Resource Allocation in Underprovisioned Multioverlay Live Video Sharing Services

Jiayi Liu, Shakeel Ahmad, Eliya Buyukkaya, Raouf Hamzaoui, Gwendal Simon

Resource Allocation in Underprovisioned Multioverlay Live Video Sharing Services

Jiayi Liu
Telecom Bretagne, France
jiayi.liu@telecom-bretagne.eu

Shakeel Ahmad
De Montfort University, UK
sahmad@dmu.ac.uk

Eliya Buyukkaya
Telecom Bretagne, France
eliya.buyukka@telecom-bretagne.eu

Raouf Hamzaoui
De Montfort University, UK
rhamzaoui@dmu.ac.uk

Gwendal Simon
Telecom Bretagne, France
gwendal.simon@telecom-bretagne.eu

ABSTRACT

In a multioverlay live video sharing service consisting of multiple independent peer-to-peer live video streaming systems, a user can simultaneously watch multiple live video streams. A major challenge for such services is the inter-overlay bandwidth competition problem, which is to find an upload bandwidth allocation between the overlays each peer has subscribed to. So far, no solution has been proposed in the literature for the important case where the overall system is underprovisioned, that is, when peers do not have enough upload bandwidth to ensure a distribution of videos at full quality. We show that an allocation of upload resources that minimizes the wastage of resources (i.e., minimizes the upload bandwidth allocated to overprovisioned overlays) can be computed in polynomial time. Then we present a generic model that allows the design of different strategies for the management of the resource deficit in underprovisioned systems. Finally, we provide relevant simulation results to demonstrate the gains in video quality resulting from the implementation of our solutions.

Categories and Subject Descriptors

C.2.4 [**Computer-Communication Networks**]: Distributed Systems; H.4.3 [**Information Systems Applications**]: Communications Applications

General Terms

Algorithms, Design, Performance

Keywords

Peer-to-Peer Live Streaming, Multioverlay, Resource Allocation

CSWS'12, December 10, 2012, Nice, France.

1. INTRODUCTION

The popularity of academic papers related to peer-to-peer (P2P) systems has followed a "bubble" evolution over the last decade. In their well-documented analysis [4], Li, Feng and Li show that after a surge of popularity from 2000 to 2008, P2P has become a "cold topic". On the one hand, this sharp decrease in scientific interest makes sense because P2P systems are now mature, well-understood, and can be fearlessly used by industry. Moreover, compared to cloud offers, the trade-off between operational costs and guarantee of performance is now less favorable to P2P systems. On the other hand, companies that implement P2P systems face new issues, which are not directly related to the P2P systems, but rather to their integration in a global system.

In this paper, we address a capacity management problem for P2P systems. This problem was brought to our attention by an independent producer of Massively Multiplayer Online Game (MMOG). MMOGs enable users to play against other players or to build groups to achieve missions. Within a group, synchronisation among teammates requires communication tools. Unfortunately, existing games offer only a few basic communications tools (if any). Surveys [1] have highlighted that streaming live screen-captured video of the game is one of the most desirable communication tools. Players can use it to show their skills, share experience with friends, or coordinate missions in strategy games.

Existing online video platforms for gamers [9, 14] rely on a centralized architecture. Even when the system is coupled with a Content Delivery Network (CDN), this solution is not cost-effective. Indeed, the popularity distribution of such service poses a major challenge: (i) a large proportion of players are likely to act as video sources, so there are many live streams to deal with, and (ii) each stream is typically watched by a small population consisting of a few friends and teammates.

MMOG producers foresee a *multioverlay* P2P system consisting of multiple P2P live video streaming systems [10]. Some users are sources, which emit a user-generated live video, while the others are peers, which receive one or several videos and participate in diffusing these videos to other peers. Each P2P network contains one source and all peers that have *subscribed* to its live stream (channel).

