RATELESS CODES NETWORK CODING FOR SIMPLE AND EFFICIENT P2P VIDEO STREAMING

Marco Grangetto, Rossano Gaeta, Matteo Sereno

Dipartimento di Informatica, Università degli Studi di Torino
Corso Svizzera 185, Torino - Italia
Email: `grangetto{gaeta,sereno}@di.unito.it`

ABSTRACT

The goal of this paper is the development of network coding solutions able to improve the performance of video streaming applications over peer-to-peer overlays. Recent advances in P2P protocols have shown that rateless codes can be profitably applied to P2P video streaming with several advantages in terms of protocol efficiency and simplification, e.g. push based video delivery, no need of packet reconciliation at the decoder. In this paper existing and novel network coding techniques based on rateless codes are presented and compared, showing that rateless codes, besides simplifying the protocol design, can significantly reduce the startup and playback delays. The proposed protocol is evaluated on real topologies, obtained crawling the widespread PPLive video streaming application. The reported experimental results show that the proposed protocol significantly reduces the startup and playback delay and allows one to increase the bitrate devoted to the video stream.

1. INTRODUCTION AND MOTIVATIONS

A number of P2P streaming applications such as Coolstreaming, SopCast, PeerCast, PPLive, TUVplayer, just to mention a few, are becoming increasingly popular and represent an innovative way to offer TV and video services over the Internet. The design of efficient and ISP friendly P2P video streaming solutions able to serve thousands or millions of users simultaneously is attracting a lot of research efforts and many issues remain to be solved. In particular, widespread solutions have been designed starting from the successful P2P file-sharing experience. The video stream is divided into chunks that are exchanged across the overlay with a pull based approach. The involved peers must signal updated information about the chunks they own so as to allow each participant to exploit parallel downloads. Delivery redundancy and reconciliation at the receiver are the solutions adopted to make the system robust to the overlay dynamics due to peers' churning.

The network coding paradigm [1, 2], where nodes in the network are allowed to combine information packets instead to simply forward them, can be exploited to tackle some of P2P video streaming open issues. As an example, in [3] random network coding is used to design a push P2P streaming algorithm without the need of explicit chunk requests. [4] recognizes that rateless codes [5] can be used as a very practical and efficient form of network coding for P2P data distribution. By means of rateless coding a set of k information packets can be recovered from a random set of $k(1+\epsilon)$ coded packets with ϵ approaching 0 for large values of k. Moreover, an arbitrarily large number of coded packets can be generated on the fly by the data source. In this work and in [4] a particular class of rateless codes, known as LT codes [6], has been used. The rateless coding principle can be used to eliminate the reconciliation issue since nodes can download coded packets relentlessly up to the collection of $k(1+\epsilon)$ packets, allowing the LT decoding and the retrieval of the corresponding k video chunks. Furthermore, using rateless codes makes the algorithm robust to peer churning since any packet sent by any node is equally important from the decoding point of view, thus making it easier to find useful information in a random overlay neighborhood. On the other hand, applying LT coding on k video packets, introduces a decoding delay that can impact on the real-time constraints of the video experience, e.g. excessive startup and playback delays. In [4] every peer in the overlay becomes a new source of LT coded packets as far as it has decoded the original k video chunks. This approach introduces a certain delay in data dissemination, which depends on the code block size k. Nonetheless, LT coding behaves optimally only for large value of k. In [7] a distributed coding technique is proposed by letting every node re-encode already coded symbols; this solution amounts to cascade several LT coding stages and is not very practical in terms of complexity and rate overhead. Another distributed rateless code design is presented in [8]; nevertheless, it is limited to the case of a single network topology with a common relay node, that can be generalized only assuming perfect knowledge of the overlay.

The major contributions of the present work are the proposal of a novel push protocol for LT coded packets distribution for P2P streaming with limited startup and decoding delays with respect to [4], and the analysis of the performance on synthetic and real P2P streaming topologies. The proposed protocol is based on LT encoding, coupled with an efficient relaying policy able to guarantee that every node in the network exploits an asymptotically optimal rateless code. Moreover, no prior knowledge of the network topology is assumed for protocol operations, thus making the solution very simple and quite general.

