Pollution-resilient peer-to-peer video streaming with Band Codes

POLLUTION-RESILIENT PEER-TO-PEER VIDEO STREAMING WITH BAND CODES

Attilio Fiandrotti

Sisvel Technology, Torino, Italy
attilio.fiandrotti@sisveltechnlogy.com

Rossano Gaeta, Marco Grangetto

Dipartimento di Informatica, Università di Torino
rossano@unito.it, marco.grangetto@unito.it

ABSTRACT

Band Codes (BC) have been recently proposed as a solution for controlled-complexity random Network Coding (NC) in mobile applications, where energy consumption is a major concern. In this paper, we investigate the potential of BC in a peer-to-peer video streaming scenario where malicious and honest nodes coexists. Malicious nodes launch the so called *pollution attack* by randomly modifying the content of the coded packets they forward to downstream nodes, preventing honest nodes from correctly recovering the video stream. Whereas in much of the related literature this type of attack is addressed by identifying and isolating the malicious nodes, in this work we propose to address it by adaptively adjusting the coding scheme so to introduce resilience against pollution propagation. We experimentally show the impact of a pollution attack in a defenseless system and in a system where the coding parameters of BC are adaptively modulated following the discovery of polluted packets in the network. We observe that just by tuning the coding parameters, it is possible to reduce the impact of a pollution attack and restore the quality of the video communication.

Index Terms— Peer-to-peer, Pollution, Network Coding, Video streaming

1. INTRODUCTION

Peer-to-Peer (P2P) video streaming is an effective way to distribute bandwidth intensive multimedia contents such as live video streams to large populations of cooperating users [1, 2]. Network Coding (NC) has shown to be a promising solution to increase the effective network throughput and to workaround typical problems with traditional P2P architectures such as the rarest-piece issue [3, 4].In random NC-based media streaming, a content is organized in independently encoded and decodable media units called *generations* and each generation is further partitioned in blocks of symbols of identical size. A source node holds the original video content and, for each generation, it transmits random linear combinations of the blocks to the nodes as network packets. The network nodes relay random linear combinations of the received packets and, once they have collected enough suitable packets, they recover the generation solving a system of equations.

NC architectures are however vulnerable to *pollution* attacks where one or more *malicious* nodes purposely transmit bogus packets to the network with the goal to disrupt the communication. Because the packets received by the nodes carry encoded payloads, there is no easy way for an honest node to tell whether a received packet is *clean* (i.e., it contains a valid linear combination of the original symbols and can be safely relayed to the other nodes) or it is *polluted* and thus it should be discarded. Whenever an honest node draws for recombination one or more polluted packets, also the recombined packet it transmits is polluted, which contributes to the further spreading the pollution over the network.

Most pollution detection and avoidance schemes rely on a two-pillars approach: pollution detection and malicious nodes isolation. First, honest nodes must be able to understand if there is an ongoing pollution attack. Because the nodes exchange coded packets, waiting to recover the original content to detect a pollution attack involves a delay during which the pollution may have already spread beyond recovery. Second, honest nodes must (cooperatively) identify the malicious nodes and isolate them from the network, e.g. via blacklisting [5, 6, 7, 8, 9]. However, discovery of a pollution attack and in particular the identification of the malicious nodes are challenging problems which require the allocation of substantial computational and network resources. For this reason, pollution avoidance mechanisms that enable the nodes to early detect an ongoing pollution attack and react with minimum computational and communication efforts are sought. In this work, we explore the possibility to build a random NC scheme which is resilient-by-design to pollution attacks by exploiting the properties of a family of low-complexity codes known as Band Codes (BC) [10]. BC are designed to control the decoding complexity of NC, i.e. the number of operations a node must perform to recover a generation, by constraining the coding operations at the source and at the network nodes to a subset of the original blocks or received packets which are drawn for recombination. We show in this work that BC can be turned into an effective countermeasure to pollution attacks as well, by:

- exploiting the packet decoding mechanism to spot the

This work was partially supported by Universit degli Studi di Torino and Compagnia di San Paolo under project AMALFI (ORTO119W8J)

presence of polluted packets in the nodes input buffer prior to whole generation recovery; this early detection mechanism allows nodes to coordinate and take proper actions against the ongoing attack;
- adaptively (after pollution detection) modifying the coding scheme so to minimize the likelihood that any of the polluted packets is drawn for recombination.

