POLITECNICO DI TORINO
Repository ISTITUZIONALE

Experimental comparison of neighborhood filtering strategies in unstructured P2P-TV systems

(Article begins on next page)

19 April 2024

Experimental comparison of neighborhood filtering strategies in unstructured P2P-TV systems

S. Traverso[a], L. Abeni[b], R. Birke[c], C. Kiraly[b], E. Leonardi[a], R. Lo Cigno[b], M. Mellia[a]

[a] DELEN, Politecnico di Torino, Italy – {lastname}@tlc.polito.it

[b] DISI, Univesity of Trento, Italy – {lastname}@disi.unitn.it

[c] IBM Research Lab. Zurich, CH – bir@zurich.ibm.com

Abstract—**P2P-TV systems performance are driven by the overlay topology that peers form. Several proposals have been made in the past to optimize it, yet little experimental studies have corroborated results. The aim of this work is to provide a comprehensive experimental comparison of different strategies for the construction and maintenance of the overlay topology in P2P-TV systems. To this goal, we have implemented different fully-distributed strategies in a P2P-TV application, called Peer-Streamer, that we use to run extensive experimental campaigns in a completely controlled set-up which involves thousands of peers, spanning very different networking scenarios. Results show that the topological properties of the overlay have a deep impact on both user quality of experience and network load. Strategies based solely on random peer selection are greatly outperformed by smart, yet simple strategies that can be implemented with negligible overhead. Even with different and complex scenarios, the neighborhood filtering strategy we devised as most performing guarantees to deliver almost all chunks to all peers with a play-out delay as low as only 6s even with system loads close to 1.0. Results are confirmed by running experiments on PlanetLab. PeerStreamer is open-source to make results reproducible and allow further research by the community.**

I. INTRODUCTION

Mesh based live P2P streaming systems (P2P-TV in short) are among the most promising solutions for inexpensive broadcast of real time video contents over the Internet. They offer content providers and broadcasters the opportunity of reaching a potentially unlimited audience without expensive infrastructural investments. Just as in file sharing P2P systems, in mesh based P2P-TV systems the video content is sliced in pieces called chunks, which are distributed onto an overlay topology exploiting a fully distributed epidemic approach. But, contrary to file sharing P2P systems, chunks are generated in real time, sequentially and (in general) periodically. They must also be received by the peers within a *deadline* to be played out, so that timely delivery is the key aspect of these systems. This makes P2P-TV systems design deeply different from file sharing applications design, and solutions proposed for file sharing P2P systems can be adapted to live P2P-TV systems only at price of large play-out delays.

Two are the key features that characterize a mesh based P2P-TV system: *i)* the algorithms adopted to build and maintain the overlay topology [1], [2], [3], [4], [5], [6], *ii)* the algorithms employed to trade chunks [7], [8]. A large body of research work has focused on the design and analysis of efficient algorithms for both the overlay topology maintenance and chunk

scheduling. Most of the previous works, however, have mainly a theoretical flavor, thus performance analysis of different proposed strategies have been carried out in rather idealized scenarios exploiting simulations or analytical models [3], [4], [5], [6]. Few works undergo implementation and present actual experiments, and even those are usually limited to few tens of peers [9], [10]. A detailed discussion of related work is presented in Sect. VIII.

Indeed, only an actual implementation allows to fully evaluate the different policies, assessing the impact of signaling, measurements, implementation issues, etc. This paper tries to fill this gap, providing a comprehensive and purely experimental comparison of different strategies for the construction and the maintenance of the overlay topology for P2P-TV systems.

The algorithms we investigate are all based on the selection of the neighbors a peer chooses, keeping the system fully distributed and without the need for external help, or a centralized 'oracle' to help peers. Algorithms are based on *selection* and *replacement* criteria, according to which each peer chooses the peers he would like to download chunks from. A simple blacklist-like hysteresis prevent peers to continuously select peers replaced due to poor performance. Overall, we explore 12 different combinations of criteria (24 if blacklisting is enabled), based on metrics such as Round Trip Time (RTT), upload capacity, number of received chunks, etc. Intuitively, these are metrics that are known to either *i)* favor traffic localization, e.g., choosing peers with smaller RTT, or *ii)* improve system performance, e.g., choosing peers with larger upload capacity [6], [7].

We test these algorithms in three network scenarios in which we control peer upload capacity, end-to-end RTT and packet loss. In the simplest scenario, peer upload capacities are heterogeneous among peers, while RTT forms 4 clusters, with intra-cluster RTT being smaller than inter-cluster RTT. Then we consider a biased upload capacity distribution, where high capacity peers are all in the same cluster. Finally, we add the impact of eventual packet loss on long-distance paths among clusters, facing an almost adversarial scenario.

Results show that simple random-based policies are outperformed by policies based on network distance coupled with policies that drop peers based on their contribution in all scenarios. The latter are experimentally proved to achieve excellent QoE even under almost adversarial network scenarios, i.e., at load close to 1.0, with heterogeneous upload bandwidth
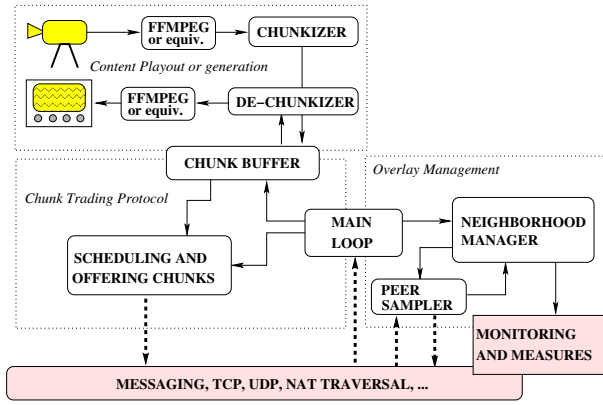
Figure 1. PeerStreamer peer architecture.

and with clustered RTT. Similar conclusions are drawn from PlanetLab experiments.

Finally, we wish to emphasize that, for the first time to the best of our knowledge, we present *reproducible experimental results* for a fully controlled and publicly available real implementation of a P2P-TV system referring to a rather large scale set-up with thousands of peers. Our results have been collected during large actual campaigns which totally amounts to more than 1000 hours of experimental tests.

The software used in this paper is released Open Source and includes all the components necessary to build a fully functional P2P-TV system including video transcoding at the source and play-out at clients.

II. PEERSTREAMER DESCRIPTION

Empowering this work is PeerStreamer[1], an Open Source P2P-TV client that stems from the developments and research of the NAPA-WINE project[2] whose overall architecture and vision are described in [11]. PeerStreamer leverages GRA-PES [12], a set of C libraries implementing building blocks for P2P-TV streaming that enables building applications with almost arbitrary characteristics, thus allowing for experimental comparison of different choices to be done efficiently. Fig. 1 describes the logic and modular organization of PeerStreamer. The overlay management, the focus of this paper, is detailed in Sect. II-B, while in the following we sketch the high level organization of the other application components.

A. PeerStreamer Architecture

PeerStreamer is based on a chunk-based stream diffusion. Peers offer a selection of the chunks they own to some peers in their neighborhood. The receiving peer acknowledges the chunks it is interested in, thus avoiding multiple transmissions of the same chunk to the same peer. The negotiation and chunk transmission phase is based on signaling exchanges with "Offer" and "Select" messages. For chunk scheduling, Offers are sent to neighbors in round-robin. They contain the buffer-map of the recent chunks the sender possesses at that time. After receiving an Offer, a peer selects one chunk based

---

on a "latest useful" policy sending back a Select message: the receiver selects the most recent chunk it does not have. This has been proven optimal for streaming systems with centralized and distributed scheduling associated to specific peer choices in [7], [13]. The *number of offers* per second a peer sends plays a key role in performance. Intuitively, it should be large enough to fully exploit the peer upload capacity, but it must not be too large to cause the accumulation of chunks to be transmitted adding queuing delay prior to chunk transmissions. We adopt Hose Rate Control (HRC) proposed in [14] to automatically adapt the number of offers to both peer upload capacity and system demand. Simpler trading schemes are less performing and can hide the impact of the overlay on the overall system performance.