2. PROPOSED STREAMING PROTOCOL

In order to apply LT coding the video stream is divided into generations; the m-th generation is constituted by a sequence of k packets x_i^m, $i = 0, \ldots, k-1$, $m \geq 0$, of constant size L. In practical applications a generation corresponds to a video stream access unit i.e., the group of pictures used to encode the video frames. LT coding is performed within each generation, yielding an arbitrarily large number of the coded packets $y_j^m = \sum_{i=0}^{k-1} g_{i,j}^m x_i^m$, $j \geq 0$. The packet degree d, i.e. the number of information packets used to generated a coded packet, must follow the so called Robust Soliton Distribution (RSD) $\tau_{\delta,c}(d)$, $d = 1, \ldots, k$, where c is a suitable positive constant and δ is the allowed failure probability at the decoder [6]. Using the RSD guarantees that any set of $(1+\epsilon)k$ coded packets can be used to retrieve the k x_i^m using a simple message passing decoder. At any receiving node the coded packets y_j^m are buffered, waiting for degree 1 packets. As far as a degree 1 packet is observed, the corresponding video packet is decoded and it is used to lower the degree of all other coded packets, i.e. those already buffered and the new incoming ones. Iterating this procedure allows one to decode the k video packets with a limited overhead ϵ. It is worth pointing out that the packet indexes used to generate a coded packet, i.e. $i : g_{i,j}^m = 1$, must be provided to decoder. This is accomplished by using synchronized pseudo-random generators for the degree and packets selection on both the encoder and decoder side. To this end each peer is associated to a unique random seed, e.g. obtained hashing its IP/port or physical address, and every coded packets is identified by a sequential counter allowing to generate the same pseudo-random outcomes on the decoder side. As an example a 16 bit counter permits to signal the generating combinations of 65536 coded packets independently of the value of k and its rate overhead is negligible for reasonable packet sizes, e.g. hundreds of bytes. This approach is clearly more convenient with respect to explicit signalling of the $g_{i,j}^m$ which would cost k bits per packet.

The P2P network with N active peers can be modeled as a finite graph $\mathcal{T}$ of size N, where a vertex represents a peer and application-level connections between peers are modeled as edges. The source node $s \in \mathcal{T}$ is the content producer and aims at streaming the video to all the other peers in the overlay. To this end it is assumed that there is a path in $\mathcal{T}$ connecting s to any peer $p \in \mathcal{T} \setminus s$. For the purpose of this study we assume a static overlay topology; in particular, each $p \in \mathcal{T}$ has a static list of the nodes connected to its outgoing links.

The push protocol developed in [4] is used as the starting point and is referred to in the following as *store and encode* (SE). At startup s is the only data source and generates a number of coded packets y_j^0 saturating its upload capacity B_u. For simplicity no packet scheduling policy is assumed and s sends coded packets on all its outgoing edges in a round-robin fashion. Every receiver p waits for coded symbols (*store phase*) and executes progressive LT decoding. As soon as p is able to decode the first packet generation x_i^0, $i = 0, \ldots, k-1$, it starts generating novel and independently coded symbol (*encode phase*) behaving as a new source node. This is achieved by using its unique random generator. As a consequence, such new packets are independent (with high probability) from all other information being propagated in the overlay and can be used by the nodes still waiting for the generation $m = 0$, which can increase their download rate by exploiting the presence of multiple and independent sources on their incoming edges. In order to make our simulations realistic every peer p is characterized by a limited download capacity B_d and when the aggregated upload rate from its sources is larger than B_d the exceeding coded packets are dropped. In order to allow real time video streaming a node that has decoded the m-th generation act as a source for $\forall x_i^l, l \leq m$ and starts storing the generation $m+1$. Periodic signalling of the generation requested by a peer is assumed. The SE protocol can be interpreted as a form of network coding since every peer in the network independently encode and propagate coded information. Nevertheless, SE suffers from block decoding delay whereas one would like to let every node propagate coded information instantaneously.

