Network coding node placement for delay minimization in streaming overlays

Nicolae Cleju, Nikolaos Thomos, and Pascal Frossard
Signal Processing Laboratory (LTS4)
Ecole Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland
nikcleju@etti.tuiasi.ro, {nikolaos.thomos,pascal.frossard}@epfl.ch

***Abstract*—Network coding has been proposed recently as an efficient method to increase network throughput by allowing network nodes to combine packets instead of simply forwarding them. However, packet combinations in the network may increase delay, complexity and even generate overly redundant information when they are not designed properly. Typically, the best performance is not achieved when all the nodes perform network coding. In this paper, we address the problem of efficiently placing network coding nodes in overlay networks, so that the rate of innovating packets is kept high, and the delay for packet delivery is kept small. We first estimate the expected number of duplicated packets in each network node. These estimations permit to select the nodes that should implement network coding, so that the innovating rate increases. Two algorithms are then proposed for the cases where a central node is aware of the full network statistics and where each node knows the local statistics from its neighbor, respectively. The simulation results show that in the centralized scenario the maximum profit from network coding comes by adding only a few network coding nodes. A similar result is obtained with the algorithm based on local statistics, which moreover performs very close to the centralized solution. These results show that the proper selection of the network coding nodes is crucial for minimizing the transmission delay in streaming overlays.**

***Index Terms*—Network coding, delay minimization, throughput maximization, overlay networks.**

I. Introduction

The recent development of overlay networks is quite interesting for multimedia transmission since these networks offer significant network diversity that can be used for improved quality of service. The traditional streaming systems based on ARQ or channel coding techniques often fail to efficiently exploit this diversity, since they suffer from high computational costs and are quite unreliable in large scale networks where 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 transmission performance. Specifically, network coding nodes randomly combine the buffered packets before forwarding them to next hop nodes. It is particularly appealing in networks with diversity, as it does not impose coordination between nodes. It allows for better adaptation to the available bandwidth and even permits to approach max-flow min-cut bound of the network graph. Overall, the network coding systems show improved resiliency to dynamics, delays, scalability and buffer capacities [3].

This work has been partly supported by the Swiss National Science Foundation, under grant PZ00P2-121906.

Multimedia delivery systems, however, face important transmission challenges due to strict timing constraints that impact the design of network coding algorithms. A practical network coding scheme is presented in [4], which proposes a proper format for distributed transmission of multimedia streams. This scheme adopts randomized network coding (RNC) techniques [5] and devises a special protocol to deal with buffering issues and timing constraints. Moreover, it introduces the concept of generations that restricts the coding operations to packets that share similar decoding deadlines. In this scheme, the network coding coefficients are communicated along with the data packets so that the decoder can recover the transmitted information. However, network coding also induces delays and computational overhead that increase with the number of network coding nodes. As delay is critical in streaming applications, nodes often have to perform packet combinations with only a few packets from their receiving buffer. Hence, the probability of generating purely redundant packets increases when network coding operations are successively repeated in many nodes. It becomes important to select efficiently the subset of nodes that perform network coding in order to maximize the performance of the streaming system.

In this paper, we discuss solutions for the placement of network coding nodes in order to minimize the transmission delays. We adopt the generation and buffer models of the practical network coding scheme proposed in [4]. We estimate the rate of innovative packets in the network nodes, and we later select the network coding nodes in order to minimize the redundant information in the network. We consider two cases where (i) a central node is aware of full network statistics or (ii) only local statistics are known by the nodes, respectively. In both cases, we further design algorithms that iteratively determine the positioning of a given number of network coding nodes such that the innovative rate is maximized. The simulation results show that only a few network coding nodes lead to throughput gains close to max-flow min-cut bound and greatly decrease the delay necessary for data 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 solutions where network coding nodes are selected randomly.

The problem of the selection of network processing nodes has been addressed in a different context in [6]. The placement

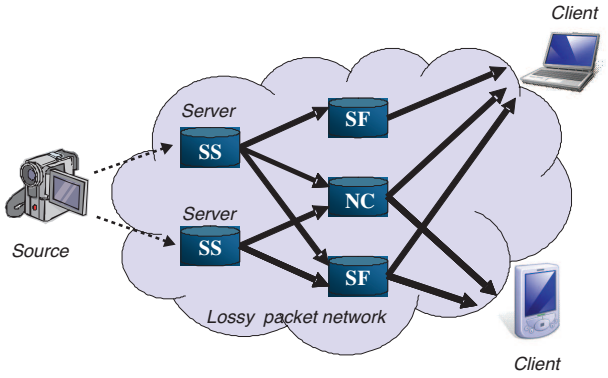


Fig. 1. Illustration of a system for streaming on overlay networks. Multiple streaming servers (SS) send information to clients on a lossy packet network via intermediate nodes that can be either network coding (NC) or 'store and forward' (SF) peers.

