Selection of Network Coding Nodes for Minimal Playback Delay in Streaming Overlays

Nicolae Cleju, Nikolaos Thomos, *Member, IEEE*, and Pascal Frossard, *Senior Member, IEEE*

***Abstract*—Network coding permits to deploy distributed packet delivery algorithms that locally adapt to the network availability in media streaming applications. However, it may also increase delay and computational complexity if it is not implemented efficiently. We address here the effective placement of a limited number of nodes that implement randomized network coding in overlay networks, so that the goodput is kept high while the delay for decoding stays small in streaming applications. We first estimate the decoding delay at each client, which depends on the innovative rate in the network. This estimation permits to identify the nodes that have to perform coding in order to reduce the decoding delay. We then propose two iterative algorithms for selecting the nodes that should perform network coding. The first algorithm relies on the knowledge of the full network statistics. The second algorithm uses only local network statistics at each node. Simulation results show that large performance gains can be achieved with the selection of only a few network coding nodes. Moreover, the second algorithm performs very closely to the central estimation strategy, which demonstrates that the network coding nodes can be selected efficiently with help of a distributed innovative flow rate estimation solution. Our solution provides large gains in terms of throughput, delay, and video quality in realistic overlay networks when compared to methods that employ traditional streaming strategies as well as random network coding nodes selection algorithms.**

***Index Terms*—Delay minimization, network coding, overlay networks, throughput maximization.**

I. Introduction

THE recent development of overlay networks offers interesting perspectives for multimedia streaming applications, since network diversity can be used advantageously for improved quality of service. The traditional streaming systems based on ARQ or channel coding techniques however generally fail to efficiently exploit this diversity. They either suffer from relatively high computational cost, require coordination between network nodes, or lead to suboptimal performance in large scale networks where local channel conditions are hard to estimate. A different paradigm has been initiated recently with network coding [1], [2], where some processing is requested from the network nodes in order to improve the packet delivery performance. Specifically, network coding nodes combine buffered packets before forwarding them to next hop nodes. This coding strategy is particularly appealing in distributed streaming systems, as it removes the need for reconciliation between nodes. It locally adapts to the available bandwidth and packet loss rate and even permits to approach the max-flow min-cut bound of the network graph. Overall, the network coding systems have shown improved resiliency to dynamics, delays, scalability, and buffer capacities in networks with diversity [3].

The application of network coding algorithms in multimedia streaming systems is however not straightforward. Specifically, multimedia streaming imposes strict timing constraints that impact the design of network coding algorithms. A practical network coding system has been presented in [4] and addresses the specific characteristics of streaming applications. It implements randomized network coding (RNC) techniques [5] in the network nodes and devises a protocol to deal with buffering issues and timing constraints. Moreover, it introduces the concept of generations that restricts coding operations to packets that share similar decoding deadlines. However, network coding systems still face important issues in practical systems due to the decoding delays imposed by successive network coding operations. This delay as well as the computational overhead in the system grow with the number of network coding nodes. It becomes therefore important to select efficiently the subset of nodes that perform network coding in order to control delay and complexity and still exploit efficiently the diversity in the network.

In this paper, we discuss solutions for the selective placement of a few network coding nodes in order to reduce the delay for video delivery. We assume that the number of network coding nodes is given beforehand, based on computational complexity or decoding delay constraints.[1] The nodes in the network are categorized into *network coding* (NC) and *store and forward* (SF) nodes. The NC nodes use the practical network coding algorithm described in [4], which has been selected for its effectiveness and simplicity. Similarly, we adopt the concept of coding generation and buffer models [4] for proper handling of the timing constraints in the stream delivery. We extend our previous work [7] that estimates the rate of non-redundant packets in each network node. This rate is an indication of the goodput of the system as it measures the number of useful and non-redundant packets that are received at a node. It permits to estimate the

Manuscript received January 18, 2011; revised April 28, 2011; accepted June 18, 2011. Date of publication July 12, 2011; date of current version September 16, 2011. This work was supported in part by the Swiss National Science Foundation under grant PZ00P2-121906. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Yiannis Andreopoulos.

N. Cleju is with the "Gheorghe Asachi" Technical University of Iasi, Iasi 700506, Romania (e-mail: nikcleju@etti.tuiasi.ro).

N. Thomos and P. Frossard are with the Ecole Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland, and also with the Signal Processing Laboratory (LTS4), Lausanne, Switzerland (e-mail: nikolaos.thomos@epfl.ch; pascal.frossard@epfl.ch).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TMM.2011.2161448

[1]The problem of finding the optimal number of network coding nodes for delay-minimal video delivery is actually an NP-hard problem [6]

decoding delay in the client nodes; this corresponds to the time necessary for collecting enough useful packets to build a full rank decoding system. Contrarily to [7], we directly estimate the delay for selecting the best subset of nodes that shall implement network coding. A direct estimation of the delay leads to better solution than the estimation of the innovative rate. While both problems are dual in the case of a single client, the minimization of the delay or the maximization of the goodput are not exactly equivalent in multicast applications.

We propose two algorithms that iteratively choose the SF nodes to be turned into NC nodes for improving the system performance until the predefined number of network coding nodes has been reached. Both algorithms differ in their view of the network status. The first algorithm assumes that a central node has complete knowledge about the status of the overlay network in terms of available bandwidth and packet loss rate. The second algorithm only uses local estimations of the network status at each node and provides a solution for distributed systems. The simulation results show that the proper selection of only a few network coding nodes already leads to throughput gains that come close to the max-flow min-cut bound and greatly decrease the delay necessary for media stream delivery. Moreover, the algorithm that only considers local network statistics performs very competitively with the algorithm that uses full knowledge of the network topology. Both algorithms even select the same nodes for network coding in most of the cases. Furthermore, they both outperform basic streaming algorithms built on store and forward approaches as well as solutions where network coding nodes are selected randomly. These observations are confirmed in realistic overlay networks where our method can improve users' video experience even in the case where only few nodes implement randomized network coding. This is due to a good balance between decoding delay and efficient use of the network diversity. Finally, minimal network knowledge is often sufficient for determining the efficient positioning of the network coding nodes.

The paper is organized as follows. In Section II, we present the framework under consideration and briefly overview the network coding principles. In Section III, we describe the model of the SF node buffer that is eventually used for delay computation. Then we present in Section IV a methodology for estimating the useful flow rate in the network nodes as well as the decoding delay. The centralized and semi-distributed algorithms for selecting the network coding nodes are presented in Section V. Simulation results are proposed in Section VI where the benefits of the proposed algorithms are evaluated for video streaming applications in various realistic network cases. Finally, the related work is discussed in Section VII and conclusions are drawn in Section VIII.

II. Network Coding Framework

We consider a streaming system that consists of servers, clients, and intermediate nodes, as illustrated in Fig. 1. The overlay network offers source and path diversity, which can be efficiently exploited with network coding techniques that randomly combine packets in the nodes. This increases the packet diversity in the network and leads to efficient exploitation of the channel resources without the need for complex scheduling or

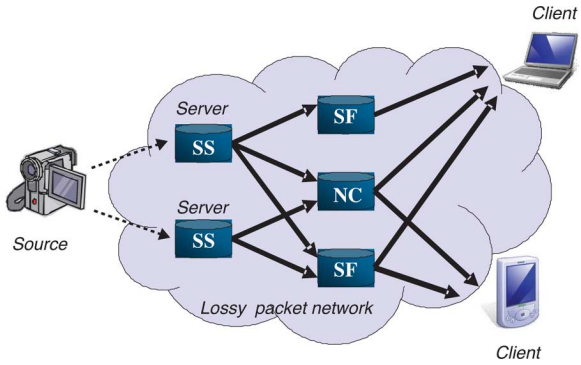

Fig. 1. Illustration of a system for streaming on overlay networks. Multiple streaming servers (SS) send information to clients on a lossy packet network via intermediate nodes that can be either network coding (NC) or "store and forward" (SF) peers.

nodes coordination mechanisms [1]. The network is modeled by a directed acyclic graph $G = (V, E)$ where V is the set of network nodes and E is the set of edges (links) in the network. Each network link between nodes u and v is characterized by a bandwidth $b_{u,v}$ (expressed in terms of packets per second) as well as a packet loss rate $\pi_{u,v}$. We assume that all servers transmit the same multimedia content to clients via intermediate nodes that could either be NC or SF nodes. We consider that the intermediate nodes are not necessarily interested in the transmitted content, but rather act as helper nodes and assist the packet delivery system. The system implements a push-based packet delivery strategy that involves lower communication and coordination overhead than a pull-based solution. We consider that the servers can also implement randomized coding on the source packets for improved robustness. The coded packets are then pushed to the clients through the successive intermediate nodes. Finally, the clients perform decoding after receiving enough packets to build a full rank decodable system of packets.

