Architecture of a Network-Aware P2P-TV Application: the NAPA-WINE Approach

R. Birke, E. Leonardi, M. Mellia, [1] DELEN – Politecnico di Torino,
A. Bakay, T. Szemethy, Netvisor Ltd.,
C. Kiraly, R. Lo Cigno, DISI – Università di Trento
F. Mathieu, L. Muscariello, Orange Labs,
S. Niccolini, J. Seedorf, NEC Laboratories Europe,
G. Tropea, Lightcomm SRL

October 1, 2010

Abstract

Peer to Peer streaming (P2P-TV) applications have recently emerged as cheap and efficient solutions to provide real time streaming services over the Internet. For the sake of simplicity, typical P2P-TV systems are designed and optimized following a pure layered approach, thus ignoring the effect of design choices on the underlying transport network. This simple approach, however, may constitute a threat for the network providers, due to the congestion that P2P-TV traffic can potentially generate. In this paper, we present and discuss the architecture of an innovative, network cooperative P2P-TV application that is being designed and developed within the STREP Project NAPA-WINE[1]. Our application is explicitly targeted to favor cooperation between the application and the transport network layer.

[1]NAPA-WINE (http://napa-wine.eu) is a 3 year STREP project started on February 1-st 2008 and supported by the European Commission within Framework Programme 7 (ICT Call 1 FP7-ICT-2007-1, 1.5 Networked Media, grant No. 214412). This paper reflects its evolution after about two years.

1 Introduction

Peer-to-peer (P2P) technology has been recently exploited to offer video service over IP (P2P-TV), as for example done by PPLive, SopCast, TVAnts, CoolStreming/UUSee, and TVUplayer to name a few commercial systems. Recently, the interest of the research community has started to raise thanks to the opportunities P2P-TV systems offer [1]. In this context NAPA-WINE (Network Aware Peer-to-peer Application over WIse NEtwork), a three years project (STREP) within the 7-th Research Framework of the European Commission, focuses on improving the performance of P2P live streaming distribution, while protecting network resources from excessive usage. In a traditional P2P-TV system, a video source chops the video stream into chunks of data, which are then exchanged among nodes distributing them to all participating peers. Peers form an overlay topology at the application layer, where neighbor peers are connected by virtual links, over which chunks are exchanged using the underlying IP network. Two networking layers can be identified: the overlay layer in which peers exchange content, and the IP layer, in which routers and links transfer the packets. The main goal of NAPA-WINE is the study of novel Peer-to-Peer architectures and systems suitable for High Quality TV live streaming over the Internet: a *P2P-HQTV* system [2]. NAPA-WINE focuses on overcoming the limitations of today's approaches, where the two layers are completely independent and at times antagonists. Instead, we envision a cooperative paradigm in which the application and network layers interoperate to optimize the service offered to end users, as depicted in Fig. 1.

We believe that an approach where the overlay and the IP network ignore each other, while acceptable for elastic applications or low-bandwidth streaming applications, may cause intolerable performance degradations for high-quality real-time (even for soft real-time) applications such as P2P-TV: If the network gets heavily congested, the application will never be able to meet its minimum requirements. As a consequence, the only successful paradigm for a large scale P2P-TV architecture is a cooperative paradigm, where the network and the application join forces to meet the quality of service requirements, and to reach the largest possible population of users.

Finally, there is an additional push to such cooperation. In traditional P2P systems, only two main actors can be identified: the network providers and the application users; and their interests are often in contrast. In the case of P2P-TV, a new main actor comes into play: the content producer, whose interests are mainly in distributing its contents through the Internet in the safest possible way, thus avoiding every possible problem, either technical or legal, which could induce users to migrate away. This is a major shift with respect to today P2P applications, where users provide contents, a shift that provides further incentives for network operators to cooperate with the content providers and the users to successfully deliver the video. Note that cooperation between network providers and P2P applications has been already investigated in the context of file sharing P2P applications (see for example the pioneering work [3]), but it has never been applied in the context of P2P-TV system design and performance evaluation.

In this context, the NAPA-WINE project is proposing an innovative network cooperative P2P-TV architecture that explicitly targets the optimization of the quality perceived by the users while minimizing the impact on the underlying transport network. The focus is on the study, design and development of a P2P-HQTV system, in which peers setup a *generic mesh* based overlay topology, over which video *chunks* are exchanged according to a swarming-like approach. A source peer produces the video stream, chops it into chunks, and injects them in the overlay where peers cooperate to distribute them.

The architecture we envision is schematically represented in the Fig 1. An important element of the cooperative P2P-HQTV system is represented by the built-in distributed monitoring tool that allows the application to continuously gather real time information both on network conditions and users' perceived performance. Information collected by the monitoring tool can be used to trigger reconfiguration algorithms acting both at the level of the chunk distribution mechanism (scheduling) and at level of the overlay topology reconfiguration. In addition, potentially useful information on the system state can be also exported to the network, so that it is aware of the status of the P2P-HQTV system.

The architecture we propose also allows the network layer to expose useful information to the application layer. As pursued by the IETF in the ALTO [6] working group with the contribution of NAPA-WINE partners, the network provider is given the capability to guide the P2P application, for example by explicitly publishing information about the status of its network, like link congestion or AS routing preferences.

2 The High Level Architecture

This section describes the main functions and requirements of a P2P-TV system, proposing the high-level architecture for the NAPA-WINE system, whose block diagram is depicted in Fig. 2.

2.1 User Module

The “user module” (blue block in Fig. 2) implements the interface between the user and the application. Besides the Graphical User Interface (GUI), it is responsible of video coding and decoding. If the considered peer is the source node, the input video signal has to be converted into a sequence of chunks, which are the atomic unit of data that will be exchanged in the P2P overlay. Depending on the type of video source this may include analog/digital conversion, encoding, adding forward error correction (FEC) information, etc. User module supports any available CODEC thanks to a standard interface. Furthermore, it implements flexible chuncking mechanism which can explicitly consider the nature of the video, e.g., forming chunks considering frame boundaries and types. When the peer acts as receiver only, the user module reassembles the audio and video stream from received chunks, so that, after decoding and resynchronization, the video can be displayed.

