PULSE: an Adaptive, Incentive-based, Unstructured P2P Live Streaming System

Fabio Pianese, *Student Member, IEEE,* Diego Perino, Joaquín Keller, and Ernst W. Biersack, *Member, IEEE*

***Abstract*— Large-scale live media streaming is a challenge for traditional server-based approaches. To appropriately support big audiences, broadcasters must be able to allocate huge bandwidth and computational resources. The costs involved with such an infrastructure exclude all but the established content producers from exploiting the Internet as a distribution medium. Publishers of not-yet-popular content, unless they manage to properly predict their maximum audience size, will likely fail to dimension correctly their broadcast infrastructure. Peer-to-peer systems for live streaming allow the users to support content distribution by contributing their unused resources: this increases the scalability of the content distribution while reducing at the same time the economical burden on the streaming provider. This paper presents and evaluates PULSE, an unstructured mesh-based peer-to-peer system designed to support live streaming to large audiences under the arbitrary resource availability as is typically the case for the Internet. PULSE is a highly dynamic system: it constantly optimizes its mesh of data connections using a feedback-driven peer selection strategy that is based on pairwise incentives. We evaluate the behavior of PULSE under realistic scenarios via simulation and emulation, and present the advantages of our approach, namely a best-effort response to system-wide resource scarcity, high resilience to node churn, and good hop-count properties of the average data distribution paths.**

***Index Terms*— Live Streaming, Peer-to-Peer, Unstructured, Mesh-Based, Incentives**

I. Introduction

PEER-TO-PEER live streaming has been a hot research topic for the last few years. The implicit constraints of this application, both on *media quality* and *timeliness of stream reception*, are quite challenging when faced together: different approaches have been so far proposed, striking in all cases a trade-off between these two strongly-correlated factors.

The reception timeliness is defined as the delay (also known as *play-out delay*) between the generation at the source of a stream of frames and its reproduction at the receiver. The fundamental restriction of a generic streaming application is that the play-out delay, once it is fixed, cannot be increased or reduced without affecting the media quality. Increasing the play-out delay amounts to 'freezing' the media for a while, while reducing it requires a receiver to skip a segment of media playback. Live streaming, additionally, requires that the maximum play-out delay be *reasonably* low, ranging from few seconds to few tens of seconds. The constraint on play-out delay is more bound to the perception of the user that she is receiving fresh media data: live streaming deals with information whose interest to the user would rapidly decay if it was delayed too much.

Media quality, on the other hand, depends on the *completeness* of the data received before its playback deadline. The qualitative effect of incomplete stream data on the user's playback experience depends on several factors, including the media coding format, the presence of redundant encoding to protect the stream, the ability of the player application to hide discontinuities in the media, and the subjective user sensitivity to visual artifacts. In general, the shorter the play-out delay, the more difficult it is to retrieve a complete media stream.

PULSE is a peer-to-peer live streaming system designed to operate in scenarios where the bandwidth resources of nodes can be highly heterogeneous and variable over time, as is the case for the Internet. To support such network conditions, we must carefully think over the quality vs. timeliness trade-off. Historically, the attention of system designers has been primarily focused on timeliness. Striving to obtain the lowest latencies possible often resulted in simple structured system designs, where tree-based overlays are built following some optimization criteria: since trees scale well with respect to maximum path length, and node placement can be optimized by bandwidth or latency, this approach was (and still is) quite popular [5][6]. Media quality, however, does suffer when node instability is present, as it happens when there are rapid membership variations due to node arrivals or departures [3].

With PULSE, we set out to explore a radically different approach, which is focused on media quality rather than timeliness. We thus explicitly require that there be no 'holes' in the data sent to the player for media play-out: *the stream can be reproduced only as long as a continuous, uninterrupted data flow from the original media have been received*. No structural constraints are imposed a priori on the distribution process: *data can be freely exchanged among nodes* that associate for the purpose, creating a *system-wide mesh of data connections*. Node associations are *driven by pairwise incentives based on implicit feedback information,* such as the amount of data exchanged in the recent past and the current reception delay of the stream. The goal of our experiment was to evaluate the performance of such a mesh-based system, especially in scenarios where classical tree-based systems reach their limits. Our results show that, with the peer selection strategies used in PULSE, the average length in hops of the data paths on the mesh are comparable to those observed on tree-based

Manuscript received November 15, 2006; revised April 19, 2007 and July 31, 2007. This work was funded by France Telecom R&D.

F. Pianese, D. Perino, and J. Keller are with France Telecom R&D - Orange Labs, 38-40 Rue du Général Leclerc, 92794 Issy-les-Moulineaux, France (e-mail: firstname.lastname@orange-ftgroup.com).

E. W. Biersack is with the Corporate Communications Department of the Institut Eurecom, 2229 route des Crêtes, 06904 Sophia-Antipolis, France (e-mail: erbi@eurecom.fr).

overlays. Moreover, due to the higher number of connections that compose the mesh, PULSE is more resilient than typical structured systems against node transience, and is able to support heterogeneous and asymmetrical node capabilities.

In this paper, we present the insights and the algorithms that stand behind the design of PULSE, along with a thorough evaluation of its performance based both on simulations and actual experiments. In Section II, we introduce the system design, its basic concepts and terminology. Section III contains the algorithms that are executed by all PULSE peers to associate with each other and to perform data exchange. The core part of this work is the analysis of simulated and experimental system traces: in Section IV we describe the simulation methodology, introduce appropriate metrics to investigate the global behavior of the peers, and leverage these metrics to understand the internal system dynamics. Section V analyzes and displays the experimental results we obtained by running our prototype node software on the Grid'5000 [22] testbed platform. A summary of relevant research work in the field is presented in Section VI. Finally, we draw our conclusions in Section VII.

II. PULSE: System Overview

This section presents the main insights behind PULSE, the basic terms and concepts, and the details on the structure of the system and of its components.

A. Design Principles

PULSE is Peer-to-peer: In PULSE, all nodes are identical except the source, which differs in that it is the first node to distribute the original stream. Nodes can freely exchange among themselves short membership messages containing information on the average stream reception performance. Based on this knowledge and on current measurements, the nodes temporarily associate to exchange the data. Over the resulting mesh of data connections, nodes can perform pairwise buffer reconciliation and engage in receiver-driven, bi-directional data exchanges.

PULSE is Mesh-based: The mesh-based approach was chosen for several reasons. First, in contrast with classical single trees, meshes offer many possible paths that data can traverse to get to their destination. Also, no nodes are prevented by structural constraints from contributing bandwidth to the system, as is the case for the leaf nodes in tree-based systems. Moreover, nodes can easily change position and timely react to both membership changes and bandwidth capacity fluctuations: a mesh-based approach gives more flexibility, decreasing the negative effects of node transience. Finally, because of its adaptive properties, we found a mesh organization more adequate to support incentives. The price to pay, compared to tree-based systems, is an increased local exchange of control messages to support and coordinate the data reconciliation process, avoiding unnecessary data duplication.

PULSE is Unstructured: Among the benefits sought by our mesh-based design is a high resilience to churn, which could be lost if the mesh itself depended on a structured substrate. Structured systems have the advantage to offer *geometrical* properties, which make their design and analysis straightforward [2], but especially for an application that does not require distributed keyword-based search we believe that the added complexity of an underlying DHT would not justify the possible small bandwidth saving over routed membership control traffic. An unstructured, randomized gossip membership protocol [16] is therefore better suited to distribute membership information with low overhead.

PULSE is Incentive-based: Incentives to share appear to us a good choice as local policies for global system optimization, especially in scenarios where peers may be *non-cooperative* either by deliberate will or by inherent lack of resources. Several papers [19][20] analyze the effects of the tit-for-tat incentive on BitTorrent [17], concluding that it allows the best resource contributors to associate and get higher download performance. This autonomous organization has in turn a positive effect on the global system performance. Especially in a live streaming context, we agree with Chu et al. [4] that placing nodes in the system according to their available bandwidth is critical to improve the data replication efficiency, in terms of both reception delay and of media quality.

Insights from these and other sources [13][18] inspired our decision to combine together two incentive mechanisms: a primary *optimistic tit-for-tat* peer selection policy, and an additional *excess-based altruistic* incentive. Intuitively, the primary mechanism should foster cooperation among resourceful nodes, while the other should both facilitate peer discovery and allow the richest nodes to contribute more effectively to the system. Altruism, however, remains optional and must never undercut the primary incentive-based mechanism.

We want to stress the fact that the fundamental role of incentives in our system is not to *enforce fairness understood as equality of contribution* between nodes. Instead, incentives are primarily intended to optimize the system structure for better global performance. For instance, resource-rich nodes located near the source are able to serve a large number of neighbors with more recent data: they benefit, in terms of reduced play-out delay, and the whole system benefits, in terms of reception performances. If nodes whose resources are scarce were systematically served recent data, either directly by the source or indirectly by normal peers, they could slow down or disrupt the distribution process. Also, since time constraints for live streaming are quite strict, we believe it is more important to fully exploit the available resources than to enforce a strict reciprocal incentive for its own sake[1].

B. About the Media Stream

The stream is a continuous flow of media data generated at the streaming source. In the following, we assume that the bitrate of the stream (SBR) is constant. The source first splits the stream into a series of *chunks* of fixed size. At this stage, the source may also apply to the data a fixed-rate error correction code, such as FEC [11], to achieve a better resilience to chunk loss. Chunks are then numbered and marked with their original timestamp (we call this time reference the *media clock*) to allow peers to correctly rebuild the original data stream and

[1] Appropriate bandwidth limiting at the application level by the user is a more than adequate solution to avoid unwanted resource overconsumption.

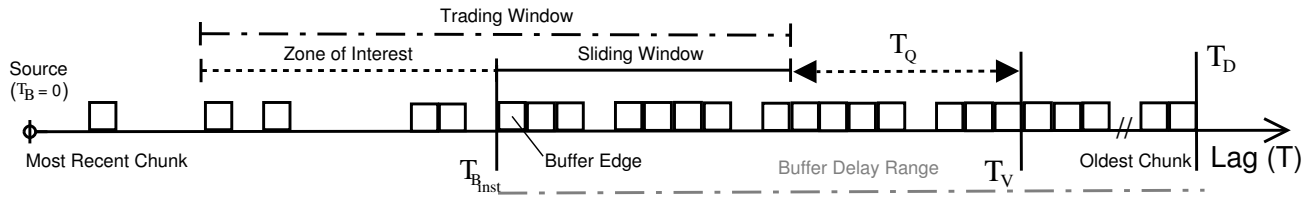


Fig. 1. The data buffer of a PULSE node

help them estimate their own play-out delay. Chunks are then made available to the nodes at a *constant rate*, i.e. always maintaining a proportional relationship between chunk IDs and the media clock[2]. This choice allows nodes to estimate delays based on chunk IDs and to predict chunk IDs given delay values in a straightforward way.