The SF nodes simply send at each transmission opportunity the first packet in their buffer, which has not been sent previously. The buffer is managed in a first-in-first-out manner, where the oldest packets are replaced by new ones when the buffer is full. When the outgoing bandwidth is larger than the incoming capacity, an SF node sends random replicates of packets from its buffer. On the other hand, the intermediate nodes that perform network coding combine randomly the buffered packets in order to generate network coded packets that are further transmitted to neighbor nodes. As suggested in [4], the NC nodes first check whether the received packets are *innovative*, where innovative packets characterize packets carrying novel information. Non-innovative packets are discarded immediately as they do not increase the symbol diversity into the network. Then the NC nodes randomly combine the remaining packets with coding operations based on randomized network coding (RNC) [8]. It is a simple and efficient network coding solution in distributed systems. RNC codes work similarly to rateless codes [9], [10] and can generate an arbitrary number of coded packets from a given set of source packets. It provides a means for simple bandwidth adaptation.

Formally, the network coding operations performed in a peer node can be written as follows. An NC node u generates M

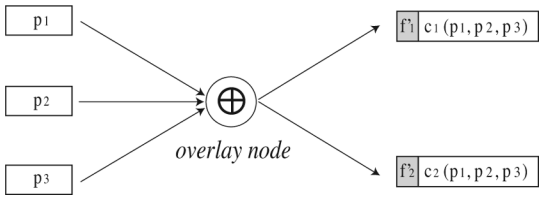

Fig. 2. NC node combines incoming packets p_i and generates network coding packets c_m. A header f'_m is appended to each coded packet and carries the coding coefficients.

packets by RNC. The mth network coded packet c_m is of the form

$$c_m = \sum_{p_i(u) \in \mathcal{N}(u)} f_{m,i} \cdot p_i(u)$$

where $\mathcal{N}(u)$ corresponds to the set of packets of the same generation that are available at node u, $p_i(u)$ denotes either a network coded packet or a native (uncoded) packet, and $f_{m,i}$ is a random coefficient over the Galois field of size q, GF(q). The basis of the Galois field is typically set to $q = 256$, as it has been shown in [4] that this guarantees high symbol diversity and low probability of building duplicate packets. As the packets combined in a node are actually combinations of the original data packets, the encoded packets can be expressed as a function of the native packets

$$c_m = \sum_{i=1}^{N} f'_{m,i} \cdot n_i \tag{1}$$

where N is the total number of native packets, e.g., for video transmission, N can be the number of video packets. The parameters n_i and $f'_{m,i}$ represent, respectively, the native packets and their corresponding coding coefficients after random network coding operations. It is worth noting that some of the coefficients $f'_{m,i}$ can be zero, which means that c_m does not contain information about the native packet n_i. As the coding coefficients are chosen randomly, a header of constant length is appended to each packet with coefficient information, so that the receiver can decode the stream and recover the original data packets. A network coded packet is thus augmented with a header containing the vector of coding coefficients. Fortunately, the header does not grow with the number of hop transmissions due to the relation of equation (1). The encoding procedure in a peer node is depicted in Fig. 2.

The decoding operations at the client basically consist in solving the system of equations that correspond to the network coding operations. Upon collecting a network coded packet, the client stores it in a buffer and adds a line into a matrix $\mathbf{F}$ that contains the coding coefficients. When a full rank system is collected, the original packets are reconstructed by solving the following equations:

$$\mathbf{c} = \mathbf{F} \cdot \mathbf{n} \Longrightarrow \begin{bmatrix} c_1 \\ \vdots \\ c_N \end{bmatrix} = \begin{bmatrix} f'_{1,1} & \cdots & f'_{1,N} \\ \vdots & \ddots & \vdots \\ f'_{N,1} & \cdots & f'_{N,N} \end{bmatrix} \cdot \begin{bmatrix} n_1 \\ \vdots \\ n_N \end{bmatrix}$$

where $\mathbf{c}$ and $\mathbf{n}$ are, respectively, vectors with the coded and source packets. The solution of the equations system is typically computed by Gaussian elimination [4].

The proposed streaming system leads to the following observations. First, the network coding nodes act somehow similarly to sources in the sense that they refresh the set of packets in the network by coding operations. This is necessary as there is a non-zero probability for the reception of duplicate packets in the network nodes when the network is mostly composed of SF nodes. These duplicates can be generated by a node that does not receive enough diverse packets, or from different nodes that independently transmit identical packets. These duplicate packets lower significantly the stream delivery performance especially in networks containing bottlenecks. However, the careful placement of a few network coding nodes in the overlay can help to reduce the number of duplicates in the network. If the number of network coding nodes becomes too large, the probability for the randomized network coding operations to generate duplicate packets becomes again non-negligible. This is especially the case if coding operations are restricted to small generations due to delay constraints. As redundancy, delay, and computational overhead might increase with the number of coding nodes, it becomes quite apparent that efficient systems should not implement network coding in every overlay node. Instead, one has to find an effective placement of network coding nodes in order to fully exploit the network diversity in overlay streaming applications.

III. SF Buffer Model

We provide in this section a buffer model for the SF nodes, which forward and possibly replicate packets if the outgoing bandwidth is sufficient. The buffer model is used to estimate the rate of replicated packets, which is an important parameter in the computation of the decoding delay in the receivers.

As illustrated in Fig. 3, we consider that each SF node has two buffers of capacity h (in packets): the Main Buffer (MB), where the incoming packets are stored, and the Copies Buffer (CB), where copies of the packets that have been recently transmitted are stored. Both buffers follow a FIFO model as the oldest packets are overwritten by the new ones when the node's buffer capacity is exceeded. In addition, since our system works with deadline-constrained data, the packets are removed from the buffers when their decoding deadline expires.

The buffering process works as follows: when a packet arrives in an SF node, it is stored in MB. When the SF node has a transmission opportunity, a packet from MB is sent and thus removed from MB. A copy of this packet is kept in CB. Whenever MB is empty and the node has other transmission opportunities, it randomly selects a packet from CB and transmits it. In other words, the node transmits packet replicates when the outgoing bandwidth is sufficient. The packets in CB are overwritten after some time by newer packets. In our model, if an SF node does not have sufficient outgoing bandwidth for replication, it does not use CB. Alternatively, if the outgoing bandwidth is large, CB is used extensively and MB is often empty. It is important to note that MB and CB are not necessarily of the same size. The size of the buffer is actually not a critical design parameter

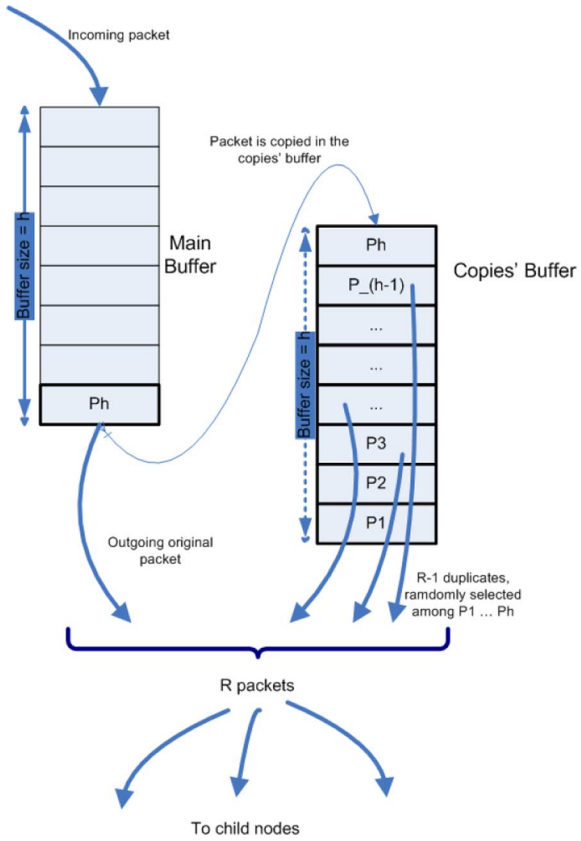

Fig. 3. Packet replications procedure followed in an SF node after the reception of the hth innovative packet.

in our system; it mostly affects the efficiency of the packet replication process with higher redundancy when CB is small.

We are now interested in computing the number of packet replicates generated by the SF node u under the proposed buffer model. *A priori*, the average number of replicates per packet $R(u)$ at node u is given by the ratio of outgoing and incoming bandwidths $b_o(u)$ and $b_i(u)$, respectively. We can write $R(u) = b_o(u)/b_i(u)$. However, the probability for a packet to be replicated depends on the order of its arrival to the SF node. Typically, a packet that arrives early spends more time in the buffer and thus has a higher probability to be replicated than a packet that arrives late and close to the decoding deadline. We thus consider three cases depending on the order of arrival of the packets. We denote the arrival by the position k of a packet in a coding generation.

