# Collaborative Recovery for Reliable Multicast in Mobile Ad Hoc Networks
# UCLA TR REPORT #040005

Yunjung Yi, Jiejun Kong and Mario Gerla
Computer Science Department, UCLA
Los Angeles, CA 90095
e-mail: {yjyi, jkong, gerla}@cs.ucla.edu

February 2, 2004

**Abstract**

Reliable multicast is a critical network service for key applications in mobile ad hoc networks (MANETs). Multimedia applications and data transfer applications have different QoS demands: delay-bounded delivery and 100% data delivery guarantee. In the paper, we present two reliable multicast protocols to meet these demands: Collaborative Opportunistic Recovery Algorithm (CORA) and Collaborative Assured Recovery Algorithm (CARA). CORA seeks to improve delivery ratio with bounded delay and minimal communication overhead. CARA is an extension of CORA designed for applications that require 100% delivery guarantee.

CORA uses Packet-Based Distance Vector (PBDV) scheme to provide *local* packet recovery service. The function of PBDV is to minimize communication overhead and packet recovery latency while maximizing packet delivery ratio. In addition, NACK aggregation and multicast congestion control techniques are integrated into CORA to address "NACK implosion" and data forwarding congestion problems respectively.

CARA extends CORA with the digital fountain concept, achieving strong reliability even in the presence of mobility and heavy channel errors.

We demonstrate the effectiveness of CORA and CARA through comprehensive simulation studies.

## 0.1 Introduction

Multicast communication is an efficient means to support key applications of mobile ad hoc networks (MANET) such as teleconferencing and data dissemination. These applications require both high data reliability and timeliness guarantees even in the presence of mobility, random link error, and frequent outages. Characteristics of MANET such as limited resources, dynamic topology, vulnerability to network congestion, challenge a reliable multicast protocol. Thus, it is extremely arduous to develop a reliable protocol which achieves both deterministic reliability and bounded-delay guarantee. In general, only the second condition–bounded delay–is strictly demanded in most multicast applications. For instance, multimedia (e.g., audio or video) multicasting applications favor bounded latency over high reliability (say 100% packet delivery ratio). Clearly better delivery ratio is critical to improve quality of service (QoS). Nevertheless, the extra cost required for acquiring more data by loss recovery should be minimized, and it is imperative that lost packets be recovered within bounded latency. In contrast, data transmission applications such as Wb (distributed whiteboard tool) [9] and battlefield data dissemination (e.g., multicast transmission of situation awareness and commands from the command post to squad leaders) need deterministic reliability. In fact, in those applications, strict latency and overhead are less important than 100% delivery ratio. Therefore, a reliable multicast protocol should be designed with consideration of the applications' demands.

In the paper, we consider two categories of reliability: (1) best-effort, opportunistic reliability and (2) strong reliability, and accordingly present two MANET reliable multicast protocols: ***C****ollaborative* ***O****pportunistic* ***R****ecovery* ***A****lgorithm* (**CORA**) and ***C****ollaborative* ***A****ssured* ***R****ecovery* ***A****lgorithm* (**CARA**).

The main goal of CORA is to minimize recovery overhead and maximize reliability within bounded latency. To achieve this goal we pursue three directions: (1) recover needed data within minimal distance; (2) reduce number of packet loss; and (3) minimize control overheads caused by loss recovery so that impact on regular data delivery is minimized. Accordingly, CORA addresses these challenges by employing (1) a localized recovery with the aid of neighbor nodes; (2) a congestion control scheme to reduce the probability of loss; and (3) a combination of promiscuous listening, NACK-piggyback, and NACK-aggregation to reduce control overheads. A unique feature of CORA is Packet-Based Distance Vector (PBDV) routing, which permits to rapidly locate copies of the missing packet while providing several other assistant features in local recovery and NACK. Each node maintains an opportunistic PBDV routing table for each multicast group.

Next, CARA is proposed to guarantee strong reliability. CARA builds on CORA. It is a source-centric recovery procedure employing digital fountain approach [4]. The digital fountain approach allows flexibility at each receiver in

that original source data can be reconstructed from *any* disjoint subset of a threshold size of the encoding symbols[1]. Thus, digital fountain approach is robust against temporary network disconnections, high mobility, random errors, and intermittent attacks.

The main contributions of CORA and CARA are: (1) efficient, low latency, scalable local recovery scheme based on PBDV; (2) ECN based end-to-end congestion control that accounts for virtual neigbhorhood congestion and can discriminate between congestion and random loss; (3) the effective application of the digital fountain approach to highly mobile, lossy scenarios.

The rest of the paper is organized as follows. Section 0.2 describes CORA/CARA's conformity and comparison with reliable multicast protocols in both IP and MANET area. Detail protocol description follows in Section 0.3 and simulation study in Section 0.4. Finally Section 0.5 concludes the paper.

## 0.2 Comparison with related work

We may categorize existing reliable multicast protocols into three classes: (1) a source-oriented reliable multicast; (2) a receiver-oriented approach; and (3) a cooperative router approach.

First, in a source oriented reliable multicast, a source is responsible to guarantee the reliability using techniques such as digital fountain [4] and reliable broadcast [23]. In the digital fountain approach, using a smart FEC coding scheme, the source node takes the burden of data reliability control. Reliable broadcast addresses the problem of reliable atomic delivery (all-or-nothing) of messages. This protocol provides a reliable multicast by performing two phases: *scattering* and *gathering*. In scattering phase, a source propagates the data to all members and all ACKs are collected to the source in gathering phase. This approach suffers from potential ACK implosion.

The second class—receiver oriented—includes most of the reliable protocols such as SRM (Scalable Reliable Multicast) [9], Anonymous Gossip (AG) [6], Route-Driven Gossip (RDG) [17], Reliable Adaptive Lightweight Multicast (RALM) [36] and Multicast Dissemination Protocol (MDP) [18]. In this approach, each receiver cooperates with other receivers and with the source to recover lost packets. This approach can be further categorized, based on whom the "recovery" NACK is sent, into multicast-NACK approach: a NACK is sent to the entire group (e.g., SRM), unicast-NACK approach: a NACK is sent to the source node (e.g., RALM) and GOSSIP-based protocol: a NACK is sent to another member (e.g., AG, RDG).

Lastly, a cooperative router approach could be hop-to-hop reliable multicast protocols such as Pump Slowly Fetch Fast (PSFQ) [37] and Reliable Multi Seg-

[1]Symbol in coding theory is a block of data to represent coding alphabet, e.g., of the size several bits or several bytes. A data packet may be comprised of multiple symbols. We use the term "symbol" whenever coding theory is involved.

ment Transport (RMST) [33] and NACK aggregation technique [32] where each intermediate router smartly aggregates duplicate NACKs. Hop-to-hop reliable multicast guarantees the reliable multicast data transmission at each forwarding node by using forward-and-send mechanism. A forward node sends a data packet if it is delivered in sequence. Otherwise, it tries to recover the lost packets before sending a new packet to next hop node. This approach can reduce the recovery overhead and latency, however, it does not work well in a dynamic network with node mobility.

The brief descriptions of protocols follow.

Digital fountain approach [4] is a source-centric reliable multicast using a smart FEC (Forward Error Correction) coding scheme. The ideal semantics of digital fountain is threshold based and set oriented, that is, the original source symbols can be reconstructed intact from *any* subset of the encoded symbols, as long as the size of the subset is bigger than or equal to the size of the original data. This way, any recipient can dynamically join a multicast session nearly at any time. Digital fountain approach can be very effective in MANET environment. Fragile links and frequent outages may temporarily disconnect a multicast receiver from the source. Once a receiver recovers the path (i.e., re-join), it can receive the rest of the encoded data. However, digital fountain users pay the price of extra encoding/decoding latency and of huge buffer requirements at the receivers. Thus, the approach is not appropriate for multimedia applications.

Reliable multicast protocols developed in IP multicast include SRM [9][8] and RMTP [24][13]. Those protocols address the issue how to handle potential "NACK/ACK" implosion problem. In SRM, each member, which detects a packet loss, multicasts a NACK to the entire multicast group. Before sending a NACK, to prevent NACK implosion, each receiver waits for a random NACK timer to suppress the duplicate NACKs. RMTP handles ACK-implosion problem by providing a multi-level ACK tree. In RMTP, members are grouped into a several local groups with a Designated Receiver (DR) in each region. An ACK from a receiver will be forwarded along the multi-level ACK tree. Each DR aggregates ACKs from its members in the region and forwards them to its higher-level DR.

MDP (Multicast Dissemination Protocol) [18] supports reliable and congestion controlled multicast transmission. It mainly targets a reliable bulk data transmission in a network with high loss rate and heterogeneous conditions. MDP also uses FEC coding scheme (e.g., Reed-solomon code) to endure high loss rate effectively. Besides, MDP proposes a rate-based congestion control which extends TCP-like scheme to multicasting.

These IP reliable multicast protocols assume the construction of a fixed multicast tree by the underlying IP multicast protocol. In MANET, routes change frequently leading to prohibitively high tree maintenance costs. Therefore the methods used by these Internet protocols are not practical in MANET.

RALM (Reliable Adaptive Lightweight Multicast) protocol [36] is another receiver-oriented recovery mechanism developed for MANET reliable multicast. RALM favors reliability over throughput. RALM works in two phases: (1) free-wheeling where a sender sends out data packets with the input sending rate from the application and (2) congestion control phase where the sender retransmits the lost packets using a NACK/ACK scheme. When a node detects the packet loss(es), it transmits a NACK. Once the source receives the NACKs, it restores the lost packets to each requester (a source picks up one requester at a time in a round-robin fashion for the recovery) and verifies recovery from the ACK issued by each requester. RALM suffers from potential NACK implosion and extremely low throughput with large group and network size. Thus, it has the limitation of scalability.

