Peer-to-Peer Netw. Appl.
DOI 10.1007/s12083-014-0297-8

Peer-to-peer error recovery for wireless video broadcasting

Bo Zhang · S.-H. Gary Chan · Gene Cheung

Received: 26 January 2013 / Accepted: 6 June 2014

Abstract Wireless video broadcasting has experienced much growth in recent years. In video broadcasting, packet loss is inevitable due to dynamic channel condition. To address this, we study peer-to-peer (P2P) error recovery. In our system, a mobile station (MS) may generate some parity packets based on its received source packets and share them by broadcasting to its neighbors via a secondary channel (e.g., Wi-Fi or Bluetooth). With parity packets from its neighbors, an MS can effectively repair its lost packets locally. An important problem is to minimize the total number of parity packets generated while achieving a certain residual loss rate at the MSs. We first formulate the problem as a linear program which can be solved efficiently as the optimal performance of the system. We then propose a novel and fully distributed algorithm based on only local information at clients. Simulation results show that our distributed solution achieves high recovery efficiency and fast convergence. It generates very low recovery traffic and high video quality. Its performance is very close to the optimal solution based on centralized approach with complete network information.

This work was supported, in part, by the HKUST Special Research Fund Initiative (SRFI11EG15) and Hong Kong Research Grant Council (RGC) General Research Fund (610713)

B. Zhang (✉) · S.-H. Gary Chan
Department of Computer Science and Engineering,
The Hong Kong University of Science and Technology,
Clear Water Bay, Kowloon, Hong Kong
e-mail: zhangbo@cse.ust.hk

S.-H. G. Chan
e-mail: gchan@cse.ust.hk

G. Cheung
National Institute of Informatics, 2-1-2 Hitotsubashi,
Chiyoda-ku, 101-8430 Tokyo, Japan
e-mail: cheung@nii.ac.jp

Keywords Wireless communication · Wireless networks · Broadcast technology · Cooperative systems · Peer-to-peer computing · Error compensation · Error recovery · Multimedia communication

1 Introduction

With the advance of mobile devices and networking, wireless video broadcasting to handhelds has become a reality [1, 2, 20]. In wireless video broadcasting, packets are broadcasted through a radio channel to mobile devices. During the process, packet loss often occurs due to the dynamic nature of wireless channel (i.e., random bit errors, burst errors or transient outages). Timely loss recovery is hence important to achieve good service quality.

In the traditional approach, a mobile station (MS) with packet loss, the so-called *errored MS*, requests the server for retransmission via a separate backward channel. This is not scalable to a large pool of MSs due to the bandwidth limitation of the wireless channel and at the server. With the development of network coding (NC), another approach lets the server perform network coding on source packets and broadcast coded packets to clients. This approach requires the knowledge of network loss condition, and is often designed for certain worst-case loss. In the case of higher losses, clients are often not able to retrieve any source packets due to the lack of NC packets received.

Given that nowadays MSs are often equipped with a (free) secondary broadcast channel (e.g., Wi-Fi or Bluetooth), such channel may be used for cost-effective cooperative error recovery. The scheme we proposed is termed Broadcast-based Peer-to-Peer Error Recovery, or BOPPER for short. In contrast with previous approaches, BOPPER adapts recovery packets according to different loss

Published online: 26 June 2014

conditions. It also reduces primary channel (e.g., 3G, LTE, DVB) traffic load by utilizing secondary channels and network coding.

A key issue in BOPPER is what type of packets to relay to the neighbors. A straightforward way is to retransmit the received source packets. However, because different MSs may lose different packets, the benefit brought by such retransmission is limited to the MSs losing the same packets. To make the recovery more efficient, we study a novel approach to generate parity packets for nodes to share with neighbors, by means of *linear network coding*. A node generates parity packets based on the possibly partial group of source packets received. When the number of received parity packets received from other peers is no fewer than the number of lost source packets, these source packets can be recovered with very high probability. Our parity generation method enjoys the following unique properties: 1) *Neighborhood-based*: An MS generates parity packets based on the error status of its neighbors, and hence the number of parity packets generated by an MS adapts local loss condition; 2) *Received-packets based*: An MS may generate parity packets based on the source packets it received via the primary channel. This is markedly in contrast to many previous works where network coding is also employed for primary channel data transmissions, and nodes can either retrieve all data packets if the number of incoming NC packets is large enough, or none of them. Furthermore, by employing NC for local error recovery, MSs with different losses may use the same parity packets for recovery, which effectively reduces recovery traffic in the secondary channel.

The basic idea of BOPPER is illustrated in Fig. 1. A server broadcasts data packets to MSs through a infrastructure (primary) wireless channel, such as a 3G WWAN network (the losses among mobiles are not likely to be highly correlated, as shown in the works of [25]). Suppose

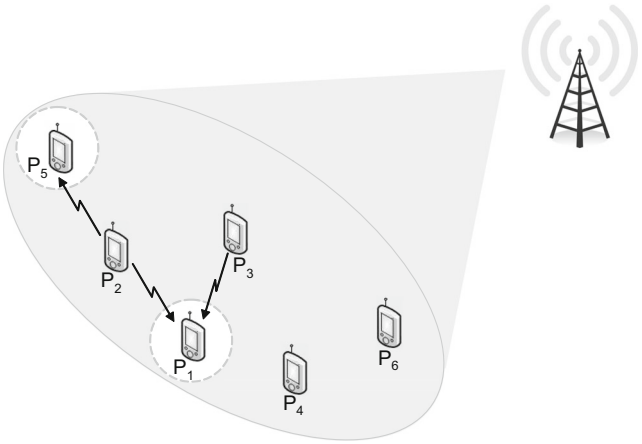


Fig. 1 Wireless peer-to-peer error recovery by means of network-coded packets

MSs P_1 and P_5 each loses a packet. P_2 and P_3 then generate some recovery parity packets and broadcast them using the device-to-device (secondary) channel. By receiving these parity packets, P_1 and P_5 may recover their lost packets.

Given the cost of parity generation and transmission (in terms of processing and bandwidth), we study in this paper how to minimize the number of parity packets in the network while meeting a certain level of recovery requirement. We address the following issues in this paper:

- *Problem formulation and exact solution for optimal parity generation:* We study the problem on the optimal number of parity packets generated by each MS, given network and MS information (e.g., connectivity and pair-wise loss rates in the primary and secondary channels). We formulate it as a linear program (LP) which can be solved efficiently. The optimal solution serves as the benchmark in our study.
- *BOPPER, a fully-distributed peer-to-peer (P2P) recovery algorithm:* With above, we propose and study a fully-distributed algorithm called BOPPER. Each MS locally exchanges information with its neighbors and dynamically decides the number of parity packets to be generated. Simulation results show that BOPPER converges fast and achieves high recovery with low bandwidth consumption even in a dense network. It achieves much better objective and subjective video quality. Its performance is very close to our optimal solution.
- *Extensive numerical study:* We conduct extensive numerical study to evaluate the performance of BOPPER. We also use video sequences to compare the subjective and objective quality. Simulation results show that BOPPER achieves high recovery efficiency and fast convergence. It generates very low recovery traffic and high video quality. Its performance is very close to the centralized optimal solution that based on the complete network information.

The rest of this paper is organized as follows. We briefly discuss related work in Section 2. We present the problem formulation on optimal parity generation as a linear program in Section 3. We propose BOPPER in Section 4. We present illustrative simulation results in Section 5. We conclude in Section 6.

2 Related work

There have been many erasure-resilient coding algorithms studied, such as Reed-Solomon code and fountain codes (e.g., Tornado code, Raptor code or Luby Transform (LT) code [3, 18, 32]). Much work has been done using FEC to recover errors [12, 13, 19, 23, 26]. FEC is normally designed

for a certain worst-case loss rate, which uses bandwidth unnecessarily in a low loss condition, and is insufficient to recover loss in a high loss condition. BOPPER is highly adaptive to heterogeneous loss rates. It is complementary to FEC, and can be applied *after* FEC to reduce transmission overhead and to achieve lower residual loss rate. We review linear network coding and discuss how it is effectively used in our system in Appendix A.