C. Lag Reference System

In Figure 1 we illustrate the fundamental concepts and variables used throughout this paper. The horizontal axis represents the *lag*, which is defined as the age of a chunk with respect to the current media clock. The "newest" chunks are on the left, while the "oldest" ones are on the very right. The lag value of a given chunk grows over time, as new data are encoded at the source and present data become older.

We chose this reference system because it eases the representation of the buffer dynamics. For instance, since the play-out rate is constant, the chunk a node should be playing at any given moment is described by a fixed lag value, which we will call T_V. This reference system also allows us to define the range of chunks a node is both interested in receiving and capable to provide at some point in time by two values: the *average lag of the chunks the node is requesting*, $T_{B_{avg}}$, and the *lag of the chunk a node is going to discard* from its buffer, T_D. We will better define the meaning of $T_{B_{avg}}$ later, in Section II-D.1. From now on, we will call the interval $[T_{B_{avg}}, T_D]$ the *buffer delay range* of a node.

The knowledge of the buffer delay range of a remote node is mainly useful in the phase of peer discovery, when it is important to find nodes that are able to provide useful chunks. We can imagine that, when the system is in a steady state, nodes tend to settle on constant average reception delays. In this situation, in order to discover a potential partner, it is sufficient to compare at any time the buffer delay ranges of the nodes. This can eliminate the need of continuously sending and requesting updated buffer information on a chunk-by-chunk basis. On the other hand, when the system is not in steady state, buffer delay ranges can fluctuate. However, the information on the buffer delay range is still much less volatile than the information on single chunks or chunk ranges: in normal operating conditions (i.e. while most nodes manage to retrieve chunks at a sufficient rate on a regular basis), a node's reception delay will typically change quite slowly over time. It is thus still possible to use the buffer delay ranges, within a reasonable time frame after their computation, as an approximate and concise representation of the current buffer content at the remote node.

[2]In the general case involving variable-bit-rate media streams, we still keep the rate of chunk generation fixed to conserve the time-ID relationship. Chunk size is then no longer fixed, but typically fluctuates around an average value.

D. Peer Structure

A PULSE peer is an application that interfaces with the network to steadily retrieve data chunks and control messages. Its goal is to reconstruct the original stream of media data and to pass it on to the player software. Its main components are 1) the *data buffer*, where chunks are stored before playback, 2) the *knowledge record*, where information is kept about remote peers' presence, data content, past relationships, and current local node associations, and 3) the *trading logic*, whose role is to request chunks from neighbors and to choose and schedule the ones that are to be sent.

1) Data Buffer: Each node has a buffer to collect and store data chunks prior to playback (Figure 1).

Definitions: The buffer uses a sliding window to regulate the stream reception. The *sliding window* is W chunks wide. Its goal is to output a stream of chunks with a desired maximum loss ratio. We call *buffer edge* the leftmost end of the sliding window. A second window, called *zone of interest*, lies on the left of the buffer edge and covers the chunks that will soon be needed as the sliding window moves. We refer to the sequence of chunks covered by these two windows as the *trading window* of the peer, since it contains all chunks the peer is trying to obtain through exchanges with neighbors.

We define as *instantaneous position* of a node in the system, referred to as $T_{B_{inst}}$, the lag value of its buffer edge from the source. This value can fluctuate quickly, so nodes keep a running average of their instantaneous position, previously referred to as $T_{B_{avg}}$, to filter the short-term position variation due to the unpredictable delays of the data exchange process.

Operation: A parameter S, called *sliding tolerance*, defines the minimal amount of chunks that have to be present inside the sliding window before it can move forward. The maximum chunk loss rate tolerated during normal peer operation is thus bound by $LR = 1 - \frac{S}{W}$. The system-wide parameter LR_{max} is equal to the amount of redundant coding performed by the source. The value of S at any peer must be set so that $LR \leq LR_{max}$ to ensure the complete recovery of the original stream. If less than S chunks are available, the sliding window cannot move. The lag of all the chunks contained in the window increases as time passes and as new chunks are generated. In this situation, the window keeps drifting on the lag axis (to the right in Fig. 1) and $T_{B_{inst}}$ grows over time at constant speed. Only when at least S chunks over W have been collected, the window is allowed to slide and to reduce its $T_{B_{inst}}$ (to the left in Fig. 1). The window will then keep sliding as long as it contains at least S chunks.

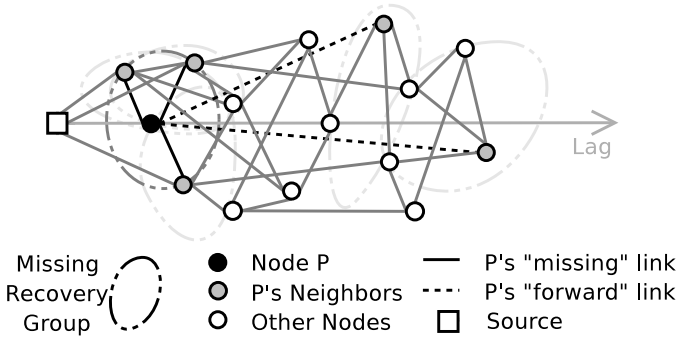


Fig. 2. A PULSE peer and its exchange sets (MISSING and FORWARD)

Initialization: After a node joins the system, it begins requesting chunks in the $[0+\delta,\ W+\delta]$ fixed delay interval[3]. As chunks are retrieved, they are put into the buffer: in this initial phase, the sliding window is not yet enabled, and chunks are requested with the goal of forming a nearly-contiguous block. At t_0, when at least $\frac{W}{2}$ over W contiguous chunks have been collected, the sliding window mechanism is first enabled around that block of chunks, and the initial buffering lag $T_{B_{INIT}} \equiv T_{B_{inst}}(t_0)$ value is set to the current buffer edge. Subsequently, the buffer keeps operating as described above. The play-out delay T_V is determined after time t_0, when the node buffer at least contains a continuous sequence of $T_{Q_{INIT}}$ chunks. Only at time $t_1 > t_0$, when $T_{B_{INIT}} - T_{B_{inst}}(t_1) = T_{Q_{INIT}}$, the media play-out is allowed to begin. T_V is then set at time t_1 to be equal to $T_{B_{inst}}(t_1) + T_{Q_{INIT}} + W$.

Reaction to Chunk Shortage: We define T_Q as the interval of chunks ranging from the rightmost end of the sliding window to the chunk being currently played. $T_Q(t)$ contains the chunks that are ready for play-out at time t. During play-out, as $T_{B_{inst}}$ is free to change and since T_V must remain constant, $T_Q(t)$ is always equal to $T_V - (T_{B_{inst}}(t) + W)$. T_Q may drop to zero due to reception shortage (i.e. when $T_{B_{inst}} + W = T_V$): this means that there is no more data that can be copied from the buffer to the player without disrupting the integrity of the stream. If the shortage persists, $T_{B_{inst}}$ may grow up to T_D, at which point a node is forced to reset its $T_{B_{inst}}$ value and repeat the initialization procedure detailed above. The visible impact for the user is a temporary interruption of playback.

The role of the T_Q interval is twofold: it grants a safety margin against variations of $T_{B_{inst}}$over time, and the changes in its size can be used by peers to evaluate their own data reception stability. For instance, the rapid decrease of T_Q is the typical consequence of an impending reception shortage: nodes can leverage this information to preemptively react to avoid playback disruption.

2) Knowledge Record: The strict timing constraints on the data retrieval process emphasize the central importance of the concept of node buffer delay range. In PULSE, the position (in terms of lag) of a node buffer carries two pieces of information: an explicit one, that is the range of chunks a node is able to serve, and an implicit one, which is an estimate of the peer's trading capabilities related to the incentive mechanisms. Intuitively, cooperating peers will be able to receive new chunks faster than selfish ones, and thus will find themselves nearer to the source, i.e. have lower $T_{B_{avg}}$ values.

Node decisions are based on the currently available local knowledge, which includes:

- direct measurements of network parameters (RTT, data throughput per connection)
- information about the *network address* and buffer delay range $[T_{B_{avg}}, T_D]$ of the other peers. This information is exchanged among all peers using a gossip/epidemic protocol such as SCAMP [16]. Nodes known with this level of detail can be chosen as partners for data exchange.
- detailed accounts of the exact buffer content of the remote peers that are currently engaged in data exchange with the local peer. This information comprises the instantaneous node position $T_{B_{inst}}$, T_D, a bitmap summarizing the chunks present in the trading window, and (optionally) explicit request bitmaps for chunks in that range. Usually, nodes known with this level of detail are currently engaged in data exchange with the local peer.
- local records of previous trading interactions, in the form of a cumulative history score H.

The peers a node P is trading data with fall into two groups, MISSING and FORWARD (Figure 2). Peers in the MISSING set are selected to favor reciprocal data exchanges, and in a way to insure that both parties have chunks the other one needs. The FORWARD peer set helps introduce a component of altruism in the system and, at the same time, allows resourceful nodes to contribute more of their upload capacity to the system. The details of data exchange will be explained in Section III.

3) Trading Logic: The trading logic controls all the aspects of chunk request, selection, and scheduling. It processes the information coming from both the local buffer and the knowledge logic to decide which chunk will be requested/sent, and from/to which neighbor.

III. ALGORITHMS

The algorithms presented in this section make up the core of the PULSE system. They determine how each node chooses its partners for data exchange, how chunks to be sent are chosen and scheduled, and which chunks are to be requested from each neighbor. The following algorithms are all based on the assumption that peers *(i)* have some knowledge of the other peers in the system, acquired from the underlying membership information protocol, *(ii)* can determine their position in the stream with respect to the media clock (in our implementation we use a loose, distributed NTP-like synchronization mechanism for this purpose), and *(iii)* know or can estimate the maximum upload bandwidth they will be able to provide.

