A Theory-Driven Distribution Algorithm for Peer-to-Peer Real Time Streaming

Lorenzo Bracciale*, Dario Luzzi*, Francesca Lo Piccolo*
Nicola Blefari Melazzi*, Giuseppe Bianchi*, Stefano Salsano*

*Universitá di Roma "Tor Vergata"
Rome, Italy
{lorenzo.bracciale,francesca.lopiccolo,dario.luzzi,blefari,giuseppe.bianchi,stefano.salsano}@uniroma2.it

***Abstract*—Many distribution algorithms have been proposed up to now for P2P real time streaming. However, due to the lack of basic theoretical results and bounds, common sense and intuitions and heuristics have driven their design so far. The consequence is that we can find in the literature a large variety of different choices about the main aspects of a P2P system, such as overlay topology, scheduling process and upload strategy. In this situation, it is difficult to establish unambiguously the absolute goodness of a particular algorithm or even the rationale behind a particular choice or solution.**

In this paper we propose and evaluate a theory-driven distribution algorithm for P2P real time streaming. We take advantage from a previous theoretical study, where: i) we derived a theoretical performance bound for forest-based overlay topologies regarding the number of nodes reachable in a given time interval or equivalently the time required to reach a given number of nodes; ii) we proved the optimality of *Streamline*, a distribution algorithm based on the serial transmission over forest-based topologies, in terms of its capability to reach such a bound.

The *Streamline* algorithm is based on some ideal assumptions that prevent its practical implementation. In this paper we remove these assumptions and present a practical and working algorithm, named Operational Streamline or simply *O-Streamline*. We also evaluate the performance of *O-Streamline*, comparing them with the optimal bounds of *Streamline*.[1]

I. Introduction

Peer-to-peer (P2P) overlay systems are being proposed to stream multimedia audio and video content from a source to a large number of end-users. The basic idea is to divide stream data in so-called "chunks", and to organize peer nodes in an overlay distribution network to relay chunks.

To achieve this goal, a plethora of very different distribution algorithms have been proposed up to now. These algorithms make specific and different assumptions and choices about the main aspects of a P2P real time streaming system, such as overlay topology, scheduling process and upload strategy. However, the actual literature is somehow confusing, as the reader can not always understand the rationale behind the choices driving the algorithm design. As a matter of fact, there are algorithms that make different choices about, for instance, the scheduling process, even if the application scenario is the same. Or there is an algorithm that combines the overlay topology X with the upload strategy Y, while another one combines the same overlay topology X with another upload strategy Z. In many such cases it is not easy to understand the reason of these choices and, most important, it is very difficult to compare different algorithms as it would be necessary to implement/simulate them. In addition, the measured/simulated results of literature proposals are strongly affected by the used test-bed or custom-made simulator and by the considered evaluation scenarios, in absence of theoretical properties guiding the system design.

In our opinion, the main reason why so different approaches have been proposed up to now is the lack of baseline theoretical results and bounds on P2P real time streaming. In absence of theoretical results, common sense and intuitions and heuristics have driven the design of the proposed distribution algorithms.

Another consequence of the lack of basic theoretical results and bounds is the impossibility to establish unambiguously the absolute goodness of a P2P streaming distribution algorithm. In other words, while the performance of different distribution algorithms may be (difficulty) compared, it is not possible to evaluate how far/close the performance of a distribution algorithm are from/to optimal performance. By optimal performance we mean the ability to reach the greatest possible number of nodes in a given time interval or equivalently it makes possible to reach a given number of nodes in the smallest possible time interval.

To provide evidence of what we argue above, in the following we list the main alternatives proposed in the literature for overlay topologies, scheduling processes and upload strategies.

As regards the overlay topology, it is possible to distinguish among:

- tree-based solutions, such as NICE [1] and ZIGZAG [2], where nodes are organized in a tree and content is recursively spread from the parent node (the streaming source) to its child nodes until all peers are reached;
- mesh-based solutions, such as CoolStreaming [3], Gridmedia [4] or PRIME [5], where each node randomly establishes overlay connections with other nodes and sends a received chunk only to neighbors still missing

[1]This work was supported by the Italian Ministry of University and Research (MiUR), with the Grant PRIN-2006099023 "Profiles" (disi.unitn.it/profiles).

that chunk, in such a way that each chunk is streamed on a different tree;

- forest-based solutions, such as CoopNet [6] or SplitStream [7], where chunks and nodes are logically organized in a finite set of groups and distribution trees, in such a way that each distribution tree includes all nodes and is used to distribute a single group of chunks. Nodes are interior nodes only in one tree and leaf nodes in all the remaining trees. The idea is to exploit the upload capacity that leaf nodes do not use in separate distribution trees, thus increasing the overall transmission capacity of the overlay network.