In the context of MMOGs, players in a team share their videos for coordination purposes, so a peer can watch several

videos simultaneously. The fundamental problem faced by a peer that subscribes to several P2P networks is how to share its uplink bandwidth among these concurrent systems. Only a few papers [10–12] have addressed this resource allocation problem, and they focused on the case where the overall system is overprovisioned. Our tests based on realistic settings will reveal that, on the contrary, *multioverlay systems are underprovisioned*, i.e., the upload bandwidth allocated to this system is smaller than the streaming demand.

Some previous works have suggested to assist the P2P overlays by resources from servers in datacenters [7,13]. This solution however does not accommodate well a scenario with many small-size overlays, as it would require reserving and managing a large number of Virtual Machines, each generating a small amount of traffic.

The current paper makes the following original contributions to the problem of resource allocation for underprovisioned multioverlay systems.

- We show how to minimize the waste of resources resulting from overprovisioning some overlays although some others are underprovisioned. We show that an optimal solution can be found in a time that is polynomial in the number of users.
- We show how to share the bandwidth deficit among channels so that a pre-determined policy is satisfied. We consider two policies: minimizing the number of underprovisioned channels, and prioritizing the most popular channels. We tackle this new problem in a generic way. We present a polynomial-time algorithm that finds allocations that are optimal in terms of resource waste and correspond to the best allocations with respect to the policy defined by the service provider.

Note that our resource allocation strategies are compatible with any P2P video streaming protocol. Some previous works have proposed to solve the resource allocation competition and the P2P video streaming delivery all at once [12]. These approaches are not agnostic to the P2P video streaming system. We believe contrarily that a clear separation between *inter-overlay* resource management and *intra-overlay* video diffusion is crucial. Indeed, the multioverlay system should leverage the advances in P2P streaming systems.

We compare our strategies to the algorithms presented in [10, 11]. With simulation parameters based on previous observations, we show that the overall system is underprovisioned, which confirms our intuition that underprovisioning is a critical issue for multioverlay systems. We implement a mesh-based intra-overlay P2P simulator and compute the Peak Signal to Noise Ratio (PSNR) of the received stream for every peer. To our knowledge, this is the first time that the quality of experience of users in underprovisioned overlays is studied.

2. SYSTEM MODEL

We first give the notations used throughout the paper (see Table 1). Then, we present the bipartite graph that is the basis for our proposals. Finally, we show that the waste of resources can be minimized by determining the maximum flow in this bipartite graph.

The system includes multiple P2P overlays, one for each video stream, and a global server. The role of the server is to authorize a peer to watch a video emitted by a source and to compute the optimal bandwidth allocation. Peers transmit an estimation of their available upload capacity to the server on a regular basis. Then, based on these reports, the server computes and sends the optimal bandwidth allocation to each peer.

P, S	set of peers, set of sources
G_s, P_s	overlay of source s and set of peers in G_s
B_p	upload capacity of a peer p
b_p^s	upload capacity reserved by p for G_s
$G(p)$	set of sources to which peer p subscribed
d_s	video bit-rate of the video emitted by s
D_s, C_s	demand and capacity of G_s
Δ_s, Δ_s^r	provisioning and relative provisioning of G_s

Table 1: Notations

2.1 Notations

Sources. The set of sources is denoted by S. A source $s \in S$ is associated with an overlay G_s, which contains the set P_s of all peers that have subscribed to the video emitted by s. To avoid confusion, $s \notin P_s$.

Peer-to-Peer Live Streaming System. Our resource allocation strategies are independent of the intra-overlay structure. They can be used with any state-of-the-art P2P live video streaming system.

Peer Uplink Management. The set of all peers is denoted by P. We denote by $G(p)$ the set of sources from which the peer p receives a video. Every peer uses its uplink to transfer the chunks it received to other peers in the same overlay. The upload capacity of p is denoted by B_p while the upload capacity that p reserves to serve video chunks in the overlay G_s is denoted by b_p^s. Clearly, $\sum_{s \in G(p)} b_p^s \leq B_p$. We assume that s reserves all of its upload bandwidth to its overlay G_s.