A. Peer Selection

Peer selection is periodically performed by each node, its time period is called *EPOCH* and is constant. PULSE uses

[3]The δ parameter introduces an offset in the initial lag interval where incoming nodes request their first chunks. Requesting chunks with higher δ may increase the speed of node initialization (when resources are available) as older chunks are usually better replicated in the system. We currently use a fixed value of $\delta = \frac{3}{8}W$.

two peer selection algorithms: an optimistic tit-for-tat selection based on the total amount of data received during the previous EPOCH (similar to BitTorrent) and a lag-constrained selection based on a cumulative trust score. The two algorithms are executed at the start of each EPOCH, and give as a result two lists of peers, the MISSING and the FORWARD list. During the next EPOCH, the local node attempts to associate and exchange data preferentially with the peers in these lists: chunk requests received from peers in the MISSING list are honored with higher priority, followed by peers in the FORWARD list, and finally by requests from other nodes.

Optimistic Tit-for-Tat: This policy aims to identify which peers, among all those about which a node has knowledge, are currently interested and able to provide data in the short term. Two pieces of information are then relevant to this choice: the fact that a peer has provided data in the recent past and may be expecting a short-term compensation to continue to do so, and the presence of a shared interest in the same window of the stream which may lead to fruitful future exchanges.

The selection policy we employ in PULSE uses a tit-for-tat choice based on information about the amount of data received during the previous EPOCH to fill the MISSING list. At least one place in the list is reserved for an optimistic selection, leading to the choice of a known node with the largest trading window overlap (network latency can be taken into account to bias this selection toward peers 'in the vicinity').

History Score: Every node maintains a record of the previous interactions with every other peer as a numeric value, which we refer to as the *history score*. This mechanism enables a peer to use data on past behavior of its fellow peers to make *informed choices* when selecting future candidates for FORWARD exchanges. The history score is computed as follows: each time a previously unknown peer is encountered, it is given an initial positive score. The score is incremented by a fixed value whenever useful chunks are received from a node while it does not belong to the MISSING/FORWARD lists. The score is decreased by some fixed quantity whenever it is chosen as FORWARD partner and receives one or more chunks from the local peer during that EPOCH.

As it is currently defined, the history mechanism can appear rather simplistic, but it proved effective to evenly distribute altruistic contributions among the peers. We believe that the original incentive model proposed by GnuNet [18] could eventually be applied to our system, further improving the strength of the relationships among resourceful nodes.

B. Bandwidth Allocation

At any given moment, each peer must maintain several connections for sending and receiving data. To simplify the problem of bandwidth allocation, PULSE peers try to establish a fixed number of outbound connections for data exchange, but do not limit the number of incoming ones. As node bandwidth is typically asymmetric, with the upload bandwidth being much smaller than the download bandwidth, it is mainly important to control the number of outbound connections.

The biggest challenge for the bandwidth allocation mechanism is the need to support upload bandwidth heterogeneity: especially in a live streaming application, it is critical to make all nodes contribute, since unused upload bandwidth reduces overall system capacity. Opening multiple connections has two benefits: a node is able to provide service to several peers, and it obtains more information to support its future exchanges. However, the more connections, the higher is the control message overhead for each node. Also, when the upload bandwidth can vary widely, it is difficult to set a fixed 'number of connections' parameter that works for all the nodes in the system. In PULSE, we approach the bandwidth allocation problem in a practical way: to avoid relying on a dynamical solution, which would greatly increase the complexity of the system response, we instead split the set of connections into two lists, MISSING and FORWARD, and use a single common set of parameters on all peers.

We believe that a large number of MISSING connections can reduce the effectiveness of the tit-for-tat selection[4]: we must in fact remember that at steady state, for a *rate-limited* application such as live streaming, no more than SBR bytes per second will be received on average by each node. To clarify this point, let us suppose that MISSING connections alone are sufficient to sustain the reception by a peer of the full stream, and that each peer opens exactly n MISSING connections to other nodes: each node will be selected on average by n peers as MISSING partner. Intuitively, a large n means a lower expected throughput from (and to) each MISSING connection: as the contribution threshold required to gain a place in the MISSING list of the remote node becomes lower, associations become more random and less related to the actual resource availability at the nodes, and system performance may suffer because of repeated sub-optimal choices. For this reason, we decided to use a small number of connections (e.g. four) to MISSING partners, so that each peer can expect a meaningful theoretical throughput on each connection (e.g. SBR/4).

On the other hand, especially for the richer nodes, opening more connections could improve the odds of finding useful chunks and fully exploiting their capacity. To take this fact into account, a variable number of connections can then be assigned to FORWARD exchanges, depending on the available outgoing bandwidth. We remember that having an open connection to some node does not imply that it will be used for data exchange, as that is determined by the chunk scheduling mechanism: however, these connections allow resourceful peers to donate their excess bandwidth to the system by providing a large number of other peers with recent chunks.

1) MISSING *List:* All the peers that sent us data during the last EPOCH are ordered by the number of non-duplicate chunks we received from them. A configurable quota (currently 3) of MISSING exchange connections is then established to the highest-ranked nodes. One connection slot is reserved for the optimistic choice: the known node with the largest trading window overlap[5] to ours is selected. If one or more

[4]The problem of defining the optimal number of connections under a "tit-for-tat peer selection" consumption game (in the rate-limited application case) is an interesting issue that we leave open for future work.

[5]In the optimistic selection, link latency can be introduced as a delay bias that is subtracted from buffer overlap. Our preliminary observations suggest that biasing choices by latency improves the awareness of the system to network locality.

tit-for-tat MISSING connection slots remain available, they are allocated *a)* to nodes with overlapping trading windows, in decreasing overlap order, and *b)* to randomly selected known nodes. Random selection is mainly used during bootstrap.

2) FORWARD *List:* Peers are ordered by decreasing history score, and selected only if their trading window is not overlapping with the local trading window (i.e., the remote node is currently "farther" from the source than the local peer). Nodes already belonging to the MISSING list are ignored.

3) About the Source: The source differs from the other peers since it doesn't need to engage in exchanges to get data chunks. It always has a complete sliding window, and its lag value is zero by definition. As a consequence, the peer selection algorithm at the source also needs to be different.

Moreover, the source lacks the data exchange feedback mechanism, and could be exploited by malicious nodes that try to retrieve all chunks directly. The attackers could then avoid contributing to the system and may even put in danger the entire distribution process, if the upload bandwidth available at the source is small. To mitigate this danger, the source has to change the subset of nodes it serves at each EPOCH, and must not send groups of contiguous chunks to the same peer.

The peer selection algorithm at the source is similar to the one used by seeds in the latest BitTorrent software versions [19]. At the beginning of each EPOCH, the source prepares a list of known peers that have a $T_{B_{avg}}$ value smaller than a fixed threshold. It then chooses randomly a subset to which it sends chunks during that EPOCH.

C. Chunk Selection: Sending

A good chunk selection strategy is one that distributes the chunks in an uniform way across the nodes to avoid situations where some chunks are much less replicated system-wide than others. It should also ensure that the buffer content of nearby nodes is different enough that they can engage in mutual transactions and concurrently exploit their multiple connections. Finally, it should prevent that several neighbors concurrently send duplicate chunks to the same node.

The chunks to be sent over a connection, regardless if MISSING or FORWARD, are selected comparing the requests received from each peer to the chunks currently held in the local buffer. Requested chunks that are available are then sorted using appropriate ordering criteria, and the first one is chosen for sending. The criterion we are currently using for ordering chunks at the sender is a "Least Sent First, Random" strategy. Each peer keeps a counter of how many times it has sent each requested chunk. The one that has been sent the least number of times is chosen to be sent first. In case of a tie, the chunk is selected randomly. It is indeed possible to queue several chunk uploads toward the same peer to benefit from the effects of transfer pipelining.

This scheduling strategy shows encouraging results, since the newest (and thus rarest, from the point of view of the sender) chunks to be received are among the first that are sent. Breaking ties with a random choice, instead of e.g. selecting the chunk whose lag is lowest, aims to avoid the preferential replication of a same single chunk which may happen in situations where several peers have their trading windows synchronized.

D. Chunk Selection: Requesting

The algorithm for chunk requests is similar to the heuristic used in DONet/CoolStreaming [8]. Its purpose is to request the rarest chunks among those that are locally available, and to distribute the requests across different possible providers.

Using the local knowledge gathered from the current neighbor set, chunks that are rarest across the neighborhood are requested with higher priority than more common ones. Chunks with an equal number of providers are preferentially requested to the MISSING neighbors that can provide them. To limit the load on any single peer, the maximum number of per-node requests is bounded.

IV. Simulation Results

We extensively evaluated the effectiveness of the algorithms in PULSE through simulation and limited deployment of a PULSE node prototype. The main purpose of this section is to support our previous claims with experimental data and real measurements, compensating for the current lack of satisfactory theoretical models. Because of space constraints, the following pages present only the most interesting results we have managed to obtain. We evaluated PULSE for different network sizes, access bandwidth distributions, buffer parameters, and node arrival patterns. We mainly concentrate our attention on the worst-case bandwidth availability and heterogeneity scenarios, which provide particularly meaningful insights that are representative of the whole result set.

The simulator: Our simulations are performed using a simple round-based simulator. The simulated network has a single-stub topology. Nodes are connected to the stub through access links whose bandwidths are configurable. Bandwidth allocation is performed using a slot-based mechanism: each slot allows the transmission of one data chunk between two nodes during one round. Network latencies are not taken into account: control information is propagated without delay, while only data transfers are affected by latency. While updated knowledge about current buffer delay ranges is available to everyone, the detailed content of the data buffer of a peer P is only known to nodes that have P in their MISSING or FORWARD exchange lists, or that received data from P during the current EPOCH.

