The Structured Way of Dealing with Heterogeneous Live Streaming Systems

Andrea Tomassilli, Nicolas Huin, Frédéric Giroire

The Structured Way of Dealing with Heterogeneous Live Streaming Systems

Andrea Tomassilli, Nicolas Huin, and Frédéric Giroire

Abstract In peer-to-peer networks for video live streaming, peers can share the forwarding load in two types of systems: unstructured and structured. In unstructured overlays, the graph structure is not well-defined, and a peer can obtain the stream from many sources. In structured overlays, the graph is organized as a tree rooted at the server and parent-child relationships are established between peers. Unstructured overlays ensure robustness and a higher degree of resilience compared to the structured ones. Indeed, they better manage the dynamics of peer participation or churn. Nodes can join and leave the system at any moment. However, they are less bandwidth efficient than structured overlays. In this work, we propose new simple distributed repair protocols for video live streaming structured systems. We show, through simulations and with real traces from Twitch, that structured systems can be very efficient and robust to failures, even for high churn and when peers have very heterogeneous upload bandwidth capabilities.

1 Introduction

Live streaming can be done either over a classic client-server architecture or a distributed one. The high bandwidth that live streaming requires may limit the number of clients that the source can serve. A small number of clients could be sufficient to saturate the source resources.
In a distributed scenario (e.g., P2P), the bandwidth required can be spread among the users and the bottleneck at the source can be reduced. So, peer-to-peer systems are cheap to operate and scale well with respect to the cen-

Andrea Tomassilli and Frédéric Giroire
Université Côte d'Azur, CNRS, Inria, Sophia Antipolis, France

Nicolas Huin
Huawei Paris Research Labs, France

tralized ones. Because of this, the P2P technology is an appealing paradigm for providing live streaming over the Internet.
In P2P context, we can choose between an unstructured or a structured overlay network. In unstructured overlay networks, peers self-organize themselves without a defined topology. Unlike the structured case, any peer can receive each piece of the video from a different peer. In structured overlay networks, peers are organized in a static structure with the source at the root of the diffusion tree A node receives data from a parent node that can be a peer or the source of the streaming. In these type of systems the content distribution is easier to manage with respect to unstructured ones. In unstructured systems the diffusion tree is done opportunistically and, as the authors show in [6], this ensures efficiency and robustness to the dynamicity of peers. In structured systems, a departure and arrival of users (churn) may break the diffusion tree. Our goal is to check if these kinds of systems can also be efficient and robust to churn.

Contributions. In this work, we study a structured network for live video streaming experiencing frequent node departures and arrivals in systems where nodes have heterogeneous upload bandwidth.

- We propose, in Section 4, simple distributed repair protocols to rebuild the diffusion tree when peers are leaving. Different protocols use different levels of information.
- Using a custom made discrete-event simulator, we compare the protocols using different metrics, i.e., delay, percentage of clients without video, number and duration of interruptions. We validate the protocols using different peer bandwidth distributions from literature.
- We study the efficiency of the protocols versus the level of information they use. We show that a repair protocol can be very efficient, even when using only a small amount of local information on its neighbors.
- Finally, we use real Twitch traces to compare our different heterogeneous protocols in a real-life scenario of a streaming session in Section 5. We show that our simple distributed repair protocols work very well in practice.

Due to lack of space, all the obtained results could not be included in the conference version and can be found in [19].

2 Related Works

There is a large amount of work on video streaming systems (see [18, 11] for surveys), such as to improve video coding, e.g., adaptive streaming [16] or multiview video [12], to reduce energy consumption [5], or to improve the way the overlay network deals with churn using techniques from optimization [17], protocols, or algorithmics. Our work lies in the last category.

