IN 47907
{dkoutson, ychu}@purdue.edu

Abstract

Many important applications in wireless mesh networks require reliable multicast communication. Previously, Forward Error Correction (FEC) techniques have been proved successful for providing reliability in the Internet, as they avoid the control packet implosion and scalability problems of ARQ-based protocols. In this paper, we examine if FEC can be equally efficient in wireless mesh networks. We implement four reliable schemes initially proposed for wired networks on top of ODMRP, a popular unreliable multicast routing protocol for wireless networks. We compare the performance of the four schemes using extensive simulations. Our results show that the use of pure FEC can offer significant improvements in terms of reliability, increasing PDR up to 100% in many cases, but it can be very inefficient regarding the number of redundant packets it transmits. Moreover, a carefully designed hybrid protocol, such as RMDP, can maintain the same high level of reliability while improving the efficiency by up to 38% compared to a pure FEC scheme.

1. Introduction

An important characteristic of many multicast applications in wireless mesh networks, such as software updates or audio/video file downloads, is that they require reliable packet delivery. Many reliable multicast protocols have been proposed for multihop wireless networks (see [1] for a survey). Some of them require feedback from all receivers, leading to the well-known problem of feedback implosion, which in turn leads to congestion. Others trade off throughput for reliability, which may be acceptable for MANET applications (e.g., search-and-rescue operations), but not for commercial mesh networks. Finally, gossip-based protocols try to mitigate these problems, however they cannot guarantee 100% reliability.

All the above protocols fall into the class of Automatic Repeat reQuest (ARQ) schemes. Forward Error Correction (FEC) has been proposed as an alternative way to provide reliable multicast in the Internet [2]. With FEC the sender sends redundant encoded packets, and the receiver can reconstruct the original data even if it receives only a fraction of the encoded packets. Hence, no feedback from the receivers is required. A carefully selected amount of redundancy can result in less overhead compared to requests and retransmissions required in ARQ schemes. However, FEC itself cannot guarantee 100% reliability. Hence, hybrid ARQ-FEC schemes, e.g., [3, 4, 5, 6, 7, 8] have been proposed to combine the advantages of both techniques.

These protocols have been evaluated in the Internet, although some of them (e.g. [4]) are claimed to work also for wireless environments. However, the wireless environment is very different from that of the wired networks. (1) In the wired Internet the main reason for losses is congestion; in wireless networks the time variability of the wireless channel is equally important. Obstacles, interference, and multipath fading lead to bursty losses, possibly decreasing the effectiveness of FEC. (2) Most wireless multicast protocols use MAC-layer broadcast to exploit the Wireless Multicast Advantage [9]. However, the commonly used 802.11 MAC protocol does not incorporate any reliability mechanism for broadcast, as opposed to for unicast, where link-layer acknowledgments and retransmissions hide some losses from the upper layer and the RTS/CTS exchange reduces the number of losses due to collisions. (3) The asymmetry of wireless links does not provide any guarantee that requests for retransmissions sent by receivers will reach the source. (4) Bandwidth is a limited resource in wireless networks, and hence a solution that provides a large amount of redundancy may not be applicable. (5) Most wireless multicast protocols use a mesh to deliver data, as opposed to the tree-based approach in the Internet. This invalidates many of the assumptions made by Internet multicast protocol design. (6) Mesh-based multicast protocols can increase reliability, since there are usually more than one paths available from the source to each of the receivers. In summary, it is entirely not obvious that the FEC-based schemes designed for the Internet will work well in wireless networks.

The above discussion argues for the need for a detailed evaluation of the existing FEC and hybrid FEC-ARQ schemes in a multihop wireless mesh network. In this paper, we compare four different reliable multicast schemes, borrowed from the wired Internet, and examine how efficiently they perform under the special characteristics of a multihop wireless mesh network. One of the four schemes is a pure-FEC based scheme, where the source just sends some amount of redundancy and no feedback is required from the receivers. Two other schemes are hybrid FEC-ARQ protocols, namely NP [3] and RMDP [4]. Finally, for completeness, we include in our comparison a pure ARQ-based scheme, ReMHoc [10], which follows the design principles of Scalable Reliable Multicast (SRM) [11], one of the most popular ARQ-based protocols for the wired Internet. ODMRP (On Demand Multicast Routing Protocol) [12], a widely used multicast protocol for wireless networks, is used as the underlying routing protocol for the four schemes.

Our results show that (1) ARQ-based protocols have a very poor performance in wireless environments, (2) the use of pure FEC can offer significant improvements in terms of reliability, increasing the PDR up to 100% in many cases, but it can be very inefficient in terms of packet transmissions, (3) a carefully designed hybrid protocol, such as RMDP, can maintain the same high level of reliability and also improve the efficiency by up to 38% compared to a pure FEC scheme.

2. Overview of Reliable Multicast Protocols

In this section we describe an ARQ-based protocol, a pure FEC-based protocol, and two hybrid protocols. All four protocols are designed to be superimposed on an unreliable multicast routing protocol.

2.1. Automatic Repeat Request

We selected ReMHoc [10] as a representative ARQ protocol for our comparison. ReMHoc follows the design principles of SRM [11], perhaps the most popular ARQ protocol for reliable multicast in the wired Internet. ReMHoc is receiver-initiated; each receiver is responsible for detecting loss, by detecting gaps in the packet sequence numbers. When a packet loss is detected, the receiver schedules a *Request* packet, asking for retransmission of the lost packet. To prevent the implosion of control packets, receivers wait for a random period of time before sending a request for a lost packet. If they receive a request for the same packet from another receiver before their timer expires, they postpone their own request by resetting the timer. This backoff is exponential in SRM, but linear in ReMHoc (proportional to the number of times this request has already been scheduled). This is because the loss rate is much higher in wireless networks than in the Internet, and hence faster response is required.

If the timer expires, a request is sent. But there is no guarantee that the request itself will not be lost, or that the repair packet will reach this receiver. Hence, the request timer is reset. In ReMHoc there is no upper bound on the number of times a request can be sent. However, we found that by allowing infinite number of requests, the control overhead grows too fast and the PDR is reduced. Therefore, we decided to allow up to five retransmissions of the same request. After requesting a packet for five times, a receiver considers this packet permanently lost and no further action is taken for that packet in the future.