The user module runs also video quality monitoring algorithms constantly monitoring the quality of experience the user is perceiving. Finally, several instances of the user module can run at the same time, allowing to watch a channel while recording another one, or maintaining pre-views of other channels with reduced quality.

2.2 Scheduler Module

The “scheduler module” (the yellow block in Fig. 2) is the core of the information distribution engine. It is responsible for receiving and sending chunks, both from/to other peers via the network, and to/from the local “user modules”. Different instances of this module might run at the same time, each taking part in one overlay serving one channel. Cross-channel information diffusion is under consideration and may help to optimize service and channel switching, but is not under development so far.

The scheduler module hosts the chunk and peer scheduling algorithms, which select both what should be offered (or requested) and to (from) whom. The decision is based on distributed information —a local database of the status of other peers stored in the active peers database. Options for designing the scheduler and the impact on overall system performance are detailed Sect. 4.

2.3 Overlay Module

We recall that in unstructured systems peers form a generic highly mesh topology, this guarantees good resilience to churn and great flexibility.

The "overlay module" (the top-right red block in Fig. 2) selects and updates the peer *Neighborhood*, i.e., the set of peers the local peer exchanges information with. A fairly large number of neighboring peers are maintained to form a well connected overlay structure, as peers may leave an overlay at any time, e. g., when the user switches channel. Discovering new peers and establishing connections to them, as well as forwarding information about them to other peers is the task of the overlay module.

The selection of peers may be based on the information stored in the repositories, which, as detailed later, store information about both peers and the transport network. The overlay and the scheduler modules are interdependent, and for this reason they are detailed together in Sect. 4.

2.4 Monitoring Module

Both the chunk scheduler and the overlay manager can greatly benefit of information about the quality of the connectivity with other peers. This includes, but is not limited to, the distance and the available bandwidth between two peers, or the presence of Network Address Translation (NAT). The "monitoring module" (the green block in Fig. 2) gathers this kind of information. It basically has two modes of operation: Passive measurements are performed by observing the messages that are exchanged anyway between two peers, e. g., when exchanging video chunks or signaling information; Active measurements, in contrast, craft special probe messages which are sent to other peers at the discretion of the monitoring module. The design of this monitoring module is one of the most innovative goals of NAPA-WINE and it will be documented in Sec. 3 in detail.

2.5 Repositories and Repository Controller

"Repositories" (in the center of Fig. 2) are databases where information about each peer is shared with all other peers. Indeed, the information generated by the monitoring module is not only useful for the local peer, but also for other peers that can benefit from it. Therefore, mechanisms for exchanging measurement values with other peers are useful. Furthermore, it may be useful to have access to information from entities that are not part of the P2P overlay, such as the so-called "network oracles”, through which, for example, network providers can share information about the status of the network.

The information acquired by each peer is locally stored and readily available, and it is summarized and exported (published) to the (logically) central repository. The contents of the repository, and data import and export are managed by the repository controller which implements a set of API to push and fetch information from the repository database.

2.6 Messaging Layer

The “messaging layer” (the orange block in Fig. 2) offers primitives to the other modules for sending and receiving data to/from other peers. It abstract the access to transport (UDP/TCP) protocols and the corresponding service access points offered by the operating system by extending their capabilities and providing an overlay level addressing, i.e., assigning a unique identifier to each peer. For example, it provides the ability to send a chunk of data to another peer, which has to be segmented and then reassembled to fit into UDP segments. The messaging layer also provides an interface to the monitoring module, which is used for passive measurements: whenever a message is sent or received an indication will be given to the monitoring module, which can then update its statistics.

Another important feature of the messaging layer is the presence of mechanisms for the traversal of NAT boxes. Network Address Translation allows to attach several computers to the Internet using only one public IP address. Therefore it is very popular with private customers, who are also the main target audience for P2P TV. However, the presence of a NAT device may prevent peers from establishing connections to other peers, therefore, special NAT traversal functions are offered by the messaging layer.

3 The monitoring Module and Repositories

The monitoring module is conceived to collect and share information useful for the construction and maintenance of the P2P overlay topology and for the scheduling process of video chunks. The main objective is to maintain up-to-date information on the quality of peers end-to-end path, and to infer information on the network’s available resources. Repositories, both local and global, are used to store metrics, estimates, and measured data requiring additional elaboration. Furthermore, a certain number of functions act on these data to produce up-to-date estimates of various parameters that can be published on global repositories through an interface named "Repository Controller". Particular care is taken to optimize both the computational and the memory requirement of the previously mentioned functionalities. The repository controller limits the amount of information that is exchanged among peers through the network to avoid waste of bandwidth. It also verifies that a peer can access the information it is requesting. Indeed, each repository stores data that can remain private and local to the peer, or information that is made publicly available. The distinction between public and private information could be based on security considerations, and also on its usefulness and the cost of its dissemination. For example, the ISP can share information about the status of its network only with clients connected to its network.

In the following, we detail the functionalities implemented in the monitoring module.

3.1 Monitoring components

Measurements are available at the chunk and segment levels, i.e., above and below the messaging layer. Several network layer metrics can be monitored: i) delay between peers (e.g., Round Trip Times (RTT), Delay Jitter), ii) loss probability, iii) path capacity and available bandwidth, iv) number of hops traversed. Other more sophisticated measures are under study. Thanks to a simple and modular architecture, measurements can be added as (compile-time) plugins, and activated on demand. The monitoring layer is implemented at every peer, and information it collects is made available to the all peers. Table 1 summarizes the network layer measurements that are currently available in the monitoring module.