Overlay Capacity. In an ideal system without control messages or network overhead, the capacity of an overlay G_s, which is denoted by C_s, would be equal to $\sum_{p \in P_s} b_p^s + B_s$, which is the aggregate upload bandwidth allocated from peers to s plus the capacity of s. In a real system, the control traffic cannot be neglected, so the overlay capacity should be reduced by a value proportional to the number of peers.

Overlay Demand. The demand of a source s corresponds to the smallest overlay capacity required to satisfy all peers in P_s. The bit-rate of the video emitted by s is denoted by d_s. Since overhead is already incorporated in the overlay capacity, we assume that the demand of s, denoted by D_s, is computed as $|P_s| \times d_s$.

Overlay Provisioning. The provisioning Δ_s of a given overlay G_s is the difference between its capacity C_s and its demand D_s, i.e., $\Delta_s = C_s - D_s$. An overlay is said to be underprovisioned when Δ_s is negative. The average upload capacity is smaller than the video bit-rate, so some peers in this overlay are unable to watch the video at full quality. On the other hand, the overlay is overprovisioned when Δ_s is positive.

Overlay Relative Provisioning. The relative provisioning Δ_s^r of a given overlay G_s is defined as the overlay provisioning divided by the number of peers in the overlay, that is, $\Delta_s^r = \frac{\Delta_s}{|P_s|}$. The more negative is the relative provisioning, the worse is the video quality experienced by the peers.

2.2 Bipartite Flow Network Model

We aim at minimizing the wastage of resources caused by allocating resources to overprovisioned overlays although they could be allocated to underprovisioned ones. We show that this problem can be solved by determining the maximum flow in a flow network.

We build an *abstract* structure, which is a flow network. A link in this flow network is an abstract representation of the existence of a relationship between two system elements. Our flow network $N = (V, E)$ is built according to source-peer relationships (Fig. 1). The set V contains a virtual fountain l, a virtual sink q, the set P of all peers in the system, and the set S of all sources. Thus $V = P \cup S \cup \{l, q\}$.

The set of edges E represents resource allocation. The capacity indicates a limitation in the amount of bandwidth resources that a node can reserve to another. The set E contains three subsets. First, $E_1 = \{(l \to p) : p \in P\}$, contains edges from the fountain to each peer p with a maximum capacity of B_p. Then, $E_2 = \{(p \to s) : p \in P, s \in G(p)\}$, contains edges from p to s if p subscribes to s with infinite maximum capacity. The third set, $E_3 = \{(s \to q) : s \in S\}$, contains edges from each source s to the sink with a maximum capacity equal to $D_s - B_s$, the overlay demand minus the source capacity.

Given demands and capacities, the resource allocation should ensure that no resource is allocated to an overprovisioned source when it could have been allocated to an underprovisioned one. Let S^+ (respectively S^-) be the set of sources with a positive (respectively negative) overlay provisioning. We look for an uplink sharing among the overlays such that the sum of all negative provisionings is minimum. Hence, our goal is to minimize $\sum_{s \in S^-} |\Delta_s|$.

PROPOSITION 1. *The total underprovisioning is minimum iff the maximum flow is achieved in network N.*

PROOF. We denote by $f_{s,q}$ the flow on the arc $(s \to q)$. The cut-set between $V \setminus \{q\}$ and $\{q\}$ bounds a flow: $|f| = \sum_{s \in S} f_{s,q}$.