There has been much work on server-retransmission (of either source packets or network coded packets) based error recovery for wireless video by feeding back loss information to the server [5–10, 27, 28]. A common assumption for all the above work is a feedback channel to the server, so that ACKs or NAKs can be reported back to trigger retransmission at the server. This approach is not scalable as mentioned in Section 1. In comparison, BOPPER recovers errors locally within the neighborhood, without the need of a feedback channel to the server.

Cooperative loss recovery has been studied in [4, 15, 21, 22, 24, 30, 31]. Source packet sharing is adapted in most of these works, reducing the recover effectiveness. NC-based cooperative recovery is adapted in [4]. However in the proposed approach, cooperative recovery is *unicast* based, limiting the effectiveness of recovery packets to a single receiver. An implementation of wireless cooperative recovery over the Android platform is presented in [17], though without the intension of optimizing any particular metric. In BOPPER, parity packets are generated and NC-coded according to both the loss of neighbors and their protection levels, and are then broadcasted to all neighbors through 1-hop broadcasting, allowing those parity packets to be maximally utilized.

BOPPER, as many other peer-cooperative recovery schemes, relies on sufficient number of neighboring nodes with low loss correlation to perform effectively. Sufficiently dense network can often be found in places such as in an airport lounge, in stadiums, on a cruise ship etc., while popular contents may even find themselves requested by large number of clients on streets. Antenna diversity (been utilized by MIMO) assures low loss correlation in most cases in many transmission networks such as 3G, LTE and DVB. So we conclude that the cases where BOPPER can perform effectively are not rare. If in extreme conditions where there is insufficient number of neighboring nodes with low loss correlation (consider single client case, or regional network outage), server retransmission appears to be the only viable method for recovery. So BOPPER, instead of a standalone error recovery scheme, can be used as a complementary scheme in addition to both FEC and source retransmission, to achieve low residual loss rate more effectively and timely.

Some other error recovery schemes work in physical/data link layers [14, 16]. The limitations of cross-layer algorithms lie in portability and deployment. Some schemes try to extract useful data from error packets [11, 14, 29]. Although error recovery transmissions can be reduced, this approach greatly increases computational load at receiver end for data processing. BOPPER is an application layer scheme, making it highly portable and easy to deploy. Furthermore, BOPPER can be applied *after* these schemes to further reduce loss.

As for the transport layer protocol, BOPPER assumes UDP or similar protocol is used. Although TCP is able to guarantee full recovery, its congestion control often severely harm wireless transmission throughput, since packet losses in wireless environment are not always due to congestion, as assumed in TCP. Furthermore, UDP greatly eases inter-device communication.

We focus on the issue of efficient content delivery in the assumed scenario instead of a complete video distribution system, while others aspects of the system such as security and pricing strategies, are orthogonal to our work.

3 Problem formulation as a linear program

In this section, we formulate the problem of parity generation in BOPPER, given complete network knowledge and independent packet loss.

In Table 1 we summarize the important symbols used in this paper.

We model the network as a directed graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of all MSs and $\mathcal{E}$ is the set of links such that $(i, j) \in \mathcal{E}$ if and only if MS j is in transmission range of MS i in the secondary channel. Each coding window consists of K source packets, and parity packets are generated based on the source packets received in each window. Because each coding window is handled independently, the following discussion is limited to one window.

For delay consideration, there is an upper bound for the number of parities that can be broadcasted by each node, denoted as $n_{\max}$. The recovery delay is hence the window size plus the transmission time of the $n_{\max}$ parities. Let n_i be the number of parity packets generated by MS i in one window and $\mathcal{R}^+$ be the set of all positive real numbers. We have

$$n_i \in \mathcal{R}^+, 0 \leq n_i \leq n_{\max}, \forall i \in \mathcal{V}. \tag{1}$$

The total number of parities generated and transmitted is then given by

$$\sum_{i \in \mathcal{V}} n_i. \tag{2}$$

Note that the number of parities generated and transmitted is proportional to the computational power, energy and bandwidth used in the system. Our target is hence to minimize the total number of parities in the system. We show

Table 1 Symbols used in the paper

Symbol	Description
K	The number of source packets in a coding window
$\mathcal{V}$	Set of MSs in the network
$\mathcal{E}$	Set of transmission edges
$\mathcal{L}_j$	Set of lost source packets in a coding window at MS j before recovery
$\mathcal{R}_k$	The set of MSs which have successfully received source packet k before recovery
$\mathcal{N}_j$	Set of neighboring MSs of MS j
$l_{i,j}$	Packet loss rate from i to j in the secondary channel
$n_{\max}$	Maximal number of parity packets broadcasted by each node in a coding window
n_i	The number of parity packets generated and broadcasted by MS i in a coding window
$N_{i,j}$	Expected number of parity packets received by MS j and from MS i in a coding window
θ_j	Target recovery level for MS j (in terms of the average number of parities)
$\mathcal{C}_j$	Set of correctly received source packets in a coding window at MS j before recovery
$\mathcal{NAK}_j$	The set of MSs whose NAKs are heard by MS j
$\hat{N}_{i,j}$	The number of parity packets received by MS j from MS i in a coding window
$\bar{\theta}_j$	The average recovery level achieved by MS j
$\hat{\theta}_j$	Recovery level achieved by MS j in a certain coding window
f_j	Current level of protection at MS j $(= \bar{\theta}_j/\theta_j)$
α	A smoothing factor between 0 and 1 for computing $\bar{\theta}_j$
ω	Maximal waiting time at an MS before sending a NAK
$W_{\max}$	A system-level upper bound (may not be strict) for ω
ϵ_p	Packet loss rate in the primary channel
ϵ_r	Residual loss rate
γ	Number of parities generated per MS per source packet

in simulation that the control overhead in the secondary channel is negligible.

Let $l_{i,j}$ be the packet loss rate from MS i to j in the secondary channel, $\forall(i, j) \in \mathcal{E}$. Due to peer mobility and collision condition, $l_{i,j}$ may change over time. The expected number of parity packets received by i from j, $N_{i,j}$, is then given by

$$N_{i,j} = n_i(1 - l_{i,j}), \quad \forall(i, j) \in \mathcal{E}. \tag{3}$$

Let $\mathcal{L}_j$ be the set of lost packets in the coding window at MS j before recovery. In order to recover all those packets, the number of parities has to be at least the number of lost packets. As some parities may be linearly dependent or lost, we require that the total parity packets received within a window to be larger than $\theta_j|\mathcal{L}_j|$, for some $\theta_j \geq 1$. In other words, let $\mathcal{R}_k$ be the set of MSs who received source packet k before recovery, and $\mathcal{N}_j = \{i|(i, j) \in \mathcal{E}\}$ be the set of all neighbors of j. We require

$$\sum_{i \in \mathcal{N}_j \cap \mathcal{R}_k} N_{i,j} \geq \theta_j|\mathcal{L}_j|, \quad \forall j \in \mathcal{V}, k \in \mathcal{L}_j. \tag{4}$$

Given Eq. 3, our problem hence is to minimize Eq. 2 subject to Eqs. 1 and 4. This is a linear program problem and can be solved efficiently.

4 BOPPER: Distributed and P2P error recovery

In this section, we describe BOPPER, a distributed algorithm that minimizes the average number of parity packets in the network while achieving a performance close to the optimal solution given in Section 3 (based on centralized and complete network information).

4.1 NAK suppression

In the algorithm, an errored MS j broadcasts a NAK that contains the $\mathcal{L}_j$ value as a bitmap to indicate its lost packets, to all MSs in its neighborhood. Clearly, the number of NAKs increases with the network size, which may overwhelm the channel. To suppress NAKs, before j broadcasts its NAK, it first checks whether $\mathcal{L}_j$ has already been covered by the NAKs it heard from others, i.e., it checks whether $\mathcal{L}_j \subseteq \bigcup_{i \in \mathcal{NAK}_j} \mathcal{L}_i$, where $\mathcal{NAK}_j$ is the set of MSs whose NAKs are heard by j. If so, its neighboring MSs have already requested all packets in $\mathcal{L}_j$, and hence j suppresses its own NAK.

