CORA: Collaborative Opportunistic Recovery Algorithm for Loss Controlled, Delay Bounded Ad Hoc Multicast

Yunjung Yi, Jiejun Kong, Mario Gerla

Computer Science Department, University of California, Los Angeles, CA 90095, USA
email: {yjyi,jkong,mario}@cs.ucla.edu

Joon-Sang Park

Department of Computer Engineering, Hongik University
72-1 Sangsoo-dong, Mapo-gu, Seoul 121-791, Korea
email:jsp@hongik.ac.kr

Abstract

In this paper, we present Collaborative Opportunistic Recovery Algorithm (CORA) designed for multicast multimedia applications with low loss as well as latency constraints in ad hoc networks. CORA is an independent service that can run atop any ad hoc multicast routing protocol. The main features of CORA are *localized recovery process*, *deterministic (as opposed to probabilistic) peer-to-peer recovery*, and *ability to trade off recovery with latency*. A key component of CORA is the Cached Packet Distance Vector (CPDV) protocol for local peer-to-peer loss recovery. CPDV finds and retrieves the nearest copy of the missing packet while providing other useful NACK aggregation features. We use simulation experiments to demonstrate the effectiveness of CORA and explore the tradeoffs of CPDV localized recovery benefits versus memory and processor overhead. In a typical simulation experiment with mobile nodes CORA yields up to 99% delivery ratio as compared to 91% delivery ratio by Gossip. This improvement is achieved with negligible overhead.

Keywords: Ad hoc networks; Multicast; Routing; Reliable

1 Introduction

A mobile ad hoc network (MANET) is a self-organizing mobile network formed by peer nodes using wireless radios. With or without the wired infrastructure, it can establish an instant communication structure for civilian and military applications.

Its minimal requirement on deployment time and space is particular useful in a hostile environment, where preexisting infrastructure cannot be easily acquired or may be damaged/destroyed at any time. Key applications in these scenarios include teleconferencing, disaster relief, data dissemination, and battlefield operations which are group-oriented and mission-critical, requiring both high data reliability and timeliness guarantees. Undoubtedly, reliable multicast is a critical building block to support these applications, even in the presence of random node mobility, frequent route outages, and random external interference.

Reliable multicast has been an active research area in wired IP networks. Various interesting and effective concepts have been proposed in reliable IP multicast protocols including local recovery [16], peer-to-peer recovery [24], randomized gossip-style recovery [2] and NACK aggregation technique [9]. In particular, peer-to-peer recovery approach, where each member peer communicates with other member peers to recover lost packets, attracts our attention since it fits well with MANET multicasting. In MANETs, the probability of location dependent random errors is non-negligible due to wireless link error and node mobility. Therefore, unless all the receivers have experienced the same loss pattern, it is highly likely that each receiver shows heterogeneous packet reception characteristic. Thus peer members can effectively rectify each other. Moreover, peer-to-peer recovery does not rely on specific nodes (such as the source or designated agents), thus it is robust against node and link failure and dynamic topology changes in the network. Lastly, peer-to-peer recovery tends to evenly distribute recovery overhead to the entire group instead of centralizing at certain nodes, and thus it shows better scalability than source-oriented retransmission mechanism.

Applying peer-to-peer recovery in MANETs is however not straightforward. The design choices underlying wired reliable multicast protocols using peer-to-peer recovery mechanism [2][24] are not apposite for MANETs due to their unique characteristics including mobility, limited bandwidth, random packet errors, and frequent outages. If these wired protocols are applied to MANETs directly, they will incur excessive control overhead for maintaining underlying routing structure, and also unreasonable long latency due to frequent route outages and heavily contended broadcast medium.

Recently, Anonymous Gossip [4] and Route Driven Gossip [11] have customized the gossip-style recovery schemes [2] to be fitted in wireless ad hoc networks. In gossip-style approaches, the packet recovery is performed in a peer-to-peer fashion. A receiver attempts to recover lost packets with the aid of a *random* set of members in the group. In "Anonymous gossip" (AG) [4], each peer member sends gossip-requests to local members with higher probability than to remote members. In "Route Driven Gossip" (RDG) [11], each peer member reuses ad hoc unicast routing path, and sends multiple requests to enhance recovery ratio. However, these solutions are *probabilistic* and their effectiveness depends on member geographic layout. In particular, if the group members are placed very sparsely and the reliabil-

ity of gossip-request and retransmission is poor, then these schemes incur the cost of gossiping but fails to improve recovery in a significant way.

Our approach named **C**ollaborative **O**pportunistic **R**ecovery **A**lgorithm (**CORA**) seeks to achieve *deterministic* and *localized* peer-to-peer recovery which *maximizes* recovery efficiency within *bounded* latency. In multi-hop wireless communications, localized schemes are always more efficient than non-localized schemes with respect to route outage, broadcast medium contention, and unpredictable wireless link errors. The key component of CORA, namely Cached Packet Distance Vector (CPDV) protocol, can deterministically locate the best/nearest copy of a lost packet and localizes the recovery process to the greatest extent. Since CPDV is a distance vector (DV) type scheme and enforces on-demand DV exchange, it incurs minimal storage overhead and communication overhead.

The main contributions of this paper are: (1) a localized peer-to-peer recovery strategy that can recover lost packets from the nearest node that stores a copy; (2) a deterministic CPDV (cached packet distance vector) implementation that realizes the previous goal; (3) a tradeoff study between localized recovery benefits versus memory and processing overhead, and; (4) a mechanism to enforce delay bound compliance.

There exists a spectrum of semantics of reliable multicast. At the one end, the strictest reliable multicast semantics, 100% packet delivery guarantee, exists. Looser reliability semantics may allow some packet losses but may have other requirements. Throughout this paper we use the term "reliable" or "reliability" to denote a high packet delivery probability and the term "strong reliability" especially to denote 100% packet delivery.

The rest of this paper is organized as follows. Related work follows in Section 2. Section 3 illustrates the operation of CORA. In Section 4, we demonstrate our contributions with the extensive simulation results. Finally, Section 5 concludes this paper and discusses future work.

2 Related Work

This section briefly introduces related work focusing on local recovery and peer-to-peer gossip-style approaches for reliable multicasting.

The idea of local recovery has been widely used in wired reliable multicast to reduce the overhead and improve the scalability. RMTP (Reliable Multicast Transport Protocol [16] achieves local recovery based on a multi-level hierarchy of receivers. In each local region, RMTP selects a Designated Receiver (DR) that caches re-

ceived data and recovers the lost packets of members in its local region. More recently, RRMP (Randomized Reliable Multicast Protocol) [24] has applied previously proposed randomized peer-to-peer recovery [2] for effective local recovery. RRMP groups receivers into a few regions based on geographical locality and let each member to recover packets from receivers in local region and, with small probability, from some receivers in remote region. Hierarchical recovery schemes however is not suitable for a mobile network and therefore is not implemented in CORA. CORA does rely on peer to peer recovery. It extends the peer to peer models to exploit the collaboration of non-member nodes in order to locate local peers. As another difference from wired networks, CORA exploits the "wireless broadcast advantage" through promiscuous listening. Promiscuous listening allows a node to acquire one-hop neighbors' caching status and/or CPDV entries with zero transmission overhead. This design choice minimizes the communication overhead caused by recovery traffic. It also reduces channel contention between recovery traffic and ongoing multicast traffic.