Request packets are multicast toward the whole group. Any multicast member that receives a request packet and has the requested packet, sends a *Repair* packet and does not propagate the request further. Similarly as for the request packets, a node postpones its transmission of a repair packet for a random period of time, and cancels it if in this time it hears another node retransmitting the same repair packet. Each repair packet is multicast to the whole group, so that all nodes that are missing the same data packet can recover by using the same repair packet.

2.2. Forward Error Correction

The key idea behind Forward Error Correction (FEC) [2, 13] is that k data packets are encoded at the sender to produce n encoded packets, where $n > k$, in such a way that any subset of k encoded packets suffices to reconstruct the original k data packets. Such a code is called an (n, k) FEC code and allows the receiver to recover from up to $n - k$ packet losses in a group of n encoded packets. A code is called systematic when the first k encoded packets are the original data packets. Systematic codes are much cheaper to decode and they allow partial reconstruction of data even when fewer than k encoded packets are received. In a systematic code, the $n - k$ encoded packets that are different from the original k data packets are called *parities*.

In this paper we use a particular class of FEC codes, called Reed-Solomon (RS) codes, which uses Vandermonde matrices to encode the data packets. [2] gives a description and a software implementation of RS codes which is used by many protocols, such as [3, 4]. It works for packet sizes up to 1024 bytes, and the proposed values for k and n are 32 and 255, respectively.

2.3. NP

NP [3] is a hybrid ARQ-FEC based protocol that uses systematic Reed-Solomon codes. In NP the data file at the sender is separated in transmission groups (TGs), and an (n, k) RS code is applied to each TG. Hence, each TG consists of k original data packets and $n-k$ parities.

In NP the transmission of each TG proceeds in rounds, which can be interleaved with the transmission of packets from other transmission groups. In round i the k original data packets of group TG_i are sent. Then in each following round j, $j > i$, l_i^{j-1} new parity packets are sent, where l_i^{j-1} is the maximum number of packets lost by any of the receivers in round $j-1$ for group TG_i, until all parities are exhausted.

In more detail, the sender transmits k data packets for TG_i followed by a POLL message POLL(i, k). With POLL(i, k), the sender informs the receivers that it has sent k packets for TG_i in the current round, and it asks for feedback about the missing packets of TG_i. Then it continues by sending the data packets for group TG_{i+1}. When it receives a NACK(i, l), it interrupts sending data packets of TG_m, $m > i$, it transmits l new parity packets for group TG_i, followed by a new POLL message POLL(i, l), and it then resumes transmission for TG_m.

For TG_i, each receiver stores data packets and parities until it collects at least k packets, which allow the reconstruction of TG_i. When a POLL(i, s) is received, NACK(i, l) is scheduled to be transmitted in the interval $[(s-l)T_s, (s-l+1)T_s]$, where l is the number of packets still missing for TG_i. When the timer for NACK(i, l) expires, NACK(i, l) is multicast to the whole group. If a NACK(i, m) is received with $m \geq l$ before the timer for NACK(i, l) expires, the timer is canceled.

Assuming an ideal environment (no NACK or POLL packets are lost) and a tree structure, the slotting and dumping mechanism of the protocol assures that the sender will receive only one NACK after every data/parity-POLL round, which will contain the maximum number of packets needed by any receiver to reconstruct TG_i.

2.4. RMDP

Reliable Multicast data Distribution Protocol (RMDP) [4] is the only protocol claimed to work in wireless environments, although it was only evaluated on the MBone. In RMDP the data file at the sender is again divided into B transmission groups of k packets each, and an (n, k) RS code is applied to each group. However, differently from in NP, in RMDP the TGs are not transmitted sequentially, but their packets are interleaved as shown in Figure 3 of [4]. The sender maintains a counter c_{s_i} for each TG_i, counting the number of packets (original plus parities) it has to send for TG_i. Similarly, each receiver maintains a counter c_{r_i} for each TG_i, counting the number of packets received for TG_i. The protocol uses only one control message type: a request $R = [f, c_{r_i}, i]$ is sent by a receiver to inform the sender that it has received c_{r_i} packets of TG_i of the file f. The sender augments each data packet p_j of TG_i with a file identifier f, the total number of original data packets constituting TG_i k, the current value of c_{s_i}, and a sequence number j. We will denote such an augmented data packet as $S = [f, k, c_{s_i}, j, p_j]$.

Whenever the sender receives a message R $[f, c_{r_i}, i]$, it sets $c_{s_i} = max(c_{s_i}, D(k - c_{r_i}))$ for TG_i where D is the amount of redundant data sent unconditionally ($D = 1$ means that redundant packets are only sent on demand). If $c_{s_i} > 0$ the sender has to transmit c_{s_i} packets (original plus/or parities) for TG_i . For example, in the beginning, $c_{s_i} = 0$ at the sender, and $c_{r_i} = 0$ at each receiver, hence the sender will set $c_{s_i} = D \times k$ upon receiving a message R, i.e, it has to send $D \times k$ packets for each TG. As we mentioned before, the sender sends the packets using TG interleaving, i.e., every τ seconds it transmits a packet from a different TG, where τ is the interarrival packet time imposed by the application.

Each receiver sends initially an R message with $c_r = 0$, and then it schedules a new request every T_R seconds. If it receives a data message S, it cancels any pending request. Then if the number of packets received for at least one TG is less than k, it schedules a new request after T_C seconds for the TG with the largest number of missing packets.

The largest difference between NP and RMDP, other than TG interleaving, is the way the sender reacts to the reception of request messages (NACKs or R messages, respectively). In NP, the sender responds immediately to each NACK, by sending the amount of packets requested. In RMDP, the sender takes no immediate action, but it updates the appropriate counter c_{s_i}, increasing the amount of packets it has to transmit in the future for TG_i.

3. Evaluation

3.1. Methodology

ODMRP For our comparison we used On Demand Multicast Routing Protocol (ODMRP) [12] as the underlying multicast protocol, and implemented ReMHoc, FEC, NP, and RMDP on top of it. ODMRP is a best-effort multicast protocol originally designed for mobile ad hoc networks; receivers do not attempt to recover from packet losses.