The NAK suppression comes with some cost: a lost packet may not be recovered. We illustrate that in Fig. 2. Both MSs P_2 and P_3 have lost the same packet. Suppose P_2 broadcasts its NAK first. This suppresses P_3's NAK. On the other hand, P_4 does not receive the NAK from P_2 due

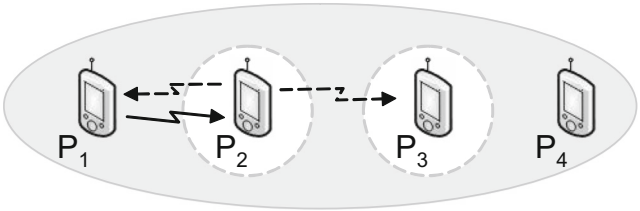


Fig. 2 NAK suppression: dashed arrows denote NAKs; solid arrow denotes recovery packets

to limited power range. P_4 is hence not aware of the loss at P_3. Since the recovery packet broadcasted from P_1 cannot reach P_3, P_3 cannot recover its lost packet. Our simulation results in Section 5 show that the suppression method could significantly reduce the NAK number with only a small penalty in recovery effectiveness.

4.2 Parity generation and error recovery

Besides $\mathcal{L}_j$, an errored MS j also needs to provide another parameter to help its neighbors determine the number of parity packets to generate. We denote the parameter as f_j (given later in Eq. 7), which indicates the current level of "protection" at j. If $f_j > 1$, it is "over protected" with many parity packets (e.g., in a low loss environment) and hence its neighbors may reduce their parity generation. On the other hand, if $f_j < 1$, then j is "under protected" and hence its neighbors need to generate more parities. The value of f_j is obtained as follows.

An errored MS j has a target recovery level θ_j as defined in Eq. 4. Let $\hat{N}_{i,j}$ be the number of parity packets received by j from MS i. j calculates its current recovery level $\hat{\theta}_j$ as

$$\hat{\theta}_j = \min_{k \in \mathcal{L}_j} \sum_{i \in \mathcal{N}_j \cap \mathcal{R}_k} \frac{\hat{N}_{i,j}}{|\mathcal{L}_j|}. \tag{5}$$

Here $|\mathcal{L}_j| \geq 1$ because errored MS j has lost packet(s).

As $\hat{\theta}_j$ fluctuates over time, we use exponential smoothing to calculate its average $\bar{\theta}_j$. That is, after calculated $\hat{\theta}_j$ for a certain coding window, MS j sets

$$\bar{\theta}_j \leftarrow \alpha \bar{\theta}_j + (1-\alpha)\hat{\theta}_j, \tag{6}$$

where α is a smoothing factor between 0 and 1. The value of f_j is then given by

$$f_j = \frac{\bar{\theta}_j}{\theta_j}, \tag{7}$$

which will be contained in the NAK message together with $\mathcal{L}_j$.

Given the above, each MS runs the following distributed algorithm. An MS successfully received all packets will skip the *NAK generation* and *Packet recovery* steps.

- *NAK generation:* After packets transmission of a coding window, each errored MS waits for a random time between 0 and ω before it sends its NAK with $\mathcal{L}_j$ and f_j, if not suppressed. Usually, an errored MS with low protection (i.e., with a low f_j) should send its NAK soon and its NAK should not be suppressed. Therefore, ω is chosen to be an increasing function in f_j. An example is

$$w = \frac{f_j}{f_j + g} W_{\max}, \tag{8}$$

for some constant g. Here $W_{\max}$ is a system-level upper bound for the maximum waiting time. In our simulations, we set $g = 1$. Clearly, ω may be defined in terms of other functions of f_j. It will be our future work to study more efficient functions.

- *Parity generation:* After the transmission of packets in a coding window, Nodes spend an "NAK collection" period to both advertise and collect NAKs to and from neighbors. For node i, this period expires when either $|N_i|(1-(1-\varepsilon_p)^K)$ number of NAKs, the number of expected NAKs from the neighbor set N_i, are received, or a max delay $h(T_{NAK} + W_{\max})$, h times the time that a single NAK transmission is needed plus $W_{\max}$ is expired. In parity generation, our algorithm adaptively adjusts the recovery level. An MS with higher loss (i.e., with a low f_j) will require more parity packets. Let $\mathcal{C}_j$ be the set of source packets in a coding window that are correctly received by j before recovery. A NAK from j is ignored by MS i if $\mathcal{C}_i \cap \mathcal{L}_j = \emptyset$. This means that i cannot offer any help to j. After NAK collection period, a node uses the smallest received f_j from NAKs that $\mathcal{C}_i \cap \mathcal{L}_j \neq \emptyset$ to adjust the parity number according to

$$n_i \leftarrow \begin{cases} \min\left(\frac{n_i}{\min_{j \in NAK_i} f_j}, n_{\max}\right), & f_j > 0; \\ n_{\max}, & \text{otherwise.} \end{cases} \tag{9}$$

Thus, if $f_j < 1$, n_i increases so as to increase the average number of parity packets received at j, and vice versa. MS i then generates n_i parity packets from $\mathcal{C}_i$ and broadcasts them.

If n_i in Eq. 9 is not an integer, i broadcasts $\lceil n_i \rceil$ parity packets with probability $\{n_i\}$, and $\lfloor n_i \rfloor$ packets with probability $(1-\{n_i\})$, where $\{n_i\} \equiv n_i - \lfloor n_i \rfloor$ is the fractional part of n_i. Thus, the expected number of

broadcast packets is exactly n_i, which is shown in the following equation:

$$\begin{aligned}&\lceil n_i \rceil (n_i - \lfloor n_i \rfloor) + \lfloor n_i \rfloor (1 - (n_i - \lfloor n_i \rfloor)) \\ &\quad = (\lfloor n_i \rfloor + 1)(n_i - \lfloor n_i \rfloor) + \lfloor n_i \rfloor (1 - (n_i - \lfloor n_i \rfloor)) \\ &\quad = (n_i - \lfloor n_i \rfloor) + \lfloor n_i \rfloor \\ &\quad = n_i . \end{aligned} \tag{10}$$

– *Packet recovery:* After received a parity packet from i (with the information of $\mathcal{C}_i$) error j checks whether it could achieve recovery from the parity. If $\mathcal{C}_i \cap \mathcal{L}_j = \emptyset$, the parity is discarded since no lost source packet at j can be recovered from it. Otherwise, j conducts progressive decoding based on Gauss-Jordan elimination.

  The decoding process could start as early as the arrival of the first parity packet. Detailed decoding process using Gauss-Jordan elimination is reviewed in Appendix A.

In Appendix B, we illustrate with numerical examples how a peer recovers packets with parities received in BOPPER.

5 Illustrative numerical results

In this section, we present our illustrative simulation setup and discuss the simulation results.

5.1 Environment and performance metrics

In our simulations, peers are uniformly distributed in a 1500×1500 m^2 area. Each peer has a transmission range of 250 m. Let ε_p and ε_s be the link loss rate in primary channel and secondary channel, respectively. Note that ε_p may be the loss rate after some error recovery mechanism such as FEC has been applied. We assume 802.11 mac with collision avoidance and exponential back-off is employed in the secondary channel. We set the bandwidth limit to 10mbps, within the data rate of the 802.11n standard. Our results show that the bandwidth used for recovery in BOPPER is far less than the available bandwidth. UDP protocol is assumed for video data transmission in both primary channel and secondary channel for efficiency purpose. Unless otherwise stated, we use the following baseline parameters:[1] $|\mathcal{V}| = 500$, $\varepsilon_p = 0.1$, $\varepsilon_s = 0.1$, $K = 10$, $n_{\max} = 2$, $\theta = 1.1$ and $\alpha = 0.5$. Other parameters settings have also been studied, with qualitatively same results obtained. The video sequence used in testing is Foreman cif sequence with resolution of 352×288 and the streaming rate of 440kbps.