The source is a standard peer, but it does not participate in the Offer/Select protocol. It simply injects copies (5 in our experiments) of the newly generated chunk into the overlay. It implements a *chunkiser* to process the media stream (e.g., an encoded file, or a live stream coming from a DVB-T card, or the video of a web-cam). The chunking strategy used in PeerStreamer is chosen to avoid mingling its effects with the topology-related ones: one-frame is encapsulated into one-chunk to avoid that a missing chunk would impair several frames due to, e.g., missing frame headers. The chunkiser is implemented using the ffmpeg libraries[3], so that several different codecs (e.g., MPEG, theora, H.264, etc.) are supported. Receiving peers, instead, implement a de-chunkiser, which reads from the local chunk buffer and pushes the chunks in the correct sequence to the play-out system.

The main loop (at the center of Fig. 1) implements the global application logic. It is responsible for the correct timing and execution of both semi-periodic tasks, e.g., sending new offers, and asynchronous activities, e.g., the arrival of a chunk or signaling message from the messaging layer.

PeerStreamer architecture is completed by the "messaging" and "monitoring and measures" modules. The messaging module is a network abstraction layer that frees the application from all details of the networking environment, e.g., the presence of NAT, middle-boxes and other communication details. It offers a connection-oriented service on top of UDP, with a lightweight retransmission mechanism that allows the recovery of lost packets without high retransmission delay.

The monitoring and measures module extracts network information by running passive and/or active measurements [11]. In this paper we rely on the measurements of *i)* end-to-end path delay between peers (e.g., RTT), *ii)* packet loss rate, and *iii)* transmission rate of a peer.

B. Overlay Management

The approach for building the overlay topology in Peer-Streamer is fully distributed: each peer builds its own neighborhood following only local measures, rules and peer sampling. The overlay topology is represented by a directed graph in which the peer at the edge head receives chunks from the

---

peer at the edge tail, which is the one sending offers. Each peer p handles thus an "in-neighborhood" $\mathcal{N}_I(p)$ and an "out-neighborhood" $\mathcal{N}_O(p)$. $\mathcal{N}_I(p)$ collects all peers that can send chunks to p (p in-neighbors); $\mathcal{N}_O(p)$ collects all peers that can receive chunks from p (p out-neighbors). Alternatively, $\mathcal{N}_I(p)$ is the set of peers that *offer* p new chunks; while p *offers* its chunks to peers in $\mathcal{N}_O(p)$. Distinguishing between $\mathcal{N}_I(p)$ and $\mathcal{N}_O(p)$ guarantees a greater flexibility in topology management than algorithms that impose the reciprocity between peers. The overlay topology $\mathcal{T}_S$ is then obtained as union of all the edges connecting peers in $\mathcal{N}_I(p)$ to p, i.e.:

$$\mathcal{T}_S = \bigcup_{p \in \mathcal{S}} \mathcal{N}_I(p) \times \{p\} \tag{1}$$

where $\mathcal{S}$ is the set of all the peers in the swarm and the symbol $\times$ denotes the Cartesian product operator[4].

Referring again to Fig. 1, the topology management is split into two separate functions. The *peer sampler* has the goal of providing p with a stochastically good sample of all the peers in $\mathcal{S}$ and their properties; PeerStreamer implements a variation of Newscast [15] for this function. The *neighborhood manager* realizes the task of filtering the peers most appropriate for interaction. Filtering is based on appropriate metrics and measures, and it is the main focus of this paper.

III. NEIGHBORHOOD AND TOPOLOGY CONSTRUCTION

In PeerStreamer every peer p selects other peers as in-neighbors and establishes a management connection with them. Thus each peer p actively selects in-neighbors to possibly download chunks when building the set $\mathcal{N}_I(p)$. Similarly, p passively accepts contacts from other peers that will form the set $\mathcal{N}_O(p)$ of out-neighbors. There is no limitation to $N_O(p)$[5].

Every peer p manages a blacklist of peers in which it can put peers that were perceived as very poorly performing in-neighbors. Peers in the blacklist cannot be selected for inclusion in $\mathcal{N}_I(p)$. Blacklisted peers are cleared after the expiration of a time-out (set to $50\,\mathrm{s}$ in the experiments).

The size N_I of $\mathcal{N}_I(p)$ is equal for every peer p: its goal is to guarantee that p has enough in-neighbors to sustain the stream download with high probability in face of churn, randomness, network fluctuations, etc. The size $N_O(p)$ of $\mathcal{N}_O(p)$ is instead a consequence of the filtering functions of the peers that select p as in-neighbor. The goal is to let the dynamic filtering functions of peers $q \in \{\mathcal{S} \setminus p\}$ select $\mathcal{N}_O(p)$ in such a way that the swarm performances are maximized. For example, peers with higher upload capacity should have larger number of out-neighbors than peers with little or no upload capacity [4].

The update of neighborhoods is periodic, maintaining the topology dynamic and variable, so that churn impairment is limited, and the swarm can adapt to evolving networking conditions. In particular, every T_{up} seconds each peer p

independently updates $\mathcal{N}_I(p)$ by *dropping* part of the old in-neighbors while *adding* fresh new in-neighborsTwo parameters are associated to this scheme: the update period T_{up} and the fraction F_{up} of peers in $\mathcal{N}_I(p)$ that is replaced at every update. The add operation guarantees $\mathcal{N}_I(p)$ has size N_I (if at least N_I peers are known). Overall, the in-neighbor update rate can be defined as

$$R_{\mathrm{up}} = \frac{F_{\mathrm{up}} N_I}{T_{\mathrm{up}}} \tag{2}$$

If not otherwise stated $N_I = 30$, $T_{\mathrm{up}} = 10\,\mathrm{s}$ and $F_{\mathrm{up}} = 0.3$. The latter two values result in a good compromise between adaptiveness and overhead. Their choice is robust, and sensitivity analysis is presented in Sect VI-C.

A. Metrics Driving The Neighborhood Selection

At every update, $\mathcal{N}_I(p)$ is the result of two separate filtering functions: one that selects the peers *to drop*, and another one selecting in-neighbors *to add*. For these filtering functions we consider both simple network attributes such as peer upload bandwidth, path RTT or path packet loss rate, and some application layer metrics, such as the peer offer rate[6] or number of received chunks from an in-neighbor.

Some metrics are static *peer metrics*: once estimated, they can be broadcast with gossiping messages and are known *a-priori*. Other metrics instead are *path attributes* between two peers and must be measured and can only be used as *a-posteriori* indicators of the quality of the considered in-neighbor as perceived by p.

Both add and drop filtering functions are probabilistic to avoid deadlocks and guarantee a sufficient degree of randomness. Considering any metric, we assign a selection probability w_q to every candidate q as

$$w_q = \frac{m_q}{\sum_{s \in \mathcal{N}_S(p)} m_s} \tag{3}$$

where m_q is the metric of q and $\mathcal{N}_S$ is either $\mathcal{N}_I$ for drop, or the set of candidate in-neighbors for add.

B. Add Filters

We consider the following four criteria to add new in-neighbors:
RND: Neighbors are chosen uniformly at random: $\forall q, m_q = 1$;
BW: Neighbors are weighted according to their upload bandwidth C_q: $\forall q, m_q = C_q$;
RTT: Neighbors are weighted according to the inverse of the RTT between p and q: $\forall q, m_q = 1/RTT_q(p)$; if $RTT_q(p)$ is still unknown, $RTT_q(p) = 1\,\mathrm{s}$[7];
OFF: Neighbors are weighted according to the rate they send offer messages R_q: $\forall q, m_q = R_q$; R_q are advertized by peers.

---

[4]Notice that since $\mathcal{N}_O(p)$ are built passively, they do not contribute to construction of the swarm topology.

[5]In the actual implementation $N_O(p)$ is limited to 200 peers, but the limit is never reached.

[6]HRC adapt the peer offer rate to peer upload capacity. It can thus be seen as an indirect measure of its available upload bandwidth.

[7]$RTT_q(p)$ are locally cached at p so that they may be available a priori. Active measurements could also be used to quickly estimate the RTT.

Table I
NUMBER OF PCs PER SUBNET.

Subnet	1	2	3	4
Number of PCs	43	63	60	38

Table II
RTTs IN ms BETWEEN SUBNETS OF PEERS.

	1	2	3	4
1	$20 \pm 10\%$	$80 \pm 10\%$	$120 \pm 10\%$	$160 \pm 10\%$
2	$80 \pm 10\%$	$20 \pm 10\%$	$140 \pm 10\%$	$240 \pm 10\%$
3	$120 \pm 10\%$	$170 \pm 10\%$	$20 \pm 10\%$	$200 \pm 10\%$
4	$160 \pm 10\%$	$240 \pm 10\%$	$200 \pm 10\%$	$20 \pm 10\%$

Table III
CHARACTERISTICS OF PEER CLASSES.