Code-based pollution countermeasures have proposed before. For example, [11, 12] proposed cryptographic and algebraic verification methods respectively, which entail however increased computational cost and communication overhead for data verification and ground truth pre-sharing respectively. In [13] some degree of redundancy is introduced for correction purposes, however this approach can cope with a limited amount of corruption only. Conversely, our BC-based pollution avoidance architecture does not entails additional computational (no additional computations are required to detect the presence of polluted packets in anode input buffer) or communication costs (each node operates autonomously, without inter-peer signaling), whereas it enjoys the low-complexity features of BC.

2. BAND CODES

In this section, we provide an high level overview of Band Codes (BC), which are thoroughly described in [10], whose notation we borrow in the rest of this paper.

2.1. Packet Encoding at the Source

The source node holds the original video content, which is organized in chunks of data of identical size called *generations*. Let x be a generation, which is further divided in k blocks of symbols $(x_1, ..., x_k)$, where k is the *generation size* as illustrated in Fig. 1. When the source transmits a packet, it produces a linear combination of the input symbols computed as $y = \sum_{i=1}^{k} g_i x_i$, where the summation operator is used to denote the bit-wise XOR (we recall that band codes enable NC over $GF(2)$). The vector $g = (g_1, ..., g_k)$, $g_i \in \{0, 1\}$, is the *encoding vector* and determines which blocks are going to be encoded by the source. The number of elements of g equal to one is known as the *degree* of the encoded packet. In classic random NC, g is typically drawn so that $\mathcal{P}\{g_i = 1\} = \frac{1}{2}\ \forall i$, which maximizes the probability that the encoded packet is *innovative* at the receiver (i.e., linearly independent from previously received packets), but also drives high the decoding complexity. Conversely, in BC-based NC first the source draws an *encoding window* of size $w \leq k$, i.e. a subset $(g_f, \ldots, g_l)$, with $f \leq l, l - f + 1 = w$, of the encoding vector from an ad-hoc distribution. Next, each element of the encoding vector is independently drawn from a binary discrete distribution so that $\mathcal{P}\{g_i = 1\} = \frac{1}{2}$ if $f \leq i \leq l$, $g_i = 0$ otherwise. It can be proven that the *degree* of the packets encoded as above follows a binomial distribution $\mathcal{B}(w, \frac{1}{2})$, which is the first step in controlling the decoding complexity at the network nodes. Finally, the source transmits to the network a packet that contains the encoded payload y plus the encoding vector g that we indicate in the following as $P(g, y)$.

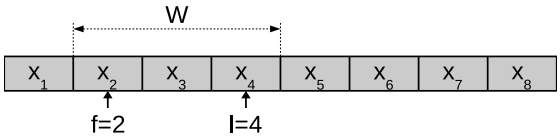

Fig. 1. Generation of size k=8 and encoding window of size W=3, with leading and trailing edges f=2 and l=4.

2.2. Packet Decoding

Network nodes receive encoded packets that are decoded with an early Gaussian Elimination-like algorithm which processes each received packet $P(g, y)$ and solves a system of k linear equations $GX = Y$, G being the $k \times k$ matrix holding the encoding vector g of the received packets and Y being the vector holding the encoded payload y. Let G_i indicate the i-th row of G and $G_{i,j}$ the element of G at row i, column j: if $G_i = 0,\ \forall\ i$, the row i-th is empty and we write $G_i = \emptyset$. For the sake of simplicity, we describe the operations on the G matrix and the received encoding vectors g and we omit the equivalent operations on y and Y. Let g_s be the leading one of g, i.e. the first element of g s.t. $g_i \neq 0$. If $G_s = \emptyset$, g is inserted in the s-th row of G, y at the s-th position of Y and the algorithm ends. If G_s is not empty and if $g = G_s$, the received packet is not innovative and the algorithm ends, otherwise a XOR between g and G_s and between y and Y_s is executed and the algorithm iterates. The algorithm arranges the encoding vectors of the received packets in an upper-triangular *band* matrix, i.e. all elements outside a band of width w are guaranteed to be null. In practical cases it takes $k' > k$ packets to recover a generation because not all received packets are innovative. The penalty $\epsilon_c = \frac{k'-k}{k}$ is usually termed as *code overhead* and corresponds to the ratio of network bandwidth wasted transmitting non innovative, hence useless, packets. Once the rank of G is equal to k, the matrix is diagonalized by backward-substitution and $Y = (x_1, \ldots, x_k)$, i.e. Y contains the original k symbols that compose the recovered generation.

