Study of Repair Protocols for Live Video Streaming Distributed Systems

Frédéric Giroire, Nicolas Huin

Study of Repair Protocols for Live Video Streaming Distributed Systems

Frédéric Giroire
CNRS
Sophia Antipolis, France
Email: frederic.giroire@cnrs.fr

Nicolas Huin
Inria Sophia Antipolis
Sophia Antipolis, France
Email: nicolas.huin@inria.fr

***Abstract*—We study distributed systems for live video streaming. These systems can be of two types: structured and unstructured. In an unstructured system, the diffusion is done opportunistically. The advantage is that it handles churn, that is the arrival and departure of users, which is very high in live streaming systems, in a smooth way. On the opposite, in a structured system, the diffusion of the video is done using explicit diffusion trees. The advantage is that the diffusion is very efficient, but the structure is broken by the churn.**

In this paper, we propose simple distributed repair protocols to maintain, under churn, the diffusion tree of a structured streaming system. We study these protocols using formal analysis and simulation. In particular, we provide an estimation of the system metrics, bandwidth usage, delay, or number of interruptions of the streaming. Our work shows that structured streaming systems can be efficient and resistant to churn.

I. Introduction

We consider in this study *live streaming systems*. In these systems, a source streams a video to a set of clients who want to watch the video in real-time. Streaming video can be done over a classic client/server architecture or a distributed (e.g. peer-to-peer (P2P)) one. Distributed solutions are very efficient for *live streaming* scenarios in which clients watch the video at the same time. The advantage is that the bandwidth of every user can be used to forward the video to other users, lightening the source load.

P2P networks are of two types, with structured or unstructured overlays. In the first type, the nodes are organized according to a (or several) logical tree(s), called *diffusion tree(s)*. The source of the video is the root and the video is distributed from the source to the leaves, fathers forwarding the video to their children. In an unstructured overlay, the tree is not explicitly defined: a node having chunks of the video forwards opportunistically these chunks to nodes who miss them. This second type of systems are the most frequently used as they handle very easily *churn*, i.e., the departure and arrival of users, which are very frequent in live video systems. *Frequent churn is the main problem of live distributed streaming system* and the main difference from classical multicast systems. *Structured overlays* have the disadvantage that churn breaks their diffusion trees. However, we have hints that such systems can in fact be very efficient. *If their structure could be maintained by using very simple distributed repair protocols even under frequent churn,* this would allow to keep the advantages of structured overlays, optimal diffusion rate and continuity of the diffusion, while being resistant to churn.

Goal of the study. Our goal is to propose simple distributed repair mechanisms for *structured* live streaming systems. To understand such systems, we then want to develop formal models, which can be efficiently simulated. Last, we aim at proving that they can be very efficient in practice (and potentially more efficient than unstructured systems).

Contributions. In this work, we study a structured network for live video streaming experiencing frequent node departure and arrivals.

- We propose different repair protocols for the diffusion tree using different amount of information in Section II-B.
- We show that a system using these protocols can be, first, formally analyzed and, second, efficiently simulated. We provide estimation of different system metrics, e.g., bandwidth usage, delay, or number of interruptions of the streaming, via simulations, as well as analysis.
- We provide analytical formulas of the system's metrics in Section III.
- We developed a discrete-event simulator, presented in Section IV-A. We used it to compare the different repair protocols. The results are presented in Section IV.
- We present first evidences that, by using simple distributed repair protocols, structured live streaming systems can be very resistant to churn.

A. Related Work.

Structured versus Unstructured Systems. There are two major classes of P2P live video streaming architectures, the first one being named either *unstructured*, *mesh-based*, *gossiping* or *torrent-like*; the second named either *structured* or *tree-based*, see for example [1] or [2] for a classification. Note that this distinction is not strict and that mixed systems were also proposed, e.g. [3].

Early systems, like [4], influenced by IP multicast, attempted at constructing a multicast tree to stream the media. To avoid the shortcomings such as resistance to churn and low bandwidth usage, this simple idea has evolved into elaborate algorithms like Splitstream, proposed in [5] or ZIGZAG, in

This work has been partially supported by ANR project Stint under reference ANR-13-BS02- 0007, ANR program Investments for the Future under reference ANR-11-LABX-0031-01, ANR VISE, CNRS-FUNCAP project GAIATO, the associated Inria team AlDyNet, the project ECOS-Sud Chile.

[6]. The signature of this group of systems is an active maintenance of an overlay structure that clearly defines the data flow, thus the name *structured overlays*. SpreadIt, proposed in [7], is the closest work to ours. It considers multiple protocols for handling arrivals and departures of peers in the network but presents few empirical results and no analysis of the system.

