Noname manuscript No.
(will be inserted by the editor)

Peer-to-Peer Media Streaming: Insights and New Developments

Zhijie Shen · Jun Luo · Roger Zimmermann · Athanasios Vasilakos

December 7, 2010 / ...

Abstract Media content delivery started to emerge and subsequently have a major impact on Internet traffic and usage over the last decade. As a consequence, researcher and practitioners soon realized that peer-to-peer (P2P) systems, due to their self-scaling properties, had the potential for improved scalability compared with traditional client-server systems. Since then, various P2P media streaming systems have been deployed successfully, which in turn has led to a series of instructive surveys on these systems and their enabling techniques.

It has been a few years since the last set of surveys was reported, and numerous technological discoveries have been achieved since. The focus of this report is to survey and discuss these new findings, which include new technological developments as well as new understandings of these and existing P2P streaming techniques through both novel modeling methodologies and measurement-based studies.

Keywords Peer-to-peer media streaming · live streaming · video-on-demand (VoD)

Z. Shen
National University of Singapore, Singapore
E-mail: z-shen@comp.nus.edu.sg

J. Luo
Nanyang Technological University, Singapore
E-mail: junluo@ntu.edu.sg

R. Zimmermann
National University of Singapore, Singapore
E-mail: rogerz@comp.nus.edu.sg

A. Vasilakos
University of Western Macedonia, Kozani, Greece
E-mail: vasilako@ath.forthnet.gr

1 Introduction

With the emergence of convenient and advanced digital multimedia capture and production technologies, the amount of available media content has increased tremendously. As a consequence, the Internet traffic caused by content distribution has grown similarly. Internet video traffic has been measured at more than one-third of all consumer network traffic in 2009 and is projected to account for 57 percent by 2014, thus being on track to become the single biggest Internet traffic generator [9]. This trend significantly challenges content providers as well as Internet service providers (ISPs) in their quest to ensure a high quality user experience. Nowadays peer-to-peer (P2P) media streaming applications have attracted large user communities. Such popularity can in part be attributed to the self-scaling properties of P2P: whenever a client participates in a system to consume streams, its upload capacity also adds to the overall system resources.

Peer-to-peer streaming systems emerged originally with two fundamental architectures. Inspired by the distribution topologies used in IP multicast, initial systems focused on *tree-based push* architectures. Under this scheme, the peers are organized in a tree-like manner for data dissemination: each peer has one parent and multiple children. The media stream is pushed from the tree root (*i.e.*, the stream server) to all the leaf nodes. One vulnerability of dissemination trees is their lack of robustness under peer churn. To remedy this weakness, *mesh-based pull* approaches were introduced. With this scheme, the overlay construction is simplified: each peer node maintains a local connectivity list that contains a number of partner peers. A node periodically exchanges media data availability information with its partners and pulls the data that it is missing from one of its partners.

The studies devoted to the designs of the aforementioned tree-based push and mesh-based pull schemes have been discussed in some prior literature surveys [14, 26, 30]. These

studies also analytically discussed the pros and cons of the proposed techniques. More recently, the mesh-based pull scheme has emerged as a favorite for commercial deployments and research has started to focus on other aspects of P2P systems. Such new issues have not been reported by prior surveys and hence are the focus of this study. More precisely, we will cover the following recent developments in this survey.

- Several modeling studies that have attempted to theoretically explain and predict the behavior of P2P streaming systems [5, 22, 27, 28].
- As a number of real-world systems have been deployed, various studies have monitored their effects by collecting traces, diagnosing defects or inefficiencies and proposing corresponding remedies [13, 33, 51].
- Deployed P2P streaming systems have created randomized and far-reaching Internet traffic, seriously concerning ISPs. Several traffic localization techniques have been proposed to relieve ISPs from heavy cross-ISP traffic [29, 36, 42, 45, 47].
- Since tree-based push and mesh-based pull schemes demonstrate complementary advantages, some researchers have devised hybrid solutions to combine them in order to obtain both their merits [34, 48, 55].
- Network coding and layered coding have been applied to P2P streaming systems to improve stream throughput and deal with heterogeneous last-hop bandwidth capacities [40, 49, 50, 53].
- With the growing popularity of wireless mobile networks, P2P architectures which were designed for wired networks require modifications to adapt to the characteristics of such new environments [24, 31, 46].
- Last but not least, as multimedia production techniques have advanced, non-traditional contents such as multiview video and 3D mesh objects, have become more available and have required the corresponding streaming techniques to meet specific properties (*e.g.*, improved user interactions) [7, 17, 23].

As we aim at tracing the latest developments in P2P streaming technology, we attempt to deliver a comprehensive coverage on various relevant topics with a proper technical depth for each topic.

The remaining sections are organized as follows. Section 2 reviews background information on P2P media streaming, including the relationship between P2P and content delivery networks (CDN), the comparison between tree and mesh schemes, the difference between live streaming and video-on-demand (VoD) and the modeling-based analysis of P2P streaming. We introduce studies that discuss the new technological developments consisting of all the areas outlined in the last paragraph (except P2P streaming modeling) in Section 3. Next, Section 4 concludes our survey, highlights challenges and provides a future outlook.

2 Foundations of P2P Media Streaming

We first summarize some of the foundations that have been presented in previous surveys. We also present principles and guidelines that have recently been discovered through either measurement-based or modeling/simulations studies. These, on one hand, serve as a background for our later presentation, and on the other hand, also help us to highlight the new developments described later in this survey. We will start our presentation with a short introduction of Content Delivery Networks (CDN), an overview of the two mainstream P2P system architectures, a description of two common types of applications (live and video-on-demand streaming), and modeling methodologies for performance evaluations.

2.1 Content Delivery Networks

An immediate idea of supporting media streaming is to reuse the client-server model that is widely used for Internet applications. However, the bandwidth-demanding nature of media streaming almost instantly questions the feasibility of this approach. As a direct improvement to the conventional client-server service model, CDN abides by the same principle but slightly expands the concept of the "server." Under the CDN model, instead of downloading from a video source server, clients may find a content delivery server that is close-by and achieve a more efficient download using that server. The video source server acts as the "server" for the content delivery servers by pushing video contents to them. Therefore, the CDN model is effectively a two-layer client-server model. If the networks connecting the source server and the content servers are adequately dimensioned and if the content servers are properly placed at strategic locations, a CDN based solution can deliver good services. In fact, YouTube[1], as the largest online video-sharing service, keeps employing CDN to deliver its most popular videos to its users, and Akamai[2] runs a profitable business as the world's largest CDN service.

The major drawback of the CDN model is its inability to take advantage of the upload bandwidth of the clients, which effectively puts all the load onto the CDN infrastructure (the involved servers and networks). In fact, the bandwidth provisioning within a CDN has to grow proportionally with the number of clients, making a CDN an expensive solution for large client populations. However, the excellent service quality of an adequately dimensioned CDN is undeniable. Therefore, a hybrid system combining both CDN and P2P has long been envisioned. For instance, *LiveSky* is such a P2P-CDN hybrid streaming system designed and de-

[1] YouTube, http://www.youtube.com/
[2] Akamai, http://www.akamai.com/

ployed recently [54]. The servers in CDN are organized in a tree structure, which consists of the source server owned by content providers, the core and the edge service nodes from top to bottom. Each edge service node serves the end users that are usually in the same ISP. The clients form a tree-mesh combined overlay to forward the received stream. The P2P-CDN hybrid architecture achieves a better balance between scalability and streaming quality guarantee, ensures shorter startup latency and reduces cross-ISP traffic.

2.2 Two Mainstream P2P System Architectures

P2P media streaming aims to utilize the uploading bandwidth of the clients, hence it has the potential to substantially reduce the traffic load on the server side. In order to achieve this goal, two mainstream architectures have been devised: namely *tree-based push* systems and *mesh-based pull* systems[3]. It is interesting to note that, whereas a media file is really "streamed" in a push system (i.e., media data are identified at the overlay by their smallest possible units such as TFRC packets), it is shared within a pull system at a coarser granularity, in the sense that a media file is "chopped" into chunks that contain many packets and a peer requests chunks instead of packets from its suppliers. We briefly summarize the main characteristics of the two architecture types.

Tree-based Push Systems: Initially proposed in a sequence of papers including, for example [3, 6, 21], and also implemented by the *end system multicast* (ESM) developed at CMU [8], the tree-based system serves a natural extension of CDN. Instead of only having two layers of the client-server structure, it has many such layers by allowing every client to become a potential server to some other clients. The following advantages are obvious given the construction of a tree-based system:

- Compared with CDN, the client upload bandwidth is better utilized, and the traffic load on the video server is significantly reduced, leading to a more scalable system.
- Within a stable streaming tree, the delay is strictly bounded: its maximum value is determined by the longest overlay path from the root (server) to the leaf nodes.

However, the disadvantages due to the rigid tree structure are also evident:

- The complexity of maintaining a stable tree is high in the face of peer churn. In particular, when internal nodes leave, streaming disruptions may happen due to a slow recovery of the streaming tree.
- The upload bandwidth is not fully utilized, as the leaf nodes that account for the major part of the system never share their upload bandwidth.

These later problems can be addressed by introducing multi-tree streaming (e.g., [6]), but at the expense of further increasing the maintenance complexity.

Mesh-based Pull Systems: Unlike the tree-based push system, the mesh-based system first appeared as practical implementations (e.g., PPLive) and was then investigated by academia (e.g., [56]). Borrowing existing techniques from unstructured P2P file sharing system such as Gnutella[4], the mesh-based system requires peers to share the information about their media repository, which guides a peer to pull its desired media chunk from others. The pros and cons of mesh-based systems are pretty complementary to those of tree-based systems. In particular, the following are the well known properties of a mesh-based system:

- It is very robust against peer churn, due to the randomness embedded in the peering procedure and also the high peering degree in a system.
- The maintenance complexity is low, thanks to its robustness against system dynamics and also the less rigid logical structure in the overlay.
- It, however, cannot guarantee a deterministic delay bound, as the path from one peer to another can be arbitrarily long. This results in various unpleasant playback experiences, such as long startup delays and playback freezes.
- Moreover, the suboptimal distribution of chunks due to random pulling may lead to a certain extent of bandwidth waste.
- Finally, the system has to strike a compromise between efficiency and delay.

The very last property stems from the "chunked" nature of the media data in such a system, which makes the store-and-forward delay at individual peers non-negligible. Though protocol efficiency is higher with larger chunk size (as the overhead can be amortized), the delay will become larger.

2.3 Two Types of Applications

Applications of media streaming systems can be broadly classified into two categories, namely live streaming and

[3] Although we can, in theory, push media content in a mesh-based system [37], the lack of a clearly defined parent-child relationship in such a system may result in blindly pushing media content to a peer already having it, which wastes the already scarce peer upload bandwidth resource. Therefore, we are not listing it as a mainstream architecture. However, we will present new insights on such a system obtained from probabilistic analysis in Sec. 2.4.1.