For each source s, the absolute underprovisioning $|\Delta_s|$ is equal to $D_s - B_s - f_{s,q}$. Note that, for a source s in S^+, we have that $D_s - B_s - f_{s,q}$ is equal to zero because the flow $f_{s,q}$ cannot be greater than $D_s - B_s$. Thus,

$$\sum_{s \in S^-} |\Delta_s| = \sum_{s \in S^-} (D_s - B_s - f_{s,q})$$

$$= \sum_{s \in S} (D_s - B_s - f_{s,q}) = A - \sum_{s \in S} f_{s,q}$$

where A is a constant. Minimizing $\sum_{s \in S^-} |\Delta_s|$ is equivalent to maximizing $\sum_{s \in S} f_{s,q}$. Moreover, (i) edges from l to P fuel the system with all capacities $C = \sum_p B_p$, (ii) edges from P to S respect peer subscription. Hence, the overall underprovisioning $\sum_{s \in S^-} |\Delta_s|$ is minimized iff the flow is maximized. □

3. POLICY DRIVEN BANDWIDTH ALLOCATION STRATEGIES

There is often more than one resource allocation that minimize the total underprovisioning because the maximum flow is not unique. The service provider has here an opportunity to design its own *policy* for preserving some sources to be affected by the overall underprovisioning. In this section, we present a *generic* way to implement such a policy.

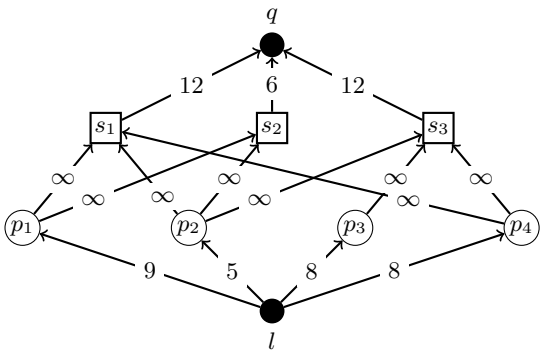


Figure 1: Example of the bipartite flow network model. Numbers in the arrows are capacities, *i.e.* limitations in the amount of bandwidth resources that a node can reserve to another.

3.1 Cost-Function Flow Network

In our flow network model, we propose to define a cost function for edges in E_3. As the minimum-cost maximum-flow problem aims at minimizing the sum of flow cost, the edges associated with a lower cost will be prioritized in the bandwidth allocation. Consequently, different bandwidth allocation strategies can be applied by using correspondingly defined cost functions. This approach is generic in the sense that various cost functions can be designed, which result in various resource allocations. We present later two distinct bandwidth allocation strategies.

Strategy I: Prioritize Channel Diversity The goal is to satisfy the maximum number of sources regardless of their popularity. A source is said to be satisfied when its relative provisioning is positive, or slightly negative. Thus, we define our first strategy as one that minimizes the total relative under-provisioning: $\sum_{s \in S^-} |\Delta_s^r|$. Let us define the cost function as:

$$cost_1(e) = \begin{cases} 1, \text{if } e \in E_1 \cup E_2 \\ 1 - \frac{1}{|P_s|}, \text{if } e \in E_3 \end{cases}$$

We now show that this policy can be obtained by the minimum-cost maximum flow corresponding to $cost_1(e)$.

PROPOSITION 2. *The minimum-cost maximum flow corresponding to $cost_1(e)$ minimizes the total relative underprovisioning $\sum_{s \in S^-} |\Delta_s^r|$.*

PROOF.

$$\sum_e cost(e) \times f_e = \sum_{e \in E_1} f_e + \sum_{e \in E_2} f_e + \sum_{e \in E_3} (1 - \frac{1}{|P_s|}) \times f_e$$

E_1 and E_2 are cut sets separating l and q, so $\sum_{e \in E_1} f_e = \sum_{e \in E_2} f_e = |f_{max}|$. Thus

$$\sum_e cost(e) \times f_e = 2|f_{max}| + \sum_{e \in E_3} f_e - \sum_{s \in S} \frac{D_s - B_s - |\Delta_s|}{|P_s|}$$

$$= 3|f_{max}| - \sum_{s \in S} \frac{D_s - B_s}{|P_s|} + \sum_{s \in S^-} |\Delta_s^r|$$

where the first two terms are constants. Thus, minimizing $\sum_e cost(e) \times f_e$ is equivalent to minimizing $\sum_{s \in S^-} |\Delta_s^r|$. □