There are two main categories of distributed systems for video live streaming: unstructured and structured. [14] provides an overview of P2P based live streaming services. Hybrid solutions are also possible [22]. In unstructured overlay networks, peers self organize themselves in an overlay network, that does not have a defined topology. CoolStreaming/DONet [23] is an implementation of this approach. In structured overlay networks, peers are organized in a static structure with the source at the root of the tree. There are many techniques used in P2P live streaming with single-source and structured topology. These techniques may fall into two categories: single-tree and multiple-tree. In the single-tree approach, each node is connected to a small number of nodes to which it is responsible for providing the data stream. ZIGZAG [20] is an implementation of this approach. In the multiple-tree approach, the main idea is to stripe the content across a forest of multicast trees where each node is a leaf in every tree except one. SplitStream [7] is an implementation of this approach.
In terms of reliability, unstructured systems are considered the best choice. As shown in [6] this kind of system handles churn (peers leaving the system) in a very efficient way. Only few works ([9, 8]) focus on tree maintenance in structured systems. In [9] the authors propose a simple distributed repair algorithm that allows the system to fastly recover and obtain a balanced tree after one or multiple failures. In [8] the authors develop a simple repair and distributed protocol based on a structured overlay network. By providing, through analysis and simulations, estimations of different system metrics like bandwidth usage, delay and number of interruptions of the streaming, they show that a structured live-streaming system can be very efficient and resistant to churn. However, their study is based on the fact that all nodes have the same bandwidth, where the bandwidth determines the maximum number of other nodes that can be served simultaneously. We extend their work to the case of peers with heterogeneous bandwidth.

3 Distributed Systems and Modeling

3.1 Modeling

We model a live streaming system as a tree of size $n + 1$ where the root represents a source streaming a live video. A summary of the variables used in this work is given in Table 1. The n other nodes are clients wanting to watch the video. The source is the only reliable node of the network; all other peers may be subject to failure. A node v has a limited bandwidth $d(v)$ used to serve its children. A node is said to be *overloaded*, when it has more than $d(v)$ children. In this case, it cannot serve all its children and some of them do not receive the video. Note that the distance, in the logical tree, between

Table 1: Summary of the main variables and terminologies used in this work.

Variable	Signification	Default value
n	Number of nodes of the tree, root not included	1022
d	Node bandwidth (or ideal node degree)	-
h	Height of the tree (root is at level 1)	-
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	$\Lambda \in [0.4, 1]$

a peer and the root gives the reception delay of a piece of media. Hence our goal is to minimize the tree depth while respecting degree constraints.

Each node applies the following algorithm with a limited 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.
- When a *node leaves* the system, one of its child is selected to replace it. The other children reattach to it. In this work, we only consider single failure. But multiple failures could be handled by considering the great grandparent 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 of Λ. A node departure (also called *churn event*) occurs after an exponential time of parameter Λ, i.e., in average after a time $1/\Lambda$. We note the individual failure rate $\lambda = \Lambda/n$. Authors in [10, 21] carried out a measurement campaign of a large-scale overlay for multimedia streaming, PPLive [1]. 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.

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 neighborhoods (parent, grandparent or child). 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 an exponential distribution of parameter μ. [13] reports that the communication time in a streaming system is on 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, for an average stay of 10 minutes, 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 and are reported in Table 1, 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.

3.2 Metrics

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

Number of people not receiving the video. Due to repair and churn events, some people do not receive the video during small periods. 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.

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

4 Protocols for Heterogeneous Systems

We present three new repair protocols using different levels of knowledge.

Description of the Protocols. To obtain a tree with minimum height while respecting the degree constraints, the following *two conditions for optimality* must hold:

1. *If there exists a node at level L, all previous levels must be complete*;
2. *For each pair of levels $(L, L+1)$ the minimum bandwidth between all the nodes at level L must be greater than or equal to the maximum bandwidth between all the nodes at level $L+1$.*

Thus, the distributed protocols that we propose try to maintain nodes with high bandwidth on top of the diffusion tree. To this end, an overloaded node has to carefully select which node is pushed under which node using its information on the diffusion tree.

Local Bandwidth Protocol (LBP) In this protocol, each node knows the bandwidth of each of its children. Moreover, a node keeps track of the number of *push operations* done on each of its children. Note that this local information is easy to maintain accurately.

When a node is overloaded, it pushes its child with the smallest bandwidth, as nodes with higher bandwidth should stay on top on the diffusion tree. This

child has to be pushed on a node receiving the video. We consider all the nodes receiving the video and push on them proportionally to their bandwidth (e.g., a node with a bandwidth of $d = 4$ should receive twice as much pushes than a node with a bandwidth of $d = 2$). In details, the expected number of pushes to each child is proportional to its bandwidth. The parent then pushes into the child with the largest difference between the push operations done and the expected number of push operations.

Bandwidth Distribution Protocol (BDP) In this protocol, nodes have additional information. Each node knows the bandwidth of each of its children and the bandwidth distribution of the subtree rooted in each of its children. Note that the bandwidth distribution can be pulled up from the subtree. This may be considered costly by the system designer. However, it is also possible for a node to estimate this distribution by keeping the information of the bandwidth of the nodes pushed into it.