The monitoring module is build of three main components.

3.1.1 The Passive Measurements Component (PaM)

implements the set of measurement functionalities needed to passively monitor exchanges between peers. This function is passive as it relies on peers' communications to reduce as much as possible any overhead. Each measure is initiated by the sender, and the receiver cooperates to complete it, e.g., sending back acknowledgment messages. Results of each measurement are then fed to the sender Monitoring Controller, which possibly elaborate them before pushing results in the P-REP (Peer-Repository) and N-Rep (Network-Repository).

3.1.2 The Active Measurement Component (AcM)

performs active measurements, i.e., it can inject pure measurement messages and data that do not carry any user information. This function possibly runs a number of different measurement probes, either periodically or on request. For example, it may be invoked to estimate the end-to-end status between the local peer and a remote peer to which no chunk is being sent. Therefore, AcM may be also seen as a bootstrap function when limited knowledge of the peers and the network is stored in the P-REP. Active measurement results in a waste of bandwidth, and therefore care must be taken when activating them.

3.1.3 The Monitoring Controller (MON-Controller)

is in charge of managing all the measurements performed by the PaM and the AcM. It implements the algorithms to decide when to trigger a particular measurements, and to process the results of each end-to-end measurements. For example, the MON-Controller can evaluate average, standard deviation and confidence intervals of a given index; it can identify and possibly discard wrong samples, etc. Considering the global network knowledge, it is supposed to implement the algorithms that infer the network status, e.g., by implementing some network tomography or virtual coordinate systems that are then stored in the N-REP. It is also responsible to push locally cached measurement results into the centralized repository, by summarizing them to avoid excessive traffic overhead. The communication of MON-Controller with the repositories is granted by the Rep-Controller.

3.2 Monitoring Example

Fig. 3 provides a simple example of measurements obtained with the current implementation of the monitoring module. It shows the measurements of the Capacity and of Available Bandwidth (top plot), and the measurements of Round Trip Time and of Loss Probability (bottom plot). A controlled test bed has been used, in which 10Mbps Ethernet link connects a source peer to a receiver peer, while interfering traffic is injected. In particular, a 4Mbps CBR source is active from time 300s to 600s; at time 600s, a 7Mbps CBR source is active up to time 900s. Then from time 1200s and up to time 1500s, TCP connections are started every 60s. From the Figure, it is possible to observe the variation of available bandwidth, the increase of the RTT and the losses induced by interfering traffic.

3.3 Repositories

In the NAPA-WINE architecture, *repositories* are envisioned as global databases, containing information to aid peer selection decisions.

Repositories are key elements in the design to achieve network awareness. They have one additional important role in solving the *bootstrapping problem* of P2P systems: namely, upon startup, a peer needs to obtain a list of neighbor candidates. Repositories are well suited for this role: the peer only needs to have access to a few well-known repository servers in its initial configuration. To increase repository resilience, various approaches (such as database replication and DHT-like technologies) can be studied, which are outside of the NAPA-WINE prototype design scope.

The NAPA-WINE architecture defines three different repositories based on the information they contain:
1) **P-REP.**: The *peer repository* is a distributed repository storing information about the peers' status and end-to-end measurements performed by the peers' monitoring modules, for instance, average bandwidth over different periods of time, round-trip time, download/upload success rates etc. Each peer measures and stores locally these quantities, for instance as an $N \times N$ matrix, N being the number of neighbor peers (those discovered and selected by the topology manager). Measures can be exported in an aggregated form to a global (distributed or centralized) database, the global P-REP, for performance monitoring and statistical analysis, possibly including information on obtained quality, visited channels, etc. A centralized prototype has been developed and is used to display on a web-service the swarm topology, its characteristics, and how different architectural choices affect the P2P-TV application.
2) **N-REP.** *Network repositories* store network-wide information inferred from P-REP values. The information stored by P-REP is hardly complete about all possible N^2 peers end-to-end paths. Indeed, each peer typically collects measurement over a subset of all peers. In particular, when joining a given channel, the new peer has little or no information on the status of other peers, and of the end-to-end path characteristics. To solve this problem, "virtual coordinate" [4] systems and network tomography [5] have been studied in the research community, whose goal is to map peers to nodes into a virtual space, in which distances among nodes reflect the actual distances in the real system, or to infer network properties from a subset of measurements. Both allow, for example, to "predict" the quality of an end-to-end path which has never been tested in the past. The N-REP stores results these systems generate. The topology manager uses this information together with

distributed algorithms like Tman to achieve the desired topological properties.
3) **E-REP.** The *external repository* is dedicated to store *out-of-the-box*, semi-static information on the network. It is conceived for network operators that can fill it with non public information, e.g., AS graph and peering points, the ranking of routing paths, routing tables, information on the routing optimization and the network topology. It can also store intra-domain information of one or multiple AS. Clearly, access to this information must be protected and granted only to a subset of peers, e.g., to only clients in the AS of the ISP. Therefore, the E-REP runs on dedicated equipments installed and managed directly by the network operator or content provider. For example, in the ideal case, the E-REP stores the full knowledge on the AS network; the whole ISP topology is therefore exposed to the P2P application.