To keep the route discovery control overhead low, ODMRP does not send explicit Join Query packets (for refreshing multicast routes), but periodically incorporates this information in the header of the data packets. This implies that not all data packets are equally important; the ones that carry Join Queries are more important than the rest, since if such a packet is lost, no route is built and subsequent data packets are also lost. This creates a conflict with FEC's principle, where *any* k out of n packets suffice to decode the original data. Hence in this paper, we decided to decouple the tree construction/maintenance from the data transmission and use explicit Join Query packets. The original ODMRP uses short route refresh intervals (the interval between two successive route discoveries) to maintain the connectivity in the presence of node mobility. In mesh networks, routers are stationary, and hence routes are much more stable than in MANETs. Therefore, we increased the values for route refresh interval and FG timeout (the time after which a forwarding node stops forwarding packets) to reduce the control overhead.

Scenario We used the Glomosim [14] simulator in our simulation study. Glomosim is a widely used wireless network simulator with a detailed and accurate physical signal propagation model. We simulated a network of 50 static nodes placed randomly in a $1000m \times 1000m$ area. We used two multicast groups with nine receivers and one source each. In this way, the traffic for one group acts as cross-traffic for the other group. Each source sent a 4MB file, consisting of 512-byte packets at a constant rate of 5 packets/sec in a low-load scenario and 20 packets/second in a high-load scenario[1].

The radio propagation range was 250m and the channel capacity was 2Mbps (the data rate used for broadcast in 802.11 MAC protocol). The *TwoRay* propagation model was used. To make the simulations realistic, we added fading in our experiments. The Rayleigh model was used, as it is appropriate for environments with many large reflectors, e.g., walls, trees, and buildings, where the sender and the receiver are not in Line-of-Sight of each other. We envision that such environments will be common for mesh networks. We simulated each protocol on 10 different randomly generated topologies and the results for each topology as well as the average over all topologies are presented.

For the implementation of ReMHoc, FEC RS codes, NP, and RMDP, we followed [10], [2], [3] and [4], respectively. For the protocol parameters that are given specific values in these four papers, we kept the same values. For the rest of the parameters (e.g., the slot size T_s in NP), we ran simulations with different values, and we selected those that resulted in the best reliability (i.e. packet delivery ratio) without suffering too much on throughput (both metrics are defined below). These parameters along with the selected values are shown in Table 1.

Evaluation metrics The following metrics are used to evaluate the various reliable multicast protocol versions:

[1]Although the source sending rate is only 80 Kbps, the actual load on the network is much higher as taking the node density into account, the total traffi c load within a transmission range is on average 800 Kbps.

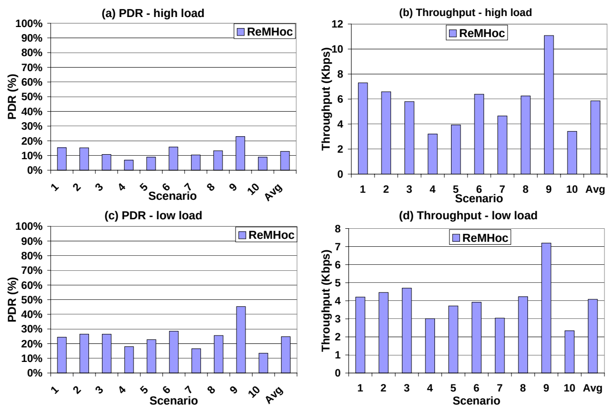

Figure 1. PDR and throughput for ReMHoc under high and low load for 10 different topologies.

Average Packet Delivery Ratio (PDR): The number of the data packets delivered to a multicast group member divided by the number of data packets transmitted by the source, averaged over all multicast group members. For FEC-based protocols, the data packets refer to the original data packets, i.e., before encoding at the source and after decoding at the destination. This metric directly measures reliability.

Average Throughput: The number of data packets (in bytes) delivered to a multicast group member divided by the total time required for this delivery, averaged over all multicast group members.

Efficiency: The number of decoded packets at a multicast group member divided by the total number of encoded packets received by that member, averaged over all multicast group members. It is a common metric in FEC-based protocols. For ARQ-based protocols there is no redundancy and efficiency is 100%, since all data/repair packets received are useful packets.

3.2. Results

ARQ We first evaluate the performance of ReMHoc, a pure ARQ protocol. Figure 1 shows the average PDR and throughput achieved by ReMHoc under high and low load.

As we observe in Figures 1(a), 1(c), the PDR is very low under both high and low load. On average, ReMHoc achieves only 13% PDR under high load, and slightly higher (23%) under low load. This PDR is much lower than that achieved by ODMRP without any reliability scheme (74% and 87%, respectively, undel high and low load). In other words, a reliable protocol achieves much lower reliability compared to a best-effort protocol which does not have any reliability characteristics. Same observations can be made for throughput which is extremely low, 5.9Kbps and 4Kbps, under high and low load, respectively.

Table 1. Parameters of the various reliable protocols and their values.

Protocol	Name - Description	Notation	Value
ReMHoc	number of times a request is sent	-	5
FEC	number of original data packets in each TG	k	32
	number of total (original + encoded) data packets in each TG	n	63,127,255
NP	number of original data packets in each TG	k	32
	number of total data packets in each TG	n	255
	slot size for NACK timers	T_s	1sec (NP), 5sec (NP_opt)
RMDP	number of original data packets in each TG	k	32
	number of total data packets in each TG	n	255
	timer for next R message after R is sent	T_R	20sec
	timer for next R message after S is received	T_C	10sec
	amount of redundant data sent unconditionally	D	1,3,5

Tang et al. observe a similarly low PDR for SRM in [15]. The poor performance of ReMHoc (or SRM) stems from the way ARQ protocols try to recover from packet losses. For each lost packet, additional request and repair packets are injected into the network, which finally leads to congestion and packet dropping. Of course request/repair suppression tries to mitigate this problem. However, suppression does not work well in wireless networks for two reasons. First, it is very possible that request packets will also be lost, like data packets. This is usually not taken into account in the wired Internet. Second, packet losses in wireless multicast are highly uncorrelated, as opposed to in the Internet. In IP multicast, a tree is built from the source to the receivers. Hence, a packet loss observed by a node is also observed by all its downstream nodes [11]. In wireless networks, most multicast protocols build a mesh instead of a tree to increase reliability. Hence, there are usually several paths from the sender to each receiver, which result in uncorrelated losses. This is true even for tree-based protocols, because of the broadcast nature of a wireless transmission.