This additional information allows us to estimate the optimal height of the subtrees of each of the children. Thus, when a node is overloaded, it can push its child with the smallest bandwidth into its child (receiving the video) with the smallest *estimated height*.

Full Level Protocol (FLP) In this protocol, each node knows the bandwidth of each of its children and the *last full level* of the subtree rooted in each of its children.

This information allows knowing in which subtree nodes are missing. The main idea is to push toward the first available slot to not increase the height of the diffusion tree. When a node is overloaded, its child with the smallest bandwidth is pushed into the child with the smallest *last full level*.

For all three protocols, a churn event is handled similarly. When a node leaves the system, the children of the falling node are adopted by their grandparent.

Discussion. In LBP, a node has no information about the underlying subtree. Push operations are carried out according to the degree of the nodes that receive the video. Since nodes may leave the system at any moment, it can thus happen that a node is pushed into the worst subtree.

In BDP, a node knows the bandwidth distribution of each subtree rooted in each one of its children. A node is pushed according to the estimated height of the subtree rooted in its children. In the estimation, a node assumes that all the levels, except the last one, is complete. Hence, a node may be not pushed on the best subtree.

In FLP, a node knows the last full level of the subtree rooted in each one of its children. A node is pushed under the node with the smallest value. In this way, we are sure that the node is pushed toward the best possible position. This may not be enough. In fact, due to nodes arrival and nodes departure, the *two conditions for optimality* may be broken.

Thus, in none of the protocols, the two conditions for optimality are always true. However, the transmission delay is not the unique factor that determines the QoS for users. Other factors like time to attach a new node, number and

	Phase 1	**Phase 2**	**Phase 3**
Arrival rate ϕ	0.062	0.025	0.025
Individual churn rate λ	1.57×10^{-5}	1.57×10^{-5}	3.15×10^{-5}
Duration (time units)	14640	112890	9648

Fig. 1: Parameters obtained from the model for the considered streaming session.

	LBP	**BDP**	**FLP**
Height	5.66	6.1	5.17
People w/o video (%)	0.21	0.24	0.22
Number of Interruptions	17.54	12.12	20.87
Time of Interruptions (%)	0.01	0.01	0.017

Fig. 2: Average metrics for Distribution 1 after 250 simulations.

duration of interruptions are as important as the transmission delay. So, we decided to keep the protocols as simple as possible taking into consideration all metrics.

Handling Free Riders. Some nodes in the system can have no upload bandwidth and thus cannot distribute the video to anyone. They are called *free riders*. They may pose difficulties and call for special treatment. An obvious observation is that protocols should not push nodes under a free rider; all our protocols prevent this. A more problematic situation arises when deadlocks appears in which some nodes do not receive the video and all their brothers are free riders. Push operations cannot solve the situation. In this case, we decided to ask the concerned free riders to rejoin the system. This is a solution we wanted to avoid as the goal of a structured protocol is to maintain as much as possible the parent-children relations of the diffusion tree. However, we considered that this is a small cost that free riders can pay, as they do not contribute to the system.

Evaluation of the protocols. We extensively studied the protocols and compared their performances for 4 bandwidth distributions of live video streaming systems that we found in the literature. The study can be found in [19]. We only present the results using Twitch traces in the next section.

5 Results with Twitch Traces

To simulate the protocols in real scenarios, we decided to use Twitch [3] as a Use Case. Twitch is a live streaming video platform that mainly focuses on video gaming and e-sport events. It first appeared in 2011, and its popularity grew very fast. Today, with 1.5 million broadcasters and 100 million visitors per month Twitch represents the 4th largest source of peak Internet traffic in the US [15]. To understand the behavior of the viewers of the stream, we monitored the 100 most famous streamers (in terms of viewers and followers [4]).