As regards the scheduling process, it is possible to distinguish among:

- push-based algorithms, such as [1] and [2], where supplier nodes decide which chunks will be served to which neighbors;
- pull-based algorithms, such as [3] or [5], where scheduling decisions are taken at receivers and a chunk is transmitted only if a receiver requests that chunk;
- hybrid push-pull algorithms, like [4], where chunks are requested in pull mode at start up and relayed in push mode in the immediate following phase.

Moreover, different local scheduling policies have been proposed: for instance, giving priority to the chunks with more stringent playback deadline [3], to the rarest chunks [8], to the neighbors with the highest upload/download capacity [3][9].

As regards the upload strategy, when the same chunk has to be uploaded to more than one child node, it can be transmitted "in series", e.g. starting the transmission towards a second child node after completion of the chunk upload to the first child, or "in parallel", i.e. sending the same chunk to more than one child node at the same time.

In this paper, we try to overcome the limitations discussed so far by proposing a theory-driven distribution algorithm for P2P real time streaming. Specifically, we take advantage from the theoretical study in [10], where i) we derived a theoretical performance bound for forest-based overlay topologies regarding the number of nodes reachable in a given time interval or equivalently the time required to reach a given number of nodes; ii) we proved the optimality of *Streamline*, a distribution algorithm based on the serial transmission over forest-based topologies, in terms of its capability to reach such a bound. However, even if *Streamline* is able to achieve optimal performance, it is not a functioning algorithm in the sense that it is based on assumptions usually not verified in real scenarios. In other words the aim of *Streamline* is to set the framework of what is doable and achievable. Then, to realize it in practice, it is necessary to remove some ideal assumptions. This is what we are going to do in this paper.

In more detail, the assumptions made in the *Streamline* proposal are:

- the distribution topology is derived thanks to a sort of Maxwell's demon that holds a global and centralized vision of the whole network. This conceptual entity is able to instruct peer nodes on how to optimally organize the overlay topology and schedule chunk transmissions;
- there is no churn.
- we do not take into account propagation delays induced by the underlying physical network topology (practically speaking, we assume that the propagation delays are negligible when compared with the chunk transmission time, or in other words, that chunks are not very small; indeed an assumption valid for most of the practical P2P streaming systems).
- we evaluate the performance by assuming that the ratio between the uplink capacity of peer nodes and the stream bit rate is equal to one.

In this paper, we propose a practical and working algorithm, named Operational Streamline or more simply *O-Streamline*. *O-Streamline*: i) does not rely on a centralized vision of the whole network ii) takes the churn into account, iii) does not assume that all peers have an uplink capacity equal to the stream bit rate. As in *Streamline* we do not take into account the signalling burden. This is left for future work also to make possible to evaluate the effect of removing ideal assumptions step by step. Otherwise, it would be difficult to assess the implication of each of this assumption on the overall system performance.

Finally, we evaluate, by means of simulations, the performance of *O-Streamline*, and we show that such performance are close to the optimal bounds derived for *Streamline*. To the best of our knowledge, this is the first work presenting a theory-driven distribution algorithm for P2P real time streaming.

The paper is organized as follows. Section II briefly recalls the basic concepts of *Streamline* and the main theoretical results presented in [10]. Section III describes our proposal. In Section IV we evaluate the performance of our algorithm. Finally, Section V concludes the paper.

II. Basic concepts of Streamline and theoretical results

In this section, we recall some basic concepts of *Streamline* and the main theoretical results presented in [10], to better understand *O-Streamline*.

Streamline organizes peers and chunks in a finite number of overlay trees and groups respectively, in such a way that: i) each tree includes all peers and ii) chunks in a group are always transmitted by using the same overlay tree, iii) for each tree, and thus for each chunk, the source node sends the chunk to its children in series, and the same holds for each peer node of the tree, excluding the leaves.

Streamline shares with CoopNet [6] or SplitStream [7] the advantage of exploiting in separate distribution trees the upload capacity that leaf nodes do not use, thus increasing the overall transmission capacity of the overlay network. However, while CoopNet and SplitStream use a parallel transmission of chunks, *Streamline* resorts to a serial transmission of chunks.

The idea behind this kind of distribution is that giving all the uplink capacity to a single child, instead of sharing it among different children, allows that child to start serving other nodes before than in the case of parallel transmission. This means that all nodes, except the last served ones, can start serving their children before than in the case of parallel transmission; in turn, served children can become fathers of other children before than in the case of parallel transmission, and so on. We note that, from the point of view of a given node, serial and parallel transmissions are the same, i.e. there is not a particular advantage, for a given node; however, if we look at the system on the whole, we will show that serializing the chunk distribution brings about significance performance advantages in terms of number of nodes reached in a given time interval or equivalently in terms of time required to reach a given number of nodes.

