Cooperative Peer-to-Peer Repair for Wireless Multimedia Broadcast

Saqib Raza* Danjue Li* Chen-Nee Chuah* Gene Cheung†
* ECE Department, UC Davis † HP Laboratories, Japan

***Abstract*— This paper explores how to leverage IEEE802.11-based cooperative peer-to-peer repair (CPR) to enhance the reliability of wireless multimedia broadcasting. We first formulate the CPR problem and present an algorithm that assumes global state information to optimally schedule CPR transmissions. Based on insights gained from the optimal algorithm, we propose a fully distributed CPR (DCPR) protocol. Simulation results demonstrate that the DCPR protocol can effectively enhance the reliability of wireless broadcast services with a repair latency comparable to that of optimal scheduling.**

I. Introduction

Multimedia Broadcast/Multicast Services (MBMS) [1] for 3rd generation (3G) cellular networks employ broadcast/multicast transport to serve rich multimedia content to large user groups simultaneously. Ensuring efficient error-free content delivery in such a context is a challenge. The feedback implosion problem [2] excludes the use of explicit feedback from receivers to confirm successful broadcast transmission. Instead of using retransmissions [3], solutions to improve MBMS reliability are mainly based on application-layer Forward Error Correction (FEC) [4]. However, FEC falls short of fixing all errors for heterogeneous users. Using more complex FEC incurs the overhead of consuming extra 3G bandwidth. Hence, it is desirable to handle the residual errors through a supplementary mechanism. Towards this end, we note that losses among MBMS subscribers, which receive the same content, are often uncorrelated. Therefore, a node can procure lost packets from other nodes in the network which have those packets.

In this paper, we propose a *cooperative peer-to-peer repair (CPR)* scheme, which leverages IEEE 802.11 peer-to-peer connections to achieve out-of-band repair of 3G broadcasting losses. As illustrated in Figure 1, we consider a two-phase streaming procedure: in the first phase, information is pushed into mobile devices via MBMS; and then in the second phase, mobile devices perform CPR to repair packets lost during broadcasting. Such a system necessitates that each mobile device has multiple wireless interfaces, which is increasingly common for modern wireless devices [5]. We assume a device can simultaneously connect to a wireless wide area network, e.g., 3G network, and to a wireless local area network (WLAN), e.g., IEEE 802.11 based ad hoc peer-to-peer network.

Nevertheless, within a WLAN, different nodes in the same neighborhood cannot transmit at the same time due to interference [6]. Hence, the CPR problem is essentially a repair scheduling problem, i.e., deciding which missing packet(s) to repair (if any) and how to schedule the corresponding repair transmissions. In the ideal case, where participating peers are aware of the network topology and the availability of packets at each peer, nodes can independently compute an optimal repair schedule. However, this may not be feasible due to two challenges: (1) the communication overhead incurred in collecting information about the topology and packet availability; and (2) the high computational complexity of obtaining the optimal schedule.

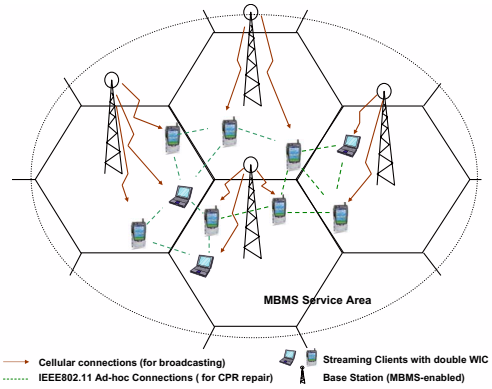

Fig. 1. Cooperative Peer-to-Peer Repair in 3G Network

It follows that a practical CPR protocol requires an efficient mechanism to propagate the network topology and packet availability information to every peer. We need to find a good tradeoff between the overhead incurred and the accuracy of the information. With this mechanism in place, we need a distributed algorithm that allows a node to *independently* schedule its transmissions based upon information available locally. Note that these two components are interrelated.