[1]Though we consider random loss in our simulations, our algorithm and results are applicable to other loss model such as pseudo-stationary time-dependent losses.

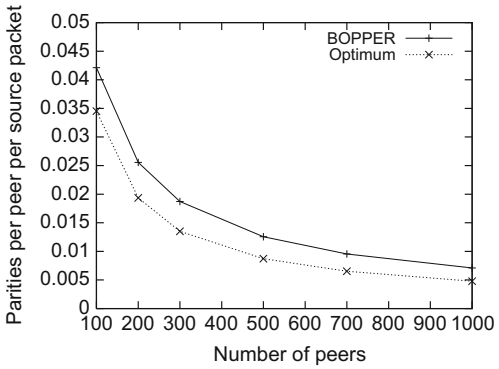


Fig. 3 Average γ vs. $|\mathcal{V}|$

We assume the primary channel bandwidth is sufficient for video broadcasting, and focus on the cooperative recovery phase. We will show later, by analyzing our results, that the recovery traffic is low.

To evaluate the optimum solution, we use the parameters obtained from the linear programming solution and simulate peer actions accordingly. We also apply BOPPER under the same setup. Finally, we compare BOPPER with its previous version in which source packets are used for recovery instead of NC-coded parities.

We study the following metrics to evaluate BOPPER:

– *Number of parities generated per peer per source packet (γ)*, defined as the total number of parities generated in a coding window divided by the total number of peers and the window size, i.e.,

$$\gamma = \frac{1}{K \cdot |\mathcal{V}|} \sum_{i \in \mathcal{V}} n_i . \tag{11}$$

– *Residual loss rate (ε_r)*, defined as the loss rate after recovery at a peer, given by

$$\varepsilon_r = \frac{\text{\# of lost packets after recovery}}{\text{\# of broadcast packets in the primary channel}} . \tag{12}$$

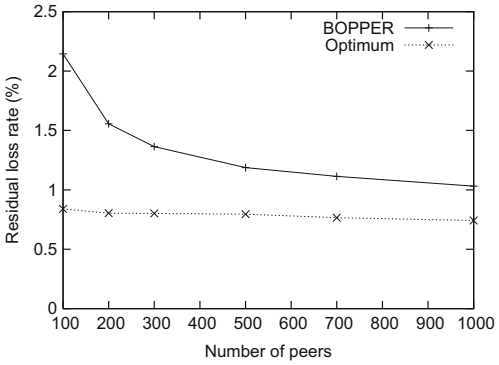


Fig. 4 Residual loss rate ε_r vs. $|\mathcal{V}|$

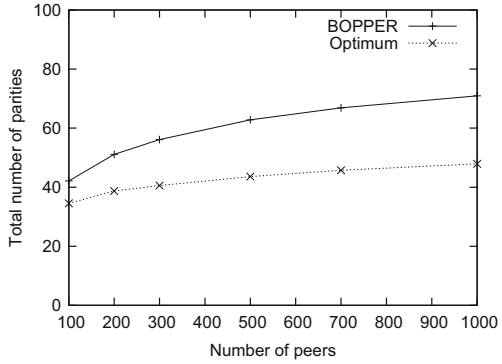


Fig. 5 Total number of parities in a coding window versus $|\mathcal{V}|$

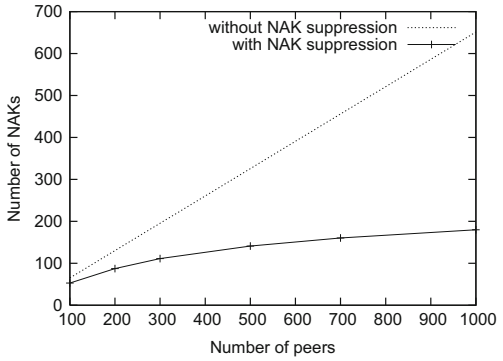


Fig. 7 Number of NAKs in a coding window vs. $|\mathcal{V}|$

– *PSNR (Peak Signal-to-Noise Ratio)*: We evaluate video quality resulted using PSNR for each GOP (group of pictures) at the receiver side. Besides this objective measure, we also show its subjective (visual) quality.
– *Total number of NAKs transmitted in a coding window*. We compute this metric to evaluate our suppression method. The number of NAKs without NAK suppression is equal to the number of errored MSs, which is given by

$$(1-(1-\varepsilon_p)^K)|\mathcal{V}|. \tag{13}$$

5.2 Illustrative results

Figure 3 shows the average (γ) value versus the network size. γ decreases as $|\mathcal{V}|$ increases because a parity packet may reach, and hence be shared by, more error peers in a denser network. From the figure, the gap between BOPPER and the optimum is small. Both of them keep γ at a low level.

Figure 4 shows the residual loss rate ε_r versus $|\mathcal{V}|$. For BOPPER, ε_r decreases as $|\mathcal{V}|$ increases. Because in a denser network, there are more neighbors available to help an errored MS. For the case of optimum, ε_r is rather insensitive to $|\mathcal{V}|$, since with the complete knowledge, a peer can easily identify appropriate neighbors for parity retrieval.

Figure 5 shows that as $|\mathcal{V}|$ increases, the total number of parities also increases. However the sharing of parities greatly reduces the increasing speed. The NAK suppression method in BOPPER further helps reducing the number of parities.

Figure 6 shows the distribution and cumulative distribution function (CDF) of the number of parities per source packet in BOPPER. From the figure, the majority of peers generate very few parities (around 0.02) for each source packet. Most peers therefore experience low recovery loads. A few peers generate slightly more parities (around 0.05). Since the number of parities generated by a peer is determined by the neighborhood loss rate, peers with different densities experience different recovery loads.

Figure 7 shows the number of NAKs in a coding window with or without suppression with respect to the network size. Without suppression, the number of NAKs increases linearly with $|\mathcal{V}|$, because the number of errored MSs linearly increases with $|\mathcal{V}|$. With suppression, the number of NAKs increases much slower. This shows that our NAK suppression method is highly efficient, even in a dense network. Figure 8 shows the number of NAKs in a coding window with or without suppression with respect to the primary network loss rate. Without suppression, the number

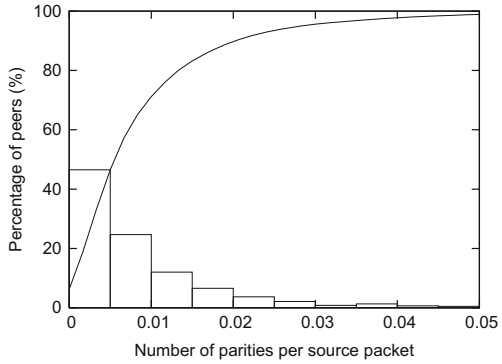


Fig. 6 Distribution (indicated by the bars) and CDF (indicated by the curve) of n_i/K

