DiCoR: Distributed Cooperative Repair of Multimedia Broadcast Losses

Saqib Raza
Dept. of Computer Science
University of California, Davis

Gene Cheung
Hewlett-Packard Laboratories
Tokyo, Japan

Chen-Nee Chuah
Dept. of Electrical & Computer Engineering
University of California, Davis

***Abstract*—Multimedia Broadcast/Multicast Service (MBMS) allows a common broadcast channel to be shared by users interested in identical content. We explore the problem of enhancing MBMS resilience by repairing packets lost during broadcast. Since MBMS broadcast consumes expensive 3G resources, we leverage the ubiquity of multi-homed mobile devices i.e., devices having both cellular and IEEE 802.11 wireless interfaces. We thus accomplish *out-of-band* repair of MBMS packet losses through an ad-hoc, peer-to-peer 802.11-based network. A fundamental challenge in scheduling repair transmissions is handling interference between distributed nodes. We present DiCoR, a fully distributed protocol for CPR. Our protocol does not assume any a priori knowledge of the network topology or peer losses, and is resilient to dynamic network changes due to node mobility or the continuous joining and leaving of peers. Detailed simulation experiments, under realistic loss models and network conditions, demonstrate that DiCoR presents a viable solution for timely out-of-band loss repair of MBMS real-time broadcast.**

I. Introduction

The point-to-multipoint mode of Multimedia Broadcast/Multicast Service (MBMS) [1] allows simultaneous distribution of identical content to multiple users in 3GPP UMTS cellular networks. Expected applications of MBMS include traffic telematics, news broadcast, music/video streaming, sports replay, and file sharing [2]. As opposed to dedicated point-to-point connections, the MBMS point-to-multipoint mode dictates that the coding scheme and transmit power are chosen in advance for a target set of users, implying a lower level of QoS for users with worse receiving conditions [3].

Previously proposed solutions to improve MBMS reliability mainly focus on application-layer Forward Error Correction (FEC) [4]. However, FEC falls short of fixing all errors for heterogeneous users. Using more complex FEC incurs expensive overhead in terms of 3G bandwidth. A possible solution is to have the MBMS base station rebroadcast lost packets. Such retransmission based solutions are challenged by the well known feedback implosion problem [5]. Blind retransmissions, i.e., rebroadcasting *all* packets, or employing strategies [2] that evade feedback implosion have been proposed. Their utility is undermined by consumption of additional 3G bandwidth [6], which is scarce and expensive. Moreover, nodes having lost packets due to poor local receiving conditions also stand to loose the retransmitted packets.

This paper considers *Cooperative Peer-to-Peer Repair* (CPR) for out-of-band repair of 3G broadcasting losses. CPR is motivated by the widespread availability of multi-homed mobile devices having both 3G cellular and IEEE 802.11 wireless interfaces [7, 8]. Such users can leverage IEEE 802.11 peer-to-peer connections to achieve out-of-band repair of 3G broadcasting losses, as depicted in Fig. 1. This paper presents a fully distributed protocol for CPR. The challenge for such a protocol is to schedule repair transmissions in the presence of interference between wireless peers [9]. Furthermore, distributed nodes are not readily aware of their surrounding network topology and distribution of peer losses. Dissemination of such control information, required to guide the scheduling of repair transmissions, incurs overhead. The challenge for a distributed protocol is to realize an efficient tradeoff between this overhead and repair effectiveness.

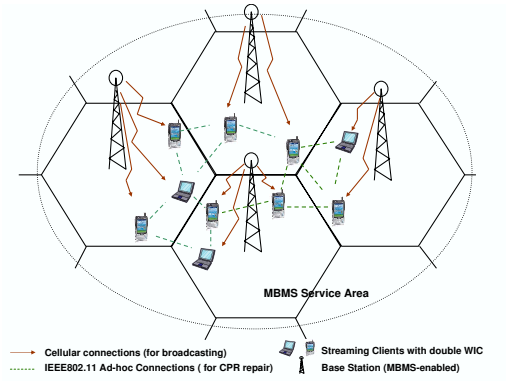


Fig. 1: Cooperative Peer-to-Peer Repair

In our short paper [10], we presented an exhaustive search algorithm that assumes availability of global state information to compute the optimal schedule for repair transmissions. [10] uses the exponential-time algorithm for tractable problem sizes to suggest heuristics and a design framework for a distributed algorithm. This paper presents a comprehensive specification of a completely distributed protocol that is derived from that design framework. [10] also presents a very rudimentary performance analysis using a simple protocol specification, and using simplistic broadcast loss models while ignoring the interaction of the 802.11 MAC layer with the protocol. We evaluate and configure DiCoR using detailed packet-level simulations under realistic conditions and interactions with the 802.11 MAC layer. Our major contributions are twofold:

• We propose a distributed protocol *Distributed Cooperative Repair* (DiCoR). DiCoR incorporates the heuristics identified in our short paper [10]. DiCoR dynamically discovers network

topology, exchanges content availability with peers, and schedules peer-to-peer repair transmissions. We give a complete specification as well as design rationale for DiCoR. Furthermore, we also conduct simulation-based parameter tuning for configurable DiCoR parameters.

• We conduct a detailed packet-level simulation study under realistic loss models and network conditions to demonstrate the efficacy of DiCoR. Our implementation in QualNet [11] includes our DiCoR algorithms, dynamic peer tracking and connection handling, message passing, and interactions of DiCoR with the IEEE 802.11 MAC-layer protocols. We show that DiCoR presents a viable solution for timely out-of-band loss repair of real-time multimedia broadcast such as mobile-TV. DiCoR successfully repairs 100% of the lost packets well before the playout deadline. The protocol converges shortly after the repair completion, minimizing the overhead due to residual request/repair messages.

To the best of our knowledge this is the first work to present a fully-specified distributed solution for CPR. A key feature of DiCoR is batching, wherein the group waits for a batch of broadcast packets before triggering the repair process. The original MBMS stream data rate acts as a self-clocking mechanism for triggering the repair process. DiCoR is designed to be resilient to dynamic network changes due to node mobility and the continuous joining/leaving of peers. We assume that peers are altruistic in that they all run the specified implementation of DiCoR for loss recovery. DiCoR can be enhanced to mitigate malicious or selfish peer behavior, but such mechanisms are not within the scope of this paper.