[4] Gnutella, http://rfc-gnutella.sourceforge.net/index.html

VoD. The majority of the studies fall into the former category as live streaming is considered a more typical application. The recent insights into the latest version of *CoolStreaming* exhibit the basic components of a P2P live streaming system [25]. The original *CoolStreaming* had a BitTorrent-like content discovery mechanism, *i.e.*, random peer selection, chunk availability information exchange and chunk swapping. However, in the new version, some important design changes were made:

- The entire video stream is divided into a number of sub-streams by using modular arithmetic rather than some coding technique. This change not only improves the streaming quality but also promotes resilience to peer dynamics.
- In addition to the cache buffer, a group of synchronization buffers are added for each corresponding sub-stream.
- The pull request actually has become a subscription command. When one pull request is received, the chunks will be consecutively pushed to the requesting peer without further requests. Hence the protocol overhead is reduced. See Section 3.3.1 for a detailed discussion.
- A peer monitors the status of the on-going sub-stream transmissions. Whenever the peer detects an inadequate streaming rate from a certain parent, it will switch to another parent selected from its local partner list.

Many of the techniques for live streaming can also be applied to VoD. Nevertheless, although the two types of applications are similar to each other, there are still some key differences which cause special treatments for VoD [19]:

- Whereas live streaming requires clients to be synchronized with the broadcast server (though they may lag slightly behind the server), VoD allows individual clients to watch whatever content they want whenever they want it. Therefore, due to the asynchronous viewpoint, discovering the peers holding the required content is more challenging in VoD. PPLive employs three content discovery mechanisms: tracker-based lookup, DHT-based lookup and gossip-based notifications. Trackers keep the freshest content distribution information; DHT is leveraged to balance the tracker load and back up a possible tracker outage; and gossip enables peers to autonomously exchange content availability information, reducing the workload imposed upon trackers.
- While the content of live streaming is generated in real-time, that of VoD is usually prepared in advance. Moreover, VoD clients contribute some secondary storage (*e.g.*, 1 GB of disk space), where the watched content is cached to serve the latter peers watching the same content. For this reason, the available peer resources are always more versatile in VoD. Given limited storage, to ensure efficient data replication, PPLive has tried several strategies and adopted the *available-to-demand* strategy that keeps the number of replications to the number of demanding peers around a certain ratio. Meanwhile, caching of multiple movies is applied as it is more flexible than single movie caching. These strategies significantly lower the server workload. In contrast, prefetching is not employed as the benefits and risks are not confirmed yet.
- VoD allows more user interactions, such as pause and jump to another viewpoint. The common method to shorten the startup latency after a jump is to prefetch some data at anchor points throughout a video. The data of the anchor points closest to the target location is used for playback if the exact data is missing there. Nevertheless, PPLive currently does not use anchor point prefetching because the measured jump rate is low (1.8 times per session [19]) and the transmission scheduling is good enough to keep the latency at an acceptable level.

2.4 Modeling Methodologies for Performance Evaluation

As has happened in many other fields of computer science, P2P media streaming started with pure system implementations. However, at some point in time, theoretical guidelines are necessary to provide general insights of system behaviors and also to suggest possible directions for optimizing system performances. The modeling techniques developed for P2P streaming can be roughly classified into two categories: stochastic modeling (e.g., [5, 22]) and combinatorial modeling (e.g., [27, 28]). While the former delivers insights into the asymptotic or "average" behavior of a system, the latter focuses on the best or worst case system performances. In this section, we discuss a few proposals that make use of these modeling techniques, with an emphasis on the insights that resulted from the respective analyses. Note that, due to the complex behavior of P2P streaming systems, all the modeling techniques are based on some relatively strong assumptions. Consequently, the insights obtained from analytically models are either rather intuitive or need to be carefully interpreted before being used as design guidelines.

2.4.1 Stochastic Models

As a networking problem, P2P media streaming lends itself to queueing theory based stochastic analysis. Kumar *et al.* [22] were among the first in applying queueing theory to obtain insights on P2P streaming performance. In particular, they aim at understanding how system throughput is affected by peer churn and by system settings (e.g., peers composition and buffer size). Assuming a server upload rate u_s, a video bit rate r, and a two-level peer composition (i.e., *super peers* with high-speed access and *ordinary peers* with residential broadband access) with respective upload rates u_1 and u_2 ($u_1 > r > u_2$), the maximum achievable stream-

ing rate $\phi(n_1,n_2)$ is obtained for a churnless system with n_1 super peers and n_2 ordinary peers:

$$\phi(n_1,n_2) = \min\left\{u_s, \frac{u_s+n_1u_1+n_2u_2}{n_1+n_2}\right\} \quad (1)$$

Based on this result, a system with peer churn is investigated to derive the universal streaming probability, i.e, the fraction of time when $\phi(n_1,n_2) \geq r$. Since the system is now dynamic, n_1 and n_2 become two $M/G/\infty$ processes $P_1(t)$ and $P_2(t)$, and super peers (resp. ordinary peers) are assumed to have an arrival rate of λ_1 (resp. λ_2) and an expected sojourn time μ_1^{-1} (resp. μ_2^{-1}). It is pretty clear that $\mathbb{E}(P_i) = \frac{\lambda_i}{\mu_i} \equiv \rho_i,\ i = 1,2$. The universal streaming probability $\mathscr{P}$, in an asymptotic regime with $\rho_1 = K\rho_2 + \beta\sqrt{\rho_2}$ and $\rho_2 \to \infty$, depends on a critical parameter $c = \frac{r-u_2}{u_1-r}$:

$$\mathscr{P} = \begin{cases} 1 & K > c \\ F\left(\frac{-\beta}{\sqrt{c+c^2}}\right) & K = c \\ 0 & K < c \end{cases} \quad (2)$$

where $1-F(\cdot)$ is the cumulative distribution function of the standard normal distribution. The following observations are immediate from the two equations:

(1) Peer churn (the changes in n_1 and n_2) introduces fluctuations in the system throughput.
(2) Whether a system offers satisfactory throughput strongly depends on the composition ratio (in terms of $\frac{\rho_1}{\rho_2}$).

In [22], a fluid queueing model is further used to derive the benefits of buffering: it mitigates the effects of a fluctuating $\phi(P_1(t),P_2(t))$. In addition, a similar model is extended to investigate multi-channel P2P live streaming in [52]. In particular, two systems, namely the *isolated channel design* (ISO) and the *view-upload decoupling* (VUD), are compared. While ISO considers a multi-channel system as multiple single-channel systems, VUD not only requires each peer to upload the viewing channel, but also assigns other channels to a peer to enable cross-channel resource sharing. The queueing model helps to demonstrate the benefits, in system throughput, of VUD, and also suggests efficient VUD design guidelines. Readers are referred to [52] for the detailed modeling technique.

One drawback of a fluid model is its inability to characterize delay, whereas there is a common desire to understand the performance tradeoff between throughput and delay. To this end, Bonald *et al.* [5] make use of large deviation theory to provide bounds (in probability) on the delay of P2P live streaming systems with an epidemic-style dissemination mechanism. Their results establish the throughput-delay optimality of a so called *random-peer*, *latest-useful-chunk* (rp/lu) dissemination mechanism, in the sense that rp/lu achieves an optimal throughput, as well as a delay within a constant term from the optimal value. More precisely, assuming that the arrival rate of media chunks (at the server) is $\lambda < 1$, then there exists a constant γ such that for all $m \geq 1$ the following large deviation bound holds for the chunk dissemination delay D:

$$\Pr(D \geq \log_2 N + m) \leq \frac{\gamma}{m} \qquad \text{for sufficiently large } N.$$

2.4.2 Combinatorial Models

In order to perform stochastic analysis, one has to assume either an ideal dissemination mechanism (e.g., [22]) or a randomized one (e.g., [5]). As a result, such analysis would not suggest to us practical system design methodologies, in particular how to construct dissemination tree(s) for media streaming[5]. In this section, we discuss two proposals [27,28] that introduce combinatorial tools to design streaming trees with specific (optimal) performances. Note that these proposals do not solely rely on combinatorics, mean field approximations are also used to bring system dynamics into the picture (but this later technique is not our focus here).

One of the main contribution of [28] is an analytical delay comparison between single-tree, multi-tree (with degree m), and a newly proposed snow-ball dissemination. This snow-ball dissemination requires each peer to keep pushing a received media chunk to other peers until the chunk is fully disseminated within the system. Consequently, the number of peers receiving a chunk increases as a geometric progression with base two, which in turn leads to a logarithmic delay. The comparison provides the following expressions for the average delays of disseminating one chunk in a system of N peers:

$\frac{N+1}{2}$	single-tree
$\frac{m+1}{2\log_2 m}\log_2 N + o(1)$	multi-tree
$\lceil \log_2 N \rceil + 1$	snow-ball

This obviously shows that snow-ball has the best performance, which is actually the minimum delay as proven in [28]. Note that the extension from single-chunk dissemination to multi-chunk streaming is highly non-trivial. As [28] shows only the existence of such an extension based on a nonconstructive proof, we will discuss a recent algorithm construction to this end in Section 3.3.3.

A similar analytical framework is also considered in [27] to investigate the server load and delay performance in trees. The objective of minimizing server load s for a given system throughput r is actually the dual problem of maximizing system throughput r under a given server load s (which has been studied in [28]). However, there are two major differences between [27] and [28]. Firstly, while [27] deems propagation delay as the dominating effect, [28] considers "chunked"

[5] Although only a push system is specified to use tree to perform media streaming, a pull system, in its steady state, also stream media content along certain tree(s) [55].

media contents for which the transmission delay becomes more prominent. Therefore, the minimum delay dissemination, as proven in [27], relies on multi-tree pushing with each tree having a depth of two and a degree of $m = N - 1$ in a system of N peers. Secondly, [27] takes one step further by investigating the case where peer selection is constrained, such that the degree of a tree can only be smaller than $N - 1$. Two major results from this analysis are:

- The constraint on the tree degree does not affect the minimum server load, and hence the system throughput. An algorithm is proposed to construct degree-constrained trees to achieve the minimum server load.
- There is a tradeoff between the server load s_{min} and the minimum delay (or tree depth) D_{min}, i.e,

$$D_{min} \geq \left\lceil \frac{N}{\left\lfloor \frac{s_{min}}{r} \right\rfloor} \right\rceil$$

3 New Technological Developments

In this section, we discuss several technological developments of P2P streaming in the last few years. These are new technologies that, we believe, should serve as the driving force to future developments.

3.1 Overlay Network Monitoring and Diagnosing

The cost-efficiency of P2P streaming has significantly interested industry. As a consequence a number of commercial P2P streaming systems have attracted large user communities in recent years. For large-scale deployments, it is crucial to be able to monitor the behavior and the performance of the considerable number of peers. The monitoring information interests the following stakeholders:

- High quality service is critical to attract users. System operators must know clients' performance in detail and across time. For instance, the statistics of peers' receiving rates may help to determine how much server capacity should be provided.
- Third-party companies require this knowledge to launch credible assessments or advertisements, and to provide users with a constructive basis for selecting an application.
- Researchers are also eager to gain insights into real systems, thus enabling solid evaluation of the various designs and algorithms. Furthermore, they would like to diagnose the deficiencies in order to inspire future research work.

In this section, we introduce three exemplar real system measurement studies and the system optimizations based on them.

3.1.1 Inferring Network-Wide Quality

While streaming quality seems clearly an important characteristic that requires monitoring, the question of what exactly are the most appropriate and measurable parameters to infer network-wide quality is not that obvious. Hei *et al.* [13] have advocated observing the buffer map, which is used by peers to advertise the video chunk availability to each other. In their study they verified and discussed the correlation between the buffer map and the network-wide quality by measuring PPLive.