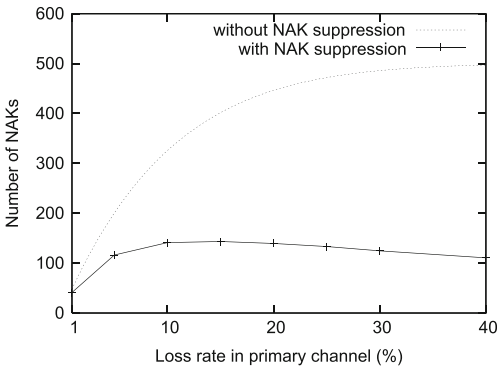


Fig. 8 Number of NAKs in a coding window vs. $|\mathcal{V}|$

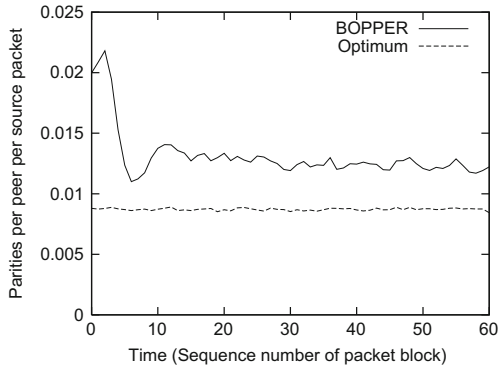


Fig. 9 Time evolution of average γ

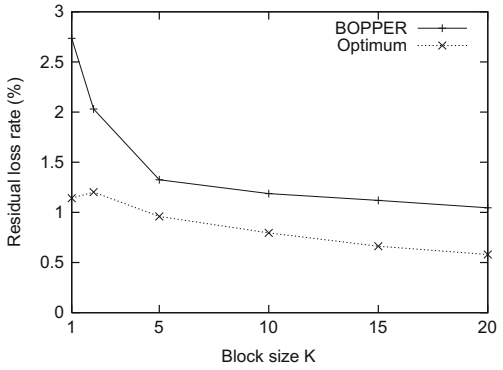


Fig. 11 Residual loss rate ε_r vs. K

of NAKs increases very fast when primary network loss rate is initially small and start increasing, as more nodes start to experience loss. The number of NAKs finally converges to $|\mathcal{V}|$, as every single node sends out an NAK. Note that many of those NAKs are duplicates. With suppression, the number of NAKs again increases much slower. This result shows that NAK suppression can successfully suppress those duplicated NAKs.

From the above figures, we can roughly estimate the recovery traffic in BOPPER. With 250 m transmission radius, a peer covers an area of $\pi \times 250^2 = 196,350\text{m}^2$. Let's consider the case that $\varepsilon_p = 10\%$, and that there are 500 peers uniformly distributed in a square area of $1500 \times 1500 = 2,250,000\text{m}^2$. There are then around $500 \times 196,350/2,250,000 = 43.6$ neighbors for each peer (a dense network). From Fig. 5, the total number of parities in a coding window is around 60 for a 500-peer network. From Fig. 7, the total number of NAKs in a window is around 140 (with NAK suppression). Each peer therefore generates 0.4 NAK messages or parity packets on average. Given the coding window size $K = 10$, each peer hence generates 0.04 messages or packets for each source packet on average. As a result, there are in total $0.04 \times 43.6 = 1.74$ NAK messages or parity packets for each source packet within a neighborhood. Suppose the data stream is of 200kbps (for a typical wireless video), the recovery traffic is then about $200 \times 1.74 = 348$kbps, which is considered a low load for secondary channels such as Wi-Fi (ranges from 1Mbps of 802.11b to 433Mbps of 802.11ac, single antenna) and Bluetooth (with data rate of 1Mbps, 3Mbps or 24Mbps). We therefore conclude that collision probability on the secondary channel is low, even for a very dense network. We could therefore expect that losses can be recovered efficiently with low delay.

Figure 9 shows the evolution and convergence of γ over time (in terms of the sequence numbers of coding windows). In BOPPER, we start with an arbitrary initial value $n_i = 0.2, \forall i \in \mathcal{V}$ with $\gamma = (|\mathcal{V}| \cdot n_i)/(K \cdot |\mathcal{V}|) = 0.02$. From the figure, γ of BOPPER starts to decrease very soon. This is because we have used a relatively large n_i (and hence a large γ value) to ensure that every errored MS receives enough parities. So γ decreases to a lower value in the steady state. BOPPER approaches its steady state after only a few windows. It hence achieves high convergence speed and is adaptive to network environment change. The line corresponding to the optimum shows the optimal γ value given complete network knowledge.

Figure 10 shows the convergence of ε_r over time (in terms of sequence numbers of coding windows). In BOPPER, ε_r converges to a stable value after around 10 coding

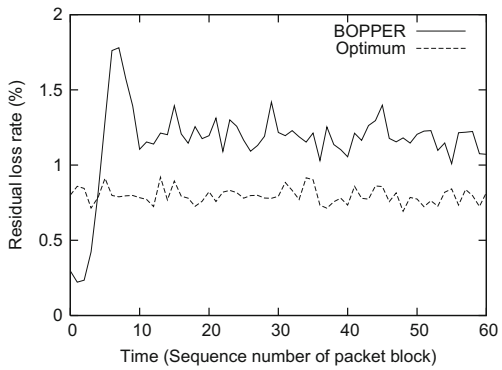


Fig. 10 Time evolution of ε_r

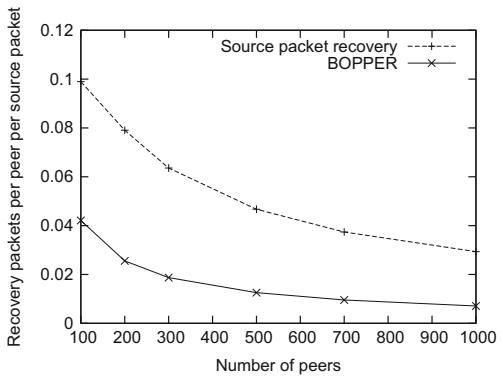


Fig. 12 Average recovery packets generated

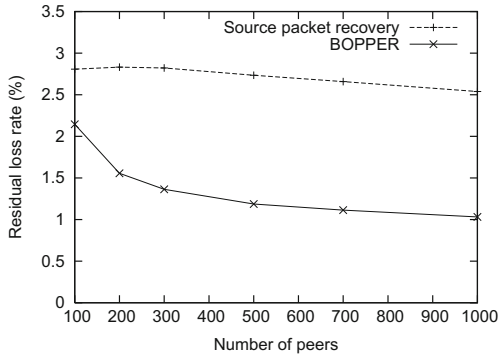


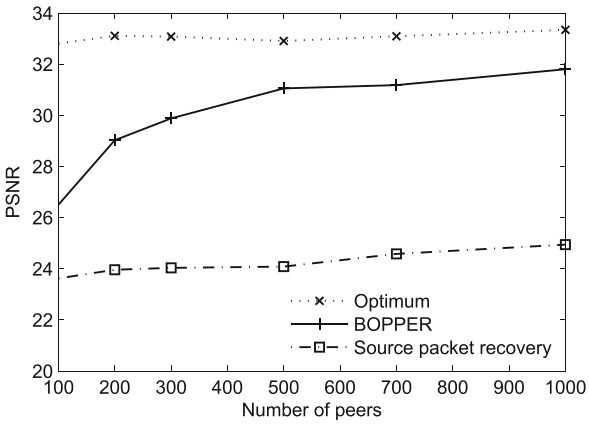
Fig. 13 Residual loss rate ε_r

windows. This again shows its high efficiency and high adaptiveness. Note that in the first several windows, the distributed algorithm achieves lower ε_r than the optimum. This is because in the first several windows, BOPPER generates much more parities than the optimum with the high starting γ value. Later, BOPPER adaptively reduces its γ value, leading to a increase of its ε_r value.