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 [6] we rather consider the placement of processing nodes in the more general case of overlay mesh networks with randomized network coding for distributed packet delivery. Note finally that the problem of finding network codes with a minimal number of encoding nodes has recently been studied in [7] where the problem is shown to be NP-hard. In this paper we rather consider randomized network codes for the implementation of practical distributed systems where we try to limit the number of coding nodes while large performance gains are achieved.

II. System description

The overall streaming system consists of servers, clients and intermediate nodes, as illustrated in Fig. 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 is characterized by its capacity c_{ij} which is expressed in terms of packets per second as well as the packet loss rate $\pi_{i,j}$. We assume that all servers transmit the same multimedia content to clients via intermediate nodes that could either be network coding (NC) or "store and forward" (SF) nodes. We consider that the intermediate nodes are not interested in the transmitted content, but rather act as helper nodes and assist the packet delivery system. The system implements a push-based strategy which involves lower communication and coordination overhead than a pull-based solution. The servers also implement randomized network coding for improved robustness, and the coded packets are then pushed to the clients through the successive intermediate nodes.

The intermediate nodes that perform network coding combine randomly the buffered packets in order to generate network coded packets that are sent to neighbour nodes. As suggested in [4], the NC nodes first check whether the received packets are innovative.[1] Non-innovative packets are discarded immediately as they do not increase the symbol diversity into the network. Then the nodes randomly combine the remaining packets with coding operations performed in large Galois fields in order to reduce the probability of generating duplicates. The SF nodes simply transmit at each opportunity the first packet in their incoming 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 the new ones when the buffer is full. When the outgoing bandwidth is larger than the incoming one, a SF node randomly replicates packets from its buffer by making sure that the packet diversity is maximal for each of the outgoing links. Finally, the clients perform network decoding after receiving enough packets to build a full rank decoding system.

When the network is mostly composed of SF nodes, there is a non-zero probability for the reception of duplicate packets in the network nodes. The 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 decrease significantly the packet diversity 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. Indeed, 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. However, when the number of network coding nodes becomes too large, the probability for the randomized network coding operations to generate duplicate packets becomes non-negligible. In addition, the delay and computational overhead in the system generally increase with the number of coding nodes. This clearly outlines the tradeoff underlying the effective placement of network coding nodes.

III. Selective placement of NC nodes

We now propose algorithms for an effective placement of the network coding nodes. The objective is to minimize the average delay at decoder by reducing the packet replication with network coding, and hence maximizing the innovative flow rate in the network. We first estimate the packet replication probability at the clients and later propose two iterative algorithms for network coding node selection.

A. Packet replication probability

In order to compute the packet replication probability, we only consider the SF nodes in the overlay. We assume that the NC nodes cancel the replication effects and we do not consider them in the computation of the packet replication probabilities. We then define $p_c^n(k)$ as the probability for the client c to receive k times a packet sent from node n. This probability can be computed recursively starting from the clients to the servers. We define D_n as the set of children for node n. The probability $p_c^n(k)$ is then given by

[1] Innovative packets are the packets that carry novel information.

$$p_c^n(k) = \sum_{m \in D_n} \rho_{n,m} \cdot (1 - \pi_{n,m}) \cdot \beta_m \cdot \{\theta^m(M) \cdot P_c^m(M+1,k) + (1 - \theta^m(M)) \cdot P_c^m(M+2,k)\} \quad (1)$$

where $\rho_{n,m} = \frac{c_{nm}}{\sum_{m \in D_n} c_{nm}}$ is the probability that a packet from node n is forwarded to a descendant node m. The parameter β_m represents the probability that a packet received by the node m is not deleted due to buffer overflow. It is written as

$$\beta_m = \begin{cases} \frac{b_o(m)}{b_i(m)}, & b_o(m) < b_i(m) \\ 1, & b_o(m) \geq b_i(m) \end{cases}$$

$b_o(m)$ and $b_i(m)$ are respectively the cumulative incoming and outgoing capacity (in packets) of node m.