II. Related Work

The idea of organizing neighboring peers to locally recover packets originated from a faraway broadcast or multicast source is not a new one; early works like [12, 13] explored such idea on wired networks with IP multicast support [14] like MBONE [15]. Obviously, in wireless environments recovery schemes need to consider wireless specific characteristics like interference and mobility that are not present in the wired counterpart, making the problem more challenging.

While there are very few related work for Cooperative P2P Repair (CPR) in the wireless environments, a related and well-studied problem is the *single-source radio broadcast* problem [16–19], where a message by a source wireless node needs to be transmitted to all other nodes in the network. Another related problem is the *multi-source radio broadcast* problem [17], where there are multiple messages, each present at an associated source wireless node, and all messages need to be transmitted to all other nodes. The CPR problem is different from the preceding problems in that a given message may initially be present at multiple wireless nodes. Furthermore, we are required to transmit the packet to only those nodes that do not initially have the packet. In a sense, the single-source and multi-source radio broadcast problems are pathological special cases of the more general CPR problem, where initially only one peer has a particular message and we must repair everyone else. In reality, however, because MBMS [1] transport itself is relatively reliable, the likely case is that a fairly large fraction—at least half—of peers receive the message. Hence a distributed protocol must be tuned to optimize the expected case of CPR rather than the pathological special case.

III. Problem Definition

We define *CPR-bci*, the broadcast version of the Cooperative Peer-to-Peer Repair problem with collisions and interference. *CPR-bci* uses the promiscuous mode of 802.11 for peer-to-peer communication, so that a transmitting node can potentially be heard by all receiving nodes within transmission range at the same time. *CPR-bci* is modeled by a graph $\mathcal{G}(\mathcal{N}, \mathcal{L}_\mathcal{T}, \mathcal{L}_\mathcal{I})$ that has one node set $\mathcal{N}$ and two link sets: a transmission link set $\mathcal{L}_\mathcal{T}$, and an interference link set $\mathcal{L}_\mathcal{I}$. We have a set of messages $\mathcal{M}$. Initially each node n_i is in possession of a set of messages $\mathcal{I}_i \subseteq \mathcal{M}$. We want to schedule transmissions so that all nodes in $\mathcal{N}$ procure every message in $\mathcal{M}$.

Transmissions take place in discrete transmission rounds. Our decision variables are a set of indicator variables of the form $s^t_{i,q} \in \{0, 1\}$, where $s^t_{i,q} = 1$ implies that n_i transmits $m_q \in \mathcal{M}$ at transmission round t, and $s^t_{i,q} = 0$ otherwise. The set comprising of nodes that possess $m_q \in \mathcal{M}$ at the beginning of transmission round t is denoted by $\mathcal{U}^t_q$. It follows that $\mathcal{U}^1_q = \{n_i | m_q \in \mathcal{I}_i\}$. We denote the set of nodes that successfully receive m_q during transmission round t by $\mathcal{R}^t_q$. $n_j \in \mathcal{R}^t_q$, i.e., n_j receives a copy of m_q during transmission round t iff $\exists n_i \in \mathcal{N}$ such that:

• $n_i \in \mathcal{U}^t_q$: n_i has a copy of m_q at transmission round t,
• $s^t_{i,q} = 1$: n_i transmits m_q at transmission round t,
• $s^t_{i,r} = 0$, $\forall m_r \in \mathcal{M} | r \neq q$: n_i does not transmit a message other that m_q at transmission round t,
• $(n_i, n_j) \in \mathcal{L}_\mathcal{T}$: there exists a transmission link that stems from n_i to n_j, and
• $s^t_{k,r} = 0$, $\forall m_r \in \mathcal{M}$ $\forall n_k \in \mathcal{N} | (n_{k \neq i}, n_j) \in (\mathcal{L}_\mathcal{T} \cup \mathcal{L}_\mathcal{I})$: all nodes n_k, $k \neq i$, such that there exists a transmission or collision link that stems from n_k to n_j, do not transmit.

It follows that $\mathcal{U}^{t+1}_q = \mathcal{U}^t_q \cup \mathcal{R}^t_q$. The *CPR-bci* problem is to find a schedule of transmissions that minimizes the total number of transmission rounds required for all nodes to procure every message in $\mathcal{M}$.

IV. Design Issues for Distributed Protocol

The *CPR-bci* formulation presents an abstract model of the real *distributed* CPR problem with certian simplifying assumptions. The design of an actual distributed protocol must address these challenges:

• *CPR-bci* assumes global knowledge of the network topology. In reality, a topology discovery mechanism is needed for nodes to learn about their surrounding topology in a distributed manner, and to adapt to changes caused by node mobility and peers subscribing/un-subscribing to the MBMS service.
• *CPR-bci* assumes *a priori* knowledge of the initial MBMS loss distribution across nodes. It also assumes global knowledge of all scheduled transmissions leading to successful repairs at every other node. In a distributed setting, a node only has information about its own initial losses and subsequent

repair status. Any loss or packet availability information at other peers must be acquired via a distributed protocol.

• *CPR-bci* assumes that peers are synchronized and can coordinate repair transmissions in discrete rounds. Such synchronous transmissions are not possible in the presence of other traffic and interaction with the MAC protocol.

• The dual link sets radio model [19] we use is more general than variants of the Unit-Disk-Graph (UDG) model [17, 18]. However, all such deterministic models fail to capture varying channel conditions that influence whether an attempted transmission succeeds or fails in a real setting.

Although *CPR-bci* represents the ideal case, it highlights scheduling in the presence of interference and collisions and provides insights in developing our heuristic algorithms. In designing a distributed protocol, the key decision for a peer at any point in time is what information to transmit, if any. Discovering the optimal schedule for *CPR-bci* is NP-Hard [20]. However, we previously solved *CPR-bci* optimally for small problem sizes and discovered two bottlenecks [10]: (1) packets that have to travel a large number of hops from the source to a node that requires it; (2) nodes that have a large number of missing packets. This motivates the following dual heuristics:

• All things being equal, a missing packet that has the furthest number of hops to travel should be given higher repair priority than those with less distance to travel.

• All things being equal, a missing packet should be repaired at a node with a larger number of missing packets, prior to repairing a packet at other node.