A design choice that we make is to wait for a *batch* of packets to be injected into the network via MBMS before initiating the repair process. This allows the cost associated with information distribution to be amortized over multiple packets. Such a batch-based approach might increase the delay of repairing a packet. To support multimedia broadcasting service, we need to ensure that the total latency due to batching and CPR repair has to be shorter than the delay tolerance (e.g., playout delay) of a packet.

II. Problem Definition

Consider an ad-hoc network formed by N wireless nodes, representing CPR-enabled subscribers that are receiving the same content via MBMS. Let n_i $(1 \leq i \leq N)$ denote a

wireless node, and d_{ij} denote the distance between n_i and n_j. For each n_i, we denote its radio communication range and interference range using r_i and r'_i ($r'_i \geq r_i$), respectively. Let $\mathcal{V}$ represent the set of wireless nodes. We further define two edge sets $\mathcal{E}$ and $\mathcal{E}'$:

$$\mathcal{E} = \{(i,j) | n_i \in V, n_j \in V, d_{ij} \leq r_i\} \quad (1)$$

$$\mathcal{E}' = \{(i,j) | n_i \in V, n_j \in V, d_{ij} \leq r'_i\} \quad (2)$$

$(i,j) \in \mathcal{E}$ implies that n_j is within the radio communication range of n_i. Similarly, $(i,j) \in \mathcal{E}'$ implies that n_j is within the interference range of n_i.

We assume that each node only uses a single 802.11-based channel to reach its peer. A transmission from n_i to n_j is successful *iff* it satisfies three conditions: (1) $(i,j) \in \mathcal{E}$ and $(j,i) \in \mathcal{E}$, (2) all nodes $\{n_k \in V | k \neq i, (k,j) \in \mathcal{E}'\}$ are not transmitting, and (3) all nodes $\{n_k \in V | k \neq i, (k,i) \in \mathcal{E}'\}$ are not transmitting. The first condition stipulates that n_j and node n_i must be within the transmission range of each other. The second and third conditions stipulate that no interference occurs during the transmission.

We now formulate our batch-based CPR repair scheduling problem. Suppose a batch of packets $\mathcal{K}$ is delivered via MBMS. Let K denote the number of packets in the batch, i.e., $\mathcal{K} = \{p_1, p_2, p_3, ..., p_K\}, K \geq 1$. Due to transmission errors, a node n_i receives a subset $R_i \subseteq \mathcal{K}$ of packets. Our goal is to find a repair schedule so that at the end of a set of peer-to-peer transmissions, each node has all the packets in $\mathcal{K}$. A $K \times N$ *Batch Map Matrix* (BMM) records the availability of each packet p_k on each node, i.e., $BMM_{ki} = 1$ if $p_k \in R_i$, and 0, otherwise. To denote the updated BMM after w transmission rounds, we add a superscript w to BMM. BMM^0 denotes the initial BMM before repairing any loss while BMM^E denotes the BMM after eventually reaching the end of the repair process, i.e., $BMM^E_{ki} = 1, \ \forall \ (k,i) | p_k \in \mathcal{K}, n_i \in \mathcal{V}$.

Let t_w be the transmission policy at the w^{th} transmission round. t_w is an $N \times 1$ matrix representing a set of node assignment decisions, i.e., $t_w = \{S^w_i\}$, where $S^w_i = k$ if n_i is selected to send packet k at transmission round w, and 0, otherwise. The solution to the batch-CPR problem is, therefore, a series of transmission policies $T_Q = (t_1, t_2, ..., t_Q)$, which can accomplish the transition $BMM^0 \stackrel{t_1}{\Rightarrow} BMM^1 \stackrel{t_2}{\Rightarrow} ... \stackrel{t_Q}{\Rightarrow} BMM^Q = BMM^E$ in Q transmission rounds. Our objective is to find a T_Q that minimizes Q. Let $\hat{Q}$ be the optimal value of Q. Mathematically, we can model the batch-CPR problem as:

$$\min_{T_Q} Q \quad (3)$$

Subject to:

$$S^w_i = k \Rightarrow BMM^{w-1}_{ki} = 1, \ \ \forall i \in \mathcal{V} \quad (4)$$

$$S^w_j > 0 \Rightarrow S^w_i = 0, \ \ \forall (j,i) \in \mathcal{E}' \quad (5)$$

$$BMM^w_{kj} = BMM^{w-1}_{kj}, \quad \forall j \in \mathcal{V} | \nexists ((i,j) \in \mathcal{E} \text{ and } S^w_i = k) \quad (6)$$

$$BMM^w_{kj} = BMM^{w-1}_{kj}, \quad \forall j \in \mathcal{V} | \exists ((h,j),(i,j) \in \mathcal{E}' \text{ and } h \neq i \text{ and } S^w_h > 0 \text{ and } S^w_i > 0) \quad (7)$$

$$BMM^Q_{k,i} = 1, \ \ \forall i \in \mathcal{V} \quad (8)$$

$$BMM^w_{kj} \in \{0,1\}, \ \ \forall j \in \mathcal{V} \quad (9)$$

$$Q \in \mathbf{Z}^+ \quad (10)$$

where Eq. (4) and (5) represent sender constraints while Eq. (6) and (7) represent receiver constraints. Specifically, Eq. (4) requires a node to be in possession of a packet that it broadcasts. Eq. (5) ascertains that if the sender is within the interference range of another node, then the other node does not transmit. Eq. (6) requires that in order to receive a packet, a node has to be within the transmission range of a node that transmits the packet. Eq. (7) stipulates that a node cannot correctly receive a packet if it is within the interference range of multiple senders. Eq. (8) requires that all losses be repaired at stage Q, while Eq. (9) and (10) are integrality constraints.

III. Optimal Solution

In this section, we present an algorithm to compute the optimal transmission policy that yields $\hat{Q}$.

We first construct a *Batch Map Tree (BMT)* initialized with the singleton element BMM^0. We then simultaneously explore and grow the BMT, until we encounter BMM^E. The BMT is explored in a breadth-first-search like manner, looking at all BMMs at level w before exploring BMMs at level $w+1$. Visiting a node BMM^w in the BMT comprises checking whether $BMM^w = BMM^E$ i.e., $BMM^w_{ki} = 1, \ \forall \ (k,i) : p_k \in \mathcal{K}, n_i \in \mathcal{N}$. If so, we terminate the process. If not, we grow the tree to include all *descendants* of BMM^w. BMM^{w+1} is a descendant of BMM^w, if and only if there exists a valid transmission policy t_w that can accomplish the transition $BMM^w \stackrel{t_w}{\Rightarrow} BMM^{w+1}$. By defining a valid transmission policy as one that satisfies the constraints in (4)-(10), we ensure that our algorithm encounters BMM^E, at level $\hat{Q}$ of the BMT. We can then retrace to the root of the tree i.e., BMM^0 to obtain the series of transitions.

Unfortunately, the algorithm has exponential complexity. The size of the BMT grows exponentially as a function of N and K. Discovering descendants of a BMM also has complexity that is exponential in N and K. To enhance the efficiency of our algorithm, we use a number of optimizations to condense the search space. The details are omitted to conserve space. However, even with the optimizations, the problem is intractable except for small problem sizes. For a slight variant of the interference constraints, [7] shows that even the simple case with $K = 1$ is NP-hard. Our motivation for studying the optimal solution is to gain insights into solving the CPR problem and use them as guidelines towards designing heuristics for a feasible CPR protocol.