Streamline assumes the following reference framework. A stream is generated by a single node (source) at constant rate R (Kbps) and segmented into chunks with fixed size L (Kb). The source starts transmitting at time instant $t = 0$. We define $T = L/R$ (s) the chunk duration, which is also the time elapsing between two consecutive chunks. Besides to the source, the network is composed of an unlimited number of nodes, which join the system simultaneously, are always on and are assumed to be homogeneous in terms of uplink capacity (equal to B Kbps) and downlink capacity. The downlink capacity is assumed to be great enough so that downlinks are not a bottleneck of the system. Each node has k overlay parents and k overlay children, being the parent set potentially different from the children set.

Streamline considers the following performance indexes:

- $N(c, k, t)$: the number of peer nodes that can complete the download of the c-th chunk after t time units, being k the number of parents/children;
- $T(c, k, n)$: the amount of time needed by n peer nodes to complete the download of the c-th chunk, being k the number of parents/children.

The previous performance indexes have been derived in [10] under the assumption that the ratio between the uplink capacity of peer nodes and the stream bit rate is equal to one, i.e., $B = R$. This is the most demanding assumption, in a streaming scenario. Furthermore, the previous performance indexes have been normalized with respect to the time needed to download a complete chunk at rate B (in bit/s). Thus, the latter time interval will be the unit of time. In addition, for the sake of simplicity, t is always assumed to be an integer value: the generalization to continuous time is rather straightforward.

In [10] we show that the following results hold:

$$N(c, k, t) = \sum_{i=1}^{t-c+1} F_k(i) \tag{1}$$

where $F_k(\cdot)$ is the k-step Fibonacci sequence, defined as follows

$$F_k(i) = \begin{cases} 0 & \text{if } i \leq 0 \\ 1 & \text{if } i = 1 \\ \sum_{j=1}^{k} F_k(i-j) & \text{if } i > 1 \end{cases} \tag{2}$$

As regards $T(c, k, n)$, it can be obtained thanks to the following implicit relation:

$$T(c, k, n) = \min\{t \geq c : \sum_{i=1}^{t-c+1} F_k(i) \geq n\} \tag{3}$$

In addition, and most importantly, in [10] we show that *Streamline* is optimal among all distribution algorithms for forest-based topologies, in the sense explained in the Introduction.

Finally, thanks to some results on k-step Fibonacci sums, the following asymptotic expression, with respect to t, for $N(c, k, t)$ has been derived:

$$N(c, k, t) \approx u(t-c) \left[\frac{\phi_k}{\phi_k - 1} \cdot \frac{\phi_k^{t-c+1}}{Q_k(\phi_k)} - \frac{1}{k-1} \right] \tag{4}$$

where i) $u(\cdot)$ denotes the unit step function, ii) ϕ_k represents the so said k-step Fibonacci constant and it is the only real root with modulo greater than 1 of the characteristic polynomial $P_k(x) = x^k - x^{k-1} - x^{k-2} - \cdots - x - 1$ of the k-step Fibonacci sequence, and iii) $Q_k(x)$ is a suitable polynomial about which more details can be found in [10]. The asymptotic expression with respect to t in (4) makes possible to state that the performance of *Streamline* depends on the number of deployed trees through the Fibonacci constant values ϕ_k; as a matter of fact, Fibonacci constants ϕ_k very rapidly tend to the limit value 2 (e.g. ϕ_4 is already 1.96595).

III. From theory to practice: our proposal

In this section, we describe our proposal to turn *Streamline* into *O-Streamline*, a practical and working distribution algorithm for P2P real time streaming.

O-Streamline tries to maintain the two basic principles of *Streamline*, that are i) the organization of peers and chunks in a finite number of overlay trees and groups and ii) the serialization of chunk transmissions.

The second principle is easy to be put in practice. The first principle involves two distinct assumptions: i) the availability of a global and centralized vision of the whole network; ii) the absence of peer churn.

As regards the first assumption, *O-Streamline* does not count anymore on a global and centralized vision of the whole network, but it builds the overlay topology in a distributed manner and schedules chunk transmissions with a limited, local view of the overall topology.

