Distribution Trees' Analysis of PULSE, an Unstructured P2P Live Streaming System

Diego Perino and Fabien Mathieu

France Telecom R&D, 38 Rue du General Leclerc 92130 Issy-les-Molineaux

Pulse est un protocole de *streaming live*, basé sur un réseau décentralisé non-structuré, qui a été testé avec succès sur simulateur, Grid'5000 et PlanetLab. Dans ce papier, nous étudions les arbres de diffusion de Pulse et montrons qu'ils ont des caractéristiques proches des arbres construits à l'aide de réseaux structurés. Nous étudions également le delai de réception d'un stream diffusé par PULSE, et montrons qu'il est comparable à celui de réseaux structurés.

Keywords: P2P, Live Streaming, Distribution Trees

1 Introduction

During last years P2P has become a very popular approach to live stream multimedia contents because of client-server model and IP multicast limitations. Existing P2P live streaming systems can be divided in three main categories : *(i) structured*, where peers are organized in hierarchical tree or multiple-tree structures and the stream is forwarded upon them (e.g. Nice [10], SplitStream [7] , PrefixStream [4]) ; *(ii) unstructured*, where the stream is divided in a sequence of pieces spread among peers through local exchanges without following pre-defined structures (e.g. CoolStreaming/DONET [3], Chainsaw, LiveSwarm, PRIME) ; *(iii) hybrid*, where both piece exchanges and hierarchical structures are used for data delivery (e.g. Bullet [9]). Despite of the existence of many proposals, only few systems have been implemented and are used today.

We designed and implemented PULSE [1], an unstructured P2P live streaming system available under LGPL license. In PULSE the stream is divided in a series of FEC-encoded pieces (chunks) exchanged among peers in order to retrieve the complete sequence and to play out the stream. Every peer has a time sliding window containing the chunks the peer is trying to retrieve from the network and a buffer to store the received chunks. The average age of the sliding window represents the average lag of a peer with respect to the source and is an indicator of a node's position in the system. The data exchange is mainly regulated by the peer and chunk selection algorithms. The peer selection algorithm runs every constant period of time (called *epoch*) and it is based on an incentive TFT mechanism similar to the one used in BitTorrent. Every *epoch* a peer renegotiates N_{TFT} data upload slots by choosing the $N_{TFT} - 1$ peers which upload most data to it in the last epoch and the peer having the lowest lag distance [†] among all known peers. If the peer has still available bandwidth, it will select more peers by using an *history* parameter which indicates the quality of previous exchanges with other peers. These last partners will be served with less priority. The chunk selection algorithm used by all other peers is based on explicit recipient requests by using a rarest first policy among known peers.

The chunks are generated and injected by a source which, every *epoch*, selects N peers to upload data to by randomly choosing among the ones with lower lags. It simply pushes new chunks to these selected peers. A more complete description of PULSE is reported in [1].

Since PULSE is an unstructured system, it doesn't explicitly build data delivery paths but piece exchanges are driven by data availability at peers, so that every chunk follows a different path from source to nodes generating its own distribution tree. In this paper, through experiments and simulations, we focus on PULSE's chunk distribution trees and stream reception delay, trying to find out some general trends and properties, and comparing these results to theoretical properties of other existing protocols.

[†] It is the difference between the local peer's lag and the lag of the remote peer.

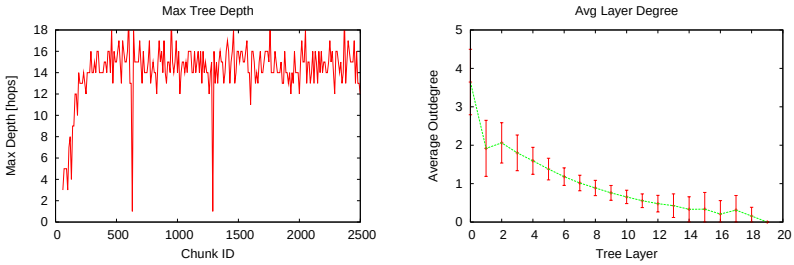

FIG. 1: 500 peers and $S = 4 * SBR$

Section 2 presents PULSE's distribution trees while Section 3 speaks of stream reception delay.

2 Distribution tree properties

In order to study PULSE's distribution trees we run our prototype on Grid'5000 platform. This testbed allows a complete control of test environment and the results are not affected by high CPU load or bandwidth bottlenecks because of Gbit links between hosts and the reservation mechanism. To allow a comparison with structured systems, in our experiences we limit the peers upload bandwidth to $1.1 * SBR$(Stream Bit Rate)[‡]. This makes us close to structured systems : these systems often consider a homogeneous scenario, where all peers have an upload bandwidth equal to SBR. PULSE works well in heterogeneous resource scenarios, even when the global available bandwidth is scare, and it is studied to react well in front of unpredictable peer behaviors. so, homogeneity is kind of worst case scenario for PULSE which we use better understand distribution trees. We run different experiments with increasing number of peers from 70 to 500 and with source upload capacity $S = 2 * SBR$ and $S = 4 * SBR$. The number of TFT upload slots is N_{TFT}=4, the stream rate is 512 Kbit/s, the sliding window size is 4.0 s and the epoch duration is 2.0 s.