As shown in [13], local recovery is indeed effective for reliable multicast protocols. In particular, in MANETs where acquiring full topology or membership information is challenging, a localized solution is desirable. In spite of its natural fitness of local recovery in MANETs, surprisingly few multicast protocol customizing local recovery in MANET has been developed. Reliable Adaptive Congestion controlled Transport protocol (ReACT) [17], an extension of Reliable Adaptive Lightweight Multicast (RALM) protocol [23], shows an example of attempts to explore local recovery in MANETs. Keeping the concept of existing local recovery approach that exploits only members, ReAct proposes a simple local recovery in MANETs [17]. ReAct allows a member to locally recover lost packets *only* from its upstream member nodes opportunistically discovered on the way back to the source, with no extra effort to find "local" members as CORA does. Thus, the capability of local recovery in ReAct is confined.

The concept of peer-to-peer recovery is not new. Anonymous Gossip [4] and Route Driven Gossip [11] use randomized gossip-style recovery algorithm [2] to improve multicast packet delivery ratio. AG and RDG propose solutions to efficiently locate other members to send requests even with dynamic topology changes. Like CORA, AG [4] involves non member nodes (specifically forwarding nodes in a multicast tree) in the recovery process. However, only a forwarding node or member maintains distance metrics to possible reachable members from itself and exchanges its distance metrics with neighbor forwarding nodes and members. RDG [11] explores the underlying unicast routing table to select members (gossipers) for the recovery request. A receiver sends a gossip request to multiple members towards which the unicast routes are known. Leveraging unicast routing information helps RDG to improve routing efficiency over AG. As a difference from CORA, recovery in both AG and RDG is probabilistic and non-localized, thus may result in expensive and potentially unproductive gossip trials over multihop wireless paths. EraMobile [14], another gossip-style scheme proposed relatively recently, transmits gossip messages

only to physical neighbors, i.e., uses epidemic dissemination, to reduce the cost of locating members for message exchanges. CORA is divergent from these randomized gossip approaches in that (1) it uses deterministic peer-to-peer recovery, and; (2) attempts localized recovery to the greatest extent. CORA also introduces the new concept of accounting for the applications' specific demands such as bounded latency.

Recently, coding approaches to reliable multicast in ad hoc networks are discussed and evaluated [7][15]. In [7] it is shown that the use of a FEC (Forward Error Correction) code, particularly Reed-Solomon codes, helps the underlying multicast routing protocol's packet delivery ratio and in [15] a network coding based protocol was proposed.

3 CORA Protocol Design

It is extremely arduous to develop a reliable protocol which achieves both deterministic reliability and bounded-delay guarantee in MANETs. In general, only the second condition–bounded delay–is strictly demanded in most multimedia (e.g., audio or video) multicasting applications. Those applications favor bounded latency over strong reliability (100% packet delivery). CORA is designed to support multimedia applications and targets to maximize packet delivery ratio while sustaining bounded latency and minimizing recovery overhead. The design choices of CORA rely on the observations of unique constraints of MANETs which are: (1) High error rate and heavy recovery overhead: The error/loss rate (unrelated to congestion) on wireless link varies over time and it may become unacceptable (e.g., above 40%) [5]; (2) Low cost of promiscuous listening: Shared and broadcast nature of wireless medium allows all neighbors to promiscuously accept packets only with negligible extra processing overhead; (3) High communication overhead and comparably low storage/processing cost: Communication overhead is expensive due to low bandwidth and limited power. Memory access and processing consume much less energy than wireless transmissions, and memory resources are nowadays relatively abundant on mobile nodes [6][10][19][22].

With these restrictions and characteristics, CORA trades off memory and processing cost for communication overhead by employing cooperative neighbor nodes keeping a short-term data cache and/or CPDV table. The basic mechanism of CORA is a hybrid approach of localized peer-to-peer recovery and source-oriented retransmission mechanism. Similar to the NACK aggregation technique used in IP multicasting [9][21], each intermediate forwarder in CORA, i.e., each router on the path back to source, aggregates NACK messages to prevent the potential NACK implosion problem. In this section we introduce CORA. The detailed description of the protocol is presented in Appendix.

3.1 Cached Packet Distance Vector

CORA creates and maintains a consolidated recovery structure G' for each multicast group G. This structure G' includes three sets of nodes, group members G_m , forwarding nodes G_f, and *recovery assistant* nodes G_{ra}. The recovery assistants are nodes that can hear the packets from a member node. Thus, $G' = G_m \cup G_f \cup G_{ra}$. The selection of G_{ra} will be further refined in Section 3.5. CORA imposes "short-term data caching" at forwarding nodes and members such that each forwarding node and member keeps the copy of incoming multicast data packets in the cache $Cdata_G$ for T_{max}. A packet in the cache is to be retransmitted if a retransmission request from a multicast receiver for the packet is received. Since a packet in the cache stays only for T_{max}, any retransmission request for the packet received after T_{max} + T_s (where T_s is the time when the packet is stored in the cache) is to be ignored, i.e. no packet is transmitted for the retransmission request. We recommend T_{max} be a multiple of the round trip time (RTT) along the network diameter. The rationale behind is that recovery (or retransmission) requests for a packet in one's local cache is expected to be received within a multiple of the RTT after the time when the packet is stored. The time for a packet to travel from a node to a multicast receiver and the time for a retransmission request to travel back from the receiver to the node constitute an RTT. [1] Data caching at a member node is straightforward as members must assemble the file anyway [11]. Data caching at forwarding nodes is used to improve the success ratio of local recovery by redundancy and to suppress unnecessary retransmissions at each forwarding node. In CORA, each node in G' (promiscuously) listens to the multicast traffic, maintains Cached Packet Distance Vector (CPDV) routing table for the group G ($CPdv_G$), and makes available its own CPDV table to other nodes to help recovering their lost packets within minimal distance and latency. The CPDV table keeps track of the min hop distance and path to each cached packet sequence number. Unlike traditional distance vector schemes, CPDV implements content based addressing, i.e., the index is not a destination address but a packet sequence number. Fig. 1 shows a simple illustration of CPDV. Node C stores min hop paths (of length = 1) to packets 1, 3, and 4. Note that CORA/CPDV assumes that a multicast data packet can be distinguished by a unique identifier, ⟨source address sequence number⟩, e.g., H1 stands for packet number 1 from node H. The sequence number field is increased by 1 at the sender for each new packet.

[1] Considering the fact that the loss of a packet is detected when the next sequence packet is received, it is better for a packet to be stored for T_{max} after the next packet is received. If traffic characteristics of the application are known a priori, a possible alternative for T_{max} is the deadline of packets, e.g., 3 seconds. Usual multimedia applications have delay constraints. A packet received after its deadline, i.e., the delay constraint is violated, is useless and only to be dropped. There is no need for packets to be recovered after their deadlines and in turn there is no need for packets to be stored beyond the point when they become useless.