O-Streamline constructs the overlay topology as follows. Peer nodes are uniformly divided into G different groups, in such a way that i) each node belongs only to one group, ii) each node establishes random overlay bidirectional connections with P peer nodes in the same group to which it belongs and O peer nodes in each of the remaining groups. This can be easily achieved in a real distributed environment. Provided that groups are progressively identified starting from 0, each node may establish its membership group if it extracts an (integer) random number with uniform distribution in the interval $[0, G-1]$. In addition, it may also establish connections with other peer nodes if a bootstrap node provides it with the addresses of the already active nodes in each group. The G groups are also used to organize the stream chunks. In more detail, if chunks are progressively indexed starting from 1, chunk i is associated with the group g such that $g = i \bmod G$.

To describe how *O-Streamline* schedules chunk transmission we let:

- L the chunk size (in bits)
- U the ratio between the stream bit rate R and the uplink capacity B of all nodes;
- $T = L/R$ the chunk duration (in s).

The scheduling algorithm operates as follows:

- the source node, which does not belong to any of the groups and it is connected to U neighbors per each group, sends the generic chunk in series to the U neighbors of the corresponding group. The source node serves the chunks to the neighbors of each group following the order implicitly established in the association between chunk identifiers and groups. The source node repeat this distribution pattern, modulo G, unless churn changes the overlay neighbors. Therefore each node which is child of the source receives a new chunk from the source every $G \times U \times \frac{L}{B}$ seconds;
- the generic peer relays only the chunks of the group it belongs to. A FIFO queue is used to manage the chunks of the group to which the generic peer belongs;
- the generic peer is interested only in chunks that arrive at the source, starting from the time instant at which the peer joins the system. This implies that peer nodes must be somehow synchronized with the timing reference of the stream. However, this problem can be solved if the bootstrap node is the source, and sends to the new peer nodes joining the system the identifier of the next chunk that is going to be transmitted by the source; the case in which the bootstrap node is not the source is left for future work;
- the generic peer serves each chunk in the FIFO queue in series to $G \times U$ neighbors (if there are $G \times U$ neighbors missing that chunk), by giving priority to the neighbors of its own group. This implies that the number of neighbors P has to be at least equal to $G \times U$. Moreover, whenever possible, the generic peer serves the chunks belonging to the same group to which the peer belongs in the same order;
- if a peer ends serving a chunk to $G \times U$ neighbors and there are no new chunks to be served in the FIFO queue, it tries to serve that chunk to other neighbors, until a new chunk to be served is enqueued in the FIFO queue.

As regards the second assumption, *O-Streamline* takes churn into account as follows. In fact, the numbers P and O of neighbors per group are set to be greater than $G \times U$ so that churn can be handled smoothly. In more detail, if peer nodes loosing a neighbor due to a disconnection have less neighbors than $G \times U$ in the group to which the disconnecting node belongs, they may look for another neighbor and maintain $G \times U$ as minimum number of connection in that group. In such a way, *O-Streamline* does not need to re-configure the whole overlay topology in case of peer churn. In the next section we describe how we set P to achieve this result in our simulations.

IV. Simulation results

In this section we evaluate the performance of *O-Streamline* by considering the following performance indexes:

- $\overline{N}_{served}(c, d)$: is the average number of nodes that complete the download of a chunk c with a chunk delivery delay less than or equal to d; the chunk delivery delay is the difference between the time instant at which the chunk arrives at the source and the time instant at which the chunk is completely received from a node. Note that this performance index is very similar to the performance index $N(c, k, t)$ used to evaluate the performance of *Streamline*. In fact, if in $N(c, k, t)$ we consider the chunk delivery delay $t - c - 1$ instead of the absolute time t of chunk delivery and we average on all chunks, we obtain $\overline{N}_{served}(c, d)$;
- chunk delivery ratio $R(n, S_n)$: with reference to a node n, the chunk delivery ratio is the ratio between the number of (completely) received chunks and the total number of chunks generated during the duration of the session of that node, S_n.

We evaluated these performance indexes via simulations by using the OPSS simulator [13]. The duration of the simulations is 1800 s. For each simulation result we assessed the 95% confidence intervals. These intervals are plotted in the figures only when they are significant; when they are small (e.g. less than 5%) they are not shown to improve the readability of the figures themselves.