Section V details our distributed protocol, DiCoR, that employs the above heuristics. Dissemination of control information such as network topology and peer losses is necessary but it incurs overhead. The challenge for DiCoR is to realize an efficient tradeoff between this overhead and the accuracy of the information. A key feature of our protocol is *batching*, where the group waits for a *batch* of broadcast packets before triggering the repair process. A major motivation of batching is that control information spanning the batch can be propagated jointly, which allows the cost to be amortized over the batch.

V. DiCoR: Distributed Protocol for CPR

A. DiCoR Parameters and Preliminaries

DiCoR is an application layer protocol. Distributed peers wait for a batch of packets ($\mathcal{M} = \{m_1, m_2, ..., m_{\text{BATCH\_SIZE}}\}$) to be injected into the network by the 3G MBMS broadcasting station before triggering the repair mechanism for that batch. Every batch has an associated REPAIR_EPOCH, which is the duration of time it takes for the next batch of packets to accumulate. The repair process for the current batch of packets terminates at the expiration of the REPAIR_EPOCH, and the repair mechanism for the subsequent batch begins. The REPAIR_EPOCH is given by $\frac{\text{BATCH\_SIZE}\times\text{PAYLOAD\_SIZE}}{\text{MBMS\_RATE}}$, where PAYLOAD_SIZE is a constant that gives the size of a packet $m_q \in \mathcal{M}$ in bytes. MBMS_RATE gives the data rate at which MBMS packets are broadcast by the 3G base station (note that the MBMS broadcast stream acts as a self-clocking mechanism to synchronize the start of repair epochs at distributed peers). All DiCoR transmissions are 802.11 broadcast transmissions. A DiCoR packet carries control information and may carry a packet $m_q \in \mathcal{M}$ as its *repair payload*. The control information works in conjunction with certain pre-defined DiCoR parameters to schedule transmissions. Specifically these parameters are REPAIR_WAIT, SOLICIT_WAIT, and POST_SOLICIT_WAIT. Their significance is outlined later.

B. DiCoR Local State Information

DiCoR maintains the following local state at a node n_i:

- δ_i : set of received packets at n_i.
- α_i : set of nodes n_i knows to be in its neighborhood (i.e., nodes within transmission range).
- $\Lambda = \{\lambda_j^q\}_{n_j \in \alpha_i, m_q \in \mathcal{M}}$: an $|\alpha_i| \times$ BATCH_SIZE matrix, such that λ_j^q represents a heuristic estimate of the urgency of m_q for a node n_j in n_i's known neighborhood.

C. DiCoR Packet

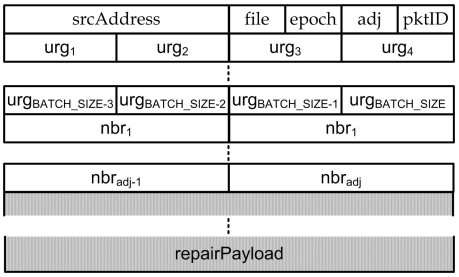


Fig. 2: DiCoR Packet Format

Prior to detailing the operation of the protocol, we outline the information contained in a DiCoR packet (Fig. 2). A DiCoR packet sent by n_i lists n_i's neighbors, as well as a measure of the *urgency* of each packet $m_q \in \mathcal{M}$ for n_i. It may also carry a repair payload $m_q \in \mathcal{M}$. The individual fields of a DiCoR packet ρ are detailed as:

- ρ.srcAddress : sender address.
- ρ.file : ID of the file being repaired.
- ρ.epoch : ID of the exact batch of ρ.file being repaired.
- ρ.adj : size of ρ.srcAddress's known neighborhood.
- ρ.pktID : ID of the repair payload carried by ρ. ρ.pktID $= 0$ implies no repair payload.
- $\rho.\text{urg}_q$: urgency of packet m_q for node ρ.srcAddress. A higher value corresponds to higher urgency. $\rho.\text{urg}_q = 0$ implies ρ.srcAddress is in possession of m_q.
- $\rho.\text{nbr}_k$: address of the k^{th} neighbor of ρ.srcAddress.
- ρ.repairPayload : repair payload carried by ρ. If ρ.pktID > 0, ρ.repairPayload $= m_{\rho.\text{pktID}}$.

D. DiCoR States & State Transitions

Fig. 3 gives a high-level state transition diagram for DiCoR at a node n_i. The states are defined for ease of exposition. All states except *WAIT* have immediate forced transitions out of them. Transitions out of *WAIT* are governed by three events: Send_Timer expire, Epoch_Timer expire, and packet reception. The

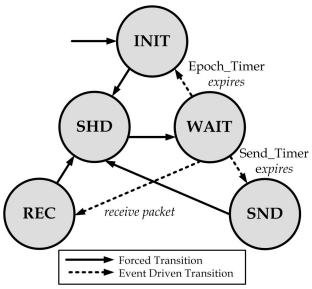


Fig. 3: DiCoR State Transition Diagram

DiCoR State	Description
INIT	Initialize local state information at n_i, and trigger the repair process for the next batch of packets.
REC	Receive a packet and update local state information at n_i based upon the contents of the packet.
SHD	Select the repair payload for a prospective DiCoR transmission from node n_i, and *schedule* it.
WAIT	Wait for Send_Timer expire, Epoch_Timer expire, or reception of a DiCoR packet at n_i.
SEND	Send the DiCoR packet scheduled for transmission and update local state information at n_i accordingly.

TABLE I: Overview of DiCoR States

essential function of DiCoR is to keep local state information updated and schedule transmissions based upon this information. Table I presents an overview of the DiCoR states.

E. Detailed Protocol Description

1) Initialization: At the inception of a repair epoch, Batch_ID is set equal to the corresponding batch. $\delta_i \subseteq \mathcal{M}$ is initialized to the set of packets in Batch_ID correctly received at n_i. A design choice we make for DiCoR is not to preserve topology information across repair epochs. This lends itself well to dynamic network behavior due to node mobility as well as nodes continuously joining and leaving the peer-to-peer network. DiCoR, therefore, initializes $\alpha_i = \{n_i\}$ i.e., the only node in n_i's known neighborhood is n_i itself. DiCoR also needs to initialize λ_i^q, $\forall m_q \in \mathcal{M}$, which is an estimate of the urgency of m_q for n_i. As will be evident later, repair transmissions to n_i are guided by n_i's advertised packet urgencies. Section IV stipulated that greater priority ought to be given to repair transmissions destined to nodes with a large number of missing packets. This is accomplished by initially setting λ_i^q equal to the lower bound for the expected value of the number of transmissions required to repair m_q at node n_i:

$$\lambda_i^q = \begin{cases} 0 & \text{if } m_q \in \delta_i, \\ \dfrac{(\text{BATCH\_SIZE } - |\delta_i|) + 1}{2} & \text{otherwise.} \end{cases} \tag{1}$$

In order to understand (1), assume that n_i has 5 missing packets. n_i requires at least 5 repair transmissions to recover all lost packets. Therefore, for a given missing packet $m_q \notin \delta_i$, the expected number of repair transmissions in the best case scenario is $(5+1)/2 = 3$.