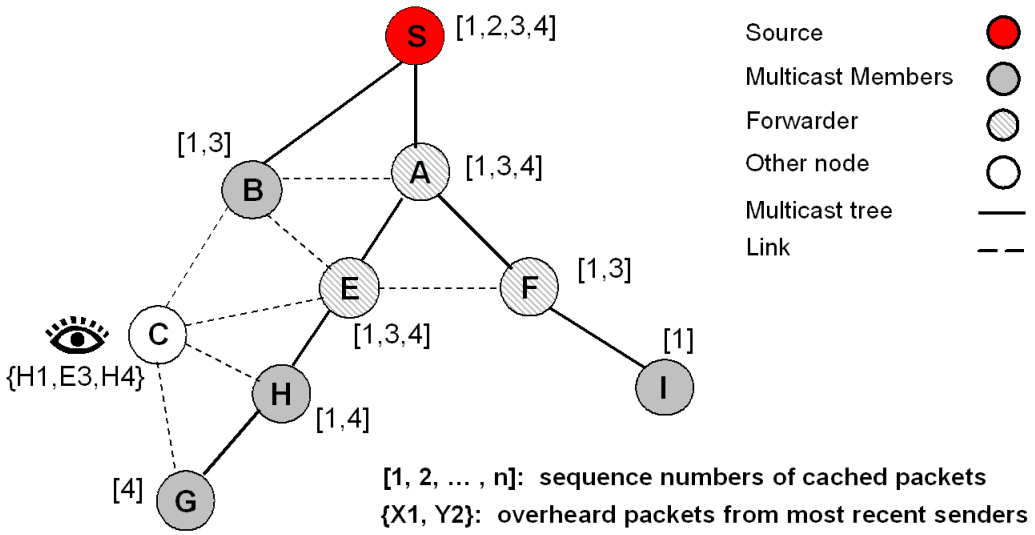


Fig. 1. A sample scenario

CPDV is not updated proactively with explicit messages thus avoiding extra communication overhead. Rather, CPDV routing information is obtained reactively and opportunistically. CORA nodes exploit control message piggyback and promiscuous listening to acquire CPDV routing information as follows: (1) By (over)hearing a data packet, a node knows the packet sender has the packet; (2) Nodes can piggyback their own CPDV metrics in control messages like NACKs. Other nodes overhearing these control messages can compute appropriate CPDV metrics. The piggyback communication overhead is small because each CPDV metric consumes tiny space (8-bit hop count in our simulation).

CORA relies on an independently designed underlying multicast protocol that provides shortest paths to a source as part of a multicast tree (e.g., ODMRP (On Demand Multicast RoutingProtocol) [8] and MAODV (Multicast Ad Hoc On-Demand Distance Vector) [18]). If such shortest path tree does not exist, it is often possible to modify the underlying protocol to acquire it (e.g., MCEDAR (Multicast Core-Extraction Distributed Ad hoc Routing) [20] and CAMP (Core-Assisted Mesh Protocol) [12]). Thus, CORA can run with any MANET multicast protocol that embeds a source tree.

3.2 CORA Recovery Overview

Upon detection of a packet loss (e.g., a skipped sequence number) a multicast group member initiates the loss recovery process which includes two sequential steps:

(1) *Localized peer-to-peer recovery*: A member first tries to recover missing packets in its locality. This procedure is further divided into three sequential sub-

steps:

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

(b) *Local query*: For lost packets with invalid CPDV entries (i.e., CPDV metric for that packet is ∞), the member tries to collect CPDV entries for the missing packets from one-hop neighbors. In the mean time, local query also carries available CPDV entries at the member to distribute multicast state information. This is implemented by an efficient QUERY/REPLY handshake: the member broadcasts a short QUERY, 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. CPDV update metrics are piggybacked in both query and reply messages.

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

(2) *Source recovery*: For packets still missing after the local recovery, the member sends a NACK to its upstream node toward the source until all the lost packets are recovered or the delay bound expires. In MANETs, the probability of packet loss is not negligible and thus a NACK for every single loss may cause NACK implosion. To avoid the problem, NACKs are deferred, aggregated, and paced at the receivers and redundant NACKs are suppressed at intermediate nodes on their way to the source.

3.3 CORA Recovery Example

Example1: 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 a recovery assistant. 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 CPDV 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 CPDV recovery returns no result. Later we will see other nodes, such as G, can take advantage of

CPDV recovery.

(2) H initiates its *local query* process. H locally broadcasts a QUERY to its neighbor nodes. In the QUERY H piggybacks its CPDV metrics, that is, 0 for packet 1 and 4, ∞ for packet 2 and 3. All local nodes update their CPDV metrics for packet 2 and 3 upon hearing this QUERY—there is no CPDV 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 does not matter whether E is H's upstream node or not.) Obviously one QUERY may incur multiple replies. As a result, RTS/CTS (Request To Send/Clear To Send) based CSMA/CA (Carrier Sense Multiple Access/Collision Avoidance) cannot be used. Therefore, in CSMA both query 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 CPDV metrics upon hearing the replies. There is no CPDV 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 to its upstream node E. E cannot recover 2 and then sends a NACK to its upstream node A. The same story repeats and finally S receives the NACK. The source then retransmits packet 2 for H.

(6) NACK implosion due to multiple nodes' simultaneous source recovery is solved by NACK suppression on the multicast tree. Each NACK forwarder sets the BC ("breadcrumb") bit to 1, then resets it to 0 when the missing data packet is returned. In this example, both E and A set their BC bits (for packet 2) during NACK 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 CPDV table knows that packet 1 is one hop away from H and packet 3 is two hops away in the direction of C. Using CPDV 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 QUERY 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 CPDV metrics.

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

3.4 CORA Packet Type

As seen in the above example, CORA implements six different packet types listed below.

(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) QUERY: QUERY packets are used for local query. A member broadcasts a QUERY control message to collect CPDV of lost data packets and meanwhile to advertise its CPDV metrics.

(3) NACK: NACK packets are used for source recovery. A member sends a NACK control message toward the source to complain lost data packets and meanwhile to advertise its CPDV metrics.

(4) REPLY: Upon receiving a QUERY packet, 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 CPDV routing table.

(5) REQUEST: This packet is used to request data packet "retransmission" to a node which has the packet in local recovery. When a member can locate some lost packets using its CPDV entries, it unicasts REQUEST packets to the corresponding next-stops.

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

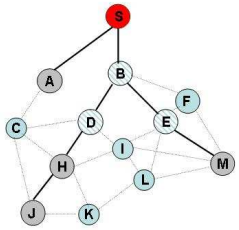
The packet format of a NACK, QUERY or REPLY packet is:

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

where $TYPE$ is the packet type, NACK, QUERY or REPLY; G is the multicast group address; S is the source address of the application session; SND is the packet sender's address; RCV is the packet receiver's address, e.g., a broadcast address in a QUERY or the multicast upstream node toward the source in a NACK; $seqNo$ is the data sequence number of the first lost packet; and $[DV]$ is a fixed field for piggybacking distance vector advertisement. 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 CPDV 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 ∞.

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

(a) An example of optimization scenario (b) Cases of recovery assistant nodes



(c) Optimized recovery structure

Fig. 2. CORA refinements/optimizations

straightforward.