We simulated a homogeneous scenario in terms of downlink and uplink capacity: all nodes, including the source, have the same uplink and downlink capacity. As regards the downlink capacity, like in *Streamline*, we set a value great enough so that downlinks are not a bottleneck of the system. As regards the uplink capacity, we remove another assumption of *Streamline*. *Streamline* assumes that the ratio between the uplink capacity of all nodes and the the stream bit rate is equal to 1. In *O-Streamline* we consider also values of this ratio greater than or equal to 1. As a consequence, when this ratio is greater than one, we had to use the OPSS simulator also to evaluate the performance of *Streamline*. As regards the churn model we make some assumptions, which must not be seen as too oversimplified, since the main goal of this paper is to present a practical and working theory-driven algorithm and to test its robustness against churn rather than to give a real model of churn. These assumptions are:

- the session times are exponentially distributed;
- when a peer node leaves the system, a new peer node joins the system. This makes possible to keep the number of simulated peer nodes constant throughout the whole

simulation and to simplify the evaluation of simulation results;

- each node may accept up to X connections per group besides the number P or O of connections per group[2]. This guarantees a topology perturbation after node disconnections. In fact, accepting more connections than the minimum number P or O, implies that new peer nodes may be placed in topological positions different from the ones of the corresponding peer nodes which are going to disconnect and to be replaced. In addition, also peer nodes loosing a neighbor due to a disconnection may look for another neighbor and maintain P or O (depending on the group to which the disconnecting node belongs) as minimum number of connection per group.

The presentation of simulation results is organized as follows. First, we refer to a churn-free simulation scenario where peer nodes are always on (subsection IV-A). This allows us to compare *O-Streamline* with *Streamline*, thus understanding *O-Streamline*'s position with respect to ideal performance bounds. Then, we introduce the churn and evaluate its implications on the performance of *O-Streamline* (subsection IV-B).

A. Comparison with Streamline, no churn

We assume that the number P of neighbors in the same group to which the generic peer belongs is equal to the number O of neighbors in the remaining groups.

We first consider a scenario with 11504 nodes, 8 neighbors per group ($P = O = 8$) and a ratio between uplink capacity and stream bit rate equal to 1.

In figure 1 we plot the cumulative distribution function of $\overline{N}_{served}(c, d)$ for three different values of the number G of groups. The curve relative to *Streamline* has been analytically derived by setting $k = 4$. In fact, even if the performance of *Streamline* improve monotonically as a function of the number k of children, we chose $k = 4$ since the improvement becomes negligible for greater values of k (see [10] for more details). We observe that as the number of groups increases, the performance of *O-Streamline* get closer to the ones of *Streamline*. For instance, with 4 groups, the difference between *O-Streamline* and *Streamline* is negligible for chunk delivery delay up to $12 - 13$ seconds; the maximum chunk delivery delay necessary to serve the whole network of *O-Streamline* is 3 seconds higher than that of *Streamline*. With 3 groups the maximum chunk delivery delay of *O-Streamline* does not worsen, while with 2 groups the maximum chunk deliver is 6 seconds higher than the maximum chunk delivery of *Streamline*.

As a consequence, we can state that there exists a threshold for the number of groups G over which improvements become negligible. In addition the number of groups G should be chosen so as to limit the signaling burden required to maintain the relationships with neighbors belonging to all the G groups.

[2]In all the simulations X is a random variable with uniform distribution in the interval (0,2).

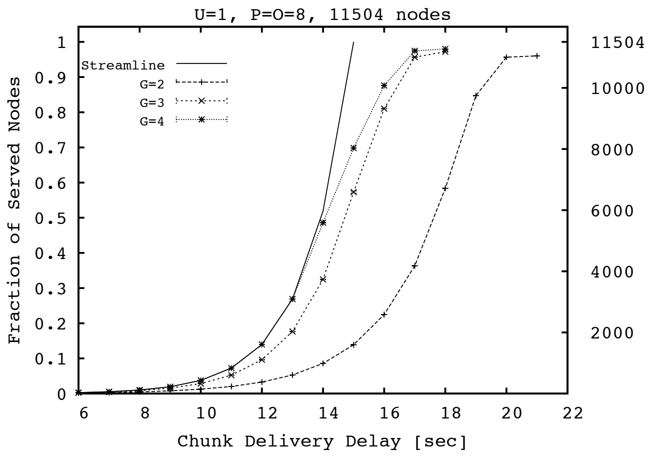

Fig. 1. Cumulative distribution function of $\overline{N}_{served}(c, d)$ in *Streamline* and *O-Streamline* in case of a network with 11504 nodes, ratio between uplink capacity and stream bit rate equal to 1 and 8 neighbors per group, for three different values of the number of groups.

In figure 2 we plot the inverse cumulative distribution function[3] of $R(n, S_n)$ (the chunk delivery ratio), for three different values of the number G of groups. As in the previous figure, the curve relative to *Streamline* has been analytically derived by setting $k = 4$. We also note that the chunk delivery ratio in *Streamline* is equal to 1 for every chunk and every node. As expected, we observe that *Streamline* succeeds in completely diffusing all chunks. Instead, *O-Streamline* has a chunk delivery ratio that increases with the number of groups. For instance 95% of nodes download at least 80%, 85% and 90% of all chunks with 2, 3 and 4 groups, respectively.