Strategy II: Prioritize Channel Popularity The goal is to prioritize the most popular channels. This can be achieved by maximizing the number of *unsatisfied* sources, for example, maximizing the total relative underprovisioning $\sum_{s \in S^-} |\Delta_s^r|$.

$$cost_2(e) = \begin{cases} 1, \text{if } e \in E_1 \cup E_2 \\ \frac{1}{|P_s|}, \text{if } e \in E_3 \end{cases}$$

PROPOSITION 3. *The minimum-cost maximum flow corresponding to $cost_2(e)$ maximizes $\sum_{s \in S^-} |\Delta_s^r|$.*

PROOF. The proof is similar to that of Proposition 2. □

3.2 Practical Optimization

Previous strategies have a common drawback: they aim at ensuring a zero provisioning to the prioritized sources, although a slightly negative relative provisioning would have a small impact on the overall quality of experience. To address this problem, we introduce a tunable *tolerable video quality parameter* k and say that an overlay has tolerable video quality if its relative provisioning $|\Delta_s^r|$ is smaller than k. The demand D_s of a source s can be interpreted as the amount of upload bandwidth required by s to be provisioned as $\Delta_s = 0$. If the system is globally very underprovisioned, rather than requiring perfect video quality on each source, we only require tolerable video quality. This can be done by tuning the parameter k. Consequently, the actual D_s is equal to $(d_s - k) \times |P_s|$.

4. IMPLEMENTATION DETAILS

This section provides practical implementation details of the multioverlay system. The system is managed by a global server called *P2PServer*. Peers send reports to P2PServer on a regular basis, e.g., every minute. This report contains an estimation of the peer's available bandwidth. Based on these reports, P2PServer computes and sends the optimal bandwidth allocation to each peer.

4.1 Coping with Peer Dynamics

The system described above is easy to implement in a static environment; however peer-to-peer live video streaming applications face the problem of peer churn. Peers are ordinary users who are free to enter and leave the system, and also to switch from one channel to another. Moreover, the available bandwidth can vary between two measurements. Even if the computation of bandwidth allocation can be done in polynomial time, it is unrealistic to repeat it after every event.

To cope with peer churn, the strategy proposed in [10, 11] can be used. The system time can be cut into *sessions*. The computation of upload bandwidth allocation in a session only involves peers and sources that exist in the system at the starting point of that session. Every event that occurs during a session will have an effect on the system until the end of the session and the start of the next session. So, the capacity of the system to handle the dynamic behavior of peers only depends on the choice of the length of a session. The shorter is a session, the better the system reacts to changes, but the more computation it requires.

nb. peers	1,000	5,000	10,000	50,000	100,000
time (sec)	0.005	0.086	0.311	7.455	31.887

Table 2: Computation time for minimum-cost maximum-flow based algorithms.

4.2 Peer-Server Communication Overhead

We are concerned about the extra-traffic generated by the system when peers send reports to P2PServer, and when P2PServer sends bandwidth allocation information to peers. However, this extra-traffic has to be seen in light of the huge amount of data needed for the live video streams. For example, if the average number of videos watched by a peer is three, two bytes are used to specify the upload bandwidth reserved to an overlay, and the bandwidth allocation is recomputed every minute, then P2PServer needs to transmit 0.8 bps per peer. If we consider in addition the 54 bytes for the TCP/IP/Ethernet packet overhead, then 0.8 Mbps server upload bandwidth would be needed for 100,000 users. Similarly, if the report sent by a peer to P2PServer includes four bytes to specify the estimated upload bandwidth and four bytes for the peer ID, then only 0.826 Mbps server download bandwidth would be needed for 100,000 users. This example shows that, from a network standpoint, the system can be implemented without much fear for scalability.