Lastly, gossip-based multicast protocols [3] feature smart members and scalability. Recently, Anonymous Gossip [6] and Route Driven Gossip [17] were proposed to improve best-effort reliability for MANET multicast protocols. Those ideas extend and apply the gossip approach [3] to be fitted in wireless ad hoc networks. The basic idea of the gossip approach is to transmit a recovery request for the lost packets to an arbitrary member, instead of the source. Because of the dynamic MANET topology, it is hard for each member to acquire/maintain total routing information to all members in the multicast group. Thus, an efficient algorithm is necessary to find a proper target member to send a query with low cost. In anonymous gossip[6], a receiver starts recovery of the lost packets by sending gossip requests to other randomly chosen multicast members. To learn routes to other members, AG slightly modifies the underlying multicast protocol. This modification requires extra overhead. In contrast, route-driven gossip [17] exploits the underlying unicast routing table to select members (gossipers) for the recovery request. A receiver sends a gossip request to (possibly multiple) member(s) chosen in its *Active View*. Each node's Active View includes the members where the routes (by unicast routing) to those members are known to this node. Thus, it reduces the overhead and improves the efficiency over AG. AG and RDG achieve scalability by distributing recovery overhead to the entire multicast group instead of centralizing at the sender. However, those protocols have two weaknesses: (1) no guarantee of strong reliability and (2) unbounded recovery latency (it may take quite a few trials to find the node which has a copy of the lost packet).

The CORA/CARA architecture differs from above reliable multicast schemes because of the smart source/forwarder/recipient design choice. We observe that Internet reliable multicast schemes do not rely on the smart forwarder approach mainly due to "the end-to-end argument" [29]. However, Internet schemes can rely on fixed multicast tree and other means available in wired networks. In contrast, MANETs are highly dynamic. The self-organizing nature of MANETs requires that all participants pay a reasonable price to help each other to provide network services in the presence of mobility and channel errors.

According to measurements on various portable computing devices [34][7][30],

the wireless interface incurs non-trivial communication overheads in terms of energy consumption. In particular for the awake mode of 802.11 interfaces, power consumed in the *transmit state* for transmissions over me-dium/large distances is significantly higher than the one consumed in the *receive state* or the *idle state*. On the other hand, nowadays mobile devices, even low-end pocket devices, normally have megabytes of memory installed. As shown in several system-level measurements [14], memory access only consumes a small portion of power—many off-the-shelf low-power memory systems consume less than $1mW$ power while wireless interfaces consume hundreds of $mW$. Based on this observation, CORA/CARA implement not only smart receivers and smart sources as previous reliable multicast protocols did, but also smart forwarders and thus trade communications for memory.

The smart forwarders in CORA/CARA play two important roles: (1) Depending on memory availability, they cache packets recently forwarded. The caching incurs no extra communication overhead. Now that the smart source, forwarders, and recipients constitute a connected subnet inside the MANET, it is feasible to run an efficient Distance Vector scheme on the subnet, to track the cached packets, so that recipients can efficiently recover lost packets before the deadline. (2) To improve upon end-to-end congestion control, we implements explicit congestion detection and notification at the smart forwarders.

## 0.3 Protocol Design

### 0.3.1 Design assumptions

The basic idea of CORA's loss recovery is to recover a packet from the nearest point that has cached the needed packet. While the lost packet can be recovered from the source, sending a NACK to a source will cause potential NACK implosion. Thus, *local recovery* and *NACK aggregation* are essential to protocol scalability. Similar to a NACK aggregation techniques used in IP multicast router assistance design [32][12], each intermediate forwarder in CORA, i.e., each router on the path back to source aggregates NACK messages. We assume that either the underlying multicast protocol provides shortest paths to source as part of a source tree (e.g., ODMRP and MAODV) or it is possible to slightly modify the underlying protocol (e.g., MCEDAR [31] and CAMP [19] to acquire such a tree.

CORA also assumes that a multicast data packet can be distinguished by a unique identifier, ⟨group address, source address, sequence number⟩). The sequence number field is increased by 1 at the sender for each new packet. All nodes cache forwarded (or received for a leaf member) multicast packets during the last $T_{max}$ seconds (where $T_{max}$ is approximately the round trip time along the network diameter). This implies that both group members and non-member forwarders are required to cache received packets. Then a packet can be recovered from a node's local cache if this node participated in the underlying unreliable multicast within $T_{max}$ time.

Furthermore, every node in the network maintains a Packet-Based Distance Vector (PBDV) routing table to register lost data packets in table entries. Unlike data packets, PBDV table entries are compact and consume small amount of space ($< 20$ bytes per entry). Therefore, a CORA node tries to maintain a PBDV table for a multicast group even if the node is neither a member nor a forwarder of the group. Each entry contains simple DV route information to reach the nearest node that has cached the packet. Then each node makes available its own PBDV table to help other nodes to recover their loss packets within minimal distance.

PBDV does not use proactive or explicit messages thus avoiding extra communication overheads. Rather, PBDV routing information is obtained reactively and opportunistically. CORA nodes extensively exploit NACK piggyback and wireless promiscuous listening to acquire PBDV routing information: (1) By (over-)hearing a data packet, a node knows the packet sender has the packet. (2) Nodes can piggyback their own PBDV metrics in control messages like NACK. Other nodes overhearing these control messages can compute appropriate PBDV metrics. The piggyback communication overhead is tractable and small because each PBDV metric consumes tiny space (8-bit hop count in our simulation).

### 0.3.2 CORA recovery overview

CORA works in two phases: multicast forwarding and loss recovery. In first phase, the source sends data packets using the underlying unreliable multicast protocol, with the sending rate adjusted by congestion control as later discussed in this section. Upon detecting packet loss, a multicast group member initiates loss recovery process which runs in background.

On each member, the recovery process includes four sequential steps:

1. *PBDV recovery*: If the lost packet sequence number has a valid entry in PBDV, the member initiates explicit request to the neighbor pointed in the PBDV entry. The retrieval may require a few hops as directed by PBDV.

2. *Local recovery*: For lost packets with invalid PBDV entries (i.e., PBDV metric for that packet is $\infty$), the member tries to recover the missing packets from one-hop neighbors. This is implemented by an efficient NACK/REPLY handshake: the member issues a short NACK with broadcast network address, and any neighbor sends back a short reply (after a random backoff to prevent collisions) if it has cached some of the lost packets or knows where they are. PBDV update metrics are piggybacked in both NACK-broadcast and replies. Thus nodes within *two hops* away of the requester can update their PBDV table accordingly. For cached packets on one-hop neighbors, CORA chooses not to send back these (long) data packets immediately to the requester. Instead, short replies are sent back to notify the requester about the cached packets. This design choice will be justified later in this section.

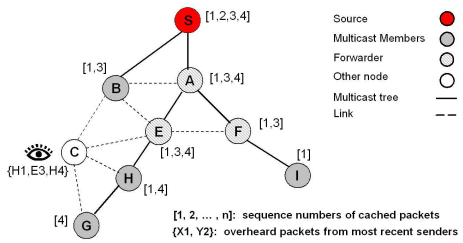


Figure 1: A sample scenario

3. *PBDV retry* : PBDV recovery is performed again if there is any reply during local recovery step. There is no "second chance" (of local recovery) for packets not recoverable from this retry.

4. *Source recovery*: For the remaining missing packets, the member sends a NACK unicast to its upstream node toward the source until all the lost packets are recovered or the application delay bound expires.

**Example 1** *Fig. 1 illustrates an example of CORA recovery process for a single multicast group. In the figure $S$ is the source node, $\{B, G, H, I\}$ are members of the group, $\{A, E, F\}$ are forwarding nodes or forwarders in the underlying multicast protocol, and $C$ is neither a member nor a forwarder. Two nodes within each other's transmission range are connected by a solid link if the link is in underlying multicast tree, or by a dotted link otherwise.*

*The source $S$ sends packets with sequence numbers from 1 to 4. The underlying multicast protocol delivers them to members; some packets are lost. The bracket beside a forwarder or a member represents the set of cached packets. The curly bracket next to node $C$ represents its most recent PBDV table, i.e., packets 1 and 4 are 1 hop away in the direction of $H$, packet 3 is 1 hop away in the direction of $E$. We now briefly describe CORA recovery process on members $H$ and $G$. Fig. 1 depicts the moment right before $H$ starts recovery.*

1. *$H$ detects that packet 2 and 3 are lost. This time $H$'s PBDV recovery returns no result. Later we will see other nodes, such as $G$, can take advantage of PBDV recovery.*

2. *$H$ initiates its local recovery process. $H$ locally broadcasts a NACK to its neighbor nodes. In the NACK $H$ piggybacks its PBDV metrics, that is, 0 for packet 1 and 4, $\infty$ for packet 2 and 3. All local nodes update their PBDV metrics for packet 2 and 3 upon hearing this NACK—there is no PBDV change on node $C$ and $E$, but $G$ takes note that packet 1 is one hop away in the direction of $H$.*

3. *Since $E$ can recover packet 3 from its cache and $C$ knows packet 3 is one hop away in the direction of $E$, both $E$ and $C$ send back a short REPLY to*

*$H$. Choosing the best metric (1 for $E$ < 2 for $C$), $H$ now knows packet 3 is one hop away from $E$. $H$ then unicasts an explicit REQUEST to recover packet 3 from $E$ (it doesn't matter whether $E$ is $H$'s upstream node or not).*

*Obviously one NACK-broadcast may incur multiple replies. As a result, RTS/CTS based CSMA/CA cannot be used. Therefore, in CSMA both NACK and its replies must be short messages so that there is no significant performance degradation due to hidden terminals.*