The first task was to acquire the traces of peers' buffer maps. A straightforward method would be to set up monitors on each peer to collect the related data and report it to a central server for post-processing. However, this approach is only feasible for the system operator who is allowed to modify the software. Usually, researchers need to come up with indirect and ingenious ways. The authors set up buffer map crawlers, organized in a master-slave architecture. The crawlers contacted a number of peers to request their buffer map information, following the protocol of PPLive. Understanding the protocol was the most difficult task because of its proprietary nature.

With the collected traces, the authors derived two correlations between quality and buffer maps:

- The buffer map width and the number of continuous available chunks from the beginning of the buffer map are two consistent metrics to infer playback continuity in terms of the frequency of playback freezes and playback reboots. Specifically, the two events occur when the buffering level declines. Furthermore, a reboot is always coupled with a large fluctuation of the advance rate of the buffer map.
- The buffering level can also determine the startup latency as the system always sets a threshold for the number of buffered chunks, above which the video may be played back.

Having determined these correlations, the authors further inspected the network-wide quality of PPLive:

- During a period of playback continuity decline, the crawlers setup up on the HK campus, the NY campus and at cable networks all detected buffering level degradations and the fluctuation of the advance rate of the buffer map. The relationship indicates that the proposed correlations are network-wide consistent. Additionally, the performance degradation is likely to be the result of a peer population increase.
- Peers' latencies in the same channel are similar to each other. Moreover, the lag of a peer is rather stable during its session.
- The buffer maps of all the concurrent peers can also be used to infer the chunk retrieval ratio, the evolution of

which experiences a knee point (a chunk is swiftly disseminated to a proportion of peers and then gradually to all). Moreover, the life time of chunks at a peer or of network-wide scale is rather stable, and most chunks are available at the majority of the participating peers.

3.1.2 Identifying Superior Peers

Another issue related to real system deployments is dealing with peer heterogeneity, *i.e.*, different upload bandwidths and session lengths. P2P systems always prefer peers that can provide a higher upload bandwidth and stay longer in the system since this group of peers improves system scalability (see Section 2.4.1 for details). Liu *et al.* [33] analyzed the traces from UUSee, a popular commercial P2P-TV application, and utilized a number of statistical methods to identify desirable peers, which were named *superior* peers. Furthermore, the authors proposed a *superiority index* as a valuable indicator for peer selection to improve streaming quality.

Their first insight from the traces was the negative correlation between the streaming quality and the peer population. Aware of this problem, the authors proposed to attract the high-bandwidth-contribution peers to stay longer by providing them with better download rates. Therefore, they investigated the factors that influence peer session lengths, and obtained the following conclusions:

- The peers that enjoy better streaming quality tend to stay in the system for a longer time. The streaming quality here is inferred from the average or initial buffering level, which has been discussed in Section 3.1.1.
- The peers joining the system in the evening stay longer than those joining at other times.
- The peers in the popular channels tend to stay longer than those in the non-popular channels.

Next, by leveraging a regression model, the authors quantified the session lengths based on these three factors. The authors checked the correlation between the session lengths and the bandwidth contributions. However, no remarkable correlation was found to exist. In other words, stable peers may not contribute more bandwidth. Instead, the investigation revealed that the bandwidth contribution is positively correlated with the streaming quality (indicated by the initial buffering level).

Finally, to integrate a peer's session length and bandwidth contribution, the authors of [33] defined the superiority index as follows

$$\textit{superiority index} = \textit{session length} \times \textit{upload bandwidth}. \quad (3)$$

The session length and upload bandwidth are predicted based on measurable metrics, *i.e.*, the initial buffering level, the peer population and the join time. With the index, the authors improved the peer selection algorithm, where the supplier may deter peers having small index values to ensure enough bandwidth provision for those having larger index values. The authors' trace-driven simulations have confirmed a streaming quality improvement achieved by their peer selection algorithm.

3.1.3 Refocusing on Servers

While most research on P2P streaming has focused on peer-side optimizations, Wu *et al.* [51] sought to improve the efficiency of server bandwidth provisioning. Through the analysis of 400 GB and 7-months worth of traces from UUSee, the authors found that the server bandwidth provisioning became inadequate with an increase in channel numbers. This motivated the authors to investigate how to allocate the limited sever bandwidth among the concurrent channels to maximize overall streaming quality. In addition, there was interest in keeping the solution ISP-friendly, for which we will give a detailed discussion in Section 3.2.

One of the contributions of the study presented by [51] was insights into several important metrics of a real system. The authors' analysis of the traces from UUSee can be summarized as follows:

- With the growth of the number of channels, the server bandwidth requirement increases as well while the streaming quality declines. As an example, during a Chinese New Year flash crowd, the streaming quality abruptly dropped because of a significant upload bandwidth deficit.
- Both the traffic originating from servers and peers increased over time when the number of channels rose.
- At a short-time scale (3-hour period), the streaming quality is observed to be positively proportional to the server bandwidth supply, but it is negatively proportional to the peer population.

Based on the observations outlined above, the authors proposed an online server bandwidth provisioning algorithm named *RATION* as follows. First, the problem was formulated into a linear program, which aims to maximize the overall streaming quality with each channel assigned a weight. Here the solution must be computed in advance, hence the peer number needs to be estimated. The authors adopted the *auto-regressive integrated moving average model* (ARIMA) to predict the future population. Moreover, the streaming quality is interpreted from the product of the server bandwidth supply and the peer population, and linear regression is applied to estimate the parameters according to the recent status. Then, the authors proposed a water-filling approach to achieve the optimal bandwidth provisioning. Instead of re-computing values every round, the algorithm repeatedly moves the over-provisioned bandwidth to the channel that has the maximum marginal utility, until all the channels have the same marginal utility. The algorithm can be carried out on a per-ISP basis to restrict cross-ISP traffic.

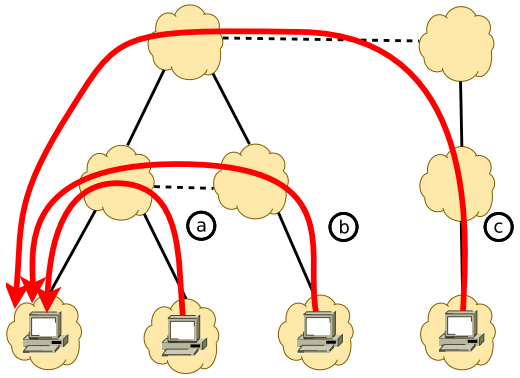

Fig. 1 Exhibition of the difference between overlay and underlay routing. The background is the organized AS-level structure. The solid lines are the consumer-provider relationships while the dashed lines are the peering relationships. The red lines represent the one-hop overlay connections, which respectively traverse 2, 3 and 5 inter-AS links.

Finally, the authors evaluated their algorithm on a group of high-performance computers, which emulated the stream servers. To be realistic, they extracted the peer behaviors from the traces and entered them into the simulation. The experimental results demonstrated a good streaming quality experience, especially for the non-popular channels.

3.2 Traffic Locality in P2P Streaming

The distributed nature of P2P streaming has resulted in randomized, wide-reaching traffic. A prior measurement study [20] shows that more than 70% of the content existing on topologically close peers was downloaded from distant peers in BitTorrent. As the traffic volume rises, traffic routing inefficiencies become more prominent and is raising considerable challenges for network resource management.

Specifically, the long-distance traffic intensifies the stress on network infrastructures and consumes excessive network capacity which could have been used in support of other applications. The biggest trouble is the cross-ISP traffic, which disturbs the traffic engineering of ISPs, increases the congestion levels at the gateways between them, and probably raises their operating cost. Aware of these challenges, ISP have resorted to throttling cross-ISP links by dropping P2P packets, which might degrade the user experience. Meanwhile, deploying some caching proxies is a partial solution. However, this approach would still increase operating costs and is likely to involve thorny copyright issues for ISPs.

Recently, researchers have begun to diagnose the cause of the traffic routing inefficiencies. They found that the existing P2P systems employ a peer-selection strategy which is unaware of the network underlay information, resulting in a largely random connection topology. For example, in Fig. 1, the connections *a*, *b* and *c* which are of different underlay distances are regarded equally distant (1-hop) from the prospective of the overlay. A number of application-layer solutions have been proposed to either skew the peer selection strategy or the media chunk scheduling mechanism such that partner nodes within close proximity are preferred. Nevertheless, only a few of the proposed techniques focus on live streaming systems [29, 36, 42, 45, 47], with most others pertaining to file sharing applications. While some characteristics are common between the two types, there still exists important difference. File sharing systems can accept flexible downloading rates while live streaming requires a sustained level of packet traffic, such that chunks can be retrieved before their playback deadlines. Thus, traffic locality is more difficult in the streaming scenario. Here we will discuss the studies focusing on live streaming systems.

3.2.1 Measurements on PPLive

Since the traffic locality degree in existing P2P live streaming systems had not been well understood, Liu *et al.* [29] conducted measurements on PPLive and found that in general, PPLive naturally exhibits a fluctuating traffic locality ranging from close to 0% to about 90%. This traffic locality seems to be a side-effect of the skewed ISP-size distribution. The authors also noted the potential for further improvements with some proactive techniques.

For the measurements, 8 hosts with PPLive V1.9 installed were deployed in 4 ISPs, *i.e.*, China Telecom, China Netcom (Unicom now), China Education and Research Network (CERNET) and George Mason University, USA. The measurements lasted 4 weeks, and more than 130 GB UDP packets were collected with Wireshark[6]. Afterwards, a peer list was extracted from the raw data, the Team Cymru service[7] was leveraged to obtain the peers' home autonomous system (AS) information, and the chunk requests and responses were mapped. Through the analysis, the authors concluded the following findings.

- Most chunks requested by a peer are downloaded from peers in the same ISP. Therefore, in some extreme cases, the traffic locality degree can reach about 90%. This is a major finding, which has not been discovered in prior studies. This phenomenon is partly determined by the peer distribution among the ISPs. As was shown by Liu *et al.* [29], most of the peers reside in China Telecom and Netcom so that the peers in these two ISPs can obtain enough local peer resources. In contrast, the peers from George Mason University, USA cannot. That is why the observed traffic locality there was not as high as that from the others.
- Partners from the same ISP respond faster to the requesting peer. This phenomenon is reasonable because the responses originated from the same ISP traversed through

[6] Wireshark, http://www.wireshark.org
[7] Team Cymru, http://www.team-cymru.org/Services/ip-to-asn.html

smaller number of physical links. Peers prefer this group of partners. According to the measurement results, most of the requested chunks are replied from the peer's top 10% of the partners, which have smaller RTT to the requesting peer. This serves as the other reason for the natural traffic locality.

3.2.2 ISP-Friendly Chunk Scheduling

Whereas the natural traffic locality was observed [29], Picconi *et al.* [42] developed an ISP-friendly chunk scheduling technique to proactively promote the locality degree. The core theory behind the technique is that chunks are allowed to be downloaded from distant peers only when the video data buffer is in danger of being drained. To implement this, the authors proposed a two-level partnership: a peer maintains two partner lists, one of the peers from the same ISP and the other of the peers from the remaining ISPs. In most cases, a peer requests chunks from the peers in the former partner list. As the buffer window containing the active chunks advances, if a chunk reaches the first half of the window but is still missing, an early starvation signal (ESS) will be generated. Detecting the ESS, the chunk scheduler will assign more workload to the peers in the latter partner list. Conversely, if no ESS is detected during a predefined interval, the workload assigned to the peers in the latter partner list will be decreased. The workload adjustment adopts a multiplicative-increase/multiplicative-decrease strategy. With this scheduler, the amount of chunks downloaded from distant peers dramatically decreases while the streaming quality remains roughly unaffected.