2) Packet Reception: A DiCoR packet ρ carries control information and may carry repair payload. Upon receiving ρ, a node n_i discards it if $\{\rho.\text{epoch}, \rho.\text{file}\}$ does not correspond to the current file or batch. Otherwise, n_i updates its local state. DiCoR first updates α_i and Λ with sender information. It sets $\alpha_i = \alpha_i \cup \{\rho.\text{srcAddress}\}$, since the sender has to be a neighbor. $\rho.urg_q$ represents the most up to date assessment of the urgency of m_q for node ρ.srcAddress. Hence $\lambda^q_{\rho.\text{srcAddress}}$ is set equal to $\rho.urg_q$ $\forall m_q \in \mathcal{M}$[1]. Secondly, if ρ carries a repair payload, it may lead to the repair of a missing packet at n_i. DiCoR, therefore, sets $\delta_i = \delta_i \cup \{\rho.\text{repairPayload}\}$. Thirdly, local state is updated to register the potential reception of ρ at n_i's neighbors. Let η represent the intersection of node sets α_i, and the set representing n_i and the neighbors of ρ.srcAddress listed in ρ. If ρ carries a repair payload, DiCoR sets the urgency of ρ.repairPayload to zero for all nodes in η. Also, if ρ results in repair at a neighboring node, the lower bound for the expected number of transmissions required to repair other missing packets decreases. DiCoR updates Λ as follows:

$$\lambda_j^q = \begin{cases} \lambda_j^q & \text{if } n_j \notin \eta \text{ or } \lambda_j^q = 0, \\ \lambda_j^q - \dfrac{1}{2} & \text{if } n_j \in \eta \ \&\ \lambda_j^q > 0 \ \&\ \rho.\text{pktID} \neq q \\ 0 & \text{if } n_j \in \eta \ \&\ \lambda_j^q > 0 \ \&\ \rho.\text{pktID} = q \end{cases} \tag{2}$$

Let ν_j be the number of packets $m_q \in \mathcal{M}$ for which $\lambda_j^q > 0$. In order to understand (2), consider that a packet gets repaired at $n_j \in \eta$. Then the lower bound for the expected value of the number of transmissions required to repair a given packet at n_j decreases by $1/2$, since $(\nu_j+1)/2 - ((\nu_j-1)+1)/(2) = 1/2$. Note that all nodes in η may not necessarily receive ρ. However, DiCoR optimistically updates local state information to preempt n_i from repairing a packet that has already have been repaired. This leads to a possible scenario where neighbors of a node n_k erroneously assume that a packet missing at n_k has been repaired. This does not pose a problem, since the next DiCoR packet sent by n_k will serve to correct the inconsistent local state information.

3) Scheduling: As discussed in Section IV, the CPR problem is essentially a scheduling problem. The primary function of our distributed protocol is to use local state information to decide the content and assess the utility of sending a DiCoR packet. A prospective DiCoR transmission by n_i has dual utility. It carries control information that guides the assessment of neighboring nodes about packets missing at n_i. Secondly, it may also carry a repair payload that allows missing packets to be repaired at receiving nodes. DiCoR defines two heuristic measures corresponding to each. The *solicit* urgency, Γ_i measures the value of the control information. The *repair* urgency, Θ_i measures the value of the repair payload. DiCoR sets $\Gamma_i = \sum_{m_q \in \mathcal{M}} \gamma_q$, where:

$$\gamma_q = \begin{cases} 0 & \text{if } m_q \in \delta_i, \\ \sum_{n_j \in \alpha_i} \lambda_j^q & \text{otherwise} \end{cases} \tag{3}$$

According to (3), a node has no solicit urgency for a packet present in δ_i. The solicit urgency for a missing packet is

[1] λ_j^q is only initially set equal to the lower bound of the expected value of the number of repair transmissions required to repair all packets at n_j. As will be evident in Section V-E4, updating $\lambda^q_{\rho.\text{srcAddress}}$ according to $\rho.urg_q$ causes λ_j^q to be related to both the heuristic measures delineated in Section IV.

derived from its urgency for n_i and for n_i's neighbors, since they can potentially depend upon n_i to send them the packet. DiCoR sets $\Theta_i = \max_{q \in \mathcal{M}} \theta_q$, where:

$$\theta_q = \begin{cases} 0 & \text{if } m_q \notin \delta_i, \\ \sum_{n_j \in \alpha_i} \lambda_j^q & \text{otherwise} \end{cases} \quad (4)$$

n_i can not repair a packet that is not present in δ_i. According to (4), for packets present in δ_i, the utility of setting one of them to constitute the repair payload is equal to the packet's cumulative urgency over all of n_i's neighbors. Since, the repair payload comprises a single packet, Θ_i is set to the repair urgency of the packet that yields the maximum repair utility. Basically it is a measure of the good that n_i can do by sending a certain repair payload. DiCoR sets the Send_Timer proportional to the value of the *solicit* and *repair* urgency:

$$\text{Send\_Timer} = \begin{cases} W_s + W_r & \text{if } \Gamma_i > 0 \text{ or } \Theta_i > 0, \\ \infty & \text{otherwise.} \end{cases} \quad (5)$$

$$W_s = \begin{cases} \text{SOLICIT\_WAIT}/\Gamma_i & \text{if } \Gamma_i > 0, \\ \text{SOLICIT\_WAIT} & \text{otherwise.} \end{cases} \quad (6)$$

$$W_r = \begin{cases} \text{REPAIR\_WAIT}/\Theta_i & \text{if } \Theta_i > 0, \\ \text{REPAIR\_WAIT} & \text{otherwise.} \end{cases} \quad (7)$$

(5) shows how the duration of time n_i waits before sending a DiCoR packet is proportional to the repair and solicit urgency. Therefore, nodes with higher repair and solicit urgencies have a greater chance of capturing the channel. If both the repair and solicit urgency are zero, n_i does not schedule a transmission.