Figure 1 shows the evolution of distribution trees' depth over time and the peers' average degree for each tree layer. These values are obtained by averaging the peers' degree of a given layer for every chunks. First, we notice that, even if each chunk follows a different path, the length of the different distribution trees is almost constant for all chunks. The source, corresponding to layer 0, distributes every chunk to 3-4 different peers if its nominal upload bandwidth is set to $S = 4 * SBR$. Sometimes it cannot distribute a chunk to a peer because of practical problems like connection managements. Starting from level one, chunks are re-distributed by peers with upload capacity equal to the stream rate. Peers belonging to level 1 , 2 and 3 show an average degree of about 2 while the ones belonging to level 4, 5 and 6 have an average degree between 1 and 2. The remaining peers present an average degree of 1 or less. In such bandwidth conditions, the first levels of distribution trees behave on average like a binary tree while, going down to leaves, the number of peers having just one child increases. However, peers have an upload capacity equal to the stream rate and they could not upload faster than this rate. To explain this behavior, we analyze the position of nodes in different distribution trees. We notice a peer is an interior node for 55% of chunk distribution trees and a leaf in the remaining 45%. A peer rarely stays in the same level for two consecutive chunk-trees(just in 10% of cases). When it happens the peer is usually in the bottom part of the considered trees. In most cases, a peer uses its available bandwidth to distribute two times the same chunk and it is a leaf in the distribution tree of the following chunk. A peer can also distribute one copy of two consecutive chunks or, less frequently, it can be a leaf for two consecutive chunks. So PULSE's distribution trees show properties similar to a *multiple tree of degree two*, particularly in their upper layers, and also respect their disjointness property. Moreover, if we look at peers' load distribution, we notice it is close to the stream rate for all nodes involved in the experience. This is another important characteristic of multiple tree structures which assure the load is *perfectly distributed among nodes*.

[‡] We cannot limit the bandwidth to exactly the stream rate because of protocol overhead.

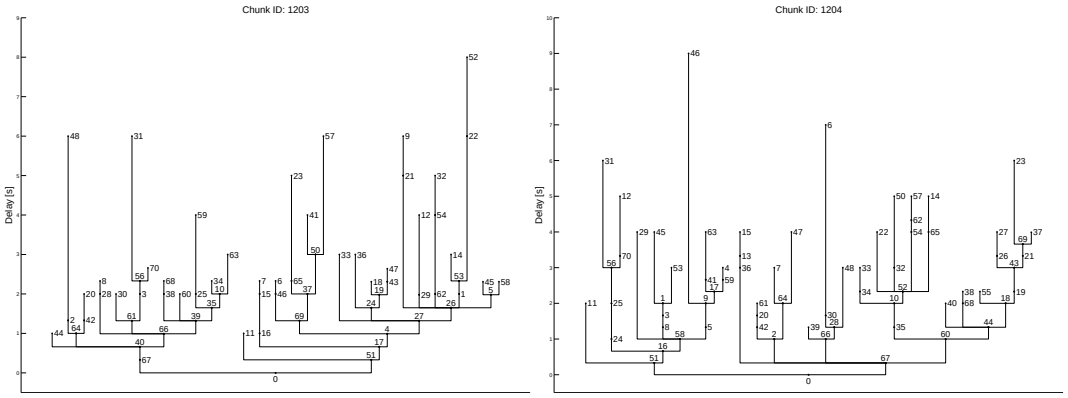

FIG. 2: 70 peers and $S = 2 * SBR$

In order to explain why distributions trees behave like binaries trees it is necessary to consider the chunk distribution mechanism. When a chunk A is injected in the system by the source, it is rare and it is owned by peers with a small buffer delay. Chunk A is required by lot of nodes, so owning peers use their upload bandwidth for its spread. Before chunk A has reached all the peers, a new chunk B is injected in the system by the source and becomes the rarest one. However, peers spreading chunk A cannot spread chunk B because they are already using their upload capacity. Thus other peers with small delay and free upload bandwidth should do that. After a few moments chunk A and B are both partially spread in two binary sub-trees (sub-tree A and sub-tree B). Now peers of sub-tree A send the chunk A to peers of sub-tree B and vice versa. Peers of sub-tree B (A) receiving chunk A (B) can send it to upper layers of their sub-tree. This explains why distribution trees present a first part similar to a binary tree and a second part more chaotic. The same results are also obtained in [6] where the two spreading phase are called *diffusion* and *swarming*.

