

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

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

^a DELEN, Politecnico di Torino, Italy; ^b DISI, University of Trento, Italy; ^c IBM Research Lab. Zurich, CH

Abstract—P2P-TV systems performance are severely impacted by the overlay topology peers form. Yet, detailed experimental studies on its features are missing. This work provides a comprehensive experimental comparison of different strategies for the construction and maintenance of the overlay topology for P2P-TV systems. We have implemented different fully-distributed strategies for the design of the overlay topology within WineStreamer, the P2P-TV application developed with EU NAPA-WINE project. An extensive campaign of experiments has been conducted in a fully controlled set-up which involves 1000 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 impact. Strategies based solely on random peer selection are greatly outperformed by smart, yet simple strategies that can be implemented at low cost and with negligible overhead. Even in face of different and complex scenarios, the neighborhood filtering strategy we devised as most performing guarantees to deliver almost all chunks to all peers with a playout delay as low as only 6s even with system loads as high as 0.9.

WineStreamer is a fully fledged P2P-TV application, open-sourced 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. Similarly to 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, differently from 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 hardly adapted to live P2P-TV systems.

Two are the key features that characterize a mesh based P2P-TV system: i) the algorithm according to which the mesh overlay topology is constructed and maintained, ii) the algorithm according to which chunks are traded among peers. A large body of research work has focused on the design and analysis of efficient algorithms for 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. Few works undergo implementation and present actual experiments, and even those are usually limited to few tens of peers. A detailed discussion of related work is presented in Sec. II.

What is still missing is a systematic comparison of different algorithms in an actual and known environment. 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 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 a *selection* and *replacement* criteria, according to which each peer chooses the peers he would like to have as parents (i.e., peers from which download chunks). Blacklisting feature allows to avoid cycling among bad parents. Overall, we explore 12 different combination of criteria (24 if blacklisting is enabled), based on metrics such as Round Trip Time (RTT), peer 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.

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 inter-cluster RTT being smaller than intra-cluster RTT. Then we consider a biased upload capacity distribution, where high capacity peers are all in the same cluster. Next, we add the impact of eventual packet loss on long-distance paths among clusters, facing an almost adversarial scenario.

Results show that policies that select peers based on network distance coupled with policies that drop peers based on their contribution are performing well in all scenarios.

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 comprising 1000 peers. All the software we have developed is Open Source and released under the (L)GPL.

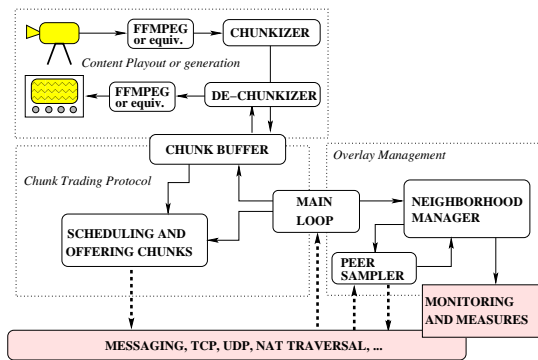


Figure 1. OfferStreamer peer architecture.

II. WINESTREAMER DESCRIPTION

Empowering this work is WineStreamer¹. WineStreamer is an Open Source P2P-TV client that stems from the developments and research of the NAPA-WINE EU project² whose overall architecture and vision are described in [15]. WineStreamer leverages GRAPES [16], a set of C libraries implementing building blocks for P2P-TV streaming that enables building applications with almost arbitrary characteristics, thus allowing for experimental works comparing different choices to be done efficiently.

Fig. 1 describes the logic and modular organization of WineStreamer. Given the focus of this paper, the overlay management is detailed in Sect. III-A, while in the following we sketch the high level organization of the other components of the application.

A. WineStreamer architecture

WineStreamer is based on a chunk-based stream diffusion. Peers that own some chunks offer a selection of them to some destination peers in their neighborhood. The receiving peer acknowledges the chunks he is interested into, thus avoiding multiple transmissions of the same chunk to the same peer. The negotiation and chunk transmission phase is sketched in Fig. 2: signaling exchanges (Offer/Select messages) are represented above the time line and chunk transmissions (colored blocks) are below the time line. 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 so large to cause the accumulation of chunks to be transmitted adding queuing delay before to chunk transmissions. In [17] we proposed Hose Rate Control (HRC) to automatically adapt the number of offers to both peer upload capacity and system demand. In this paper we adopt HRC: simpler trading schemes are less performing and can hide the impact of the overlay on the overall performance of the system.