Once we schedule a DiCoR transmission, we drop any prior DiCoR packets in the outbound queue at the network layer. Such a packet is potentially based on stale information. As a consequence, n_i has at most one pending transmission. This is done to preempt head of line blocking wherein a DiCoR packet is blocked behind *stale* DiCoR packets.

4) Send Packet: When the Send_Timer expires at n_i, n_i creates a DiCoR packet ρ and forwards it to the network layer. The contents of the DiCoR packet follow from the local state information present at n_i:

$$\rho.\text{urg}_k = \begin{cases} \lceil \sum_{n_j \in \alpha_i} \lambda_j^k \rceil & \text{if } k \notin \delta_i, \\ 0 & \text{otherwise.} \end{cases} \quad (8)$$

(8) sets the advertised urgency $\rho.\text{urg}_k$ of a packet m_q equal to the cumulative urgency estimates of m_q for n_i's neighbors. Therefore, if a chain of nodes solicit a packet, its urgency increases, and consequently DiCoR will expedite its repair. This is in keeping with the other heuristic delineated in Section IV that greater priority ought to be given to a packet that has a greater number of hops to travel.

If $\Theta_i > 0$, a repair payload is selected by setting $\rho.\text{pktID} = \arg\max_{k|m_k \in \delta_i} \theta_k$. If ρ carries a repair payload, local state is optimistically updated to register the dispatch of a repair packet to n_i's neighbors. This update is analogous to what is done upon packet reception, described by (2). As mentioned in the case of packet reception, all nodes in α_i may not necessarily receive ρ. The update represents a heuristic to keep local state current.

VI. Simulation Study

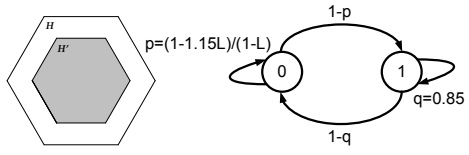
A. Basic Simulation Setup

We begin by describing our basic simulation setup. We use regular hexagons to represent two types of cells: *macro-cells* and *micro-cells*. Macro-cells have a spacing of $1000m$ between the base stations of adjacent cells. The corresponding spacing for micro-cells is $360m$. Nodes representing MBMS subscribers are *uniformly* distributed across a cell. The typical bit rate available to MBMS subscribers is 128 kbps [6]. However, MBMS can offer a maximum bit rate of 384 kbps [1]. We, therefore, set MBMS_RATE equal to 384 kbps. We set PAYLOAD_SIZE equal to 1000 bytes. As a consequence, REPAIR_EPOCH is equal to BATCH_SIZE $\times$ 20.83 ms. A logical maximum threshold for the BATCH_SIZE is determined by the capacity of the playout buffer at distributed peers. The buffer should be able to hold $2 \times$ BATCH_SIZE number of packets. This includes the batch being repaired during the current repair epoch and the batch being accumulated for repair. Assuming we can buffer content up to a maximum of 15 secs [21], the range for BATCH_SIZE lies between 1 and 360.

We differentiate CPR from the radio broadcast problem in Section II on the basis of the initial distribution of messages across peers. It follows that the choice of the particular *loss model* used to characterize MBMS broadcasting losses is significant. We use $\mathcal{L}$ to denote the average packet loss rate for a given batch across all nodes in a cell. We consider four different models for the distribution of packet losses:

• The *i.i.d.* (IID) model assumes losses to be independent and identically distributed across different nodes and across time. A node fails to receive a packet with probability $\mathcal{L}$.

• The *Spatial Locality* (SL) model accounts for signal attenuation as it propagates through space. We consider a simple model wherein we define two regions within a cell. The regions are defined by a regular hexagon ($\mathcal{H}'$), that is concentric with our cell ($\mathcal{H}$). The length of a side of $\mathcal{H}'$ is set to $a/\sqrt{2}$, where a is the length of the side of $\mathcal{H}$. Consequently, the shaded and unshaded regions defined by $\mathcal{H}$ and $\mathcal{H}'$ in Fig. 4a have equal area. A node fails to receive a packet with probability $\mathcal{L}_i = 0.75\mathcal{L}$ if it resides in the shaded region, and fails to receive a packet with probability $\mathcal{L}_o = 1.25\mathcal{L}$ otherwise. Since the nodes are uniformly distributed, the overall expected packet loss rate is $\mathcal{L}$.



(a) SL Model (b) TL Model

Fig. 4: Loss Models

• The *Temporal Locality* (TL) model accounts for burstiness in packet losses due to transient channel conditions. We model this burstiness using the well known Gilbert model [22]. The first-order markov chain that characterizes our Gilbert model is shown in Fig. 4b [22]. State 0 represents that the last packet

was received correctly and state 1 represents otherwise. p is set such that the overall expected packet loss rate is $\mathcal{L}$.

• The *Spatial & Temporal Locality* (S&TL) model combines the SL and TL models. Packet reception is characterized by the markov chain shown in Fig. 4b, with $p = \frac{1-1.15\mathcal{L}_i}{1-\mathcal{L}_i}$ and $p = \frac{1-1.15\mathcal{L}_o}{1-\mathcal{L}_o}$, for nodes residing in the shaded region and unshaded regions of Fig. 4a, respectively.

The simulation platform we use for DiCoR is the QualNet simulator [11], which is the successor of the previous GloMoSim simulation library [23]. The configuration parameters of the protocol stack in our simulations use the default values. We employ IEEE 802.11 MAC DCF with RTS and CTS disabled, and our channel propagation model is the two-ray ground reflection model [24]. We adopt 802.11g as our modulation scheme. All DiCoR transmissions are broadcasts and do not employ any ARQ mechanism. We define a couple of metrics to measure the efficacy of our protocol. The *repair latency* of a node is defined as the lapse of time from the start of the repair epoch for all packets missing at the node to be repaired. We consider a node to have *converged* if all entries in Λ stabilize to zero. Stabilizing to zero means that the entries remain zero for the remainder of the repair epoch. Hence, a converged node implies that all missing packets at the node are repaired and the node has stopped receiving solicitations from its neighbors. In other words, a converged node will not solicit for or repair any packets. The *converge latency* of a node is defined as the lapse of time from the start of the repair epoch for the node to converge. The repair and converge latency for a batch is the lapse of time for all nodes to get repaired and converged, respectively.