Class	Bandwidth	Percentage of Peers
1	$5\,\text{Mb/s} \pm 10\%$	10 %
2	$1.6\,\text{Mb/s} \pm 10\%$	35 %
3	$0.64\,\text{Mb/s} \pm 10\%$	35 %
4	$0.2\,\text{Mb/s}, \pm 10\%$	20 %

C. Drop Filters

For what concerns the criteria to select neighbors to be dropped, we consider:

RND: Neighbors are dropped randomly: $\forall q, m_q = 1$;

RTT: Neighbors are dropped with a probability directly proportional to the RTT between p and q: $\forall q, m_q = RTT_q(p)$;

RXC: Neighbors are dropped with a probability proportional to the inverse of the rate at which it transferred chunks to p: $\forall q, m_q = 1/RXC_q(p)$; this metric assigns a quality index related to the in-neighbor ability to successfully transfer chunks to p; $RXC_q(p)$ are evaluated on a window of 3 s.

D. Blacklisting Policies

Finally a peer in $\mathcal{N}_I(p)$ is blacklisted if one of the following criterion is met:

CMR: the ratio of corrupted/late chunks among the last 100 chunks received by p from q exceeds a threshold of 5%;

PLOSS: the packet loss rate from q to p exceed a threshold of 3%; measured over the last 300 packets received;

RTT: $RTT_q(p)$ is greater than 1 s.

Observe that this blacklist-based filter can be easily adapted to fight known problems of P2P systems such as free-riding and content pollution. However, we do not include these matters in our evaluation since they are out of the scope of this paper, i.e. the study of strategies for the overlay costruction.

Combining add and drop criteria we define 12 different overlay construction and maintenance filters. In the following, we name them stating the "ADD"-"DROP" policies, e.g., BW-RTT for add BW and drop RTT. Sect. V reports results for different resulting combinations. Blacklisting can be superposed (or not) to all of them, and its impact will be studied selectively. We tested also other metrics and combinations, whose results are less interesting. RND-RND is used as a baseline benchmark, as it is a policy based on pure random sampling of the swarm.

IV. TEST-BED CONFIGURATION

We need to benchmark the different algorithms in a known and reproducible scenario. To this aim, we run experiments in a possibly complex but fully controlled network to avoid fluctuations and randomness due to external impairments. The test-bed is built in labs available at Politecnico di Torino, with 204 PCs divided in four different subnets. Table I shows the number of PCs in each subnet. We used `tc`, the standard Linux Traffic Controller tool, together with the `netem` option to enforce delay and packet dropping probability when

needed. The chosen RTT distribution is described in Table II. The upload bandwidth is limited by the application itself, exploiting the feature of a simple leaky bucket (its memory being 10MB) to limit the application data rate to a given desired value. Peer upload capacities C_p are shown in Table III. Configurations in Tables II and III have been designed to resemble a world-wide geographic scenario, where peers are distributed over continents (clusters), and they rely on different kinds of access technologies, i.e., ADSL or FTTH interfaces, that provide different up-link capacity. Those configurations are not meant to be representative of any actual case, but rather they are instrumental to create benchmarking scenarios with different properties. Each PC runs 5 independent instances of PeerStreamer simultaneously, thus, a swarm of 1020 peers is built in every experiment, if not otherwise stated. The source peer runs at an independent server (not belonging to any of the subnets). It injects in the swarm 5 copies of each newly generated chunk, corresponding to roughly 6 Mbit/s.

The well known *Pink of the Aerosmith* video sequence has been used as benchmark. The nominal sequence length corresponds to 200s, with a time resolution equal to 25 frame/s. The sequence is looped for a total stream duration of about 20 min. After the initial 12 min of experiment, each peer starts saving on local disk a 3 min long video that we use to compute QoE metrics.

We selected the H.264/AVC codec to encode the video sequence. A hierarchical type-B frames prediction scheme has been used, obtaining 4 different kinds of frames that, in order of importance, are: IDR, P, B and b. The GOP structure is IDR$\times 8$ {P,B,b,b}. The nominal video rate of the encoder r_s is $1.2\,\text{Mb/s}$ if not otherwise specified. This corresponds to a system load $\rho = 0.9$ – defined as $\rho = r_s/E[C_p]$ where $E[C_p] = 1.32\,\text{Mbit/s}$ is the average upload bandwidth of peers.

The source node generates a new chunk at regular time, i.e., every new frame. The chunk size is instead highly variable due to the encoded video characteristics. Each peer implements a chunk buffer of 150 chunks. Given the one-frame⇔one-chunk mapping, and 25 fps of the video, this corresponds to a buffer of 6s, i.e., the play-out deadline is only 6 s.

A. Network Scenarios

The generic setup described above is used as a base for three different scenarios to evaluate significant situations. The first scenario, *G_Homo* hereafter, is geographically homogeneous: the distribution of the peers of different C_p classes is the same in any area, so that there is the same distribution of bandwidth everywhere. This scenario is useful to understand the fundamental behavior of different neighborhood filtering strategies.
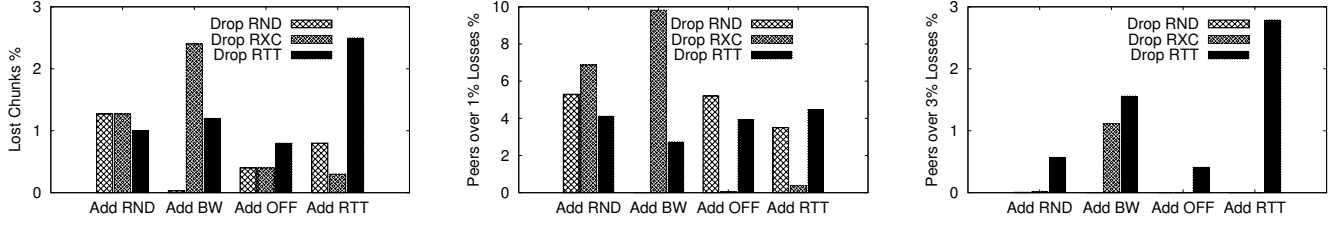
Figure 2. Frame loss for different strategies in *G_Homo* scenario: F_{loss} (average) (left), percentage of peers whose $F_{loss}(p) > 0.01$ (center), percentage of peers whose $F_{loss}(p) > 0.03$ (right).

The second scenario, *G_Bias* hereafter, assumes that bandwidth rich peers (Class 1) are all concentrated in a single subnet. This situation is particularly challenging for a topology management system that tries to localize traffic to reduce the network footprint of the application.

The third and final scenario, *G_Lossy* hereafter, is again geographically homogeneous, but the long-haul connections between the subnets 1–3, 1–4, 2–3, 2–4 are subject to packet loss with probability $p = 0.05$, while the intra-subnet links and the links between 1–2 and 3–4 are lossless. This situation is particularly useful to understand if black-listing can really help in building better topologies, or if its use should be limited to isolate misbehaving and malicious nodes.

Finally, churning of peers is modeled: a fraction P_{no-ch} of peers never leaves the system, while $P_{ch} = 1 - P_{no-ch}$ churning peers have a permanence time uniformly distributed between 4 and 12 min. To keep the number of peers constant, once a churning peer has left the system, it will be off for an average time equal to 30 sec before re-joining the swarm (with a different ID, i.e., as a new peer).

B. Performance Indices

As performance indices to assess the QoE, for each peer p, we consider the *frame loss probability*, $F_{loss}(p)$, and the SSIM (Structural Similarity Index), $S_{ssim}(p)$, a well-known method for measuring the similarity between two images in the multimedia field [16]. Given the highly structured organization of the video streams, the degradation of the received video quality becomes typically noticeable for values of $F_{loss}(p)$ higher than 1%, while loss probability of a few percent (3-4%) significantly impair the QoE. In the following, we report both average frame loss, $F_{loss} = E_p[F_{loss}(p)]$, and the percentage of peers that suffer $F_{loss}(p)$ larger than 1% and 3%, respectively.