By the design of our packet replication protocol, the number of replications is the same for every packet with at most one unit difference. In particular, a packet can only be duplicated either M or $M+1$ times, with

$$M = \left\lfloor \frac{r^m}{b_i(m)} \right\rfloor,$$

where r^m is the total number of duplicates generated at node m (i.e., $r^m = b_o(m) - b_i(m)$). We denote by x^m the number of duplicates transmitted by the node m for each packet it receives. We can define the probability that this number of duplicates is M as

$$\theta^m(M) = Prob(x^m = M) = 1 - \left(\frac{r^m}{b_i(m)} - M \right).$$

Respectively, we have $\theta^m(M+1) = 1 - \theta^m(M)$. Finally, $P_c^m(M,k)$ in Eq. (1) is the probability that k duplicates of a packet reach the client c, when M replicates of this packet had originally been generated by node m. Note that k might be larger than M due to successive replication stages in the network. We assume that each packet replica travels independently through the network, and we can compute the probability $P_c^m(M,k)$ as

$$P_c^m(M,k) = \sum_{l_1=0}^{k} \sum_{l_2=0}^{w_2} \cdots \sum_{l_{M-1}=0}^{w_{M-1}} \prod_{j=1}^{M-1} p_c^m(l_j) \cdot p_c^m(w_M) \quad (2)$$

where $w_i = k - \sum_{j=1}^{i-1} l_j$. By iterating between Eq. (1) and Eq. (2), we can estimate the number of duplicates received from any source in the network. Note that, for the special case where k is zero (i.e., no copy of a packet transmitted from node n is received at client c), the probability of replicates can be written as

$$p_c^n(0) = \sum_{m \in D_n} \rho_{n,m} \cdot \{\pi_{n,m} + (1 - \pi_{n,m}) \cdot (1 - \beta_m) + (1 - \pi_{n,m}) \cdot \beta_m \cdot \{\theta^m(M) \cdot P_c^m(M+1,0) + (1 - \theta^m(M)) \cdot P_c^m(M+2,0)\}\}. \quad (3)$$

The three terms in the right side of Eq. (3) correspond respectively to the fact that a packet can be either erased on the link connecting nodes n and m, lost due to buffer overflow at m, or all replicas are lost in the case the packet has been replicated at node m.

We can now compute the packet replication probabilities at the clients with an iterative algorithm, as illustrated in Algorithm 1. The algorithm starts by first initializing the probabilities for all clients. All paths in the directed acyclic graph are visited backwards from clients to servers. Every intermediate node computes by Eqs. (1) and (2) the probabilities that each client receives k replicas. The intermediate nodes are considered only when all their child nodes have been already processed. The NC nodes are excluded from the analysis. This process terminates at servers. When a packet is received k times by a client, it means $k-1$ duplicates are received (only the first received packet is innovative). Therefore, the expected number of packet replicas received by each client c is given by

$$N(c) = \sum_{s \in S} \left\{ b_o(s) \cdot \sum_{k=2}^{N_{max}} p_c^s(k) \cdot (k-1) \right\} \quad (4)$$

where S denotes the set of servers and NC nodes and N_{max} is the maximal number of duplicates that can be generated by the network. The expected flow of innovative packets at client c is finally given by

$$I(c) = b_i(c) - N(c) \quad (5)$$

Algorithm 1 Computation of innovative flow rates

1: **Initialization:** set the probabilities of the client nodes:

$$p_c^j(1) = \begin{cases} 1, j = c \\ 0, j \neq c \end{cases} \qquad p_c^j(k) = 0, k \neq 1$$

2: **while** there are unprocessed nodes remaining **do**
3: Process next node (in inverse topological order), applying recursively Eqs. (1) and (3).
4: **end while**
5: **The expected number of duplicate packets** received by each client c is computed from Eq. (4).
6: **The expected flow of innovative packets** for client c is computed from Eq. (5).

B. Node selection algorithms

The problem of the optimal selection of the NC nodes is known to be an NP-hard problem. We focus here on a greedy approach that searches at each step the optimal placement for a novel network coding node, assuming that all other NC nodes are known. The candidate nodes for turning into NC mode are only the SF intermediate nodes. The algorithm iteratively examines all nodes backwards from clients to servers and finally outputs the subset of nodes that should implement network coding.

We now consider two different cases with (1) a centralized solution with global knowledge of the network and (2) a solution where all the nodes only have a local view of the network. The Algorithm 1 is used in both cases to compute

the innovative rates, whose computation stops before reaching the server in the second scenario.

1) Global information: A fully centralized algorithm is devised to accurately determine the number of duplicate packets received by each client node (see Algorithm 2). Global information about the network is used to compute the innovative rate at each client. This leads to the selection of K NC nodes by iteratively computing the change that would maximize the increment in innovative rate at the clients. The value of K can be determined based on the maximum number of nodes that the network can support or the delay that the transmitted data can tolerate.