Furthermore, the authors analyzed the efficiency of their heuristic. The cost of inter-ISP and intra-ISP traffic are denoted $\overline{c}$ and $\underline{c}$, respectively while d_i and λ_i respectively represent the required download rate to receive an uninterrupted stream and the part originating from the stream server. Then, the aggregate rate into an ISP from the stream server $\lambda_J = \sum_{i \in J} \lambda_i$, where J indicates an ISP. Hence, the cost of satisfying all the peers in one ISP with adequate download rates can be formulated as

$$c(J) = (d_i - \lambda_J)\overline{c} + d_i(n(J) - 1)\underline{c}, \tag{4}$$

where $n(J)$ represents the number of peers in J. The equation implies that theoretically, only d_i inter-ISP rate is required to import a complete copy of the stream into an ISP while the remained download rate can be supplied locally. Afterwards, the overall cost of all the ISPs is derived as

$$c^* = \sum_J c(J) = d_i(|J| - 1)\overline{c} + d_i(n - |J|)\underline{c}, \tag{5}$$

where n is the number of all the peers and $|J|$ is the number of the ISPs. Finally, the authors proved that if the peers are uniformly distributed among the ISPs and the ingress rates from the stream server into each ISP is equal, the traffic cost can be no larger than $c^* + o(d_i|J|\overline{c})$; otherwise, the cost is restricted within $c^* + O(d_i|J|\overline{c})$.

Meanwhile, Magharei *et al.* [36] proposed a similar chunk scheduling, named *OLIVES*. The authors started from theoretically analyzing the impact of traffic locality on P2P streaming. Based on the scheduling proposed in their prior study [35], they determined the theoretically maximal and the feasible amounts of cross-ISP connections to ensure the streaming quality. Then, it was pointed out that the feasible amount of connections still cannot preserve the streaming quality due to connection misallocations caused by suboptimal scheduling. Hence the minimal required redundancy of the cross-ISP connections was derived. With this knowledge, a two-tier scheduling strategy was proposed. The inter-ISP scheduling is enforced on some boundary peers, which are designated to download a complete copy of stream. The local tracker for each ISP coordinates the boundary peers to avoid misallocation. Meanwhile, all the peers use an intra-ISP scheduling that aims to disseminate the streams from the boundary peers to the internal ones.

Another similar work from Tomozei *et al.* [47] proposed an intriguing flow control theory based on the *Implicit-Primal-Dual* scheme. With this theory, a distributed rate allocation based on local information was devised. Incorporating network coding, the algorithm can achieve near-optimal cross-ISP traffic restrictions. While the algorithm has the desirable property of avoiding a centralized bottleneck, it suffers from a slow adaptation to any load dynamics. For example, the presented 5-node network requires roughly 5,000 time units to stabilize (even when no changes are induced in the network topology).

3.2.3 ISP-Friendly Peer Selection

While the three solutions introduced above [36, 42, 47] focus on optimizing chunk scheduling, Shen *et al.* [45] only modified the peer selection mechanism, which affects few of the internal components of a P2P application, resulting in a solution that is lightweight to implement on real systems.

The intuitive method uses biased peer selection, which was first proposed for file sharing applications. In this method, the number or the ratio of the partners from the same ISP is predefined. However, this hard configuration may cause the QoS decline for streaming systems. Fig. 2 shows the tradeoff between traffic locality and streaming quality guarantee. A peer needs to download at rate of 20 to receive an uninterrupted stream. If 80% of the partners must be selected from the same ISP, as the partnerships in figure (a) show, the peer cannot satisfy its streaming demand. In contrast, the partnerships in (b) can achieve the rate requirement by sacrificing a portion of the traffic locality. Network conditions such as shown for AS 1 are possible. For example, the bandwidth

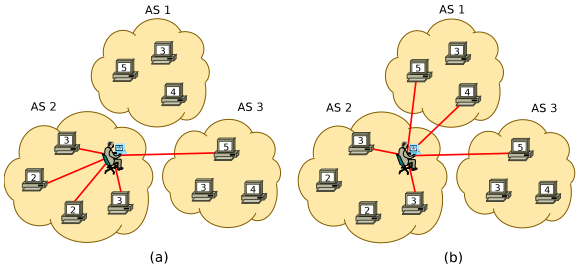

Fig. 2 Demonstration of the tradeoff between traffic locality and streaming quality guarantee. The number in each PC icon means the upload capacity. Assuming download rate 20 is required for an uninterrupted stream, the partnership (a) cannot satisfy the rate requirement while the partnership (b) can.

provided to the subscribers of some ISPs is poor or too many peers compete for the peer upload capacity locally.

Recognizing this tradeoff, Shen *et al.* [45] proposed an adaptive peer selection mechanism which adapts the locality degree according to the streaming quality feedback: if the average streaming rate provided by topologically close partners is smaller than that of all the partners, then during the next round the neighborhood update grows less biased to topologically close partners; otherwise, the update grows more biased to these partners. Furthermore, the neighborhood update is conducted in the distributed manner to avoid overwhelming the tracker server. Each peer only maintains a partial view of the participating peers and knows the network distance to the partners in this view. To introduce possible close peers to each other, peers utilize biased gossip, *i.e.*, they prefer a close partner to communicate with and acquire that partner's close partners. This is based on the knowledge that if peer a is close to peer b and peer b is close to peer c, peers a and c are close with high probability. Therefore, with the aforementioned techniques, a QoS-preserved, local-clustering overlay can be achieved in a adaptive and dynamic manner.

3.3 Hybrid P2P Infrastructures

As mentioned in Section 2.2, the two mainstream system architectures for P2P streaming have very much complementary pros and cons. Therefore, it is natural to believe a combination of both may achieve the best of both worlds. We discuss here a few proposals aiming at improving P2P streaming performance by using a hybrid system architecture that marries push and pull.

The idea of a hybrid P2P streaming appeared almost at the same time when the mesh-based pull system started to attract the attention of academia [57]. In general, most proposals since then take a mesh-based pull system as the basis, and their common goal is to "distill" push trees out of the mesh structure and to optimize the performance of these trees. We will focus on a few contributions that deal with the following three issues on the push trees:

- Push trees with stable links [55]
- Push trees with stable peers [48]
- Push trees with optimal delay [34]

3.3.1 A Pull-Push Hybrid Approach

As commonly recognized in the P2P community, a tree-based architecture has excellent performance in terms of delay of media delivery, if the overlay structure can be held stable. In the approach proposed by Zhang *et al.* [55], the link stability is considered as the criterion to switch from (mesh-based) pull to (tree-based) push.

The authors of [55] started by investigating the mesh-based pull approach, based on simulations (including experiment on PlanetLab) and analysis. The protocol under scrutiny does not come from any commercial product such as PPLive, but it has most of the crucial components of a pull system, including in particular:

- Periodically buffer maps are exchanged among neighboring peers.
- Media contents are pulled from peers that are supposed to have them (according to the up-to-date buffer map).
- Unreliable transportation mechanisms (UDP or TCP with very short buffers) are used for media content delivery.

The outcome of this investigation confirms that a pull system is near-optimal in terms of bandwidth efficiency, in the sense that the upload capacity of peers is almost fully utilized and maximum system throughput is nearly achieved. Another observation made by the authors, albeit with a lack of a concrete justification, is that, though the mesh-based pull system is initially self-organized into an unstructured random mesh, the steady state of the system (if it is ever reached) is actually a set of packed spanning trees.

Based on the aforementioned observations (in particular the second one), the hybrid system takes the following steps to smoothly meld push with pull.

1. Consecutive media chunks[8] are first grouped into *chunk groups*, and consecutive chunk groups are further grouped into *chunk parties*.
2. The buffer map is explicitly requested by the peers, instead of periodically pushed to peers in a common pull system. This allows the receivers to control the switching between push and pull.
3. Peers initially pull chunks from other peers according to the received buffer map. However, if a chunk belonging to the first chunk group of some chunk party is received, the receiver will send the sender a sub-stream

[8] The original literature uses the term "packets", but they are actually super packets that is packed with 1250-byte streaming data. To unify the terminology, we use "chunks" to refer to these super packets.

subscription, which allow the rest of the chunks in the same party to be pushed directed to the receiver. This is the *first stage push*.

4. Once over 95% of the received packets at a peer are pushed directly from its neighboring peers, this peer will stop pulling buffer maps. This brings it into the *second stage push*. If the chunks delivery ratio drops below 95% or a peer that provides a significant among of media data quits, the receiving peer will fall back to the first stage push as described in the previous step.

The thresholds applied to decide whether to switch to a certain stage of the hybrid operations serve as implicit tests that identify the stability of chunk input from certain links to a given peer. Once certain stability level is reached, more pushes kick in to reduce the protocol overhead. Otherwise pulls are invoked to cope with system dynamics. As we discussed in Section 2.3, a similar technique was later applied to the new version of *CoolStreaming* [25].

One potential problem with this proposal is the stability of the system itself. As the peer stability is not tested, it is possible that peer churns may lead to frequent switches between different stages, resulting in extra overhead and delay in coping with data outages. In addition, as also observed in [55], pull systems can have a significant playback delay: 25 seconds for a relatively small system (size ranging from 0 to 10,000). Therefore, the push trees used in this proposal is not delay-optimal.

3.3.2 A Two-Tier Hybrid Approach

In order to avoid constructing volatile trees, it is the peer stability (rather than the link stability) that should be tested. This is the proposal presented by Wang *et al.* [48]. The general idea is to construct a two-tier media streaming system, with a tier-1 backbone that applies a push system in trees consisting of stable peers and a tier-2 pull system to accommodate instable peers.

The first major contribution of [48] is a study that demonstrates the significance of stable peers and an algorithm to identify them. Using a trace-driven study based on PPLive, the authors of [48] conducted a statistical analysis on the significance of stable peers. According to their definition, peers are considered stable if their lifetime exceeds 40% of the observed period (about four hours). It stems from a well known observation that *the longer a node resides in a channel, the more it is likely to stay longer* [4]. The conclusion of this study is:

> *Although the fraction of stable peers is insignificant among the whole observed peer population (5.5% to 15.4%), they constitute a significant portion (*$>70\%$ *in every snapshots of the system.*

This conclusion is not hard to understand, as far as one realizes that the node lifetime follows a heavy tail distribution. Knowing the significance of the stable peers, a randomized algorithm is devise to identify them. The algorithm is based on the *s*-Index defined as

$$SI = \begin{cases} \frac{2s}{L-t} & \text{if } s \leq \frac{L-t}{2} \\ 1 & \text{otherwise} \end{cases} \quad (6)$$

where SI is the value of *s*-Index, s is the node duration in a session so far, t is its arrival time, and L is the session length. Basically, the *s*-Index is set to 1 if a peer stays $\frac{1}{3}$ of the residual session time since its arrival; otherwise the value increases linearly with the node duration s. Given an *s*-Index of a certain peer, a hard threshold H could be used to judge whether the peer is stable or not, but a randomized threshold is more robust against peer dynamics. In particular, a peer is identified as stable in the s-th time slots after its arrival with probability $2/[(L-t)(H-SI)+2]$; this produces a promotion probability that increases linearly over time.