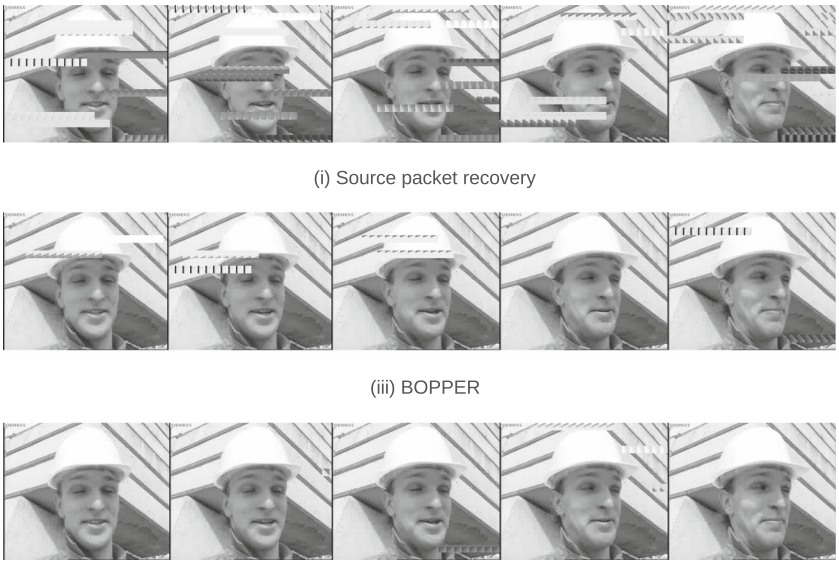
Figure 11 shows residual loss rate ε_r versus window size K. In BOPPER, When K increases, ε_r keeps decreasing. This is because with a larger K, a parity packet may be used to recover more different losses. We do not need a large K value to achieve low loss as shown in the figure. It is sufficient to set K to between 5 and 10. Clearly, smaller K leads to lower recovery delay in the system.

Figure 12 compares the numbers of recovery packets using network coding (as in BOPPER) and using source packet recovery. For source packet recovery, the only difference is that instead of network coding, it shares source packets for cooperative recovery. After receiving NAKs from the neighbors, a peer randomly selects a source packet

Fig. 14 Objective and subjective video quality



(a) PSNR vs. network size



(b) Subjective quality comparison

that has been requested by some NAK. The peer then repeats this process for n_i times to select in total n_i source packets and re-transmits them. As shown in the result, BOPPER generates much fewer recovery packets. We also compare the two schemes in terms of the residual loss rate in Fig. 13. The result shows that BOPPER achieves lower residual loss rate than source packet recovery.

Objective and subjective video quality is shown in Fig. 14. With different recovery efficiencies, source packet recovery and BOPPER result in two different residual losses. Thus the two algorithms result different residual losses. We record a instance of losses and introduce the loss into the video stream to calculate corresponding PSNR values. As shown in Fig. 14a, while the optimum is able to achieve nearly full PSNR (37.824dB for the original stream source), the performance of BOPPER quickly rises as the network get denser due to P2P effect. Figure 14b shows the subjective visual quality for a few consecutive I frames sampled from received video stream in a 500-peer network. The source stream is encoded using H.264 codec with a PSNR of 37.824dB. We can see that subjectively, the visual quality difference between BOPPER and the optimum is small, while both schemes outperform the source packet recovery version. From Fig. 14 we can see that BOPPER results better video quality than the source packet recovery version, both objectively and subjectively.

6 Conclusion

In order to repair the lost packets in wireless video broadcasting, we study in this paper a relay-based error recovery scheme. The scheme makes use of a broadcast-based secondary wireless channel to share parity packets, which are generated by mobile peers from their received source packets using linear network coding.

We seek to minimize the parity packets generated so as to reduce the processing and bandwidth cost in the network. We first formulate the problem on optimal parity generation as a linear programming problem, given complete knowledge of the network. The problem can be solved efficiently and its optimal solution serves as the benchmark in our comparison. We then present a fully distributed algorithm called BOPPER which generates parity packets based on only local information at peers.

We have conducted simulations to evaluate our schemes. Our results show that BOPPER achieves low recovery traffic (in parity generated), low residual loss rate, fast convergence, and low overhead (efficient NAK suppression). It performs close to optimal solution as obtained by complete and centralized network knowledge. With its use of linear network coding to generate parity packets, BOPPER performs much better than the traditional schemes based on source packet retransmissions. It achieves much better objective and subjective video quality.

Appendix A: Linear network coding and its decoding in BOPPER

Given a group of source packets, a parity packet is a linear combination of these source packets with certain coefficients. Suppose that K source packets in a coding window are $M_1, \ldots, M_k$. A peer may select a set of coding coefficients $[c_1^j, \ldots\ldots, c_k^j]$ for parity packet j. It then generates a parity packet as

$$\left\{ x_j = \sum_{i=1}^{K} c_i^j \cdot M_i \right\}, \text{ for } j \in [1, h]. \tag{14}$$

Note that a coefficient c_i^j could be 0. In other words, if a source packet in the coding window is missing, we can simply set its coefficient to 0. This is a fundamental difference from FEC and fountain codes. Similarly, the peer can select another set of coefficients for another parity packet. If all peers share a common pseudo random number generator, only the random seed used to produce a corresponding coefficient series need to be embedded in the header of the parity packet. Therefore, a parity packet is of the same size as a single source packet, excluding a few bytes for the seed.

For decoding, suppose a peer has lost d source packets in a coding window. When it receives h parity packets, it needs to solve (14).

Here the unknowns are the lost source packets M_i. Clearly, this is a linear system with h equations and d unknowns. We can use Gauss-Jordan elimination to solve the system. We need $h \geq d$ to have a chance of recovering all unknowns. Note that $h \geq d$ is not a sufficient condition, as some of the coefficient combinations might be linearly dependent.

We use a simple example to illustrate the progressive decoding. Suppose there are 3 source packets in a coding window, which are denoted as M_1, M_2 and M_3. For simplicity, we use a numerical value to represent the value of a source packet or a parity packet. Suppose error peer j only correctly received M_1 in the coding window. It knows that $M_1 = 10$, but not the values of M_2 and M_3. Suppose the first parity packet arriving at j has the value 1220, with a coefficient set (5, 18, 27). It indicates that

$$5M_1 + 18M_2 + 27M_3 = 1220. \tag{15}$$

The equations can be written in the augmented matrix form as

$$\left[\begin{array}{ccc|c} 1 & 0 & 0 & 10 \\ 5 & 18 & 27 & 1220 \end{array}\right]. \tag{16}$$

We use R_1 and R_2 to represent row 1 and row 2 in the matrix, respectively. We will use elementary row operations to transform this matrix into reduced echelon form. Let $R_2 \leftarrow (R_2 + (-5)R_1)/18$. We have

$$\left[\begin{array}{ccc|c} 1 & 0 & 0 & 10 \\ 0 & 1 & 1.5 & 65 \end{array}\right]. \tag{17}$$

Later, the second parity packet arrives at j with a value 570 and a coefficient set (34, 4, 5). It indicates that

$$34M_1 + 4M_2 + 5M_3 = 570. \tag{18}$$

Hence, the matrix can now be written as

$$\left[\begin{array}{ccc|c} 1 & 0 & 0 & 10 \\ 0 & 1 & 1.5 & 65 \\ 34 & 4 & 5 & 570 \end{array}\right]. \tag{19}$$

Let $R_3 \leftarrow R_3 + (-34)R_1 + (-4)R_2$. We have

$$\left[\begin{array}{ccc|c} 1 & 0 & 0 & 10 \\ 0 & 1 & 1.5 & 65 \\ 0 & 0 & -1 & -30 \end{array}\right]. \tag{20}$$

Let $R_2 \leftarrow R_2 + 1.5R_3$, and $R_3 \leftarrow (-1)R_3$. We have

$$\left[\begin{array}{ccc|c} 1 & 0 & 0 & 10 \\ 0 & 1 & 0 & 20 \\ 0 & 0 & 1 & 30 \end{array}\right]. \tag{21}$$

This reduced echelon form matrix indicates that

$$M_1 = 10,\ M_2 = 20 \text{ and } M_3 = 30. \tag{22}$$

Therefore, with two linearly independent parity packets, error peer j could recover its lost source packets M_2 and M_3.

Appendix B: An illustrative example of error recovery in BOPPER

In this section, we illustrate with numerical examples to show how BOPPER achieves error recovery. Suppose each coding window consists of 5 source packets, i.e., K = 5. In a window, the server transmits five source packets $\{M_1, M_2, M_3, M_4, M_5\}$.