On the other hand, we have systems inspired by BitTorrent, one of the best-known peer-to-peer protocols, described in [8]. The core idea of this class of overlays is organizing the peers into a random, highly-connected graph and disseminating the data using a simple, probabilistic algorithm. The first instance of an unstructured system was introduced in [9] as a way of enhancing a single-tree overlay. It was then the base for the first real peer-to-peer network that streamed video to a big number of simultaneous clients [1]. The characteristic of this group of networks is that they do not have an explicit overlay structure for the data flow, thus the name *unstructured overlays*.

Unstructured systems are widely regarded the better choice. That is often explained by the complexity of making a structured system reliable. However, we show in this study that reliability can be ensured, for a simple system, efficiently by a simple algorithm.

Analysis of Structured Systems. The existing analysis of these systems focus on the feasibility, construction time and properties of the established overlay network, see for example [5], [10] and [11] for a theoretical analysis. But these works usually do not consider over the issue of tree maintenance. Generally, in these works, when some elements of the networks fail, the nodes disconnected from the root execute the same procedure as for initial connection. This is not the case in our study. To the best of our knowledge, there are no theoretical analysis, except [12], on the efficiency of tree maintenance in streaming systems, reliability is estimated by simulations or experiments as in [5].

In [12], the authors propose an efficient maintenance scheme for trees. The distributed algorithm ensures that the tree fastly recovers to a "good shape" after one or multiple failures occur. The authors give analytic upper bounds of the convergence time. This paper is a starting point of our study. We introduce new repair algorithms and then provide an average case analysis of these protocols.

II. Distributed Protocols and Modeling

A. Modeling

We consider a system, a source streaming a live video, and n nodes which want to watch the video. A summary of the variables used in this work is given in Table I. This source is the single reliable node of the network, all other peers may be subject to failure. Nodes are organized according to a tree of size $n+1$. The source is the root of the tree. Nodes forward the video to their children in the tree. Each node has a given bandwidth allowing him to serve a given number of other nodes d. In this study, we consider that all nodes have the same bandwidth 2. A node is said to be *overloaded*, when it has more than d children. In this case, he cannot serve all its children and some of them do not receive the video. Note that the delay between broadcasting a piece of media by the source and receiving by a peer is given by its distance from the root in the logical tree. Hence our goal is to minimize the tree depth, while following degree constraints.

Variable	Signification	Default value
n	Number of nodes of the tree, root not included	1022
d	Node bandwidth (or ideal node degree)	2
h	Height of the tree (root is at level 1)	10
μ	Repair rate (avg. operation time: 100 ms)	1
λ	Individual churn rate (avg. time in the system: 10 min)	$\frac{1}{6000}$
Λ	System churn rate ($\Lambda = \lambda n$)	$\frac{1022}{6000} \approx 0.17$

Terminology	Values
unit of time	100 ms
systems with low churn	$\Lambda \in [0, 0.4]$
systems with high churn	$\Lambda \in [0.4, 1]$

TABLE I: Summary of the main variables and terminologies used in this work.

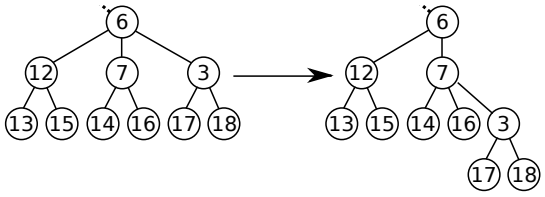

Fig. 1: Example of a *push* operation carried out by the overloaded Node 6.

Each node applies the following algorithm without the knowledge of the whole network.

- When a node is *overloaded*, it carries out a *push* operation. It selects two of its children, and the first one is reattached to the second one, becoming a grandchild. Figure 1 presents an example of such operation.
- When a *node leaves* the system, one of its child is selected to replace him. It reattaches to its grandfather. The other children reattach to it. An example is given in Figure 2. In this work, we only consider single failure but multiple failures could be handled by considering the great grandfather of a node or by reattaching to the root.
- When a *new node arrives*, it is attached to the root.

Churn. We model the system churn rate with a Poisson model of rate Λ. A node departure (also called *churn event*) occurs after an exponential time of parameter Λ, that is in average after a time $1/\Lambda$. We note the individual failure rate $\lambda = \Lambda/n$. In this work, we study scenarios with constant size population.

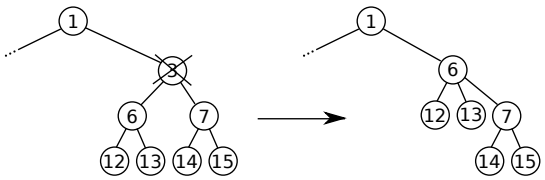

Fig. 2: Example of the operation made after the departure of Node 3.