Figure 2 shows the distribution trees of two consecutive chunks ; peers receiving the chunk with the same delay are plotted on the same row and thus peers of the same distribution layer can be on different rows. We observe the behaviors explained before but also some particular phenomena. As said before, PULSE's distribution trees are not built by following strict rules but derive from local exchanges driven by data availability. As a consequence of this exchange mechanism, we can just find out some general and averaged trends instead of strict rules, because particular behaviors can be found on each tree. For instance in Figure 2, peer 61 and peer 37, are central nodes of degree 2 for chunk 1203 and leaves for chunk 1204. Peer 6 is twice a leaf probably because of its big buffer delay, while peer 54 just distributes once both chunks. As example of a particular behavior we can mention node 27 that distributes chunk 1203 tree times ; this can be explained by the fact that the third copy is made with 1.5 seconds of delay and thus node 27, has newly available bandwidth to distribute again this chunk. By decreasing the source bandwidth capacity, we could expect important degradation on the system performances. But this is not the case : if we half the source capacity, the delay only increases of less than one second over seven (from $S = 4 * SBR$ to $S = 2 * SBR$).

3 About reception delay

Even if PULSE's distribution trees present some theoretical properties, we have not proposed a complete and rigorous model yet. Some models have been proposed for simpler protocols and network conditions [8] but they are not suitable to our system. In order to analyze the stream reception delay, we thus implement a simulator of the PULSE system. We run simulations with 500 nodes having upload bandwidth equals to the stream rate, the source upload capacity is 4*SBR and all the other parameters are set as in the previous section. Once again this scenario misses lot of good properties of PULSE system but is useful to compare it to structured system. Note that, in PULSE chunks are stored in a buffer whose delay may vary during time. We decided to take the *biggest* delay among all nodes during all simulation time. As we can expect from distribution tree analysis the delay achieved by peers is almost constant over time with a peak of 2.5 seconds. For structured systems the reception delay can easily be computed because of the mathematical properties

Protocol	Maximal delay d=2 [s]	Maximal delay d=4 [s]	Buffer overhead [s]
Optimal	1.18		-
Single tree	2.24	2.24	$O(0)$
Multiple tree (SplitStream)	2.24	2.24	1/SBR
Cluster tree	2.24 (m=2) 2.61 (m=5)	2.61(m=4) 2.86 (m=9)	(m-1)/SBR
ZigZag	4.48 (m=3) 71 (m=9)	2.24 (m=3) 35.86 (m=9)	$O(0)$
PrefixStream	2.49(m=2) 3.24(m=8)	2.49(m=2) 4.61 (m=16)	(m-1)/SBR
PULSE	2.5		4.0

TAB. 1: Comparison between PULSE and existing protocols

of data delivery paths. So it is not necessary to implement them in order to find out their performances. The values contained in Table 1 are derived from [4] by setting the parameters according to our experimental environment. The number of nodes involved is n=500, everyone providing a bandwidth exactly equal to the stream rate, and we assume the latency is negligible with respect to the time required to emit a chunk (L«T). As concern the nodes' degree we compute the delay for d=2 and d=4.

Table shows the delay achieved by PULSE is about *twice the optimal delay*, and our system performs well with respect to other structured protocols (see Table 1).

4 Conclusions

In this paper, we studied PULSE's chunk distribution trees derived from real testbed experiments. We showed that the PULSE's unstructured approach, which leads to more resiliency and adaptiveness with respect to structured systems, is able to build distribution trees with similar properties to structured protocols. We then analyzed, through simulation, PULSE's stream reception delay. We showed PULSE, with an unstructured approach, is able to achieve delays that can be compared to structured approaches.

Références

[1] F. Pianese, J. Keller, and E. W. Biersack, "PULSE, a Flexible P2P Live Streaming System", in Proc. of IEEE Global Internet Symposium 2006

[2] Grid'5000 - Grid platform testbed - `http://www.grid5000.org`

[3] X. Zhang, J. Liu, B. Li, and T.-S. P. Yum, "CoolStreaming/DONet : A Data-driven Overlay Network for Live Media Streaming", in IEEE INFOCOM '05, March 2005

[4] A. Gai and L. Viennot, "PrefixStream : A Balanced, Resilient and Incentive Peer-to-Peer Multicast Algorithm", INRIA research repor 5514, March 2005

[5] R. Karp, A. Sahay, E. Santos, and K. Schauser, "Optimal broadcast and summation problem in the logp model", In Proc of ACM Symp. on Parallel Algorithms and Architectures (SPAA), pages 142-153, 1993

[6] N. Magharei, R. Rejaie and Y. Guo, "Mesh or Multiple-Tree : A Comparative Study of Live P2P Streaming Approaches"', to appear in Proc. IEEE INFOCOM '07, May 2007

[7] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, "SplitStream : High-Bandwidth Multicast in Cooperative Environments", in ACM SOSP '03, October 2003

[8] L. Massoulie, A. Twigg, C. Gkantsidis, and P. Rodriguez, "Decentralized broadcasting algorithms", to appear in Proc.IEEE INFOCOM '07, May 2007

[9] Dejan Kostic, Adolfo Rodriguez, Jeannies Albrecht, Admin Vahdat," Bullet : High Bandwidth Data Dissemination Using an Overlay Mesh",in Proc. of ACM SOSP, 2003

[10] Suman Banerjee, Bobby Bhattacharjee, Christopher Kommareddy, "Scalable Application Layer Multicast", Technical report, UMIACS TR-2002.