3.5 Algorithm Refinements

We further refine CORA to reduce memory and process overhead at non-member nodes. The basic CORA scheme *volunteers* all nodes which can overhear multicast transmission as recovery assistant (RA) nodes. This simple solution can lead to unnecessary caching and process overhead at a node where CPDV is not of service to any member's recovery process. For instance, in Fig. 2(a) CPDV maintained at node O does not help any member's recovery. Only a member can request a CPDV at a RA node. Thus, a node without any neighbor member does not qualify as RA. In Fig. 2(b) we further analyze the cases where RA's CPDV may help the recovery. Case I: the RA connects a member to at least one forwarding node or another member which is not the direct upstream node (i.e., parent) in the multicast tree (example: nodes C and I). RA nodes satisfying this case allow a member to access data cache and CPDV of a forwarding node or a member within two hops away. Going back to Fig. 2(a), nodes G and K do not qualify as they hear both a member and its upstream. Case II: a node becomes RA if it has at least one member and another RA as neighbors. In this case, two intermediate RA nodes can connect two members which are up to three hops away. Two members more

than three hops away each other can opportunistically exchange state information through combinations of these two cases. For instance, in Fig. 2(a), CPDV entries of member A may be propagated to member M along the path A-C-H-I-M.

To support these refinements, CORA exploits the underlying multicast routing protocol. In this section we explain the CORA refinements based on ODMRP. In ODMRP, the data source establishes and updates group membership and multicast routes on demand. ODMRP uses Query and Reply phases to construct a multicast structure. While a multicast source has packets to send, it floods member-advertising packets, called Join Query, periodically. The periodic floods of the Join Query packet refresh the membership information and update the routes. When the Join Query packet reaches a multicast receiver, the receiver sends back a Join Reply to the source through the reverse path. A node N, say, which is not a member nor forwarding node, becomes a *candidate* RA if it overhears "Join Reply" from a neighbor member (say M). Candidate RA N remembers B the direct upstream node of M. If N hears a "Join Reply" packet or a multicast data packet from any other neighbor that is not B (i.e., there is another forwarding node or member among neighbors), N becomes RA and sets RA_FLAG. Consistent with the *soft state* principle of ODMRP, an RA node resets RA_FLAG if timer expires before refresh. In the example in Fig. 2(c), this scheme excludes node O, N and G from RA qualification.

4 Performance Simulation Study

In this section we investigate CORA behavior under various conditions using QualNet [1]. Our simulation study consists of two parts. The first is performance comparison of CORA with existing schemes and the second is evaluating performance impact of specific CORA design features (e.g., CPDV, refinements, etc).

4.1 Simulation Model

To perform a simulation study, we have implemented CORA in QualNet and we use ODMRP as the underlying ad hoc multicast routing protocol. In our simulations, ODMRP sends Join Query every 3 seconds. MAC(Media Access Control) is 802.11 DCF(Distributed Coordination Function). Radio propagation follows the two-ray ground path-loss model. A node's transmission range and bandwidth is 376m and 2Mbits/sec, respectively. Each simulation scenario uses 100 nodes randomly placed on a 1500 $\times$ 1500 m^2 field and lasts 200 seconds. All the results are averaged over 10 runs with various random seeds. We use recovery bound = 12 seconds and assume that maximum recovery delay is 6 seconds. Thus, CORA attempts to recover a lost packet up to T_{bound} = 6 seconds after the loss is detected. Also, we use

Group size	5	10	15	20	25	30
Avg Distance	2.84	2.55	2.27	1.92	1.85	1.80
Src recovery	0.67	0.39	0.27	0.23	0.13	0.09

Table 1
Average recovery distance & probability of source retransmission of a packet

N = 32, thus each NACK carries route information about 32 consecutive packets. For timeout values, we use T_{nack} = 3 seconds, T_{cpdv} = 0.5 seconds, T_{local} = 0.3 seconds and T_{source} = 4 seconds. We use the maximum numTry = 2, thus a NACK will be (re)transmitted toward the source at most two times. We assume that each node has at least 40 Kbytes space for the CPDV table(s) which can contain about 1000 entries (with T_{max} = 20S and maximum number of (over)heard multicast packets per second is 100).

4.2 Performance comparison study with other protocols

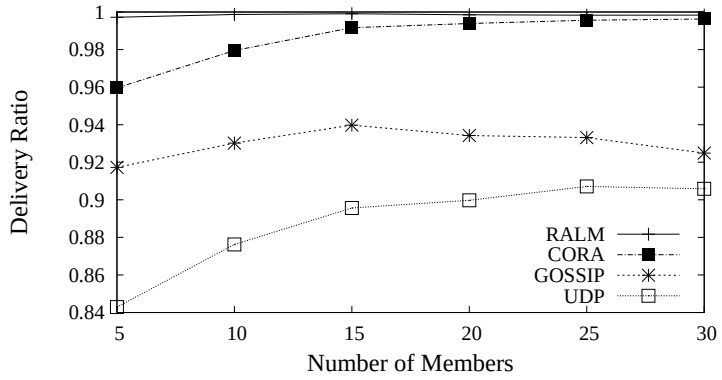
In this section, we evaluate the recovery process of CORA compared to existing reliable multicast protocols in a static network with lossy channel (due to random errors). We compare CORA with GOSSIP, RALM and UDP.

GOSSIP: We implement a simple gossip-style recovery algorithm to compare the efficiency of peer-to-peer random recovery approach to CORA deterministic approach. In GOSSIP, a member periodically sends gossip request (if there is any packet loss) to another randomly chosen member. GOSSIP assumes that each member caches incoming data packets. Upon receiving a gossip-request, the member retransmits the missing packets available in its data cache. In the simulation, a member can issue a new gossip request every 3 seconds. Note that we use static routes among members to avoid routing overhead. Thus, the extra overhead of GOSSIP accounts only for gossip request and retransmitted data packets.

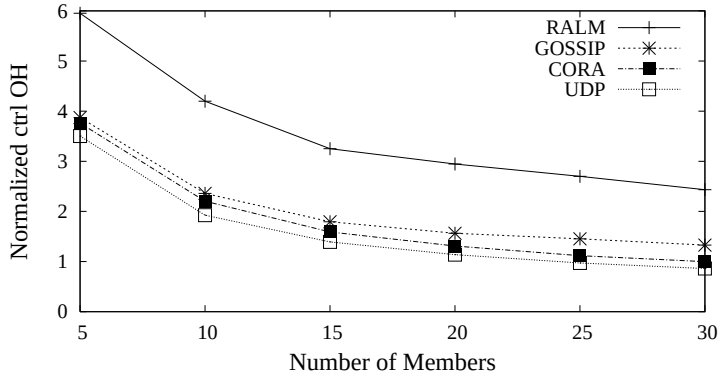
RALM (Reliable Ad hoc Lightweight Multicast) [23]: RALM is a source-oriented retransmission mechanism where a source retransmits the lost packets using a NACK/ACK scheme. When a node detects the packet loss(es), it transmits a NACK to the source without local recovery attempt. Once the source receives 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 received from each requester.

UDP: As a reference, we compare CORA with UDP which does not incorporate any recovery scheme.