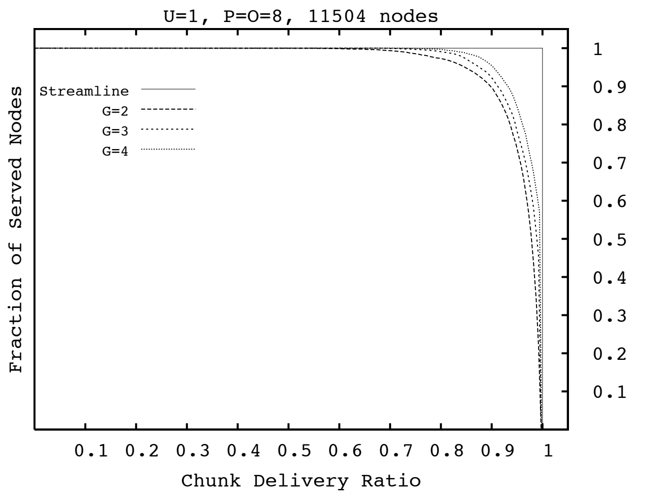

Fig. 2. Inverse cumulative distribution of chunk delivery ratio in *Streamline* and *O-Streamline* in case of a network with 11504 nodes, ratio between uplink capacity and stream bit rate equal to 1 and 8 neighbors per group, for three different values of the number of groups.

Now we consider a second scenario with 17472 nodes, 2

[3]i.e. the complement to one of the cumulative distribution function

groups and a ratio between uplink capacity and stream bit rate equal to 2.

In figure 3 we plot the cumulative distribution function of $\overline{N}_{served}(c,d)$ for three different values of the number of neighbors per group (we recall that we assumed $P = O$). The curve relative to *Streamline* has been derived via simulations under the assumption of 4 parents/children per node.

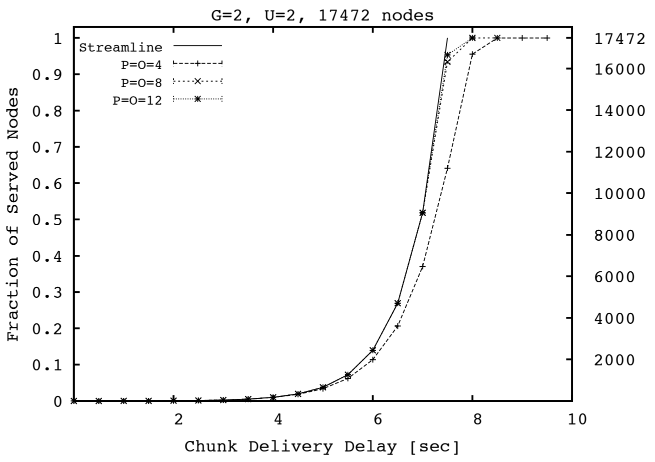

Fig. 3. Cumulative distribution function of $\overline{N}_{served}(c,d)$ in *Streamline* and *O-Streamline* in case of a network with 17472 nodes, ratio between uplink capacity and stream bit rate equal to 2 and 2 groups, for three different values of the number of neighbors per group.

We observe that as the number of neighbors per group increases, the performance of *O-Streamline* get closer to the ones of *Streamline*. For instance, with 12 neighbors per group, the difference between *O-Streamline* and *Streamline* is negligible for chunk delivery delay up to 6 seconds; the maximum chunk delivery delay necessary to serve the whole network of *O-Streamline* is 1 second higher than that of *Streamline*.

Another observation is that the performance of *O-Streamline* get closer to the ones of *Streamline* as the ratio between uplink capacity and stream bit rate increases. As a matter of fact if we compare figure 1 and figure 3 for the cases of $G = 2$ and $P = O = 8$ the maximum chunk delivery delay is 21 s for $U = 1$ and 8.5 s for $U = 2$.

Finally, we note that we do not present results on the chunk delivery ratio in this scenario, since the choice of $U = 2$ guarantees the complete diffusion of all chunks, regardless of the number of neighbors per group.

B. Impact of churn

We consider a scenario with 17472 nodes and a ratio between uplink capacity and stream bit rate equal to 2. The average session time of each node is equal to 10 minutes.

In figure 4 we plot the cumulative distribution function of $\overline{N}_{served}(c,d)$ for three different values of the number G of groups. The number of neighbors per group is 8 ($P = O = 8$). We observe that: i) the performance worsen with respect to what happens in absence of churn, ii) the lower the number of groups, the more significant the worsening, iii) a single group implies a maximum chunk delivery delay of about 25 seconds; using 2 or 3 groups makes possible to reduce this delay to about $16 - 17$ seconds.