We first consider the *full recovery case*. Suppose MS P_1 has received packets $\{M_3, M_4, M_5\}$ while its neighbors P_2 and P_3 received packets $\{M_2, M_3, M_4, M_5\}$ and $\{M_1, M_3, M_4, M_5\}$, respectively. So now we have

$$\begin{cases} \mathcal{C}_1 = \{M_3, M_4, M_5\}, \\ \mathcal{C}_2 = \{M_2, M_3, M_4, M_5\}, \\ \mathcal{C}_3 = \{M_1, M_3, M_4, M_5\}; \end{cases} \tag{23}$$

and

$$\begin{cases} \mathcal{L}_1 = \{M_1, M_2\}, \\ \mathcal{L}_2 = \{M_1\}, \\ \mathcal{L}_3 = \{M_2\}. \end{cases} \tag{24}$$

Assume that this is the very first coding window. We have $\hat{\theta}_i = 0, \bar{\theta}_i = 0,\ f_i = 0, \forall i$.

Suppose MS P_1 broadcasts its NAK with L_1 and f_1 to the neighbors P_2 and P_3. When P_2 receives the NAK from P_1, it finds that its own lost packets is only a subset of the lost packets at P_1, i.e., $\mathcal{L}_2 \subseteq \mathcal{L}_1$. Hence, P_2 suppresses its own NAK broadcasting. Similarly, P_3 also suppresses its NAK broadcasting.

Meanwhile, P_2 finds that $\mathcal{C}_2 \cap \mathcal{L}_1 \neq \emptyset$. It means that P_2 holds some source packets which are required by P_1. P_2 hence selects $n_{\max}$ sets of random coefficients to generate parity packets from its C_2 (since $f_1 = 0$) and broadcasts them. Similarly, P_3 also generates $n_{\max}$ number of parity packets and broadcasts them for P_1 (hopefully P_2 could overhear them). After P_1 receives both of P_2's and P_3's parity packets, P_1 can use Gauss-Jordan elimination to solve the linear system to recover M_1 and M_2. Meanwhile P_2 and P_3 can also recover lost packets from each other's parity packet.

After parity recovery, each peer i then updates f_i. Take P_1 for example. For M_1 it receives $n_{\max}$ number of parities from P_2; for M_2 it receives the same number of parities from P_3. We also have $|\mathcal{L}_1| = 2$. Suppose we set $n_{\max} = 2$, $\alpha = 0.5$ and $\theta_1 = 1.1$. We have

$$\hat{\theta}_1 = \min\left(\left(\frac{2}{2} + 0\right), \left(0 + \frac{2}{2}\right)\right) = 1, \tag{25}$$

$$\bar{\theta}_1 = 0.5 \times 0 + (1 - 0.5) \times 1 = 0.5, \tag{26}$$

and hence

$$f_1 = \frac{\bar{\theta}_1}{\theta_1} = \frac{0.5}{1.1}. \tag{27}$$

In the next round, if P_1 needs to send a NAK, the above f_1 value will be sent together.

We next consider the *partial recovery case*. Suppose now we have

$$\begin{cases} \mathcal{C}_1 = \{M_3, M_4, M_5\}, \\ \mathcal{C}_2 = \{M_2, M_3, M_4, M_5\}, \\ \mathcal{C}_3 = \{M_4, M_5\}; \end{cases} \tag{28}$$

and correspondingly,

$$\begin{cases} \mathcal{L}_1 = \{M_1, M_2\}, \\ \mathcal{L}_2 = \{M_1\}, \\ \mathcal{L}_3 = \{M_1, M_2, M_3\}. \end{cases} \tag{29}$$

P_3 broadcasts its NAK with $\mathcal{L}_3$ to P_1 and P_2, both of which suppress their NAKs after received P_3's NAK. P_1 and

P_2 then generate independent parity packets for P_3. M_2 and M_3 can then be recovered from P_1's and P_2's parities. However none of the three MSs is able to recover M_1. Hence the residual loss after parity recovery is

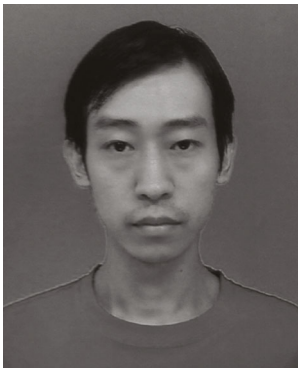
$$\begin{cases} \mathcal{L}_1 = \{M_1\}, \\ \mathcal{L}_2 = \{M_1\}, \\ \mathcal{L}_3 = \{M_1\}, \end{cases} \tag{30}$$

which is clearly better than the loss before cooperative recovery.