The scenarios: In all the simulation scenarios, the source has a 4*SBR upload bandwidth. SBR is fixed at 16 chunks per second. The size of the sliding window is $W = 32$ chunks (2 seconds of stream data). The trading window is $2W$ chunks wide. The FEC rate LR_{max} is set to 20%. The number of MISSING connections is always 4. The duration of an EPOCH is set to 2 seconds. There are up to one thousand nodes in each simulation, and their access bandwidth distribution scenarios are the following:

HH-LB (high heterogeneity, low bandwidth) scenario - four bandwidth classes: 4% of VERY RICH peers, with 4*SBR upload and 4*SBR download bandwidth; 20% of RICH peers, with 2*SBR upload and 2*SBR download bandwidth; 21%

of NORMAL peers, with SBR upload and 2*SBR download bandwidth; and 55% of POOR peers, with SBR/2 upload and 2*SBR download bandwidth. This amounts to a per-peer average upload bandwidth of 1.045*SBR.
This scenario aims to show the behavior of the system when bandwidth resources are scarce and asymmetrically distributed throughout the population. The total upload capacity is only 4% higher than the total download bandwidth required to provide every peer with a complete stream.

HH-HB (high heterogeneity, high bandwidth) scenario - four bandwidth classes: 4% of VERY RICH peers, with 10*SBR upload and 10*SBR download bandwidth; 20% of RICH peers, with 3*SBR upload and 3*SBR download bandwidth; 21% of NORMAL peers, with SBR upload and 2*SBR download bandwidth; and 55% of POOR peers, with SBR/2 upload and 2*SBR download bandwidth. This amounts to a per-peer average upload bandwidth of 1.485*SBR.
Here we noticeably increase the upload capacity of the two richest bandwidth classes. As a consequence, the total available bandwidth exceeds the minimum amount required for the complete stream distribution by 50%. The resulting scenario aims to approximate the heterogeneous bandwidth distribution observed by recent studies [15] on resource availability in peer-to-peer file-sharing networks.

Common observations about all simulation runs include:

- The simulator parameters and bandwidth distribution ranges have been chosen to model the diffusion of a 1 Mbps FEC-protected stream.
- Only the evolution of the sliding window of the node buffer is actually tracked: simulated nodes do not actually set a play-out instant (T_V).
- The T_D value is large enough to be considered a threshold for irreversible receiver starvation, after which it is necessary to reset the buffer, which will result in the loss of a short media segment. Proactive shortage detection and recovery based on T_Q are not simulated to reduce complexity. We remember that, by performing simple estimates based on T_Q, as suggested in Section II-D.1, it is possible to increase the responsiveness of the system, reducing the time during which play-out is disrupted at a node as a consequence of a buffer reset event.
- In the initial phase, as well as under massive instantaneous arrivals, the simulated system often shows some instability. This is mainly an artifact due to the randomness associated with the simultaneous arrival process and the oracle-like knowledge model used in the simulator. Nodes that cannot connect at their first attempt have to wait up to T_D and then reset their buffer before retrying. This phase however rarely lasts more than 50 seconds, and the subsequent steady-state behavior is not affected.

A. Global Impact of Tit-for-Tat Peer Selection

Our first task will be to evaluate the effect of a simple tit-for-tat peer selection policy on the whole network structure. To do so, we set to zero the number of FORWARD connections, allowing nodes to only establish the four "base" MISSING connections. In Figure 3 we plot of the evolution over time of the average value by class of $T_{B_{avg}}$ for a simulated system under the HH-LB scenario, with 1000 nodes, simultaneous arrivals at $t = 0$, and no departures.

Figure 3a shows the evolution of the average class lag with no FORWARD connections. As we can see, the effect of tit-for-tat peer selection based on the bandwidth received in the previous EPOCH is quite interesting: few EPOCHs are indeed required for nodes to find a stable set of neighbor peers with sufficient resources to guarantee a steady supply of data chunks. In this case, for example, the system reaches this stable state for the first time after 50 seconds.

We will study how this convergence process works in more detail later in this section. For the moment, we observe in Fig. 3a that the different bandwidth classes appear to settle around different values of $T_{B_{avg}}$, with richer classes being nearer to the source than poorer ones. This is in agreement with our initial intuition that the coordinated effect of incentive-based (tit-for-tat) and performance-based (chunk interest range) peer selection would allow the generation of clusters of peers with similar resource availability.

A noticeable aspect of Fig. 3a is that the equilibrium reached by the system is quite unstable: between $t = 120s$ and $t = 200s$ the average lag suddenly increases for all classes, to the point that most nodes are forced to reconnect. The same thing appears to be happening after $t = 270s$. An accurate observation of the bandwidth usage plots (not included here) reveals that the available upload capacity is not completely exploited, chunk losses are quite high, and most nodes must rely on FEC to recover the original stream.

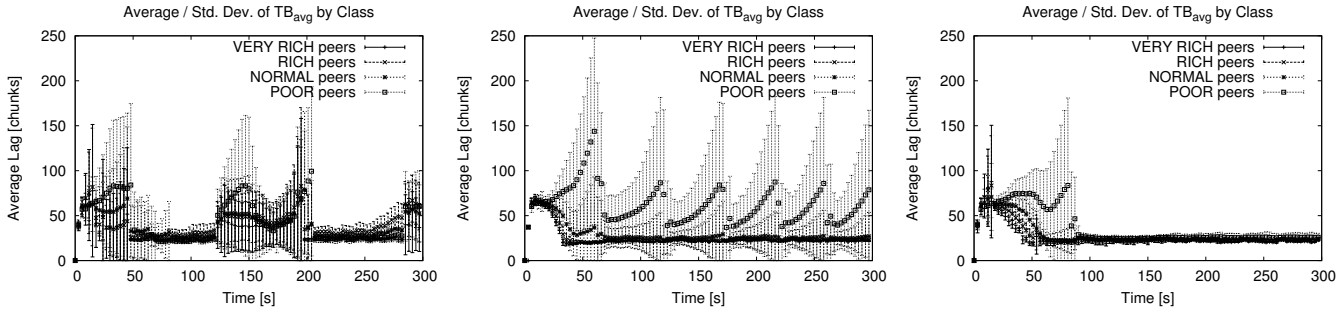
B. Tit-for-Tat vs. Tit-for-Tat+History

Inspecting the bandwidth traces, we notice that especially the rich peers are often contributing less than what they could actually offer to the system. To address this issue, we now evaluate the effect of enabling the second selection mechanism. We introduce FORWARD connections with the double goal of fostering altruism and reinforcing the main incentive: the history-based peer selection policy biases the altruism toward those nodes that contributed the most during the past interactions, but does not exclude poorer nodes and free riders. Simulations show that, by introducing connections to peers that otherwise would not qualify for tit-for-tat selection, the overall system performance and stability improve dramatically.

In Fig. 3b and 3c we see how the addition of (respectively) four and eight FORWARD connections impacts the performance of the previous scenario. The presence of only a few FORWARD connections has a strong stabilizing effect on the whole system, greatly reducing the variance of the three richest classes' lag. In Fig. 3b the poorest class is the only one to suffer from starvation, with periodic re-connections of a part of its population. It is interesting to notice that the subsequent re-connections of these poor nodes do not affect significantly the performance of the remaining classes. In Fig. 3c we see that in the case of eight FORWARD connections all classes manage to stabilize around a small lag value of about 30 chunks (i.e. about 2 seconds) with a low, constant variance. Bandwidth usage plots (again not included here) confirm a

	NO FWD				4 FWD				8 FWD			
Scenario	VR	R	N	P	VR	R	N	P	VR	R	N	P
HH-HB	0	0	23	181	0	0	0	0	0	0	0	0
HH-LB	13	53	59	242	0	0	4	183	0	0	0	0

TABLE I
BUFFER RESET STATISTICS: UNSTABLE PEERS BY CLASS AT STEADY STATE



(a) No FORWARD (b) 4 FORWARD (c) 8 FORWARD

Fig. 3. HH-LB Scenario: Average Class Lag vs. Number of FORWARD Connections

higher bandwidth consumption by the richest classes, and show a significant reduction of chunk losses, which almost disappear with eight FORWARD connections. The effects of the improved bandwidth allocation are evident on the amount of "unstable" peers in the system, that either periodically reset their lag target or get stuck in initialization phase (Table I).

C. But What's Going On? Capturing Class Relationships

To better understand the behavior of the PULSE system, it is necessary to take into account its dynamic nature: the associations between nodes are quite volatile, determined at the same time by shared interest in a particular stream data range, by the knowledge of current local chunk availability, and by past exchange performance. Rather than studying the system from the point of view of the single node and its relationships to its neighbors, we will instead concentrate our attention on the global relationships between bandwidth classes. This will allow us to draw a statistical picture of the effects of our two peer selection mechanisms and to get insights into the evolution of the system over time.

We will introduce two metrics to describe the likelihood for a node from one class to choose an exchange partner from any other class. The first one, *Class Affinity*, depends on choices made while selecting MISSING partners, while the second, *Class Friendliness*, describes FORWARD peer selection.

1) Class Affinity: The choice of nodes for MISSING data exchanges is critical to the good performance of the system, as MISSING partnerships convey the largest share of stream data to most resourceful nodes. It is thus very important to select *resourceful nodes that have a common data interest range*, to maximize the reciprocal benefit that both peers can obtain from the relationship. We thus define the concept of "Class Affinity" between classes α and β as:

$$CA(\alpha,\beta)(t) = \frac{\sum_{n\in\alpha} \|n\text{'s MISSING links to } \beta\|}{\sum_{n\in\alpha} \|n\text{'s total MISSING links}\|} \quad (1)$$

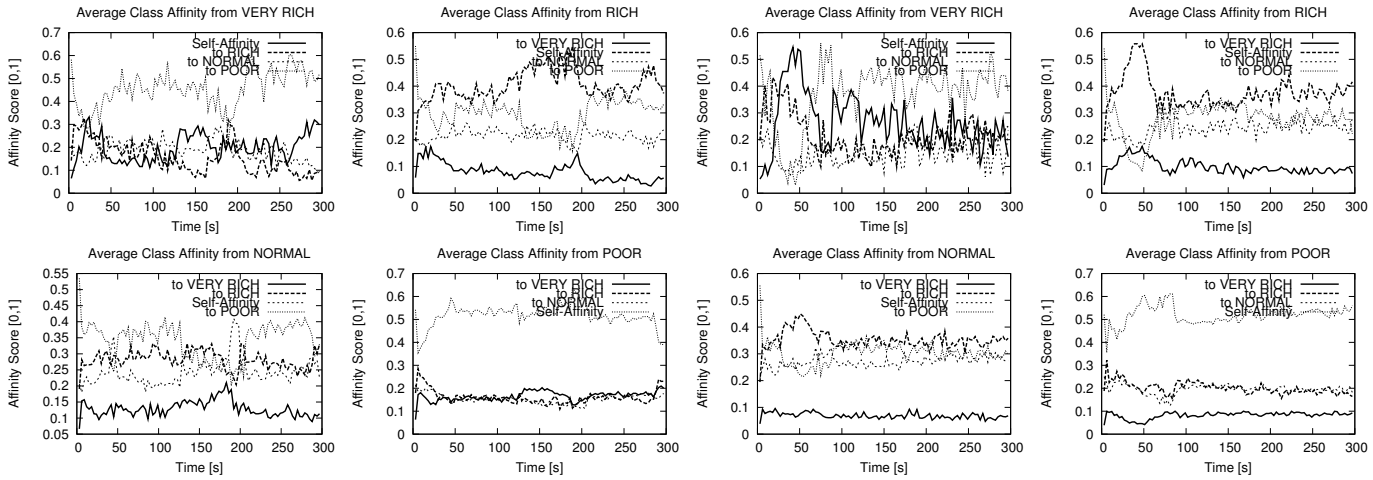
The cohesion among nodes from a same class can be captured by its *Self-Affinity* value $CA(\alpha,\alpha)$. Intuitively, its value will be higher when nodes from one class have many connections to nodes of the same class, which indicates a strong generalized reliance of a certain node class on neighbors with similar bandwidth capacity. Affinities toward other classes, richer or poorer, are also interesting to analyze to understand how nodes from different classes manage to cooperate and obtain chunks under different bandwidth scenarios.