2.3. Packet Recombination

In traditional RNC, at each transmission opportunity the network nodes recombine each of the received packets with identical probability $p_r = \frac{1}{2}$. However, totally random recombinations at the nodes alter the packet degree distribution imposed by the source, which converges to $\mathcal{B}(k, \frac{1}{2})$, bloating the decoding complexity [10]. BC hence rely on a packet recombination scheme which preserves the degree distribution imposed by the source and keeps the decoding complexity bounded. First, the network nodes recombine and trans-

mit the rows of the G matrix (and the encoded payloads in the Y vector) rather than the received packets. Rows of G are indeed linearly independent combinations of the original generation blocks $(x_1, \ldots, x_k)$, so linear combinations of G rows are linear combinations of the original blocks. At each transmission opportunity, the node draws an encoding window of size W with a scheme similar to that used by the source. Let $c = (c_1, \ldots, c_k), c_i \ \{0, 1\}$ be the encoding vector of the rows of G: we set $c_i = 0$ for any rows G_i such that at least one element equal to one falls outside the drawn window, otherwise we draw the elements of c such that $\mathcal{P}\{c_i = 1\} = \frac{1}{2}$. Finally, the recombined packet $P^r(g^r, y^r)$ is such that $g^r = \sum_{i=1}^{k} c_i G_i$ and $y^r = \sum_{i=1}^{k} c_i Y_i$. It turns out that the degree distribution of packets recombined according to such scheme is still the binomial distribution $\mathcal{B}(W, \frac{1}{2})$, i.e. the packet recombination preserves the degree distribution and thus the decoding complexity.

In [10] the parameter $W \leq k$ is adjusted to trade coding overhead for computational complexity: in this paper we will leverage on the same parameter to limit pollution spreading as described in Section 3.

3. PROPOSED ARCHITECTURE

First, in this section we model a typical pollution attack consisting in the injection of bogus packets into the network which are then recombined with clean packets in the network by unaware nodes . Next, we show how the decoding scheme described the previous section can be exploited as a pollution detection mechanism, allowing a node to spot the presence of a polluted packet among those it received. Next, we propose a pollution avoidance scheme where a node adaptively adjusts its own coding window size whenever it detects a polluted packet or receives a pollution warning from a neighbor.

3.1. Pollution Attack Model

First of all we define what kind of pollution attack a malicious node can practically attempt. The goal of a malicious node is to prevent other nodes in the overlay from correctly decoding a generation; to this end, the malicious node can create a fake linear combination $y^p = \sum_{i=1}^{k} g_i^p x_i^p$, where $g_i^p \neq g_i$ and/or $x_i^p \neq x_i$ and then upload a polluted packet $P(g, y^p)$ with $g = (g_1, ..., g_k)$. In other words, the malicious node transmits a packet whose payload y^p is not in agreement with the signaled encoding vector by randomly altering the encoding vector or the coded payload.Altering the coded payload to get y^p is a safe option for the malicious node since any receiving node that has not yet decoded the generation ignores the original information x_i and has no means to discriminate between polluted and non polluted packets. On the contrary, altering the encoding vector can be dangerous since all nodes in the networks know the statistical properties of the encoding vectors (window size and degree distribution) and can potentially use this information to identify the polluters. Hence, in the following we assume a pollution attack model where, at each transmission opportunity, a malicious node decides with probability p_{poll} whether to pollute the forwarded packet, which is polluted by randomly changing the coded payload and leaving the encoding vector unaffected.

Second, we qualitatively show why a malicious node can carry on an effective attack by altering only a few transmitted packets. Every time an (honest) node draws for recombination a subset of the received packets (or G rows, as in BC), it is sufficient that just one of the recombined equations is polluted for the relayed packet to be polluted as well. Random recombinations are hence the key factor which multiplies the disruptive effect of the injection of a handful of polluted packets in the network. We define as *pollution overhead* the ratio $\epsilon_p = \frac{r_p}{k}$ of a node output bandwidth wasted transmitting polluted packets. We experimentally show later on that a malicious node needs to alter only a small share of the packets it forwards to significantly increase the honest nodes pollution overhead and disrupting the service quality.