The E-REP also has an interface for connecting to Application Layer Traffic Optimization (ALTO) servers. The goal of an ALTO service —as currently being standardized in the IETF— is to provide applications with information they can use to optimize their peer selection [6]. The kind of information that is meaningful to convey to applications via an out-of-band ALTO service is any information that applications cannot easily obtain themselves (e.g., through measurements) and which changes on a much longer time scale than the instantaneous information used for congestion control on the transport layer. Examples for such information are operator's policies, geographical location or network proximity (e.g., the topological distance between two peers), the transmission costs associated with sending/receiving a certain amount of data to/from a peer, or the remaining amount of traffic allowed by a peer's operator (e.g., in case of quotas or limited flat-rate pricing models). ALTO services can be provided by network operators, third parties (e.g., entities which have collected network information or have arrangements with network operators that enable them to learn such information), or even by user communities. The ALTO-interface gives such parties a concrete method for making this kind of information available to the E-REP: The E-REP can use the standard interface of ALTO servers, providing such ALTO-information to NAPA-WINE peers.
4) **Rep-Controller.** The peer accesses repositories via its *Repository Controller (Rep-Controller)* modules. This software component implements the repository protocol and exposes it over a client API. A single Rep-Controller is able to communicate to multiple repository servers concurrently, and has advanced services like data caching and batch publishing of measurement data (to save precious bandwidth).

The repository access protocol is designed for the following basic use cases.

A) Publish measurement results made by peers' monitoring modules (or N-REP processors) into the global database. This is a simple operation, publishing an $(originator, peer_1, peer_2, measurement_{ID}, value)$ record. $originator$ is typically a peer (whose Monitoring Module has generated the data), equal to $peer_1$; $peer_2$ is non null only for peer-pair measurements.

B) Retrieve the list of "best" Peer IDs according to given criteria. This is the main "retrieval" primitive, designed to support the topology manager in the neighborhood selection and ranking. The operation is specified as

$$(maxPeers, constr_1, \ldots, constr_n, weight_1, \ldots, weight_m)$$

where constraints are specified on measurement values, and weights form a utility function the result list is ordered by. This allows formulating queries such as "return a list of at most 10 peer IDs which are closer to $peer_2$ than 5 TTL hops and have smaller round trip time to $peer_3$ than 500ms and sort the list by the peers' own reported upload bandwidth".

C) Retrieve measurement results or inferred values for fine-tuning trading algorithms. This primitive allows the direct query of published (measurement or other) values relevant to a given peer.

D) Retrieve repository meta-data such as the list of $measurement_{ID}$s the repository has records for. This allows client peers to parameterize their peer listing queries accordingly.

Since information stored in the repositories is sensitive, the repository controller enforces security and policies so that only authorized peers can retrieve information. For example, access to the information about ISP topology or AS routing stored in the E-REP is granted only to peers actually connected to that ISP.

4 Scheduler and Overlay Modules

A P2P-TV client needs to communicate very efficiently with other peers to receive and redistribute the huge amount of information embedded in a video stream. In addition, TV being a real time application, information must arrive in short time and with small delay variation. The application goal is then to deliver all the video information to all peers in the system in the smallest possible amount of time. To reach this goal, a distributed system and algorithm must be adopted. The key enabling factors for efficient communication by a peer are: *who* are the peers to communicate with, i.e., its neighboring peers, and *what* data is exchanged within this neighborhood. The first factor drives the overlay management strategy, while the second one dictates the goals of the scheduling algorithms at each peer.

Fig. 4 sketches the relationship between the functions that compose the overlay management, and the functions that compose the overall strategy for offering and searching information within the neighborhood of each NAPA-WINE node. Interfaces toward the repositories, the user module and the messaging layer are also indicated.

The design architecture followed allows flexibility and adaptivity, so that, for instance, chunk and peer selection strategies can be tuned even at run time based on feedback received from other peers either via gossiping or through the repositories.

The Neighborhood database (in green in Fig. 4) is populated as soon as a peer participate in the distribution of a TV channel. For the sake of clarity we describe both scheduling and overlay management as if only one channel is present, but we are also studying the extension to the joint management of multiple channels.

Always referring to Fig. 4, the components in yellow are related to chunk scheduling, transmission and reception, while those in blue refer to topology management and signaling in general (exchange of buffer maps, i.e., the list of chunks available at each peer, availability to service chunks, etc.). The two functionalities interact through the Neighborhood database as well as the chunk buffer (i.e., the structure where chunks are stored for trading and before play-out), and the related buffer maps of neighbors.

4.1 Building and Maintaining Neighborhoods

The overlay network in P2P systems is the result of a distributed algorithm that builds and maintains the neighborhood at each peer. When a peer joins the system, it selects an initial set of neighbors (we call this phase bootstrapping); then the set of neighbors of every node in the system is dynamically optimized over time (overlay maintenance).

The bootstrapping phase is most naturally helped by the Repositories. For the maintenance phase, basing everything only on the Repositories could result in limited scalability and resilience. This is the reason why topology maintenance is performed exploiting information retrieved from the repositories as well as information locally distributed via a gossiping-based mechanism (the latter mechanism will allow the system to work even with Repositories absent).

Culling of neighbors is mainly based on the perceived "quality" of the neighbors; indices that impact this choice are the inter-peer throughput, the measured

RTT, the chunk loss rate, etc. Addition of neighbors is instead based on a mixed strategy of optimization (e.g., adding the most stable and resourceful peers) and randomization to avoid fragile topologies (e.g., group of peers with too few connections toward the rest of the overlay).

Our architecture will make overlay creation and maintenance algorithms much more effective since it offers through the repositories continuously fresh information to the peers about the presence of new resourceful peers. As an example, Fig. 5 shows different overlay topologies obtained using different neighbor selection algorithms: In A), a complete random choice is performed; In B), peers are selected according to the Autonomous Systems they belong to, as reported by the E-REP; In C) peers prefer to selected nodes with large upload bandwidth, as reported by the P-REP; Finally, in D) high upload bandwidth peers within the same AS are preferred.

4.2 Scheduling chunks and peers

Once a topology has been setup, each peer participates in the chunk exchange procedure, with the twin goal of receiving the stream smoothly and in time and cooperate in the distribution procedure. We do not discuss the problems of fairness and security here, because they are not the main focus of NAPA-WINE, and therefore we assume peers are honest and cooperative.