2) Class Friendliness: Likewise, we define the concept of "Class Friendliness" as:

$$CF(\alpha,\beta)(t) = \frac{\sum_{n\in\alpha} \|n\text{'s active FWD links to } \beta\|}{\sum_{n\in\alpha} \|n\text{'s total active FWD links}\|} \quad (2)$$

Class Friendliness is similar to Class Affinity in that it describes the likelihood of an interaction among different node classes. However, unlike Affinity, the FORWARD interactions do not directly depend on bandwidth received from the target node: other factors, which are rather difficult to control, like instantaneous lag difference and cumulative exchange history, have a greater impact on this metric.

Results: We computed Affinity and Friendliness values for the previous simulation scenarios and provide a sample of the results in Fig. 4. We start by observing the evolution over time of Class Affinity in the HH-LB scenario with a) no FORWARD connections and b) 8 FORWARD connections. The plots suggest that there is a correlation between the system's convergence status and the value of the Affinity metric: when instability is present, as in Fig. 4a, the Affinity values tend to widely fluctuate, especially when node reconnections take place. For this reason, we will articulate our analysis in two



(a) Class Affinity, no FORWARD

(b) Class Affinity, 8 FORWARD

Fig. 4. HH-LB Scenario: Class Affinity vs. Number of FORWARD Connections

parts, examining separately the behavior of the system in its transitory phase and at steady state.

Comparing Figure 4a and 4b, we see that only the Affinity values for the two richest classes show meaningful differences *during the convergence phase* (from $t = 0$ to $t = 70s$): Self-Affinity for the the VERY RICH (VR) class is roughly two times higher when FORWARD connections are present, peaking at $CA(VR, VR) = 0.55$ versus a peak value of 0.32 without FORWARD, while Self-Affinity for the RICH (R) class peaks at $CA(R, R) = 0.57$ compared to 0.41 without FORWARD. We also remark that richer classes show lower Affinity scores to poorer ones when the system converges if FORWARD connections are enabled: the Affinity of Very Rich and Rich classes toward the POOR (P) class is much lower and reaches a minimum at about $t = 50s$, where $CA(VR, P) = 0.02$ (against a minimum CA value of about 0.31 without FORWARD connections) and $CA(R, P) = 0.09$ (against a minimum of 0.23).

On the other hand, *at steady state* (after $t = 70s$) we can notice that the differences in Affinity values are globally less meaningful. The only exception is the Affinity value between Poor and Very Rich peers $CA(P, VR)$, which is halved in presence of FORWARD connections (from 0.19 to 0.10), probably meaning that Poor nodes obtain less chunks directly from Very Rich nodes and thus tend to reciprocate less often. Relationships between the other classes do not seem to be affected by the presence of FORWARD connections: MISSING connections are established by richer classes toward poorer ones roughly with the same probability in both scenarios.

The fact that Self-Affinity for the richest class is initially much higher when FORWARD connections are allowed is a side effect of the interaction between altruism and tit-for-tat selection, which improves the relationships among peers with extra resources. In fact, data contributed over the FORWARD connections is also taken into account for the tit-for-tat selection at the receiver: this increases the likelihood that the receiver will choose to reciprocate and establish a MISSING connection on the following EPOCH. As richer peers have more spare resources, they can rapidly gain more MISSING relationships. Friendliness results (not shown) give additional insights: in both scenarios, the Self-Friendliness value for each class is the highest, meaning that nodes with a similar level of contribution tend to help each other out by establishing FORWARD connections. The fact that $CA(P, VR)$ is lower when FORWARD connections are enabled is a further hint that the altruistic mechanism produces a stricter organization of the nodes by decreasing upload availability. We can thus conclude that the FORWARD exchanges do no interfere negatively on the outcomes of the tit-for-tat selection, but rather help the richest nodes to exploit their bandwidth potential.

Understanding System Stability: We can now get back to Figure 3 to understand the effect of additional FORWARD connections on system stability. The fundamental consequence of tit-for-tat during the convergence phase is to foster cooperation among nodes that can contribute data. Rich nodes with recent data will spread the data to many neighbors, often gaining MISSING connections in return. At the end of the convergence process, the system under a pure tit-for-tat selection can be described as strongly polarized: rich nodes tend to select other rich nodes, while the poor mostly associate among themselves.

Massive disconnections, as seen in Figure 3a, can then be explained by the policy used at the source to distribute new chunks to the system. We notice how these disconnections mostly happen when the entire node population is concentrated in a small lag range: in this situation, as receivers are randomly selected by the source among the nodes with low average lag, all nodes in the system are equally likely to receive chunks from the source. When too many chunks are sent to poor peers, rich peers begin to starve: since the poor do not replicate chunks fast enough, the rich cannot get enough useful chunks to fill their sliding window. At the same time, rich nodes start to be abandoned by their rich neighbors, whose sliding window is also blocked. Meanwhile, the poor nodes that received the recent chunks, once the rich nodes fall behind, cannot manage

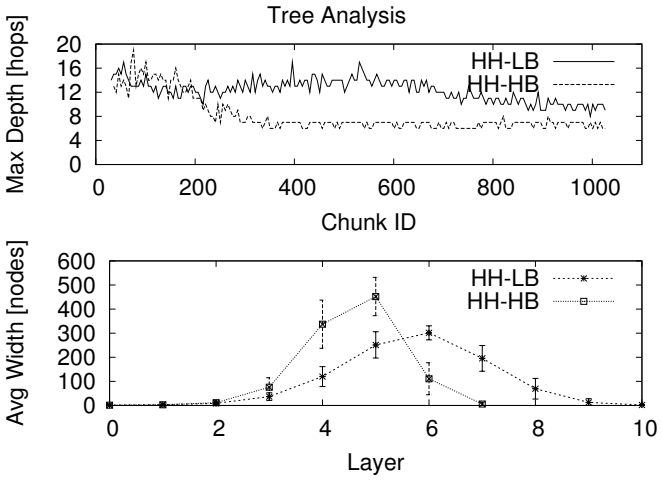


Fig. 5. Comparison of Maximum Depths and Average Widths of Distribution Trees (1000 nodes, 8 FORWARD)

to keep up with the data reception (because their upload alone is not sufficient to sustain them) and also begin to starve, in a chain reaction. Recovering from this situation requires that rich nodes reset their buffer and begin to receive fresh data again, getting back to their previous position.

Amazingly, massive disconnections can happen even if the total upload bandwidth is sufficient for a complete stream replication: the spare system capacity is left unused because of the *content bottleneck* caused by poor peers, i.e. the reduction of bandwidth utilization due to the absence of useful chunks at the rich nodes. In general, more data connections allow nodes to find missing chunks faster, as the average network diameter becomes smaller. We believe that breaking the strict tit-for-tat incentive by establishing "less optimized" connections allows rich nodes to associate more often to the poorer nodes: as we saw in Figures 3b and 3c, FORWARD connections have indeed the ability to effectively reduce the lag difference and performance skew between the rich and the poor.

D. Class Relationships and Distribution Trees

We will now analyze how differences in overall bandwidth availability and in its distribution among nodes impact the performance of the PULSE system. We will use other indirect metrics to evaluate the consequences of the node association process, namely the maximum length in hops and the average node degree of the distribution trees relative to each single chunk. These metrics express the characteristics of the paths that chunks traverse inside the system to reach all the nodes.

Even if PULSE is a mesh-based system, each data chunk follows a tree-shaped path on the overlay mesh. These trees will differ from chunk to chunk, depending on the current shape of the overlay: as the overlay connections are continuously renegotiated by each node in an independent way, one could expect that the properties of the different trees will vary a lot across different chunks. Actually, this is not the case: Fig. 5a shows the maximum tree depth for the first 1000 chunks. It can be easily noted that subsequent trees have similar depth, and - more importantly - that tree depths tend to decrease over time, settling around an asymptotic minimum value.

We can indeed explain this observation with the aggregation process among resourceful nodes, as described in the previous pages. Through subsequent iterations of the MISSING peer selection, nodes with excess bandwidth associate more often and manage to get more data than poorer nodes. The presence of FORWARD connections speeds up the discovery process and increases the amount of data that rich nodes will exchange. As a consequence:

1) richer nodes tend to attain a lower lag value,
2) rich nodes tend to further associate with rich nodes,
3) as the stream source tends to give chunks to nodes with lower lag values, the rich nodes will often end up closer to the root of subsequent chunk distribution trees.

As a last consequence, we expect that rich nodes close to the root will give origin to larger, shorter trees. The traces in Figure 5 compare the properties of distribution trees under our two bandwidth distribution scenarios. We observe that, according to our expectations, the trees obtained for the same population size with the HH-HB scenario do converge faster, are usually shorter, and have top layers that are wider on average than those from the HH-LB scenario. We argue that this result can be ascribed to the incentive-based peer selection.