4. *All nodes within two-hop range of $H$ also update their PBDV metrics upon hearing the replies. There is no PBDV change on node $A$, but $B$ and $F$ know packet 4 is one hop away from $E$, $G$ knows packet 3 is two hops away in the direction of $C$.*

5. *$H$ is still missing packet 2; it thus enters the source recovery step. This results in a NACK-unicast to its upstream node $E$. $E$ cannot recover 2 and then sends a NACK-unicast to its upstream node $A$. The same story repeats and finally $S$ receives the NACK-unicast. The source then retransmits packet 2 for $H$.*

6. *It is important to note that multiple members may start source recovery at same time to request the same packet. This situation is likely caused by early packet loss near the source. Therefore, source $S$ should not unicast the lost packet back to each requesting member. CORA solves this problem by exploiting wireless broadcast and a "breadcrumb" (BC) navigation bit—each NACK-unicast forwarder sets the BC bit to 1, then resets it to 0 when the coming back data packet is forwarded upon a set BC bit. This way, minimal transmission is used while all requesting members can receive the data packet from the source.*

   *In this example, both $E$ and $A$ set their BC bits (for packet 2) during NACK-unicast forwarding. When packet 2 comes back from source $S$, both $A$ and $E$ will cache it and locally re-broadcast it, then reset their BC bits to 0.*

7. *$H$ recovers packet 2 and its CORA recovery process ends if no further packet loss is detected.*

8. *Now $G$ starts loss recovery. Its PBDV table knows that packet 1 is one hop away from $H$ and packet 3 is two hops away in the direction of $C$. Using PBDV recovery $G$ sends out explicit REQUESTs to $H$ and $E$, respectively. The same "breadcrumb" navigation technique is also used for each multi-hop request because there may be multiple distributed members requesting the same lost packet in the neighborhood.*

9. *$G$ enters local recovery process and issues a NACK-broadcast to recover packet 2. By $H$ and $C$'s REPLY ($C$ knows packet 2 is one-hop away from $H$ when $E$ rebroadcasts it in $H$'s source recovery phase), all nodes within two-hop range of $G$ update their PBDV metrics.*

*10. Finally G sends an explicit REQUEST to H and recovers packet 2.*

### 0.3.3 CORA packet type

As seen in the above example, CORA implements five different packet types:

1. DATA: DATA packets deliver application data. In CORA, each data packet is identified by
$$\langle G, S, seqNo \rangle,$$
where $G$ is multicast group address, $S$ is data source address, and $seqNo$ is data packet sequence number. We use $\langle S, seqNo \rangle$ as packet ID in each multicast group.

2. NACK: A member sends a NACK control message to complain lost data packets and meanwhile to advertise its PBDV metrics. The member can send a NACK-broadcast using a broadcast address, or a NACK-unicast using a node's address. NACK-broadcast is used in local recovery and NACK-unicast is used in source recovery. In either case the NACK transmission is heard by all neighbors due to the wireless broadcast medium.

3. REPLY: Upon receiving a NACK-broadcast, a node sends back a REPLY control message if (1) the node has cached some of the lost data packets, or (2) the node can locate some of the lost packets in its local PBDV routing table.

4. REQUEST: This packet is used to request data packet "retransmission" to a node which has the packet. When a member can locate some lost packets using its PBDV entries, it unicasts REQUEST packets to the corresponding next-stops. If several lost packets have the same next-stop (next hop) in the PBDV routing table, the member aggregates multiple requests into a single REQUEST packet for each next-stop.

5. REJECT: When a CORA node cannot forward REQUEST packet due to invalid PBDV entry, it optionally sends back a REJECT message to flush the related PBDV routing tables.

More details of each packet will be explained and discussed in following protocol details.

Note that a CORA node does not aggregate multiple NACKs for different sources or groups, i.e., the recovery process is separately performed for each source and group. Thus the extension of recovery process to multiple group and sources is straightforward. For sake of simplicity, we explain the recovery process of packets from a single source $S$ in a multicast group $G$. $Pkt(k)$ refers a source packet with sequence number $k$ from now on.

### 0.3.4 PBDV table maintenance

Each node maintains a PBDV routing table for each multicast group. Each PBDV table has the format:

| $packetID = (S, seqNo)$ | | $nextStop$ | $D$ | $Ts$ | $BC$ |
|---|---|---|---|---|---|
| 32-bit | 32-bit | 32-bit | 8-bit | 32-bit | 1-bit |

where $packetID$ is the key column identifying each lost packet; $nextStop$ is the best-known next stop's address to reach the destination which caches the data packet identified by $packetID$; $D$ is distance metric in distance vector schemes (hop count in our simulation); $Ts$ is timestamp, so that an entry is recycled after timeout $T_{max}$; and $BC$ ("bread crumb" bit) indicates that the current node is on the recovery forwarding path of the missing packet, thus upon receiving the needed packet the node should rebroadcast it to its neighbors. Using $BC$ flag, each node forwards only once the REQUEST or NACK for a packet. When the $BC$ bit for a packet is set, a node needs not to forward duplicate NACKs or REQUESTs for the packet. Also by this implicit aggregation mechanism, when the needed data packet comes back, the forwarder uses wireless broadcast rather than multiple unicasts to serve multiple members waiting for the same lost data packet.

Like other distance vector schemes, in the PBDV table each node only keeps track of the best next-stop and minimal distance to the target destination which caches the packet. Unlike those destination-based distance vector schemes, PBDV is packet-based, hence the address of the (remote) destination is not needed in PBDV.

Distance vector advertisement in PBDV is exchanged via piggybacking on NACK packets (and REPLY packets if there is any). The packet format of a NACK or REPLY packet is:

| $TYPE$ | $G$ | $S$ | $ECN$ | $SND$ | $RCV$ | $seqNo$ | $[DV]$ |
|---|---|---|---|---|---|---|---|
| 4-bit | 32-bit | 32-bit | 1-bit | 32-bit | 32-bit | 32-bit | N-unit |

where $TYPE$ is the packet type, NACK or REPLY; $G$ is the multicast group address; $S$ is the source address of the application session; $ECN$ is the explicit congestion notification flag (used for congestion control described later); $SND$ is the packet sender's address; $RCV$ is the packet receiver's address, e.g., a broadcast address in a NACK-broadcast or the multicast upstream node toward the source in a NACK-unicast; $seqNo$ is the data sequence number of the first lost packet; and $[DV]$ is a fixed field for piggybacking distance vector advertisement (currently in our simulation $N = 32$ units/packets and 8-bit per unit/packet). In the $[DV]$ field, the $i$-th unit $[DV(i)]$ is the distance metric about packet ID $\langle S, seqNo + i\rangle$ copied from the sender's PBDV table. If no route is known to the sender about packet ID $\langle S, seqNo + i\rangle$, then the $i$-th unit $[DV(i)]$ is set to $\infty$.

In details, a node $A$ updates its PBDV table upon receiving or overhearing a DATA, a NACK, or REPLY packet from node $B$ as follows:

- Upon forwarding a data packet of $seqNo$, the node $A$ caches the packet, then creates or updates the entry

$$\langle (S, seqNo), A, 0, \text{current time}, 0 \rangle.$$

- Upon overhearing a data packet of $seqNo$ not in local cache, $A$ creates or updates

$$\langle (S, seqNo), B, 1, \text{current time}, 0 \rangle.$$

- Upon receiving or overhearing a NACK or REPLY with $seqNo$, $A$ updates its PBDV table iterating on the embedded $[DV]$ list. Note that $[DV(i)]$ affects $A$'s PBDV table entry identified by $\langle S, seqNo + i \rangle$.

  1. If a NACK-unicast packet's $RCV = A$, i.e., $A$ is the NACK's designated forwarder, then $A$ creates an entry for each missing packet $[DV(i)]$

$$\langle (S, seqNo + i), NULL, \infty, 0, 1 \rangle,$$

  with the $BC$ flag set to 1.

  2. Otherwise, if $[DV(i)]$ is less than $\infty$ (i.e., not an invalid entry), and $([DV(i)] + 1)$ is less than the current distance $D$ registered in the corresponding PBDV table entry, then $A$ updates the entry with $B$'s address, the new distance metric, and current time.

### 0.3.5 CORA packet recovery

Upon detecting packet losses, a member first uses PBDV table to locate packets. If still there is a packet loss (i.e., PBDV routing does not cover all losses), it issues an NACK to recover the loss. In MANET, the probability of packet loss is not negligible and thus a NACK for every single loss may cause excessive NACK implosions as well. To avoid this impact, a NACK can be deferred, for example, $Pkt(k)$ is NACKed only when $k \leq seqNo_{new} - N$ (where $seqNo_{new}$ is the newest sequence number received from $S$ and $N$ is currently 32 defined in $[DV]$ field). And further, we can limit the frequency of issuing NACK, say, a member should wait $T_{nack}$ after previous NACK before issuing another NACK if the previous recovery process is still on-going. In addition, NACK transmissions are implemented with random backoff to avoid collision. We describe more details about each recovery category below.

#### PBDV recovery

A member sends a unicast REQUEST message for $Pkt(k)$ if the entry for $Pkt(k)$ is found in its PBDV. For each REQUEST forwarder selected by distance vector routing, if for some reason the forwarder has recently cached the needed packet $Pkt(k)$, it directly sends back the data packet without further forwarding REQUEST. If the forwarder cannot forward the REQUEST because the route is removed from its PBDV due to timeout $T_{max}$, or because the network

is partitioned, it optionally sends a REJECT message to the original requester. Otherwise, it forwards the REQUEST and sets $BC = 1$ in its PBDV table entry for $Pkt(k)$.