We proceed to compute the best and worst case bounds. Let g_{ki} denote the hop-count distance from n_i to the nearest peer which has p_k. The number of transmission rounds required to repair all missing packets at n_i is bounded by $g_i = \max_{k:p_k \in \mathcal{K}} g_{ki}$. If the total number of missing packets at n_i is $h_i = (K - \sum_{k=0}^{K} BMM^0_{ki})$, we require at least $\max(g_i, h_i)$ transmission rounds to repair all packet losses at n_i. Therefore, repairing all the losses in the entire network requires at least $\hat{b} = \max_{i:n_i \in \mathcal{V}} \max(g_i, h_i)$ transmission rounds, i.e., $\hat{b}$ is a best case bound for $\hat{Q}$.

We also determine a *loose* worst case bound for $\hat{Q}$ as: $\hat{w} = K \times N - \sum_{k=0}^{K} \sum_{i=0}^{N} BMM^0_{ki}$. This stems from the observation that there always exists a a transition series that repairs one missing packet at a time, and hence $\hat{Q}$ can not be greater than the total number of zeros in BMM^0.

We simulate the batch CPR repair scheduling problem and compare the optimal solution $\hat{Q}$ to $\hat{b}$ and $\hat{w}$. Our nodes are uniformly distributed over a $700m \times 700m$ area, with $r_i = 150m$ and $r'_i = 180m$, $\forall n_i \in \mathcal{N}$. Every node has a uniform probability $P_r = 0.7$ of correctly receiving a packet. We only consider initial BMMs such that there is at least one copy of each packet in the network. Results are averaged over 50 simulation runs with random topologies and initial batch maps.

Figure 2 plots $\hat{Q}$ against the best and worst case bounds for batch size $K = 1$. A very interesting observation is that $\hat{Q}$ closely tracks the best-case bound $\hat{b}$. Figure 3 plots $\hat{Q}$ for $K \geq 1$. We again observe that $\hat{Q}$ closely tracks the function $\hat{b} \times ln(K)$. The reason why $\hat{Q}$ is close to $ln(K) \times \hat{b}$ rather than $K \times \hat{b}$, can be explained by observing that packet repair is not a purely sequential process. We can exploit some parallelism for batch CPR, e.g., available links not used for repairing a given packet can be used to repair some other packet.

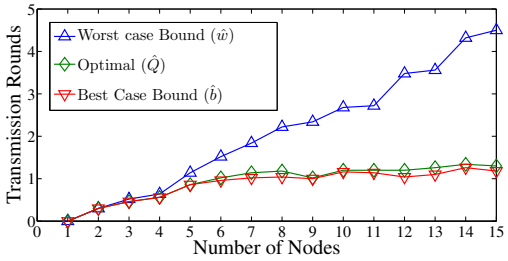

Fig. 2. Performance of Single-Packet CPR

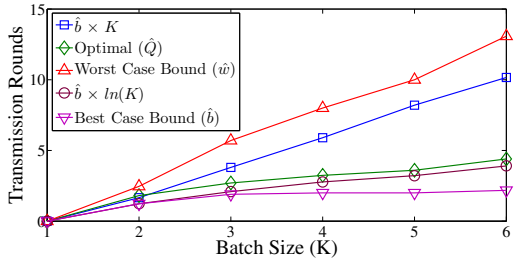

Fig. 3. Performance of Batch CPR for a Six Node Network

IV. Distributed CPR Protocol

This section investigates how to design an efficient distributed CPR protocol. In the context of the BRS problem, we define a *repair epoch* as the duration within which broadcast losses associated with a batch need to be repaired. The repair epoch for a batch is triggered as soon as the last packet for that batch is broadcast into the network. Since mobile subscribers/CPR participants may join and leave the system at anytime, we assume that all peers have no information or assumptions about packets missing at other peers or about the initial topology. The only initial information available is a list of packets that are missing at the node itself.

Recall from Section I that distributed CPR requires a mechanism to propagate information about network topology and packet availability with the desired tradeoff between overhead incurred and the accuracy of the information. A design choice we make for lowering this overhead is for control information propagation and packet repair to occur *simultaneously*, instead of treating them as two discrete phases.