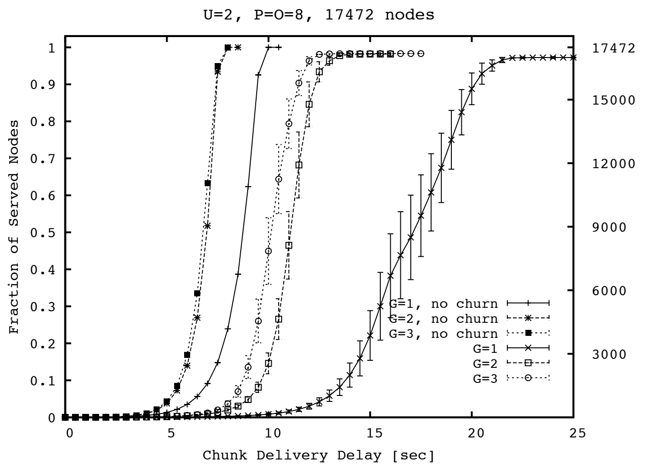

Fig. 4. Cumulative distribution function of $\overline{N}_{served}(c,d)$ in *O-Streamline* in case of a network with 17472 nodes, ratio between uplink capacity and stream bit rate equal to 2 and 8 neighbors per group, for three different values of the number of groups.

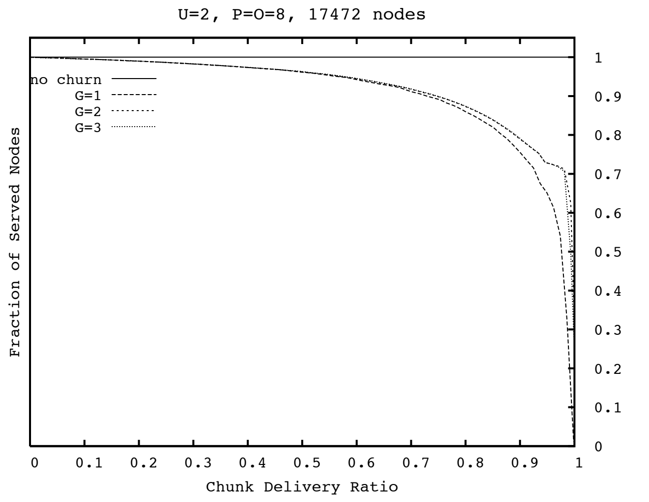

Fig. 5. Inverse cumulative distribution of chunk delivery ratio in *O-Streamline* in case of a network with 17472 nodes, ratio between uplink capacity and stream bit rate equal to 2 and 8 neighbors per group, for three different values of the number of groups.

In figure 5 we plot the inverse cumulative distribution function[4] of $R(n, S_n)$ (the chunk delivery ratio), for three different values of the number of groups. The number of neighbors per group is 8 ($P = O = 8$). Also this performance index improves as a function of the number of groups.

In figure 6 we plot the cumulative distribution function of $\overline{N}_{served}(c,d)$, for three different values of the number of neighbors per group (we recall that $P = O$). We observe that: i) performance worsen with respect to performance in

[4]i.e. the complement to one of the cumulative distribution function

absence of churn, ii) increasing the number of neighbors per group improves mostly the maximum chunk delivery delay and only marginally the lower values of chunk delivery delay. We remark that increasing the number of neighbors per group leads to an increment of the signalling burden required to mantain the relationships.

In figure 7 we plot the inverse cumulative distribution function[5] of $R(n, S_n)$ (the chunk delivery ratio), for three different values of the number of neighbors per group (we recall that $P = O$). The variation of the number of neighbors per group does not affect significantly the performance.

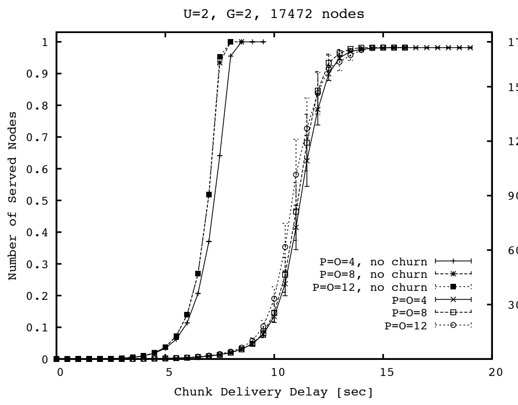

Fig. 6. Cumulative distribution function of $\overline{N}_{served}(c, d)$ in *O-Streamline* in case of a network with 17472 nodes, ratio between uplink capacity and stream bit rate equal to 2 and 2 groups, for three different values of the number of neighbors per group.