For chunk scheduling, offers are sent neighbors in round-robin. They contain the buffermap of the recent chunks the peer 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 latest useful policy means the peer selects the most recent chunk it does not have. This has been proven optimal for streaming systems with centralized scheduling in [?].

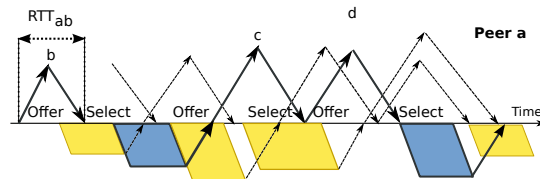


Figure 2. Pictorial representation of chunk exchanges from one peer with the offer-select protocol used by WineStreamer.

The source acts as a standard peer, but it does not participate in the offer/select protocol. It simply injects one or more 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 MPEG TS stream coming from a DVB-T card, or the raw video of a webcam). The chunking strategy used in WineStreamer is trivial to avoid mingling its effects with the topology-related ones: one-frame \rightarrow one-chunk policy to avoid that a missing chunk would impair several frames due to, e.g., missing frame headers. The chunkiser is implemented using the ffmpeg libraries³, so that several different codecs (e.g., MPEG video, theora, H.264, etc.) are supported. Receiving peers, instead, implement a de-chunkiser, which reads from the chunk buffer and pushes the chunks to the playout system where they can be played in the correct sequence.

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 of a signaling message from the messaging layer below.

WineStreamer architecture is completed by the “messaging” and “monitoring and measures” modules. The messaging module is a network abstraction layer, so that it frees the application from all details of the networking environment, e.g., the presence of NAT, middleboxes and other communication details. It offers a connection-oriented service on the top of UDP, with a lightweight retransmission mechanisms that allows to recover few percentages of packet loss without suffering high retransmission delay.

The monitoring and measures module extracts network layer information by running passive or active measurements: passive measurements are performed by observing the messages that are exchanged anyway between two peers. Active measurements, in contrast, craft special probe messages which are sent to other peers at the discretion of the monitoring module. Several network layer metrics are monitored [15]. Among those, in this paper we rely on the passive measurements of *i*) end-to-end path delay between peers (e.g., RTT, Delay Jitter), and *ii*) packet loss rate.

¹available at <http://www.peerstreamer.org>

²<http://napa-wine.eu>

³<http://www.ffmpeg.org>

B. Overlay Management

The approach for building the overlay topology in WineStreamer 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 chunk to p (p parents); $\mathcal{N}_O(p)$ collects all peers that can receive chunks from p (p children). 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 the union of all the in-neighborhoods

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

where \mathcal{S} is the set of all the peers in the swarm⁴.

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} along with their properties; WineStreamer implements a variation of Newscast [?] for this function. The *neighborhood manager* realizes the task of filtering the peers that are deemed the 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 WineStreamer every peer p selects other peers as in-neighbors and establishes a management contact with them. Thus each peer p actively selects parents 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 children. 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 parents. 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 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 parents 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 parent. 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 children than peers with little or no upload capacity.

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

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

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 $\mathcal{N}_I(p)$ dropping part of the old in-neighbors while adding fresh new parents. Two 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). If not otherwise stated $N_I = 30$, $T_{\text{up}} = 10$ s and $F_{\text{up}} = 0.3$. The latter two values result in a good compromise between adaptiveness and overhead. Their choice is robust and we omit sensitivity analysis due to lack of space.

A. Metrics driving the neighborhood selection

At every update, $\mathcal{N}_I(p)$ is the result of two separate filtering function: one that selects the peers to drop, and another one selecting parents to add. For these filtering function we essentially 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 rate⁶ or number of received chunks from a parent.

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 parent 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

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

where m_q is the metric of q and \mathcal{N}_S is either \mathcal{N}_I for drop filtering or the set of candidate parents for add filtering.