Things get even worse due to bursty losses. We observed in ReMHoc bursty losses of up to 90 consecutive data packets. Such bursts of lost packets cause bursts of requests, and subsequently, bursts of repairs sent by the source. This results in the MAC layer queues of the nodes one hop away of the source being fully occupied by repair packets for certain periods of time, during which all new data packets are simply dropped. On average, each receiver sent 1260 requests under high load but it received only 472 repair packets. The same numbers for low load were 3564 and 1149, respectively. This shows that the request-repair mechanism does not perform well in wireless networks.

In general, the ARQ model for the Internet performs very poorly in wireless networks. The conclusion is that an efficient reliable protocol for wireless multicast should avoid or at least limit the transmission of request packets, and be able to react to bursty losses without causing congestion in the network.

FEC In this section we evaluate the performance of pure FEC, where no feedback is sent by the receivers. Rizzo in [4] proposes the use of a (255, 32) RS code as a good compromise between encoding/decoding speed and efficiency. However, this implies a large amount of redundancy (seven times the file size). In a wireless network of limited capacity, this might not be a good choice, as high traffic will compete with traffic from other sources. Hence, we examine the use of smaller values for n, while keeping the value for k equal to 32. Figure 2 shows the results for PDR, throughput, and efficiency in case of high and low load, for three different amounts of redundancy, namely for n = 63, 127, and 255.

As we observe in Figures 2(a), 2(d), the use of FEC significantly improves reliability. For high load, the use of (63, 32), (127, 32), and (255, 32) codes achieves on average PDRs of 89.5%, 98.4%, and 98.9%, respectively. For low load, the PDR with a (63, 32) code increases up to 97.9%, while for the other two codes it remains the same. More importantly, (127, 32) and (255, 32) codes give a 100% PDR in 5 and 8 out of 10 scenarios, respectively, under high load, and in 8 out of 10 scenarios under low load.[2]

Figures 2(b), 2(e) show that the average throughput achieved is the same for a (127, 32) and a (255, 32) code, equal to 46Kbps under high load and 15.5Kbps under low load. Taking into account the fact that the sending rate is 81.9Kbps and 20.5Kbps, under high and low load, respectively, we conclude that 100% reliability comes at the cost of reduced throughput, especially under high load. It also comes at the cost of very low efficiency, as we observe in Figures 2(c), 2(f). The efficiency of a (127, 32) code is on average 36% and 29% under high and low load, respectively. For a (255, 32) code it decreases even more – 19% and 14%, respectively. It means that a lot of packets are

[2]In both scenarios 5 and 6, where PDR is less than 100% even with a (255, 32) code, the reason is that one node is almost isolated from the rest of the network, due to the random placement of nodes. Such pathological cases should not happen with a planned deployment.

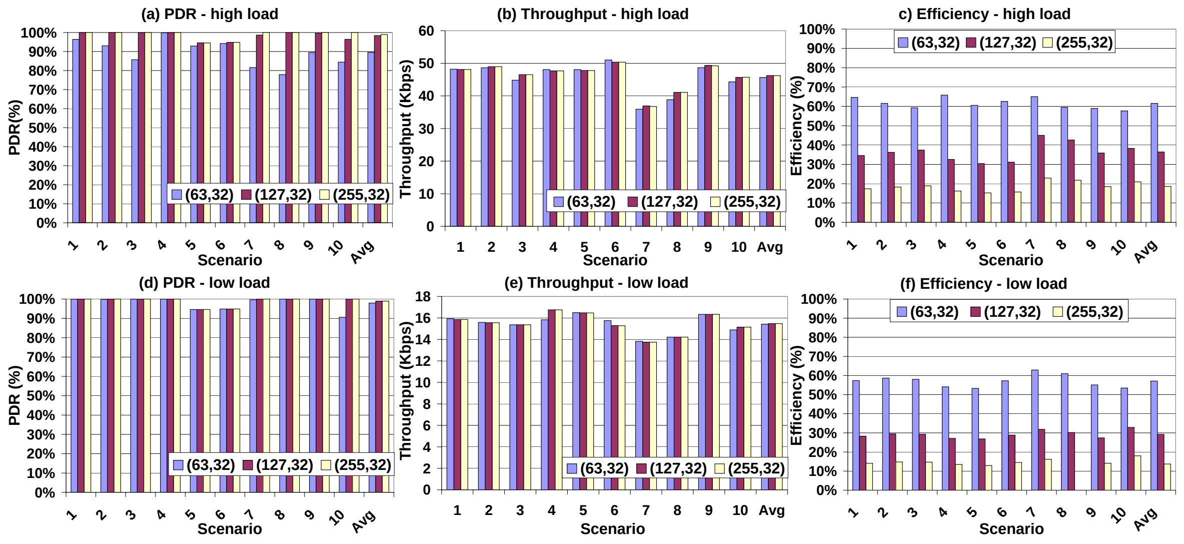

Figure 2. Comparison of RS codes of different redundancy in terms of PDR, throughput, and efficiency for 10 different topologies.

wasted in order to achieve finally 100% PDR. This is due to high uncorrelated packet losses occurring in wireless environments.

In general, a (127, 32) RS code achieves almost the same PDR as a (255, 32) RS code, but with doubled efficiency. Hence, in the overall comparison in Section 3.2, we will use this code. Since 100% PDR can be achieved in practical scenarios with a (127, 32) RS code, which sends redundant data equal to only 3 times the file size, as opposed to 7 times as proposed in [4], the use of a hybrid FEC-ARQ scheme will be preferred only if it offers higher throughput or efficiency compared to the pure FEC scheme. This is examined in the following sections.

NP In NP, redundant packets are sent only after explicit requests (NACK packets from the receivers). Hence there is no reason to keep the value of n low in the RS code. If the protocol performs well, only a part of the redundant packets is going to be sent. Hence for the NP evaluation, we used a (255, 32) RS code in all cases. Figure 3 shows the results for PDR, throughput, and efficiency in case of high and low load, for the original NP protocol (NP) and an optimized version (NP_opt) which we will describe in the following.