B. Parameter Tuning

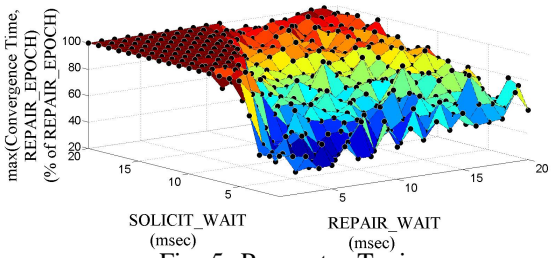
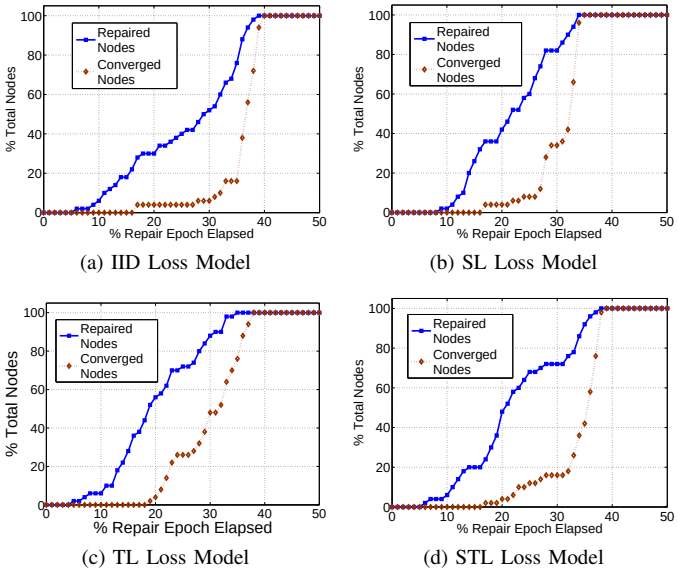


Fig. 5: Parameter Tuning

The motivation behind our first set of experiments is to find appropriate values for the SOLICIT_WAIT and REPAIR_WAIT parameters. We consider a network of 50 nodes uniformly distributed in a macro-cell. Let $B = \{50, 75, 100, 125\}$, $M = \{\text{IID, SL, TL, ST\&L}\}$, and $L = \{0.2, 0.3, ..., 0.8\}$ represent the set of values for the BATCH_SIZE, loss model, and the average loss rate $\mathcal{L}$, respectively. We run 400 experiments by varying both SOLICIT_WAIT and REPAIR_WAIT from 1 to $20msec$ for every tuple in the Cartesian product of B, M, and L . Fig. 5 presents the results averaged over all our simulation runs for each element in $B \times M \times L$. The vertical axis plots the maximum of the convergence latency and the REPAIR_EPOCH, for different values of SOLICIT_WAIT and REPAIR_WAIT. A lower value represents faster completion of the DiCoR repair mechanism. We see that the fastest convergence time corresponds to SOLICIT_WAIT $= 2$ msec and REPAIR_WAIT $= 3$ msec. These values yield desirable results for a wide range of scenarios. We found them to work well in our micro-cell setting as well. We, therefore, tune SOLICIT_WAIT and REPAIR_WAIT to 2 msec and 3 msec, respectively, for the rest of our simulation study.

C. Efficacy of DiCoR

We now look at the repair and converge latency of DiCoR. We consider a network of 50 nodes uniformly distributed in a macro-cell. We expect the typical number of DiCoR peers in the same cell subscribing to the same content to be much smaller than that. We will show in Section VI-D that the performance of DiCoR is adversely affected as the network density increases. Hence, our choice of 50 nodes is a conservative estimate designed to provide a worst-case analysis with respect to DiCoR performance for more typical scenarios. We set the *raw* 802.11g data rate to be 36 Mbps, $\mathcal{L} = 0.4$, and BATCH_SIZE $= 100$. Fig. 6 shows the percentage of repaired and converged nodes at different stages of the repair epoch. The results are averaged over 100 batches.

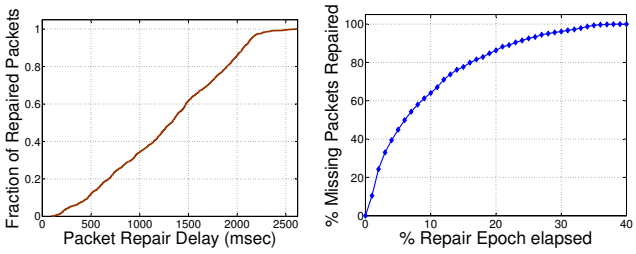


(a) IID Loss Model (b) SL Loss Model

(c) TL Loss Model (d) STL Loss Model

Fig. 6: DiCoR Performance for different Loss Models

The key result is that 100% of the missing packets get repaired, and DiCoR peers converge well before the expiration of the repair epoch. Fig. 6 shows how the repair process is completed within 40% of the repair epoch for all four loss models. We observe similar timely repair for different 802.11g data rates and different number of nodes in both the macro-cell and micro-cell settings. Specifically, any combination of these parameters that results in an average node degree between 2 and 15 yields a batch convergence latency between 25% and 65% of the repair epoch. The detailed results are omitted to conserve space. We did not observe any significant difference between using the four loss models. We, therefore, include results only for the STL loss model in the rest of this section. Fig. 7 and Fig. 8 correspond to the same set of experiments as Fig. 6d. Fig. 7a shows the CDF for the delay between the



(a) Packet Repair Delay (b) Packet Repair Progress

Fig. 7: Packet Repair Prognosis

scheduled delivery time of a missing packet and its eventual repair time at a node. Note that a packet must at least wait for its associated repair epoch to start, and then wait for the time it takes for the packet to be repaired. Hence the maximum repair delay for a packet is $2 \times$ REPAIR_EPOCH. We observed the average repair delay to be $0.61 \times$ REPAIR_EPOCH, and the maximum repair delay to be $1.25 \times$ REPAIR_EPOCH. Given our BATCH_SIZE of 100, this corresponds to an average and maximum repair delay of 1.3 and 2.6 seconds, respectively. Fig. 7b shows that the percentage of missing packets repaired increases at a decreasing rate. This may be attributed to the fact that since many packets are missing in the early stages of a repair epoch, a single DiCoR packet carrying a repair payload tends to repair missing packets at multiple nodes.