Performance however should also take into account the cost for the network to support the application. As *network cost* ζ we consider the average of the distance traveled by information units. Formally, let $b_q(p)$ the number of bits peer p received from peer q; the peer p network cost $\zeta(p)$ is computed as

$$\zeta(p) = \frac{\sum_q RTT_q(p)b_q(p)}{\sum_q b_q(p)} \tag{4}$$
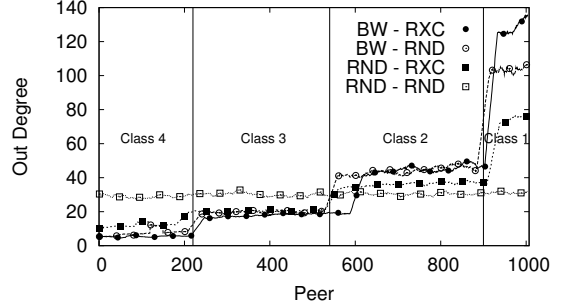
while the average network cost is $\zeta = E_p[\zeta(p)]$.

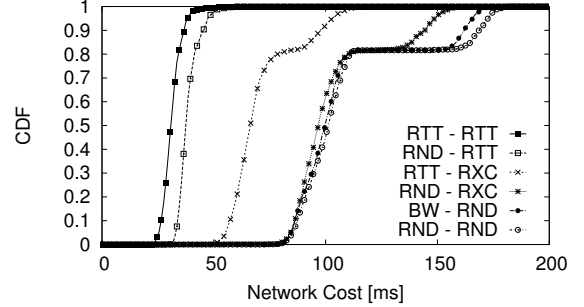Figure 3. Out-degree distribution of peers, *G_Homo* scenario.

Figure 4. CDF of the distance traveled by information units, *G_Homo* scenario.

V. Controlled Environment Experiments

A. G_Homo Scenario

We start considering the case in which the distribution of C_p is geographically homogeneous.

The left-hand plot in Fig. 2 shows the average frame loss probability experienced by different policies, while center and right-hand plots report the percentages of peers that experienced $F_{loss}(p) > 0.01$ and $F_{loss}(p) > 0.03$, respectively.

RND-RND is the reference, and we immediately observe that the other algorithms modify the loss *distribution*, i.e., they can have a different impact on different percentiles. For instance BW-RTT improves the average loss rate and the percentage of peers with $F_{loss}(p) > 0.01$, but at the expense of the percentage of peers with bad quality ($F_{loss}(p) > 0.03$), while RTT-RTT improves the number of peers with $F_{loss}(p) > 0.01$, but both the average and the percentage of peers with bad quality ($F_{loss}(p) > 0.03$) are worse.

In general adding policies sensitive to peers bandwidth (BW and OFF for adding and RXC for dropping) appear to be the more effective in reducing the losses. However the behavior of BW-RXC for which F_{loss} tops at 2.5% indicates that using a single metric for selecting the neighborhood can be dangerous.
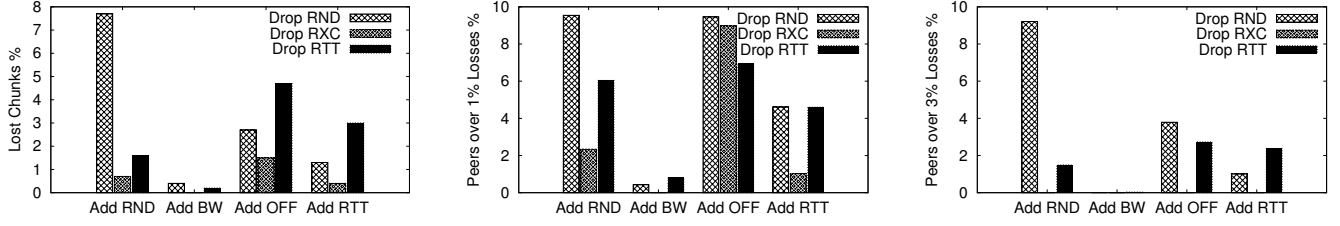
Figure 5. Frame loss for different strategies in *G_Homo* scenario with $N_I = 20$: F_{loss} (average) (left), percentage of peers whose $F_{loss}(p) > 0.01$ (center), percentage of peers whose $F_{loss}(p) > 0.03$ (right).

BW-RXC biases too much the choices toward high bandwidth peers, which become congested and are not able to sustain the system demand. To better grasp these effects, Fig. 3 reports the smoothed[8] histogram of the out-degree $N_O(p)$. Observe that $N_O(p)$ of peers belonging to different classes is significantly different as long as bandwidth aware policies are adopted; out-degrees are instead independent for RND-RND as expected. In principle it would be desirable to have an out-degree of a peer proportional to its up-link bandwidth. This is roughly achieved by adopting BW-RND policy. Under BW-RXC, instead, the degree distribution depends too much on C_p. As a result, high bandwidth peers tends to be oversubscribed while medium and low bandwidth peers may be underutilized.

Policies sensitive to RTT perform well in the considered scenario, with the exception of RTT-RTT, which is too aggressive in strictly selecting the closest in-neighbors. Indeed, as observed in [5], policies that force a too strict localization of traffic induce performance degradations due to poor topological properties of the swarm. To complement previous information Fig. 4 reports the Cumulative Distribution Function (CDF) of network cost $\zeta(p)$. As expected, RTT aware policies significantly reduce this index thanks to their ability to select in-neighbors within the same area.

Remark A - As a first consideration, we can say that: *i)* bandwidth aware policies improve the application performance; *ii)* RTT aware policies reduce the network cost without endangering significantly the video quality if applied to add peers; when used to drop peers, however, RTT poses significant bias impairing QoE; *iii)* the preference toward high bandwidth peers/nearby peers must be tempered to achieve good performance. The policy RTT-RXC improves quality and reduces the network cost at the same time, offering the best trade-off in this scenario. Interestingly, this policy is also easy to be implemented, since it requires to measure simple and straightforward metrics. Bandwidth aware schemes offers better QoE performance, at the cost of more cumbersome available capacity estimation.

B. G_Homo with Smaller N_I

We consider the same network scenario but we set $N_I = 20$. This is a more critical situation where choosing the good in-neighbors is more important. The value of N_I is related with

---

[8]The distribution of $N_O(p)$ inside classes is binomial as expected from theory. This distribution results in a large noisiness of the plot, so we apply a smoothing window of length 30 in plotting, basically showing the average N_O in each class.

the signaling overhead which increases with N_I, so having small neighborhood is desirable. However, a too small N_I would impair the availability of chunks.

Results are plotted in Fig. 5 (the y-scales in Figs. 2 and 5 are different for readability reasons, and this is the reason why at first sight some policies seem to perform better with a smaller N_I). The performance of RND-RND significantly degrades in this case. The reason is that the out degree of Class 1 peers under RND-RND is often not enough to fully exploit their bandwidth. Bandwidth aware strategies, instead, successfully adapt $N_O(p)$ to C_p maintaining high performance. Also RTT-RND and RTT-RTT, which are bandwidth unaware, perform better than RND-RND, since RTT-aware selection policies reduce the latency between an offer and the actual chunk transmission that follows it, helping in exploiting the peer's bandwidth. Results for network cost are similar to those in Fig. 4 and are not reported for the sake of brevity.

Remark B - Random selection policies, which are widely employed by the community as baseline and in the wild [17], are robust, but perform poorly if the number of peers in the neighborhood is small: all peers suffer 8% of frame loss, i.e., practically making it impossible to decode the video. As already seen with $N_I = 30$, the policy that combines bandwidth and RTT awarenesses (RTT-RXC) definitely improves both performance and network costs. Similarly, wisely selecting high-capacity in-neighbors is vital, as testified by the excellent performance of add BW policies.

C. G_Bias Scenario

Maintaining unchanged the C_p distribution, we localize all high bandwidth peers in geographical area 1. This scenario, in principle, constitutes a challenge for the policies that try to localize traffic. Indeed as side effect of the localization we can potentially have a "riches with riches", "poors with poors" clusterization effect that may endanger the video quality perceived by peers in geographical regions other than 1.