As Figure 3(a) shows, the average PDR under high load is only 78%, much lower than the PDR achieved by pure FEC, and only slightly better than the PDR of ODMRP (about 74%). Hence, NP in its original version fails to provide reliable data delivery in wireless networks. The reason can be explained by the way the protocol works. As we explained in Section 2.3, the sender sends a POLL message after each round, and the receivers respond to POLL messages with NACK messages containing the number of missing packets. Upon receiving a NACK, the sender sends the appropriate amount of redundant packets. In other words, there is a chain of three types of messages, POLL, NACK, and parity packets, each of which is sent as a response to the previous one. If one of them is lost, the chain breaks. If a POLL is lost before it reaches any receiver or in the more rare case that all the NACKs generated by different receivers as a response to a specific POLL are lost, the receivers will not recover any lost packets.

Note that even if a NACK is received, it is not always the NACK with the maximum number of lost packets, since NACK suppression may not work perfectly in wireless networks, as we explained in Section 3.2. Under low load, the problem is not so intense, because there is much less contention for the channel, and the probability of collisions is lower. As Figure 3(d) shows, the PDR under low load is quite high, about 92%, but again it is worse than pure FEC.

The above discussion shows that in wireless networks this handshake mechanism (POLL - NACK) is not efficient. A second mechanism is necessary to initiate NACK messages even if POLLs are lost. On the other hand, the rationale behind this design was to keep the control overhead low. Simply sending NACK packets in response to packet losses and using no POLL messages could lead to uncontrolled bursts of NACK packets (NACK implosion), similar to what we observed in ReMHoc in Section 2.1. Our solution to the problem is as follows. We keep the POLL-NACK exchange the same as in the original NP protocol, but every time a NACK is sent, we reset the timer for that NACK. If a POLL message comes before the timer expires, we cancel it, since a new NACK will be sent in response to the POLL message. However, if the timer expires before any POLL message comes, the same NACK is re-sent. Note

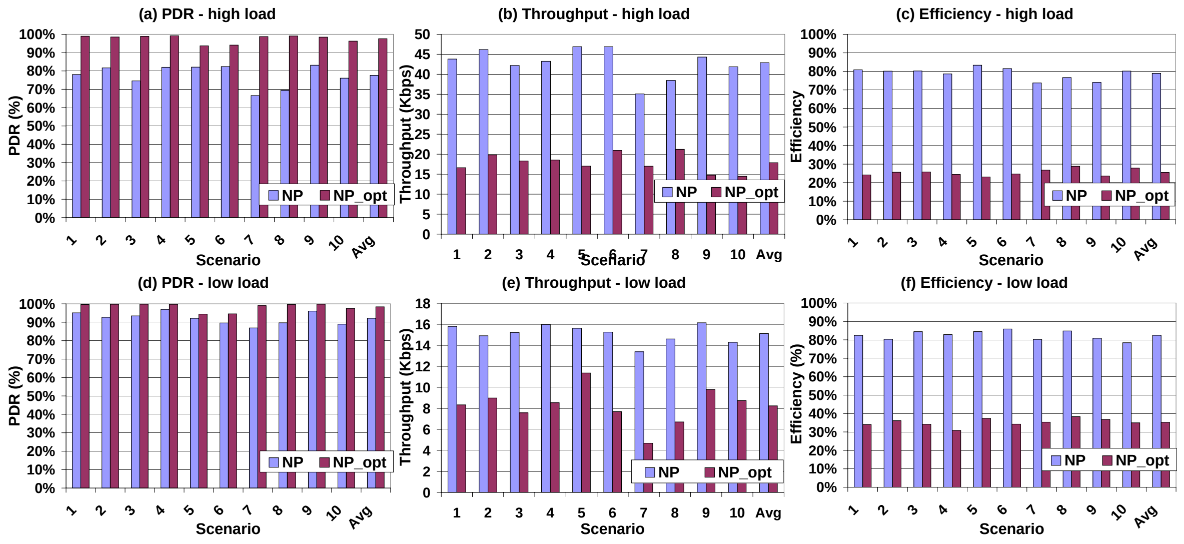

Figure 3. PDR, throughput and efficiency of the original NP protocol and an optimized version for 10 different topologies. NP and NP_opt use a (255, 32) RS code.

that we need this mechanism only as a backup mechanism, which should not interrupt the POLL-NACK exchange, and should not create more, unnecessary NACKs. Hence we set this timer to a very large value (200 sec), to make sure that it will only be activated if the basic mechanism of the protocol is not working anymore. We call this new version of NP NP_opt (optimized NP).

Figures 3(a), 3(d) show that NP_opt increases significantly the reliability compared to NP. The PDR with NP_opt is on average 97.5% under high load and 98.4% under low load, which are very close to 100%. If we look at the total number of NACK and POLL messages sent on average with NP and NP_opt, we will see that NP_opt sends on average about 7 times more NACK messages and 4 times more POLL messages compared to NP. But these extra control messages do not cause extra overhead, because they are sent in a much longer time period, compared to NP. To verify this, we look at the time the last useful data/parity packet was received. This time is on average 610 sec for NP and 3480 sec for NP_opt under high load, and 2049 sec and 8190 sec, respectively under low load.

The above values show again that increased reliability comes at the cost of low throughput. Figures 3(b), 3(e) verify this, showing that average throughput with NP and NP_opt, respectively, is 42.8Kbps and 17.8Kbps under high load, and 15.1Kbps and 8.2Kbps under low load. Similarly the efficiency drops from 79% for NP to 25% for NP_opt under high load and from 82% to 35% under low load.

Overall, the value of throughput for NP is close to that achieved by a pure (127, 32) FEC code, and the efficiency much better, but the PDR much lower. On the other hand, NP_opt achieves a PDR almost equal to that of a pure (127, 32) FEC code, but still less than 100%, with lower throughput and efficiency. A general conclusion is that in wireless environments, trying to recover from bursty losses by immediately responding to those losses is not a good solution, since it can very easily lead to congestion. Instead, a protocol that schedules future response to current requests might be a better option. This is the main design idea in RMDP, the second hybrid protocol, which we study next.

RMDP In RMDP we also used a (255, 32) RS code, as in NP, for the reasons we explained previously. Figure 4 shows the results for PDR, throughput, and efficiency in case of high and low load, for three different values of D, 1, 3, and 5, denoted as RMDP_1, RMDP_3, RMDP_5, respectively.