Thus, when a node leaves the system, we consider that a new node appears. Authors in [13], [14] carried out a measurement campaign of a large-scale overlay for multimedia streaming, PPLive [15]. Among other statistics, the authors report that the median time a user stays in such a system is around 10 minutes. In this study, we use this value as the default value (after normalization see the following on default values).

Repair rate. When a node has a *push* operation to carry out, it has first to change its children list, and then to contact its two children implicated in the operation so that they also change their neighborhood (father, grandfather or children). This communication takes some amount of time, that can vary depending on the node to contact and congestion inside the network. To take into account this variation, we model the repair time as a random variable with exponential distribution of parameter μ. [16] reports that the communication time in a streaming system is in average 79 ms. Thus, we believe that assuming an average repair time of 100 ms is appropriate.

Default values. In the following, for the ease of reading, we normalize the repair rate to 1. We call *unit of time* the average repair time, 100 ms. The normalized default churn rate, λ is thus $1/6000$ and the system churn rate is $\Lambda = n\lambda \approx 0.17$. These default values are indicated as typical examples, but in our experiments we present results for a range of values of Λ between 0 and 1. We talk of *low churn systems* for values of Λ below 0.4, and of *high churn* systems for values above 0.4.

B. Description of the Protocols

We now define *four different protocols* according to four different ways to select the children during the operations and four different *levels of knowledge* of the streaming system. We will study the trade-off between knowledge and performance in the following.

SMALLEST SUBTREE PROTOCOL (SSP). In this protocol, each node knows the subtree size of each of its sons. When a churn occurs, the son with the largest subtree of the failing node takes over the role of its father by adopting every of its sibling. It is itself adopted by its grandfather. When a node is overloaded, its son with the third largest subtree is pushed into the son with the second largest one.

RANDOM PROTOCOL (RP). In this protocol, nodes do not keep information about their subtree sizes. They store their children in a queue and each new node attached to them is put at the end of it. A node receiving the video only gives it to the two children at the start of its queue. When a churn occurs, the eldest son takes over the role of its father by adopting every of its sibling. It is adopted by its grandfather. There is no interruption in the video transmission between the son and the father. The order in the grandfather children is conserved (i.e. If the father was the first child of its parent, the son takes the first place in the grandparent sons). Every of its sibling is reattached to the new father and thus is at the end of the queue, in the same order as it was in the failing node queue. An overloaded node chooses at random the son that will be pushed and also the node that will receive the pushed node.

NO INTERRUPTION PROTOCOL (NIP) This protocol shares some similarity with the random protocol. The same operation is done when a node leaves the system: the eldest child takes over the role of the failing node. When an overloaded carries out a push operation, it chooses uniformly at random a node *not receiving* the video and pushes it under a node *receiving* the video. This ensures no interruption of the video distribution.

PARTIAL INFORMATION PROTOCOL (PIP) The idea of the protocol starts with the observation that for most operations, it is relatively easy to determine the largest subtrees *without storing the exact size of all subtrees*. When a new node arrives, we know that it has a subtree size of one and that it has to be pushed to the bottom of the tree. So we will label it as *new*. Then, during this arrival process, we will have to decide in which subtree to push it. If we do the hypothesis that the tree stays relatively well balanced during the protocol lifetime, we may choose at random without adding too much imbalance in the tree. When there is a churn, we know that one of the children replaces its father. It will have three children, one large (its former brother), that we label *big*, and two smaller ones, that we label *normal*. Hence, it has to push one of the smaller into the second one. Again, if we suppose that the tree is relatively balanced, this choice can be made randomly. The subtree pushed is relabeled as *big*, its new brothers are labeled as *normal*, and we remove the label of its former brothers.

C. Metrics

To evaluate the performance of the different protocols, we are interested by the following metrics.

Time to attach a new node. When a new node arrives into the system, it has to attach to a node which has enough bandwidth to forward the video to it. Basically, it is first attached to the root and then pushed to the bottom of the tree to become a leaf. Ideally, this takes a small amount of time, logarithmic in the number of nodes watching the video, n, see Section III.

Repair time after a node departure. When a node leaves the system, a repair processed starts as described in Section II-A. Basically, one of its son replaces it and some nodes are pushed in its subtree. The simulator records this repair time at each churn event.

Number of people not receiving the video. Due to these two phenomenas, attachment of a new node and repairs, some people do not receive the video during small periods of time during the life of the protocols. We study what fraction of the nodes do not receive the video and during which amount of time.

Height of the tree or delay. The height of the diffusion tree gives the maximum delay between the source and a node. Ideally, it is equal to $\lfloor \log_d n \rfloor + 1$, where d is the maximum node degree.