3.2. Pollution Detection

Now, we analyze the effects of the reception of a packet $P(g, y)$ at a network node on the decoding algorithm described in Section 2.2, which has the 2 possible outcomes in Fig. 2.

- $P(g, y)$ is such that its encoding vector g can be inserted in empty row G_s of the decoding matrix G and the payload y is stored in Y_s (we recall that s is the position of the first non-null element of g): in this case the node has no evidence of an ongoing pollution attack.
- $P(g, y)$ in such that $G_s \neq \emptyset$ and $g \neq G$: an XOR between (g, y) and (G_s, Y_s) is computed and the algorithm iterates on the result.
- $P(g, y)$ in such that $G_s \neq \emptyset$ and $g = G$ and $y = Y_s$: the received packet is not innovative and there is no evidence of an ongoing attack.
- $P(g, y)$ in such that $G_s \neq \emptyset$ and $g = G$ but $y \neq Y_s$: the two encoding vectors match but the two encoded payloads do not, thus at least one of the packets received so far is polluted.

The first point of the previous list shows that a polluted packet (g, y) could be inserted in (G_s, y_s) without the node knowing the packet is polluted: since new packets may be combined with rows of G (third point), pollution potentially propagates to clean packets as well when the algorithm iterates. On the other hand, the last point in the list also carries a good news, i.e. the possibility for a node to unveil inconsistency among the received equations thus permitting to detect pollution on the fly and without the need of an external verification mechanism. In the following we will refer to this possibility as early pollution detection: whenever $g = G_s$ but $y \neq Y_s$, then the generation is flagged as polluted.

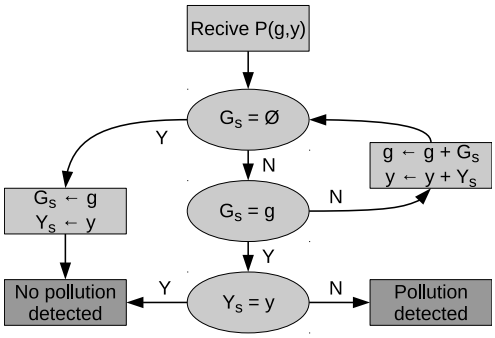

Fig. 2. The proposed packet pollution detection mechanism is invoked each time a packet $P(g, y)$ is received. If $g = G_s$ while $y \neq Y_s$, at least one of the packets received is polluted.

3.3. Pollution Avoidance and Warning

As soon as a honest node realizes that at least one of the received packets is polluted, it immediately broadcasts a pollution warning to all its neighbor peers. Whenever a transmission opportunity arises, the node checks if it has detected a pollution attack or has received a pollution warning during the previous B_w seconds, which we define as the node backoff window. If no pollution attack is reported, the node simply draws a recombination window of size $W = k$, i.e. all the rows of G are drawn for recombination, which yields maximum coding efficiency; otherwise, the node restricts the set of G rows to draw for recombination to a random window of size $W < k$. By constraining the recombinations to a subset of the G rows, the probability that one (or more) polluted equations are drawn for recombination decreases, reducing the likelihood that an honest node relays a polluted packet. Whereas we do not provide an analytic analysis of the gains entailed by such scheme, we experimentally prove later on that it significantly reduces the pollution overhead while preserving low-complexity decoding.

4. EXPERIMENTAL EVALUATION

In this section, first we describe the push-based P2P architecture that we use to distribute a video stream to a multitude of cooperating peers, next we describe the experimental testbed we have set up for performing our experiments, and finally we report the results of our experiments on said testbed.

4.1. Random-Push based Live Video Streaming

For our experiments, we consider a typical random-push P2P network where the peers are arranged into an unstructured, non-acyclic, mesh overlay. Mesh topologies are challenging when it comes to pollution detection, as there is no simple way for a node to restrict the source of the pollution to one or few ancestor in the overlay hierarchy as in tree-like schemes. A central tracker manages the overlay, answering join requests from nodes that want to enter the network and running a master list of all the peers in the network. Each peer exchanges coded packet with a subset of the network nodes known as neighbors: periodically, each node gracefully disconnects from a few neighbors and estabilishes new neighborhood relationships to account for the peer churning typical of P2P networks. In our experiments, we constrain the size of each peer neighborhood not to exceed N_s peers, so to keep the overlay sparse and minimize the signaling overhead. A server peer holds a test video stream encoded at constant bit rate B_v and subdivides it in a sequence of independently encoded generations of similar playout duration and number k of blocks. The server node disseminates encoded packets randomly among the nodes in the neighborhood. The peers are periodically allocated transmission opportunities and at each opportunity a peer randomly draws one of its neighbors and transmits it a recombined packet for the generation that is closer to its playout deadline and has not been recovered yet. Each peer informs its neighbors about which generations it has already decoded and which not with appropriate signaling, and every time a peer detects a polluted packet in its input buffer, it broadcasts a pollution warning to its neighborhood.