In Figures 4(a), 4(d), we observe that the PDR for RMDP with D = 3, or D = 5, reaches 100% with both high and low load, for all the scenarios except the two pathological scenarios we mentioned when we discussed the results of FEC. With D=1, the sender initially sends only the original data packets for each TG (32 packets), and extra packets are only sent after requests. This increases the protocol overhead, since many more requests (R packets) are initiated, and the probability for collisions increases, reducing the PDR. With $D > 1$, some redundancy is sent in the beginning, reducing the number of subsequent requests for more packets. We measured the number of requests sent in each case, and we found that on average each receiver sends 274 requests when D = 1, 45 requests when D = 3, and only 22 requests when D = 5, in case of high load, and the numbers are similar under low load.

Hence, RMDP, the only protocol designed to work in both wired and wireless environments (although previously evaluated only in the former), does achieve the goal of pro-

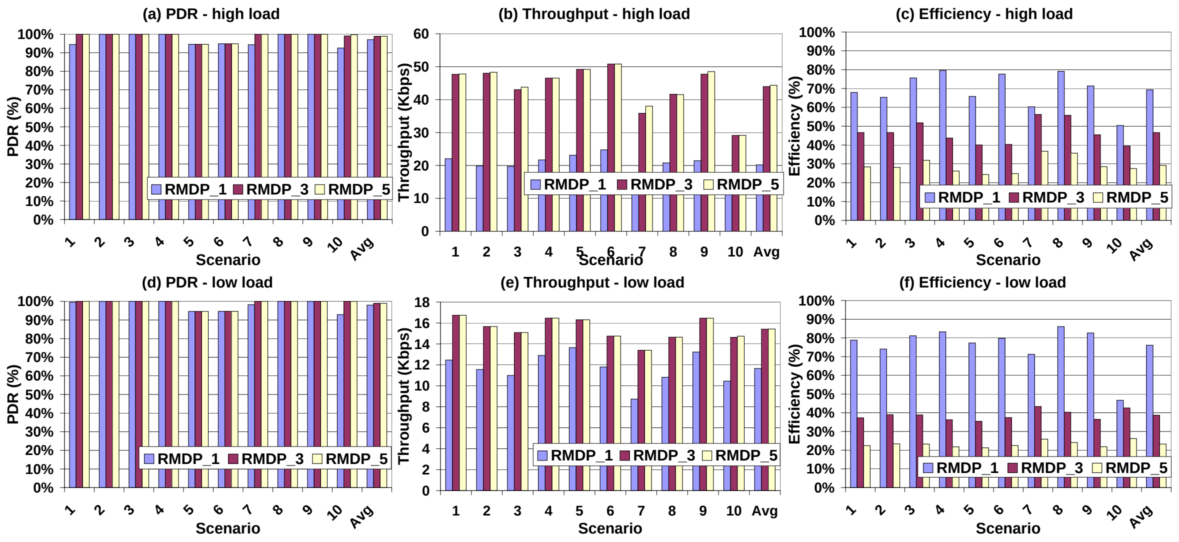

Figure 4. PDR, throughput, and efficiency of RMDP for three different values of D for 10 different topologies. RMDP uses a (255, 32) RS code.

viding reliable multicast data delivery in wireless networks. As we explained previously, RMDP does not take any action to recover from losses the moment they are observed. On the receiver side, a timer for a request for the TG with the largest number of losses is set every time a data packet is received, but in most cases this timer is reset upon reception of the next data packet. Only if a receiver experiences a large interval without receiving any packets, it initiates an R message. Also, reseting the timer after sending a request ensures that the protocol will never stop working (the mechanism is similar to that we applied in NP_opt). On the sender's side, no repair packets are sent upon the reception of a request. The only action taken is scheduling more parities for future transmission by increasing the appropriate c_s counter. This significantly reduces overhead and eliminates the problem of bursty losses, but it can potentially reduce throughput, since it may take longer for the receivers to complete the necessary number of packets to decode.

Figures 4(b), 4(e) show that throughput increases with D, since more packets are sent in advance. RMDP with D = 1, 3, and 5 achieves on average throughput equal to 20.2 Kbps, 43.9 Kbps, and 44.4 Kbps, respectively, under high load, and 11.7 Kbps, 15.3 Kbps, and 15.4 Kbps, under low load. For D = 3 or 5 these values are very close to those achieved by pure FEC in Figures 2(b), 2(e). But the great advantage of RMDP over pure FEC is efficiency. Figures 4(c), 4(f) show that the efficiency of RMDP with D = 1, 3, and 5, is 69%, 47%, and 29%, respectively, under high load, and 76%, 39%, and 23% under low load. For example, with D = 3, RMDP with a (255, 32) FEC code improves efficiency by 30% and 34% compared to a pure (127, 32) FEC code, under high and low load, while achieving the same PDR and the throughput reduced only by less than 5%.

In general, there is a tradeoff in the amount of redundancy sent unconditionally (without any request). By sending more packets unconditionally, we can increase throughput but the efficiency is reduced. D = 3 seems a good compromise between these two conflicting factors, and hence we will use this value in our overall comparison.

Overall Comparison In this section, we compare all five protocols, in order to give an overall picture of their relative performance. For this comparison, the best version of each protocol is used. FEC uses a (127, 32) RS code, and RMDP has D = 3. Figure 5 shows the PDR, throughput, and efficiency for ODMRP_static, ReMHoc, a (127, 32) FEC code, NP_opt, and RMDP with D = 3, for 10 different topologies.

Figures 5(a), 5(d) show that FEC and hybrid ARQ-FEC based protocols offer significant improvements in terms of reliability, increasing the PDR close or up to 100%. In contrast, pure ARQ-based protocols such as ReMHoc have a very poor performance in wireless environments, reducing the PDR to unacceptable levels, much lower than the best-effort ODMRP. As we mentioned before, ReMHoc is very vulnerable to bursty packet losses, because both the sender and the receivers try to respond immediately to each single packet loss detection. This causes the well-known request/repair implosion problem, and it finally leads to congestion. The request/repair suppression mechanism cannot mitigate the problem, because it does not work properly in wireless environments.