Algorithm 2 Centralized NC node selection

1: **for** i = 1 to K **do**
2: **for** each candidate node n_{test} in the set of SF nodes. **do**
3: Add n_{test} to the set of NC nodes.
4: Estimate the flow of innovative packets received by every client (Algorithm 1).
5: Remove n_{test} from the set of RNC nodes.
6: **end for**
7: Select among the different candidates the node n_{test} that maximize the innovative rate.
8: Add this node permanently to the set of NC nodes.
9: **end for**

2) Local information: The centralized approach above is probably unrealistic in large networks because it requires that a hypernode is aware of the network status and is able to track all packet replications. We therefore propose to distribute the node selection algorithm to address a scenario where each node only has a local view of the network. An algorithm similar to the centralized solution is applied in each node's neighbourhood in order to compute the benefit of replacing a SF node by a NC node. We assume that every node knows the total input capacity of the clients even if clients are not part of the node's subnetwork. Thus, the information about clients' total input capacity is propagated upwards on the network. Every node uses the received $p_i^m(k)$ and clients' input capacities $N_I(i)$ for computing its probabilities $p_i^n(k)$. This data are successively forwarded to the ancestor nodes. The above procedure is repeated till the servers are reached.

Note that, in a fully distributed scenario, each node independently decides whether it should be replaced by a NC node and the decision is taken by comparing the estimated benefit with a predetermined threshold value. In our implementation the decision is, however, greedy and centralized. Thus, each node independently computes the potential gain that arises if it turns into a NC node. These gains are sent to a central node that eventually decides about the location of the NC nodes. The algorithm used for selecting K network coding nodes is illustrated in Algorithm 3.

Algorithm 3 Distributed NC node selection

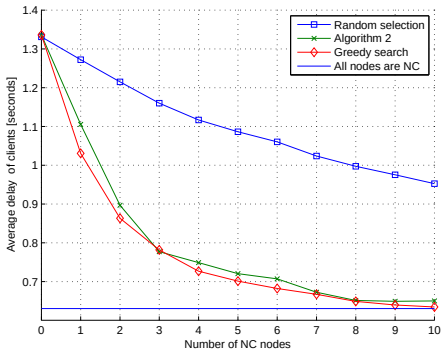
1: **for** i = 1 to K **do**
2: **for** every SF node n **do**
3: Estimate the flow of innovative packets received by every client when node n is not NC (computation is limited to the neighbourhood of n).
4: Temporarily transform node n into NC.
5: Estimate the new flow of innovative packets received by every client (computation is limited to the neighbourhood of n).
6: Compute benefit of transforming into NC.
7: **end for**
8: Globally select node with largest benefit and add it permanently to the set of NC nodes.
9: **end for**

IV. Simulation results

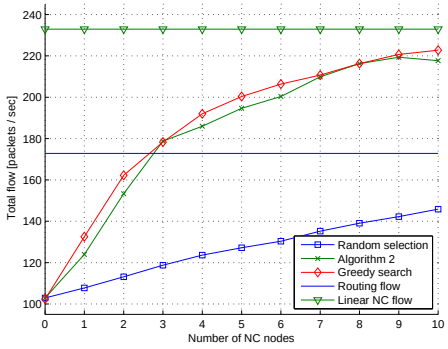
We evaluate here the performance of the proposed network coding node selection algorithms for multimedia streaming in irregular overlay networks using NS-3 [8] simulator. We create the irregular network topologies starting from regular graphs where nodes are grouped into coding stages, depending on the hop distance to the server. The regular topologies have the same number of nodes per coding stage and each node is connected to all nodes in the next coding stage. We consider regular graphs with 26 coding stages and 4 nodes per coding stage where all the nodes have the same number of parent and children nodes. Then we create irregular topologies by randomly pruning or shifting network links. Pruning simply consists in removing a link from the regular topology. Shifting consists in randomly changing the destination of some links, while making sure that cycles are avoided. The link pruning and shifting probabilities follow uniform distribution and the pruning and shifting rates are set to 5 %. We also force all peer nodes to have at least two incoming and two outgoing links, since path diversity is at the core of network coding solutions. Finally, the packet loss rate of each link is set to 5% and the capacities of the link follow uniform distribution in the range $[5, 20]$ packets/second.

The network coding operations are performed in a Galois field of size GF(256), and the size of a coding generation is 32 packets. The decoding is performed by gaussian elimination. We analyze the performance in terms of decoding delay. We compute the average delay as the time needed for each client to receive 32 linearly independent packets (i.e., a generation) in order to be able to decode the source information. The decoding times are, however, not included in the overall delay analysis as they mainly depend on the implementation. Finally, all simulation results are averages of 100 simulations.