B. Add Filters

We consider the following four criteria to add new parents:

- RND**: Candidate neighbors are selected uniformly at random. $\forall q, m_q = 1$;
- BW**: Candidate neighbors are weighted according to their upload bandwidth C_q . $\forall q, m_q = C_q$;
- RTT**: Candidate 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, $w_q = 1s^7$.
- OFF**: Candidate neighbors are weighted according to the rate they send offer messages R_q . $\forall q, m_q = R_q$;

⁶HRC 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.

C. Drop Filters

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

RND: Neighboring peers are dropped uniformly at random. $\forall q, m_q = 1$;

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

RXC: Neighboring peers 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 essentially assigns a quality index related to the parent ability to successfully transfer chunks to p .

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 200ms.

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. VI report results for different resulting combinations. Blacklisting can be applied (or not) to all of them, and its impact will be studied selectively. We tested also other metrics and combinations, but we only report interesting ones. RND-RND is used as a benchmark, as it is a policy based on pure sampling of the swarm.

E. Graph theory considerations

Depending on the selection strategy of $\mathcal{N}_I(p)$, \mathcal{T}_S as defined in (1) has different characteristics. Let $N_I(p) = |\mathcal{N}_I(p)|$, if every peer selects in-neighbors at random and $N_I(p)$ is binomially distributed, then \mathcal{T}_S is an Erdős–Rényi graph. If instead $N_I(p)$ is constant, then \mathcal{T}_S is an n -regular (directed) graph with $n = N_I$. RND-RND falls in this category, provided the peer sampler provides the neighborhood manager with a stochastically unbiased sample of the set of peers S .

Notice that also in case $N_I(p) = k \forall p$, which is the case we consider in this work, $N_O(p)$ is in any case binomially distributed by construction, and this property is exploited by the filtering function of peers to provide a topology \mathcal{T}_S whose characteristics are different from an n -regular (undirected) graph and are exploited to obtain globally good performance.

Actually, the filtering functions provide constraints for the evolution of the topology; formally, this means that evolution of \mathcal{T}_S falls in the category of *constrained random graphs* processes. This rather simple discussion leads to the observation that the performance of the constraining rules can be computed as rewards on a stochastic chain whose state is the graph itself [?], which guarantees that experimental measures are correct and meaningful, in that all the instances of the

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 AREAS OF PEERS.

	1	2	3	4
1	20 ± 10%	80 ± 10%	120 ± 10%	160 ± 10%
2	80 ± 10%	20 ± 10%	140 ± 10%	240 ± 10%
3	120 ± 10%	170 ± 10%	20 ± 10%	200 ± 10%
4	160 ± 10%	240 ± 10%	200 ± 10%	20 ± 10%

topology with similar characteristics will yield similar rewards and performance.

IV. TESTBED CONFIGURATION

Following the scientific approach, 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 fluctuation and randomness due to external impairment. The testbed is built in labs available at Politecnico di Torino with 204 PCs divided in four different subnets. Table ?? shows the number of PCs available in each subnet. We used `tc`, the standard Linux Traffic Controller tool, together with the `netem` option to setup RTTs among subnets and packet dropping probability when needed. The RTT distribution is described in Table ???. 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. Each PC is running 5 peers (5 independent instances of WineStreamer) simultaneously, thus, a swarm of 1020 peers is considered in every experiment. The source peer run in 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 (see below).

The well known *Pink of the Aerosmith* video sequence has been used as benchmark. The nominal sequence length corresponds to 200s, its spatial resolution is 352×240 pixels, while the time resolution is 25 frame/s. The sequence is looped for a total stream duration of about 15 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 for encoding 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 Mb/s if not otherwise specified. This corresponds to a system load $\rho = 0.9$ – defined as $\rho = r_s/E[C_p]$ where

Table III
CHARACTERISTICS OF PEER CLASSES.

Class	Bandwidth	Percentage of Peers
1	5 Mb/s ± 10%	10 %
2	1.6 Mb/s ± 10%	35 %
3	0.64 Mb/s ± 10%	35 %
4	0.2 Mb/s, ± 10%	20 %

$E[C_p] = 1.324\text{Mbit/s}$ is the average upload bandwidth of peers.

The source node generates a new chunk at regular time, i.e., every new frame, also enabling a stricter real-time streaming. The chunk size is instead highly variable. Each peer implements a chunk buffer of 150 chunks. Given the one-frame \leftrightarrow one-chunk mapping, and 25 fps of the video, this corresponds to a buffer of 6s, i.e., the playout deadline is only 6s.