After sending/forwarding a REQUEST, the sender/forwarder waits for the needed data packet using a timeout $T_{pbdv}$ (set to 0.5 second in our simulation). If the needed data packet is not received within the timeout or a REJECT message is received, the sender/forwarder removes the route entry from its PBDV table and/or resets $BC$ bit to 0.

Each node invalidates an PBDV entry if the item is not updated for a timeout $T_{max}$ by checking $Ts$ field.

**Local recovery**

In local recovery, a member $A$ broadcasts a NACK to neighbor nodes with $RCV$ field set to broadcast address. After broadcasting a NACK, it sets $T_{local}$ timer (set to 0.3 second in our simulation) to wait for replies from its one-hop neighbors. A neighbor node knowing route information or caching some of the lost packets sends a REPLY to $A$. As we described previously, the REPLY packet includes a $[DV]$ vector for packets identified by sequence number $[seqNo, seqNo + N]$ where $seqNo$ is copied from the NACK message and $N$ is a pre-defined system parameter (32 in our simulation).

Similar to CSMA/CA RTS/CTS handshake's coverage area, which includes both RTS sender's neighborhood and CTS replier's neighborhood, a NACK-broadcast and its multiple REPLY messages cover two hops away from the requester $A$. Therefore, some two-hop neighbors of the requester $A$ can obtain PBDV routing information for all packets within the range $[seqNo..seqNo+N]$. This design is efficient due to two reasons. (1) Like RTS/CTS handshake in CSMA/CA, NACK-broadcast/REPLY handshake uses short packets in wireless transmissions. The communication overhead of such PBDV exchange is the $N*8$ bits $[DV]$ list, which is negligible for a reasonable $N$ value. Because one NACK can be heard by multiple local nodes, there are potentially multiple replies. As CSMA/CA cannot be used in the one-NACK-many-REPLIES handshake, two replies are vulnerable to CSMA hidden-terminal problem if the transmissions (e.g., data transmission) are long. In contrast, it is well-known that CSMA is much more efficient when multiple short transmissions are competing the channel. Therefore, even though some one-hop neighbors can recover some needed packets from their caches, CORA chooses to send back short REPLY control packets rather than longer data packets. (2) With reasonable network density and number of members, there are more intermediate forwarders and members by two hops away. Since receivers often exhibit heterogeneous packet receptions, the two-hop neighbors likely have more packets needed by the requester $A$.

Whenever a node overhear a NACK or an REPLY, its PBDV table is updated accordingly. For the requester $A$, after $T_{local}$ timer expires it should re-process data cache recovery and PBDV recovery again to prevent unnecessary NACK. If all packets are recovered, the recovery process ends. Otherwise, Source recovery is invoked.

**Source recovery**

If local recovery fails, a NACK will be sent to the previous hop toward the source following the source-based multicast structure built by the underlying multicast protocol. After sending an NACK, each receiver sets a timer $T_{source} * T_{backoff}$. After each timeout, it retries the source recovery procedure and doubles the backoff time $T_{backoff}$. After a few retrials, a receiver gives up the recovery and sends the data to the application layer. We use very small number of retrials (e.g., 2) to keep the recovery overhead low.

As we described in PBDV table maintenance, the upstream node updates its PBDV table upon receiving the source-oriented NACK. If the $i$-th unit in the NACK is unknown ($[DV(i)] = \infty$), then the upstream node inserts a new entry $\langle (S, seqNo + i), NULL, \infty, 0, 1 \rangle$ with $BC$ set to 1. If the upstream node can recover some packets in its data cache, it treats these packets as recovered and acts like the source by broadcasting these packets. The neighbor nodes, following "BC" bit, will forward the packets by rebroadcasting them until the needed packets reach the requesters. If all packets are recovered at the node, then this node stops forwarding the NACK to the source. Otherwise, the node updates the $SND$, $RCV$, and $[DV]$ fields in NACK based on its PBDV table, then forwards the NACK to its upstream node again. By this mechanism, duplicate NACKs can be discarded, that is, if the local BC bits for all lost packets in a NACK is set to 1, then the NACK will be discarded. This forwarding procedure is repeated until the source receives the NACK. Upon receiving a NACK, the source locally broadcasts the lost data packets, and the neighbor node with $BC = 1$ will rebroadcasts the data packets until these packets reach the member requesters.

**Discussions**

**Why use distance vector scheme in packet-based routing?** In CORA, PBDV routing tables are obtained opportunistically. Instead of actively transmitting explicit extra route packets, CORA reactively exploits on-going NACK and REPLY messages to maintain PBDV tables.

A distance vector scheme is consistent to this design choice because it incurs minimal communication overhead in packet-based routing. A distance vector scheme is by its nature a localized protocol—a node only needs to know its best one-hop neighbor in data forwarding. In most other DV schemes, routing is destination-based, and the remote destination is a non-local issue. Fortunately in PBDV, routing is packet-based. PBDV is not trying to route a packet to certain destination. In contrast, PBDV's goal is to recover a specific data packet from anywhere which is the nearest place to the local requester. In NACK and REPLY control packets we only need to piggyback tiny distance vector metric (e.g., 8-bit hop-count, or even smaller if MANET's scale is not very large). We choose not to embed other information to elongate NACK and REPLY control packets.

**Possibility of loops in PBDV routing** Like other distance vector schemes,

our PBDV table maintenance mechanism does not prevent a loop. Since a node invalidates an entry by timeouts or receiving REJECT messages, a loop can be formed.

**Example 2** *Suppose A has cached a packet needed by C, and a distance vector driven path for the packet (C→B→A was formed. At node C, this means C's table entry is 2 for the needed packet. As A roams out, B tried to recover the packet from A, but failed. Then B removed the entry, but the optional REJECT message is either not sent or lost in transmission.*

*Now by another NACK or REPLY, C advertises the distance vector information that C is 2 hops away from the caching point of the needed packet. B overhears this message and updates its table entry to be 3 on next-stop C, thus forms a routing loop B→C→B.*

Such loops compromise routing integrity and reduce routing performance. Like sequence based DV schemes (e.g., DSDV [25], AODV [26]), it is feasible to avoid loop forming by using per-vector timestamp or sequence in PBDV vector advertisements. We do not follow this approach because we want to limit the size of NACK and REPLY packets, which must be transmitted using CSMA due to one-sender-many-repliers requirement. If timestamp or sequence is added for each packet in the piggybacked $[DV]$ advertisement list, the elongated NACK/REPLY packet transmissions significantly degrade CSMA's performance.

In contrast, we choose an efficient design to trivialize the threat caused by routing loops. Even if a virtual distance vector loop exists on a needed packet's forwarding path, the following reasons justify CORA's design:

- The efficient 1-bit "breadcrumb" navigation design prevents a REQUEST packet from going around infinitely because $BC$ is set to 1 at first forwarding time, then a loop REQUEST packet is dropped at second forwarding trial as $BC$ is already 1.
- The original requester set a timeout $T_{pbdv}$ to wait for the needed data packet. Its PBDV entry is recycled upon timeout.
- PBDV routing table entries are recycled within $T_{max}$ time.

Therefore, instead of adding per-vector timestamp or sequence to incur extra communication overheads, we allow potential temporary inconsistency in PBDV routing.

### 0.3.6 Congestion Control

While it is recognized that congestion control is essential to realize a reliable multicast protocol in MANET, not much work has been done in this field. In Internet IP multicast, a score of protocols such as TCP-Friendly Multicast Congestion Control (TFMCC) [38], PGMCC [28] and NORM (Nack-Oriented

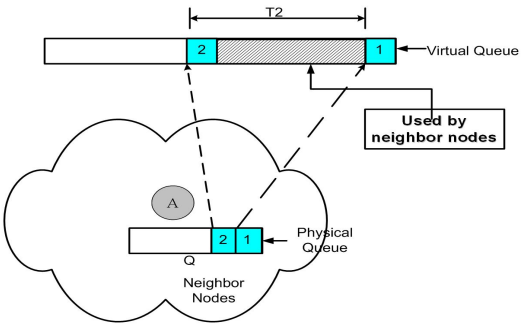


Figure 2: Overview of Congestion Control

Reliable Multicast) [2] have been proposed to extend TCP(-like) unicast congestion control protocol to multiple members. As in TCP, these protocols use packet loss as an indication of network congestion, since the error probability due to other reasons is extremely low in wired network. As MANET experiences comparably high loss rate due to non congestion related reasons such as node mobility, random link error and jamming, this assumption of congestion and loss correlation is no longer valid. The risk of conventional congestion control in a hostile environment, for example is that an adversary could force the progressive shut off of a multicast stream by simply jamming the channel. Thus, a loss differentiation is necessary to accomplish an effective TCP-like multicast congestion control. Several protocols to handle this problem have been proposed as surveyed in [5]. However, those schemes retain the end to end semantics of TCP and attempt to discriminate loss based on end-to-end statistics. For ad hoc, multicast applications the end to end discrimination is extremely difficult because node mobility, multipath routing and path breakage lead to very dynamic fluctuations of end to end statistics such as round round-trip time, packet delivery ratio and bandwidth estimation.

Thus, we believe that some form of reaction from each intermediate node (i.e., active queue management) is unavoidable to achieve an effective multicast congestion control scheme in MANET. Some unicast MANET schemes have proposed to exploit the cooperation of intermediate nodes (e.g., Ad hoc Transport [35] and Neighborhood RED [39]). However, to our knowledge, not many cooperative multicast congestion control has been proposed.

In the paper, we mainly focus on developing an ECN based mechanism considering neighborhood congestion. We do not address here "selective" rate reduction to handle different terminal speeds and different congestion conditions in different areas. Instead we assume that for the mission to be properly accomplished *all* the receivers that are connected to the multicast group must receive the same data or, in the case of multimedia, the same rate.