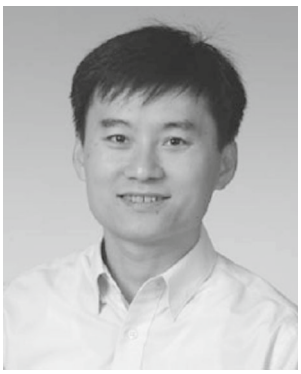
References

1. Digital multimedia broadcasting. http://www.worlddab.org/
2. Digital video broadcasting. http://www.dvb.org/
3. Abdullah NF, Piechocki RJ, Doufexi A (2010) Raptor code for wireless ad hoc vehicular safety broadcast. In: GLOBECOM workshops (GC Wkshps), 2010 IEEE, pp 1087–1091. doi:10.1109/GLOCOMW.2010.5700102
4. Alnuweiri H, Rebai M, Beraldi R (2012) Network-coding based event diffusion for wireless networks using semi-broadcasting. Ad Hoc Netw 10(6):871–885
5. Amer A. B., Gebali F. (2008) Quality of service support and backoff strategies in wireless networks with error control protocol. In: PM2HW2N '08: Proceedings of the 3nd ACM workshop on performance monitoring and measurement of heterogeneous wireless and wired networks, ACM, New York, pp 83–90. doi:10.1145/1454630.1454643
6. Bajic IV (2006) Non-causal error control for wireless video streaming with noncoherent signaling. IEEE Trans Multimed 8(6):1263–1273
7. Bajic IV (2007) Efficient error control for wireless video multicast. IEEE Trans Broadcast 53(1):276–285
8. Chachulski S, Jennings M, Katti S, Katabi D (2007) Trading structure for randomness in wireless opportunistic routing. In: Proceedings of the 2007 conference on applications, technologies, architectures, and protocols for computer communications SIGCOMM '07, ACM, pp 169–180
9. He T, Chan S-H, Wong C-F (2008) HomeMesh: a low-cost indoor wireless mesh for home networking. IEEE Commun Mag 46:79–85
10. Wu X-X, Chan S-H, Mukherjee B, Bhargava B (2004) MADF: mobile-assisted data forwarding in wireless data networks. J Commun Netw 6(3):216–225
11. Chen B, Zhou Z, Zhao Y, Yu H (2012) Efficient error estimating coding: Feasibility and applications. IEEE/ACM Trans Netw 20(1):29–44. doi:10.1109/TNET.2011.2157357
12. Gomez-Barquero D, Aguilella AF, Cardona N (2008) Multicast delivery of file download services in 3G mobile networks with MBMS. In: IEEE international symposium on broadband multimedia systems and broadcasting, pp 1–6. doi:10.1109/ISBMSB.2008.4536640
13. Hartung F, Horn U, Huschke J, Kampmann M, Lohmar T, Lundevall M (2007) Delivery of broadcast services in 3G networks. IEEE Trans Broadcast 53(1):188–199
14. He J, Yang J, An C, Li X (2009) BPR: a bit-level packet recovery in wireless sensor networks. In: SAC '09: Proceedings of the 2009 ACM symposium on applied computing, ACM, New York, pp 59–65. doi:10.1145/1529282.1529293
15. Hou F, Ho PH, Shen X (2006) A novel differentiated retransmission scheme for MPEG video streaming over wireless links. Int J Wirel Mob Comput 1(3):260–267
16. Jamieson K, Balakrishnan H (2007) PPR: Partial packet recovery for wireless networks. In: ACM SIGCOMM, Kyoto
17. Keller L., Le A., Cici B., Seferoglu H., Fragouli C., Markopoulou A. (2012) Microcast: cooperative video streaming on smartphones. In: Proceedings of the 10th international conference on Mobile systems, applications, and services, MobiSys '12, ACM, New York, pp 57–70. doi:10.1145/2307636.2307643
18. Lee D, Song H (2011) A robust luby transform encoding pattern-aware symbol packetization algorithm for video streaming over wireless network. IEEE Trans Multimed 13(4):788–796. doi:10.1109/TMM.2011.2124448
19. Lee JW, Chen CL, Horng MF, Kuo YH (2011) An efficient adaptive FEC algorithm for short-term quality control in wireless networks. In: 13th international conference on advanced communication technology (ICACT), pp 1124–1129
20. Leu JS, Tsai CW, Yi CW (2009) Improving adaptive streaming service across wired&wireless networks. In: Proceedings of the 2009 10th international conference on mobile data management: systems, services and middleware, pp 614–618
21. Li S, Chan SHG (2007) BOPPER: Wireless video broadcasting with peer-to-peer error recovery. In: Proceedings of IEEE international Conference on Multimedia & Expo (ICME), Beijing
22. Li Y, Huang Q, Huang W (2011) A cooperative retransmission strategy for error-prone wireless networks. In: Eighth international conference on wireless and optical communications networks (WOCN), pp 1–5
23. Li Y, Wang Z, You X, Lie Liu Q, Daneshmand M (2010) Error recovery based on FEC in network-layer for intermittently connected mobile networks. In: 5th international ICST conference on communications and networking in China (CHINACOM), pp 1–6
24. Liu X, Cheung G, Chuah CN (2009) Structured network coding and cooperative wireless ad-hoc peer-to-peer repair for WWAN video broadcast. IEEE Trans Multimedia 11(4):730–741. doi:10.1109/TMM.2009.2017636
25. Liu X, Sridharan A, Machiraju S, Seshadri M, Zang H (2008) Experiences in a 3G network: interplay between the wireless channel and applications. In: MobiCom '08: Proceedings of the 14th ACM international conference on mobile computing and networking, ACM, New York, pp 211–222. doi:10.1145/1409944.1409969
26. Nguyen H, Tran LN, Hong EK (2011) On transmission efficiency for wireless broadcast using network coding and fountain codes. IEEE Commun Lett 15(5):569–571
27. Rozner E, Iyer AP, Mehta Y, Qiu L, Jafry M (2007) ER: efficient retransmission scheme for wireless lans. In: Proceedings of the 2007 ACM CoNEXT conference, CoNEXT '07, ACM, pp 8:1–8:12
28. Schier M, Welzl M (2012) Optimizing selective ARQ for H.264 live streaming: A novel method for predicting loss-impact in real time. Multimed IEEE Trans 14(2):415–430. doi:10.1109/TMM.2011.2178235
29. Sen S, Schmitt S, Donahue M, Banerjee S (2009) Exploiting "approximate communication" for mobile media applications. In: HotMobile '09: Proceedings of the 10th workshop on mobile computing systems and applications, ACM, New York, pp 1–6. doi:10.1145/1514411.1514420
30. Sinkar K, Jagirdar A, Korakis T, Liu H, Mathur S, Panwar S (2008) Cooperative recovery in heterogeneous mobile networks. In: 5th annual IEEE communications society conference on sensor, mesh and Ad Hoc communications and networks, SECON '08, pp 395–403. doi:10.1109/SAHCN.2008.55

31. Wu H, Zheng J (2011) CoRET: a network coding based multicast retransmission scheme for mobile communication networks. In: IEEE international conference on communications (ICC), pp 1–5. doi:10.1109/icc.2011.5962911
32. Yuan X, Sun R, Ping L (2010) Simple capacity-achieving ensembles of rateless erasure-correcting codes. IEEE Trans Commun 58(1):110–117. doi:10.1109/TCOMM.2010.01.050175



Bo Zhang received his B.Sc.degree in computer science from the Southampton University, UK in 2006, and the M.Sc. degree in data communication networks and distributed systems from University College London in 2007. He is currently pursuing the Ph.D. degree at the Department of Computer Science and Engineering in HKUST, supervised by Prof. S.-H. G. Chan. His research interest includes wireless video live broadcasting technologies, multimedia networking, overlay, and wireless peer-to-peer networks.



Dr. S.-H. Gary Chan received MSE and PhD degrees in Electrical Engineering from Stanford University (Stanford, CA) in 1994 and 1999, respectively, with a minor in business administration. He obtained his B.S.E. degree (highest honor) in Electrical Engineering from Princeton University (Princeton, NJ) in 1993, with certificates in Applied and Computational Mathematics, Engineering Physics, and Engineering and Management Systems. He is currently an Associate Professor of the Department of Computer Science and Engineering, Director of Sino Software Research Institute, and Co-director of Risk Management and Business Intelligence program, The Hong Kong University of Science and Technology (HKUST), Hong Kong. His research interest includes multimedia networking, overlay streaming and technologies, and wireless communication networks.

Dr. Chan is a member of Tau Beta Pi, Sigma Xi, and Phi Beta Kappa. He has been an Associate Editor of IEEE Transactions on Multimedia (2006-11), and is a Vice-Chair of Peer-to-Peer Networking and Communications Technical Sub-Committee of IEEE Comsoc Emerging Technologies Committee. He has been Guest Editors of IEEE Transactions on Multimedia (2011), IEEE Signal Processing Magazine (2011), IEEE Communication Magazine (2007), and Springer Multimedia Tools and Applications (2007). He was the TPC chair of IEEE Consumer Communications and Networking Conference (CCNC) 2010, Multimedia symposium in IEEE Globecom (2007 and 2006) and IEEE ICC (2007 and 2005), and Workshop on Advances in Peer-to-Peer Multimedia Streaming in ACM Multimedia Conference (2005).

Dr. Chan is a member of Tau Beta Pi, Sigma Xi, and Phi Beta Kappa. He has been an Associate Editor of IEEE Transactions on Multimedia (2006-11), and is a Vice-Chair of Peer-to-Peer Networking and Communications Technical Sub-Committee of IEEE Comsoc Emerging Technologies Committee. He has been Guest Editors of IEEE Transactions on Multimedia (2011), IEEE Signal Processing Magazine (2011), IEEE Communication Magazine (2007), and Springer Multimedia Tools and Applications (2007). He was the TPC chair of IEEE Consumer Communications and Networking Conference (CCNC) 2010, Multimedia symposium in IEEE Globecom (2007 and 2006) and IEEE ICC (2007 and 2005), and Workshop on Advances in Peerto- Peer Multimedia Streaming in ACM Multimedia Conference (2005).

Dr. Gene Cheung received the B.S. degree in electrical engineering from Cornell University, Ithaca, NY, in 1995, and the M.S. and Ph.D. degrees in electrical engineering and computer science from the University of California, Berkeley, in 1998 and 2000, respectively. From 2000 to 2009, he was a Senior Researcher with Hewlett-Packard Laboratories Japan, Tokyo, Japan. He is currently an Assistant Professor with the National Institute of Informatics, Tokyo. He has published over 15 international journals and 70 conference publications. His research interests include media representation and network transport, single-/multiple-view video coding and streaming, and immersive communication and interaction.

Dr. Cheung has been serving as the Associate Editor of the IEEE TRANSACTIONS ON MULTIMEDIA since 2007 and as the Associate Editor of the Digital Signal Processing Applications Column in the IEEE Signal Processing Magazine since 2011. He has also served as the Area Chair in the IEEE International Conference on Image Processing 2010 and the Technical Program Cochair of the International Packet Video Workshop 2010. He serves as the Track Cochair for the Multimedia Signal Processing track in the IEEE International Conference on Multimedia and Expo 2011. He was a corecipient of the Top 10% Paper Award in the IEEE International Workshop on Multimedia Signal Processing 2009.