Number of interruptions and interruption duration. We monitor the number of interruptions of the video diffusion to a node during the protocol lifetime, as well as the distribution of interruption durations.

III. Analysis

In this section, we analyze the Smallest Subtree Protocol. We give analytical formulas to estimate different metrics: the repair time, the average number of people not receiving the video, and the number of interruptions.

The analysis is done under an *independence hypothesis* of the failures, meaning that the repair of one failure is done before another happens. Remark that it implies that the diffusion tree is always a balanced binary tree when a failure happens. Indeed, we are considering a system with constant population in which a new node arrives when a node leaves. Using the information about subtree sizes, SSP will push this new node in the optimal position. We will consider here complete binary trees of size $n = 2^h - 2$, but the analysis extends to other values. The independence hypothesis is evaluated by simulation for different values of churn in Section IV-B.

Analysis of the repair time. We study the repair time of the diffusion tree when a node selected at random leaves the system. We consider the position of the node in the tree.

If the failing node is a leaf, the tree is still balanced and nothing happens.

If the failure happens at depth $i < h$, the tree is repaired (that is, all nodes have degrees less or equal to 2) after $h-i-1$ push operations. Note that when $i = h-1$, no pushes are needed.

We give now the average time to repair the tree after a node departure. In average, a node carries out an operation in a time $\frac{1}{\mu}$. Hence, the average time to carry out $h-i-1$ operations is $\frac{h-i-1}{\mu}$. The number of nodes at depth i, $2 \leq i \leq h$, is 2^{i-1}. Thus, the probability that the node leaving the system is at depth i is $\frac{2^{i-1}}{n}$. Recall than $n = 2^h - 2$. If we note T the time to repair the diffusion tree, we have

$$\begin{aligned}\mathbb{E}[T] &= \frac{1}{2^h-2}\sum_{i=2}^{h-2} 2^{i-1}\frac{h-i-1}{\mu} = \frac{2^{h-1}-2(h-1)}{\mu(2^h-2)} \\ &\sim \frac{1}{2\mu} \text{ when } h \text{ is large.}\end{aligned}$$

The Smallest Subtree Protocol is very efficient. Indeed, only one half operation is needed in average to repair the tree after a node departure.

Analysis of the number of nodes not receiving the video. During the repair process, some nodes do not receive the video. Indeed, recall that only the two biggest subtrees of an overloaded node receive the video. When a node at level i leaves the system, its child with the largest subtree replaces it. It becomes overloaded with three children of subtree sizes $2^{h-i}-1$, $2^{h-i-1}-1$, and $2^{h-i-1}-1$ (A node at depth i is of height $h-i+1$. A subtree of height i contains 2^i-1 nodes). Thus, $2^{h-i-1}-1$ nodes do not receive the video. At each repair, the height of the tree is reduced by one and this number is divided by around two. That is after k repairs, it is equal to $2^{h-i-1-k}-1$, see Figure 3. A push operation is done in average in a time $\frac{1}{\mu}$. Hence, if we note n_i the number of people without video when the failing node is at level i, we have

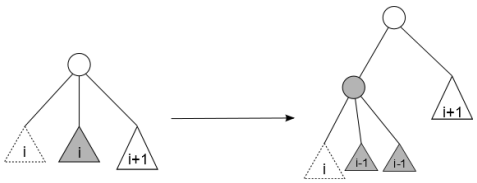

Fig. 3: Effect of a push operation during a repair.

$$\mathbb{E}[n_i] = \frac{\Lambda}{\mu}\sum_{k=0}^{h-i-1}(2^k-1) = \frac{\Lambda}{\mu}\left(2^{h-i}-h+i-1\right).$$

Recall now that the probability that the failing node is at level i is $\frac{2^{i-1}}{2^h-2}$. If we note n_{bad}, the number of people without video, we have

$$\begin{aligned}\mathbb{E}[n_{bad}] &= \sum_{i=2}^{h-2}\frac{2^{i-1}}{2^h-2}n_i = \frac{2^{h-1}(h-5)+2(h+1)}{2^h-2}\frac{\Lambda}{\mu} \\ &\sim \frac{h-5}{2}\frac{\Lambda}{\mu} \text{ when } h \text{ is large.}\end{aligned}$$