We gathered the number of viewers for each moment of their streaming sessions. We noticed that most of the streaming sessions can be divided into 3 phases: (1) the start of the stream where the number of viewers increases at an extremely high rate, (2) the central part of the stream where the number of viewers increases at a slower rate than at the beginning of the stream and

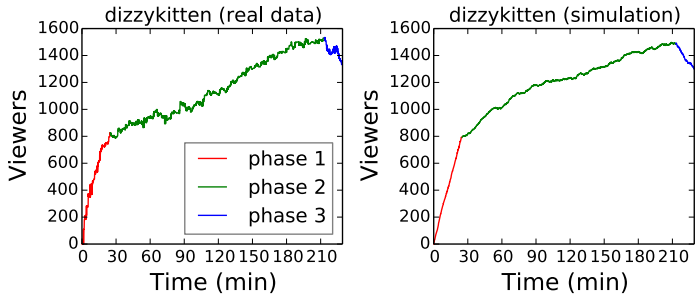

Fig. 3: Number of viewers as a function of the time for a streaming session of the user *dizzykitten*.

(3) the end of the stream where the number of viewers decrease (see Figure 3 (Left))
We defined a model to represent streaming sessions that follow the 3–phases pattern. This model allows us to abstract from the real data and to repeat the simulations several times to estimate the quantitative behavior of the protocols more easily. Using the fact that, on average, a user spends 106 minutes on *twitch.tv* [2], for each phase, we calculate the arrival rate and the individual churn rate, modeled as a Poisson model. Table 1 shows the rates given to the simulator for the considered streaming session. To calculate the leaving rate of the 3rd phase, we assumed that the arrival rate of Phase 3 is the same as the one of Phase 2. Fig. 3 compares the data obtained from monitoring the user with the data obtained from the model.

We simulated our 3 protocols using the data generated from the model for our metrics of interest and according to the four distributions of bandwidth presented in the previous section. Table 2 summarizes the average results of the 3 protocols after 250 simulations using Distribution 1. Results for the other distributions are similar. They are omitted due to lack of space, but they can be found in [19].

Height of the diffusion tree and delay. We see that *all protocols achieve a very small height of the diffusion tree*: around 5 or 6 for the average. Recall that we have around 1500 users at the maximum of the stream. The protocols are thus very efficient. The evolution of the diffusion tree height is given as an example in Figure 4. We see the increase of height when the users connect to the stream till a maximum height of 6 for FLP and LBP, and of 7 for BDP. FLP gives the tree with the smallest height for all the distributions. The results of LBP and BDP are different from the simulation case of the previous section. Since the individual churn rate is very small during the experiment ($\sim 1.5 * 10^{-5}$), *pushing according to the children's bandwidths (local information) reveals to be a good strategy*, leading to a better height than BDP for 2 of the 4 distributions. In particular, LBP behaves better than BDP in the case of very distant values of bandwidth (Distributions

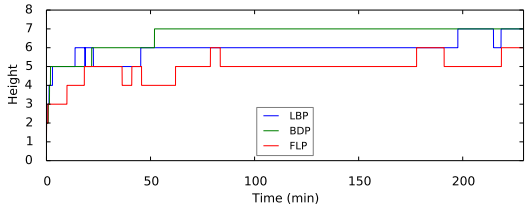

Fig. 4: Height of the diffusion tree during time for an example of Twitch session *dizzykitten*.

1 and 3) and worst when the values of bandwidth are close between them (Distribution 4).

Percentage of people without the video during time. In this case, the 3 protocols behave similarly, as in the simulation case. They are very efficient: on average, only 0.2% of peers are unable to watch the video. Recall that we count the users arriving and waiting to be connected at the right place of the tree.

Number of interruptions during the diffusion. The protocols have similar behavior in most cases. For Distributions 1, 2 and 3 the number of interruptions ranges from 12 to 21, and for Distributions 4, it ranges from 5 to 6. This means that, in the worst case, a peer staying for all the duration of the stream experiences an interruption every 10 minutes. In all cases, the duration of these interruptions is very small. Considering all protocols and all the distributions, we see that a node is never interrupted for more than the 0.02% of the time. A peer remaining during the whole stream session is thus interrupted for less than 3 seconds. A buffer of few seconds (e.g., 10s) for the video makes these interruptions imperceptible to the end-users. For a video rate of 480 kbps, it corresponds to a buffer size of only 40MB.