NP solves partly this problem, since on the receiver's side response is not immediate, but only after the reception of a POLL message. In other words, the receivers in NP

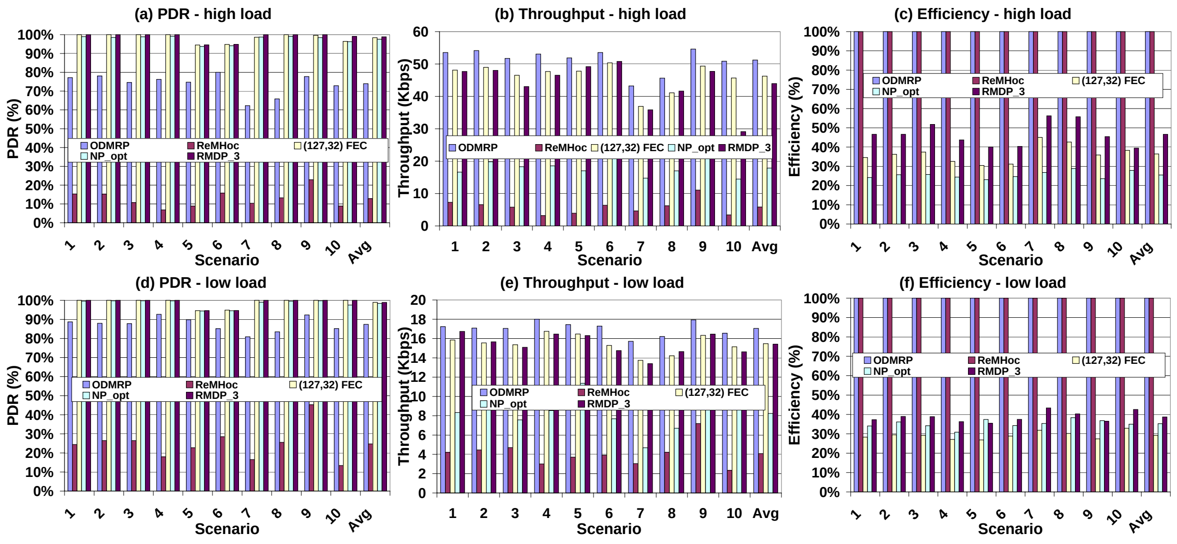

Figure 5. PDR, throughput, and efficiency for ODMRP, ReMHoc, a (127, 32) FEC code, NP_opt, and RMDP with D = 3, for 10 different topologies. NP_opt and RMDP use a (255, 32) RS code.

perform some kind of NACK aggregation. However, this aggregation could cause even larger problem on the sender's side and increase the burstiness of repair packets. Finally, the use of a timer for NACKs with a carefully selected value mitigates this new problem, and hence NP_opt performs almost as well as FEC and RMDP in terms of PDR.

Pure FEC and RMDP are the only two protocols that can achieve 100% PDR, because they take no immediate response in case of packet losses. Our experiments with FEC show that we can always achieve 100% PDR by increasing the amount of redundancy sent. However, FEC can be very inefficient, wasting many packets, since the amount of redundancy to be sent is decided in advance. On the other hand, RMDP sends only a fraction of redundant packets in advance, and the rest is sent only if necessary. For this reason, RMDP is the most efficient among the three protocols. As Figures 5(c), 5(f) show, the efficiencies of RMDP, NP_opt, and FEC are on average 47%, 26%, and 34%, respectively, under high load, and 37%, 35%, and 29%, under low load. Increased efficiency, especially under high load, is another advantage of RMDP over NP. For ODMRP and ReMHoc efficiency is 100%, since there are no redundant packets, but this efficiency is meaningless, since it is counterbalanced by the low PDR.

Finally, we observe in Figures 5(b), 5(e) that increased reliability comes at the cost of reduced throughput, since for all three protocols that increase reliability, i.e., FEC, NP, and RMDP, the average throughput is lower than for ODMRP. However, for FEC and RMDP, this reduction is very small – throughput is 46 Kbps for FEC and 44 Kbps for RMDP vs. 52 Kbps for ODMRP under high load, and 15 Kbps for both FEC and RMDP vs. 17 Kbps under low load, and it can be tolerated, since the gains in reliability are much higher. On the other hand, throughput for NP is very low – 18 Kbps and 8 Kbps, under high and low load, which shows once more that the design of NP is not appropriate for wireless environments.

4. Related Work

Recently, many protocols have been proposed for reliable multicast in mobile ad hoc networks. A survey on reliable multicast protocols for ad hoc networks [16] classifies them into deterministic and probabilistic ones, depending on whether the delivery is fully reliable or not. Deterministic protocols [17, 18, 19, 20, 15, 21] provide deterministic guarantees for packet delivery ratio, but they incur high overhead when the mobility or the group size increases. In these cases, they resort to flooding, resulting in severe performance degradation. In contrast, probabilistic protocols [22, 23] do not offer hard delivery guarantees, but they incur much less overhead compared to the former. All the above protocols fall in the class of ARQ. As mentioned before, no work other than RMDP [4] has considered the use of FEC for reliable multicast in wireless networks. To our best knowledge, we are the first to study the performance of FEC for multicast in wireless mesh networks.

In our comparison we used Reed-Solomon codes in the FEC-based schemes. Reed-Solomon codes are very easy to simulate, because they are deterministic, meaning that a receiver knows in advance that it can obtain the original data if it receives *any* k out of n packets. A drawback of Reed-Solomon codes is that k and n have to be kept small (the maximum values are 64 and 255, respectively). Tornado codes [24] allow the use of very large values for k and n, but they are non-deterministic, and the receiver has to perform a partial decoding of the data stream in order to decide when it should stop receiving. Moreover, the number of packets required for decoding is $(1+\varepsilon)\times k$, but ε is different for each receiver and unknown beforehand, which complicates both the design of hybrid protocols. Finally, rateless codes is a new class of erasure codes in which an arbitrary number of encoded packets can be produced on demand. [25] uses LT codes [26], one type of rateless codes, to implement Digital Fountain, an ideal protocol that allows any number of heterogeneous receivers to acquire content with the optimal efficiency at the times of their choosing. Rateless codes are also probabilistic codes, and hence they cannot be easily incorporated into the current hybrid protocols, since receivers can not report how many parities they need.

Finally, adaptive FEC techniques that adapt the number of parities sent based on an estimation of the loss rate have been proposed for multicast in wireless LANs [27, 28]. In our future work we plan to study if these techniques can also be applied to multihop mesh networks.