Analysis of the number of interruptions. Recall that when a node leaves the system at level i, its first child is overloaded with three children of subtree sizes $2^{h-i}-1$, $2^{h-i-1}-1$, and $2^{h-i-1}-1$. Only the two biggest sons of a node receive the video. Thus, $2^{h-i-1}-1$ nodes are interrupted. With a repair, the interruption for these nodes stops and the height of the tree interrupted is reduced by one, that is after k repairs, it is equal to $2^{h-i-1-k}-1$. Every interrupted node is only interrupted once. Hence, if we note $int_i(t)$ the number of interruption when the failing node is at level i as a function of the time, we have

$$E[int_i(t)] = t\Lambda\sum_{k=0}^{h-i-1} 2^k - 1 = t\Lambda(2^{h-i}-h+i-1)$$

Recall now that the probability that the failing node is at level i is $\frac{2^{i-1}}{2^h-2}$. If we note $n_{int}(t)$, the average number of interruptions, we have

$$\begin{aligned}\mathbb{E}[n_{int}(t)] &= \sum_{i=2}^{h-2}\frac{2^{i-1}}{2^h-2}int_i = \frac{2^{h-1}(h-5)+2(h+1)}{2^h-2}t\Lambda \\ &\sim \frac{h-5}{2}t\Lambda \text{ when } h \text{ is large.}\end{aligned}$$

Analysis of the arrival time. When a new node arrives in the system, it is first attached to the root. It does not receive the video until it is then pushed to the bottom of the tree by $h-2$ successive push operations. Since a node carries out an operation in an average time of $\frac{1}{\mu}$, if we node T the time for the new node to receive the video, we have:

$$\mathbb{E}[T] = \frac{h-2}{\mu}.$$

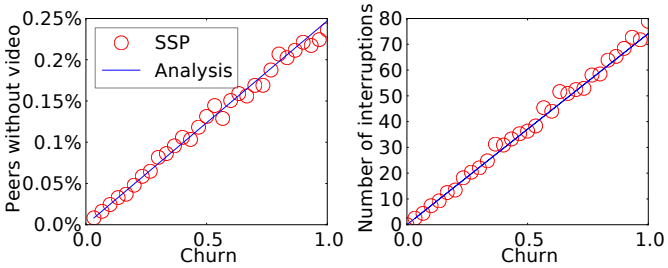

Fig. 4: Comparison between the formal model of Section III and the simulations for: (Left) Average fraction of peers not receiving the video as a function of the churn rate due to the repair (1022 nodes). (Right) Average number of interruptions per node (1022 nodes) for a time of 30,000 units of time.

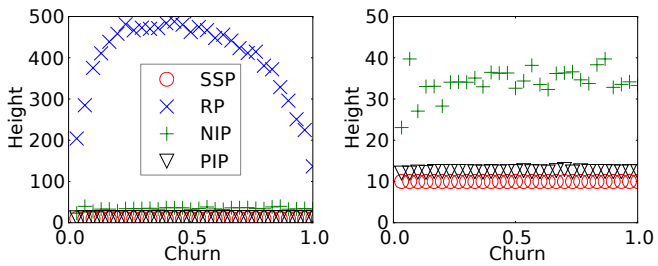

Fig. 5: Average height (over time) of the tree (1022 nodes) as a function of the churn rate for the four repair protocols. Left: y-axis range from 0 to 500. Right: y-axis range from 0 to 50.

IV. Simulations

We developed a discrete-event simulator of the streaming system described in Section IV-A. We use it to analyze and compare the different protocols. A summary of results is provided in this section for different metrics.

A. The simulator

Our desire to focus on high level simulation led us to develop a custom C++ discrete-event simulator to evaluate metrics of the system described in Section II-C. We did not use low level network simulators like NS-2 or OMNET because they would require more computation time and give metrics non pertinent to our analysis. Our goal is to focus on the tree structure after churns and reparations to validate our protocols.

B. Validation of the analytical model

To validate our analysis, we first compared the results given by the simulator to the analytical formulas obtained in Section III.

People without the video. In Figure 4 (Left) is given the average fraction of people not receiving the video. New nodes joining the systems are not taken into account until they start to receive the video. Only the non distribution of video due to the repairs is counted. As an example, for the default churn rate value, 0.17, only 0.03% of the nodes do not receive the video in average, and for a high value of churn 1, only 0.25%.

We see that the closed formula models very closely the system. Recall that the formulas were given for low churn, that is when a churn event is repaired before another churn event. This corresponds to values of the churn rate Λ between 0 and 0.5. However, we see that, even for larger churn values (> 0.5), the system behavior is well predicted by the formula.

Interruptions. The average number of video interruptions is given in Figure 4 (Right) as a function of the churn rate. Again, the interruptions are due to the repair of the diffusion tree when there is a node departure.

This number is the average over all nodes of the number of interruptions per node for a period of 30,000 units of time. For example, for a value of churn of 0.2, the diffusion for a node is interrupted 20 times in average, that is, a node experiences an interruption every 1500 units of time, that is every 150 seconds. As we will see it later, the durations of the interruptions are very short. Thus, they are without consequence for the end-user experience.