The other contribution of [48] is a tree construction algorithm, LBTree, that organizes the stable peers into a tier-1 backbone. The two main ideas involved in LBTree are the following:

- Locating peers with higher stability closer to the media source, and involving the tree structure towards a balanced one with information exchanges.
- Establish *side links* to connect peers in different branches of a tree; a peer is connected with peers either at the same level or at higher levels, as far as they do not share any ancestor except for the root.

As the tree construction is based on pure heuristics, no guarantee can be promised either on throughput or on playback delay.

3.3.3 A Hybrid Approach with Delay Optimal Push

Though the two-tier infrastructure proposed in [48] appears to be promising, the drawback of LBTree motivates one to think about an optimal tree construction for the tier-1 backbone. Here we briefly discuss a recent proposal on building delay optimal push trees [34].

The tree construction algorithm presented in [34] is actually inspired by the principle of minimum delay chunk dissemination proposed in [28]. We refer to Section 2.4.2 for more detailed discussion on [28]. In a nutshell, the snowball streaming for single-chunk dissemination can be summarized as follows [28]:

> *After receiving a new chunk, a peer keeps pushing that chunk to other peers who have not received it, until every peer has received the chunk.*

By doing this, the number of peers receiving a certain chunk increases exponentially in time. Consequently, the mechanism produces a delay logarithmic in the system size, which is proven to be optimal. Unfortunately, though it is trivial

to implement this principle for single-chunk dissemination, it becomes highly nontrivial when it comes to multi-chunk dissemination (i.e., chunk streaming). Only the existence of such an algorithm is shown in [28], while the main contribution of [34] is to devise such an algorithm.

The first observation made in [34] is that the single-chunk dissemination procedure can actually be represented by an unbalanced push tree (called *snow-ball tree* or SBT), as shown in Fig. 3 for a 16-peer system. Basically, the peers

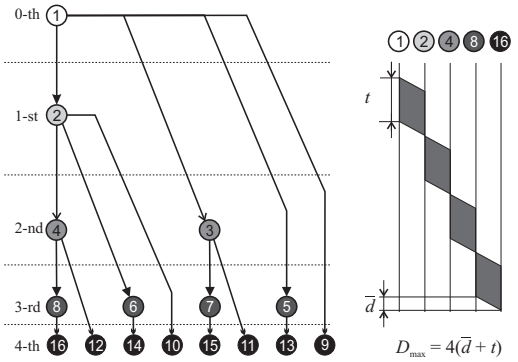

Fig. 3 The SBT of single chunk dissemination. We illustrate both maximum delay calculation and the definition of *peer level set* in the figure.

that receive the chunk in the k-th time slot are put at the k-th level of the tree. The geometric progression nature of R_k (the number of peers that receive the trunk at the end of kth time slot) guarantees that each peer at the k-th level definitely has a parent in the previous levels. As illustrated in Fig. 3, the maximum delay in the SBT is indeed logarithmic in the system size up to some constant.

In order to extend the SBT for multi-chunk streaming, one needs to achieve the following:

> *A set of SBTs such that, if the server takes turns to push chunks to their roots in a round-robin fashion, the minimum delay streaming is achieved in every involved SBT.*

Obviously, this is a scheduling problem. Given certain constraints on the schedulability of tree links belonging to a certain number of consecutive SBTs, the problem can actually be represented as a maximum independent set problem in a graph induced by those schedulability constraints. Fortunately, we do not need to resort to the solution technique to this NP-hard problem to address delay optimal multi-chunk streaming, thanks to the special structure of SBT. The major result of [34] is that, by periodically assigning peers to consecutive SBTs and by taking different periods for different levels of the trees, the resulting multi-SBT structure does achieve the design goal.

3.4 Coding Enhanced P2P Streaming

Whereas coding techniques (e.g., the H.263 standard) were widely used since the inception of video streaming over IP to mask the unreliable network layer, P2P media streaming incurs new issues that demand the application of novel coding mechanisms. On one hand, relaying on many low bandwidth peers (instead of a few high bandwidth servers) to deliver media contents requires a coding mechanism that collects sufficient media chunks before the playback deadline while avoiding redundant transmissions. On the other hand, the heterogeneity of the peers' upload bandwidth demands a flexible coding technique to deliver adaptive media quality based on the available bandwidth. To this end, *network coding* [1] and *layered coding* [44] are intensively used recently to improve the performance of P2P streaming. In this section, we discuss a few up-to-date proposals about applying these coding techniques, either separately or jointly, to P2P streaming.

3.4.1 Random Push with Random Network Coding

Network coding was initially proposed within an information theoretical framework [1], but its advantage in terms of throughput over conventional flow network has since been demonstrated under practical network settings (e.g., [15]), and in particular for P2P content distribution (e.g., [12]). The intuition behind this coding advantage is that, as network coding equalized the importance of different content blocks, the bottleneck resulting from the existence of "rare blocks" are eliminated. However, the requirements of P2P media streaming differs significantly from conventional content distribution, in that there is a stringent deadline for delivering each media chunk in order to maintain smooth playback. Therefore, it becomes less obvious whether network coding may still be advantageous for P2P media streaming, as the incurred non-elastic traffic demand not only long term throughput, but also instantaneous deliver rate.

In order to confirm the advantage of network coding in P2P streaming, Wang and Li [49] performed a fair comparison between using and not using network coding, based on a mesh-based pull system similar to PPLive. The following conclusions were drawn in [49]:

- Network coding is advantageous when the supply of the upload bandwidth barely exceeds the bandwidth demands in the session.
- Network coding maintains stable buffer levels when peers are volatile.
- The computation costs introduced by network coding are very low with typical media streaming rates.

Based on these observations, the authors were also motivated to redesign the P2P streaming strategies to better leverage on network coding. This led to the R^2 design [50], where

random (mesh-based) push are combined with random network coding, which we briefly discuss in the following.

In order to take full advantage of network coding, R^2 adopts a mesh-based push rather than pull, and the push is random in the sense that two distributions are used to control which chunk[9] is to be pushed to a downstream peer. There are three critical components of this random push:

- *Event triggered buffer map exchanges*: A peer sends its buffer map to its neighbors upon playing back a chunk or downloading a new chunk. The buffer map is either piggybacked with an outgoing transmission if possible, or sent separately. This serves as a timely notification of missing chunks and also as a stop sign of completed chunks.
- *Randomized push within priority region*: A priority region of length τ is set immediately after the playback time of a peer, and a chunk is uniformly (the first distribution) chosen to be pushed to other peers.
- *Randomized priority after priority region*: If no missing chunks are in the priority region for a downstream peer, the pushing peers will choose chunks out of the priority region. The priority of such a choice follows a Weibull distribution with PDF $\frac{k}{\lambda}\left(\frac{x}{\lambda}\right)^{k-1} e^{-(x/\lambda)^k}$; this gives preference to chunks that are earlier in time.

Apparently, what R^2 adopts is not a pure push, as the buffer maps are still exchanged, which is similar to a pull mechanism. However, it does differ from mesh-based pull in that the sending of a chunk is randomized, in contrast to the deterministic sending in response to a pull. Note that, even though R^2 appears to have a higher buffer map exchange rate than traditional push mechanisms, it actually has a lower overhead due to the use of much larger segments (which is in turn the result of applying network coding, as we will discuss soon).

The networking coding technique used by R^2 is the Chunked Codes proposed by Maymounkov *et al.* [38]. Basically, each chunk is further divided into n blocks, and the coding is only applied with each chunk. The goal is mainly to reduce the coding complexity. Upon choosing a chunk p to push, the peer generates i.i.d. coding efficient $[c_1^p, c_2^p, \cdots, c_m^p]$ with $m \leq n$ and randomly chooses m blocks $[b_1^p, b_2^p, \cdots, b_m^p]$ to produce a coded block x as follows:

$$x = \sum_{i=1}^{m} c_i^p b_i^p$$

For the need of decoding, the coding efficient has to be sent along with the coded block in some form. R^2 suppresses this overhead by only sending the seed for generating the i.i.d. sequence $[c_1^p, c_2^p, \cdots, c_m^p]$ rather than the sequence itself. The decoding is done through Gauss-Jordan elimination, which can be performed progressively while receiving new coded blocks. Upon receiving a total of n blocks, the original segment can be recovered. Applying network coding also allows R^2 to use UDP instead of TCP (required by most mesh-based pull systems) as its transport protocol. This stems from the inherent error resilience of random network coding, and using UDP helps to further reduce the streaming overhead.

[9] It is actually termed "segment" in [50].

3.4.2 Pull Scheduling for Layered Streaming

Bandwidth heterogeneity has been a "pain" to P2P streaming systems since the very beginning, as many protocols that should theoretically work well under the assumption of homogeneous bandwidth among all peers have a substantial performance degradation when facing real-world realities. A rather natural solution to this problem to to adapt the service quality to the available bandwidth: peers that have limited bandwidth would get a lower quality service (e.g., in terms of video bit rate). The enabling technology to achieve this goal can either be *layered coding*[10] (LC) or *multiple description coding* (MDC), but LC is generally preferred due to its higher coding efficiency. Quite a few proposals have been concerned with applying LC to P2P streaming (e.g., [11,43]), but we only focus on a very recent one [53].

The main challenge brought on by adopting LC in P2P streaming is a need for more complex chunk scheduling algorithms. Whereas a scheduling algorithm for non-layered streaming only is concerned with throughput maximization, scheduling layered streaming has to take other constraints into account. For example, scheduling a higher layer to be delivered in addition to a lower layer may render a vain transmission if packet loss occurs in the lower layer. Therefore, the LayerP2P proposed in [53] aims to tackle the following three problems:

- Fetching as many data blocks as possible with the latest playback times (maximizing instantaneous data rate).
- Ensuring the integrity of video chunks by subscribing to appropriate layers upon approaching their respective playback time.
- Compensating the absence of blocks in subscribed layers when facing a sudden bandwidth drop.

Motivated by all these requirements, LayerP2P comes with a 3-stage model for data scheduling. In particular, a video steam is encoded into several layers by LC, and each layers is partitioned into chunks (similar to a traditional pull system). Layers and chunks are identified by their respective IDs. Each node has a buffering window that contains the interested chunks, and the window slides forward periodically. Apart from the left-most part of the window that have just been played or will be played very soon, the rest of the

[10] It is also known as *scalable video coding* (SVC), which is an extension of H.264/MPEG-4 AVC video compression standard.

window is partitioned into 3 stages: remedy stage, decision stage, and free stage.