The first case includes the earliest packets that reach the node while CB is not full. Every packet is replicated with a uniform probability $1/x$, $x < h$, where x is the number of packets available in CB. Hence, the packet replication rate decreases as the buffer level increases. After some point, CB becomes full, but the early packets stay for some more time in CB before getting flushed; the replication rate is $1/h$ in this case. These two factors are described in the large parenthesis of (2). The average number of copies for the kth packet is thus given by

$$R_k(u) = 1+(R(u)-1)\left(\sum_{x=k}^{h}\frac{1}{x}+(k-1)\frac{1}{h}\right), \text{ for } k\in[1,h]. \quad (2)$$

The second case corresponds to packets that reach the node while CB is full. When CB is full, each new packet overwrites the oldest packet in CB. Each packet has a lifetime in CB that corresponds to the time necessary to collect h new packets in the SF node. The replication probability is equal to $1/h$ and the average number of copies is then equivalent to the average replication rate in the node. In this stationary mode, we have

$$R_k(u) = R(u), \quad \text{for } k\in[h+1,K] \quad (3)$$

where K is the number of packets that fully traverse CB until the head of the buffer's queue. Finally, the third case corresponds to the packets that do not spend a full lifecycle in CB due to the expiration of the decoding deadline. When the decoding deadline expires, CB is flushed, and the packets in the buffer at that moment have less opportunities to be replicated, since they do not traverse fully the buffer. If $k' = k - K$ denotes the position of the packets in CB when the buffer is flushed, the average number of copies of the late packets can then be written as

$$R_k(u) = 1+(R(u)-1)\cdot\frac{h-k'+1}{h}, \text{ for } k\in[K+1,|\mathcal{N}(u)|] \quad (4)$$

where $|\mathcal{N}(u)|$ is the overall number of packets of the same coding generation that reach the node u. These packets are not necessarily innovative for a client c as this depends also on the packets received from all other nodes connected with the client c. The number of innovative packets received from client c that have been transmitted from a node u is denoted $N_c(u)$. The exact calculation of $|\mathcal{N}(u)|$ is equivalent to the calculation of the decoding delay and cannot be computed analytically. It can only be estimated through an iterative procedure that will be described in Section IV. The fraction in (4) denotes the percentage of the time that a packet from a generation stays in the CB prior to the deletion of all packets of the generation.

In summary, two main factors affect the packet replication rate, the FIFO behavior of the CB buffer and the expiration of the decoding deadline that causes the deletion of packets in the buffer. Thus, the first $h-1$ packets are replicated more than average and the last packets are replicated less than average, while the intermediate packets have constant replication rate. Finally, it should be noted that, depending on the bandwidth value and the delay constraints, there are situations where the buffer does not reach the stationary regime of (3) and the computation of the number of replicates shall be adapted accordingly.

We use the above buffering model to compute an equivalent packet replication rate $\hat{R}(u)$ for all packets in an SF node, which is more precise than the average value $R(u) = b_o(u)/b_i(u)$. The equivalent replication rate is estimated so that the number of packets at the client c is preserved with respect to the case where the packet replication rate is computed independently for each packet. We assume that each packet travels independently to the client c, and we pose the following equivalent condition:

$$\sum_{k=1}^{|\mathcal{N}(u)|}\left\{1-(\epsilon_c(u))^{R_k(u)}\right\} = |\mathcal{N}(u)|\cdot\left\{1-(\epsilon_c(u))^{\hat{R}(u)}\right\} \quad (5)$$

where $\epsilon_c(u)$ is the probability of losing a packet between the node u and the client c. The average number of copies $R_k(u)$ is

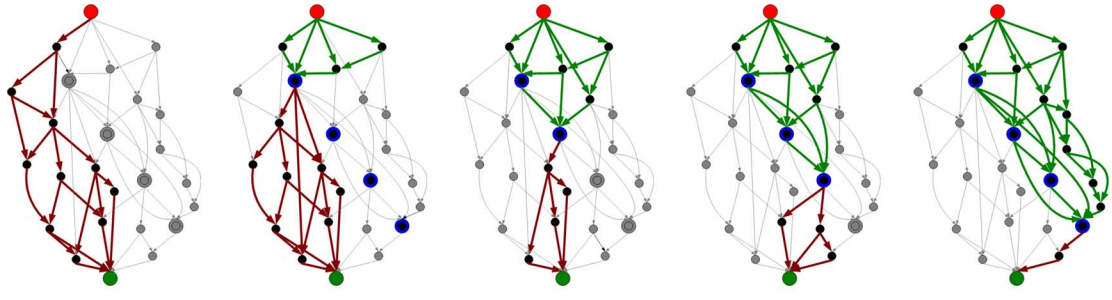

Fig. 4. Flow decomposition of the network graph by the proposed algorithm. The top (red) node and the bottom (green) node are, respectively, the source and the client of the considered topology. The SF and NC nodes are represented, respectively, by the big and small black nodes. Flow from the (a) source, (b) first NC node, (c) second NC node, (d) third NC node, and (e) fourth NC node.

computed from (2)–(4). Rewriting the above equation, we can express the equivalent replication rate as

$$\hat{R}(u) = \frac{\log\left(1 - \frac{\sum_{k=1}^{|\mathcal{N}(u)|}\left\{1-(\epsilon_c(u))^{R_k(u)}\right\}}{|\mathcal{N}(u)|}\right)}{\log\left(\epsilon_c(u)\right)}. \quad (6)$$

We use this replication rate estimate in the computation of the decoding delay in Section IV.

IV. Delay Analysis

A. Estimation Methodology

Our objective is to minimize the decoding delay by the proper placement of NC nodes in the overlay. The decoding delay is the time required to gather a sufficient number of innovative packets for decoding. In the analysis below, we restrict our attention to cases where a full rank system is built at decoder and estimate the delay necessary for this situation to happen. We further construct our analysis for one coding generation, while the extension to multiple generations is straightforward. The decoding delay depends on the rate of innovative packets at the client. The innovative rate increases monotonically with the number of useful packets at the client [4], which corresponds to the number of different packets that reach the client. Hence, a higher rate of useful packets leads to a smaller decoding delay.

In order to compute the decoding delay, we consider that SF nodes replicate packets in case of large outgoing bandwidth. We further consider that new packets are generated only at sources and NC nodes. We treat these nodes independently in the computation of the delay at the client as illustrated in Fig. 4. We assume that the probability of generating two identical packets in the sources or NC nodes is negligible due to the large size of the Galois field. In more details, we first estimate the delay noticed by the client when packets are sent from a given source through all the paths connecting this source to the client, except for the paths that traverse NC nodes [see Fig. 4(a)]. Next, we consider the NC node that is the closest to the source and all the paths that connect it to the client, except for those passing through other NC nodes [see Fig. 4(b)]. Similarly, all other NC nodes are considered only when all their parent NC nodes have been visited. This procedure is repeated for the unprocessed NC nodes and the corresponding graphs are shown, respectively, in Fig. 4(c)–(e). The total delay is computed under the assumption that all sources and NC nodes send independent streams. Equivalently, we assume that the total useful flow is equal to the sum of the useful flows generated by the source and all the NC nodes.

As we consider lossy network paths, we have to take into account that some of the packets generated by network nodes do not reach their destination. We thus estimate the probability $\epsilon_c(u)$ that a packet sent by the node u does not reach the client c. This probability is computed on the subgraph that contains all the paths connecting the node u with the client c but excludes the paths that traverse other NC nodes since these typically alter the set of received packets. The corresponding subgraphs are colored in red in Fig. 4. We denote by $D(u)$ the set of children of node u (excluding the NC nodes) in the subgraph and we define

$$\rho_{u,v} = \frac{b_{u,v}}{\sum_{v \in D(u)} b_{u,v}}$$

as the probability that a packet transmitted by node u is forwarded to a descendant node $v \in D(u)$. In addition, we write the probability that a packet is deleted in a node due to buffer overflow as

$$\beta(u) = \begin{cases} 1 - \frac{b_o(u)}{b_i(u)}, & b_o(u) < b_i(u) \\ 0, & b_o(u) \geq b_i(u). \end{cases}$$

Recall that $b_o(u)$ and $b_i(u)$ are, respectively, the cumulative incoming and outgoing bandwidth of node u. Then, a packet sent by node u might not reach the client c due to one of the following three causes. The packet can be lost during its transmission to the child node v or it can be lost at the node v due to buffer overflow. Finally, it can be lost along with all its possible copies during the transmission from the child node v to the client c. Overall, the probability $\epsilon_c(u)$ is given as

$$\epsilon_c(u) = \sum_{v \in D(u)} \rho_{u,v} \cdot \{\pi_{u,v} + (1 - \pi_{u,v}) \cdot \beta(v)\} + \sum_{v \in D(u)} \rho_{u,v} (1 - \pi_{u,v}) \cdot (1 - \beta(v)) \cdot \epsilon_c(v)^{\hat{R}(v)} \quad (7)$$

where $\pi_{u,v}$ denotes the packet loss probability over the network segment connecting nodes u and v. The probability $\epsilon_c(u)$ can be computed recursively backwards starting from the clients up to the server or NC nodes. Specifically, we first set to zero the loss probabilities for all the clients. Then, all nodes in the directed acyclic subgraph are visited backwards. Each node is visited

only when all its children nodes have been processed already. Once $\epsilon_c(u)$ is known, we can compute the rate of useful packets received by the client c from a source or an NC node by multiplying the respective outgoing rate by the probability of correct reception $1-\epsilon_c(u)$.