Therefore, a node must make the following decision at any point in time: a) if the node were to transmit, what should it transmit. Since we simultaneously propagate information as well as repair packets, therefore, the question is what control information must the node propagate, and which packet should the node repair; b) should the node transmit or should it defer transmitting the packet.

A concept associated with both these decisions is the notion of *urgency*. We need to estimate how urgent it is to propagate the control/state information. We also need to estimate how urgent is to repair any given packet. We refer to the former as *solicit urgency* and to the latter as *repair urgency*. To compute the solicit and the repair urgency, every node n_i maintains the following *local* state information:

- *Known Neighbors List*, $\mathcal{J}$; gets populated as n_i receives transmissions from its neighbors.
- *Urgency Assessment Matrix*, $\mathcal{A}$; where its entries $\mathcal{A}_{jk}$ represent n_i's estimate of the urgency of p_k for node $n_j \in \mathcal{J}$. Since initially $\mathcal{J} = \{n_i\}$, $\mathcal{A}_{ik}$ is set to 1 if n_i did not receive the broadcast of p_k, and is 0 otherwise.
- *Urgency Map*, $\mathcal{U}$; where its entries $\mathcal{U}_k$ are the overall estimated urgency of packet p_k for n_i, defined as $\sum_{j:n_j \in \mathcal{J}} \mathcal{A}_{jk} \times \mathcal{A}_{ik}$.

Since a packet will be repaired only after a repair is solicited, the solicit urgency for n_i is given by $\sum_{p_k \in \mathcal{K}} \mathcal{U}_k$.

To choose the packet to repair, we use our observed relationship between $\hat{Q}$ and $\hat{b}$ from Section III. We infer that one limiting bottleneck in the batch CPR problem are packets that have a large number of hops to travel to get from a node that has the packet to a node that requires it. The other limiting bottleneck are nodes that have a large number of packets to repair. We, therefore, propose the following dual heuristics:

- Everything else being equal, CPR should repair a missing packet that has *the furthest to travel* prior to repairing a missing packet with a lesser number of hops to traverse. In other words, packet p_e should be repaired before p_f if $\max_{i:n_i \in \mathcal{V}}(g_{ei}) \geq \max_{i:n_i \in \mathcal{V}}(g_{fi})$.
- Everything else being equal, CPR should repair missing packets at nodes with *a larger number of missing packets* prior to repairing packets at other nodes. In other words n_i gets priority over n_j if $h_i > h_j$.

Therefore, the repair urgency for a packet p_k present at node n_i, is computed as a function of a) the sum of the missing packets at each node $n_j \in \mathcal{J}$ for which $\mathcal{A}_{jk} > 0$, and b) the sum of the urgency of p_k for nodes $n_j \in \mathcal{J}$, given by $\sum_{j \in \mathcal{J}} \mathcal{A}_{jk}$. The packet that has the greatest non-zero repair urgency constitutes the repair payload, and the repair urgency for n_i is set equal to the repair urgency of that packet. If no packet has a non-zero repair urgency, then n_i's transmission will carry no repair payload. The solicit and repair urgency allow a node to estimate the utility of capturing the channel from a global perspective. This is accomplished by setting a *Wait Timer* proportional to the overall urgency. Hence nodes with a higher urgency value will succeed in capturing the channel. This serves as a scheduling heuristic to expedite the total time taken to complete the batch-repair.

Each node transmits $\mathcal{U}$, $\mathcal{J}$, and the repair payload (if any). Upon receiving a transmission from node n_i, each neighboring node updates its local $\mathcal{A}$ as the following. It sets $\mathcal{A}_{ik} = \mathcal{U}^i_k$, where $\mathcal{U}^i$ is n_i's urgency map. Further, suppose the transmission carries a repair payload p_k, and n_c is in the list of known neighbors of both n_i and n_j. Then n_j assumes that n_c also received the transmission and sets $\mathcal{A}_{ck} = 0$. n_j can then recompute its repair and solicit urgency and set a new wait timer. The protocol continues until $\mathcal{A}_{jk} = 0$ at every node in the network.