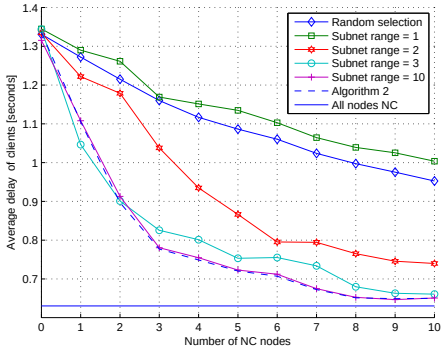
First, we compare the proposed centralized algorithm (Algorithm 2) with two methods that choose the position of 10 NC nodes in an overlay network consisting of 108 nodes by: (a) a scheme (greedy search) using Algorithm 2 but using instead of estimates the real network statistics and (b) randomly place the NC nodes. Fig. 2 (a) and (b) shows respectively the average delay times for each client and the effective throughput. From the results, it is obvious that the decay of decoding times and the corresponding increase of throughput is sharp for the first few selected NC nodes. The gains become less pronounced after the addition of 8 NC nodes. We can also see that the



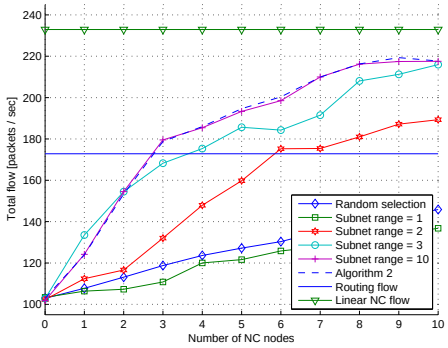
(a) Average client delay for the centralized scheme.



(b) Total multicast flow for the centralized scheme.



(c) Average client delay for the distributed scheme.



(d) Total multicast flow for the distributed scheme.

Fig. 2. Performance of the NC node selection algorithm for an irregular network of 108 nodes. Linear NC and routing flow correspond to the extreme cases where all nodes are NC and SF nodes respectively.

proposed algorithm performs similarly to the greedy exhaustive algorithm. The random replacement algorithm performs poorly and is comparable to the other methods only for large number of NC nodes. We also compare with the case when all nodes perform network coding. From Fig. 2(b) we observe that with 10 NC nodes the maximum throughput is almost achieved, but with a significantly smaller computational cost compared to a full network of NC nodes. Finally, the routing flow denotes the achievable multicast throughput when the network only contains SF nodes. We note that the throughput becomes smaller than that of the routing scheme when there are less than two NC nodes, which is due to the random forwarding policy of our scheme.

We also investigate the performance of the distributed solution from Algorithm 3, where the number of replicas is estimated locally, but the decision on the position of NC nodes is made globally. We analyze the influence of the size r (in hops) of the neighbourhood that represents the local view of each node. We use the same network settings as above. The average delays and throughput are presented in Fig. 2(c) and (d) respectively. From the results, it can be seen that when r is equal to one the distributed scheme performs similarly to the random selection scheme. In this case, the local network statistics are not sufficient for an effective selection of NC nodes. When r increases, however, the distributed solution performs close to the centralized algorithm.

V. Conclusions

We have presented a streaming system based on network coding for multimedia transmission in overlay mesh networks. In our scheme, only a few nodes perform RNC in order to meet a tradeoff between computational complexity, delay and packet duplicates. Two different algorithms are proposed for an effective placement of the NC nodes. The first algorithm assumes full network knowledge, while only local statistics are available for the second one. The experimental evaluation on irregular networks shows that both schemes could achieve the same throughput as a full network coding system with only a few well-positioned NC nodes.

References

[1] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung, "Network Information Flow," *IEEE Trans. Information 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. Information 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 Journal on Selected Areas in Communications*, 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. on Communication Control and Computing*, Monticell, IL, USA, Oct. 2003.

[5] T. Ho, M. Medard, J. Shi, M. Effros, and D. R. Karger, "On Randomized Network Coding," in *41st Allerton Annual Conference on Communication, Control, and Computing*, Monticello, IL, USA, Oct 2003.

[6] M. Wu, S. Karande, and H. Radha, "Network Embedded FEC for Optimum Throughput of Multicast Packet Video," *EURASIP Journal on Applied Signal Processing*, vol. 20, no. 8, pp. 728–742, Sep. 2005.

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

[8] "The network simulator - ns3," [Online], Available on web site `http://www.nsnam.org`.