4.2. Experimental Setup

The experiment consists in distributing a 10 minutes long H.264/AVC video sequence encoded at C_v = 500 kbit/s by a source node whose output bandwidth is equal to B_s = 20 Mbit/s and the output bandwidth of the peer nodes is constrained to 1 Mbit/s. The peers population amounts to N=1000 nodes, with N_h=980 honest and N_m=20 malicious nodes, respectively; the nodes neighborhood size is constrained to N_s=25 peers. The source node and the N_h honest peers join the streaming session at time t = 0 s and leave the session at time t = 600 s. Malicious nodes join at time t=210 and leave at time t=450 s, i.e. they operate for a 240 s interval that in the following we refer to as *pollution interval*. During such interval, malicious nodes purposely alter the payload of each transmitted packet with probability p_{poll} = 0.01. We consider a generation successfully recovered by a node if enough innovative, non polluted, packets are received prior to the generation decoding deadline. We measure the quality of the video received by the honest nodes in terms of Continuity Index (CI), which corresponds to the average fraction of generations successfully recovered in the network by an honest node. We mainly measure the network utilization efficiency in terms of pollution overhead ϵ_p, defined in Section 2 as the fraction of a node output bandwidth wasted transmitting polluted packets. We experiment with generations of k = 25 blocks, which yield generations of 0.5 seconds considering blocks of data of size C_s = 10 kbit. We consider two different NC coding strategies. In the following, we indicate as *Reference* a totally random NC strategy where the source node encodes random combinations of the original blocks and the

network nodes forward random combinations of the received packets. The *Proposed* strategy is the BC-based coding strategy described in Section 2, where all coding operations are constrained within a coding window of width $W = \frac{k}{2}$, which was shown to halve the decoding complexity while maintaining reasonable encoding overhead.

4.3. Results

To start with, we investigate the effect of the the pollution attack, focusing on the effect of the arrival and the departure of the malicious nodes in and from the network. Fig. 3 shows the pollution overhead ϵ_p and the CI as a function of the streaming time for the *reference* coding scheme (the dashed horizontal asymptotes represent the mean values over the pollution interval). During the interval 0 - 210 s, only the 980 honest nodes are in the network, ϵ_p is null and the CI is equal to 1, i.e. all nodes recover the video entirely. At time t=210 s, 20 malicious nodes enter the network and start trasnmitting polluted packets. Each malicious node randomly alters the payload of 1% of the transmitted packets and malicious nodes represent 2% of the peers, thus only 0.02% of the overall network traffic is directly polluted by the malicious nodes. However, the figure shows that the average pollution overhead is about 20%, i.e. random recombinations at the nodes increased the pollution overhead by a 1000-fold factor. As a result, the network nodes are unable to correctly recover most of the generations, and the CI sinks to about 25%, i.e. only one generation out of four is correctly recovered. This experiment shows that totally random combinations at the nodes quickly spread the pollution and malicious nodes need to inject only a handful of polluted packets to completely disrupt the communication.

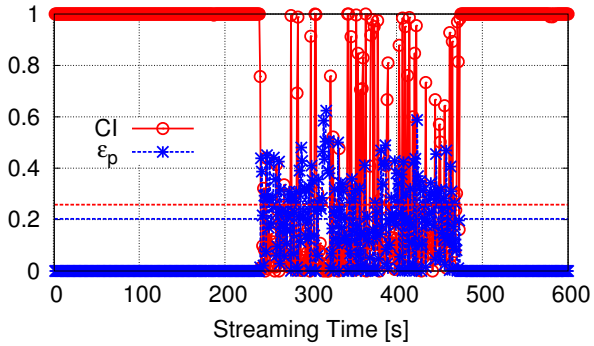

Fig. 3. Video quality and pollution overhead ϵ_p for a totally random NC scheme: random recombinations spread the pollution disrupting the video communication.