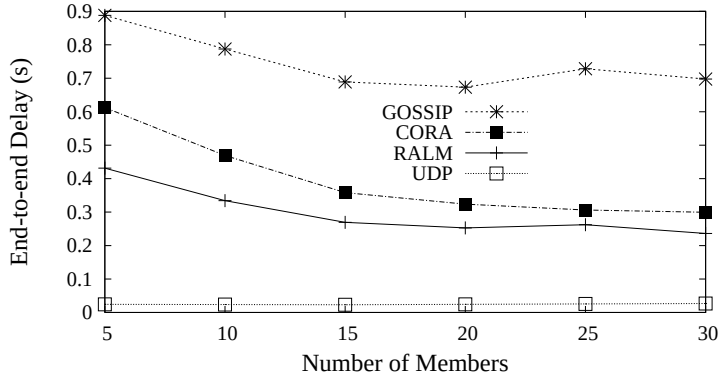
Through this experiment, we assume a single multicast group of variable size (from 5 to 30 nodes), with a single source. The traffic load if very light, so no loss is



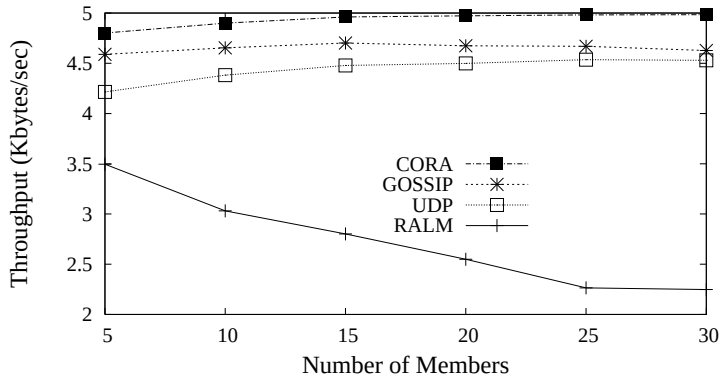
(a) Delivery ratio



(b) Normalized control overhead (Total number of transmitted packets divided by total number of delivered packets)



(c) End-to-end packet latency



(d) Throughput

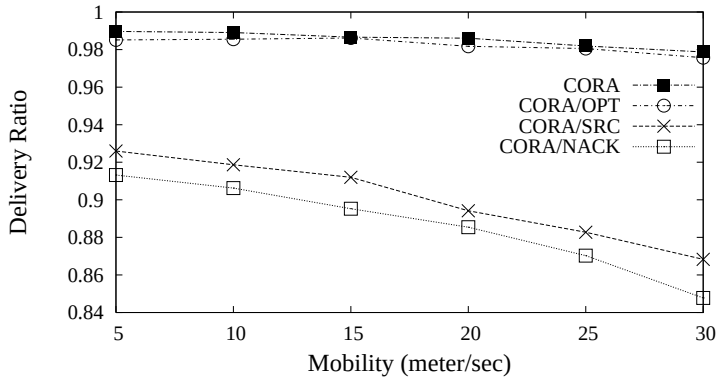
Fig. 3. Results in static scenarios with random error

caused by congestion. All losses are caused by random interference or by mobility (if any). The traffic source is a CBR (Constant Bit Rate) application with 5 Kbytes/sec rate using 512 bytes fixed packet size. We use scenarios with static nodes, i.e., no mobility, and simulate random errors by randomly dropping the packet upon receiving a packet at the MAC layer. Whenever a new packet comes in, each node decides whether it will accept or drop the packet based on the given error probability. We use the byte error rate at each link 10^{-4} and thus the p with a packet of 512 bytes size becomes approximately 0.05.

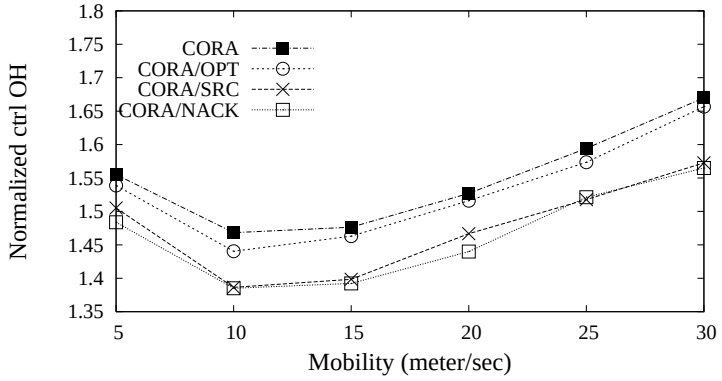
The results shown in Fig. 3 demonstrate the efficacy of CORA compared to other protocols. First, CORA vs UDP. CORA improves the delivery ratio and throughput compared to UDP with very small (less than 10%) extra overhead. In fact, the extra overhead of CORA decreases as the number of members increases because the capability of NACK/retransmission aggregation 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 and the probability of the recovered packet retransmitted from the source. The average distance of a recovered packet is less than 2 (recovered within two hops away), and less than 25% of packets are recovered from the source and more than 75% of the recovery is performed locally with more than 20 receivers and keeps decreasing with group size.

Secondly, CORA vs GOSSIP. The recovery efficiency of CORA is better than that of GOSSIP. As shown in Fig. 3(b), the control overhead of CORA is lower than that of GOSSIP. Also, CORA achieves higher delivery ratio and lower average packet latency than GOSSIP as shown in Fig. 3(a) and 3(c). In fact, the delivery ratio of GOSSIP slightly degrades with group size due to the increase of control overhead. This implies that CORA is more scalable to the group size than GOSSIP approach because of efficient CPDV mechanism.

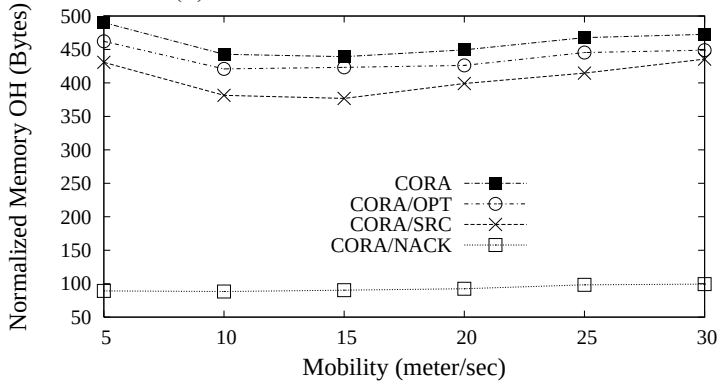
Lastly, CORA vs RALM. Our results show that RALM is not suitable for constant bit rate applications such as periodic dissemination and fixed rate multimedia. Since RALM is designed for 100% reliability, it favors reliability over throughput and overhead and incorporates TCP-like congestion control where a RALM source reduces the transmission rate upon receiving NACK messages from receivers. As a result, RALM achieves lower latency and higher packet delivery ratio than CORA, but it suffers from significantly degraded throughput as shown in Fig. 3(d). Notably, even with far better throughput, CORA achieves delivery ratio and end-to-end latency comparable to RALM.