5. Conclusions

In this paper we examined the applicability of FEC-based reliable multicast protocols initially proposed for the wired Internet in wireless mesh networks. We compared four different reliable multicast schemes, one ARQ-based (ReMHoc), one FEC-based (use of RS codes), and two hybrid protocols (NP and RMDP). Our simulation study shows that ARQ-based protocols perform very poorly in wireless environments, because they cannot cope with bursty packet losses. Moreover, we found that trying to respond immediately to packet losses or to requests for retransmissions is not a good design choice for wireless protocols, because it can very quickly lead to congestion. The use of FEC without any feedback from the receivers, and hence without any need for retransmissions, can offer perfect reliability in most practical scenarios, but at the cost of low efficiency. RMDP can increase the efficiency while keeping the PDR at the same level as pure FEC, by scheduling more packet transmissions in the future based on feedback from the receivers, instead of immediately responding to this feedback. In our future work, we plan to validate our simulation results on a real mesh network testbed [29].

Acknowledgment

This work was supported in part by NSF grant CNS-0626703.

References

[1] L. Huang and H. Hassanein, “A performance comparison of reliable multicast protocols over ad hoc networks,” in *22nd Biennial Conference on Communications*, 2004.

[2] L. Rizzo, “Effective erasure codes for reliable computer communication protocols,” *ACM Computer Communications Review*, vol. 27, no. 2, 1997.

[3] J. Nonnenmacher, E. Biersack, and D. Towsley, “Parity-based loss recovery for reliable multicast transmission,” in *ACM SIGCOMM*, 1997.

[4] L. Rizzo and L. Visicano, “RMDP: an FEC-based reliable multicast protocol for wireless environments,” *Mobile Computing and Communications Review*, vol. 2, no. 2, 1998.

[5] J. Yoon, A. Bestavros, and I. Matta, “Adaptive reliable multicast,” in *Proc. of ICC*, 2000.

[6] E. Schooler and J. Gemmel, “Using multicast FEC to solve the midnight madness problem,” Technical Report, MSR-TR-97-25, Tech. Rep., 1997.

[7] R. Kermode, “Scoped hybrid automatic repeat request with forward error correction (SHARQFEC),” in *Proc. of ACM SIGCOMM*, 1998.

[8] J. Gemmel, “Scalable reliable multicast using erasure-correcting resends,” Technical Report, MSR-TR-97-20, Tech. Rep., 1997.

[9] M. Cagalj, J.-P. Hubaux, and C. Enz, “Minimum-energy broadcast in all-wireless networks: NP-Completeness and distribution,” in *Proc. of ACM MobiCom*, September 2002.

[10] A. Sobeih, H. Baraka, and A. Fahmy, “ReMHoc: A reliable multicast protocol for wireless mobile multihop ad hoc networks,” in *IEEE IPCCC*, 2004.

[11] S. Floyd, V. Jacobson, C.-G. Liu, S. McCanne, and L. Zhang, “A reliable framework for light-weight sessions and application level framing,” *IEEE/ACM Transactions on Networking*, 1997.

[12] S.-J. Lee, M. Gerla, and C.-C. Chiang, “On-Demand Multicast Routing Protocol,” in *Proc. of IEEE WCNC*, September 1999.

[13] A. J. McAuley, “Reliable broadband communication using a burst erasure correcting code,” in *Proc. of ACM SIGCOMM*, 1990.

[14] X. Zeng, R. Bagrodia, and M. Gerla, “Glomosim: A library for parallel simulation of large-scale wireless networks,” in *Proc. of PADS Workshop*, May 1998.

[15] K. Tang, K. Obraczka, S.-J. Lee, and M. Gerla, “Reliable adaptive lightweight multicast protocol,” in *Proc. of ICC*, 2004.

[16] E. Vollset and P. Ezhilchelvan, “A survey of reliable broadcast protocols for mobile ad-hoc networks,” University of Newcastle upon Tyne, Tech. Rep. CS-TR-792, 2003.

[17] E. Pagani and G. Rossi, “Reliable broadcast in mobile multihop packet networks,” in *Proc. of MobiCom*, 1997.

[18] S. Gupta and P.Srimani, “An adaptive protocol for reliable multicast in mobile multi-hop radio networks,” in *Proc. of IEEE Workshop on Mobile Computing Systems and Applications*, 1999.

[19] T. Gopalsamy, M. Singhal, and P. Sadayappan, “A reliable multicast algorithm for mobile ad hoc networks,” in *Proc. of ICDCS*, 2002.

[20] K. Tang, K. Obraczka, S.-J. Lee, and M. Gerla, “A reliable, congestion-controlled multicast transport protocol in multimedia multi-hop networks,” in *Proc. of WPMC*, 2004.

[21] V. Rajendran, Y. Yi, K. Obraczka, S.-J. Lee, K. Tang, and M. Gerla, “Combining source- and localized recovery to achieve reliable multicadt in multi-hop ad hoc networks,” in *Proc. of Networking*, 2004.

[22] R. Chandra, V. Ramasubramaniam, and K. Birman, “Anonymous gossip: Improving multicast reliability in mobile ad hoc networks,” in *Proc. of ICDCS*, 2001.

[23] J. Luo, P. Eugster, and J.-P. Hubaux, “Route driven gossip: Probabilistic reliable multicast in ad hoc networks,” in *Proc. of IEEE Infocom*, 2003.

[24] M. Luby, M. Mitzenmacher, A. Shokrollahi, and D. Spielman, “Efficient erasure correcting codes,” *IEEE Transactions on Information Theory*, vol. 47, 2001.

[25] J. Byers, M. Luby, and M. Mitzenmacher, “A digital fountain approach to asynchronous reliable multicast,” *Journal on selected areas in communications*, vol. 20, 2002.

[26] M. Luby, “LT codes,” in *The 43rd Annual IEEE Symposium on Foundations of Computer Science*, 2002.

[27] D. Xu, B. Li, and K. Nahrstedt, “QoS-directed error control of video multicast in wireless networks,” in *Proc. of ICC*, 1999.

[28] P. Chumchu, Z. G. Zhou, and A. Seneviratne, “A model-based scalable reliable multicast transport protocol for wireless/mobile networks,” *IEICE Transactions on Communications*, vol. 4E88-B, no. 4, 2005.

[29] Mesh@Purdue, “http://www.engineering.purdue.edu/MESH.”