We see that the analytical formula gives a very good estimation of the behavior of the system. This is true even for high churn.

C. Comparison of the protocols

In this section, we compare by simulation the performances of the different protocols that we propose. Recall that these different protocols use different levels of information. Several providers of live video may have different criteria on the different metrics and on the implementation, leading to different choices of the adequate protocol.

Levels of needed information. We recall that, for every protocol, each node has at least to store the ids (and addresses) of children, father and grandfather. In RP and NIP, each node stores the order in which its children attached to it. In SSP, each node needs to know the size of its subtree. Last, in PIP, only two additional bits are necessary: a new node arriving in the system is tagged as *new* till it finds a place in the diffusion tree and a node pushed by a repair process is tagged as *big*. For the detailed discussion, the reader may refer itself to Section II-B.

We now compare the protocols for the different metrics. As we will see, the protocols do not behave at all similarly for some metrics. *To improve the readability of the figures, for each metric, we propose two plots with the same data, but with different scales of the y-axis.*

Height of the diffusion tree and delay. Figure 5 shows the average height (over time) of the tree as a function of the churn rate.

The first observation is that RP exhibits a very different behavior than the other protocols, see Figure 5 (Left). It behaves very badly: the height of the diffusion reaches a value of 500. It is not of order logarithmic in n, the number of nodes. On the contrary, it gets close to a linear height! The diffusion tree looks like a path in this case. The protocol is very inefficient. Thus, pushing the nodes randomly is not possible in practice.

We are now interested by the three other protocols (Figure 5 (Right)). They behave a lot better. First note that the average height does not depend on the churn rate. Thus, the difference between the three protocols only is the value of the height:

- SSP has a constant height of 10 which is exactly $\lfloor \log_d n \rfloor + 1$ for $n = 1022$. The protocol is optimal for the maximum height, and thus delay, of a node.
- PARTIAL INFORMATION PROTOCOL has an average height around 12.3. The height is not very far from the optimal 10.
- NO INTERRUPTION PROTOCOL behaves worstly. The average weight is around 33. It is nevertheless a lot better than the completely random protocol RP, and could still be used in practice.

The explanation of the different protocols' efficiency is due to their different behaviors, (1) when there is a churn event, and, (2) when there is an arrival of a new node.

First, when a failing node is close to the root, a large subtree does not receive the video anymore and has to be pushed in the tree by the repair process. The SSP protocol succeeds to reconstruct a perfectly balanced tree by using the information about the subtree sizes. It knows exactly what is the right node to push. The two protocols RP and NIP do not have this information: they blindly push the subtree. It can thus happen that a large subtree ends up at the bottom of the tree at the end of the repair process. This can increase its height significantly: in the worst case, the height can be doubled with only one churn event. We see that PIP has performance not too far from optimal. We see that, even without the information of the subtree sizes, a simple guess of the node to be pushed is efficient in terms of delay. The protocols recognizes large subtrees (of size 2^i) from small subtrees (of size 2^{i-1}), leading to a near optimal repair process.

Second, when there is an arrival of a new node in the tree. SSP can push this new node exactly at the right position of the diffusion tree. PIP and NIP cannot distinguish two subtrees with a small difference of size and thus push the new node randomly to the bottom of the diffusion tree. RP has in this case a very bad behavior. As it does not have any information, it does not know that a node is new. Hence, it pushes randomly a node that can have a very large subtree, instead of the new node of subtree size one! On the contrary, NIP does not push subtrees receiving the video to ensure a continuity of the video diffusion. Hence, the arrival of a new node in the system cannot trigger that a large subtree is pushed at the bottom of the tree.

To sum up, the repair protocol SSP *is optimal in terms of tree height and, thus, of delay. This protocol uses information about node subtree sizes. If this information is considered too costly to maintain by an operator, it can obtain close to optimal performances with* PIP*, which uses a very small amount of information (two node labels).*

Percentage of people without the video during time. Figure 6 shows the average percentage of nodes that do not receive video as a function of the churn rate.

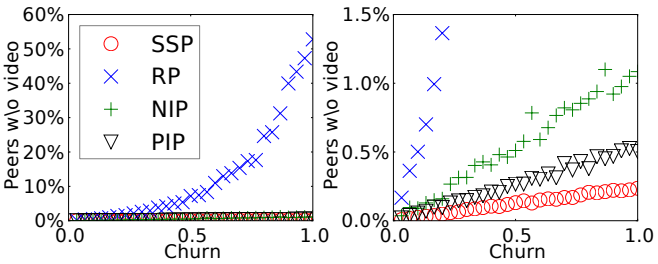

Fig. 6: Average fraction of peers not receiving the video as a function of the churn rate (1022 nodes). Left: y-axis range from 0 to 60%. Right: y-axis range from 0 to 1.5%.