E. How Fair is Fair? Introducing a 'Soft' Fairness Metric

Finally, we introduce one last metric to characterize the internal system organization. We believe that relevant information can be found looking at the correlation between the lag experienced by nodes and their bandwidth resources. As we focus our attention on the average class performance, we are especially interested in the relative lag performance between nodes from different bandwidth classes. Intuitively, we will say that it is *fair* that a node with a higher bandwidth contribution is rewarded with a lower average lag. This concept is different from the usual idea of fairness, in the sense that it does not consist in "equality of resource contribution among nodes".

We will thus call *Soft Fairness* the property of a PULSE system where nodes from a richer bandwidth class obtain, on average, a lower node lag than nodes from a poorer class. Our definition of Soft Fairness between two classes α and β, whose upload bandwidth (UB) is $UB(\alpha) > UB(\beta)$, will be:

$$SF(\alpha,\beta)(t) = \frac{\sum_{n\in\alpha}\sum_{m\in\beta} I(\,T_B(n)(t) \le T_B(m)(t)\,)}{\|\alpha\|\,\|\beta\|} \tag{3}$$

where the indicator function $I(x)$ is defined as $I(x) = \begin{cases} 1 & \text{if } x \text{ is true} \\ 0 & \text{otherwise} \end{cases}$ and $T_B(n)(t)$ is the lag of node n measured at time t.

To include the relationships between classes with higher-to-lower UB, we will extend the Soft Fairness metric to support the evaluation of the "reverse" fairness:

$$SF'(\alpha,\beta)(t) = \begin{cases} SF(\alpha,\beta)(t) & \text{if } UB(\alpha) > UB(\beta) \\ 1 - SF(\alpha,\beta)(t) & \text{otherwise} \end{cases} \tag{4}$$

Soft Fairness takes values between zero and one, with higher SF values indicating a higher degree of soft fairness. In this case, nodes that contribute more to the system get a steadier incoming bandwidth than less-contributing ones, allowing them to settle on a lower lag value than poorer classes. On the other hand, values of SF near zero indicate that the system is *unfair*, since those who contribute less can systematically get in return the needed data chunks, resulting in a better lag performance. Intermediate values could suggest that there is no strong correlation between the upload bandwidth provided by a node class and the lag it manages to obtain.

We wish to point out that the purpose of this metric is only to evaluate the correlation between node contribution and its position in the system. Soft Fairness, unlike other quantitative metrics, is not a desirable property of a PULSE system. Systems with a high value of Soft Fairness are not guaranteed to show a better performance in terms of node lag, which mainly depends on the total bandwidth availability.

In Table II we compare the Soft Fairness results from the HH-LB scenario, sampled at steady state every 3 seconds and averaged over the 90 seconds, with those from an HH-HB simulation, with and without FORWARD connections. The differences in the Soft Fairness values between the two scenarios when FORWARD connections are allowed are quite impressive: we see that, under global bandwidth excess, the Soft Fairness of the two poorer classes is low with respect to all other classes. This means that, more often than not, NORMAL and POOR nodes obtain a slightly lower lag than their richer counterparts. We argue that, when resources abound, the tit-for-tat incentive mechanism becomes less relevant and is preempted by the altruism present in the peer selection algorithms.

To further investigate the issue, we turn to the results obtained when FORWARD connections are disabled. We notice that there is a much smaller deviation in Soft Fairness values between the two scenarios. We believe that this is due to the lack of FORWARD connections, the primary altruistic mechanism. Also, some degree of unfairness is present in both cases, for all classes except VERY RICH, probably as a consequence of the altruistic discovery mechanism used by MISSING peer selection. Finally, we can appreciate how the presence of FORWARD connections enhances the clustering effect of the tit-for-tat incentive, especially when the total available bandwidth is scarce (Table II, values in bold).

F. Response to Perturbations

We performed additional experiments of the PULSE system under various different arrival and departure patterns. We experimented with flash-crowd, uniform, and spike node arrivals, and with exponential and burst node disconnections. In general, we found that the effects of node transience on the behavior of PULSE are usually minor, sometimes barely noticeable, and their extent depends mostly on the availability of excess bandwidth. We also observe that the evaluation of bursty node arrivals suffers a bit from the simplified, oracle-like knowledge model used by the simulator.

In Fig. 6 we see three sample lag traces of the reaction of the system to node transience. In these plots, nodes can establish up to eight FORWARD connections. We first present the effect of an *instantaneous spike* arrival on our usual bandwidth scenarios. With this arrival pattern, 750 nodes join the network at $t = 120s$, when the initial 250 nodes should have reached steady state. In Fig. 6a, we show an HH-LB scenario absorbing a spike of arrivals: it can be noticed that most nodes from all the classes (including the richest ones) are affected and forced to reconnect, but the perturbation lasts for a very short time. By $t = 150s$, in fact, all the classes have reached again convergence. If we increase the available bandwidth, however, the impact of the spike is much reduced. This is the case of Fig. 6b, where the HH-HB scenario is shown. Here we can notice that all classes temporarily increase their average lag, but there are no disconnections. The lag increases up to 80 chunks on average but is absorbed very rapidly (in about ten seconds). In this case, the nodes' media play-out is not affected.

Figure 6c shows the effect on a HH-LB scenario of the simultaneous departure at $t = 120s$ of 50% of the nodes, chosen at random. We see that the impact of node departures on the system performance is even lower, as long as there is enough serving capacity in the system. Results under exponential departures after $t = 120s$, which we will not present here, did also show a strong resilience of the system against widespread and subsequent node departures.

V. EXPERIMENTAL RESULTS

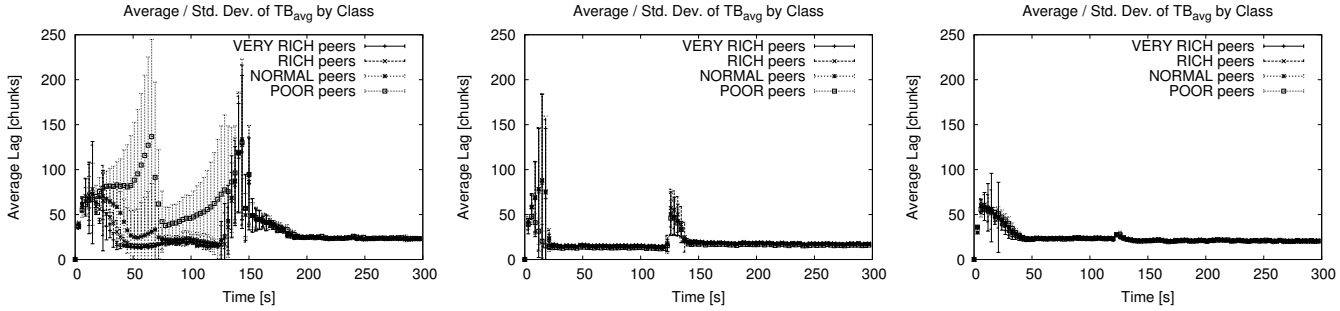
We implemented a full prototype of the PULSE node and tested its behavior on the French Grid'5000 [22] testbed, which offers similar functionalities as Emulab [21]. The features provided by Grid'5000, namely a closed network environment where hosts can be reserved, give us a relatively controlled environment for testing our application. The high-speed links of Grid'5000 assure the absence of bandwidth bottlenecks between nodes and low latencies.

Our experiments are performed by artificially limiting the outbound bandwidth used by the application. A first experiment was performed to validate a simple symmetric scenario (where all peers have 2*SBR upload bandwidth) in order to make sure that the system behaved consistently in case of excess bandwidth. We have then reproduced the two heterogeneous scenarios described in Section IV, in order to compare the results with those from the simulator. The number of peers used for this experimental evaluation is 800.

In all these experiments, at the beginning the peers join sequentially with arrival intervals of around half a second. At the end of the experiments, all the peers leave at the same time. The SBR is set to 16 chunks/s with a chunk size of 4 KB. The size of the sliding window is again 32 chunks, while the trading window size is 64 chunks wide. Every peer can select 4 other peers for MISSING exchanges and 8 other peers for FORWARD exchanges. The PULSE prototype uses UDP for the control messages and TCP for data exchanges. Control messages are issued to neighbors at a minimum fixed rate, which becomes self-clocking with data exchange and cannot exceed on any connection the chunk rate. Request/response messages use a bitmap of the buffer to encode the requests, so that more than one chunk can be requested with a single

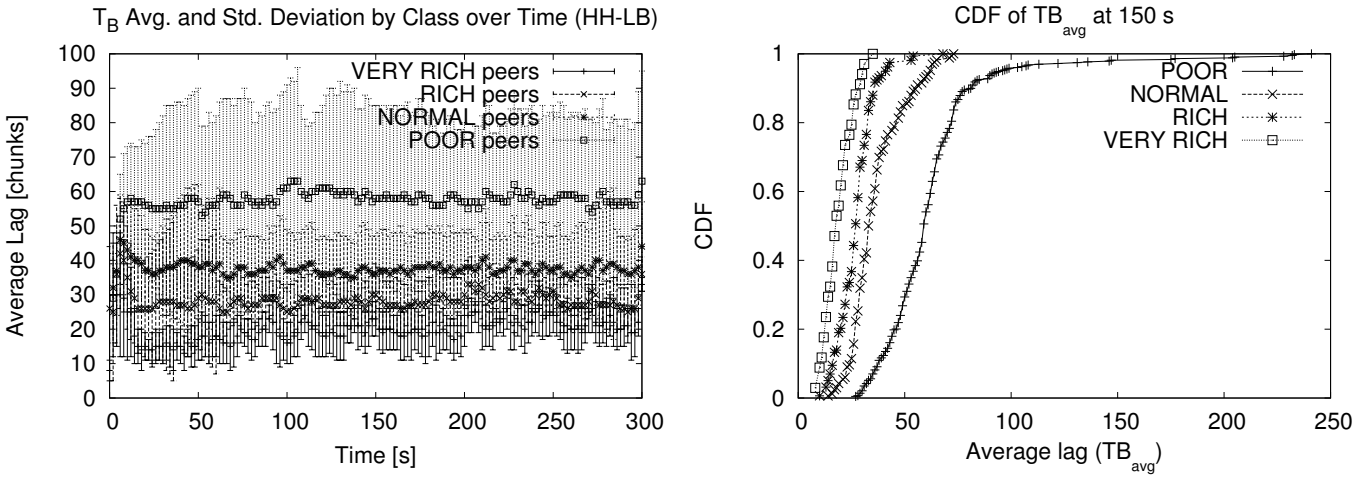
Soft Fairness between Different Bandwidth Classes (rows to columns)																	
	HH-LB, 8 FWD				HH-HB, 8 FWD					HH-LB, NO FWD				HH-HB, NO FWD			
	VR	R	N	P	VR	R	N	P		VR	R	N	P	VR	R	N	P	
VR	=	**0.81**	**0.81**	**0.80**	=	0.75	0.66	0.36		=	**0.82**	**0.76**	**0.64**	=	0.86	0.67	0.53	VR
R	0.63	=	0.65	0.71	0.55	=	0.51	0.23		0.76	=	0.42	0.41	0.81	=	0.25	0.35	R
N	0.69	0.53	=	0.65	0.46	0.30	=	0.30		0.68	0.34	=	0.48	0.56	0.20	=	0.45	N
P	**0.74**	**0.64**	**0.58**	=	0.21	0.12	0.16	=		**0.55**	**0.32**	**0.38**	=	0.46	0.34	0.38	=	P