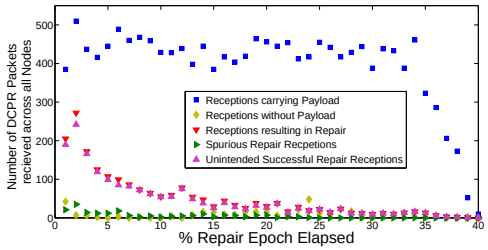
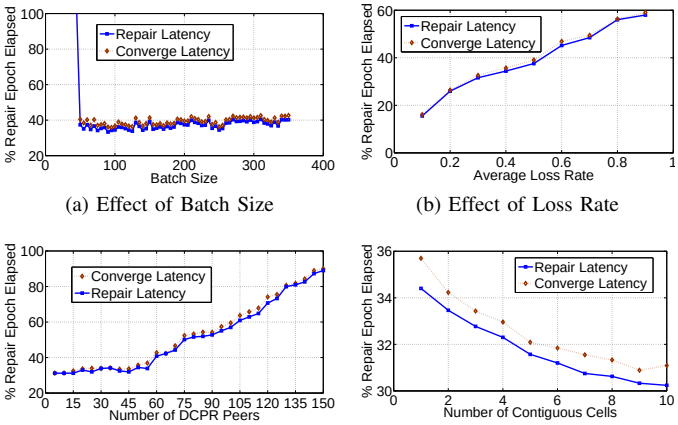


Fig. 8: Packet Reception Prognosis

Fig. 8 showcases the profile of received packets across nodes. We divide the repair epoch into 100 equally sized sub-epochs and report the average number of packets for each category received at a node. Fig. 8 shows that DiCoR packets are received at a steady rate throughout the repair epoch until nodes start to converge. This suggests that DiCoR transmissions do not swamp the network such that no useful work is done. The vast majority of DiCoR receptions not carrying a repair payload are restricted to the start of the repair epoch, when DiCoR is bootstrapping. A significant amount of receptions leading to successful repair are *unintended*, meaning that the node was not an intended recipient of the repair payload. This stems from the broadcast nature of DiCoR wherein once a packet is repaired at a node that had solicited for it, the packet may also get repaired at other nodes in the neighborhood that did not solicit for it. This is also one of the reasons why DiCoR performance using the SL model is comparable to other models with similar $\mathcal{L}$. The potential for some nodes to act as bottlenecks is offset by losses being co-located in space. Hence, a single DiCoR repair transmission replaces missing packets at a large number of nodes. A reception is *spurious* if the receiving node is an intended recipient of a repair payload that is no longer required. This stems from inconsistent local state at the sending node. It is interesting to note that the number of such receptions is very low, which is indicative of the accuracy of the control information propagated and the heuristics employed for updating the local state.

D. Sensitivity Analysis

We now look at how different DiCoR parameters affect DiCoR performance. Wherever applicable, we use our default setting with 50 nodes uniformly distributed in a macro-cell, and the *raw* 802.11g data rate equal to 36 Mbps, $\mathcal{L} = 0.4$, and BATCH_SIZE $= 100$. Results are averaged over 100 batches. As mentioned before, DiCoR performance for these settings is similar in nature to other settings and hence, we limit the presented results to our default settings. Fig. 9a shows the DiCoR repair and converge latency as a function of BATCH_SIZE. We vary BATCH_SIZE in increments of 10. We observe a certain threshold like behavior in that DiCoR does not converge for BATCH_SIZE $\leq$ 40 and converges once BATCH_SIZE $\geq$ 50. The converge latency resides within $33\% - 47\%$ of the repair epoch for $50 \leq$ BATCH_SIZE ≤ 350, which covers reasonable values of BATCH_SIZE for our problem. We also observed that the average repair delay for a packet as a function of REPAIR_EPOCH, is similar for different values of BATCH_SIZE. However, since REPAIR_EPOCH increases with BATCH_SIZE, repair delays are optimized (and the required size of the playout buffer minimized) if we choose the lower BATCH_SIZE from a set of values with similar converge latencies.



(a) Effect of Batch Size (b) Effect of Loss Rate

(c) Effect of Network Size (d) Contiguous Cells

Fig. 9: DiCoR Sensitivity Analysis

Fig. 9b shows that DiCoR performance scales well with $\mathcal{L}$. For these experiments, we assume that at least one copy of a packet is present in the network. Section VII indicates how this assumption can be relaxed for future work. One reason for the observed scalability is that as $\mathcal{L}$ increases, a single DiCoR repair transmission replaces missing packets at a large number of nodes. We do not see the same scalability as the node density increases. Fig. 9c shows that beyond a network size of 55, DiCoR convergence latency deteriorates rapidly. This can be attributed to increased channel contention.

E. Resilience to Dynamic Behavior

We also simulated various scenarios with node mobility and nodes subscribing/unsubscribing with the MBMS/DiCoR service. Our experiments showed that the repair and converge latencies for such dynamic networks are similar to those for our experiments with static networks. We omit the detailed results of the dynamic scenarios since the reason of DiCoR's resilience to dynamic behavior is evident. DiCoR does not assume any a priori knowledge of the network topology and does not preserve topology information across repair epochs. Assuming a BATCH_SIZE of 100, the REPAIR_EPOCH is approximately equal to 2.1 secs. The topology does not evolve significantly at such a fine time-scale. Dynamic subscription of nodes to the MBMS/DiCoR service poses a slight problem. It is evident that nodes joining CPR at the later stages of a repair epoch may not get repaired or converge. Furthermore, we observed that in such cases, DiCoR transmissions scheduled in one repair epoch spill-over to the next repair epoch, thus increasing its repair/converge latency. This is circumvented by a simple rule that requires a joining node to wait for the ongoing repair epoch to complete and participate in DiCoR from the start of the next repair epoch.