In this paper the LT *relay and encode* (RE) protocol is proposed. The idea is to let every node propagate the received coded packets as soon as possible. A peer p waiting for the m-th generation is allowed to relay the received y_j^m on its outgoing edges with the only constraint that every y_j^m can be forwarded only once (*relay phase*). In absence of loops this simple rule guarantees that no duplicated packets will flow across the overlay links and assures that every set of coded data received by any $p \in \mathcal{T}$ carries optimal LT coded symbols. It is worth pointing out that such constraint limits the upload rate during the relay phase. As in the SE case when p is able to decode generation m it switches to the encode phase; thus it turns to be an independent source for packet generations $l \leq m$ and can concurrently start relaying coded packets of generation $m+1$. The relaying opportunity clearly improves the performance from the point of view of the propagation delay. The hypothesis on the absence of loops in the overlay can be relaxed by adopting a technique able to detect duplicated packets. Every peers stores the sources address of the incoming packets and each sender is associated to a ranking index; duplicated packets can be recognized by the associated sequential counter. Every time an original packet is received the sender ranking index is incremented; on the contrary, every duplicated packet decrements the same index. When the ranking index of a certain source is below a given threshold, i.e. the sender is pushing a lot of useless packets, such incoming link is pruned from the overlay.

3. EXPERIMENTAL RESULTS

The SE and RE protocols have been simulated by means of an event driven simulator. The protocols have been tested on both synthetic random graphs and real P2P streaming topologies. In the first case 30 instances of directed Erdős-Rényi (ER) random graphs [9] with $N = 10^4$ nodes have been considered; both the outgoing and incoming edges degree is modeled with a Poisson distribution whose average has been fixed to 20. The real topologies have been obtained from the popular PPLive streaming application. An active crawler able to gather topological information of PPLive channels has been developed [10] allowing us to capture snapshots of the overlay supporting a PPLive channel. The size of captured snapshots varies according to a daily behavior ranging from 4000 to 8000 peers, with an average degree of 69. All the results reported in the following are averaged on 30 graph instances with randomly selected sources. Peers' upload and download capacities are subdivided into 3 classes: $B_u = B_d = 100$ Mbps, $B_u = B_d = 10$ Mbps and ADSL nodes with $B_u = 512$ kbps, $B_d = 7$ Mbps. Analyzing PPLive snapshots one notices that high capacity nodes are characterized by a large number of outgoing links. Therefore, bandwidth classes have been allocated depending on the number of edges departing from a peer so as to achieve a realistic distribution; the first two symmetric classes cover less than 2% and about 7% of peers respectively, whereas 91% of the nodes are ADSL. The node s is always included in the highest capacity class, regardless of the number of outgoing links.

LT coding block size has been fixed to $k = 1000$ with $L = 1000$ bytes, as a compromise between complexity/delay and coding overhead. Random packet combinations are generated with two RSD settings, $c = 0.05$, $\delta = 0.0001$ and $c = \delta = 0.01$ yielding overheads $\epsilon = 0.38$ and $\epsilon = 0.17$, respectively.

For each node p that is reachable from the source s in h hops the simulator estimates the time instant $t_D^m(p, h)$ when LT decoding of the m-th generation is fully accomplished. In order to rank the protocol behavior as a function of the distance from the source the average decoding delay at distance h can be computed as $t_D^m(h) = \sum_{p \in \mathcal{T}_h} t_D^m(p, h)/|\mathcal{T}_h|$, where $\mathcal{T}_h$ is the subset of nodes in $\mathcal{T}$ h hops away from s. It is worth pointing out that previous measurements for $m = 0$ represent the start-up delay. Finally, from inter-arrival times of the packet generations one can estimate the maximum throughput that can be offered to the video streaming application:

$$R^m(h) = \frac{Lk}{T_D^m(h) - T_D^{m-1}(h)}$$

In Fig.1 $T_d^m(h)$, $m = 0, \ldots, 9$ for the ER graphs is reported for both SE and RE protocols. The simulation results show that using RE one can remarkably lower the startup delay. As an example, for $h = 2$, RE and SE startup delays are about 50 s and 400 s, respectively. Moreover, RE decoding times do not increase much as a function of h. In Fig.2 the value of $R^m(h)$ averaged over 10 generations is shown for $\epsilon = 0.38$ and $\epsilon = 0.17$. RE is able to supply about 300 kbps to all peers in the overlay, whereas SE rate is limited to about 100 kbps. The rate gain of RE can be clearly exploited to support higher quality video streaming. It can be noticed that $R^m(0)$ is very high for both protocols since it represents the download rate experienced by nodes directly connected to s.