Fig. 6 reports the CDF of $F_{loss}(p)$ for the strategies performing better in the *G_Homo* scenario, plus the benchmark RND-RND. In this case if RTT is the only metric used as in RTT-RTT, the performance degrades unacceptably, and peers in area 1 are in practice the only one receiving a good service. In general, any policies based on drop RTT perform poorly. Strategies RTT-RXC, RND-RXC and BW-RND perform similarly; however, the only policy that can also reduce the network cost is RTT-RXC, as shown in Fig. 7 that reports the CDF of $\zeta(p)$.
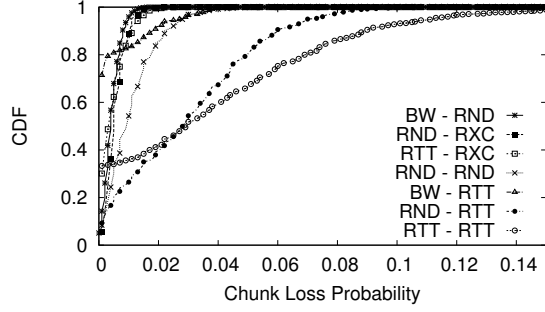
Figure 6. CDF of the frame loss probability for four different strategies, *G_Bias* scenario.
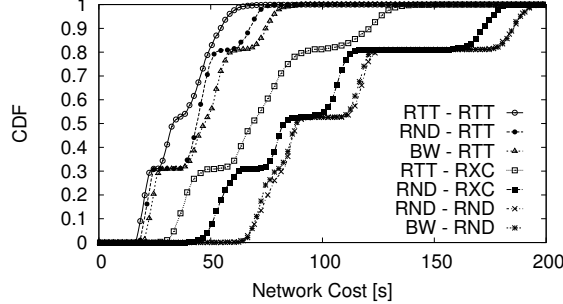

Figure 7. CDF of distance traveled by information units, *G_Bias* scenario.
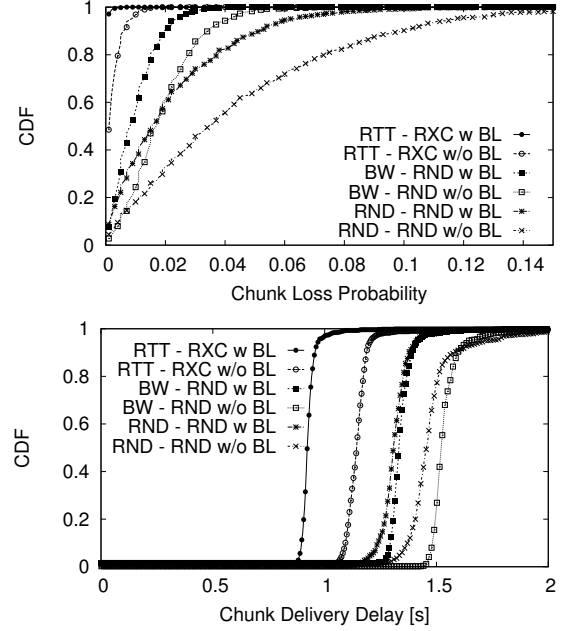

Figure 8. CDF of chunk loss probability (top) and CDF of chunk delivery delays (bottom) for six different strategies with and without adopting blacklist mechanism in *G_Lossy* scenario.

Remark C - This result essentially proves that also in *G_Bias* scenario it is possible to partially localize the traffic without endangering the video quality perceived by the user, as long as RTT awareness is tempered with some light bandwidth awareness, as in RTT-RXC. Interestingly, the RTT driven policies perform much better if the RTT is used to *add* peers rather than to *drop* peers. Indeed, in this latter case, aggressively dropping far away, but high capacity, in-neighbors penalizes peers which are located in areas where little high capacity peers can be found.

D. G_Lossy Scenario

We consider another scenario in which large bandwidth peers are uniformly distributed over the four subnets, but packet losses are present in some long haul connections.

Fig 8 plots the CDF of frame losses (top) and the CDF of chunks delivery delays (bottom) for the selected policies. Blacklisting improves the performance of every policy. RTT-RXC emerges again as the most performing policy and with blacklisting practically all peers are able to receive all chunks. This is an excellent result, since the system is facing a very challenging scenario while working with a load of 0.9.

Benefits of the blacklisting mechanism are confirmed by Table IV that reports the normalized volume of incoming

traffic for peers in cluster 2 from peers in all clusters. Keeping in mind that in *G_Lossy* scenario peers belonging to cluster 2 experience lossy paths from/towards peers in cluster 3 and 4 (as explained in Sec. IV), it is easy to see that volumes of incoming traffic from cluster 3 and 4 are nicely reduced thanks to blacklisting mechanism.

Remark D - Blacklisting can play a significant role to avoid selecting lossy paths. Indeed, exploiting the blacklist mechanism every peer should identify and abandon poorly performing peers, biasing the neighborhood toward good performing in-neighbors. This effect reinforces policies that naturally bias the selection of neighbor peers employing peer quality. RND-RND, BW-RND and RTT-RXC have emerged as the most promising criteria (RND-RND being the baseline benchmark). RTT-RXC with blacklisting is shown to guarantee excellent performance to all peers even in this almost adversarial scenario.

VI. VIDEO PERFORMANCE EVALUATION

A. Video performance versus load

In the previous sections we have benchmarked the system versus increasingly difficult scenarios, showing the benefits and drawbacks of overlay topology filtering strategies. Now we summarize the results by depicting the actual average QoE by reporting S_{ssim} for different policies and different system loads. We consider the final *G_Lossy* scenario, and we increase r_s from 0.6 Mb/s to 1.4 Mb/s. Recall that $E[C_p] = 1.324$ Mb/s.

Fig. 9 shows average S_{ssim} considering RND-RND, BW-RND and RTT-RXC with and without blacklisting. SSIM is a measure of the distortion of the received image compared against the original source (before encoding and chunkization).

Table IV
AVERAGE FRACTIONS OF INCOMING TRAFFIC FOR CLUSTER 2.

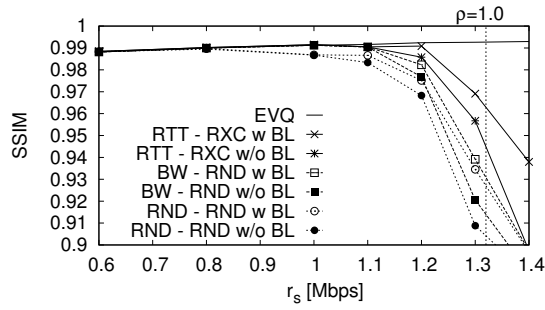|	1 - good	2 - local	3 - bad	4 - bad + far
RND - RND w/o BL	0.23	0.32	0.28	0.15
RND - RND w BL	0.28	0.34	0.24	0.12
BW - RND w/o BL	0.22	0.35	0.27	0.14
BW - RND w BL	0.23	0.36	0.24	0.13
RTT - RXC w/o BL	0.12	0.68	0.11	0.07
RTT - RXC w BL	0.13	0.70	0.09	0.05

Figure 9. S_{ssim} (average) index when varying video rate r_s in *G_Lossy* scenario.

Table V
AVERAGE S_{ssim} WHEN INCREASING THE NUMBER OF INVOLVED PEERS N. *G_Homo* SCENARIO, $r_s = 1.2$ MB/S.

N	204	612	1040	1428	1836	2080
RND - RND	0.858	0.812	0.829	0.783	0.799	0.799
RTT - RXC	0.984	0.981	0.988	0.979	0.988	0.991

It is a highly non linear metric in decimal values between -1 and 1. Negative values correspond to negative images, so are not normally considered at all. Values above 0.985 are typically considered of excellent quality. SSIM has been computed considering the video between min. 12 and 13 (60x25 frames) received by 200 peers (50 for each class), and then averaging among all of them.

The EVQ (Encoded Video Quality) curve in the plot is the reference value for the encoding rate and it obviously increases steadily as r_s increases. In general, when the system load is small $\rho << 1$, average S_{ssim} increases for increasing r_s thanks to the higher quality of the encoded video. However, as ρ approaches 1, different topologies behave differently: S_{ssim} rapidly drops due to missing chunks which impair the quality of the received video, but the degradation is highly influenced by the topology. Notice how RTT-RXC scheme outperforms RND-RND and BW-RND for every value of r_s. Fig. 9 also shows the benefits of the blacklist mechanism for every scheme.

Remark E - RTT-RXC with blacklisting guarantees optimal QoE for $\rho < 1$ whereas RND-RND policies is not able to guarantee good QoE for $\rho > 0.75$.