We now evaluate the performance of DCPR through simulations. As in Section III, every node has a uniform probability $P_r = 0.7$ of correctly receiving a packet. In *Experiments 1* through *4*, our nodes are uniformly distributed over a $700m \times 700m$ area, while *Experiment 5* has them uniformly distributed over a $1000m \times 1000m$ area. For all experiments $r_i = 150m$ and $r'_i = 180m$, $\forall n_i \in \mathcal{V}$. The MBMS bit rate and IEEE 802.11 bit rate are set as 0.2 Mbps and 5.5 Mbps, respectively. Every packet in the batch is 1000 bytes long. We note from our earlier simulation results in Section III, that $ln(K) \times \hat{b}$ is a good approximation for $\hat{Q}$. We, therefore, refer to $ln(K) \times \hat{b}$ as the optimal approximation. The results presented are averaged over 25 simulation runs with random topologies and initial batch maps. We also evaluated the performance of DCPR for different values of P_r with similar results. We present results only with $P_r = 0.7$ to conserve space.

Figure 4 plots the total time taken by DCPR to repair all packets for different combinations on N and K. In all cases, we observe that the repair time of DCPR is comparable to our optimal approximation. This is in spite of the fact that we assumed complete global state information for our optimal approximation. Moreover, the DCPR repair time includes the DCPR protocol overhead incurred as a result of the extra control information propagated in DCPR packets. Figure 4 shows that DCPR repairs all packets within $8 - 15\%$ of the total time for the repair epoch. Hence, we see that DCPR is a viable solution for timely repair of lost packets and facilitates reliable 3G multimedia broadcast.

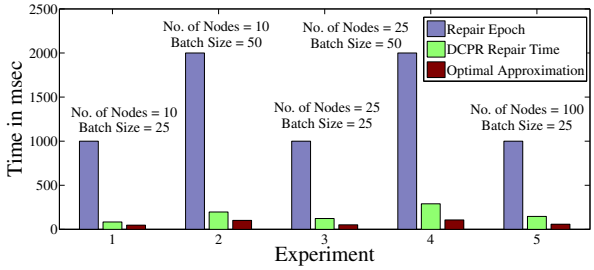

Fig. 4. Simulated Repair Time of DCPR

V. Conclusions

This paper presents the design of Cooperative Peer-to-Peer Repair (CPR) to facilitate reliable multimedia broadcast over 3G networks. CPR leverages IEEE 802.11 connections between peers that receive the same multimedia content through 3G to cooperatively repair missing packets locally. We outlined the fundamental design requirements of CPR, presented a formal problem description, and an optimal algorithm to solve it (assuming complete global state information). We then used insights gained from our optimal algorithm to design a practical protocol, DCPR. DCPR operates in a fully distributed fashion. Our simulation results show that the performance of DCPR is comparable to the optimal algorithm, and that DCPR is a viable mechanism to recover packets lost during 3G multimedia broadcast.

References

[1] "Technical Specification Group Services and System Aspects; Multimedia Broadcasting/Multicast Service; Protocol and Codecs," June 2005, 3GPP TS.26.346 v6.1.0.

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

[3] 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.

[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] P. Sharma, S.-J. Lee, J. Brassil, and K. Shin, "Distributed Communication Paradigm for Wireless Community Networks," in *IEEE ICC*, 2005, vol. 3, p. 15491555.

[6] K. Jain, J. Padhey, V. N. Padmanabhan, and L. Qiu, "Impact of interference on multi-hop wireless network performance," in *Proc. of ACM MOBICOM*, San Diego, CA, September 2003.

[7] G. Cheung, D. Li, and C. Chuah, "On the Complexity of Cooperative Peer-to-Peer Repair of Wireless Broadcasting," in *IEEE Communications Letters*, November 2006, vol. 10, no. 11, pp. 742–744.