4.3 Algorithm Computation Time

Another concern is the scalability in terms of computation time. We measured the exact computation time for the centralized minimum-cost maximum-flow based algorithms. We used the *preflow* maximum flow algorithm and a scaling approximation minimum-cost flow algorithm [3]. The measurement was done on a typical server (2×4 cores Intel(R) Xeon(R) 2.67GHz CPUs). We computed the average running time for 5 different runs. Results are given in Table 2. We changed the instance size by increasing the number of peers from 1,000 to 100,000. For each instance, the number of sources was set to 10% of the population. The number of channels a peer watches was randomly chosen between 1 and 5.

Our measurement demonstrates that a practical implementation of the minimum-cost maximum-flow algorithm can compute the resource allocation for very large instances (with 100,000 peers and 10,000 sources) in reasonable time. It also demonstrates that refreshing the bandwidth allocation every minute is an appropriate choice. All in all, the small peer to server communication overhead and the fast resource allocation algorithm demonstrate the feasibility of managing a centralized server to recompute periodically an optimal resource allocation in a dynamic environment.

5. SIMULATION RESULTS

We evaluated the proposed inter-overlay bandwidth allocation algorithms through simulations based on a realistic model. The chunk diffusion of each overlay was simulated with *P2PTVSim* [5]. Using these results, we evaluated the video quality at each receiving peer by measuring the average luminance PSNR.

5.1 Simulation Setting

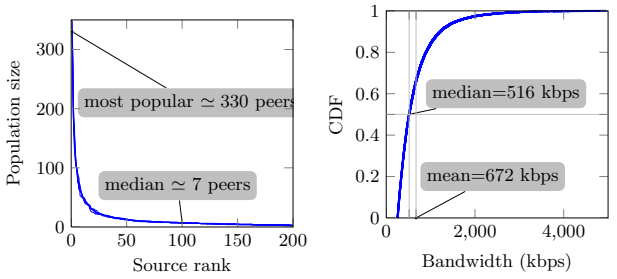
Our simulation is based on an MMOG video sharing ap-

plication: players publish their own videos and share them with their friends. The Xfire measurements presented in [6] show that these platforms are social networks. We thus look for real traces of social networks that have the same characteristics as the one exhibited in [6]. We found that the Facebook social network of the Smith College (MA) [8] is very close to Xfire, especially the average social degree (around 65 connections per user). This social network contains 2,970 users.

We then selected the sources. We ensured that the Pareto rule is obeyed, that is, 80% of videos are published by 20% of the most active peers. It means that the most socially active peers have a higher probability to be sources. A peer decides to become a source based on a probability equal to $e^{-\frac{i}{\tau N}}$, where i denotes its rank in terms of social degree in decreasing order, N denotes the total number of peers, and τ is a parameter. We chose $\tau = 0.1$; on average, 214 channels are created.

We then selected the viewers. We model the peer watching decision such that the resulting video popularity follows the same Zipf's law as in [6]. Sources are ranked by their social degree in decreasing order. Let s_i be the ith source with n_{s_i} denoting its number of friends. We associate each source s_i with an *attractiveness index* α_i which is equal to the probability that its friends watch this video. The index α_i is calculated such that the channel population follows Zipf's law: $\frac{n_{s_i}}{n_{s_1}} \cdot \alpha_i \sim i^{-b}$. Fig. 2(a) shows the resulting channel popularity.

For the upload capacity of peers, we followed some recent works that model home upload capacity with a *log-normal* distribution ranging from 256 kbps to 5 Mbps [2]. Fig. 2(b) shows the cumulative distribution function (cdf) of the upload bandwidth distribution. Each video diffusion in an overlay was mesh-based pull-based, and evaluated with 400 chunks. The video bit rate varied from 240 kbps (when the system was overprovisioned) to 330 kbps (when it was underprovisioned). The P2P system overhead (control messages and redundant transmission) was set to 50 kbps.