A. Network Scenarios

The general networking scenario sketched above is declined in three different flavors that allow the exploration of different 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 areas, so that there is the same distribution of bandwidth everywhere. This scenario is useful to understand the fundamental behavior of different neighborhood filtering strategies.

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 and reduce the network footprint of the application.

The third and final scenario, $Lossy$ 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 only 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.

V. PERFORMANCE EVALUATION

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 [?]. Given the highly structured organization of the video streams, the degradation of the received video quality become typically noticeable for values of $F_{loss}(p)$ higher than 1%, while loss probability of 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

$$\zeta(p) = \frac{\sum_q RTT_q(p)b_q(p)}{\sum_q b_q(p)} \quad (2)$$

while the average network cost is $\zeta = E_p[\zeta(p)]$

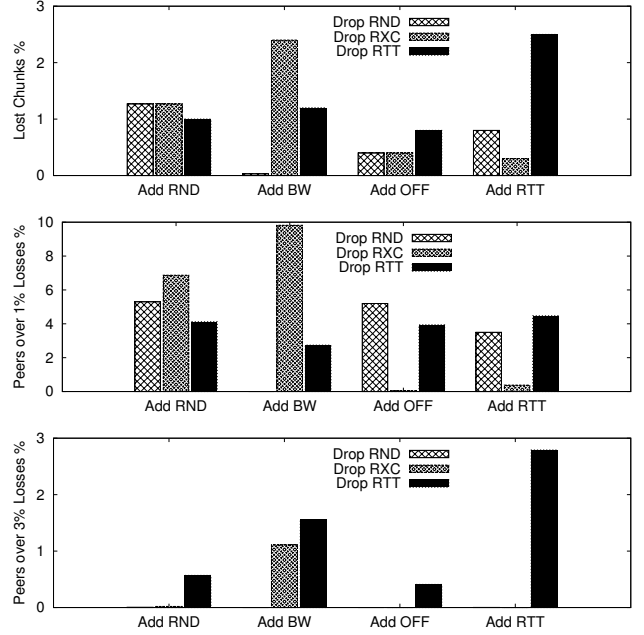


Figure 3. Frame loss for different strategies in G_Homo scenario: F_{loss} (average) (top), percentage of peers whose $F_{loss}(p) > 0.01$ (center), percentage of peers whose $F_{loss}(p) > 0.03$ (bottom).

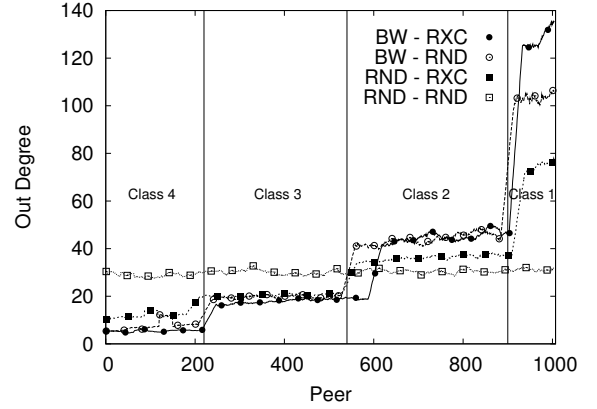


Figure 4. Out-degree distribution of peers, G_Homo scenario.

A. G_Homo scenario

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

The top plot in Fig. 3 shows the average frame loss probability experienced by different policies, while center and bottom plots report, respectively, the percentages of peers that experienced $F_{loss}(p) > 0.01$ and $F_{loss}(p) > 0.03$.

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) appears to be the more effective in reducing the losses. However the behavior of BW-RXC for which F_{loss} tops to 2.5% indicates that using a single metric for selecting the neighborhood can be dangerous. 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. 6 reports the smoothed⁸ 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 again too aggressive in strictly selecting the closest parents. Indeed, as already observed in [?], policies that force a too strict localization of traffic may induce significant performance degradations due to poor topological properties of the swarm. To complement previous information Fig. 7 reports the Cumulative Density Function (CDF) of network cost $\zeta(p)$. As expected, RTT aware policies tends to significantly reduce this index thanks to their ability to select parents within the same area.