Scheduling the information transfer is probably the single function that most affects performance and network friendliness. This function includes protocol as well as algorithmic problems. First of all, peers need to exchange information about their current status to enable scheduling decisions. This is a requirement in an unstructured system, where the stream flow does not follow strictly the overlay topology (e.g., a tree). The information exchanged refers to the state of the peer with respect to the flow, i.e., a map of which chunks are "owned" by a peer, and which are missing. This task is carried out by the Information Signaling Strategy block in Fig. 4. This block is in charge of i) sending buffer maps to other nodes with the proper timing, ii) receiving them from other nodes and merging the information in the local buffer map data base, iii) negotiating if this and other information should be spread by gossiping protocols or not, and to which depth it should spread in the topology. Besides the buffer map exchange, the signaling includes Offer/Request/Select primitives used to trade chunks. These messages can be piggybacked on chunks for efficiency.

Another key protocol decision is about *Pushing* or *Pulling* information. A chunk is pushed when the peer owning the chunk decides to send it to some other peer, while it is pulled when a peer needing the chunk requests it from another peer. From a theoretical point of view, as shown in [8], pushing is more effective: assuming a synchronous system in which peers coordinate to avoid sending more than one copy of the same chunk, in [8] we have demonstrated the existence of an optimal distributed scheduling that achieves the minimum delivery time to all peers $t_d = \lceil \log_2(N) \rceil + 1$ chunk times, where N is the number of peers, while no such algorithms are known for pulling systems. Practical implementations, however, often prefer a pull based mechanism, because it guarantees that no conflicts arise at the receiver. Other options include mixed Push/Pull strategies [10], and Offer/Select chunk trading [11] that can be associated to both Push and Pull strategies.

Regardless of the protocol and the signaling strategy used, the core of a scheduler is the algorithm to choose the *chunk* to be pulled or pushed and the *peer* to communicate with. During the project we have developed many algorithms and strategies that optimize the performance of the system and that are being implemented in our prototype [7, 8, 9].

For example, Fig. 6 reports the 95th percentile of the delivery delay of all chunks to all peers in an overlay topology in which each peer has 20 neighbors selected at random. 1000 peers are considered (with different upload bandwidth), chunks last 1s, so that propagation delay is negligible compared to chunk transmission time. We report the performance of five schedulers:

RUc/RUp: a random chunk is sent to a random peer that needs it;

RUc/BAp: a random chunk is preferentially sent to high upload bandwidth peers that needs it;

LUc/RUp: the youngest chunk is sent to a random peer that needs it;

DLc/ELp: deadline based algorithm that diffuse the chunk with the smallest deadline (see [8] for details) to a peer that is in the best possible state to further diffuse it.

DLc/BAELp: same as above, but preferentially selecting high upload bandwidth peers.

As it can be seen, more advanced schedulers allow to reduce the chunk delivery time, with improvements up to a factor of 5 at high load. Notice that information provided by the P-REP, such as the peer upload bandwidth, leads to a significant improvement.

Immediate Pushing of chunks may have drawbacks in some scenarios (e.g., when the RTTs is not negligible compared to chunk transmission time) and explicit acceptance of the chunk transfer may guarantee better usage of resources. We have investigated signaling mechanisms where the transmitters and the receivers peers explicitly exchange signaling messages to agree on the chunk to be transferred; the scheme requires the sender to maintain a local queue of already scheduled chunks to seamlessly exploit peer upload bandwidth [11].

5 Conclusions and On-Going Work

TV applications exploiting the P2P communication paradigm are taking momentum and are already a commercial reality. Their overall architecture is however imprinted by file-sharing applications and they operate without any coordination with the IP network, often resulting in poor, even wasteful resource usage, to the detriment of both network and users, and preventing them from the possibility to scale to High Quality (let alone High Definition) TV.

We have presented in this paper the architecture of a P2P-TV system as being developed in the NAPA-WINE project, that is designed with the goal of efficiency and cooperation between the application and both the network operator and the content provider, aiming at optimizing resource usage and minimize the P2P-TV application impacts on the transport network.

The key features are: i) the development of algorithms for topology management and information scheduling that, starting from network measures, minimize the usage of network resources while preserving the application performance; and ii) the presence of shared repositories that can be used to exchange information between the network and the application, so that the decisions taken by peer regarding connectivity, topology, signaling, and information transfer can be taken with the appropriate knowledge-base.

The overlay topology management and the chunk scheduling of information has been identified as key features for the application to be network-friendly. The first function enables building efficient and rational overlay topologies that are correctly mapped on top of the transport network structure (e.g., considering minimal number of hops between neighbors, locality w.r.t. Autonomous Systems, etc.). The second function guarantees that the network capacity is exploited without waste (e.g., by minimizing retransmissions and pursuing an efficient distribution of chunks, etc.).

These key features must however be supported by measurements, that in most cases are dynamic and can only be implemented in the application overlay itself. As a consequence, the Measurement Module assumes a central role in the system, and the development of efficient algorithms and methods for measurement is just as fundamental to the overall system as efficient topology management and scheduling.

The described architecture is being implemented in the NAPA-WINE project and is made publicly available as software libraries under LGPL license, freely downloadable from the project web site.

Acknowledgments

We are deeply in debt with all the people participating in NAPA-WINE and contributing to the project success with their work and research.

References

[1] L. Jiangchuan, S.G. Rao, L. Bo, H. Zhang, "Opportunities and Challenges of Peer-to-Peer Internet Video Broadcast." *Proceedings of the IEEE*, Vol. 96, no. 1, pp. 11–24, Jan. 2008.

[2] E. Leonardi, M. Mellia, A. Horvart, L. Muscariello, S. Niccolini, D. Rossi, "Building a Cooperative P2P-TV Application over a Wise Network: the Approach of the European FP-7 STREP NAPA-WINE." *IEEE Communications Magazine*, Vol. 46, n. 20, pp. 20–22, Apr. 2008.