Next, we define $N_c(u)$, the number of packets received by any node u that are potentially useful for the client c. These packets correspond to the data that reaches the node u and contains information that is potentially useful for the client c. It depends on the paths connecting the source nodes and the NC nodes to the node u, i.e., the subgraphs colored in green in Fig. 4. It also depends on the set of SF nodes on these paths. The rate of useful packets transmitted by the sources corresponds to their outgoing bandwidth, as they are able to generate any number of different packets via network coding. However, the useful rate in NC nodes may be smaller than their outgoing bandwidth, as they may have only part of the source information in their buffer. Finally the number of useful packets in SF nodes cannot be larger than the number of incoming packets. It is however difficult to estimate in a direct way. We therefore propose below to compute the delay in a recursive manner.

B. Decoding Delay

In this section, we estimate the delay t_c at a client node c. The decoding delay depends on the rate of useful packets received from the multiple sources or network coding nodes. We estimate the time necessary to form a system of full rank G at the decoder, where G is the generation size in packets. In practice, the client might need to collect a slightly larger number of packets, $\tilde{G}=G/(1-x)$[4] for forming a system with G innovative packets. This is due to the possibility that useful packets from a source might still be redundant and not completely independent of packets generated by other sources. The exact value of the overhead factor x depends on the coding system (e.g., it can be upper-bounded by $1/q$ for RNC, where q is the GF size). However, our analysis is relative, and compare different configurations to select the option that leads to the minimal decoding delay. It becomes therefore equivalent to work with G or $\tilde{G}$ since the solutions that lead to the fastest delivery of G and, respectively, $\tilde{G}$ packets are identical. We choose to work with G in the rest of this section.

We compute the average decoding delay by first estimating the time necessary to collect enough packets from each source or NC node independently. Under the assumption that each one of these multiple collection processes represents a uniform flow of packets, we can finally approximate the expected decoding delay as the time necessary for the collection of a sufficient number of packets from multiple independent flows.

The complete algorithm for computing the decoding delays is given in Algorithm 1. Note that the algorithm uses an iterative procedure to compute the decoding delays, since the equivalent packet replication rate in SF node (see Section III) cannot be exactly computed at first. The algorithm initializes the replication rate to an average value given by the input and output bandwidths of each node, and refines this value along with the successive decoding delay estimations. The NC nodes are examined in the order of their proximity to the sources, i.e., the nodes that are closer to the sources are processed first. The number of useful packets $N_c(u)$ is computed recursively at all NC nodes, starting from those that are close to the sources. Then the algorithm considers NC nodes that receive packets from NC nodes that have been already visited. This specific procedure is applicable in our framework as we consider the iterative selection of network coding nodes. Nodes are checked in a greedy way and the algorithm improves at each stage the current solution by the selection of the additional network coding node that brings the largest delay reduction. The overall algorithm typically converges only after a few iterations.

Algorithm 1: Delay computation algorithm

1: Initialize replication rates for every node: $\hat{R}(u)=b_o(n)/b_i(n)$.

2: **repeat**

3: **for** each client node c **do**

4: Compute $\epsilon_c(u)$ for sending nodes u from (7).

5: Compute $N_c(s)=b_o(s)\cdot(1-\epsilon_c(s))$ for all sources s.

6: **for** each NC node u **do**

7: Compute $t_c(u)$ using (10)–(14) setting node u in SF mode.

8: Compute $t_c(u)'$ using (10)–(14) setting node u in silent mode.

9: Compute $\Delta N_c(u)=G\cdot\left(1/t_c(u)-1/t_c'(u)\right)$.

10: Compute $N_c(u)$ using (8).

11: **if** $N_c(u)<b_o(u)$ **then**

12: Compute $t_c(u)$ using (10)–(14) setting node u in NC mode.

13: **else**

14: Compute the expected decoding delay $t_c(u)$ using (15).

15: **end if**

16: **end for**

17: Compute the average decoding delay t_c considering all sources and NC nodes simultaneously, with (16).

18: **end for**

19: **for** each SF node u **do**

20: Estimate the total number of packets received by each node, per generation: $|\mathcal{N}(u)|=b_i(u)\cdot\max_c t_c$.

21: Update the replication rate $\hat{R}(u)$ with (6).

22: **end for**

23: **until** Until convergence of t_c.

We describe now the delay estimation algorithm in more details. The steps 7–10 of Algorithm 1 correspond to the estimation of the useful rate $N_c(u)$ in NC nodes, which is necessary for computing the initial replication rate of the node u. As this

rate is difficult to estimate in a direct way, we choose a differential method by comparing the delay $t_c(u)$ observed by the client when the node u is active and, respectively, silent. From the expected delay in each node's operation mode, we can compute $\Delta N_c(u)$ that is the difference between useful rates at client c when the node u is active or, respectively, silent. Finally, the rate $N_c(u)$ of the packets at node u that are useful for the client c is computed by solving

$$\Delta N_c(u) = \begin{cases} N_c(u) \cdot \left(1 - \epsilon_c(u)^{\hat{R}(u)}\right), & b_o(u) > b_i(u) \\ N_c(u) \cdot \frac{b_o(u)}{b_i(u)} \cdot (1 - \epsilon_c(u)), & b_o(u) < b_i(u) \end{cases} \tag{8}$$

where the first and second conditions correspond to the cases when the node u in SF mode has a small incoming, respectively, outgoing bandwidth. Note that, when the network does not contain any NC nodes, the rate of useful packets that are transmitted is simply equal to the output bandwidth of the sources. In this case, the useful rate received by the client is simply $N_c(s) = b_o(s) \cdot (1 - \epsilon_c(s))$.

Then we compute the delay due to packets sent by the different sources and NC nodes in the network. We consider two cases. First, we consider the NC nodes that have limited incoming bandwidth [i.e., $N_c(u) < b_o(u)$ in line 12 of Alg. 1] and the sources with outgoing bandwidth larger than the source rate. The probability of generating useful packets in such nodes evolves as the buffer fills in. We start by estimating the number of different packets received by the client c when the node u is the only source of information. In average, the node u sends $\nu(u) = b_o(u)/N_c(u)$ packets in the inter-arrival time of two consecutive packets in its buffer, which are combinations of the same set of input packets. Out of these $\nu(u)$ packets, k packets, can be considered as useful for the client c if the decoding system has a rank deficiency of $k < \nu(u)$. Furthermore, due to packet losses and bandwidth variations in the network, each of the packets generated by the node u arrives at the client c with probability $\epsilon_c(u)$. The probability $A_k(u)$ that k out of the $\nu(u)$ packets arrive at the client c is

$$A_k(u) = \binom{\nu(u)}{k} (1 - \epsilon_c(u))^k \, \epsilon_c(u)^{\nu(u)-k}. \tag{9}$$

Note that in general, $\nu(u)$ does not have an integer value. We therefore perform an interpolation between the values of $A_k(u)$ evaluated on the integer values closest to $\nu(u)$. This is necessary as the packet replicas can only be integers, while the average can be a real number.

We then consider the probability $P_c(u, r, n)$ for the client c to collect r useful packets from data sent by a node u that possesses n useful packets. This probability can be computed recursively as

$$P_c(u, r, n) = \sum_{k=0}^{\nu(u)} P_c(u, r-k, n-1) \cdot A_k(u), \; \forall r \in [1 \cdots n-1]. \tag{10}$$

This relation holds for $r < n$. The first factor in (10) represents the probability that the client c has received in total $r - k$ packets from the node u, while u possesses $n-1$ packets that are useful for client c. The second term comes from (9). The sum of terms over all admissible k leads to the probability $P_c(u, r, n)$.

When $r = n$, the rank of the equation system received by node u cannot exceed r as it is equal to the generation size. Thus, (10) becomes

$$P_c(u, n, n) = \sum_{k=1}^{\nu(u)} P_c(u, n-k, n-1) \cdot \sum_{l=k}^{\nu(u)} A_l(u). \tag{11}$$

We further denote by $\mathcal{P}_c(u, r, n)$ the probability that the client c collects r useful packets precisely due to the arrival of the nth useful packet in the sending node u. This probability differs from (10), which corresponds to the case where decoding is possibly exactly with arrival of the rth packet. $\mathcal{P}_c(u, r, n)$ can be written as

$$\mathcal{P}_c(u, r, n) = \sum_{k=1}^{\nu(u)} \left[P_c(u, r-k, n-1) \cdot \sum_{l=k}^{\nu(u)} A_l(u) \right] \tag{12}$$

where $P_c(u, r-k, n-1)$ includes all possible events that lead the node u to collect packets with rank $r - k$ when it receives the $n - 1$ useful packet for client c. Note that, due to integer constraints, $P_c(u, r, n)$ and $\mathcal{P}_c(u, r, n)$ are also computed by performing interpolation for the integer values closest to $\nu(u)$.