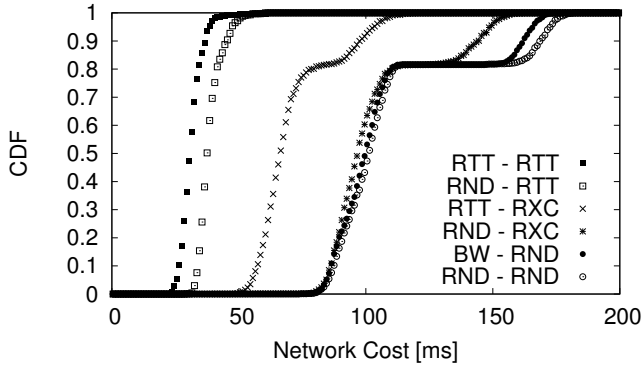


Figure 5. CDF of the distance traveled by information units, G_Homo scenario.

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; iii) the preference toward high bandwidth peers/nearby peers must be tempered to achieve good performance. For instance a policy like RTT-RXC improves quality and reduces the network cost at the same time.

⁸The 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.

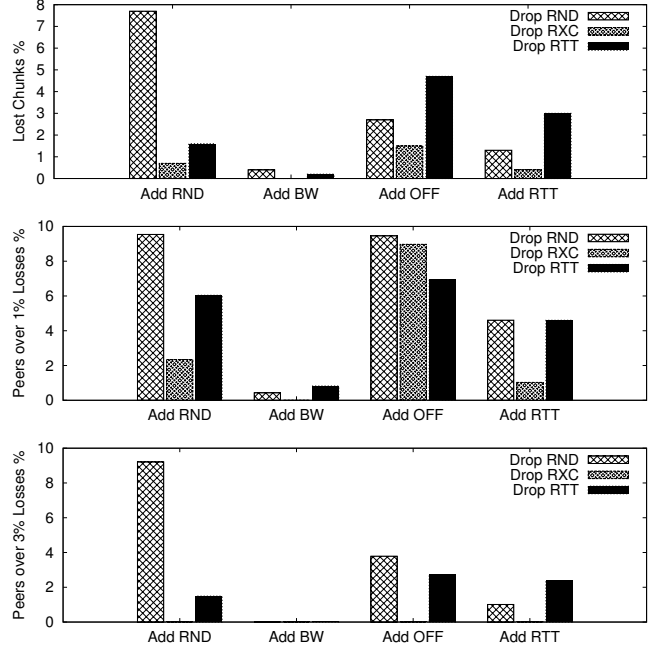


Figure 6. Frame loss for different strategies in G_Homo scenario with $N_I = 20$: F_{loss} (average) (top), percentage of peers whose $F_{loss}(p) > 0.01$ (center), percentage of peers whose $F_{loss}(p) > 0.03$ (bottom).

Next, we consider the same network scenario but we set $N_I = 20$. This is a more critical situation and we expect that choosing the good parents is more important and expect RND policies to suffer more. Notice that the value of N_I is related to the total number of peers, so that for actual global broadcasting to millions of peers having a policy that performs well with a small N_I is very important, as the signaling overhead increases with N_I .

Results are plotted in Fig. 8 (the y-scales in Figs. 3 and 8 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. 7 and are not reported for the sake of brevity.

Random selection policies, which are widely employed by the community, are robust, but perform well only if the number of peers in the neighborhood is fairly large, which, for live streaming applications, implies a large overhead for signaling. As already seen with $N_I = 30$, also in this case the policy that combines bandwidth and RTT awarenesses (RTT-RXC) definitely improves both performance and network costs.

B. G_{Bias} scenario

Maintaining unchanged the C_p distribution, we localize all Class 1 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 other geographical regions than 1.

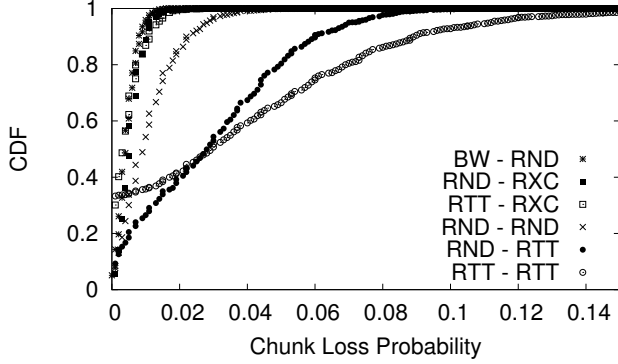


Figure 7. CDF of the frame loss probability for four different strategies, G_{Bias} scenario.