Next, we experiment with our BC-based coding scheme with an adaptive coding window size W and a backoff window B_w equal to 1 s. In the beginning, all nodes operate with a coding window of size $W = k$, i.e. as the reference NC scheme in Fig. 3. However, as soon as one of the nodes detects a polluted packet in its input buffer or receives a pollution warning from a neighbor, it starts recombining the transmitted packets with a reduced window $W = \frac{k}{2}$. Finally, after B_w seconds the node has not detected any polluted packet and has not received any pollution warnings, it starts recombining with a full size window $W = k$ again. Fig. 4 shows the results of the experiment: by comparing what happens in the pollution interval with Fig. 3, we see that the average pollution overhead has dropped from 20% to about 1% (light green dashed line). That is, by simply recombining packets from a narrower window W whenever a pollution attack is detected, the pollution overhead drops by 10 times with respect to the reference scheme and the CI increases from 0.25 to 0.8.

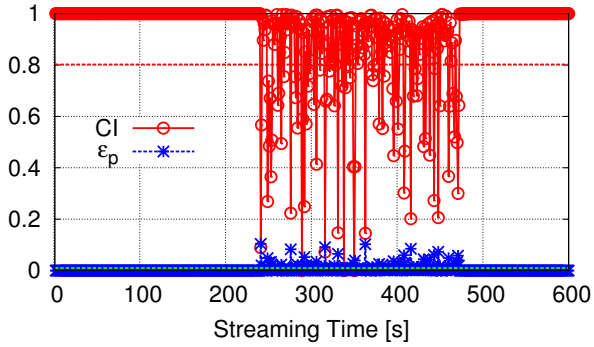

Fig. 4. Video quality and pollution overhead ϵ_p for our pollution-adaptive coding scheme: constraining packet recombinations to a smaller W constrains the pollution spreading and improves video quality.

Next, we explore the effect of the backoff window size B_w on the quality of the received video and the efficiency of the coding scheme. Fig. 5 shows the CI, the pollution and the coding overhead when the backoff window B_w varies from 0 to 5 s. When B_w=0, our BC-based coding scheme operates as the reference scheme in 3: the pollution overhead is about 20% and the CI is about 0.24. As B_w increases, the pollution overhead drops from 20% to 2% and the CI soars to about 0.8 for B_w = 1 s. As B_w increases over 1 s, the nodes tend to recombine packets on a smaller window W for longer time compromising the average code efficiency. Because in our experiments the output bandwidth of the nodes is constrained to 1 Mbit/s, there is not enough bandwidth to allow all nodes to collect enough packets and the CI drops. This experiment showed that increasing the backoff window beyond a certain threshold does not increase the video quality any further: thus in the rest of the experiments we keep B_w = 1 s.

Finally, in Fig. 6 we increase the number of malicious nodes N_m in the network up to 40 (we recall that in all previous experiments, we had N_m = 20) for the reference and proposed recombinations schemes. The pollution overhead increases linearly with N_m, however our proposed strategy achieves a pollution overhead of just 1.6% when N_m=40,

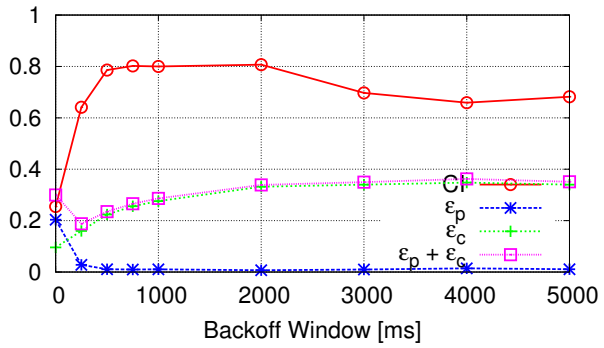

Fig. 5. Video quality, code overhead ϵ_c and pollution overhead ϵ_p for our pollution-adaptive coding scheme.

whereas with the reference strategy almost 35% of the packets transmitted in the network are polluted. The figure clearly shows that when the number of polluters doubles, our strategy enables a graceful degradation of the video quality; conversely, the reference strategy cannot cope with the increased pollution rate and the CI sinks to 15%.