- *Remedy stage*: Chunks in this stage are very close to their playback time, so they should get very high priority compared with others in terms of bandwidth allocation. The tricky part is to decide the width of this stage: a too small value does not give enough time to request the missing chunks, whereas a too large value leads to a large startup latency. LayerP2P sets this value to be proportional to the bandwidth variation ratio, i.e., the variation divided by the bandwidth after drop, which should allow just enough time to request the missing chunks.
- *Decision stage*: The decision to be made during this short stage is which layer(s) to subscribe to. Basically, the decision is made based on the availability (in the current peer and also its neighbors) of the chunks in a certain layer. A layer is subscribed only if all its lower layers are subscribed. In addition, if the availability goes beyond a certain threshold, the layer is subscribed for sure, otherwise a randomized choice is made based on probability. We omit the detailed discussion on how to measure the availability here.
- *Free stage*: This is a stage is very similar to a traditional pull system, where chunks can be freely pulled from neighbors. However, the pull is confined by the current layer subscription: no chunks beyond the subscription are pulled.

3.4.3 Layered Streaming with Network Coding

Since both network coding and layered coding may benefit P2P streaming systems from different aspects, a desired development would be to combine them such that we may obtain "the best of both worlds." However, as network coding equalizes video chunks while layered coding prioritizes video contents, combining them is actually a challenge. We hereby discuss a very recent proposal, *Chameleon* [40], that tackles this challenge.

As *Chameleon* adopts SVC (rather than a theoretical coding algorithm) as its LC component, the protocol is specified according to the SVC standard. An SVC video consists of a sequence of coded picture sets, with each set termed *group of pictures* (GOP) and each coded picture in a set termed *access unit* (AU). Note that each GOP consists of one base layer AU and higher layer AUs that allow refinements. In order to adapt to the streaming system and also to facilitate further combining with network coding, *Chameleon* applies a three-stage data structure. At the first stage, an SVC stream is divided into *segments*, which consists of an integer number of GOPs. At the second stage, the segment is further divided into *chunk*s. However, this is not just a simple partition, as the AUs of all the GOPs in a segment are re-arranged such that each chunk contains AUs from the same layer. In particular, chunk 1 of each segment contains the base layer AUs and is necessary for every peer, while other chunks are delivered according to different subscription requirements. Apparently, this second stage data structure is dedicated to facilitate the underlying mesh-based pull streaming system. Finally, each chunk is divided into blocks in the third stage, and random network coding is applied to blocks within each chunk. As this stage is very similar to that of R^2 (see Section 3.4.1), we do not discuss it in details.

In general, *Chameleon* can be considered as a mesh-based pull system complemented by network coding and SVC. Therefore, it has most features of a traditional pull system. However, new components are also introduced due to the integrating of network coding and SVC. We list a few in the following:

- *Quality adaptation*: *Chameleon* uses two thresholds, add_threshold and drop_threshold to control the quality level of the video requested by a peer. If the buffer level goes below (resp. beyond) drop_threshold (resp. add_threshold), the quality level will be decreased (resp. increased). This is intuitive as low (resp. high) buffer level is a sign of bandwidth scarcity (resp. abundance) and hence the peer should obtain a lower (resp. higher) quality of service.
- *Receiver-driven peer coordination*: The pull message sent by a receiver differs from a traditional one in that the request is at the layer level (rather than at the chunk level) and starts from the lowest available layer. In addition, several senders may receive the same pull message, and they collaboratively serve the receiver (which is possible due to the use of network coding). Similar to R^2, a stop notification should be sent to the senders upon finishing decoding to avoid redundant transmissions.

3.5 Streaming Across Heterogeneous Networks

With the growing adoption of 3G networks and the upgrade of mobile handsets, the mobile video market is rapidly expanding. Also, Cisco published a forecast about global mobile data traffic [10]:

> *Globally, mobile data traffic will double every year through 2014, increasing 39 times between 2009 and 2014. Almost 66 percent of the world's mobile data traffic will be video by 2014.*

Aware of the trend, the major Internet video companies have provided client software on different mobile platforms, such as Android, iOS, RIM, Symbian.

Nevertheless, the Internet video companies that rely on P2P architectures, such as PPLive and PPStream, have been slow to embrace the mobile market. One possible reason is that even though the P2P paradigm has demonstrated great

success in delivering videos over the wired Internet, its extension to mobile wireless networks is hindered by particular network conditions and the limited capacity of mobile handsets. Several prominent challenges are listed as follows:

- Compared to wired networks, the bandwidth of wireless networks is still limited. As the number of mobile video users grows, the bandwidth of access points (AP) and cell towers is quickly exhausted. Furthermore, the wireless link quality can be unstable and is vulnerable to interference. This complicates the streaming quality assurance.
- Video streaming is a bandwidth-intensive application, which consumes a considerable amount of energy through the radio module usage. The P2P architecture worsens the energy consumption situation because the peers have to take the additional task of uploading video data.
- Mobile handsets can move (*e.g.* on a bus or train). Unlike devices connected to the wired Internet, handsets change their connected AP from time to time so that their IP addresses are dynamic. This difference may complicate the traditional method of using IP addresses to distinguish peers. Moreover, connections between peers become fragile, and are likely to be broken during AP handovers [39].
- From a non-technical perspective, mobile users are sometimes charged for data usage. Hence, there may be reluctance to forward a stream.

We focus on a few studies dealing with these challenges [24, 31, 46] in this section.

3.5.1 P2P-Friendly Infrastructure Upgrade

As mentioned earlier, APs tend to become bottlenecks when the number of wireless users increases. Tan *et al.* [46] noticed that the current wireless networks (WiFi in their study) are not P2P-friendly because, unlike the traditional clients under a client-server architecture, peers need to upload massive amounts of data. The authors summarize the problems of significant upload traffic in wireless networks as follows:

- An AP is likely to be congested with the upload traffic and the dependent peers in other parts of the Internet may be affected.
- The channel sharing nature of WiFi indicates that peers' upload traffic can congest their download link as well, interfering with the streaming quality itself.
- The extra upload traffic can increase the number of transmission errors, degrading the signal quality and the streaming quality as a consequence.

Recognizing the challenge from upload traffic, Tan *et al.* [46] set out to reduce the upload traffic of P2P live streaming over wireless networks. They began by conducting a measurement study to understand the traffic patterns of wireless local area networks (WLAN) and found that

> *On average, for P2P-based streaming, there are more than 80% duplicated data packets in successive down- and upload data streams.*

The high duplication rate opens a significant opportunity of reducing the upload traffic. To exploit this potential, the authors propose a caching middleware deployed on an AP. The middleware caches a copy of the downloaded packets, which are identified with the Rabin fingerprinting scheme. The peers connected to the AP do not upload the entire data packets, but upload an identity tag, which is small in size. Therefore, the upload traffic within the WLAN greatly decreases. When the AP receives a tag, it will find the corresponding packet and send it to the destination. The authors implemented a prototype of this solution, named *SCAP*, The experimental results show that *SCAP* improves the throughput of the WLAN by up to 88% with a decrease of the response delay to the peers outside the AP as a bonus.

3.5.2 Collaborative Video Streaming Among Mobiles

One aspect Tan *et al.* [46] did not exploit is the multi-radio feature of mobile devices. With the evolution of mobile technology, increasingly mobile handsets are equipped with more than one network interface. In addition to the master interface for telecommunication (*e.g.*, 3G), the handsets also support WiFi and Bluetooth as secondary interfaces for short-range data exchanges. Witnessing this hardware trend, Leung *et al.* [24] proposed an innovative *collaborative streaming among mobiles* (*COSMOS*) protocol, which is similar to a traditional P2P architecture but leverages one of the secondary interfaces, incorporates the simulcast technique and adopts the multiple description coding (MDC) technique.

Overall the *COSMOS* protocol operates such that some peers, denoted as *pullers*, download a sub-stream from a content provider through the master network interface, and the obtained sub-stream is then re-broadcast among peers through the secondary network interface. The protocol aims to utilize the free-of-charge secondary radio interface to reduce the bandwidth consumption cost of the master interface. The experimental results exhibit an appealing bandwidth usage reduction of the master network interface (over 50% in most cases).

The major challenge of this solution is to ensure that every peer is reachable through broadcasts via the secondary network interface. The authors proved that this problem is NP-hard [24]. Therefore, they proposed two of heuristics – one that is peer-density-aware while the other is not. The peer-density-unaware solution is straightforward. Whenever a peer obtains a packet, it checks the time-to-live (TTL) flag. If the TTL is greater then zero, the peer will reduce it by

1 and re-broadcast the packet. Otherwise, there will be no forwarding. The TTL is initialized by the pullers to fix the broadcast scope (2 in their study).

However, the authors detected a problem with the unaware heuristic in the non-uniform peer density scenario [24]:

> *When user density is high, using a fixed broadcast scope leads to high packet redundancy in the wireless channel. On the other hand, when user density is low, it may be beneficial to extend the broadcast scope to reduce the streaming cost.*

Hence the authors improve the protocol by allowing a peer to re-broadcast a packet only when few of its neighbors have received it before. To enable this protocol, peers exchange data availability periodically. To quantify the potential gain of a re-broadcast, a peer computes the fraction of the peers that have obtained the same sub-stream from it or have not received the sub-stream yet. If the fraction is above a certain threshold, the peer will re-broadcast the packet.

Another challenge of this solution is to balance the cost of streaming through the master network interface. To achieve fairness, the puller can stop pulling from the content provider when it has received all the sub-streams. Before the puller stops, it will broadcast a *switch* flag embedded into its packets. When a non-puller receives the flag and no longer receives the sub-streams for some period, it will become the puller. According to the experimental results, the *COSMOS* protocol achieves a satisfactory fairness.

3.5.3 Optimization versus Energy Consumption

While Leung *et al.* [24] proposed a collaborative streaming over mobile wireless networks, they omitted the optimization of the handsets' energy consumption. To prolong the battery life, Liu *et al.* [31] proposed an energy-aware collaborative streaming method that incorporates a burst transmission scheme. The general idea is similar to that of Leung *et al.* [24]: the participants of a cooperative group alternately download the video stream in terms of a series of bursts through a *wireless metropolitan area network* (WMAN), and then broadcast the stream to others in the group through a WLAN. The energy saving originates from the different data rates: the rate of WMAN is usually smaller than that of WLAN due to the larger area to cover. Therefore, as Fig. 4 presents, the idle time between two sequential bursts is longer in the WLAN environment if the stream is transmitted at full speed, and longer idle time indicates less energy consumption. The simulation-based experiments show that the method proposed in [24] can achieve a high energy savings up to 70%.

The approach faces two technical issues. The first is how to elect an agent to download the stream from the WMAN. The method should be robust and fair. To achieve this, the authors devised a distributed algorithm:

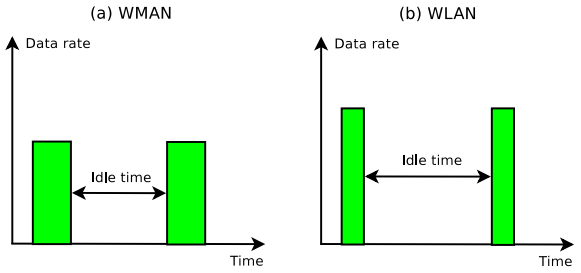

Fig. 4 Visualization of the data rate impact on the length of idle time.

1. Each participant of the cooperative group maintains a contribution list which records the accumulated data quantity of the stream relayed to each participant over the WLAN.
2. Subsequently the sender's contribution is updated on all the participants during each burst round. The lists on all the participants are supposed to be consistent.
3. The list is then sorted. The participant who relays the least amount of data is designated as the agent to download the stream from the WMAN in the next burst round.
4. Apart from the agent, the next k participants who relay the least amount of data are designated as the backups, in case that the agent leaves or fails before completing the next burst round.