We also studied scenarios where base stations in multiple cells simultaneously broadcast identical MBMS content. DiCoR peers can, therefore, also cooperate with peers in adjacent cells to repair broadcast losses (Fig. 1). Fig. 9d shows how this actually leads to a slight improvement in the repair/converge latency. Our network is grown radially from a single cell and results are averaged over 25 batches.

VII. Conclusions

Our simulation study sets MBMS_RATE equal to the maximum MBMS bit rate of 384 kbps. For more typical values of MBMS_RATE, DiCoR stands to yield better repair and converge latencies relative to the REPAIR_EPOCH. Even with high loss rates, DiCoR successfully repairs all lost packets well before the playout deadline. Furthermore, DiCoR promptly converges and ceases sending DiCoR packets soon after neighboring peers are repaired. The major contribution of this work is presenting DiCoR as a viable solution for cooperative out-of-band peer-to-peer repair. Our current scheme requires at least a single copy of a packet to be present within each connected network component for DiCoR to converge. This can be circumvented by investigating a hybrid solution between CPR and explicit retransmissions. For instance, the 3G base station can be explicitly requested to repair packets for which the solicit urgency crosses a critical threshold. DiCoR also presents a base framework for future enhancements. Potential directions include prioritizing repair of packets based on content or delivery deadlines, and incorporating mechanisms to make DiCoR resilient to malicious or selfish peer behavior. In conclusion, DiCoR presents a viable solution and a promising out-of-band repair framework for eagerly anticipated multimedia content delivery services such as mobile-TV.

References

[1] *Technical Specification Group Services and System Aspects; Multimedia Broadcast/Multicast Service (MBMS) user services; stage 1 (3GPP TS.26.246 version 6.3.0 Release 6)*, March 2006.

[2] H. Jenkac, T. Stockhammer, G. Liebl, and W. Xu, "Retransmission strategies for MBMS over GERAN," in *IEEE WCNC 2005*, New Orleans, LA, USA, March 2005.

[3] H. Jenkac, T. Stockhammer, G. Liebl, and G. Munich, "H. 264/AVC Video Transmission over MBMS in GERAN," *Proc. MMSP 2004*.

[4] T. Stockhammer, W. Xu, T. Gasiba, M. Luby, and M. Watson, "Raptor codes for reliable download delivery in wireless broadcast systems," in *IEEE CCNC*, Las Vegas, NV, USA, January 2006.

[5] J. Crowcroft and K. Paliwoda, "A multicast transport protocol," in *ACM SIGCOMM*, August 1988.

[6] J. Afzal, T. Stockhammer, T. Gasiba, and W. Xu, "Video Streaming over MBMS: A System Design Approach," *JOURNAL OF MULTIMEDIA*, vol. 1, no. 5, pp. 25, 2006.

[7] P. Sharma, S.-J. Lee, J. Brassil, and K. Shin, "Distributed communication paradigm for wireless community networks," in *IEEE International Conference on Communications*, Seoul, Korea, May 2005.

[8] Gene Cheung, Puneet Sharma, and Sung-Ju Lee, "Smart media striping over multiple burst-loss channels," in *IEEE Journal of Selected Topics in Signal Processing*, August 2007, vol. 1, no.2.

[9] K. Jain, J. Padhye, V. Padmanabhan, and L. Qiu, "Impact of interference on multi-hop wireless network performance," in *Proceedings of the ACM International Conference on Mobile Computing and Networking (MobiCom 2003)*, San Diego, CA, September 2003.

[10] S. Raza, D. Li, C.-N. Chuah, and G. Cheung, "Cooperative peer-to-peer repair for wireless multimedia broadcast," in *IEEE International Conference on Multimedia and Expo*, Beijing, China, July 2007.

[11] S.N. Technologies, "QualNet," *Available HTTP: http://www. scalable-networks. com*.

[12] R. Xu, A. Myers, H. Zhang, and R. Yavatkar, "Resilient multicast support for continuous-media applications," in *International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, St. Louis, MI, May 1997.

[13] S. Floyd, V. Jacobson, S. Mccanne, and L. Zhang, "A reliable multicast framework for light-weight sessions and application level framing," in *IEEE/ACM Trans. Networking*, December 1997, vol. 5, no.6.

[14] S. Deering, "Host extensions for IP multicasting," August 1989, IETF RFC 1112.

[15] S. Casner, "Frequently asked questions (FAQ) on the multicast backbone," May 1993, http://www-mice.cs.ucl.ac.uk/multimedia/projects/mice/faq.html.

[16] I. Chlamtac and S. Kutten, "On broadcasting in radio networks–problem analysis and protocol design," in *IEEE Transactions on Cummunications*, December 1985, vol. 33, no.12.

[17] R. Gandhi, S. Parthasarathy, and A. Mishra, "Minimizing braodcast latency and redundancy in ad hoc networks," in *MobiHoc*, Annapolis, MD, June 2003.

[18] S. C.-H. Huang, P.-J. Wan, X. Jia, H. Du, and W. Shang, "Minimum-latency broadcast scheduling in wireless ad hoc networks," in *IEEE INFOCOM 2007*, Anchorage, AL, May 2007.

[19] Z. Chen, C. Qiao, J. Xu, and T. Lee, "A constant approximation algorithm for inteference aware broadcast in wireless networks," in *IEEE INFOCOM 2007*, Anchorage, AL, May 2007.

[20] Gene Cheung Saqib Raza and Chen-Nee Chuah, "On the complexity of cooperative peer-to-peer repair problem with collisions," Tech. Rep. ECE-CE-2007-2, Computer Engineering Research Laboratory, University of California, Davis, 2007, http://www.ece.ucdavis.edu/cerl/techreports/2007-2/.

[21] J. G. Apostolopoulos, W. Tan, and S. J. Wee, "Video streaming: Concepts, algorithms, and systems," *Handbook of Video Databases*, March 2003.

[22] A. Konrad, B.Y. Zhao, A.D. Joseph, and R. Ludwig, "A Markov-Based Channel Model Algorithm for Wireless Networks," *Wireless Networks*, vol. 9, no. 3, pp. 189–199, 2003.

[23] X. Zeng, R. Bagrodia, and M. Gerla, "GloMoSim: a library for parallel simulation of large-scale wireless networks," *Workshop on Parallel and Distributed Simulation*, pp. 154–161, 1998.

[24] T.S. Rappaport, *Wireless Communications: Principles and Practice*, IEEE Press Piscataway, NJ, USA, 1996.