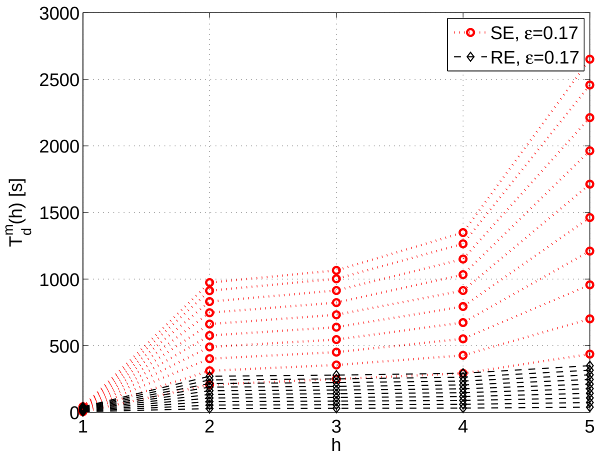

Fig. 1. $T_d^m(h)$ for ER random graph, $\epsilon = 0.17$.

Fig.3 and Fig.4 show the same kind of experimental results for the real PPLive snapshots. The collected PPLive snapshots are far less homogeneous than ER random graphs. In particular, 95% of the nodes is within 3 hops from s, forming a highly connected and clustered subgraph, whereas the remaining 5% is represented by weakly connected nodes. This observation explains why the RE performance is very similar to the one reported for ER up to $h = 3$ while RE and SE are not very effective for $h > 3$. Nevertheless, it is worth noticing that these latter nodes represent a very limited percentage of the overlay. For $h = 2$, RE startup delay turns out to be about 30 s, whereas SE is approximatively 3 times slower. According to Fig.4 RE almost doubles the supported video rate with respect to SE. Finally, comparing the achieved rate for the two values of LT coding redundancy, it turns out that in the presented scenario protocol optimization yields larger improvement than code optimization.

4. CONCLUSIONS AND FUTURE WORK

In this paper a simple and efficient algorithm for P2P video streaming based on rateless coding has been proposed. Every peer in the overlay relays and, as soon as possible, generates new and independent coded packets according to the

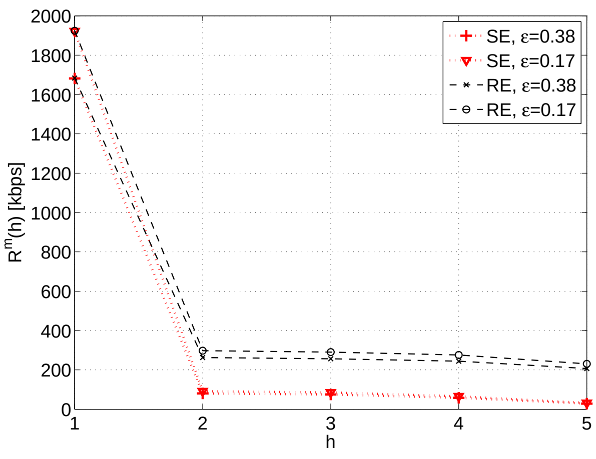

Fig. 2. $R^m(h)$ for ER random graph.