TABLE II
SOFT FAIRNESS: COMPARING HH-LB AND HH-HB SCENARIOS (8 AND NO FORWARD)



(a) Spike arrivals (t=120s), HH-LB Scenario (b) Spike arrivals (t=120s), HH-HB Scenario (c) Burst departures (t=120s), HH-LB Scenario

Fig. 6. Effects of Node Transience on Global Lag Performances (8 FORWARD)



(a) HH-LB Scenario: Avg. Class Lag vs. Time (b) HH-LB Scenario: Class Lag CDF at t=150s

Fig. 7. Results for the Node Lag from Experiments on Grid'5000 (HH-LB)

message. The limit on outstanding requests between two nodes is currently set to 2, and request timeouts are set to 0.5s.

Figure 7a shows the evolution of the average node lag over time in the HH-LB scenario. We can notice that peers from all the classes are able to reach very quickly a constant average lag, which remains largely stable afterward. Peers belonging to the VERY RICH class experiment the lowest average lag value, followed by peers belonging to the RICH class, to the NORMAL class and, finally, by the peers from the POOR class. These results validate the simulation and confirm the formation of clusters among peers with the same resource availability. The variance of the average lag value indicates a higher instability of the poorest peers compared to the other classes. In the above experiment, 104 buffer reset events were observed during the five minutes: 100 involved POOR peers, 2 NORMAL peers, and 2 RICH peers.

Figure 7b displays a snapshot at steady state of the cumulative distribution function (CDF) of average node lag by class in the HH-LB scenario. We can see that, while all the nodes that contribute at least as much as they receive obtain

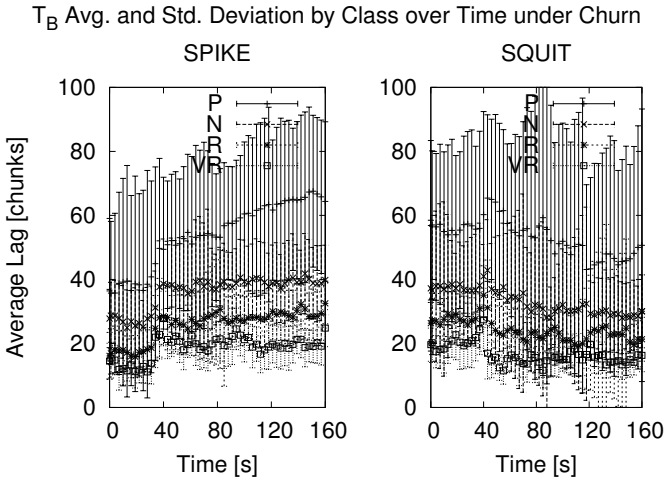


Fig. 8. Response to Massive Node Churn on Grid'5000 (HH-LB)

a lag lower than 50 chunks (about 12 seconds), more than 90% of the P nodes manage to obtain a stable streaming rate with a slightly higher lag. Only a small fraction of the P class is actually suffering shortage, with less than 5% of P nodes that risk playback disruption because of an upcoming buffer reset. Examining the data distribution trees for this scenario, we could observe that the maximum tree depth is on average 12 (it fluctuates between 11 and 14 hops) and that trees are usually fairly large near the source, with a rapid growth of layer population in the first few layers (3, 11, 27, 53, 83, 104 nodes on average) and a sharp drop after the 10th layer.

Finally, we tested the response to churn of the emulated PULSE system. In Figure 8 we apply the same membership patterns we used in our simulations to an emulated LH-LB scenario, with the churn event scheduled at $t = 40s$. We can observe how, in the instantaneous arrival scenario (Figure 8, SPIKE), all the resource-rich classes do not suffer any instability upon the arrival of 600 peers, but quickly adapt to the new situation and settle on a slightly higher lag value. The POOR class needs a longer time to find a new stable configuration, but its lag fluctuates gracefully. On the other hand, it is fairly hard to notice the immediate effect on the system of the sudden departure of 400 peers (Figure 8, SQUIT). The only visual cue is a small increase in the lag of the POOR class, which lasts for less than ten seconds: afterward, we can see a slow decrease in the average lag for every class, as the system converges to a new steady state with a halved node population.

We have been conducting experiments also on PlanetLab [23] in order to evaluate the behavior of the PULSE prototype in a less controlled network environment. In fact, on PlanetLab, latencies are on average higher and more widely spread compared to Grid'5000, while link bandwidths are lower. Moreover, the resource availability of PlanetLab hosts is very unpredictable, and their CPU load is often so high that it becomes hard to obtain meaningful traces (the node software execution is slowed down and the peers become unresponsive). Despite this difficult environment, our preliminary results so far largely confirm the observations performed in testbed conditions. We leave a thorough evaluation of PULSE on the PlanetLab environment as future work.

A. Remarks on the Results

It is interesting to compare the results obtained on the testbed to the simulated ones: in general, experimental results show higher average lag values and bigger variances than the simulations. This fact can be explained by the differences in the control information exchange between the real and the simulated system: over a real network, control information is spread via messages, which suffer from link delay and can be lost: as peers depend on these messages for making their decisions, they are less responsive and efficient than in the simulations. Apart from these differences, which were indeed expected, we want to emphasize that the properties of the PULSE algorithms match reasonably well across simulated and experimental environments.

We have observed in our extensive simulations that the PULSE system scales well to fairly large user populations: leaving the value of all the parameters unchanged, we managed to simulate systems with populations of up to ten thousand nodes for the different scenarios. The results show a system-wide average lag that grows logarithmically with the population size and whose slope depends on the amount and distribution of bandwidth resources. These observations corroborate our results on the properties of the PULSE mesh and of the chunk distribution trees and also suggest that PULSE may scale upward as well as tree-based systems, in terms of node lag and path length.

VI. Related Work

Several recent papers describe peer-to-peer networks that support live streaming. They can be categorized into *structured* or *unstructured*, referring to the way control information and knowledge are propagated, and *mesh-based* or *tree-based*, based on the way data are exchanged by the nodes. Table III summarizes the design choices adopted by the most relevant systems that have been proposed to date.

The first tree-based approaches to p2p streaming derive from early research on application-level overlays [5][6]. The main motivation was to provide the benefits of an easily deployable multicast infrastructure when no native support for IP multicast was available: to this end, an unique overlay tree was used to convey data. NICE [5] organized the nodes in a hierarchical, cluster-based single tree overlay, trying to optimize the average delay through appropriate node management policies and cluster-head selection criteria. ZIGZAG [6] improves on this design by adding redundancy to the cluster management mechanisms to better cope with node churn. End System Multicast [4] is the first large-scale video distribution system based on a single-tree multicast infrastructure to have been deployed and for which measurements have been collected. The fundamental shortcoming of all tree-based systems is due to the limitations imposed by the tree structure. Single trees artificially limit the available service capacity, as the leaf nodes, which constitute a preponderant share of the population, are prevented from contributing bandwidth to the system. Moreover, each internal

Control Plan & Knowledge Mgmt.	Data Plan	
	Tree-Based	Mesh-Based
Implicit	NICE, ZIGZAG, ESM	=
Structured	Splitstream (DHT)	Bullet (Tree)
Unstructured	Chunkyspread	DONet, Chainsaw, PULSE

TABLE III
SUMMARY OF MAIN APPROACHES TO LIVE STREAMING

node is a possible bandwidth bottleneck for the sub-tree it serves, and packet losses do *accumulate* while descending the tree. Finally, the maintenance, optimization, and recovery of failed overlay tree links can become a daunting task under heavy churn. Data loss during the tree repair process can be partially masked by buffering and recovery mechanisms, but eventually it will severely disrupt the play-out quality.

Multiple-tree based overlays have been proposed as a solution to the above problems. By encoding the stream as independent *MDC stripes* [12] and streaming them over several trees, these systems overcome the most important among the above limitations. Splitstream [3], for instance, does create multiple interior-node-disjoint trees. Every peer is an interior node in at most one tree, thus mitigating the bottleneck data loss problem (a single failure results in the interruption of at most one stripe, which is not critical and can be masked by the encoding method), and the capacity limiting problem (each node is likely to be an interior node in at least one tree). On the other hand, the control overhead is clearly higher than in the single-tree case: in general, multiple-tree systems often rely on an underlying DHT substrate for tree-building and maintenance purposes. The fact of combining two structured systems with different goals and purposes – low-latency and high-bandwidth data distribution versus efficient lookup and resilient connectivity – has raised unexpected issues: Splitstream has been shown [9] to suffer from churn and bandwidth heterogeneity, as it tends to create non-DHT links that increase the system complexity. Moreover, the tree-building criterion used by Splitstream prevents the exploitation of node heterogeneity and the use of incentive-based neighbor selection. Chunkyspread [10], a recent multiple-tree based unstructured system, removes the requirement for a DHT substrate by adopting a non-hierarchical approach to tree-building. Using a gossip protocol, nodes exchange the list of the data stripes they currently receive along with a compact Bloom filter representation of the list of their ancestors for each stripe. Bloom filters are used to constrain peer selection and assure that the resulting stripe distribution paths will be free of loops. Peer selection is based on the *load* advertised by the neighbors and on their *latency* in receiving specific stripes.