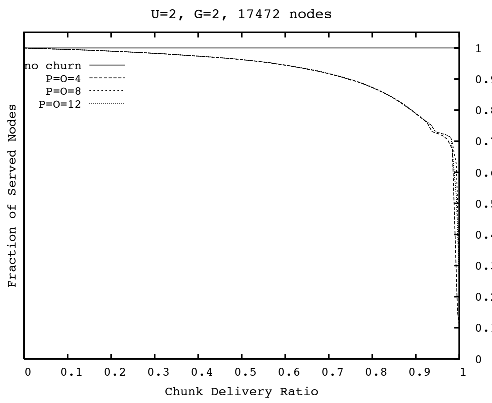

Fig. 7. Inverse cumulative distribution of chunk delivery ratio in *O-Streamline* in case of a network with 17472 nodes, ratio between uplink capacity and stream bit rate equal to 2 and 2 groups, for three different values of the number of neighbors per group.

V. Conclusions

O-Streamline is a distribution algorithm designed starting from a theoretical framework. For this reason, the reader can understand the rationale behind its design and ascertain its pros and cons. More important, *O-Streamline* is both amenable to a real implementation and has performance close to absolute bounds for this class of algorithms.

[5]i.e. the complement to one of the cumulative distribution function

References

[1] S. Banerjee, B. Bhattacharjee and C. Kommareddy, *Scalable application layer multicast*, in Proceedings of ACM SIGCOMM, Pittsburgh, PA, USA, 2002.

[2] D. A. Tran, K. A. Hua and T. Do, *ZIGZAG: an efficient peer-to-peer scheme for media streaming*, in Proceedings of IEEE INFOCOM, San Francisco, CA, USA, 2003.

[3] X. Zhang, J.C. Liu, B. Li and P. Yum, *CoolStreaming/DONet: A data-driven overlay network for efficient live media streaming*, In Proceedings of IEEE INFOCOM, Miami, FL, USA, 2005.

[4] M. Zhang, L. Zhao, Y. Tang, J. Luo and S. Yang, *Large-Scale Live Media Streaming over Peer-to-Peer Networks through Global Internet*, in Proceedings of ACM Multimedia, Singapore, Singapore, 2005.

[5] N. Magharei and R. Rejaie, *PRIME: Peer-to-Peer Receiver-drIven MEsh-based Streaming*, in Proceedings of IEEE INFOCOM, 2007.

[6] V. N. Padmanabhan, H. J. Wang and P. A. Chou, *Distributing streaming media content using cooperative networking*, in Proceedings of NOSSDAV, Miami Beach, FL, USA, 2002.

[7] M. Castro, P. Druschel, A. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, *Splitstream: High-bandwidth multicast in cooperative environments*, in Proceedings of the 19th ACM Symposium on Operating Systems Principles, The Sagamore, NY, USA, 2003.

[8] M. Zhang, Y. Xiong, Q. Zhang and S. Yang, *On the Optimal Scheduling for Media Streaming in Data-driven Overlay Networks*, in Proceedings of IEEE GLOBECOM, 2006.

[9] L. Bracciale, F. Lo Piccolo, D. Luzzi, S. Salsano, G. Bianchi and N. Blefari-Melazzi, *A push-based scheduling algorithm for large scale P2P live streaming*, in Proceedings of QoS-IP 2008, on line available at netgroup.uniroma2.it/p2p/QoSIP08.pdf.

[10] G. Bianchi, N. Blefari Melazzi, L. Bracciale, F. Lo Piccolo, S. Salsano and D. Luzzi, *Streamline: an Optimal Distribution Algorithm for Peer-to-Peer Real-time Streaming over Forest-Based Topologies*, submitted to ACM SIGCOMM 2008 and on line available at netgroup.uniroma2.it/p2p/streamline.pdf.

[11] T. Silverston, O. Fourmaux *Source vs Data-driven Approach for Live P2P Streaming* , ICNICONSMCL '06 Washington, DC, USA

[12] E. W. Biersack, P. Rodriguez, and P. Felber, *Performance Analysis of Peer-to-Peer Networks for File Distribution*, in Proceedings of the 5th International Workshop on Quality of Future Internet Services (QofIS'04), Barcelona, Spain, 2004.

[13] L. Bracciale, F. Lo Piccolo, D. Luzzi and S. Salsano *OPSS: an Overlay Peer-to-peer Streaming Simulator for large-scale networks*. The simulator code is on line available at http://minerva.netgroup.uniroma2.it/p2p.