Fig. 11 reports the CDF of $F_{loss}(p)$ for the five 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. However, observe that strategy RTT-RXC performs as well as RND-RXC and BW-RND, but it can also reduce the network cost, as shown in Fig. 12 that reports the CDF of $\zeta(p)$.

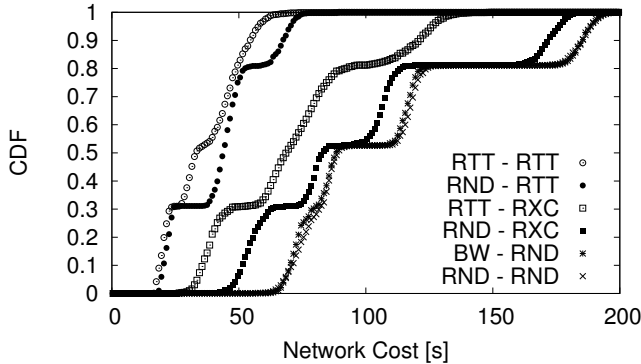


Figure 8. CDF of distance traveled by information units, G_{Bias} scenario.

This result essentially proves that also in G_{Bias} conditions 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 bandwidth awareness.

C. Lossy scenario

We consider another challenging scenario in which large bandwidth peers are uniformly distributed over the four sub-

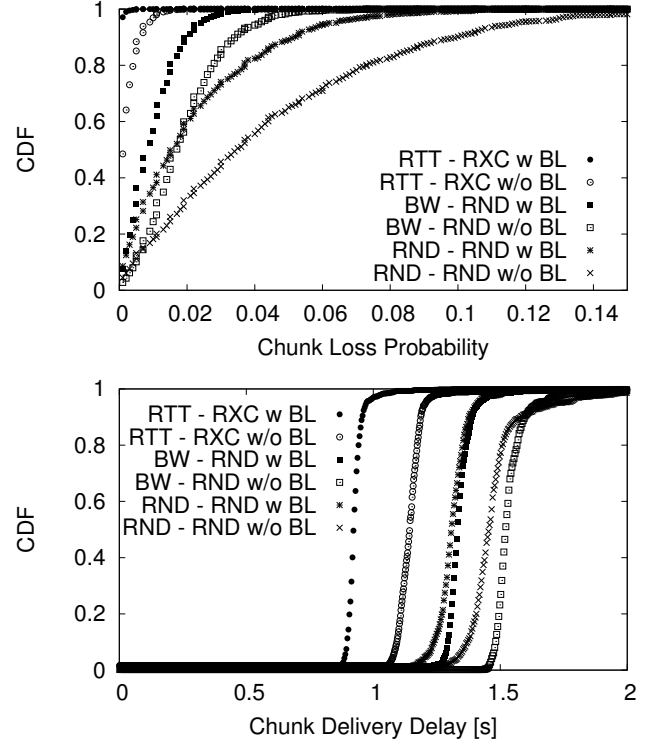


Figure 9. CDF of chunk loss probability (top) and CDF of chunk delivery delays (bottom) for three different strategies with and without adopting blacklist mechanism in $Lossy$ scenario.

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 - RXCH w/o BL	0.12	0.68	0.11	0.07
RTT - RXCH w BL	0.13	0.70	0.09	0.05

nets, but packet losses are present in some long haul connections. In this case, we expect that blacklisting can play a significant role to avoid selecting lossy paths. Indeed, exploiting the blacklist mechanism every peer should identify and abandon poorly performing parents, biasing the neighborhood toward good performing peers. This effect should reinforce policies that naturally bias the selection of neighbor peers employing peer quality. We only plot the results for RND-RND, BW-RND and RTT-RXC as these latter have emerged as the most promising criteria and RND-RND is the benchmark. The performance of other policies did not show any really interesting behavior in this scenario.

Fig 13 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.