The arrival time of the nth useful packet at the sending node u can be computed from the useful packet rate $N_c(u)$. We assume that $N_c(u)$ represents a constant rate and that the arrival times of packets in node u are uniformly distributed. Now, one can compute the expected number of useful packets $E_c(u)$ that are necessary at the sending node u for the client c to receive G useful packets. It is expressed simply as

$$E_c(u) = \sum_{p=G}^{\infty} p \cdot \mathcal{P}_c(u, G, p). \tag{13}$$

The decoding delay for the client c when the NC node u is the only source of information can be estimated by dividing the expected number of necessary packets by the inter-arrival time between two useful packets. It is written as

$$t_c(u) = \frac{E_c(u)}{N_c(u)}. \tag{14}$$

Then, we consider the sources and the NC nodes that are over-provisioned in bandwidth [i.e., the set of nodes u where $N_c(u) > b_o(u)$, line 14 in Alg. 1]. We assume that they transmit packets that are all potentially useful for the client c. The number of useful packets from node u that reach the client c in this case is given by the rate $N_c(u) = b_o(u) \cdot (1 - \epsilon_c(u))$. When this rate is uniform, the decoding delay when the node u is the only source of information is given by

$$t_c(u) = \frac{G}{b_o(u) \cdot (1 - \epsilon_c(u))}. \tag{15}$$

Finally, the average decoding delay at client c is computed by considering all the sources and NC nodes as independent sources of information with uniform useful rates $1/t_c(u)$. We can write the decoding delay as

$$t_c = \frac{1}{\sum\limits_{u \in S} \frac{1}{t_c(u)}} \tag{16}$$

where S is the set of sources and NC nodes.

V. Selective Placement of NC Nodes

Equipped with methods to estimate the decoding delay on the overlay network, we can design algorithms to decide which nodes in the network should perform network coding. We address the problem of placing A network coding nodes in the overlay network, such that the average delay observed by the clients is minimized. This is typically achieved by selecting network coding nodes such that the packet replication rate is decreased and the innovative flow rate in the network is increased.

However, the optimal selection of the NC nodes is known to be an NP-hard problem [6]. Hence, we design a greedy approach that iteratively searches for the optimal placement of a new network coding node while all the previously added NC nodes are fixed. The candidate nodes for implementing network coding are all the remaining SF nodes in the overlay network. It selects the SF node whose transformation into an NC node brings the highest benefit for the clients. This procedure is repeated until all the A network coding nodes have been selected.

We propose now two variants of the iterative selection algorithm that both use Algorithm 1 for computing the delay, but differ in their view of the network resources. The first algorithm assumes that a central node possesses a global knowledge of the network; it iteratively selects the network coding nodes in a centralized manner. When global network knowledge is not a reasonable assumption, the centralized algorithm still serves as a performance benchmark for other greedy NC node placement algorithms. The second algorithm uses only a local view of the network resources at each node for computing the gains in innovative rate and decoding delay. This algorithm is probably more realistic in practice and can be implemented in a distributed way.

In more details, the centralized algorithm uses the knowledge about the full network and available resources in order to determine the number of innovative packets received by each client. It leads to the iterative selection of A network coding nodes by computing at each stage the benefit of turning any of the SF nodes into an NC node. The candidate node that brings the highest delay decay with its transformation is selected as a new NC node. The algorithm is described in Algorithm 2.

Algorithm 2: Centralized NC node selection

1: **for** $i = 1$ to A **do**

2: for each node u in the set of SF nodes. **do**

3: Turn temporarily u into an NC node

4: Estimate the average decoding delay at the clients $t_c(u)$ (using Algorithm 1).

5: Turn u back into an SF node.

6: **end for**

7: Select the node u^* that minimizes the decoding delay, i.e., $u^* = \arg\min_u \sum_c t_c(u)$

8: Turn permanently u^* into an NC node.

9: **end for**

The second algorithm relaxes the assumption that a central node is aware of the full network status. Instead, the nodes only use local network information for the estimation of the delay. We define a neighborhood around each node. Then, an algorithm similar to the centralized solution above is applied in each neighborhood in order to determine the benefits of turning SF nodes into NC nodes. In particular, each node uses the estimation of the reception probability $\epsilon_c(u)$ that is given by all nodes u in the neighborhood and computes an estimation of the decoding delay based on local information, i.e., the capacities and the loss rates of the subnetwork around the node u. Note that $\epsilon_c(u)$ is also calculated considering only the statistics of node's u neighborhood. These estimations are transmitted periodically to a central agent, which finally makes the decision on the placement of new NC nodes. As the delay estimations are done locally, the computational complexity requirements are less important on the central node. In addition, the communication costs are reduced in the semi-distributed algorithm, as only the local delay estimation are transmitted to the central node. The procedure is summarized in Algorithm 3.

Algorithm 3: NC node selection with local information

1: **for** $i = 1$ to A **do**

2: **for** every SF node u **do**

3: Temporarily transform node u into an NC node.

4: Estimate the average delay $\hat{t}_c(u)$ at the clients using Algorithm 1 with local information consisting of the network statistics within horizon of node u.

5: Transform node u back into an SF node.

6: Transmit $\hat{t}_c(u)$ to a central agent.

7: **end for**

8: Select the node u^* that minimizes the delay, $u^* = \arg\min_u \sum_c \hat{t}_c(u)$

9: Turn permanently u^* into an NC node.

10: **end for**

Both algorithms permit to select a few network coding nodes in the system, such that the coding delay and overall computational complexity in the network is limited. At the same time, the system maintains a high innovative rate for sustained streaming performance. The choice of the number of NC nodes is typically determined by the admissible delay or tolerable complexity in the network. For example, constraints on decoding delay impose a limit on the maximum number of NC nodes in the system. However, the problem of determining the optimal number of NC nodes is out of the scope of this paper. We rather assume that the number of coding nodes or helpers in the streaming system is given *a priori*. The proposed algorithms then solve the problem of placing efficiently the NC nodes in the overlay network.

Finally, it has to be noted that the second algorithm is not fully distributed, as it still uses a central agent to select the NC nodes.

However, since it uses only local information, the proposed solution is certainly amenable to a fully distributed algorithm. One could imagine that each node decides independently if it should implement network coding or not, by comparing the local estimation of the gain in delay to a pre-defined threshold. Alternatively, a distributed consensus solution could be deployed for a coordinated selection of the NC nodes with minimal information exchange between the overlay nodes.

VI. Simulation Results

A. Setup

In this section, we analyze the performance of the proposed NC node selection algorithms for the transmission of video streams in overlay networks. We generate overlay networks based on realistic network bandwidth values and adjacency measurements from the Planet Lab [11], as provided in a snapshot of their network taken on November 24, 2009 by their Scalable Sensing Service (S^3) [12]. Our schemes are evaluated in network topologies obtained from PlanetLab, which provides realistic topologies with nodes and segments of different capacities. The networks under consideration have one source node, three client nodes, and a variable number of intermediate nodes. We create network topologies in the following way. First, the source nodes are positioned, then the nodes are randomly added one-by-one to the topology. For every new node, four nodes are randomly chosen as parent nodes. However, if the new node is not directly connected to any of the selected parents according to the PlanetLab measurement data, the node is removed and a new node is selected. After all nodes have been added to the network, the nodes that cannot be reached by the source and the nodes that are not connected with any client are removed. The resulting network graphs are directed and acyclic by construction. The edge capacity of each link is set to 1/200 of the PlanetLab capacity values in order to get realistic values for the link bandwidth. The packet loss rate of each link is set to 5%. We use random networks that consist of 32, 56, and 100 nodes with two different network diameter values of 6 and 8 hops, where the network diameter represents the maximum distance between any pair of nodes in the network. For all network simulations, we consider the placement of up to 10 NC nodes in each network. In each case, the performance results are averaged over sets of 10 random networks and 100 simulations for each network. Simulations are performed using the NS-3 [13] network simulator. Finally, we assume that the topology, the loss rate, and the link bandwidth do not change with the time. Nevertheless, extension to networks with low and moderate dynamics is straightforward, but requires periodic run of the node selection algorithms at the price of more communications.

The network coding operations are performed in a Galois field of size $\mathrm{GF}(2^8)$ since this field size has been shown to result in a good compromise between performance (packets are innovative with high probability) and information overhead [4]. The generation size is set to 32 packets, which is reasonable for real-time video streaming applications. The packet size is 512 bytes. The packet size is not a critical parameter for our algorithm as the packet diversity, which drives the performance of network coding, is only determined by the generation size. For the chosen values of packet and generation sizes, the network coding overhead remains reasonable, i.e., about 6% over the packet payload. Larger generation sizes would provide higher packet diversity but result in larger overheads that can even reduce the throughput gains due to network coding. The decoding is finally performed by Gaussian elimination. Since we are interested in analyzing the performance in terms of decoding delay, we compute the average delay as the time needed by each client to receive 32 linearly independent packets (i.e., a complete generation). This is the minimal number of packets for the clients to decode the source information.