For simplicity, we assume that there is no other traffic except for multicast data (we ignore the control packets such as NACK, REPLY as the overhead is negligible). Further we assume fixed packet size and bandwidth in this study.

We introduce congestion detection mechanism first and ECN-based AIMD rate control following.

To detect network congestion, each node monitors two variables: its own channel queue length and the transmission delay $Tx$ of a packet. Average value of each metric is used in the protocol. Recall that multicast uses MAC broadcast mode and there is no ACK from receivers. Thus the transmission delay $Tx$ of a packet is the interval from when the packet gets to the front of the queue to the time the packet is actually transmitted (e.g., in Fig. 2, $T2$ for packet 2). If the queue is not empty, this interval is equal to the time between two subsequent actual transmissions.

As illustrated in Fig. 2, there are two queues that are of interest in evaluating congestion. One queue is the node's own channel queue. In our scheme, a node suspects network congestion if the average queue size exceeds a threshold $Q$.

Due to the shared medium, a node should consider neighbor nodes' load to determine more accurate load given to the network. To count this fact, we also monitor "virtual", distributed queue represented by all the packets residing in all active neighbor nodes competing for the same channel. The larger this queue, the larger the "stretch" of the transmission time $Tx$ (as the node will defer to other transmissions before starting its own transmission). Note that 802.11 broadcast mechanism does not employ RTS/CTS handshaking and retransmission. Thus, the packet will be deferred only by its own backoff and the transmissions of neighbor nodes. Thus, the transmission stretch, i.e., ratio of actual per-packet transmission delay and $Tx$ is a good indication of the virtual distributed queue congestion.

Typically, if the virtual, distributed queue is congested, the physical queue is congested as well. But not vice versa, there are cases in which only one or a few nodes is congested. Thus, we declare the node congested if both queues are congested i.e., the node's own queue exceeds a threshold $Q$ and the estimated virtual queue size exceeds a threshold $Q_v$. To convert transmission delay to a virtual queue size, we use an algorithm similar to that presented in [39].

When congestion is perceived, a node sets the ECN bit in the next data packet.

We note that the above congestion detection mechanism is not affected by wireless errors since those errors have no influence on carrier sensing and transmission time $Tx$. Wireless errors will of course cause packet loss. But, as we shall see, the source will not slow down its transmission rate unless a specific ECN flag is carried by the NACK.

When a member receives a packet with ECN bit set it checks for recent packet loss (e.g., some sequence numbers are still missing). If so, it sends an NACK with setting the ECN field to 1 toward the source along a source tree. Otherwise, if there have been no losses, the node delivers the packet to application and

ignores the ECN bit. A NACK with ECN bit is forwarded directly to the source (without NACK aggregation) via a source tree. Local recovery on such NACK is not attempted as it may aggravate congestion.

The source upon receiving the NACK packet with the ECN set decreases the rate consistently with the AIMD philosophy.

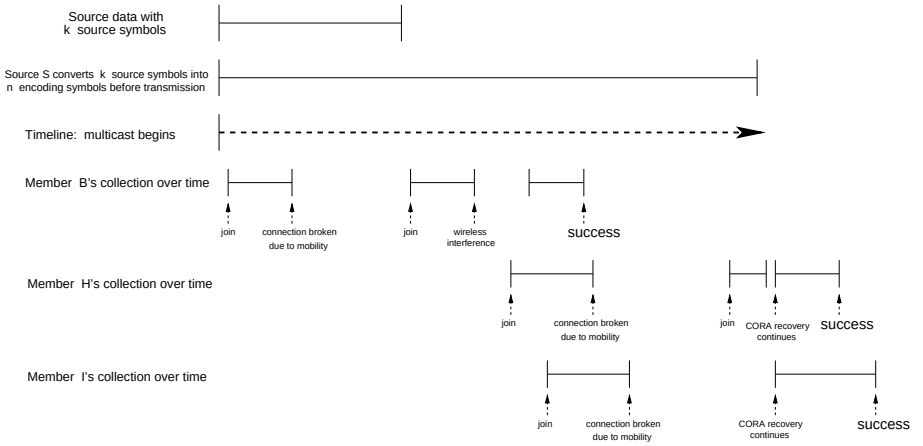


Figure 3: **Interaction between a multicast source and multiple recipients**

In addition, the AIMD scheme based on only NACKs has a potential problem in that NACKs may not be delivered to the source if the network is already congested. A source, however, will keep increasing its rate as there is no NACK (with ECN) and thus lead network collapse. To prevent this failure, we use regulated ACKing scheme. Using a random probability $p$ where $p$ is adjusted to member group size, a member sends an ACK to the source with $p$ probability. We can avoid ACK implosion problem by adaptation on $p$ and ACK aggregation. For example, if $p = 0.01$ and there are 1000 receivers, 10 ACKs will be returned on average from any of the multicast branches. In summary, in the AIMD feedback control scheme implemented in CORA, the source periodically increments its rate only if it receives some ACKs and no NACK with ECN bit set. Either it receives a NACK with ECN bit set or it fails to receive any ACK within a timeout, the source will decrease the rate.

### 0.3.7 Strong Reliability: Collaborative Assured Recovery Algorithm (CARA)

As CORA gives up the recovery of lost packet(s) after a few retrials, it does not guarantee 100% packet recovery. In particular, with high random error probability and mobility, the reliability of recovery process will be degraded. Notably, in such cases, receiver-oriented recovery mechanisms suffer from critically heavy overhead. Thus, we believe that a source-centric recovery mechanism

is inevitable to meet strong reliability, especially in highly lossy environments. CARA, based on our observation, employs digital fountain approach on top of CORA to support deterministic reliability.

In FEC coding schemes, a source data block of $k$ symbols can be encoded into a stream of $n \geq k$ symbols, so that a recipient can restore the original $k$ source symbols from any subset of $k$ encoded symbols in the output stream. Unfortunately, not all FEC coding schemes are implemented so that they support the digital fountain mode of operation. For some popularly used *small block FEC codes* (e.g., Reed-Solomon codes [27], $k$ is in the scale of hundreds), *data carousel* has to be used to address dynamic membership [20, 21]. In other words, although error correction is implemented inside each FEC block, the forward error correction property does not extend across different FEC blocks. Therefore, the source must loop through the transmission of all FEC blocks again and again, and recipients must listen until they have received one copy of each FEC block. Such data carousel schemes normally overload the source and the network because they reapeat large numbers of FEC blocks.

Far superior approximations of the digital fountain are *large block FEC codes* (e.g., Tornado codes [16]) and *expandable FEC codes* (e.g., LT codes [15]). In these coding schemes, a single FEC block can be very large and in fact it can cover the entire source data file. The price paid for this feature is that a recipient needs slightly more than $k$ encoding symbols to restore the original $k$ source symbols. If $k * (1 + \epsilon)$ encoding symbols are needed, we say the *reception overhead* is $\epsilon * 100\%$, e.g., the reception overhead is 5% if $k$*1.05 encoding symbols are needed by the recipients. Byers et al. [4] showed that 5% reception overhead can recover nearly all source data encoded in Tornado codes.

The choice between delay-bounded CORA and strongly reliable CARA is application driven. We assume the symbol size (i.e., number of bits needed to represent coding alphabet) of the underlying large FEC scheme is well known. Given a buffer capacity $k$ suitable for all MANET nodes, a CARA source divides its application data into large blocks of size $k$. Six new packet types are added. Five of them are copies of CORA packet types, i.e., DF_DATA, DF_NACK, DF_REPLY, DF_REQUEST, and DF_REJECT. The format of a CARA packet type is same as its CORA peer, except a $seqNo$ field is replaced by a $\langle blkNo, seqNo \rangle$ composite field. The sixth CARA packet type

$$\langle DFCARA, G, S, blkNo, \text{control message} \rangle$$

is used by a CARA source to advertise digital fountain parameters used in data block $blkNo$, e.g., the values of $k$ and $n$.

A CARA source can put any number of encoding symbols into a DF_DATA packet and multicast it. A CARA member can restore the original source data block, namely $k$ source symbols, upon receiving $k * (1 + \epsilon)$ encoding symbols of the block from a number of DF_DATA packets. Based on Figure 1 used in Example 1, Figure 3 demonstrates a possible scenario for group members $B$, $H$, and $I$ to receive 100% source data encoded by $S$. At the beginning $B$ is source

$S$'s neighbor. After it receives $\approx \frac{k}{3}$ encoding symbols, its session is broken due to mobility. After a while $B$ re-joins the group, it continues to receive another $\approx \frac{k}{3}$ encoding symbols when a long burst of wireless interference interrupts its communication. Fortunately, this does not prevent $B$ from receiving another $\approx \frac{k}{3}$ encoding symbols from source $S$'s unreliable multicast operations. Once $B$ has received $k * (1 + \epsilon)$ encoding symbols, $B$ uses the well-known large block FEC decoder to recover 100% source data.

Member $H$ and $I$ are not as lucky as $B$—since they are multi-hop away, wireless interference, late join, and node mobility significantly decrease the number of encoding packets they received. They failed to accumulate $k*(1+\epsilon)$ encoding symbols from $S$'s unreliable multicast. Nevertheless, CARA inherits CORA's multi-pass recovery design. Whenever $H$ and $I$ detect packet loss, they seek to recover encoding packets by the help from their PBDV tables, their local neighborhood, and the source $S$. Therefore, $H$ and $I$ are able to recover enough encoding symbols to decode 100% source data.