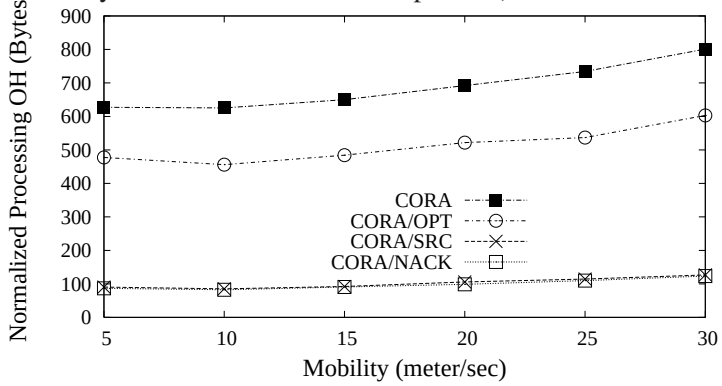
(a) Delivery ratio



(b) Normalized Control Overhead



(c) Normalized Memory Overhead (Total memory access in bytes divided by total number of received packets)



(d) Normalized Processing Overhead (Total processed extra control packets in bytes divided by total number of received packets)

Fig. 4. Results in mobile scenarios

4.3 Performance study of CORA components

In the second experiment, we evaluate the benefits introduced by key CORA design features, namely, CPDV mechanism, data cache at forwarding nodes and optimization/refinement scheme. We compare CORA with (1) CORA/SRC: CORA without CPDV mechanism (no local recovery) so that only NACK aggregation and data recovery at forwarding nodes on the source tree are used; (2) CORA/NACK: same as CORA/SRC, but without data caching at forwarding nodes. Thus, only NACK-aggregation technique is used with end to end retransmission; (3) CORA/OPT: CORA with the earlier mentioned optimization/refinement technique.

For this experiment, we use node mobility without random error where each node moves following random-way point model with min speed "0" and max speed "x"(x = 5 to 30 meter/sec) and 0 pause time. A single group with a source and 10 group members is used in this case.

Fig. 4 illustrates the comparison results. First, Fig. 4(a) shows that localized CPDV directed recovery in CORA greatly improves robustness to mobility as compared to CORA/SRC and CORA/NACK. This is explained by the fact that the CPDV scheme allows each receiver to recover packets from the nearest point so that the success probability of retransmission can be maximized. This is probably the most important result of this batch. Since CPDV recovery is the most unique feature of CORA, this result tells us that there is significant advantage in using it.

The second result is a sort of negative result. It tells us that the performance gain by data cache at forwarding nodes, i.e., performance difference between CORA/SRC and CORA/NACK, is not significant. So, if CPDV is not implemented, it may not be critical to cache packets at non member (forwarding) nodes. Since ODMRP periodically reconstructs the multicast structure and reselects the forwarding nodes following a topology change, cached data packets at old forwarding nodes are "out of reach". Data caching at forwarding nodes is still very helpful for local recovery and CPDV.

Thirdly, the results show that CORA/OPT manages to maintain the same delivery ratio as CORA yet reducing memory and process overhead. In fact, the optimization scheme reduces process overhead of CORA up to 25% as shown in Fig. 4(d). By eliminating unnecessary recovery assistant nodes, CORA/OPT further reduces redundant REPLYs and thus slightly lessens extra communication overhead.

5 Conclusions and Future Work

In this paper, we presented ***C****ollaborative* ***O****pportunistic* ***R****ecovery* ***A****lgorithm* (CORA), a controlled loss, bounded delay multicast protocol. CORA applies to multicast sources with a fixed data rate. It attempts to minimize packet loss rate by exploiting local recovery, with bounds on latency and overhead. The centerpiece of CORA is an efficient local recovery mechanism based on Cached Packet Distance Vector (CPDV). Simulation studies clearly demonstrate the efficacy of CPDV and CORA compared to other reliable multicast approaches.

In the future, we plan to extend the CORA design to handle issues that are critical in multicast applications. The first issue is congestion control. MANETs are extremely sensitive to overload. The multicast mode of operation is particularly exposed to this problem since it lacks a TCP like end to end congestion control mechanism. In this study we assumed that the source rate was fixed a priori, based on careful engineering of resource allocations. In practice, the a priori resource allocation is difficult at best in a mobile environment. Moreover, an increasing number of applications allow the *adjustment* of the source rate to match network conditions (e.g., delay tolerant data delivery, adaptive coded video sources, adaptive resolution data dissemination, etc). It is thus appropriate to develop an extension of CORA that includes source rate adaptation based on end-to-end and network feedback.

The interpretation of network feedback, however, presents its own challenges. A well known challenge in MANETs is the ability to discriminate losses due to congestion from errors caused by link breakage and random interference/jamming. This issue is essential for schemes where feedback from destinations is used to determine the source transmission rate. A source should reduce the rate only if the loss is indeed due to congestion. The local recovery of CORA recovers well from *random type* packet losses due to motion and channel error. In case of wide spread congestion, local recovery does not work and NACK packets will be delivered to the source. Thus, feedback that reaches the source is likely indication of network congestion. We are planning to work on a congestion control scheme which utilizes this implicit loss discrimination feature by CORA.

Local recovery, however, has its limits in recovery from random errors. Source coding (e.g., erasure codes) can also significantly improve packet delivery rate [3]. Moreover, source coding eliminates the repeated retransmission of a conventional recovery scheme and thus might be more suitable for multimedia applications with critical delay constraints. In future extensions of CORA, we will consider mixed traffic environments, with multicast sources with different ranges of delivery ratio and latency requirements. One of the challenges will be the judicious tradeoff between recovery and source coding.

CORA/CPDV uses a single routing metric, hop count. That is, a lost packet is

recovered, if possible, using the shortest path. However, the shortest path is not necessarily the best path in term of recoverability in MANETs. Other metrics that represent reliability, stability, throughput of paths may be better ones. In the future, we plan to adopt or develop other routing metrics that can be used in CORA/CPDV.

Finally, since T_{max}, the time duration for which a packet is to be stored in the cache, is a parameter that can affect the performance of CORA, one of the immediate future works is a simulation study of the impact of T_{max} value on the CORA performance and to find out the best value for T_{max}. Although, T_{max} was recommended be the RTT along the network diameter, since the RTT along the network diameter is difficult to estimate considering the dynamics of MANETs, we plan to develop an algorithm finding a proper value of T_{max} in the future.

Acknowledgements

This work was supported in part by the National Science Foundation under Grant No. 0520332, the US Army under MURI award W911NF-05-1-0246, and the U.S. Army Research Laboratory and the U.K. Ministry of Defence under Agreement Number W911NF-06-3-0001. The views and conclusions contained in this document are those of the author(s) and should not be interpreted as representing the official policies, either expressed or implied, of the National Science Foundation, the U.S. Army Research Laboratory, the U.S. Government, the U.K. Ministry of Defence or the U.K. Government. The U.S. and U.K. Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

References

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

[2] 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.

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

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

[5] D. S. J. DeCouto, D. Aguayo, B. A. Chambers, and R. Morris. Effects of loss rate on ad hoc wireless routing. technical report MIT-LCS-TR-836, March 2002.

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

[7] D. Koutsonikolas and Y. C. Hu. The case for fec-based reliable multicast in wireless mesh networks. In *Proceedings of IEE/IFIP DSN*, 2007.