We evaluate the performance of the centralized and semi-distributed algorithms (resp. Algorithms 2 and 3) denoted as "FHNP" (Full Horizon Node Placement) and "LHNP" (Limited Horizon Node Placement), respectively. We compare them with a greedy search algorithm (FHNPR) similar to Algorithm 2 but that uses the actual delays obtained by NS-3 simulations instead of the delay estimates from Algorithm 1 for the node selection. We also compare to a baseline scheme called in the following as "RANDSEL" that randomly places the NC nodes in the network. For the sake of completeness, we finally study the performance of a scheme where all nodes are NC nodes (this scheme is called "All nodes NC") and a scheme with only SF nodes. The performance of the latter is equal to the theoretical maximum that can be sustained when only routing is enabled and can be found by typical maximum flow algorithms. Finally, we provide comparisons with the centralized algorithm proposed in [7], which minimizes the delay via throughput maximization.

Note that the minimum delay and maximum effective throughput obtained with Linear NC are computed by considering that a high capacity hyper-source node connected with all sources. In this case, the overall throughput is computed as the sum of the throughputs from the hyper-source to each client node. For the routing case, we consider as well a hyper-sink node linked with all client nodes, and we compute the maximum throughput between the hyper-source and hyper-sink with standard graph maximum flow algorithms. We should point out that the links connecting the hyper nodes with the networks sources and clients are error free and have infinite capacities, so they do not introduce extra delays.

B. Decoding Delay

We first study the decoding delay for each of the algorithms. Fig. 5 illustrates the normalized average decoding delays for the network clients as a function of the number of NC nodes added in the network, for two different network diameter sizes. The decoding delays are normalized to the performance obtained when all nodes perform network coding. We show the performance of LHNP for different horizon values of the neighborhood in the local gain estimations. We observe a sharp reduction of the delivery times with the addition of the first few NC nodes for all the algorithms, but especially for the proposed algorithms. The gains become less important after a few NC nodes have been placed in the network. We can thus see that the NC nodes are well positioned in order to improve the delivery performance. The results also highlight the inefficiency of the RANDSEL algorithm, which becomes competitive with the other methods

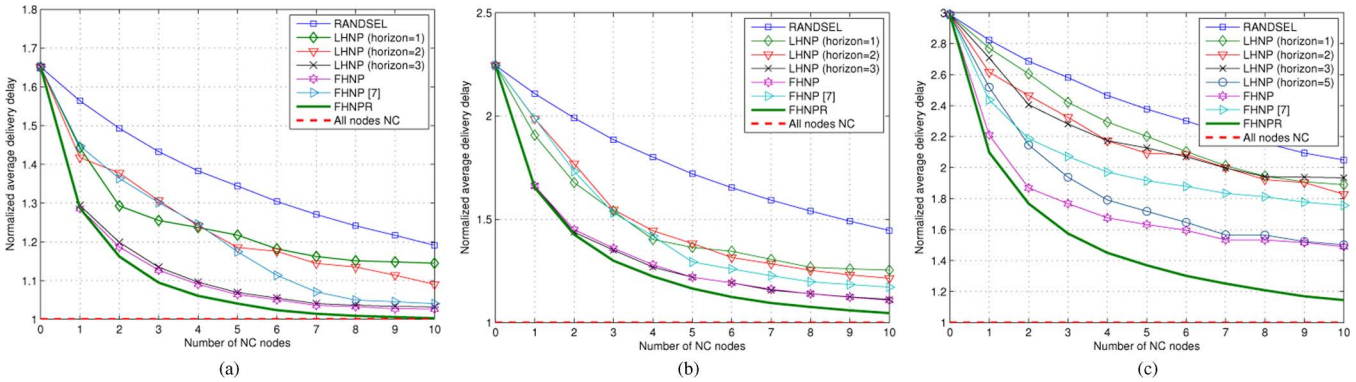

Fig. 5. Normalized average decoding delays versus the number of NC nodes for: (a) networks containing 32 to 56 nodes and network diameter equal to six, (b) network containing 32 to 56 nodes and network diameter equal to eight, and (c) network containing 100 nodes and network diameter equal to six.

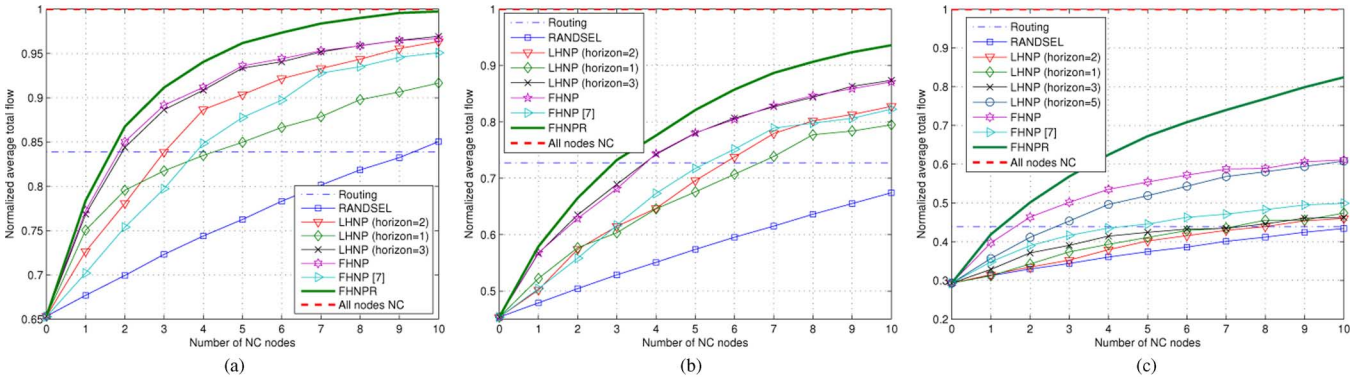

Fig. 6. Normalized achievable throughput versus the number of NC nodes: (a) networks containing 32 to 56 nodes and network diameter equal to six, (b) network containing 32 to 56 nodes and network diameter equal to eight, and (c) network containing 100 nodes and network diameter equal to six.

only for large number of NC nodes. We can see that the FHNP performs similar to the FHNPR. This confirms that the proposed delay estimation strategy in FHNP is accurate as it comes close to the actual delay values measured by the network simulator. The improved performance of the proposed method compared to [7] is coming from the buffering model and the probabilistic approach followed for modeling the replication process. Finally, we can notice that for larger networks the LHNP needs more network knowledge (larger horizon) to approach the performance of FHNP. This is due to the significant network heterogeneity that necessitates the inclusion of more network coding nodes to remove the bottlenecks effect.

C. Innovative Rate

We further look at the average normalized effective throughput in the network (i.e., the number of useful packets received by the clients), as a function of the number of NC nodes in the network. Normalization is performed with respect to the effective throughput achieved by the scheme where all nodes perform network coding. Fig. 6(a) and (b) shows the effective throughputs for two different network diameter values, while Fig. 6(c) for larger networks. The results confirm the earlier observations on the decoding delay performance. A few well-selected nodes are able to bring a large throughput gain. Further performance improvements become less important as the number of NC nodes increases. We see also that the algorithms proposed in this paper provide the best performance among the schemes under comparison. The node selection algorithm with local information improves with the size of the neighborhood but generally stays close to the centralized algorithm when the neighborhood is sufficiently large. In the case where the neighborhood is limited to one node, the proposed algorithm still outperforms RANDSEL since the decisions are not totally blind. The reason for the inferior performance of the semi-distributed scheme compared to the centralized one simply comes from the fact that the local network statistics are not sufficient for accurately estimating the delay when the neighborhood is small. In addition, we can observe that a few NC nodes are sufficient for obtaining higher throughputs than the one in routing algorithms where the nodes simply forward packets randomly to their descendants. It confirms the fact that our methods are appropriate for deployment in low-cost networks, where a few helpers or network coding nodes are sufficient for improved throughput and efficient data delivery. From the evaluation, it is clear that the buffer model and the improved modeling of the replication process gives advantages over the FHNP algorithm presented in [7]. It is worth to note that for large network topologies consisting of 100 nodes, the performance gains over [7] are smaller than those observed for small networks. This is due to the fact that more network nodes are needed to cope with the network heterogeneity in large networks. Finally, we can observe that our algorithms tend to perform better in networks with larger node's horizon