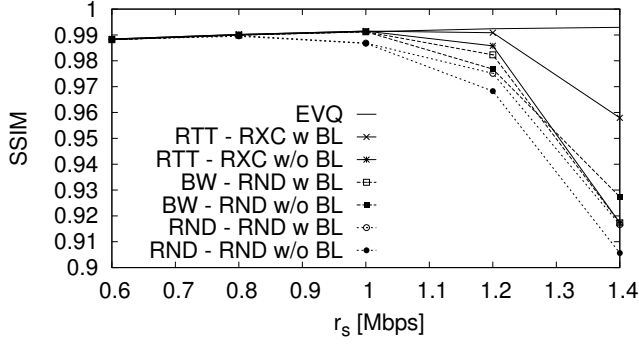


Figure 10. S_{ssim} index when varying video rate r_s in *Lossy* scenario.

Benefits of the blacklisting mechanism are confirmed by Table ?? that reports the normalized volume of incoming traffic for peers in cluster 2 from peers in all clusters. Keeping in mind that in *Lossy* scenario peers belonging to cluster 2 experience lossy paths from/towards peers in cluster 3 and 4 (as explained in Sec. V), it is easy to see that volumes of incoming traffic from cluster 3 and 4 are nicely reduced thanks to blacklisting mechanism.

D. 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 *Lossy* scenario, and we increase the video rate from 0.6 Mb/s to 1.4 Mb/s. Recall that $E[C_p] = 1.324$ Mb/s.

Fig. ?? shows S_{ssim} considering RND-RND, BW-RND and RTT-RXC with and without blacklisting. S_{ssim} is a measure of the distortion of the received image compared against the original source (before encoding and chunkization). 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 0.95 are typically considered of good quality. S_{ssim} has been computed considering video frames received by 200 peers (50 for each class), and then averaging among all of them. The initial 12 min of the video have been discarded to focus on steady state performance. The S_{ssim} is computed for 1 min of the video given the enormous computational burden of this task.

The EVQ (Encoded Video Quality) curve in the plot is the reference value for the encoding level and it obviously increases steadily as r_s increases. When the system load $\rho < 1$, S_{ssim} increases for increasing r_s thanks to the higher quality of the encoded video. As soon as the system is overloaded, the S_{ssim} rapidly drops due to missing chunks which impair the quality of the received video. Notice how RTT-RXC scheme outperforms RND-RND and BW-RND for every value of r_s . Fig. ?? also shows the benefits of the blacklist mechanism for every scheme.

VI. RELATED WORK

Many popular commercial applications such as *PPLive*, *SopCast*, *Octoshape* were proposed in recent years, but their creators did not publish any information about their internal implementation, making impossible any statement about their overlay topology design strategies. Focusing on available literature on purely mesh-based P2P-TV streaming 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. An early solution called *GnuStream* was presented in [1]. Based on *Gnutella* overlay, *GnuStream* implemented a simple load distribution mechanism: peers were expected to contribute to chunks dissemination in a way proportional to their current capabilities. A more refined solution called *PROMISE* was introduced by Hefeeda et al. in [2]. Even if *PROMISE* 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*[3] or *CAN* - some naive location awareness based on number of IP hops. *DONet* (or *Coolstreaming*) [4] is a successful P2P streaming system implementation. This system employs a scheduling policy based on chunk rarity and available bandwidth of peers, but its data-driven overlay topology does not exploit any information coming from underlying network levels. In [5] the brand new implementation of *Coolstreaming* application is presented. Many new features were introduced to improve the streaming service and, in particular, authors proposed a new neighbor re-selection heuristic based only on peers uplink bandwidth. In [6], authors showed the design aspects related to 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 [7] authors presented a study about some key design issues related to mesh-based P2P-TV systems. In particular they focused their attention on understanding what are 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 and maintenance, in [8] the problem of building an efficient overlay topology, taking into account both latency and bandwidth, has been formulated as an optimization problem; however, the complex interactions between overlay topology structure and the chunk distribution process are completely ignored in [8], where continuous streams of information are distributed (in a purely push fashion) among peers.

In [9] a theoretical investigation of optimal topologies is formulated, considering latency and peer bandwidth hetero-

geneity; scaling laws are thus discussed. In [10], a distributed and adaptive algorithm for the optimization of the overlay topology in heterogeneous environments has been proposed, but network latencies are still ignored. The authors of [11] propose a mechanism to build a tree structure on which information is pushed. They combine two ideas: good topological properties are guaranteed by means of prefixes based on peers identifiers (similarly to what is done in other structured P2P systems) and latency awareness is used to select a specific peer between those with the same prefix. 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, [12] shows with simple stochastic/fluid models that overlay topologies with small-world properties can be effectively exploited to support chunk distribution in P2P-TV systems.