## 0.4 Performance Simulation Study

We implement CORA in QualNet [1], a packet-level network simulator, and investigate its behavior under various conditions. Also, we simulate CARA using QualNet in a highly mobile network. Our simulation study consists of three parts. (1) We investigate the performance of PBDV in various scenarios. The main purpose of this experiment is to demonstrate that CORA's recovery process achieves much better reliability than UDP with only very small extra overhead. Moreover, we will also study the effectiveness of an alternate, very simple FEC scheme. Since the digital fountain approach is not a good solution for a streaming data due to its latency, we study a simple FEC coding scheme which is less powerful than digital fountain coding scheme but requires less latency and thus can be applicable to a multimedia data transmission. (2) Next, we wil show the advantages of CORAs congestion control scheme compared to existing multicast congestion control protocols. In this experiment, we will also show the robustness of CORA against random errors. (3) Finally, CARA will be evaluated using a large scale mobile network scenario. The experiment aims to clarify the usefulness of the CARA in highly lossy networks.

In our study, ODMRP (On-Demand Multicast Routing Protocol) [11] is used as the underlying ad hoc multicast routing protocol. For the simulation study, we use the IEEE 802.11 DCF MAC and the two-ray ground path-loss model for the channel. A node's transmission range and bandwidth is 376m and 2Mbits/sec, respectively. Each simulation run continues 200 seconds and all results are averaged over several runs with various random seeds. In our simulation, we use recovery bound = 12 seconds. Thus, a recovered packet will be delivered to the application only if it is recovered within 12 seconds after sending an NACK. Also, we use $N = 32$, thus each NACK carries route information about 32 consecutive packets. For timeout values, we use $T_{nack} = 5$ seconds, $T_{pbdv} = 0.5$ seconds, $T_{local} = 0.3$ seconds and $T_{source} = 4$ seconds. We

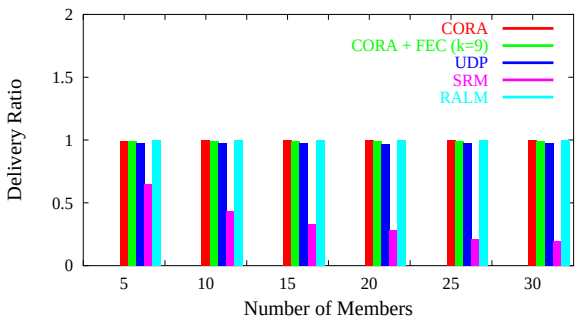


Figure 4: Delivery Ratio in static network

use the maximum numTry = 2, thus a NACK will be (re)transmitted toward the source at most two times.

### 0.4.1 Performance Investigation on Loss Recovery Process

In this section, we evaluate the recovery process of CORA and the effectiveness of a simple FEC scheme in three representative scenarios: (1) a static network; (2) a mobile network and (3) a static network with lossy channel (due to random errors). To focus on the loss recovery part of CORA, the congestion control scheme of CORA is not included in this experiment. The studied FEC coding scheme works as follows: a source generates an extra parity packet every $k$ original data packets (assuming a fixed data size). Thus, a receiver can generate $k$ original packets by receiving any $k$ packets out of $k + 1$ messages.

Commonly for three scenarios, 100 nodes are randomly placed in 1500 $\times$ 1500 $m^2$ field. Through this experiment, we use only one multicast group with a single source. Without congestion control scheme, we use CBR (Constant Bit Rate) application with 10 Kbytes/sec rate using 512 bytes fixed packet size. As references, we use SRM (Scalable Reliable Multicast) [9][8], RALM (Reliable Ad hoc Lightweight Multicast) [36] and UDP (User Datagram Protocol). We exclude gossip-based protocols such as Anonymous Gossip [6] or Route-Driven Gossip [17] because the approach of gossip-based protocol is totally different from CORA and there is no easy way to fairly compare these two different approaches to our knowledge.

First, we present the performance of CORA and the FEC scheme in a static network with various numbers of members from 5 to 30 in step of five. In this scenario, with given moderate offered load (10 Kbytes/sec), a packet is dropped only due to hidden terminal or collision. Note that ODMRP uses IEEE 802.11 DCF broadcast mechanism without RTS/CTS handshaking and retransmission.

Fig. 4 and 5 show the result. In static network with given moderate offered load, the delivery ratio of UDP is very high. Fig. 4 shows that UDP achieves

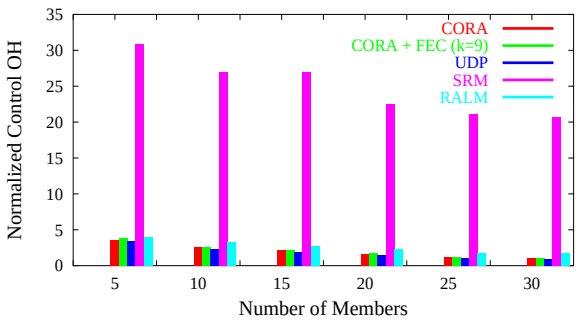


Figure 5: Normalized control overhead (i.e., total number of sent packets v.s. total number of delivered packets) in static network

| Group size | 5 | 10 | 15 | 20 | 25 | 30 |
|---|---|---|---|---|---|---|
| Avg Distance | 2.84 | 2.55 | 2.27 | 1.92 | 1.85 | 1.80 |

Table 1: Average recovery distance of a packet

approximately 97% delivery ratio in this case. In other words, about 2 or 3% of total packets is lost due to the hidden terminal and collision.

In the result, we observe three facts. First of all, the extra overhead of CORA to that of UDP is very small (less than 10%) and actually decreased as the number of members increases. As the group becomes denser, the success probability of local recovery will grow. Table 1 shows the average distance of the transmission of a recovered packet. The average distance of a recovered packet is less than 2 (recovered within two hops away) with more than 20 receivers and keeps decreasing with density. Moreover, even with such a small extra overhead, the delivery ratio of CORA approaches the upper bound given by RALM. We also note that CORA limits the recovery bound, thus all packets are recovered within the bound of 12 seconds.

Secondly, SRM fails even in the static network due to superfluous overhead. The delivery ratio of SRM becomes even worse than that of UDP. Note that we observe about 100% reliability with SRM using lower offered load (5Kbytes/sec) in the same scenario (this result is not included in the paper). In SRM, a receiver sends a NACK to the entire group to recover packet loss(es). To locate multicast members, this recovery procedure invokes Join Query flooding which aggravates the network contention and congestion and thus results in the increase of packet losses with CSMA mechanism. More packet losses will generate more NACK packets even worse. Finally, SRM suffers from excessive uncontrolled recovery overhead as shown in Fig. 5. The normalized control overhead of SRM is more than 500% of that of UDP. This result shows that the reliable multicast protocol designed in the MANET should minimize the control overhead and also consider the congestion control. Without controlling network congestion, injecting extra packets to recover the lost packets may dramatically aggravate network congestion and thus performs even worse than unreliable UDP. As we

| Mobility | 20 | 40 | 60 | 80 | 100 |
|---|---|---|---|---|---|
| Avg Loss | 4.23 | 5.57 | 6.7 | 6.84 | 7.91 |

Table 2: Average number of lost packets at each NACK issue

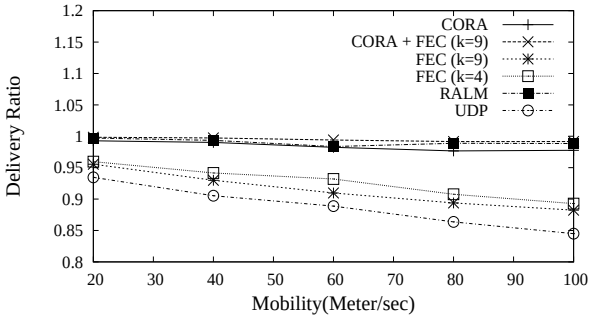


Figure 6: Delivery Ratio in mobile network

clearly demonstrate that SRM does not work well in the MANET scenarios, we omit the result of SRM in the following results.

Lastly, FEC works efficiently with single packet losses, although it will increases the overhead and end-to-end latency to wait all necessary $k$ packets.

Now, we add node mobility in the network. Each node in the network moves following random-way point model with min speed "0" and max speed "x"(x = 20 to 100 meter/sec) and 0 pause time. In this simulation scenario, we use randomly chosen 10 receivers in a group with a single source.

The results are shown in Fig. 6 and 7. We first note that even though the redundancy by FEC coding slightly improves the delivery ratio compared to UDP, the benefit of FEC coding scheme is not significant in the presence of node mobility. The main reason of this outcome is the fact that, often a loss

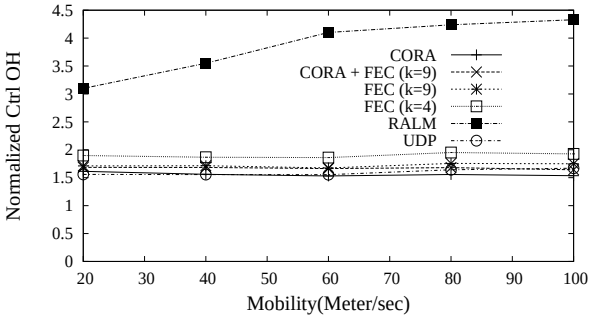


Figure 7: Normalized control overhead in mobile network

| Mobility | 20 | 40 | 60 | 80 | 100 |
|---|---|---|---|---|---|
| CORA | 22.07 | 28.32 | 33.81 | 40.85 | 38.83 |
| CORA + FEC(k=9) | 12.45 | 19.41 | 28.99 | 35.26 | 34.5 |

Table 3: Recovery overhead (number of NACKs or transmitted pkts/sec)

event comprises multiple consecutive packet losses in mobile networks rather than a single packet loss. Once a link breakage happens, it takes a while to detect the link breakage and discover a new path in most multicast protocols. During that period, packets fail due to the lack of route. Table 2 illustrates the average number of lost packets at each NACK issue. We count the all lost packets even though some of them are already recovered by PBDV recovery before a receiver issues an NACK.