(a) Channel popularity (b) Upload Bandwidth

Figure 2: Simulation settings

We used a 600-frame video by concatenating 300 frames of the Foreman sequence and 300 frames of the Mother and Daughter sequence. Both sequences are in CIF format and have a frame rate of 30 fps. The video was compressed with the H.264/AVC encoder at bit rates ranging from 240 to 330 kbps using the H.264 high profile. We used the GOP structure IBBPBBPBBP (10 frames per GOP). Each chunk corresponds to one GOP, so each GOP is played back independently of the other chunks. At the receiver side, we

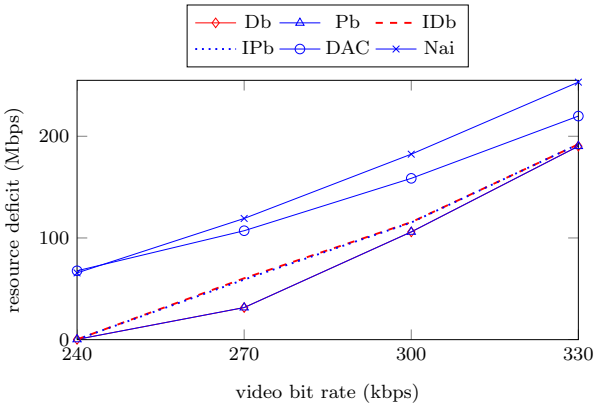


Figure 3: Amount of missing upload capacity.

used the standard frame copy error concealment technique to deal with lost frames. With this technique, the last frame of the last decoded GOP is used to represent all frames of a missing chunk.

We compared the following six algorithms:

DAC: This algorithm was proposed in [10, 11]. It fairly allocates upload bandwidth based on overlay demands, but it is blind to the provisioning of overlays.

Diversity-based **(Db)**: corresponds to the strategy that enhance channel diversity.

Popularity-based **(Pb)**: corresponds to the strategy that serves the most popular channels first.

Improved diversity-based **(IDb)** and *Improved popularity-based* **(IPb)**: include the tolerable video quality parameters to the policies of **Db** and **Pb**, respectively. The system is overprovisioned with video bit rate 240 kbps, then k was set to 0 kbps. For the other video bit rates, k was set to 50 kbps. *Naive* **(Nai)**: This algorithm equally divides the upload bandwidth among the overlays.

5.2 Provisioning Results

Fig. 3 shows the total underprovisioning $\sum_{s \in S^-} |\Delta_s|$ of all algorithms for all video bit-rates. The two maximum-flow based approaches, (**Db** and **Pb**), minimize the total underprovisioning, producing identical curves. For video bit rate 240 kbps, the system is overprovisioned, and **Db**, **Pb**, **IDb** and **IPb** have zero underprovisioning, which means that every overlay in the system is well provisioned. We observe that both **IDb** and **IPb** (which also have identical curves) can have a slightly larger underprovisioning than their non-improved counterparts. Indeed, the diminution of the overlay demand can prevent the computation of a maximum flow in the flow network. Finding the best tolerable video quality parameter according to the system state and the video bit-rate is left as a future work. The **DAC** allocation according to the demand favors the most demanding overlays, with the cost that less demanding overlays are underprovisioned. It performs almost as badly as the **Nai** algorithm when the system is slightly underprovisioned, and then improves when all demands grow.

5.3 PSNR Results

We provide objective video quality results by measuring the average luminance PSNR at each receiving peer. To illustrate the distribution of the PSNR, we show the com-

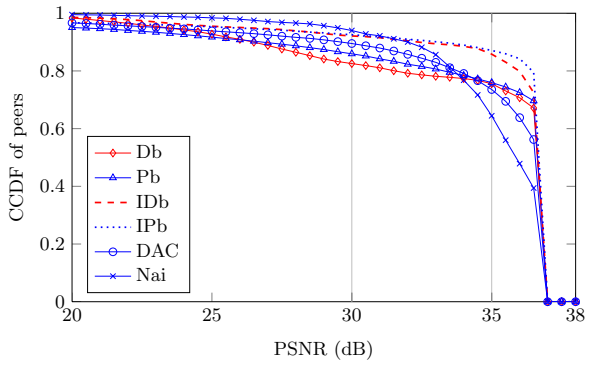


Figure 4: CCDF of PSNR for video bit rate 330 kbps.