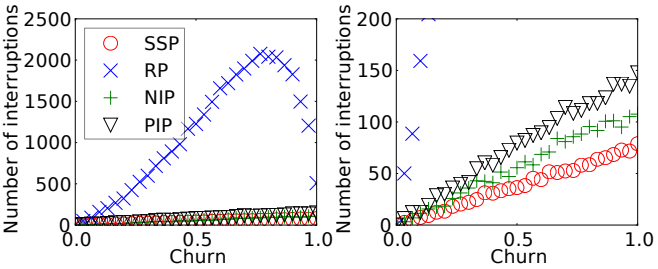

Fig. 7: Average number of interruptions of the streaming per node as a function of the churn rate after 10,000 units of time. Left: y-axis range from 0 to 2500. Right: y-axis range from 0 to 200.

We see again, in Figure 6 (Left), that RP behaves a lot worse than the other three protocols. As much as 52% of the nodes do not get the video for a churn rate of 1. It already reaches 6% for a churn rate of 0.5. This protocol is thus very inefficient.

The three other protocols, SSP, NIP and PIP, behave similarly, and are very efficient. The average percentage of people without the video is very small for churn values expected in a viable live streaming system (churn rate between 0 and 0.4). For a churn rate of 0.2, the average percentage is respectively 0.04%, 0.1% and 0.15 % of the nodes for the three protocols, see Figure 6 (Right).

Number of interruptions during the diffusion. We studied the number and durations of interruptions of the video diffu-

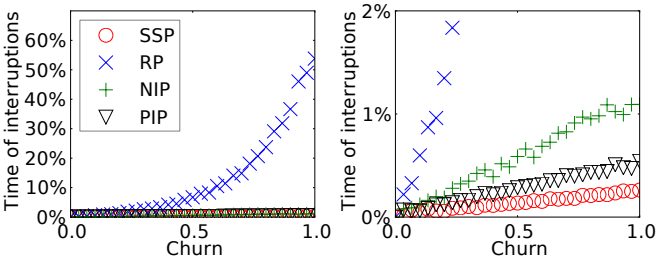

Fig. 8: Average fraction of time for which the streaming was interrupted as a function of the churn rate (after 30,000 units of time for 1022 nodes).

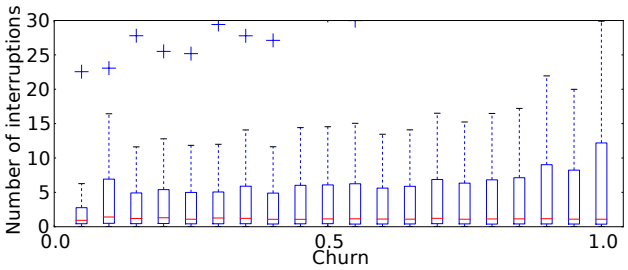

Fig. 9: Distribution of the interruption durations for different churn values (1022 nodes over 30,000 units of time). Boxplots give: median value (red line), first and third quartile (box), maximum value (blue cross).

sion process for a node in the system. Indeed, the diffusion to a node can be interrupted after the departure of one of its ancestors. We report in Figure 7 the average number of interruptions for a node present in the system during 30,000 units of time (3,000 seconds for the default values), as a function of the churn rate. Again, we see in Figure 7 (Left) that RP behaves very badly with a peak of 2,000 interruptions for a churn of 0.8. This represents an interruption every 15 units of time, that is every 1.5 second. This protocol is not viable. On the contrary, we see in Figure 7 (Right) that SSP, NIP and PIP behaves very well and similarly. Even for a high value of churn like 1, the number of interruptions per node is at most 150, representing an interruption every 200 units of times. For a low churn, e.g. $\Lambda = 0.1$, the number of interruptions is close to 10, that is an interruption every 3,000 units of time, that is 5 minutes. We note that NIP behaves better than PIP for this metric, when it is behaving worse for the other metric. The explanation is that the NIP was specially designed to avoid the interruption of the video diffusion. During the repair process, the two nodes receiving the video are never pushed, even if they have a smaller subtree than a node not receiving the video. This is not the case for SSP, PIP and RP.

We plot in Figure 8, the average fraction of time of node was interrupted during the simulation, that during 30,000 units of time. We see that for a value of churn of 0.1, a node is interrupted in average for 0.15% of the time only.