B. Scaling with swarm size

Considering again *G_Homo* scenario, we study how the system scales when increasing the size of the swarm N from 200 to 2000 peers. Due to the lack of space, we only report in Table V the average S_{ssim} for three different values of N. RND-RND and RTT-RXC schemes have been adopted as benchmark. Transmitted video was encoded at $r_s = 1.2$ Mb/s, i.e. system load $\rho = 0.9$. The simple bandwidth-aware scheme, RTT-RXC, always ensures better performance with respect to RND-RND, i.e. the average S_{ssim} improves from 0.8 to 0.99, a remarkable gain. Increasing N has a negligible impact on performance, especially when the smart RTT-RXC policy is adopted. Indeed, in RND-RND case, the topology overlay evolution causes more random results due to the totally random nature of the scheme.
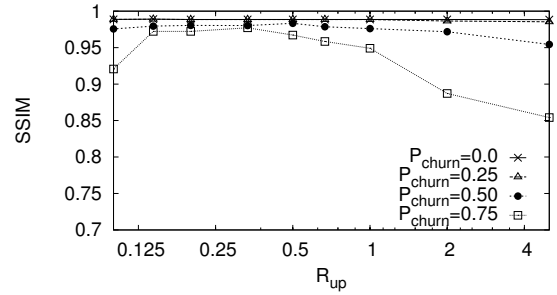

Figure 10. Average S_{ssim} vs R_{up} for different fractions of churning peers P_{churn}. Scheme RTT-RXC in *G_Homo* scenario with 1000 peers and $r_s = 0.8$ Mb/s.
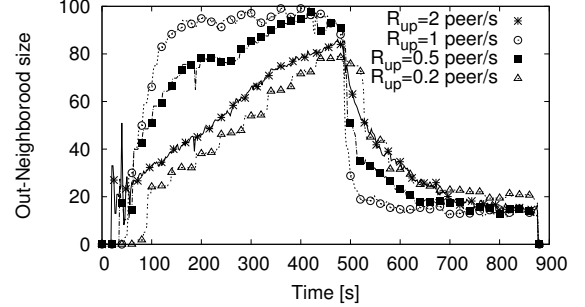

Figure 11. The evolution during time of the average outgoing neighborhood size setting different R_{up} values. Scheme RTT-RXC in *G_Homo* scenario with $r_s = 1.0$ Mb/s.

C. Sensitivity to Update Rate with Churning

We investigate what are the best trade-off values for the frequency to update the incoming neighborhood, R_{up} as defined in (2).

Consider a *G_Homo* scenario with P_{churn} fraction of peers that join and leave the swarm. In Fig. 10 we report the S_{ssim} (computed and averaged over peers that never leave the system) when varying R_{up}. In particular, we fix $N_I = 30$, $F_{up} = 0.3$ and change $T_{up} \in [2, 100]$ s accordingly. For this case we adopted scheme RTT-RXC and $r_s = 0.8$ Mb/s. The plot shows that the system is very robust to different R_{up} values. Only under stressed scenarios, such as for $P_{churn} >= 0.5$, R_{up} becomes critical: too high R_{up} does not let the swarm achieve a stable state, impairing performance. On the other hand, too low R_{up} induces peers to react slowly to sudden changes brought by churning peers.

We considered the *G_Homo* scenario again, but forcing all high-bandwidth peers to experience an abrupt up-link bandwidth reduction from 5 Mb/s to 0.64 Mb/s (on average) at time 480 s from the beginning of the video transmission. While this scenario is rather artificial, it allows to gauge the reactiveness of the topology to such abrupt changes. We consider the RTT-RXC scheme. Fig. 11 reports the evolution over time of the average size of the outgoing neighborhood N_O of class 1 peers. Different values of in-neighborhood update rate R_{up} are considered. Two observation holds: first, smaller values of R_{up} slow down system reactiveness. However, too large values, e.g., $R_{up} = 2$ peer/s, impair the performance as well: in this case, peers have not enough time to collect significant measurements about the in-neighbor "quality" (amount of re-

ceived chunks), and thus find it difficult to distinguish "good" from "bad" in-neighbors. Also in this case $R_{up} = 1$ peer/s setup represents a good trade off.

Remark F - Fast topology updates allow the overlay topology to react quickly *i)* to changes in the network scenario and *ii)* to prune quickly peers which left the system in, e.g., heavy churning conditions. However, too fast updates introduce instability in the overlay construction process, driving peers to never achieve a stable incoming neighborhood, and thus leading to bad system performance. The best trade-off R_{up} value is $R_{up} = 1$ peer/s, i.e., $T_{up} = 10$ s.

VII. PLANETLAB EXPERIMENTS

We now present similar experiments on PlanetLab. We selected 449 nodes scattered worldwide. No artificial latency or packet loss are imposed, so that they reflect the natural Internet conditions. Peer upload capacity has been limited by PeerStreamer embedded rate limiter; two classes are present: half of peers have 2 Mbit/s at their up-link, and 0.64 Mbit/s the other half. Average upload capacity results to 1.32 Mbit/s. Observe that this is an upper-bound to the actual available peer upload bandwidth which may be reduced due to competing experiments running on the same PlanetLab node or due to other bottlenecks on the access links of the node. Thus, in general, the actual upload capacity of a peer is $C'_p \leq C_p$.

Fig. 12 reports each peer's individual SSIM performance, $S_{ssim}(p)$, for $r_s = 0.8$ Mbit/s (top) and $r_s = 1.0$ Mbit/s (bottom). $S_{ssim}(p)$ has been sorted in decreasing values to ease visualization and each curve represents the average of 5 different runs. Observe that when the amount of system resources is large enough with respect to the video-rate, i.e, when $r_s = 0.8$ Mbit/s (top plot), different schemes for topology management perform rather similarly. Observe, however, that there is always a certain fraction of nodes that cannot receive the video due to congestion at local resources. Increasing system load, i.e. $r_s = 1.0$ Mbit/s (bottom plot), highlights differences among schemes and confirms results obtained in the controlled environment: random-based policies (RND-RND) perform badly in general; same holds for schemes based on pure proximity that can lead to disconnected topologies and, then, to bad QoE performance (RTT-RND). However, if combined with bandwidth-awareness, proximity-based schemes achieve the goal of localizing traffic without impairing performance (RTT-RXC).

VIII. RELATED WORK

Many popular commercial applications such as *PPLive, SopCast, Octoshape* were proposed in recent years, but no information about their internal implementation has been made available, making any statement about their overlay topology design strategies impossible. Only a recent study suggests that simple random based policies are adopted by SopCast [17]. Focusing on available literature on purely mesh-based P2P-TV systems, many solutions can be found, but also in this case, to the best of our knowledge, none of them provides *general and detailed* guidelines for the overlay topology design process.
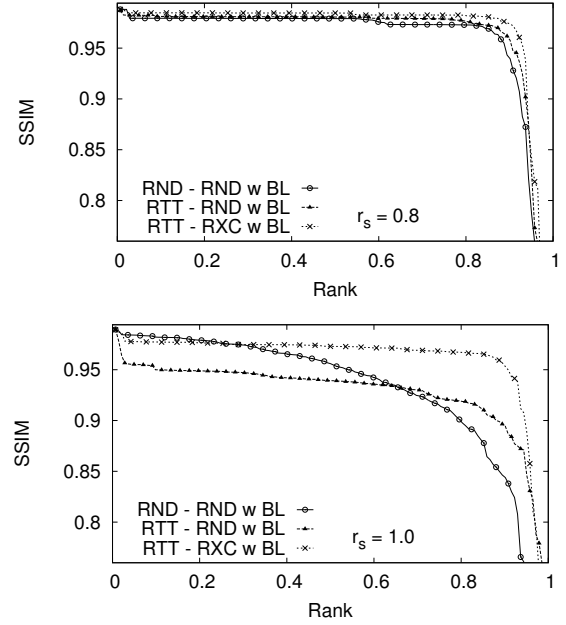

Figure 12. $S_{ssim}(p)$ for $r_s = 0.8$ Mb/s and $r_s = 1.0$ Mb/s on PlanetLab experiments.

An early solution called *GnuStream* was presented in [18]. Based on *Gnutella* overlay, *GnuStream* implemented a load distribution mechanism where peers were expected to contribute to chunks dissemination in a way proportional to their current capabilities. A more refined solution called *PROMISE* was introduced in [19]. Authors proposed an improved seeder choice based on network tomography techniques; peers were interconnected through *Pastry* overlay topology which implements —as many others P2P substrates like *Chord* [20] or *CAN*— some location awareness based on number of IP hops. *DONet* (or *Coolstreaming*) [2] is a successful P2P-TV system implementation. This design employs a scheduling policy based on chunk rarity and available bandwidth of peers, but its data-driven overlay topology does not exploit any information from underlying network levels. Many new features were introduced in [21] to improve the streaming service and, in particular, authors proposed a new neighbor re-selection heuristic based only on peers up-link bandwidth. In [22], authors showed the design aspects of their application called *AnySee*. Even if partially based on multicast, this hybrid mesh-based system relies on an overlay topology that aims at matching the underlying physical network while pruning slow logical connections. However, no deep investigation about performance of their overlay design strategy is provided. In [23] authors presented a study about some key design issues related to mesh-based P2P-TV systems. They focused on understanding the real limitations of this kind of applications and presented a system based on a directed and randomly generated overlay. Some fundamental improvements were introduced: e.g., the degree of peers' connectivity proportional to their available bandwidth.