However, FEC considerably lessens the burden of recovery by improving the delivery ratio. Table. 3 shows the recovery overhead which includes the total number of NACKs and retransmitted data packets per 1 second. We compare the overhead of CORA with and without FEC scheme with $k = 9$. The potential advantages of using FEC in reliable multicast are (1) FEC improves the delivery ratio and thus decreases the number of loss events; (2) FEC allows more flexible data recovery in that a node needs to recover any $k$ packets out of $k + l$ packets ($l$ is the number of extra coding packets). Thus, it relaxes the reliability of recovery process.

Secondly, the delivery ratio of CORA slightly decreases with node mobility. The main reason of this imperfection is the fact that the node mobility also degrades the reliability of loss recovery process (the reliability of NACK packets and data packets). As CORA gives up the packet recovery after a few trials (say two), some packets are not recovered. And the higher node mobility results in the larger probability of such a failure. However, still, CORA keeps very low extra overhead with much better reliability compared to UDP.

Lastly, RALM pays significant extra overhead to fulfill a strong reliability. Still, it does not achieve 100% in this case as RALM also gives up the recovery for a packet after considerable number of trials. It pays more than 100% extra overhead with high mobility (from 40 to 100). The control overhead will become larger as we increase the number of members and network size. This result of RALM infers that a receiver-responsible approach to achieve a strong reliability in the presence of very high loss rate suffers from extremely heavy overhead and thus may be not very efficient.

The simulation study in the presence of random error follows. For this experiment, we simulate random errors by using a receiver-side random drop where a packet is dropped at a receiver. To randomly drop a packet, each receiver chooses a random number $r$ in the range [0, 1] whenever a new packet comes in. If $r$ is less than $p$, the given random error probability, a receiver drops the packet. Otherwise, it receives the packet. Note that a receiver indicates a

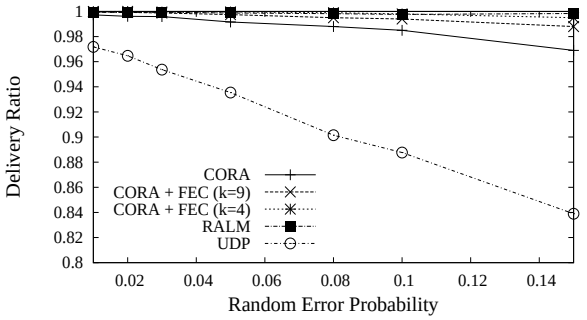


Figure 8: Delivery Ratio with variable random error probability

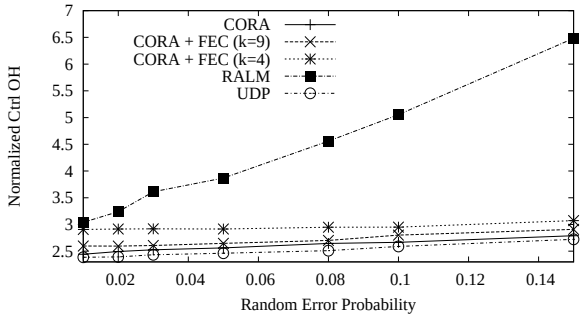


Figure 9: Normalized control overhead with variable random error probability

node who hears the packet at each hop, different from a member in a multicast group. We increase the random error probability from 0.01 to 0.15.

Fig. 8 shows the delivery ratio and Fig. 9 demonstrates the normalized overhead. In the results, three interesting facts are remarked. First, FEC coding scheme efficiently helps CORA in the presence of random error. A loss event due to the random error generally includes a single packet loss. One evident trend is shown in Fig. 9 is that the effectiveness of CORA with FEC increases with error probability (we can monitor this fact by control overhead change). Control overhead of CORA considerably grows with high error rate (0.15) compared to with low error rate (0.01), but the difference of control overheads with CORA + FEC (k=4) in two points is negligible.

Secondly, like high mobile cases, CORA cannot achieve very high reliability with high error probability. However, it still improves the reliability up to 13% at all cases with less than 10% extra overhead compared to UDP.

The shown results clearly demonstrated that CORA achieves far better reliability than UDP (in all cases, it achieved more than 95% reliability) with minimal overhead. Also, the results show that a simple FEC scheme can be very useful in the presence of wireless errors.

### 0.4.2 Study on Congestion Control

We now investigate the performance of our proposed congestion control scheme compared to RALM and TFMCC (TCP-Friendly Multicast Congestion Control) [38]. In TFMCC, a sender chooses the CLR (Current Limiting Receiver) exhibiting the worst throughput using TCP throughput equation [22] among members. Each TFMCC receiver calculates the permissible throughput from a predefined equation using as inputs the packet error rate $p$ and the round-trip time $RTT$ upon receiving a packet. The CLR feedbacks the estimated throughput $R_e$ to the sender using TCP-like acknowledgement, and the sender increases the rate if the current rate is less than $R_e$ and adjusts the current rate to $R_e$ otherwise. As TFMCC is developed for IP multicasting, a TFMCC receiver assumes that a packet loss indicates network congestion. Thus, TFMCC may be not robust against wireless errors due to node mobility, random error or jamming.

In this experiment, we use the same network scenario as in previous experiment but increase the number of groups to five. A source and randomly chosen ten receivers form a group. For RALM, we use CBR application with 5 Kbytes/sec data rate. We use static network with varying random error probability from 0 to 0.1.

The result demonstrates a few interesting properties. First of all, CORA achieves a fairly good throughput compared to TFMCC and RALM. Without considering wireless losses, the throughput of TFMCC is much lower than that of CORA. Moreover, the throughput of TFMCC considerably degrades with increasing random error probability. As RALM sacrifices the throughput to achieve 100% reliability, it achieves lower throughput than TFMCC and CORA.

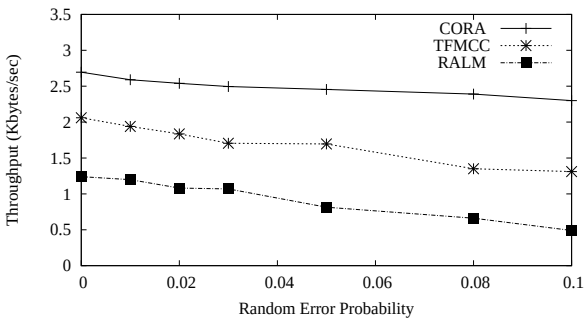


Figure 10: Throughput with congestion control

| p | 0 | 0.01 | 0.02 | 0.05 | 0.1 |
|---|---|---|---|---|---|
| CORA | 0.95 | 0.96 | 0.94 | 0.97 | 0.96 |

Table 4: Fairness Index

Secondly, the throughput degradation of CORA with the increment of random error probability is not significant once the loss of capacity caused by retransmissions is accounted for. As shown in previous experiment with random error, the control overhead with high error rate (0.10) increases approximately 10% from that with low error rate (0.01). Considering this, the throughput degradation (about 10%) from $p = 0.01$ to $p = 0.1$ is very reasonable. We also test CORA with varying mobility. Due to the page limitation, the result are not included in the paper; the general behavior however is similar to that with random errors in Fig. 10. These results confirm that the congestion mechanism of CORA is tolerant to random losses (caused by mobility and by random channel errors). It can thus properly discriminate between congestion and random loss.

Now, we show the fairness index [10] to show the fairness of CORA. We measure the throughput ($x_i$) of each source at different group and calculates the fairness index as follows:

$$f(x_1, \cdots, x_5) = \frac{(\Sigma_{i=1}^{5} x_i)^2}{5\Sigma_{i=1}^{5} x_i^2} \tag{1}$$

Table 4 shows the fairness index for different random channel error rates. As shown, CORA does not starve any flow among five multicast flows and shows fair throughputs.

### 0.4.3 Effectiveness of CARA

As our last experiment, we simulate digital fountain approach in highly mobile network. In this experiment, we want to show that (1) a smart source-oriented approach is desirable to achieve a strong reliability with reasonable overhead;

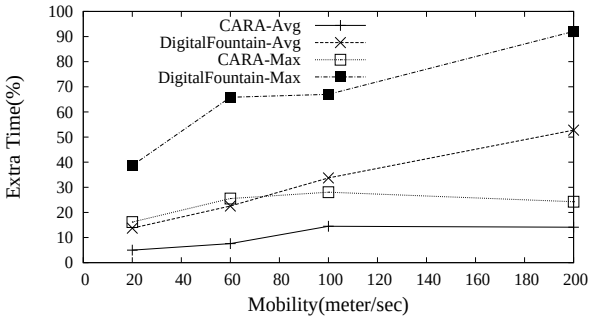


Figure 11: Total Extra Time (%) (It shows the percentage (over $Time_{org}$) of difference between ($T_{leave}$-$T_{start}$) and $Time_{org}$ where $Time_{org}$ is the total transmission time to send $K$ original packets.)

(2) even with digital fountain approach, CARA gives benefit with small extra cost. We measure the total time to receiver all necessary packets with CARA and only digital fountain coding scheme. We measure the time = $T_{leave}$ - $T_{start}$) where $T_{start}$ represents the time of first packet arrival at a receiver and $T_{leave}$ is the time when a receiver receives enough packets. For the high loss rate, we use high node mobility in a large scale network where 500 nodes are randomly placed in 3000 x 3000 $m^2$ field. We use a multicast group with a source and 20 members. A source pumps in packets with 2Kbytes/sec rate wit 512 bytes packet size. For the simulation we use $N$ (number of encoded packets) = 400 and $K$ (number of original packets) = 200. We increase the node mobility with minimum speed = 0 and maximum speed = $x$ meter/sec (the range of x is [20, 200]).