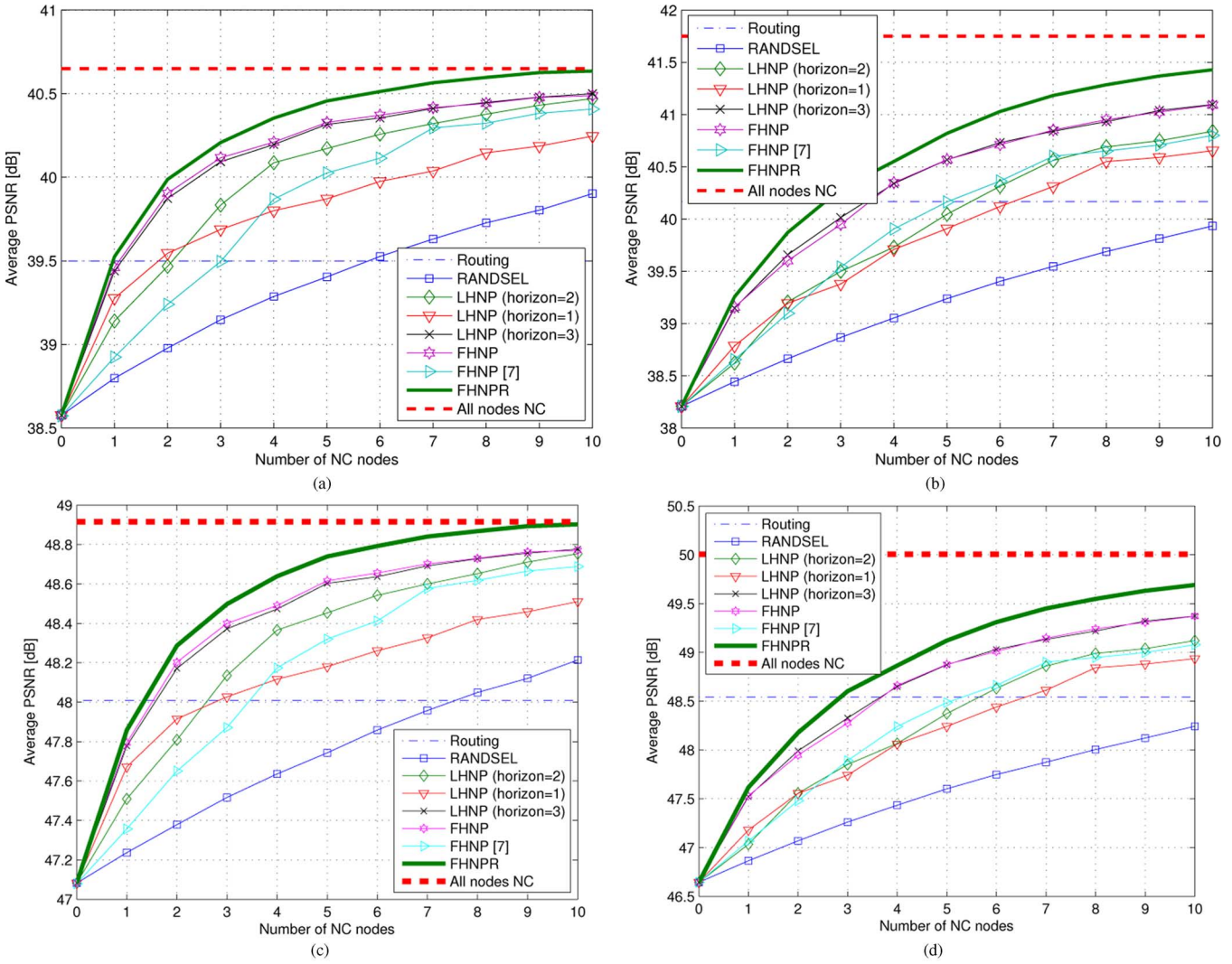

Fig. 7. Average PSNR quality at clients versus the number of NC nodes in the networks for: (a) Foreman CIF sequence when network diameter is six, (b) Foreman CIF sequence when network diameter is eight, (a) Akyio CIF sequence when network diameter is six, and (d) Foreman CIF sequence when network diameter is eight.

values,[2] as they are able to exploit more efficiently the available resources and the diversity in the overlay network.

D. Video Quality

Finally, for the sake of completeness, we study the advantages of the node selection algorithms from the viewpoint of video quality. We estimate the average PSNR quality measured at the clients with respect to the number of NC nodes in the network for all methods under comparison in the transmission of the Foreman CIF and Akyio CIF sequences encoded by the the JM12.2 [14] of the H.264/AVC standard [15]. The video source rate is chosen to be equivalent to the throughput values computed by the server in each algorithm under comparison. The 300 frames of the Foreman CIF video sequence are encoded as *IPPPPP*… with frame rate 30 fps. The GOP size is 30 frames. Prior to transmission, the video encoded packets are divided into consecutive generations of 32 packets of 512 bytes each. We consider in our illustrative video simulation that the playback delay at the clients is larger than the minimal average delay computed by our algorithms; in this case, virtually all packets reach the decoder on time for decoding.

The evaluation results for Foreman CIF sequence are illustrated in Fig. 7(a) and (b). It is interesting that the improved throughput values translate into higher PSNR quality, which confirms the above observations about the benefits of proper NC node selection. We can have gains that exceed 1.5 dB with only two NC nodes, whereas we reach gains of 3 dB for seven NC nodes. The PSNR gains saturate as the number of NC nodes increases, but quality gains can still be noticed. As expected, larger gains are observed for the centralized algorithm; however, the semi-distributed algorithm offers significant gains as well. For a neighborhood of three hops, the performance of the semi-distributed even becomes identical to that of the centralized scheme. Finally, we see that RANDSEL gives small PSNR gains for a few NC nodes, which confirms the poor performance of a random selection of the network coding nodes and supports the development of effective selection algorithms. We

[2]A node's horizon is defined as the longest short path between the considered node and any other network node. Hence, a node with horizon three contains in its neighborhood all the nodes that are at most three hops away from this node.

also evaluate all schemes for transmission of the Akyio CIF sequence. This sequence has static background and moderate motion in the face and shoulders areas. The results are presented in Fig. 7(c) and (d). The evaluation confirms the observations we have reached for Foreman sequence with however a higher video quality due to lower motion in the Akyio sequence.

The above experiments have been proposed for illustrating the importance of the effective placement of network coding nodes. In practice, the channel conditions and even the network topologies are dynamic, which influences the goodput of the system. If the network coding algorithm is not able to finely adapt to these dynamics, the decoding might not be successful and the video quality degrades. Several solutions such as more conservative choice of the playback delay or rate adaptation at the source can be used in such cases.

VII. Related Work

The problem of finding a minimal set of network coding points in a network has been mostly studied from a theoretical perspective so far. First, the special case of two source messages is examined in [16] where it is proved that the number of coding nodes is independent of the total number of network nodes. In [17], the minimum number of network coding nodes is computed through graph coloring techniques. It is then shown in [18] that the number of coding nodes is upper bounded by the number of receivers. A unification of network coding and tree packing theorems is further presented in [19], where network coding is restricted to pre-selected edges. These include only input edges of relay nodes and not the input edges of clients where simple routing is applied. This choice is made in order to achieve the min-cut max-flow limit of the network and save both processing and implementation complexity. The relation between links capacities and the number of coding nodes is investigated in [20], where it is shown that in directed acyclic networks arbitrary amounts of gain can be noticed when subsets of nodes of arbitrary size are used for coding. Finally, the problem of finding network codes with a minimal number of encoding nodes has recently been studied in [6] where the optimization problem is however shown to be NP-hard.

While the previous work mostly consider that the network is fully known at a central node, a decentralized algorithm for minimizing the number of network coding packets flowing in a network has been presented in [21]. It also addresses the design capacity approaching network codes that minimize the set of network coding nodes. However, this algorithm does not provide any guarantee that the minimum set of network coding nodes can always be determined. While [21] consider capacity approaching codes without delay constraints, we rather use well performing network codes and consider the available resources in the network in order to select a set of network coding nodes, such that the overall delay is kept small in multimedia applications. The choice of randomized network codes is mostly geared towards the implementation of practical distributed systems where large benefits are expected by the proper choice of a limited number of network coding nodes.

In general, the previous works about the selection of coding nodes do not consider delay issues, which are most important in streaming applications. The problem of the selection of network processing nodes in multimedia streaming applications has been addressed in [22] in a framework that is however slightly different than ours. The placement of a limited number of network-embedded FEC nodes (NEF) is considered in networks that are organized into multicast trees. The placement is chosen in order to enhance the robustness to transmission errors and to improve the network's throughput. NEF nodes first decode and successively re-encode the recovered packets in order to increase the symbol diversity. A greedy algorithm is proposed for placing NEF nodes. Although the proposed method is efficient, it is computationally expensive and unrealistic to be deployed in dynamic networks. In contrast to [22], we consider the placement of processing nodes in the more general case of overlay mesh networks with randomized network coding for distributed packet delivery.