Turning our attention on more theoretical studies about the overlay topology formation, in [3] the problem of building an

efficient overlay topology, taking into account both latency and bandwidth, has been formulated as an optimization problem; however, the interactions between overlay topology structure and the chunk distribution process are ignored.

In [24] a theoretical investigation on optimal topologies is formulated, considering latency and peer bandwidth heterogeneity; scaling laws are thus discussed. In [4], a distributed and adaptive algorithm for the optimization of the overlay topology in heterogeneous environments has been proposed, but network latencies are still ignored. Authors of [25] propose a mechanism to build a tree structure on which information is pushed. They show that good topological properties are guaranteed by location awareness schemes. Similar in spirit, but in unstructured systems, we propose in this paper an overlay topology design strategy that, taking into account latency and peer heterogeneity, aims at creating an overlay with good properties and low chunk delivery delays. In highly idealized scenarios, [26] shows with simple stochastic models that overlay topologies with small-world properties are particularly suitable for chunk distribution in P2P-TV systems.

Finally, in [9], authors experimentally compare unstructured systems with multiple-tree based ones, showing that former systems perform better in highly dynamic scenarios as well as in scenarios with bandwidth limitations. This strengthen our choice of exploring topology management policies for mesh-based streaming systems.

IX. CONCLUSIONS

The impact of P2P-TV overlay topologies have been studied mainly using analysis or simulation. Few proposals undergo implementation, and almost none have been extensively benchmarked in large scale test-beds.

This work aims at filling this gap. Leveraging the PeerStreamer application developed within the framework of the NAPA-WINE project, we developed a P2P-TV system where it is possible to change the strategies for building neighborhoods of peers, and hence the overall topology, without changing other algorithms of the application, thus isolating the impact of topology management from other effects.

In a fully controlled networking environment, we have run a large campaign of experiments measuring the impact of different filtering functions applied to the management of peer neighborhoods. Results show that proper management, based on simple RTT measurements to add peers, coupled with an estimation of the quality of the peer-to-peer relation to drop them, leads to a win-win situation where the performance of the application is improved while the network usage is reduced compared to a classical benchmark with random peer selection.

PeerStreamer is released as Open-Source to make results reproducible and allow further research.

REFERENCES

[1] V. Pai, K. Kumar, K. Tamilmani, V. Sambamurthy, and A. E. Mohr, "Chainsaw: Eliminating trees from overlay multicast," in IPTPS, Ithaca, NY, US, February 2005.

[2] X. Zhang, J. Liu, and T. Yum, "Coolstreaming/donet: A data-driven overlay network for peer-to-peer live media streaming," in *IEEE INFOCOM*, Miami, FL, US, March 2005.

[3] D. Ren, Y. T. H. Li, and S. H. G. Chan, "On Reducing Mesh Delay for Peer-to-Peer Live Streaming," in *IEEE INFOCOM*, Phoenix, AZ, US, April 2008.

[4] R. Lobb, A. P. Couto da Silva, E. Leonardi, M. Mellia, and M. Meo, "Adaptive Overlay Topology for Mesh-Based P2P-TV Systems," in *ACM NOSSDAV*, Williamsburg, VA, US, June 2009.

[5] A. Couto da Silva, E. Leonardi, M. Mellia, and M. Meo, "Chunk Distribution in Mesh-Based Large Scale P2P Streaming Systems: a Fluid Approach," *IEEE Trans. on Parallel and Distributed Systems*, vol. 22, no. 3, pp. 451–463, March 2011.

[6] X. Jin and Y.-K. Kwok, "Network aware P2P multimedia streaming: Capacity or locality?" in *IEEE P2P*, Kyoto, JP, August 2011.

[7] Y. Liu, "On the minimum delay peer-to-peer video streaming: how realtime can it be?" in *ACM Multimedia*, Augsburg, DE, September 2007.

[8] T. Bonald, L. Massoulié, F. Mathieu, D. Perino, and A. Twigg, "Epidemic live streaming: optimal performance trade-offs." in *SIGMETRICS*, Annapolis, MD, US, June 2008.

[9] J. Seibert, D. Zage, F. S., Nita-rotaru, and C., "Experimental comparison of peer-to-peer streaming overlays: An application perspective," in *IEEE LCN*, Montreal, QC, CA, October 2008.

[10] F. Picconi and L. Massoulié, "Is there a future for mesh-based live video streaming?" in *IEEE P2P*, Aachen, DE, September 2008.

[11] R. Birke, E. Leonardi, M. Mellia, A. Bakay, T. Szemethy, C. Kiraly, R. Lo Cigno, F. Mathieu, L. Muscariello, S. Niccolini, J. Seedorf, and G. Tropea, "Architecture of a Network-Aware P2P-TV Application: the NAPA-WINE Approach," *IEEE Comm. Magazine*, vol. 49, June 2011.

[12] L. Abeni, C. Kiraly, A. Russo, M. Biazzini, and R. Lo Cigno, "Design and implementation of a generic library for P2P streaming," in *Workshop on Advanced Video Streaming Techniques for Peer-to-Peer Networks and Social Networking*, Florence, IT, October 2010.

[13] L. Abeni, C. Kiraly, and R. Lo Cigno, "On the Optimal Scheduling of Streaming Applications in Unstructured Meshes," in *IFIP Networking*, Aachen, DE, May 2009.

[14] R. Birke, C. Kiraly, E. Leonardi, M. Mellia, M. Meo, and S. Traverso, "Hose Rate Control for P2P Streaming Systems," in *IEEE P2P*, Kyoto, JP, August 2011.

[15] N. Tölgyesi and M. Jelasity, "Adaptive peer sampling with newscast," in *Proc. of the 15th International Euro-Par Conf. on Parallel Processing.* Berlin, Heidelberg: Springer-Verlag, 2009, pp. 523–534.

[16] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Trans. on Image Processing*, vol. 13, no. 4, pp. 600–612, April 2004.

[17] I. Bermudez, M. Mellia, and M. Meo, "Passive characterization of SopCast usage in residential isps," in *IEEE P2P*, Kyoto, JP, August 2011.

[18] X. Jiang, Y. Dong, D. Xu, and B. Bhargava, "Gnustream: a P2P media streaming system prototype," in *IEEE ICME*, Washington, DC, US, 2003.

[19] M. Hefeeda, A. Habib, B. Botev, D. Xu, and B. Bhargava, "Promise: peer-to-peer media streaming using collectcast," in *ACM Multimedia*, New York, NY, US, 2003.

[20] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," *SIGCOMM Comput. Commun. Rev.*, vol. 31, pp. 149–160, August 2001.

[21] B. Li, S. Xie, Y. Qu, G. Y. Keung, C. Lin, J. Liu, and X. Zhang, "Inside the new coolstreaming: Principles, measurements and performance implications," in *IEEE INFOCOM*, Phoenix, AZ, US, April 2008.

[22] X. Liao, H. Jin, Y. Liu, L. M. Ni, and D. Deng, "Anysee: Peer-to-peer live streaming," in *IEEE INFOCOM*, Barcelona, ES, April 2006.

[23] N. Magharei and R. Rejaie, "Prime: Peer-to-peer receiver-driven mesh-based streaming," in *IEEE INFOCOM*, Anchorage , AK, US, May 2007.

[24] T. Small, B. Liang, and B. Li, "Scaling laws and tradeoffs in Peer-to-Peer live multimedia streaming," in *ACM Multimedia*, Santa Barbara, CA, US, October 2006.

[25] T. Locher, R. Meier, S. Schmid, and R. Wattenhofer, "Push-to-Pull Peer-to-Peer Live Streaming," in *Distributed Computing*, ser. Lecture Notes in Computer Science, A. Pelc, Ed. Springer Berlin / Heidelberg, 2007, vol. 4731, pp. 388–402.