Finally, it is worth to mention [14], where the authors experimentally compare the performance of unstructured systems and structured, multiple-tree based systems. Results in [14], indicates that unstructured systems tends to outperform tree-based systems in highly dynamic scenarios as well as in scenarios characterized by bandwidth limitations, which strengthen our choice of exploring topology management policies for mesh-based streaming systems.

VII. 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 testbeds.

The work presented in this paper aims at filling this gap. Leveraging the WineStreamer application developed within the framework of the EU NAPA-WINE project, we have 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 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.

Finally, WineStreamer is a fully fledged P2P-TV application that we offer as open-source to the community to render results reproducible and to allow further research.

ACKNOWLEDGMENT

This work was partially supported by the European Commission under the FP7 STREP Project “Network-Aware P2P-TV Application over Wise Networks” (NAPA-WINE).

REFERENCES

- [1] X. Jiang, Y. Dong, D. Xu, and B. Bhargava, “Gnustream: a p2p media streaming system prototype,” in *Proceedings of the 2003 International Conference on Multimedia and Expo - Volume 1*, ser. ICME '03. Washington, DC, USA: IEEE Computer Society, 2003, pp. 325–328. [Online]. Available: <http://portal.acm.org/citation.cfm?id=1153922.1154349>
- [2] M. Hefeeda, A. Habib, B. Botev, D. Xu, and B. Bhargava, “Promise : Peer-to-peer media streaming using collectcast,” *Area*, p. 45, 2003. [Online]. Available: <http://portal.acm.org/citation.cfm?doid=957013.957022>
- [3] 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. [Online]. Available: <http://doi.acm.org/10.1145/964723.383071>
- [4] 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.
- [5] 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*, 2008, pp. 1031–1039.
- [6] X. Liao, H. Jin, Y. Liu, L. M. Ni, and D. Deng, “Anysee: Peer-to-peer live streaming,” in *IEEE INFOCOM*, 2006.
- [7] N. Magharei and R. Rejaie, “Prime: Peer-to-peer receiver-driven mesh-based streaming,” in *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*, may 2007, pp. 1415–1423.
- [8] 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, Arizona, US, April 2008.
- [9] 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.
- [10] 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, Virginia, US, June 2009.
- [11] T. Locher, R. Meier, S. Schmid, and R. Wattenhofer, “Push-to-pull peer-to-peer live streaming,” in *Lecture notes in computer science*, Berlin, DE, 2007.
- [12] 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.
- [13] 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 Transactions on Parallel and Distributed Systems*, vol. 22, no. 3, pp. 451–463, March 2011.
- [14] J. Seibert, D. Zage, F. S., Nita-rotaru, and C., “Experimental comparison of peer-to-peer streaming overlays: An application perspective,” in *33rd IEEE Conference on Local Computer Networks, Proceedings*, 2008, pp. 1–6.
- [15] 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 Communication Magazine*, vol. 49, June 2011. [Online]. Available: http://www.tlc-networks.polito.it/mellia/papers/Architecture_Napa-Wine.pdf
- [16] L. Abeni, C. Kiraly, A. Russo, M. Biazzi, 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, 2010*, Florence, Italy, October 2010, pp. 1–6.
- [17] R. Birke, C. Kiraly, E. Leonardi, M. Mellia, M. Meo, and S. Traverso, “Hose Rate Control for P2P Streaming Systems,” in *Proceedings of the 11th International Conference on Peer-to-Peer Computing 2011 (P2P'11)*, Kyoto, JP, August 2011, conference.
- [18] F. Dabek, R. Cox, F. Kaashoek, and R. Morris, “Vivaldi: a decentralized network coordinate system,” in *Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*, ser. SIGCOMM '04. New York, NY, USA: ACM, 2004, pp. 15–26. [Online]. Available: <http://doi.acm.org/10.1145/1015467.1015471>
- [19] S. Lee, Z.-L. Zhang, S. Sahu, and D. Saha, “On suitability of euclidean

embedding for host-based network coordinate systems;" *Networking, IEEE/ACM Transactions on*, vol. 18, no. 1, pp. 27–40, feb. 2010.