[3] V. Aggarwal, A. Feldmann, C. Scheideler, "Can ISPS and P2P users cooperate for improved performance?" *SIGCOMM Comput. Commun. Rev.*, Vol. 37, n. 3, pp. 29–40, Jul. 2007.

[4] T. S. E. Ng, H. Zhang, "Predicting internet network distance with coordinates-based approaches." In Proc. *IEEE Infocom, 2001*, Vol. 1, pp. 170–179, New York, NY, USA, June 2002.

[5] Y. Yardi, "Network Tomography: estimating source-destination traffic intensities from link data." Journal American Statistics Association, pp.365-377, 1996.

[6] J.Seedorf, S.Kiesel, M.Stiemerling, "Traffic Localization for P2P Applications: The ALTO Approach." In Proc. *IEEE P2P 2009*, Seattle, WA, USA, Sept. 2009.

[7] A. Couto da Silva, E. Leonardi, M. Mellia, M. Meo, “A Bandwidth-Aware Scheduling Strategy for P2P-TV Systems”, In Proc. *IEEE P2P 2008*, Aachen, Germany, Sept. 2008.

[8] L. Abeni, C. Kiraly, R. Lo Cigno, “On the Optimal Scheduling of Streaming Applications in Unstructured Meshes,” In Proc. *IFIP Networking 2009*, Aachen, Germany, May 11–15, 2009

[9] L. Abeni, C. Kiraly, R. Lo Cigno, “Robust Scheduling of Video Streams in Network-Aware P2P Applications,” In Proc. *IEEE International Conference on Communications (ICC’09)*, Cape Town, South Africa, May 23–27, 2010.

[10] A. Russo, R. Lo Cigno, “Delay-Aware Push/Pull Protocols for Live Video Streaming in P2P Systems,” In Proc. *IEEE International Conference on Communications (ICC’09)*, Cape Town, South Africa, May 23–27, 2010.

[11] R. Fortuna, E. Leonardi, M. Mellia, M. Meo, S. Traverso, “QoE in Pull Based P2P-TV Systems: Overlay Topology Design Tradeoffs,” in Proc. ACM P2P 2010, Delft, The Netherlands, August 2010.

Biographies

Robert Birke received his PhD in 2009 from Politecnico di Torino, where he is currently holding a PostDoc position. In the past years he participated in different Italian and European funded research projects and currently he is working on the European Napa-Wine project about P2P television. His research interests are in the field of network measurements and peer to peer streaming application design.

Emilio Leonardi is Associate Professor at the Dipartimento di Elettronica of Politecnico di Torino. His research interests are in the field of: performance evaluation of wireless networks, P2P systems, packet switching. He is the scientific coordinator of the European 7-th FP STREP project NAPA-WINE.

Marco Mellia (SM’08) received his Ph.D. degree in 2001 from Politecnico di Torino where he is now Assistant Professor. His research interests are in the fields of traffic measurement, P2P applications and green network design. He has co-authored over 150 papers published in international journals and conferences, and he participated in the program committees of several conferences including IEEE Infocom and ACM Sigcomm. He is the Work Packages coordinator in the NAPA-WINE project.

Csaba Kiraly is research associate at Dipartimento di Ingegneria e Scienza dell’Informazione, University of Trento, Italy, where he is responsible for the coordination of NAPA-WINE activities and chief developer of the NAPA wine streamers and schedulers. He is co-author of several scientific papers.

Renato Lo Cigno is Associate Professor at Dipartimento di Ingegneria e Scienza dell'Informazione, University of Trento, Italy. He is co-author of more than 120 scientific publications in international conferences and journals. He is Area Editor of Computer Networks (Elseveir). His interests are in design and performance evaluation of wired and wireless networking protocols and applications. He is the coordinator of WP2 in NAPA-WINE.

Arpad Bakay received both his MSc and University Doctorate degrees from the Electrical Engineering School of the Budapest Technical University. Since then he has worked for various commercial, for-profit companies as project and products architect, and later also as executive responsible for R&D. In the meantime he also kept academic positions as researcher and lecturer at various Hungarian and foreign universities (ELTE - Hungary, Vanderbilt - USA, BTU - Hungary). He is the coordinator of WP4 in NAPA-WINE.

Tivadar Szemethy obtained is PhD in 2006 from Vanderbilt University, USA, where he participated in projects related to Model-Integrated Computing, Domain-Specific Modeling and Model Verification. Since 2006, he works as software R&D consultant (NETvisor Plc, Hungary) and part-time lecturer at the University of West Hungary.

Fabien Mathieu was a student at the Ecole Normale Superieure, from 1998 to 2002. He completed a PhD in Computer Science in 2004. He currently works as a researcher at Orange Labs. His research interests include P2P content distribution modeling and performance evaluation.

Luca Muscariello is senior researcher at Orange Labs on performance evaluation of communication networks. His research interests include simulation and analytical modeling of wired and wireless networks. He received a MSc and a PhD deegree in Communication Engineering from Politecnico di Torino.

Saverio Niccolini obtained his Master degree in Telecommunications and his PhD in Telematics from the University of Pisa in 2000 and 2004 respectively. He is currently the technical manager of the Real-Time Communications group at NEC Laboratories Europe focusing his research interests on application-level monitoring technologies for trustworthiness applications, cost-efficient traffic management and programmable networks based on OpenFlow.

Jan Seedorf is Research Scientist at NEC Laboratories Europe, working in the real-time communications group. He received a B.Sc. degree in 2001 and a Diploma degree in 2002 in Computer Science from the University of Hamburg, Germany. His research interests include decentralized content distribution, P2P technologies, and security.

Giuseppe Tropea received his Laurea Degree (MSc) in Informatics Engineering from University of Catania, Italy, in October 2002. He is involved in several European projects, both with a small enterprise (Lightcomm S.R.L.) and the Italian Consorzio Nazionale Inter-universitario per le Telecomunicazioni (CNIT). He is co-author of several research papers dealing with privacy protection in digital services and video distribution for P2P communities.