This algorithm ensures that the workload enforced on each participant is similar and the communication protocol is resilient to participant dynamics.

The second issue is how to synchronize the wake-up time. In other words, the participating handsets need to know the start of each burst round. The video delivery over WMAN uses the MPEG-2 Transport Stream (TS) standard. The TS packets embed the information about the start time of the next burst round and are sent to the agent. When the agent receives the time, it broadcasts this information to others in the cooperative group over the WLAN.

One bonus benefit of this communication protocol is the quick channel switching. As a WLAN covers a smaller area than a WMAN, the transmission delay is less significant over WLAN. Moreover, the collaborative nature enables a newcomer to quickly get a unicast stream from the group instead of waiting until next burst round. The experimental results demonstrate an up to 98% reduction of the channel switching delay.

3.6 Media Streaming Beyond 2D Video

Using the P2P paradigm to stream media beyond traditional 2D video is another interesting topic. In recent years, some new applications, such as multiview video and networked

virtual environments[11] (NVE) have emerged. These applications may consume much more bandwidth than traditional 2D videos. Let us examine multiview video as an example:

> *Even after state-of-the-art compression, multiview representations are very data intensive: 38dB PSNR at about 5 Mbps is a common operating point for a 704 × 480, 30fps, 8 camera sequence with MVC encoding [23].*

Such a considerable bandwidth consumption is more likely to exhaust content providers' bandwidth resources with a client-server architecture. Therefore, since the P2P paradigm has exhibited great success in delivering 2D video, a number of studies have extended its application to beyond-2D media streaming. Here we introduce three representative studies of this category:

- Multiview video streaming [23],
- 3D mesh object streaming [7],
- 3D content streaming in NVE [17].

3.6.1 Multiview Video Streaming

Multiview (or free-viewpoint) video streaming is a new multimedia application which allows the user to interactively control the viewpoint and dynamically generate new views of the scene. To obtain realistic views for free-viewpoint rendering, multiple cameras are set up with careful calibration. In addition, efficient coding techniques are required to achieve good video compression rates – the codec used is often compatible with the H.264/MPEG-4 AVC standard. The part that concerns us in this survey, *i.e.*, media streaming, also becomes more challenging because of the higher bandwidth consumption and more interactivity.

Kurutepe *et al.* [23] conducted a preliminary study of using the P2P paradigm to streaming multiview video. They investigated the approach of a multi-tree based overlay incorporating the MDC technique proposed by Castro *et al.* [6], and discovered the similarity to the P2P multiview video streaming if each captured view is encoded independently. Therefore, as shown in Fig. 5, the authors proposed a solution that a multi-tree overlay is constructed for multiview video dissemination, and each tree is responsible for one captured view.

It is noteworthy that this study was among the earliest on this topic, and it focused on demonstrating the feasibility of P2P multiview video streaming. However, the proposed solution is far from satisfactory. For example, a subset of views are enough to support the users' present viewpoint while downloading the remaining views is unnecessary. Therefore, an interesting problem is how to selectively stream multiview video with a P2P architecture.

[11] Unlike the traditional massively multi-player online game (MMOG) where the content are pre-installed or added by patches, modern NVEs such as Second Life allow users to create objects, scenes, *etc.*. The content is streamed to users on demand, usually when it is within the users' region of interest (ROI).

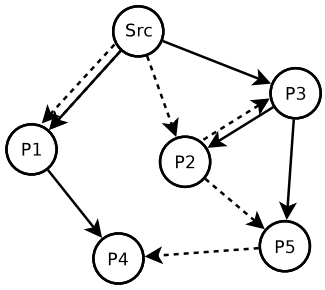

Fig. 5 A sample multi-tree overlay to disseminate two independent views where each peer is an internal node of just one tree.

3.6.2 3D Mesh Object Streaming

Another type of state-of-the-art multimedia content is polygon meshes, which use a collection of vertices, edges and faces to model 3D objects. Recently, mesh object creation techniques have experienced rapid development.

> *For example, the Stanford model of the David statue, containing 28 million vertices and 56 million triangles, is still 70 MB in size even with state-of-the-art compression and requires around 10 minutes to download at 1 Mbps [7].*

Potential application contexts of mesh objects may be virtual art galleries, virtual museums, virtual auctions, geometry video, *etc.* Users may inspect objects with interactive operations: zooming in to view fine details, rotating an object back-and-forth to examine all facets. One potential challenge in transmitting mesh objects is still the heavy bandwidth consumption, even though progressive streaming techniques can be leveraged.

To overcome the challenge, Cheng *et al.* [7] proposed a P2P-based solution. They introduced a new content organization method which is different from that of P2P video streaming. The vertices are organized in a tree structure and are assigned an ID each. The child vertices are always the refined representation of the parent vertex. The technique adopts a receiver-driven approach where the receiver determines which vertex is required according to the user's viewpoint and requests it from the sender. The authors also summarize the other differences between progressive mesh streaming and live video streaming:

- Unlike live video streaming which has stringent playback deadlines, mesh streaming is a delay-tolerant application.
- While the peers in a live video streaming session have almost synchronized data access patterns, peers randomly inspect mesh objects.

– The session length of the mesh streaming is often much shorter than that of live video streaming.

These differences imply that a good content discovery mechanism is more crucial to mesh streaming. The authors considered two methods. The first one employs a lookup server that is set up to record the vertex ownership. For a small group of users the overhead is acceptable and this method is able to achieve high efficiency owing to its global knowledge. However, as the number of users grows, the server tends to be a bottleneck. Therefore, to ensure scalability, the authors also proposed a hierarchical lookup method, where one of the owners of a vertex is elected as the leader that traces the ownership of this vertex. The leadership is also organized in a tree structure as well, rooted at the lookup server. A peer first finds the leader, who will in turn select a sender for the peer. That peer then requests the vertex from the sender, and it hence becomes a member of the owner group of this specific vertex.

The authors conducted trace-driven simulations the results of which show that their proposed solution reduces the server overhead by more than 90%, keeps control overhead below 10%, and achieves low average response time.

3.6.3 3D Content Streaming in NVE

While Cheng *et al.* [7] focused on single 3D mesh object streaming, NVEs actually contain a massive amount of 3D data of various types, such as meshes, textures, animations, and scene graphs. With the growing size of 3D content, the pre-installation and patching method becomes unsuitable for setting up NVEs on PCs. Hu *et al.* [17] use two examples to corroborate this point:

> *The social MMOG Second Life*[12] *depends on 3D streaming to deliver over 34 TB of user-created models, textures and behavior scripts.*
>
> *Google Earth*[13] *and NASA World Wind*[14] *currently consist of terabytes of data (70TB and 4.6TB, respectively).*

These systems resort to streaming to make the content accessible to users on demand. P2P is again considered as a good paradigm for content delivery in NVEs. Hu *et al.* [17] proposed such a prototype, named *FLoD*, discussing comprehensive design issues. To begin with, the authors analyzed the differences among live streaming, video-on-demand (VoD) and NVEs, which are summarized in Table 1.

The authors also discussed a key design issue, *i.e.*, how to construct a dissemination overlay. One of the implied assumptions in NVEs is that logically-close peers may own and need similar content. Hence it is better to construct an overlay according to peers' logic locations, and the *VON* overlay, proposed in an early study [16], was leveraged here. The *VON* overlay organizes peers according to a Voronoi diagram, ensuring that each peer connects directly with its *Area of Interest* (AOI) neighbors (the interested reader is referred to [16] for detailed explanations).

Table 1 Summary of the differences among live streaming, video-on-demand (VoD) and NVEs.

	Live	VoD	NVE
Start position	same	arbitrary	arbitrary
Access pattern	linear	linear	non-linear
Transmission sequence	same	same	unique
Group switching	infrequent	infrequent	frequent

Additionally, there is a graphics module that decides which data is to be downloaded. Prefetching and prioritization are employed to improve the user experience. On the other hand, which peers to contact is decided by the networking module. Before sending out a request, the peer communicates with its neighbors first to understand the content distribution. Then, the node randomly chooses among the peers that own the requested data to balance the workload. The downloaded data will be locally cached to serve other peers.

The authors evaluated their solution through both real-world and simulation-based experiments. The results show that the workload significantly drops with the P2P solution, and that both the fill ratio and the latency, which correlate with the QoS, are also improved. One problem of the solution is the high protocol overhead of the *VON* overlay, which grows logarithmically with the peer population.

4 Conclusions and Challenges

In this paper, we surveyed the latest developments in P2P media streaming, covering a wide range of topics: modeling- and measurement-based system performance analysis, traffic localization, advanced network coding techniques, hybrid overlays and heterogeneous networks, beyond-2D media streaming. Whereas numerous technological advances in P2P streaming have been made in the past few years, there are still several major problems that deserve further attention and investigation.

One of the major obstacles of making P2P streaming a well accepted Internet service is its contention with ISPs. Although recently proposed traffic locality techniques (see Section 3.2 for details) have demonstrated the promising confinement of cross-ISP traffic, it is not well-understood yet whether traffic locality may have a positive impact on the other two stakeholders, *i.e.*, the content providers and end users. For example, since the delay between topologically close peers is usually small, will the video lag decrease with

[12] Second Life, http://secondlife.com/
[13] Google Earth, http://earth.google.com/
[14] NASA World Wind, http://worldwind.arc.nasa.gov/

the traffic locality techniques? As one of the few investigations, Huang *et al.* [18] used a simple scenario to show that ISP-friendly peer-assisted VoD may still benefit (at least partially) content providers by reducing their operating costs. However, as the evaluation is based on a conservative inference of AS relationships, the evaluation provided stayed at a qualitative level. A quantitative evaluation will be necessary to better understand how to strike a balance among these three stakeholders.

For a given P2P system, especially those involving a hybrid overlay infrastructure, numerous system parameters are involved. By far, there is a lack of understanding on the fundamental trade-off among different system parameters, as most modeling or measurements papers are focusing on a small set of parameters. This is particularly important for systems operating under adverse network conditions: though we know some systems tend to perform better than some others [2], but it is not yet clear why that is happening. One hindrance to the development on this aspect is the lack of open platforms: many current investigations are done by taking the proprietary platforms as black boxes and infer information through their input and output (see Section 3.1.1 for some discussions). Of course, open platform would allow selfish behaviors to interfere with the system performance. Fortunately, the community is on the way to tackling this issue (e.g., [32]), though there are still a lot to be accomplish on this direction.

Although it is commonly agreed that a hybrid CDN-P2P may result in good performance, it would often be very costly for small startups to build their own CDN infrastructure from scratch. Fortunately, resorting to cloud services (e.g., Amazon S3) may allow content providers to construct their own CDNs without actually owning the hardware [41]. On one hand, we may face a lot of problems to build these so-called content delivery clouds, as cloud services are traditionally built for computing or storage instead of providing high bandwidth network access. On the other hand, content delivery clouds may greatly facilitate the beyond-2D media streaming in the sense that certain computing intensive operations (e.g., rendering) can be done in the cloud.