Mesh-based designs aim to further reduce the structural constraints in live media streaming systems to improve their resilience against churn and node transience. In this case, the stream is always broken up in a series of chunks, which are distributed by the source to few nodes in the system. Nodes must exchange chunks to retrieve a sufficiently complete stream before the play-out deadline. Bullet [14] is an early approach that combines a single-tree and a mesh: the tree is used to convey both data chunks and control information, while the mesh is created independently by the nodes based on the control information. An advantage of this scheme is that the control protocol running on the tree can have a very low overhead. On the other hand, no mechanism to encourage bandwidth contribution has been implemented in the system. Chainsaw [7] is a proof-of-concept example of a simple mesh-only system, which uses randomized peer- and chunk-selection algorithms. However, it has not been tested on scenarios with asymmetrical bandwidth availability. Coolstreaming/DONet [8] introduced a better chunk scheduling algorithm that takes into account the individual chunk deadlines for play-out. While each node performs a periodical long-term optimization of the mesh by replacing the least contributing neighbors, the system is not meant to address the combined effects of upload bandwidth heterogeneity under global shortage and high churn rates. User experiences with the Coolstreaming application seem to indicate that it suffers from a high play-out latency, probably due to conservative data buffering. PULSE does improve with respect to these systems as its dynamically optimized mesh supports high bandwidth heterogeneity and churn, while its buffering requirements are typically low even under a moderate shortage of global serving capacity.

It is challenging to perform a comparison of PULSE against other systems in the literature, as different architectures have been evaluated using purpose-built metrics and often rely on incompatible assumptions. Even if we restrain the comparison to unstructured mesh-based systems only, we have to acknowledge a widespread lack of published evaluation results[6]. We believe that a thorough comparison of the performance of different live streaming systems is a very interesting subject by itself, which will be a fundamental part of our future work.

VII. CONCLUSIONS

We have presented and evaluated PULSE, an unstructured, mesh-based peer-to-peer system for the distribution of live media. We showed that the tit-for-tat peer selection policy can be successfully adapted to the requirements of live streaming: PULSE uses an incentive-driven feedback loop, based upon bandwidth measurements of the application data and subsequent evaluations of node lag differences that is a robust and lightweight mechanism to estimate the service capacity of neighboring peers and effectively guides future node associations. PULSE demonstrates that a mesh-based architecture is a perfectly viable approach to live streaming that can flexibly

[6]The only relevant term of comparison is a mention in [8] about an average chunk hop count of about 7 for a Coolstreaming system with 200 nodes.

accommodate heterogeneous populations under high churn, while still providing low play-out latency.

Moreover, we studied the effects of the local incentive policy on the global system behavior through simulations and experiments, and defined novel metrics - such as affinity, friendliness, and soft fairness - to evaluate the behavior of PULSE. We showed that, when resources are globally scarce, incentives allow resource-rich peers to retrieve the complete stream with low delay, while the least-contributing peers may suffer starvation. The tit-for-tat selection favors synchronization between peers with similar upload and rewards contributors offering an upload capacity greater or equal than the stream rate. Conversely, lesser contributors may obtain discontinuous service and eventually be forced to reconnect. When the available resources exceed the demand, on the other hand, the altruistic allocation mechanism tends to prevail on tit-for-tat, but still produces systems with short distribution trees. Node performances become then almost homogeneous, regardless of individual bandwidth contribution. The presence of altruism guarantees in all cases an efficient utilization of resources at the richest nodes.

As far as we know, PULSE is the only live streaming system that currently implements pairwise incentives as the short-term peer selection policy. As a part of the future work, we plan to perform additional experiments to study the impact of network latency. Also, we expect to widely deploy an optimized version of our node software, to evaluate the response of PULSE to realistic applicative workloads and typical usage patterns.

Finally, we believe that "local pairwise incentives based on measurements" could become a new paradigm to address problems of uneven resource distribution in generic distributed systems. To apply this paradigm in other contexts, however, we argue that four base assumptions should still hold in the target application: 1) most nodes should be free to associate (low structural constraints), 2) the role of nodes should be symmetrical (common interest of the entire population in a specific resource), 3) node interactions should be repeated frequently (peers engaged in an iterated *prisoner dilemma*), and 4) node contributions should be implicitly measurable (quantitative feedback evaluation should be possible). Finding appropriate applications, or adapting them to fit into the above guidelines, will surely be an interesting challenge.

REFERENCES

[1] K. Sripanidkulchai, A. Ganjam, B. Maggs, and H. Zhang, "The Feasibility of Supporting Large-Scale Live Streaming Applications with Dynamic Application End-Points", in *Proc. of ACM SIGCOMM '04*

[2] M. Castro, M. Costa, and A. Rowstron, "Peer-to-peer overlays: structured, unstructured, or both?", *Tech. Report MSR-TR-2004-73*, Microsoft Research, Cambridge, UK, July 2004

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

[4] Y. Chu, A. Ganjam, T. S. E. Ng, S. Rao, K. Sripanidkulchai, J. Zhan, and H. Zhang, " Early Experience with an Internet Broadcast System Based on Overlay Multicast", in *Proc. of USENIX '04*, June 2004

[5] S. Banerjee, B. Bhattacharjee, and C. Kommareddy, "Scalable application layer multicast" in *Proc. of ACM SIGCOMM '02*

[6] D. A. Tran, K. A. Hua, and T. T. Do, "A Peer-to-Peer Architecture for Media Streaming", in *IEEE JSAC*, vol. 22, no. 1, January 2004

[7] V. Pai, K. Kumar, K. Tamilmani, V. Sambamurthy, and A. E. Mohr, "Chainsaw: Eliminating Trees from Overlay Multicast", in *Proc. of the 4th International Workshop on Peer-to-Peer Systems*, February 2005

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

[9] A. R. Bharambe, S. G. Rao, V. N. Padmanabhan, S. Seshan, and H. Zhang, "The Impact of Heterogeneous Bandwidth Constraints on DHT-Based Multicast Protocols", in *Proc. of the 4th International Workshop on Peer-to-Peer Systems*, February 2005

[10] V. Venkataraman, K. Yoshida, P. Francis, "Chunkyspread: Heterogeneous Unstructured End System Multicast", in *Proc. of the 14th IEEE ICNP*, November 2006

[11] T. Nguyen, and A. Zakhor, "Distributed Video Streaming with Forward Error Correction", in *Proc. of Packet Video Workshop*, April 2002

[12] V. K. Goyal, "Multiple description coding: Compression meets the network", in *IEEE Signal Process. Mag*., September 2001

[13] Y.-H. Chu and H. Zhang, "Considering Altruism in Peer-to-Peer Internet Streaming Broadcast", in *Proc. of 14th IEEE NOSSDAV*, June 2004

[14] D. Kostic, A. Rodriguez, J. Albrecht, and A. Vahdat, "Bullet: High Bandwidth Data Dissemination Using an Overlay Mesh", in *ACM SOSP* '03, October 2003

[15] S. Saroiu, P. K. Gummadi, and S. D. Gribble, "A Measurement Study of Peer-to-Peer File Sharing Systems", in *Proc. of Multimedia Computing and Networking*, 2002

[16] A. J. Ganesh, A.-M. Kermarrec, and L. Massoulié, "Peer-to-peer membership management for gossip-based protocols", in *IEEE Trans. Comput*., Vol. 52, No. 2, February 2003

[17] B. Cohen, "Incentives Build Robustness in BitTorrent", in *Proc. of Workshop on the Economics of P2P Systems*, 2003

[18] C. Grothoff, "An Excess-Based Economic Model for Resource Allocation in Peer-to-Peer Networks", in *Wirtschaftsinformatik*, June 2003

[19] A. Legout, G. Urvoy-Keller, and P. Michiardi, "Understanding BitTorrent: An Experimental Perspective", *Tech. Report inria-00000156*, INRIA, Sophia Antipolis, November 2005

[20] A. Legout, G. Urvoy-Keller, and P. Michiardi, "Rarest First and Choke Algorithms Are Enough", in *Proc. of IMC '06*, October 2006

[21] B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb, and A. Joglekar, "An Integrated Experimental Environment for Distributed Systems and Networks", in *Proc. of OSDI '02*, December 2002

[22] Grid'5000 - Grid platform testbed - http://www.grid5000.org [WWW]

[23] PlanetLab - http://www.planet-lab.org/ [WWW]

Fabio Pianese received his B.S. degree in Electronic Engineering from Politecnico di Torino, Turin, Italy, and his M.S. degree in Computer Science from Université de Nice - Sophia Antipolis (UNSA), France. He also obtained a degree in Telecommunications from the Institut Eurecom. He has been working toward his Ph.D. degree in Computer Science at UNSA since November 2004. His research focuses on peer-to-peer systems and applications. He is funded and supported in his research by France Telecom R&D - Orange Labs.

Diego Perino graduated in Networking Engineering at Politecnico di Torino, Turin, Italy, and at the Institut Eurecom, Sophia-Antipolis, France, in September 2006. He also received a Master degree from Université de Nice-Sophia Antipolis (France) in September 2006. Since November 2006 he is a Ph.D. student at Orange Labs Paris (France). His current research interests are peer-to-peer systems, interaction between overlays and underlay, and (in)validation of theoretical models of Internet protocols.

Joaquín Keller obtained a Master degree in Mathematical Logic from Jussieu University (Paris VII), France, and a Ph.D. on Distributed Systems from University of Versailles, France. He is a senior researcher at France Telecom R&D in Paris and his current interests include peer-to-peer networks, distributed algorithms, and multimedia communication. He is the designer of Solipsis, a serverless virtual world, and Maay, a personalized distributed search system.

Ernst W. Biersack received his M.S. and Ph.D. degrees in Computer Science from the Technische Universität München, Munich, Germany and his habilitation from the University of Nice, France. Since March 1992 he has been a Professor in Telecommunications at Institut Eurecom, in Sophia Antipolis, France. His current research is on Peer-to-Peer Systems, Network Tomography, and LAS Scheduling in Edge Routers. Among other awards, he has received (together with J. Nonnenmacher and D. Towsley) the 1999 W. R. Bennet Award of the IEEE for the best paper published in the year of 1998 in the ACM/IEEE Transactions on Networking.