Table 1: Summary of the network layer measurement available in the Monitoring Module.

Index	**Accuracy**	**Usage / Criticalities**
Hop Count	High	Topology optimization
Packet Loss	High	Triggering scheduling & topology update and QoE monitoring
Chunk Loss	High	Triggering scheduling & topology update and QoE monitoring
Delay Measurements		
Latency	> 10ms	Source and stream absolute time alignment / Clock synchronization below NTP precision is hard to obtain
RTT	< 1ms	Chunk and peer scheduling / Timestamping accuracy limits precision
Jitter	< 1ms	Delay jitter may be an indication of congestion
Bandwidth Measurements		
Mid term Throughput	High	Neighborhood selection and chunk/peer scheduling
Capacity	Low	Pacing chunk scheduling for altruistic peers to avoid clogging a PHY bottleneck (e.g., ADSL links beyond a LAN)
Available Bandwidth	Low	Fundamental to avoid congestion in the network / Difficult to achieve high accuracy due to correlations and the lack of efficient algorithms to perform the measure

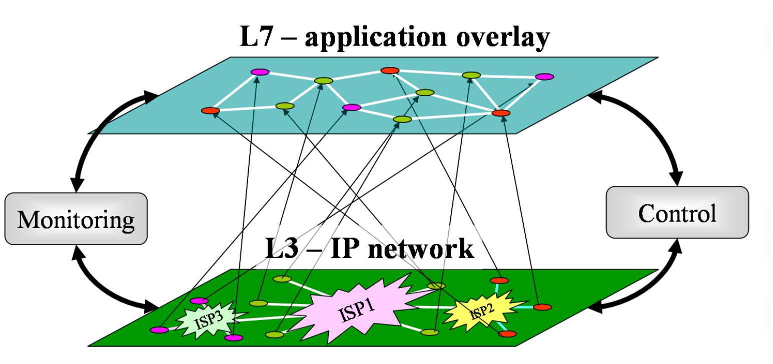

Figure 1: Schematic representation of the NAPA-WINE approach. Nodes at the network layer (L3) and peers at the application layer (L7) form the network and overlay topology respectively. Cooperation among the two layers is possible thanks to Monitoring and Control capabilities offered by the NAPA-WINE architecture.

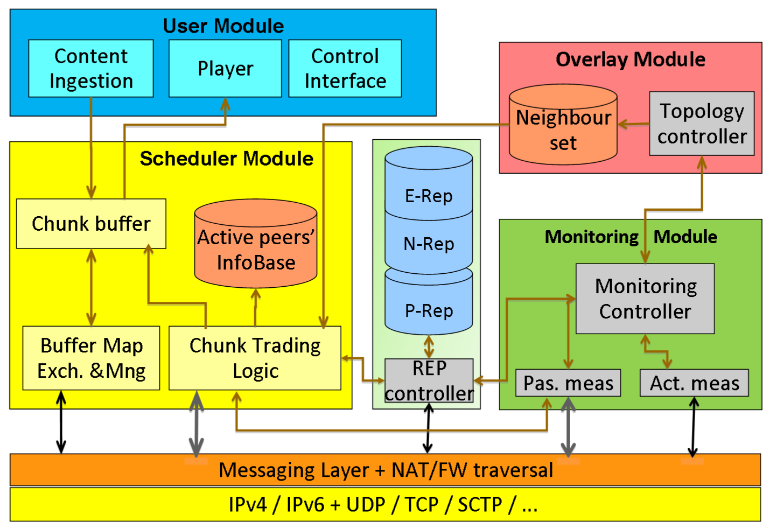

Figure 2: Global NAPA-WINE architecture. User, Scheduler, Overlay, Messaging, Monitoring, and Repository modules are highlighted using different colors. Arrows represent the relationship among modules, distinguishing data path (gray), signaling path (black) and internal communication (brown).

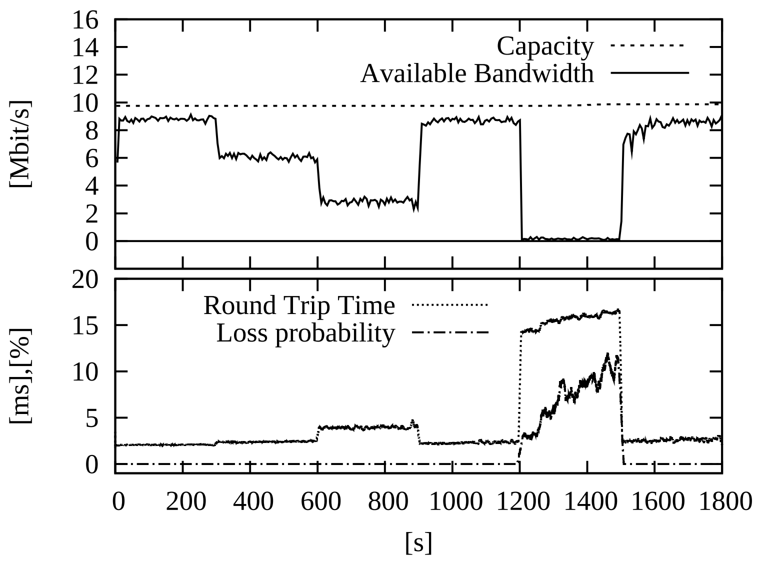

Figure 3: Measurements obtained from the monitoring module.