[8] S.-J. Lee, W. Su, and M. Gerla. On-demand multicast routing protocol in multihop wireless mobile networks. *ACM/Kluwer Mobile Networks and Applications, special issue on Multipoint Communications in Wireless Mobile Networks*, 2002.

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

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

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

[12] 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.

[13] J. Nonnenmacher, M. S. Lacher, M. Jung, E. Biersack, and G. Carle. How bad is reliable multicast without local recovery. In *Proceedings of IEEE INFOCOM*, 1998.

[14] O. Ozkasap, Z. Genc, and E. Atsan. Epidemic-based Approaches for Reliable Multicast in Mobile Ad Hoc Networks. *SIGOPS Oper. Syst. Rev.*, 40(3):73–79, 2006.

[15] J.-S. Park, D. S. Lun, Y. Yi, M. Gerla, and M. Médard. Codecast: A network coding based ad hoc multicast protocol. *IEEE Wireless Communications*, 13(5), 2006.

[16] 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.

[17] V. Rajendran, K. Obraczka, Y. Yi, S.-J. Lee, K. Tang, and M. Gerla. Combining source and localized recovery to achieve reliable multicast in multi-hop ad hoc networks. In *Proceedings of IFIP Networking*, 2004.

[18] E. M. Royer and C. E. Perkins. Multicast operation of the ad-hoc on-demand distance vector routing protocol. In *Proceedings of ACM/IEEE MOBICOM*, 1999.

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

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

[21] T. Speakman, D. Farinacci, and et. Al. PGM Reliable Transport Protocol Specification. IETF Internet Draft, draft-speakman-pgm-spec-05.txt, November 2000.

[22] 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.

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

[24] Z. Xiao and K. Birman. A randomized error recovery algorithm for reliable multicast. In *Proceedings of IEEE INFOCOM*, 2001.

APPENDIX: Detailed Protocol Design

CORA works in two phases: multicast forwarding and loss recovery. In the first phase, while the source sends data packets using any underlying unreliable multicast protocol, CORA maintains CPDV and data cache. Upon detecting packet loss(es), a multicast group member initiates loss recovery process. For sake of simplicity, we explain the recovery process of packets from a single source S in a multicast group G. $Pkt(k)$ refers to a source packet with sequence number k hereafter.

Phase I. Multicast Forwarding

In multicast forwarding phase, every node in G' maintains a CPDV table and every node in the multicast tree G keeps a short-term data cache for the multicast group G.

Each CPDV entry 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.

Upon (over)hearing a new DATA packet	
$P_d = < G, S, i >$ at node n_a from node n_b on time t_s	
if $n_a \in G$	//I am a member or forwarding node
$Cdata_G = Cdata_G \cup P_d$	//add data to cache
$PDV_G = PDV_G \cup \langle S, i, \mathbf{n_a}, 0, t_s, 0 \rangle$	//add PDV entry
else if $n_a \in G_{ra}$	//I am a recovery assistant node
$PDV_G = PDV_G \cup \langle S, i, \mathbf{n_b}, 1, t_s, 0 \rangle$	//add PDV entry
Remove old data cache and PDV entry	

The pseudo code of CPDV and data cache maintenance is illustrated above. Upon (over)hearing a packet, a member or forwarding node keeps the packet in its data

cache and adds a new CPDV entry with distance = 0. A non-forwarding/member node only updates its CPDV entry with distance without data caching for a overheard data packet.

Phase II. Recovery Process

Upon detecting packet loss(es), a multicast group member initiates loss recovery process which consists of following sequential steps. The pseudo code for loss recovery is presented at the end of this section.

CPDV recovery

A member sends a unicast REQUEST message for $Pkt(k)$ if the entry for $Pkt(k)$ is found in its CPDV. If several lost packets have the same next-stop (next hop) in the CPDV routing table, the member aggregates multiple requests into a single REQUEST packet for each next-stop. Note that CPDV recovery can be performed right after a member acquires a CPDV entry for the lost packet without deferring. 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 CPDV 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 CPDV table entry for $Pkt(k)$.

After sending/forwarding a REQUEST, the sender/forwarder waits for the needed data packet using a timeout T_{cpdv} (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 CPDV table and/or resets BC bit to 0.

Local Query

In local query, a member n_a collects new CPDV entries available at one hop neighbor nodes and advertises its own CPDV entries at the same time. A member n_a broadcasts a QUERY packet to neighbor nodes and waits T_{local} (set to 0.3 second in our simulation) for REPLYes from neighbors. Let k be the starting sequence number of the lost packets being recovered at this recovery process. To advertise both lost packets and available CPDV entries, n_a uses $[DV]$ field in QUERY for packets identified by sequence number [k..k+N]. If no route is known to n_a about packet ID $\langle S, k+i \rangle$ (i.e., the lost packet), then the i-th unit $[DV(i)]$ is set to ∞.

A neighbor node knowing route information or caching some of the lost packets sends a REPLY to n_a. As we described previously, the REPLY packet includes a $[DV]$ vector for packets identified by sequence number $[seqNo, seqNo+N]$ where $seqNo$ =k is copied from the QUERY message.

Similar to CSMA/CA RTS/CTS handshake's coverage area, which includes both RTS sender's neighborhood and CTS replier's neighborhood, a QUERY and its multiple REPLY messages cover two hops away from the requester n_a. Therefore, some two-hop neighbors of the requester n_a can obtain CPDV routing information for all packets within the range $[k..k + N]$. This design is efficient for three reasons. (1) Like RTS/CTS handshake in CSMA/CA, QUERY/REPLY handshake uses short packets in wireless transmissions. The communication overhead of such CPDV exchange is the $N * 8$ bits $[DV]$ list, which is negligible for a reasonable N value. Because one QUERY can be heard by multiple local nodes, there are potentially multiple replies. As CSMA/CA cannot be used in the one-QUERY-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 n_a.

Whenever a node in G' (over)hears a QUERY or an REPLY, its CPDV table is updated accordingly following DV routing mechanism. Once T_{local} timer expires and n_a has acquired new CPDV entries for missing packets, it retries CPDV recovery. If all packets are recovered (note that packets with valid CPDV entries are treated as recovered), the recovery process ends. Otherwise, Source recovery is invoked.

Source recovery

For packets which are not recovered by local recovery, a NACK will be sent to the previous hop toward the source following the source-based multicast structure built by the underlying multicast protocol.

NACK implosion problem is exacerbated in MANET since the probability of packet loss is not negligible. To alleviate the problem, we first pace NACK transmissions at the receivers. 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. Also CORA regulates the NACKs by limiting the frequency of NACKs and bounding recovery latency.

Thus, a member should wait T_{nack} after previous NACK before issuing another NACK if the previous recovery process is still on-going. Moreover, a node can issue an NACK for a lost packet only within finite time T_{bound} after the point when the loss for the packet is detected. If a node cannot issue a NACK within T_{bound} due to the limited NACK frequency, which is generally caused by high loss rate and network congestion, it does not attempt to recover the packet.