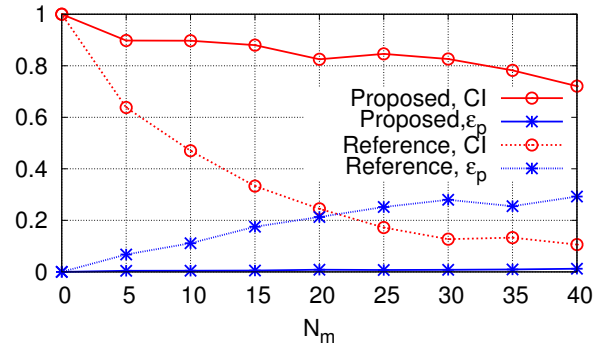

Fig. 6. Video quality and pollution overhead as the number of malicious nodes in the network N_m increases for the two considered coding schemes.

5. CONCLUSIONS AND FUTURE WORKS

This work shows how it is possible to gain resilience against pollution attacks in random NC-based video streaming by appropriately controlling the packet recombination strategy at the network nodes without the need of complex countermeasures such as malicious nodes identification. In detail, we exploit the properties of BC, a family of codes for controlled-complexity NC, to adjust the packet recombination scheme on the fly, easing the propagation of the pollution through the network. Experiments in a realistic video streaming scenario show a ten-fold reduction in the polluted traffic rate and a large improvement in the resulting video quality. These findings prove the potential of the packet recombination scheme in mitigating the effect of pollution attacks, prompting the potentials for a packet recombination scheme that is designed from scratch to be inherently resilient to pollution attacks.

6. REFERENCES

[1] X. Zhang, J. Liu, B. Li, and T.S.P. Yum, "CoolStreaming/DONet: A data-driven overlay network for efficient live media streaming," in *proceedings of IEEE Infocom*. Citeseer, 2005, vol. 3, pp. 13–17.

[2] G Huang, "PPLive: A practical P2P live system with huge amount of users," in *Proceedings of the ACM SIGCOMM Workshop on Peer-to-Peer Streaming and IPTV Workshop*, 2007.

[3] M. Grangetto, R. Gaeta, and M. Sereno, "Rateless codes network coding for simple and efficient P2P video streaming," in *IEEE International Conference on Multimedia and Expo, 2009 (ICME 2009)*.

[4] M. Wang and B. Li, "Network coding in live peer-to-peer streaming," *IEEE Transactions on Multimedia*, vol. 9, no. 8, pp. 1554–1567, Dec 2007.

[5] Q. Wang, L. Vu, K. Nahrstedt, and H. Khurana, "MIS: Malicious nodes identification scheme in network-coding-based peer-to-peer streaming," in *INFOCOM, 2010 Proceedings IEEE*, march 2010, pp. 1 –5.

[6] Y. Li and J.C.S. Lui, "Stochastic analysis of a randomized detection algorithm for pollution attack in P2P live streaming systems," *Performance Evaluation*, vol. 67, no. 11, pp. 1273 – 1288, 2010.

[7] X. Jin and S.H.G. Chan, "Detecting malicious nodes in peer-to-peer streaming by peer-based monitoring," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 6, pp. 9:1–9:18, March 2010.

[8] R. Gaeta, M. Grangetto, and L. Bovio, "DIP: Distributed identification of polluters in P2P live streaming," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMCCAP)*, vol. 10, no. 3, pp. 24, 2014.

[9] R. Gaeta and M. Grangetto, "Identification of malicious nodes in peer-to-peer streaming: A belief propagation-based technique," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 24, no. 10, pp. 1994–2003, 2013.

[10] A. Fiandrotti, V. Bioglio, M. Grangetto, R Gaeta, and E. Magli, "Band codes for energy-efficient network coding with application to P2P mobile streaming," *IEEE Transactions on Multimedia*, vol. 16, no. 2, pp. 521 – 532, February 2014.

[11] M. N. Krohn, M. J. Freedman, and D. Mazieres, "On-the-fly verification of rateless erasure codes for efficient content distribution," *Security and Privacy, IEEE Symposium on*, 2004.

[12] C. Gkantsidis and P. Rodriguez, "Cooperative security for network coding file distribution," in *IEEE INFOCOM*, 2006.

[13] T. Ho, B. Leong, R. Koetter, M. Medard, M. Effros, and D.R. Karger, "Byzantine modification detection in multicast networks with random network coding," *Information Theory, IEEE Transactions on*, vol. 54, no. 6, pp. 2798 –2803, june 2008.