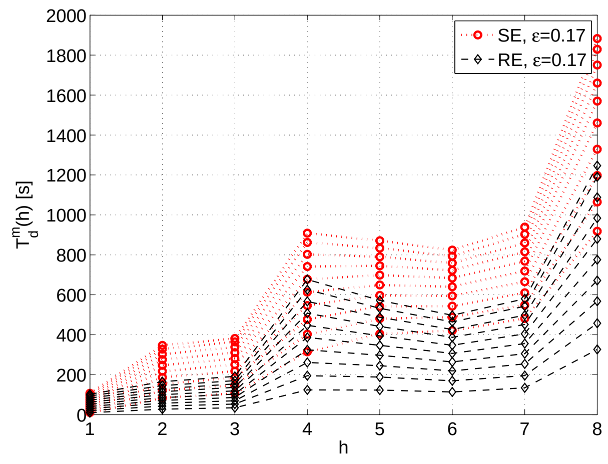

Fig. 3. $T_d^m(h)$ for PPLive overlay, $\epsilon = 0.17$.

network coding principle. As a consequence, the reception phase is very simple and amounts to retrieve a sufficiently large set of coded information without the need of explicit requests and reconciliation. The proposed RE protocol aims at reducing the side effect introduced by rateless coding, namely the startup and decoding delays caused by block coding. The proposed solution improves on previous results in the field, exhibiting limited delays and supporting higher bitrates. Future works include the design of optimal scheduling policies for the relay phase, the analysis of the protocol behavior in presence of peer churning and the deployment of a complete streaming application.

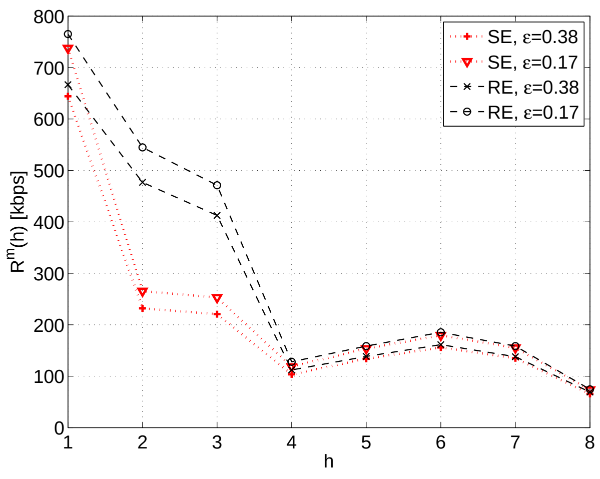

Fig. 4. $R^m(h)$ for PPLive overlay.

5. REFERENCES

[1] R. Koetter and Medard M., "An algebraic approach to network coding," *IEEE/ACM Transactions on Networking*, vol. 11, no. 5, pp. 728–795, Oct. 2003.

[2] C. Gkantsidis and P. Rodriguez, "Network coding for large scale content distribution," in *IEEE INFOCOM 2005*, Mar. 2005.

[3] M. Wang and Li Baochun, "R^2: Random push 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] Wu Chuan and Li Baochun, "rStream: resilient and optimal peer-to-peer streaming with rateless codes," *IEEE Transactions on Parallel and Distributed Systems*, vol. 19, no. 1, pp. 77–92, Jan. 2008.

[5] M. Mitzenmacher, "Digital Fountains: A Survey and Look Forward," in *IEEE Information Theory Workshop*, October 2004, pp. 271–276.

[6] M. Luby, "LT codes," in *IEEE Symposium on Foundations of Computer Science*, November 2002, pp. 271–280.

[7] R. Gummadi and R.S. Sreenivas, "Relaying a fountain code across multiple nodes," in *IEEE Information Theory Workshop, 2008*, May 2008, pp. 149 – 153.

[8] S. Puducheri, J. Kliewer, and T.E. Fuja, "The design and performance of distributed LT codes," *IEEE Transactions on Information Theory*, vol. 53, no. 10, pp. 3740–3754, Oct. 2007.

[9] M. E. J. Newman, S. H. Strogatz, and D. J. Watts, "Random graphs with arbitrary degree distributions and their applications," *Physical Review E*, vol. 64.

[10] S. Spoto, R. Gaeta, M. Grangetto, and Sereno M., "Analysis of PPLive through active and passive measurements," in *International Workshop on Hot Topics in Peer-to-Peer Systems*, 2009.