Secondly, Each upstream node n_b = RCV performs NACK suppression by (1) discarding redundant NACKs, i.e., NACKs for the same data packet; (2) recovering lost packets available in data cache. If the local BC bits for all lost packets depicted in a NACK is set to 1, i.e., requests have been already sent for the packets, then the NACK will be discarded. Again, each NACK forwarder sets local BC bits to 1 for all data packets that a NACK request to retransmit and resets to 0 when it actually receives and transmits those data packets. If the upstream node can recover some packets in its data cache, it treats these packets as recovered and broadcasts these packets. The neighbor nodes, following BC bit, will forward the packets by rebroadcasting them until the needed packets reach the requesters.

For requests not suppressed, the upstream node inserts a new CPDV entry with BC set to 1. For instance, the i-th unit in the NACK is unknown ($[DV(i)] = \infty$) and the request for $Pkt(seqNo+i)$ is not suppressed, then the upstream node inserts a new entry $\langle (S, seqNo + i), NULL, \infty, 0, 1\rangle$ with BC set to 1. Note that if there is an CPDV entry for $\langle (S, seqNo+i)\rangle$ with BC = 0, then it overwrites the old one. 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 CPDV table, then forwards the NACK to its upstream node again.

This forwarding procedure is repeated until the source receives the NACK. The source, like other upstream nodes, locally broadcasts the lost data packets upon receiving a NACK, and the neighbor node with $BC = 1$ will rebroadcasts the data packets until these packets reach the member requesters.

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. We use very small number of retrials (e.g., 2) to keep the recovery overhead low.

Like QUERY packet, NACK packet also carries CPDV entries using $[DV]$ field. Thus, a node in G' (over)hearing a NACK packet accordingly updates its CPDV.

The pseudo code for loss recovery

Loss recovery procedure at **member** node n_a for the packet $Pkt(k)$ where $k \leq (seqNo_{new} - N)$

(1) Performing **CPDV** recovery

if $Pkt(k) \notin Cdata_G$ & $CPdv(k) \in CPdv_G$ //$Pkt(k)$ is lost and and CPDV entry is found

SendRequest(G,S,k, $CPdv(k).nextStop$) //REQUEST pkt

Set T_{cpdv} timer

Upon T_{cpdv} timer expires

// Invalidate CPDV entry $CPdv(k)$ if $Pkt(k)$ is not recovered yet

(2) Performing **local query** with starting $seqNo = k$ //Excute every T_{nack} or no current loss recovery process running

QUERY.seqNo = k //set seqNo

for each $i = 0$ to $N - 1$

if $Pkt(k + i) \notin Cdata_G$ & invalid $CPdv(k + i)$ // no CPDV entry for Pkt(k+i)

QUERY.DV[i] = ∞ //set DV invalid

NeedToQuery = TRUE //need to send QUERY

else

QUERY.DV[i] = $CPdv(k + i)$ //advertise CPDV for Pkt(k+i)

if NeedToQuery == TRUE //send Query

SendQuery(G,S,k) //send query to neighbors

Set T_{local} timer

Upon receiving a REPLY with $seqNo = k$

//Update CPDV table

Upon T_{local} timer expires

Retry CPDV recovery if new CPDV entries are acquired and treat those as recovered

(3) Performing **source recovery** with starting seqNo = k' //Excute only after local recovery fails

for each $i = 0$ to $N - 1$

if $Pkt(k' + i) \notin Cdata_G$ & invalid $Pkt(k' + i)$ //packet is lost and CPDV entry is not valid

NACK.DV[i] = ∞ //set DV invalid

NeedToNack = TRUE

else

NACK.DV[i] = $CPdv(k + i).D$ //advertise CPDV for Pkt(k+i)

if NeedToNack == TRUE

SendNack(G,S,k') //to upstream node

Set T_{nack} timer

Loss recovery at a **RA/member/forwarding node n_b upon receiving** a packet from a node

Upon receiving a **REQUEST** for $Pkt(k)$
- **if** $Pkt(k) \in Cdata_G$ //cached data
  - **Retransmit**(G,S,k) //broadcast retransmission
- **else if** $CPdv(k) \in CPdv_G$ //CPDV entry found
  - **if** $CPdv(k).BC$ == FALSE //BC flag is not set
    - $CPdv(k).BC$ = TRUE //set BC flag to forward
    - **ForwardRequest**(G,S,k, $CPdv(i).nextStop$) //forward REQUEST following CPDV entry
    - Set T_{cpdv} timer
  - **else** ignore //the retransmission request for the packet has been already made

Upon receiving a **QUERY** with $seqNo = k$
- Update CPDV table
- **for each** i = 0 to $N - 1$
  - **if** $Pkt(k+i) \in CPdv_G$ //some packets can be recovered
    - QUERY.DV[i] = CPdv(k+i).D //send DV entry
    - **if** QUERY.DV[i] == ∞ //Pkt(k+i) is lost/requested
      - NeedToReply = TRUE //need to send reply
  - **else** // not in CPDV
    - QUERY.DV[i] = ∞
- **if** NeedToReply == TRUE & myID $\neq$ QUERY.SRC //has new DV info and I am not the source
  - **SendReply**(G,S,k, n_c) //send reply
- **else** ignore QUERY and return //no new CPDV information

Upon receiving/overhearing a **REPLY** with $seqNo = k$
- //Update CPDV table

Upon receiving/overhearing a NACK with $seqNo = k$
- Update CPDV table
- **if** NACK.RCV $\neq$ myId //I am not the upstream node
  - drop NACK and return
- **for each** i = 0 to $N - 1$
  - **if** NACK.DV[i]== ∞ & $Pkt(k+i) \in Cdata_G$
    - **Retransmit**(G,S, k+i) //broadcast retransmission
    - NACK.DV[i] = 0 //update DV entry in NACK
  - **else if** NACK.DV[i] == ∞ & CPdv(k+i).BC == TRUE
    - //the nack for the requested packet is already sent
    - do nothing
  - **else if** NACK.DV[i] == ∞ & CPdv(k+i).D $\neq \infty$
    - //this node has CPDV entry for Pkt(k+i)
    - CPdv(k+i).D = ∞ & CPdv(k+i).BC = TRUE
    - NeedToNack = TRUE //need to forward NACK
      - //invalidate CPDV entry & set BC flag
  - **else if** NACK.DV[i] == ∞ //$Pkt(k+i)$ is not recovered
    - add CPdv(k+i) & CPdv(k+i).BC = TRUE
    - NeedToNack = TRUE //need to forward NACK
  - **else if** NACK.DV[i] $\neq \infty$ //CPDV advertise
    - NACK.DV[i] = CPdv(k+i).D //Update CPDV information
- **if** NeedToNack == TRUE & myId != NACK.SRC //I am not the source
  - **ForwardNack**(G,S,k, its upstream node)& forward NACK to upstream node
- **else** ignore NACK and return

Upon receiving a **retransmitted** $Pkt(i)$
- **if** $CPdv(i).BC$ == TRUE //BC flag is set – need to forward
  - **Retransmit**(G,S,i) //broadcast retransmission
  - $CPdv(i).BC$ = FALSE //set BC flag to false
  - **if** $n_b \in G$ //I am a forwarding node or member
    - $Cdata_G = Cdata_G \cup P_d$ //add data to cache
    - $CPdv_G = CPdv_G \cup \langle S, i, \mathbf{n_b}, 0, t_s, 0 \rangle$ //add CPDV entry