Fig. 11 shows the percentage of extra time to correctly deliver $K$ * (1 + $\epsilon$) packets ($\epsilon = 0.05$) compared to the total time to transmit $K$ packets at the source. In the result, the average extra time over all receivers and the maximum extra time measured at the worst receiver are shown. The result demonstrates that CARA notably reduces the total time to receive all necessary packets with the aid of recovery process. The benefit of recovery process becomes significant with increase of node mobility. Also, CARA achieves 100% delivery ratio with less than 60% extra overhead of time.

## 0.5 Conclusion

In the paper, we presented two MANET reliable multicast protocols: ***C****ollaborative* ***O****-pportunistic* ***R****ecovery* ***A****lgorithm* (CORA) and ***C****ollaborative* ***A****ssurred* ***R****ecovery* ***A****lgorithm* (CARA).

MANET is extremely vulnerable to network overload. Thus, a reliable multicast protocol, which generally incurs extra recovery overhead, should be designed to function with minimal overhead. Further, congestion control is essen-

tial to realize the reliable multicast protocol since uncontrolled overload causes high loss rata and following makeup recovery overhead will aggravate network congestion.

CORA addressed above two problems by developing Packet-Based Distance Vector (PBDV) and virtual queue based congestion control. These designs do not add significant communication cost to underlying reliable multicast protocol: (1) Data caching incurs zero communication overhead. (2) PBDV is an on-demand distance vector scheme opportunistically maintained by promiscuous listening and short control messages. (3) CORA's passive congestion detection incurs zero communication overhead, and ECN only adds 1 extra bit in the messages.

Through our simulation study, we demonstrated three important features of our proposed ideas. (1) CORA improves reliability with minimal extra recovery overhead. The localization of CORA improves the scalability with group size. (2) CORA's congestion control scheme works effectively in MANET multicast in spite of its simplicity. (3) CARA shows that digital fountain approach fits well in dynamic topology in MANET.

In the future, we will enhance the proposed schemes and do more comparative study with recently-proposed reliable multicast protocols in MANETs. Critical enhancements include robust and secure countermeasures against enemy's attacks. Besides, we will investigate the performance of CORA and CARA for heterogeneous multicast applications, such as audio/video conference, multimedia streaming, file download, remote command execution, and mobile groupware.

# Bibliography


[1] Scalable networks, http://www.scalble-solutions.com.

[2] B. Adamson, C. Bormann, M. Handley, and J. Macker. NACK-Oriented Reliable Multicast (NORM) Building Blocks. `http://www.ietf.org/internet-drafts/draft-ietf-rmt-bb-norm-08.txt`, 1996.

[3] K. P. Birman, M. Hayden, O. Ozkasap, Z. Xiao, M. Budiu, and Y. Minsky. Bimodal multicast. *ACM Transactions on Computer Systems*, 17(2):41–88, 1999.

[4] J. W. Byers, M. Luby, M. Mitzenmacher, and A. Rege. A Digital Fountain Approach to Reliable Distribution of Bulk Data. In *ACM SIGCOMM*, pages 56–67, 1998.

[5] S. Cen, P. Cosman, and G. Voelker. End-to-End Differentiation of Congestion and Wireless Losses . In *Multimedia Computing and Networking (MMCN)*, 2002.

[6] R. Chandra, V. Ramasubramanian, and K. P. Birman. Anonymous Gossip: Improving Multicast Reliability in Mobile Ad-Hoc Networks. In *IEEE ICDCS*, pages 275–283, 2001.

[7] L. M. Feeney and M. Nilsson. Investigating the Energy Consumption of a Wireless Network Interface in an Ad Hoc Networking Environment. In *IEEE INFOCOM*, 2001.

[8] S. Floyd, V. Jacobson, C.-G. Liu, S. McCanne, and L. Zhang. A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing. In *ACM SIGCOMM*, pages 342–356, 1995.

[9] S. Floyd, V. Jacobson, C.-G. Liu, S. McCanne, and L. Zhang. A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing. *IEEE/ACM Transactions on Networking*, 5(6):784–803, 1997.

[10] R. Jain. How to measure quantitatively?,". In *ATM Forum Tra#c Management Working Group, ATM-Forum/94-881)*, 1994.

[11] S.-J. Lee, M. Gerla, and C.-C. Chiang. On-demand multicast routing protocol. *Proceedings of IEEE WCNC*, 1999.

[12] B. Levine and J. Garcia-Luna-Aceves. Improving Internet Multicast with Routing Labels. In *IEEE ICNP*, pages 241–250, 1997.

[13] J. C. Lin and S. Paul. RMTP: Reliable Multicast Transport Protocol. In *IEEE INFOCOM*, pages 1414–1424, 1996.

[14] J. R. Lorch and A. J. Smith. Software Strategies for Portable Computer Energy Management. *IEEE Personal Communications Magazine*, 5(3):60–73, 1998.

[15] M. Luby. LT Codes. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 271–282, 2002.

[16] M. Luby, M. Mitzenmacher, A. Shokrollahi, D. Spielman, and V. Stemann. Practical Loss-Resilient Codes. In *ACM Symposium on Theory of Computing (STOC)*, pages 150–159, 1997.

[17] J. Luo, P. T. Eugster, and J.-P. Hubaux. Route Driven Gossip: Probabilistic Reliable Multicast in Ad Hoc Networks. In *IEEE INFOCOM*, 2003.

[18] J. Macker and W. Dang. The multicast dissemination protocol (MDP) framework. IETF Internet Draft, draft-macker-mdp-framework-02.txt, work-in-progress, 1996.

[19] E. L. Madruga and J. J. Garcia-Luna-Aceves. Scalable Multicasting: The Core-Assisted Mesh Protocol. *ACM/Baltzer Mobile Networks and Applications, Special Issue on Management of Mobility*, 6(2):151–165, 2001.

[20] J. Nonnenmacher, E. W. Biersack, and D. Towsley. Parity-Based Loss Recovery for Reliable Multicast Transmission. In *ACM SIGCOMM*, pages 289–300, 1997.

[21] J. Nonnenmacher, E. W. Biersack, and D. Towsley. Parity-based loss recovery for reliable multicast transmission. *IEEE/ACM Transactions on Networking*, 6(4):349–361, 1998.

[22] J. Padhye, V. Firoiu, and D. Towsley. A stochastic model of tcp reno congestion avoidance and control, 1999.

[23] E. Pagani and G. Rossi. Reliable broadcast in mobile multihop packet networks. *Proceedings of ACM/IEEE MOBICOM'97*, pages 34–42, September 1997.

[24] S. Paul, K. K. Sabnani, and S. B. J. C. Lin. Reliable Multicast Transport Protocol (RMTP). *IEEE Journal on Selected Areas in Communications*, 15(3):407–421, 1997.

[25] C. E. Perkins and P. Bhagwat. Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers. In *ACM SIGCOMM*, pages 234–244, 1994.

[26] C. E. Perkins and E. M. Royer. Ad-Hoc On-Demand Distance Vector Routing. In *IEEE WMCSA'99*, pages 90–100, 1999.

[27] I. Reed and G. Solomon. Polynomial Codes over Certain Finite Fields. *SIAM Journal of Applied Mathematics*, 8(2):300–304, 1960.

[28] L. Rizzo. pgmcc: a TCP-friendly single-rate multicast congestion control scheme. In *ACM SIGCOMM*, 2000.

[29] J. H. Saltzer, D. P. Reed, and D. D. Clark. End-To-End Arguments in System Design. *ACM Transactions on Computer Systems*, 2(4):277–288, 1984.

[30] E. Shih, P. Bahl, and M. Sinclair. Wake on Wireless: An Event Driven Energy Saving Strategy for Battery Operated Devices. In *ACM MOBICOM*, pages 160–171, 2002.

[31] P. Sinha, R. Sivakumar, and V. Bharghavan. MCEDAR: Multicast Core Extraction Distributed Ad-hoc Routing. In *IEEE WCNC*, 1999.

[32] T. Speakman, N. Bhaskar, R. Edmonstone, D. Farinacci, S. Lin, A. Tweedly, and L. Vicisano. PGM reliable transport protocol specification. *IETF Internet Draft, draft-speakman-pgm-spec-03.txt, work-in-progress*, June 1999.

[33] F. Stann and J. Heidemann. RMST: Reliable Data Transport in Sensor Networks . In *In Proceedings of the First International Workshop on Sensor Net Protocols and Applications)*, 2003.

[34] M. Stemm and R. H. Katz. Measuring and reducing energy consumption of network interfaces in hand-held devices. *IEICE Transactions on Communications*, E80-B(8):1125–1131, 1997.

[35] K. Sundaresan, V. Anantharaman, H.-Y. Hsieh, and R. Sivakumar. ATP: A Reliable Transport Protocol for Ad-hoc Networks. In *ACM MOBIHOC)*, 2003.

[36] K. Tang, K. Obraczka, S.-J. Lee, and M. Gerla. Reliable Adaptive Lightweight Multicast Protocol. In *IEEE ICC*, 2003.

[37] C.-Y. Wan, A. T. Campbell, and L. Krishnamuthy. PSFQ:a reliable transport protocol for wireless sensor networks. In *ACM WSNA)*, 2002.

[38] J. Widmer and M. Handley. Extending equation-based congestion control to multicast applications. *ACM SIGCOMM*, 2001.

[39] K. Xu, M. Gerla, L. Qi, and Y. Shu. Enhancing TCP fairness in ad hoc wireless networks using neighborhood RED. In *ACM MOBICOM*, pages 16–28, 2003.