[26] J. Chakareski, "Topology construction and resource allocation in p2p live streaming," in *Intelligent Multimedia Communication: Techniques and Applications*, ser. Studies in Computational Intelligence, C. Chen, Z. Li, and S. Lian, Eds. Springer Berlin / Heidelberg, 2010, vol. 280, pp. 217–251.

Summary Review Documentation for

"Experimental comparison of neighborhood filtering strategies in unstructured P2P-TV systems"

Authors: S. Traverso, L. Abeni, R. Birke, C. Kiraly, E. Leonardi, R. Lo Cigno, M. Mellia

REVIEWER #1

The paper presents the results of an empirical measurement study for understanding the performance impact of design decisions in a P2P live-streaming P2P. The authors consider a significant chunk of the parameter space and highlight several interactions between optimization criteria that significantly hurt (or improve) performance. In addition, the authors have made their video streaming testbed software open source so others can potentially test other scenarios.

Strengths: The paper strengths include: (i) there is a quite extensive empirical study of the P2P streaming design space; (ii) the authors are making their software open source, which should facilitate future research in this area; (iii) rhe scale of the experiments is fairly large for a deployment that has not yet been adopted by end users.

Weaknesses: The weaknesses include: (i) it was hard for me to see the connection between these empirical results and what was predicted by theory. Was it the same? Where was it different? (ii) I would have liked to see the authors take a position on whether they think RTT-RXC is the optimal solution in practice, or whether they think there is an alternative approach that might to better; (iii) the ACME-Streamer software makes a set of assumptions about system design that may make it difficult to try more advanced strategies to build the overlay topology (not using one-hop neighbors).

REVIEWER #2

The authors have implemented and will make available an open-source mesh-based P2P streaming video service. They propose several similar techniques to select neighbors to add, and to replace. Combinations of these methods are investigated experimentally, in a small scale campus testbed, and in PlanetLab. The results indicate that policies based on upload bandwidth (aggregate, and per peer) are consistently high quality. This is ambitious, outstanding work with main contributions of (a) the software, and (b) the method of evaluation. The actual findings on topology construction, while reasonable, are not the major contribution.

Strengths: The paper strengths include: (i) the paper is well written and easy to understand both the proposal and the results; (ii) the implementation is outstanding and will be particularly beneficial if made available to the community, as indicated; (iii) the evaluation is very thorough and can serve as a model for other researchers on how to investigate their ideas and demonstrate their benefits. Sensitivity to parameter choices was also measured; (iv) the results showing RTT-RXC is a good overall choice are interesting and useful, since this

is easy to implement, but probably can be improved upon. (v) the implementation in Planetlab was also useful.

Weaknesses: The weaknesses include: (i) there is no attempt to address free-riding or content pollution, which have turned out to be actual problems in P2P; (ii) in the graphs it is difficult to tell the differences between the lines, since stipple patterns and symbols are quite small.

REVIEWER #3

The paper experimentally analyzes different peer selection strategies for maintaining an unstructured P2P-TV system. A wide variety of strategies are considered, with RTT-RXC (peers with low RTT more likely to be added, peers with low transfer rate to us more likely to be dropped) along with blacklisting emerging as a winning combination. The paper gives detailed evaluation on both a controlled network with 200 PCs and also within the PlanetLab testbed.

Strengths: The paper strengths include: (i) the paper is very well written, provides an accessible reference for related work and distills take-away messages from the evaluation in a clear and lucid manner; (ii) the evaluation is done in the context of a fully functioning video streaming system, allowing metrics such as the SSIM to be used; (iii) the controlled evaluation on 200 PCs are supplemented with actual PlanetLab experiments, lending further credibility to the findings.

Weaknesses: First, the RTT-RXC policy emerges as a clear-cut winner throughout your evaluation. However, given that a majority of the evaluation in the paper is based on experiments on the test-bed configuration, I worry that tc/netem settings (such as the RTT in Table II) are too generous for the RTT approach to show any drawbacks compared to other strategies, and that the PlanetLab evaluation does not happen to display it. Also, the blacklisting time-out of 50s feels arbitrary. Why should the time-out even be a constant? Last, the paper does not convincingly address the concern that certain add/drop strategies can lead to a system partition (akin to Chord splitting into two separate rings) or provide any theoretical justification (e.g., by using game theory) for the preferable properties of the strategy it found to work best.

REVIEWER #4

The paper is an experimental study of different strategies to choose neighbours in a P2P-TV system, on a cluster and on Planetlab. The paper shows that performance is better with bandwidth-aware strategies, but at a higher network cost, while RTT-based strategies are almost always performing well, at a lower network cost.

Strengths: The paper strengths include: (i) the study is based on a real implementation, although the test-beds are not (neither the cluster nor Planetlab can be considered as realistic peer-to-peer settings for P2P-TV); (ii) different variants are studied, and most of the pros and cons of each variant are presented; (iii) for each part, conclusions are written outside of the details, helping for fast reading.

Weaknesses: The paper gives too many details about the implemention itself (and it is not clear for the non-specialist which details matter in this context), but fails to provide a formal specification of the protocols. In addtion, as in many experimental studies like this one, it is hard to generalize these results to other settings, with different parameters (upstream/downstream bandwidth, RTTs, NATs, video properties, etc.), although it is still interesting to have a comparison of the different variants in this setting. Last, the different variants are not all studied in the different settings. Sometimes, it looks like the authors have a preference from the beginning, and focus mostly on their preferred variant (RTT-RXC).

REVIEWER #5

This paper presents experimental results gained from an open source P2P live streaming system. The focus is on a systematic comparison of the influence of different selection strategies for peers which are used as parents of a stream. The paper shows that adding peers based on their RTT, coupled with removing peers providing low bandwidth, outperforms random selection strategies.

Strengths: First, the paper provides a large set of results gained from experiments in a real testbed, even though the selected testbed parameters (RTT, ..) appear a bit specific. The results are also complemented with experiments in Planetlab. Also, the results have been gained with an open source P2P-TV implemenation which allows to reproduce the experiments, even though the implementation is not accessible at this time due to the double blind review process. Finally, the paper analyses different selection strategies in a systematic manner, covering a quite broad range of possible strategies and showing their trade-offs. The paper also provides relevant conclusions for each of the evaluated scenarios.

Weaknesses: Some of the results seem to be a bit specific to the considered scenario and testbed environment. The paper should discuss more to what extent the results can be generalized. Also the results do not have confidence intervals which suggests that only one run has been conducted for each experiment. Moreover, the campus scenarios do not assume any cross traffic which is not very realistic. Finally, the authors claim negligible overhead, but this is not revealed in the paper.

The paper seems to ignore more incentive-compatible selection strategies e.g. tit-for-tat. Thus, it is questionable if the proposed strategy would provide good performance in case of free riders. To that end the paper also completely leaves out selection strategies for children rather than parents. Finally, the comparison to Random selection strategies cannot be considered state of the art, since more sophisticated strategies have been proposed.

The QoE part is weak and the applied QoE metric appears questionable. More justification is required as to how the considered metrics relate to real QoE. To that end the overall performance criteria appears unclear. It would be useful to provide results for the performance of the entire swarm, as the goal was to maximize that.

RESPONSE FROM THE AUTHORS

We would like to thank the Reviewers for their suggestions that helped us to improve our work.

All reviewers appreciated the experimental results, and the extensive evaluation conducted on synthetic test bed and Planetlab scenario. Our goal was to provide a fair and unbiased evaluation of intuitive overlay construction policies. To the end, we are happy to offer the PeerStreamer implementation as Open Source to allow further investigation and improvement by the research community.

In preparing the final version of the paper we addressed all the minor editorial comments. Only two questions were addressed by some more in depth changes:

Reviewer 5 questioned the usage of active and passive measurement. This has been addressed by specifying that in this paper we rely only on passive measurements. Nonetheless, active measurements could result useful in some cases, e.g., in case the path capacity toward a peer has to be evaluated.

Reviewer 5 questioned the choice of the testbed networking emulation. This has been addressed by noting that the configurations in Tables II and III have been designed to resemble a world-wide geographic scenario, where peers are distributed over different continents and rely on different up-link capacity as given by current access technologies, i.e. ADSL or FTTH interfaces. The resulting scenarios are not meant to be representative of any actual case, but rather they are instrumental to create different benchmarking cases, each with different properties. This could artificially stress the differences on performance. Yet, we expect that the RTT-RXC policy results the best also in wild Internet deployment, as confirmed by the PlanetLab experiments.