Last but not least, delivering media content in mobile and wireless environments will continue to be a challenge. Though the role and advantage of P2P systems in these environments may not be as prominent as in the wired Internet, it is reasonable to believe in their potential in balancing load and improving system scalability. While applying successful developments in the Internet (e.g., coding schemes) to mobile and wireless environments is definitely a way to proceed, we should also take the particularities of wireless communication systems into account, and construct the overlay network jointly with the underlying supporting mechanisms.

References

1. R. Ahlswede, N. Cai, S. Li, and R. Yeung. Network Information Flow. *IEEE Trans. on Information Theory*, 46(4), 2000.
2. E. Alessandria, M. Gallo, E. Leonardi, M. Mellia, and M. Meo. P2p-tv systems under adverse network conditions: a measurement study. In *Proc. of the 28th IEEE INFOCOM*, 2009.
3. S. Banerjee, b. Bhattacharjee, and C. Kommareddy. Scalable Application Layer Multicast. In *Proc. of the 8th ACM SIGCOMM*, 2002.
4. M. Bishop, S. Rao, and K. Sripanidkulchai. Considering Priority in Overlay Multicast Protocols under Heterogeneous Environments. In *Proc. of the 25th IEEE INFOCOM*, 2006.
5. T. Bonald, L. Massoulié, F. Mathieu, D. Perino, and A. Twigg. Epidemic Live Streaming: Optimal Performance Trade-Offs. In *Proc. of the 32nd ACM SIGMETRICS*, 2008.
6. M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowston, and A. Singh. SplitStream: High-Bandwidth Multicast in Coorperative Environments. In *Proc. of the 19th ACM SOSP*, 2003.
7. W. Cheng, D. Liu, and W. T. Ooi. Peer-assisted view-dependent progressive mesh streaming. In *Proc. of the 17th ACM Multimedia*, 2009.
8. Y. Chu, S. Rao, S. Seshan, and H. Zhang. A Case for End System Multicast. *IEEE J. on Sel. Areas in Communications*, 2002.
9. I. Cisco Systems. Cisco Visual Networking Index: Forecast and Methodology, 2009-2014. White Paper, 2010.
10. I. Cisco Systems. Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2009-2014. White Paper, 2010.
11. L. Dai, Y. Cui, and Y. Xue. Maximizing Througput in Layered Peer-to-Peer Streaming. In *Proc. of IEEE ICC*, 2007.
12. C. Gkantsidis and P. Rodriguez. Network Coding for Large Scale Content Distribution. In *Proc. of the 24th IEEE INFOCOM*, 2005.
13. X. Hei, Y. Liu, and K. Ross. Inferring Network-Wide Quality in P2P Live Streaming Systems. *IEEE J. on Sel. Areas in Communications*, 2007.
14. X. Hei, Y. Liu, and K. Ross. IPTV over P2P Streaming Networks: The Mesh-Pull Approach. *IEEE Comm. Mag.*, 2008.
15. T. Ho, R. Koetter, M. Medard, D. Karger, and M. Effros. The Benefits of Coding over Routingin a Randomized Setting. In *Proc. of IEEE ISIT*, 2003.
16. S.-Y. Hu, J.-F. Chen, and T.-H. Chen. VON: a scalable peer-to-peer network for virtual environments. *IEEE Network*, 2006.
17. S. Y. Hu, T. H. Huang, S. C. Chang, W. L. Sung, J. R. Jiang, and B. Y. Chen. FLoD: A Framework for Peer-to-Peer 3D Streaming. In *Proc. of the 27th IEEE INFOCOM*, 2008.
18. C. Huang, J. Li, and K. Ross. Can Internet Video-on-Demand be Profitable. In *Proc. of the 13th ACM SIGCOMM*, 2007.
19. Y. Huang, T. Fu, D.-M. Chiu, J. Lui, and C. Huang. Challenges, Design and Analysis of a Large-scale P2P-VoD System. In *Proc. of the 14th ACM SIGCOMM*, 2008.
20. T. Karagiannis, P. Rodriguez, and K. Papagiannaki. Should Internet Service Providers Fear Peer-assisted Content istribution? In *Internet Measurement Conference (IMC)*, 2005.
21. D. Kostić, A. Rodriguez, J. Albrecht, and A. Vahdat. Bullet: High Bandwidth Data Dissemination Using an Overlay Mesh. In *Proc. of the 19th ACM SOSP*, 2003.
22. R. Kumar, Y. Liu, and K. Ross. Stochastic Fluid Theory for P2P Streaming Systems. In *Proc. of the 26th IEEE INFOCOM*, 2007.
23. E. Kurutepe and T. Sikora. Feasibility of Multi-View Video Streaming Over P2P Networks. In *Proc. of the 2rd IEEE 3DTV*, 2008.
24. M.-F. Leung and S. H. G. Chan. Broadcast-Based Peer-to-Peer Collaborative Video Streaming Among Mobiles. *IEEE Trans. on Broadcasting*, 2007.
25. B. Li, S. Xie, Y. Qu, G. Keung, C. Lin, J. Liu, and X. Zhang. Inside the New Coolstreaming: Principles, Measurements and Performance Implications. In *Proc. of the 27th IEEE INFOCOM*, 2008.

26. J. Liu, S. Rao, B. Li, and H. Zhang. Opportunities and Challenges of Peer-to-Peer Internet Video Broadcast. *Proceedings of the IEEE*, 2008.
27. S. Liu, R. Zhang-Shen, W. Jiang, J. Rexford, and M. Chiang. Performance Bounds for Peer-Assisted Live Streaming. In *Proc. of the 32nd ACM SIGMETRICS*, 2008.
28. Y. Liu. On the Minimum Delay Peer-to-Peer Video Streaming: how Realtime can it be? In *Proc. of the 15th ACM Multimedia*, 2007.
29. Y. Liu, L. Guo, F. Li, and S. Chen. A Case Study of Traffic Locality in Internet P2P Live Streaming Systems. In *Proc. of the 29th IEEE ICDCS*, 2009.
30. Y. Liu, Y. Guo, and C. Liang. A Survey on Peer-to-Peer Video Streaming Systems. *Springer Peer-to-Peer Net. App.*, 2008.
31. Y. Liu and M. Hefeeda. Video streaming over cooperative wireless networks. In *Proc. of the 1st ACM Multimedia Systems*, 2010.
32. Z. Liu, Y. Shen, K. Ross, S. Panwar, and Y. Wang. Substream Trading: Towards an Open P2P Live Streaming System. In *Proc. of the 16th IEEE ICNP*, 2008.
33. Z. Liu, C. Wu, B. Li, and S. Zhao. Distilling Superior Peers in Large-Scale P2P Streaming Systems. In *Proc. of the 28th IEEE INFOCOM*, 2009.
34. J. Luo. Practical Algorithm for Minimum Delay Peer-to-Peer Media Streaming. In *Proc. of IEEE ICME*, 2010.
35. N. Magharei and R. Rejaie. PRIME: Peer-to-Peer Receiver-drIven MEsh-based Streaming. In *Proc. of the 26th IEEE INFOCOM*, 2007.
36. N. Magharei, R. Rejaie, V. Hilt, I. Rimac, and M. Hofmann. ISP-Friendly Live P2P Streaming. Technical report, University of Oregon, 2009.
37. L. Massoulié, A. Twigg, C. Gkantsidis, and P. Rodriguez. Randomized decentralized broadcasting algorithms. In *Proc. of the 26th IEEE INFOCOM*, 2007.
38. P. Maymounkov, N. Harvey, and D. Lun. Methods for Efficient Network Coding. In *Proc. of the 44th Allerton Conference*, 2006.
39. I. Moraes, M. E. Campista, L. H. Costa, O. C. Duarte, J. Duarte, D. Passos, C. V. de Albuquerque, and M. Rubinstein. On the impact of user mobility on peer-to-peer video streaming. *IEEE Wireless Comm. Mag.*, 2008.
40. A. Nguyen, B. Li, and F. Eliassen. *Chameleon*: Adaptive Peer-to-Peer Streaming with Network Coding. In *Proc. of the 29th IEEE INFOCOM*, 2010.
41. M. Pathan, J. Broberg, and R. Buyya. Maximizing Utility for Content Delivery Clouds. In *Proc. of the 10th WISE*, 2009.
42. F. Picconi and L. Massoulie. ISP Friend or Foe? Making P2P Live Streaming ISP-Aware. In *Proc. of the 29th IEEE ICDCS*, 2009.
43. M. Qin and R. Zimmermann. Improving Mobile Adhoc Streaming Performance through Adaptive Layer Selection with Scalable Video Coding. In *Proc. of the 15th ACM Multimedia*, 2007.
44. H. Schwarz, D. Marpe, and T. Wiegand. Overview of the calable Video Coding Extension of the H.264/AVC Standard. *IEEE Trans. on Circuits and Systems for Video Technology*, 17(9), 2007.
45. Z. Shen and R. Zimmermann. ISP-Friendly Peer Selection in P2P Networks. In *Proc. of the 17th ACM Multimedia*, 2009.
46. E. Tan, L. Guo, S. Chen, and X. Zhang. SCAP: Smart Caching in Wireless Access Points to Improve P2P Streaming. In *Proc. of the 27th IEEE ICDCS*, 2007.
47. D.-C. Tomozei and L. Massoulie. Flow Control for Cost-Efficient Peer-to-Peer Streaming. In *Proc. of the 29th IEEE INFOCOM*, 2010.
48. F. Wang, J. Liu, and Y. Xiong. Stalbe Peers: Existence, Importance, and Application in Peer-to-Peer Live Video Streaming. In *Proc. of the 27th IEEE INFOCOM*, 2008.
49. M. Wang and B. Li. *Lava*: A Reality Check of Network Coding in Peer-to-Peer Live Streaming. In *Proc. of the 26th IEEE INFOCOM*, 2007.
50. M. Wang and B. Li. R^2: Random Push with Random Network Coding in Live Peer-to-Peer Streaming. *IEEE J. on Sel. Areas in Communications*, 2007.
51. C. Wu, B. Li, and S. Zhao. Multi-channel Live P2P Streaming: Refocusing on Servers. In *Proc. of the 28th IEEE INFOCOM*, 2009.
52. D. Wu, Y. Liu, and K. Ross. Queueing network models for multi-channel p2p live streaming systems. In *Proc. of the 28th IEEE INFOCOM*, 2009.
53. X. Xiao, Y. Shi, Y. Gao, and Q. Zhang. LayerP2P: A New Data Scheduling Approach for Layered Streaming in Heterogeneous Networks. In *Proc. of the 28th IEEE INFOCOM*, 2009.
54. H. Yin, X. Liu, T. Zhan, V. Sekar, F. Qiu, C. Lin, H. Zhang, and B. Li. Design and Deployment of a Hybrid CDN-P2P System for Live Video Streaming: Experiences with LiveSky. In *Proc. of the 17th ACM Multimedia*, 2009.
55. M. Zhang, Q. Zhang, L. Sun, and S. Yang. Understanding the Power of Pull-Based Streaming Protocols: Can We Do Better? *IEEE J. on Sel. Areas in Communications*, 2007.
56. X. Zhang, J. Liu, B. Li, and T.-S. P. Yum. CoolStreaming/DONet: A Data-Driven Overlay Network for Efficient Live Media Streaming. In *Proc. of the 24th IEEE INFOCOM*, 2005.
57. M. Zhou and J. Liu. A Hybrid Overlay Network for Video-on-Demand. In *Proc. of IEEE ICC*, 2005.