We plot in Figure 9 the distribution of the duration of an interruption for SMALLEST SUBTREE PROTOCOL. We see that the median time is 1. More than half of the interruptions lasts 1 unit of time. The value of the third quartile is less than 5 units of time for almost all values of churn rates. The maximal interruption lasts less than 30 units of time for a system with low churn (churn rate between 0 and 0.4).

To summarize, in a system in which nodes stay on average 15 minutes (churn rate $\Lambda = 0.1$), a node watching a video for one hour will experience 12 interruptions of median duration 100 ms, few interruptions of duration 500 ms, and if it is not lucky, one interruption of 2.5 seconds. A buffer of few seconds (e.g. 10s) of video will make these interruptions imperceptible to the end-users. For a video rate of 480 kbps, it corresponds to a buffer size of only 40MB.

V. CONCLUSION, CURRENT AND FUTURE WORK

In this work, we study a live video streaming system via formal analysis and simulation. We show that, using a simple repair protocol, a *structured* peer-to-peer system can be very efficient. The diffusion tree can be maintained thanks to independent distributed operations of the nodes. This leads to well-balanced diffusion trees, with almost optimal (logarithmic) distance to the source. We additionally show that the diffusion of the video is interrupted only for very short durations of times, imperceptible by an end user.

We are currently investigating analytical models of the streaming system. In particular, the closed formulas are given for SMALLEST SUBTREE PROTOCOL. We wish to obtain models for the other protocols. We are also working on models for higher churn rates. For these rates, the source becomes a bottleneck, as new peers attach to it. We can estimate repair times by modeling the number attached to the source as a Markovian queuing system.

REFERENCES

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

[2] B. Li, Z. Wang, J. Liu, and W. Zhu, "Two decades of internet video streaming: A retrospective view," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 9, no. 1s, pp. 33:1–33:20, Oct. 2013.

[3] F. Wang, Y. Xiong, and J. Liu, "mtreebone: A collaborative tree-mesh overlay network for multicast video streaming," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 21, no. 3, pp. 379–392, 2010.

[4] Y. hua Chu, S. Rao, and H. Zhang, "A case for end system multicast," in *Proc. of ACM Sigmetrics*, 2000, pp. 1–12.

[5] M. Castro, P. Druschel, A. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, "SplitStream: high-bandwidth multicast in cooperative environments," in *Proceedings of the nineteenth ACM symposium on Operating systems principles*, 2003, p. 313.

[6] D. A. Tran, K. A. Hua, and T. Do, "Zigzag: An efficient peer-to-peer scheme for media streaming," in *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, vol. 2. IEEE, 2003, pp. 1283–1292.

[7] H. Deshpande, M. Bawa, and H. Garcia-Molina, "Streaming live media over a peer-to-peer network," *Technical Report*, 2001.

[8] B. Cohen, "Incentives build robustness in bittorrent," in *Workshop on Economics of Peer-to-Peer systems*, vol. 6. Citeseer, 2003, pp. 68–72.

[9] S. Banerjee, S. Lee, B. Bhattacharjee, and A. Srinivasan, "Resilient multicast using overlays," in *ACM SIGMETRICS Performance Evaluation Review*, vol. 31, no. 1. ACM, 2003, pp. 102–113.

[10] V. Venkataraman, K. Yoshida, and P. Francis, "Chunkyspread: Heterogeneous unstructured tree-based peer-to-peer multicast," in *14th IEEE International Conference on Network Protocols*, 2006, pp. 2–11.

[11] G. Dan, V. Fodor, and I. Chatzidrossos, "On the performance of multiple-tree-based peer-to-peer live streaming," in *26th IEEE International Conference on Computer Communications*, 2007, pp. 2556–2560.

[12] F. Giroire, R. Modrzejewski, N. Nisse, and S. Pérennes, "Maintaining balanced trees for structured distributed streaming systems," in *20th Colloquium on Structural Information and Communication Complexity (SIROCCO)*, vol. LNCS 8179, Ischia, Italy, 2013, pp. pages 177–188.

[13] X. Hei, C. Liang, J. Liang *et al.*, "Insights into PPLive: A Measurement Study of a Large-Scale P2P IPTV System," in *International Word Wide Web Conference. IPTV Workshop*, 2006.

[14] L. Vu, I. Gupta, J. Liang, and K. Nahrstedt, "Measurement of a large-scale overlay for multimedia streaming," in *Proceedings of the 16th international symposium on High performance distributed computing*. ACM, 2007, pp. 241–242.

[15] "PPLive homepage, http://www.pplive.com/."

[16] B. Li, Y. Qu, Y. Keung, S. Xie, C. Lin, J. Liu, and X. Zhang, "Inside the new coolstreaming: Principles, measurements and performance implications," in *27th IEEE International Conference on Computer Communications*, 2008.