6 Conclusion

In this study we examined the problem of delivering live video streaming in a P2P overlay network using a structured overlay. We have proposed 3 distributed protocols to repair the diffusion tree of the overlay, when there is churn. The protocols use different amounts of information. Using simulations and experiments with real traces from Twitch, we have shown that our protocols behave well with respect to fundamental QoS metrics, even for very heterogeneous bandwidth distributions. *Our main result is that, with very simple distributed repair protocols using only local information, structured overlay networks can be very efficient and resistant to churn.*

References

1. PPLive homepage, http://www.pplive.com/.
2. Twitch Blog, https://blog.twitch.tv/twitch-hits-one-million-monthly-active-broadcasters-21dd72942b32
3. Twitch homepage, http://www.twitch.com/
4. Twitch Statistics, http://socialblade.com/twitch/
5. Bacco, M., Catena, M., De Cola, T., Gotta, A., Tonellotto, N.: Performance analysis of webrtc-based video streaming over power constrained platforms. In: 2018 IEEE Global Communications Conference (GLOBECOM), pp. 1–7. IEEE (2018)
6. Bonald, T., Massoulié, L., Mathieu, F., Perino, D., Twigg, A.: Epidemic live streaming: optimal performance trade-offs. In: ACM SIGMETRICS Performance Evaluation Review, vol. 36, pp. 325–336. ACM (2008)
7. Castro, M., Druschel, P., Kermarrec, A.M., Nandi, A., Rowstron, A., Singh, A.: Splitstream: high-bandwidth multicast in cooperative environments. In: ACM SIGOPS Operating Systems Review, vol. 37, pp. 298–313. ACM (2003)
8. Giroire, F., Huin, N.: Study of Repair Protocols for Live Video Streaming Distributed Systems. In: IEEE GLOBECOM (2015)
9. Giroire, F., Modrzejewski, R., Nisse, N., Pérennes, S.: Maintaining balanced trees for structured distributed streaming systems. In: International Colloquium on Structural Information and Communication Complexity, pp. 177–188. Springer (2013)
10. Hei, X., Liang, C., Liang, J., et al.: Insights into PPLive: A Measurement Study of a Large-Scale P2P IPTV System. In: International Word Wide Web Conference. IPTV Workshop (2006)
11. Hoque, M.A., Siekkinen, M., Nurminen, J.K.: Energy efficient multimedia streaming to mobile devicesa survey. IEEE Communications Surveys & Tutorials **16**(1) (2014)
12. Kito, T., Fujihashi, T., Hirota, Y., Watanabe, T.: Users' demand-based segment scheduling for progressive multi-view video transmission. In: IEEE GLOBECOM (2018)
13. Li, B., Qu, Y., Keung, Y., Xie, S., Lin, C., Liu, J., Zhang, X.: Inside the new coolstreaming: Principles, measurements and performance implications. In: 27th IEEE International Conference on Computer Communications (2008)
14. Li, B., Wang, Z., Liu, J., Zhu, W.: Two decades of internet video streaming: A retrospective view. ACM Trans. Multimedia Comput. Commun. Appl. (2013)
15. MacMillan, D., Bensinger, G.: Amazon to buy video site twitch for $970 million (2014). URL http://www.wsj.com/articles/amazon-to-buy-video-site-twitch-for-more-than-1-billion-1408988885
16. Nihei, K., Yoshida, H., Kai, N., Satoda, K., Chono, K.: Adaptive bitrate control of scalable video for live video streaming on best-effort network. In: 2018 IEEE Global Communications Conference (GLOBECOM), pp. 1–7. IEEE (2018)
17. Park, J., Hwang, J.N., Wei, H.Y.: Cross-layer optimization for vr video multicast systems. In: 2018 IEEE Global Communications Conference (GLOBECOM) (2018)
18. Seufert, M., Egger, S., Slanina, M., Zinner, T., Hoßfeld, T., Tran-Gia, P.: A survey on quality of experience of http adaptive streaming. IEEE Communications Surveys & Tutorials **17**(1), 469–492 (2015)
19. Tomassilli, A., Huin, N., Giroire, F.: The structured way of dealing with heterogeneous live streaming systems. Tech. rep., Inria
20. Tran, D., Hua, K., Do, T.: Zigzag: an efficient peer-to-peer scheme for media streaming. In: IEEE INFOCOM (2003)
21. Vu, L., Gupta, I., Liang, J., Nahrstedt, K.: Measurement of a large-scale overlay for multimedia streaming. In: ACM HPDC (2007)
22. Wang, F., Xiong, Y., Liu, J.: mtreebone: A hybrid tree/mesh overlay for application-layer live video multicast. In: IEEE ICDCS (2007)
23. Zhang, X., Liu, J., Li, B., Yum, T.: Coolstreaming/donet: a data-driven overlay network for peer-to-peer live media streaming. In: IEEE INFOCOM (2005)