Finally, game theoretic concepts are adopted in a recent work [23] for developing socially optimal distributed algorithms that decide on the nodes that should combine packets. Specifically, incentives such as extra download bandwidth are given to network nodes in order to change their status to network coding and indirectly minimize the delays in the system. However, this algorithm does not offer any guarantee that limited resources will be used efficiently, since all the nodes may potentially desire to become network coding nodes. It is not appropriate when a certain number of network coding nodes shall be placed effectively in a network.

VIII. Conclusions

We have considered the problem of the placement of a predefined number of network coding nodes in an overlay media streaming system. We have proposed novel algorithms that iteratively select the best nodes for network coding such that the delay is decreased. The deployment of network coding gets positioned as a valid solution for exploiting the network diversity in streaming applications. We show that the selection of a small number of network coding nodes is able to provide an effective tradeoff between packet duplicates, decoding delay, and computational complexity. The experimental evaluation in irregular and realistic networks shows that the proposed node selection schemes achieves the same throughput as a system where all nodes perform network coding, but with a dramatically smaller number of network coding nodes and hence lower complexity. In addition, we show that the quality of experience in video streaming applications is greatly improved by the proper selection of network coding nodes. Finally, the proposed algorithm is amenable to the implementation of distributed solutions that are able to adapt to the local characteristics of a dynamic network topology. It could also offer important insights in the design of effective media delivery solutions, where helpers nodes could be positioned in an overlay networks for maximizing the quality of service offered to the media clients.

References

[1] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung, "Network information flow," *IEEE Trans. Inf. Theory*, vol. 46, no. 4, pp. 1204–1216, Jul. 2000.

[2] S.-Y. R. Li, R. W. Yeung, and N. Cai, "Linear network coding," *IEEE Trans. Inf. Theory*, vol. 49, no. 2, pp. 371–381, Feb. 2003.

[3] M. Wang and B. Li, "R^2: Random rush with random network coding in live peer-to-Peer streaming," *IEEE J. Select. Areas Commun.*, vol. 25, no. 9, pp. 1655–1666, Dec. 2007.

[4] P. A. Chou, Y. Wu, and K. Jain, "Practical network coding," in *Proc. 41st Allerton Conf. Communication, Control, and Computing*, Monticello, IL, Oct. 2003.

[5] T. Ho, M. Medard, J. Shi, M. Effros, and D. R. Karger, "On randomized network coding," in *Proc. 41st Allerton Annual Conf. Communication, Control, and Computing*, Monticello, IL, Oct. 2003.

[6] M. Langberg, A. Sprintson, and J. Bruck, "Network coding: A computational perspective," *IEEE Trans. Inf. Theory*, vol. 55, no. 1, pp. 147–157, Jan. 2009.

[7] N. Cleju, N. Thomos, and P. Frossard, "Network coding node placement for delay minimization in streaming overlays," in *Proc. Int. Conf. Communications (ICC'10)*, Cape Town, South Africa, May 2010.

[8] T. Ho, M. Medard, R. Koetter, D. R. Karger, M. Effros, S. Jun, and B. Leong, "A random linear network coding approach to multicast," *IEEE Trans. Inf. Theory*, vol. 52, no. 10, pp. 4413–4430, Oct. 2006.

[9] A. Shokrollahi, "Raptor codes," *IEEE Trans. Inf. Theory*, vol. 52, no. 6, pp. 2551–2567, Jun. 2006.

[10] M. Luby, "LT codes," in *Proc. of the 43rd Annual IEEE Symposium on Foundations of Computer Science (FOCS '02)*, Vancouver, BC, Canada, Nov. 2002, pp. 271–280.

[11] PlanetLab. [Online]. Available: http://www.planet-lab.org/.

[12] P. Yalagandula, P. Sharma, S. Banerjee, S. Basu, and S. J. Lee, "S^3: A scalable sensing service for monitoring large networked systems," in *Proc. SIGCOMM Workshop Internet Network Management (INM'06)*, Pisa, Italy, Sep. 2006.

[13] The Network Simulator—Ns3. [Online]. Available: http://www.nsnam.org.

[14] Jvt Reference Software Version 12.2. [Online]. Available: http://bs.hhi.de/suehring/tml/.

[15] Information Technology—Coding of Audio-Visual Objects—Part 10: Advanced Video Coding. Final Draft International Standard, 2003, iSO/IEC FDIS 14 496-10.

[16] A. Tavory, M. Feder, and D. Ron, "Bounds on linear codes for network multicast," in *Proc. Electronic Colloq. Computational Complexity (ECCC'03)*, 2003.

[17] C. Fragouli, E. Soljanin, and A. Shokrollahi, "Network coding as a coloring problem," in *Proc. Conf. Information Science and Systems (CISS07)*, Princeton, NJ, Mar. 2004.

[18] C. Fragouli and E. Soljanin, "Information flow decomposition for network coding," *IEEE Trans. Inf. Theory*, vol. 52, no. 3, pp. 829–848, Mar. 2006.

[19] Y. Wu, K. Jain, and S. Y. Kung, "A unification of network coding and tree packing (Routing) theorems," *IEEE Trans. Inf. Theory*, vol. 52, no. 6, pp. 2398–2409, Jun. 2006.

[20] J. Cannons and K. Zeger, "Network coding capacity with a constrained number of coding nodes," *IEEE Trans. Inf. Theory*, vol. 54, no. 3, pp. 2398–2409, Mar. 2008.

[21] K. Bhattad, N. Ratnakar, R. Koetter, and K. R. Narayanan, "Minimal network coding for multicast," in *Proc. IEEE Int. Symp. Information Theory (ISIT 2005)*, Seattle, WA, Sep. 2005.

[22] M. Wu, S. Karande, and H. Radha, "Network embedded FEC for optimum throughput of multicast packet video," *EURASIP J. Appl. Signal Process.*, vol. 20, no. 8, pp. 728–742, Sep. 2005.

[23] N. Thomos, H. Park, E. Kurdoglu, and P. Frossard, "NC node selection game in collaborative streaming systems," in *Proc. Int. Conf. Acoustics, Speech, and Signal Processing, ICASSP'10*, Dallas, TX, Mar. 2010.

Nicolae Cleju received the Diploma degree (valedictorian) in electronics engineering from the "Gheorghe Asachi" Technical University of Iasi, Iasi, Romania, in 2007, the M.Sc. degree in electrical engineering from the Swiss Federal Institute of Technology, Lausanne, Switzerland, in 2009, and the M.Sc. degree in signal processing from the "Gheorghe Asachi" Technical University of Iasi in 2010, where he is currently pursuing the Ph.D. degree in the Signal Processing Laboratory of the Faculty of Electronics, Telecommunications, and Information Technology.

Nikolaos Thomos (S'02–M'06) received the Diploma and the Ph.D. degrees from the Electrical and Computer Engineering Department of the Aristotle University of Thessaloniki, Thessaloniki, Greece, in 2000 and 2005, respectively.

Currently he is postdoctoral research fellow with the Signal Processing Laboratory of Swiss Federal Institute of Technology (EPFL), Lausanne, Switzerland. Previously, he was a postdoctoral and graduate research fellow with the Informatics and Telematics Institute/Centre for Research and Technology Hellas, Thessaloniki, Greece. His research interests include network coding, multimedia communications, joint source and channel coding, and distributed source coding.

Dr. Thomos has been awarded the highly esteemed Ambizione career award in 2008 from the Swiss National Science Foundation targeted to prospective researchers. He is a member of the Technical Chamber of Greece.

Pascal Frossard (S'96–M'01–SM'04) received the M.S. and Ph.D. degrees, both in electrical engineering, from the Swiss Federal Institute of Technology (EPFL), Lausanne, Switzerland, in 1997 and 2000, respectively.

Between 2001 and 2003, he was a member of the research staff at the IBM T. J. Watson Research Center, Yorktown Heights, NY, where he worked on media coding and streaming technologies. Since 2003, he has been a professor at EPFL, where he heads the Signal Processing Laboratory (LTS4). His research interests include image representation and coding, visual information analysis, distributed image processing and communications, and media streaming systems.

Dr. Frossard has been the General Chair of IEEE ICME 2002 and Packet Video 2007. He has been the Technical Program Chair of EUSIPCO 2008, and a member of the organizing or technical program committees of numerous conferences. He has been an Associate Editor of the IEEE TRANSACTIONS ON MULTIMEDIA (2004-), the IEEE TRANSACTIONS ON IMAGE PROCESSING (2010-), and the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY (2006-). He is an elected member of the IEEE Image, Video and Multidimensional Signal Processing Technical Committee (2007-), the IEEE Visual Signal Processing and Communications Technical Committee (2006-), and the IEEE Multimedia Systems and Applications Technical Committee (2005-). He has served as Vice-Chair of the IEEE Multimedia Communications Technical Committee (2004–2006) and as a member of the IEEE Multimedia Signal Processing Technical Committee (2004–2007). He received the Swiss NSF Professorship Award in 2003, the IBM Faculty Award in 2005, and the IBM Exploratory Stream Analytics Innovation Award in 2008.