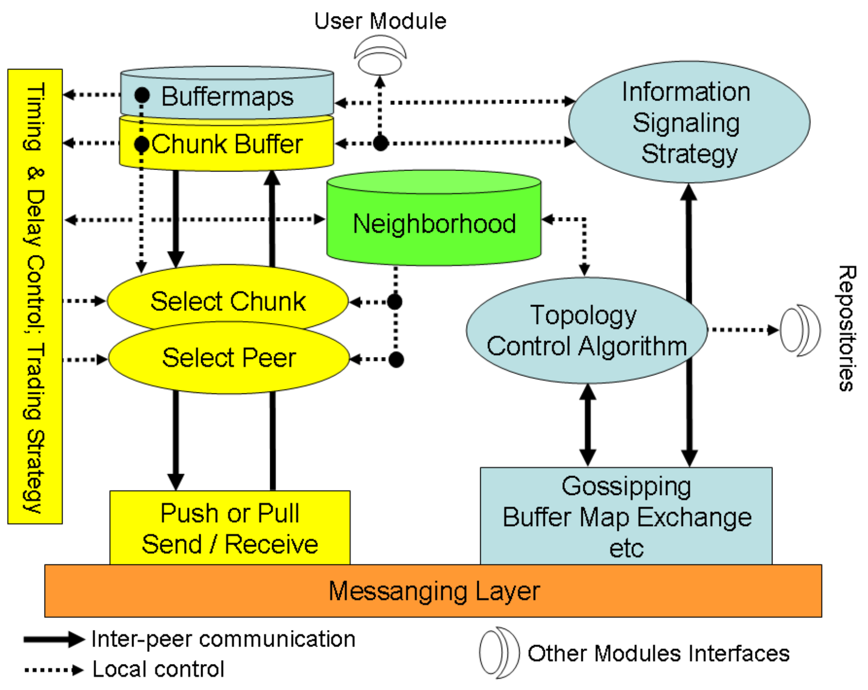
User Module
Buffermaps
Chunk Buffer
Information Signaling Strategy
Timing & Delay Control; Trading Strategy
Neighborhood
Select Chunk
Select Peer
Topology Control Algorithm
Repositories
Push or Pull Send / Receive
Gossipping Buffer Map Exchange etc
Messanging Layer
Inter-peer communication
Local control
Other Modules Interfaces

Figure 4: Logical organization of the scheduling and topology management modules, around the neighborhood data-base

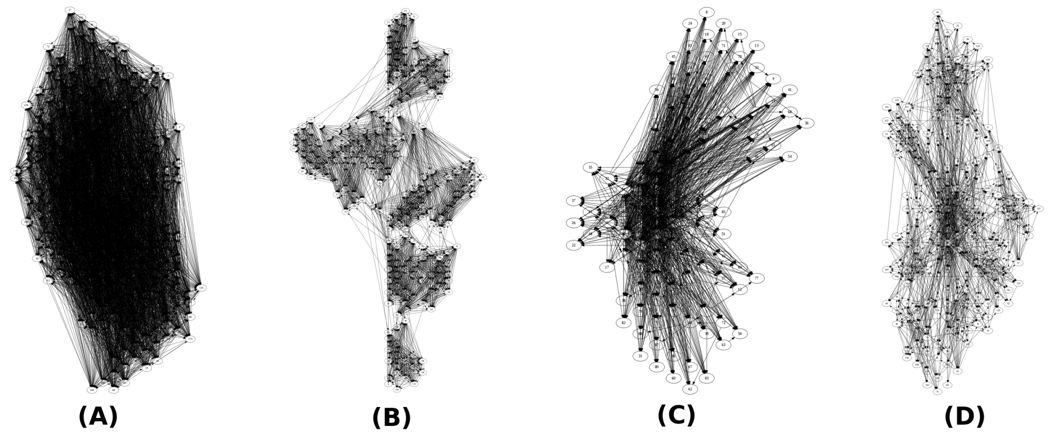

Figure 5: A) completely randomly chosen neighbors, B) neighbors chosen based on locality information provided by E-REP, C) neighbors chosen based on upload bandwidth peer information, D) combination of B) and C).

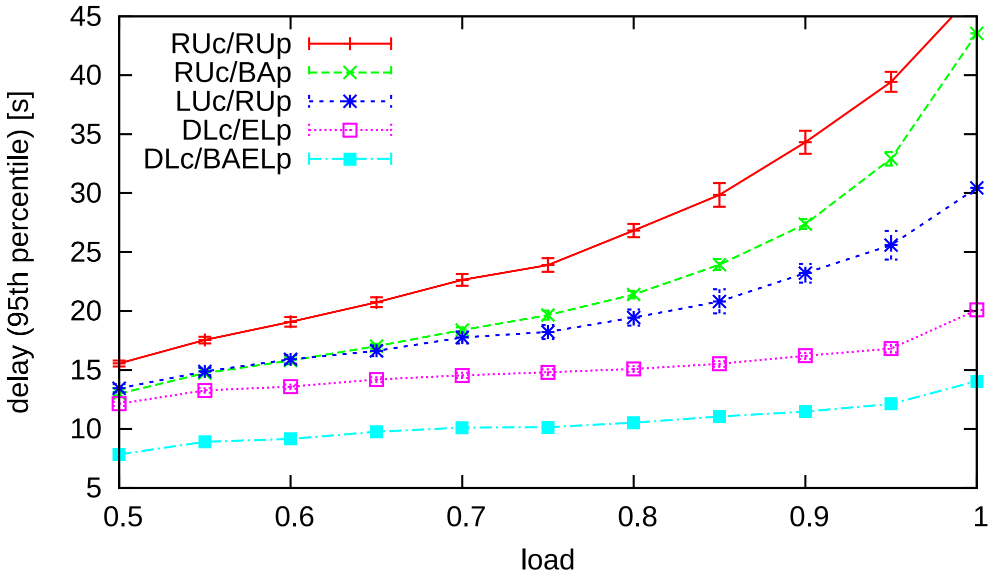

Figure 6: Performance of several scheduling policies for immediate Push. 95th percentile of the chunk delivery delay versus load.