plementary cumulative distribution function (ccdf) of the PSNR for video bit rate 330 kbps (Fig. 4). We distinguish three sets of allocations: the maximum-flow allocations (**Db**, and **Pb**), their improved counterparts (**IDb** and **IPb**), and **DAC** and **Nai**. For the first two groups, lines almost overlap between allocations. For the maximum achievable PSNR value (37 dB), we observe that a significant proportion of peers experienced high video quality: 70% (respectively 90%) of peers for the maximum-flow allocations (respectively the improved ones). For **DAC** and **Nai**, this percentage is 40% and 60%, respectively. With both **IDb** and **IPb**, only 15% of peers get video with PSNR below 35 dB, which demonstrates the potential of these policies. Note also that the maximum-flow allocations (**Db**, and **Pb**) have extreme bandwidth allocation strategies that result in around 20% of peers with PSNR below 30 dB.

6. CONCLUSION

Some (supposedly scalable and robust) technical solutions can meet unexpected capacity issues when implemented in a real system. In this paper, we consider the challenge of peers contributing to several overlays. This problem is almost ignored in the P2P streaming literature and can prevent an efficient deployment of P2P technologies.

First, *global underprovisioning can be minimized by a smart resource allocation algorithm.* Flow-based algorithms, such as the one presented in this paper, are traditionally used for finding efficient matching; they are appropriate for resource allocation problems as well. Fortunately, there exist very efficient implementations of flow algorithms. From our experience, a server can easily manage up to thousands of simultaneous peers. Moreover, distributed flow algorithms can be implemented in clusters of servers. In summary, the implementation of optimal resource allocation is possible, and it significantly reduces the impact of underprovisioning compared to a naive approach.

Second, *sharing the deficit in a smart way is not restricted to fair policies.* In this study, the resources are under the exclusive control of the service provider, and it is probable that business-oriented motivations prevail over fairness. We made a first step in the development of such pre-determined allocation policies with our original minimum-cost maximum flow algorithm. Note that this algorithm can apply to other policies as well. For example, it is straightforward to define a cost function to serve in priority a premium class of fee-paying sources.

Third, *tolerating a slight underprovisioning for everybody is a remarkably efficient way to reduce the impact of underprovisioning.* Multimedia services include multiple technologies to cope with packet loss, including data redundancy and channel coding techniques. As a matter of fact, a small amount of packet loss has imperceptible (or negligible) impact on the Quality of Experience. Our study shows that it is possible to leverage this characteristics to reduce the impact of underprovisioning in general. We take advantage of internal loss correction techniques. By purposely introducing a small underprovisioning, the overall system has far better performance. Our future works will include the development of algorithms to set this tolerable underprovisioning according to the context.

7. ACKNOWLEDGMENTS

The research leading to these results has received funding from the European Commission's Seventh Framework Programme (FP7, 2007-2013) under the grant agreement no. ICT-248175 (CNG project).

8. REFERENCES

[1] CNG. http://www.cng-project.eu/.
[2] M. Dischinger, A. Haeberlen, P. K. Gummadi, and S. Saroiu. Characterizing residential broadband networks. In *ACM IMC*, 2007.
[3] A. V. Goldberg. An efficient implementation of a scaling minimum-cost flow algorithm. *Journal of Algorithms*, 22:1–29, 1997.
[4] B. Li, Y. Feng, and B. Li. Rise and fall of the peer-to-peer empire. *Tsinghua Science and Technology*, 17(1):1–16, Feb. 2012.
[5] P2PTVSim. http://napa-wine.eu/cgi-bin/twiki/view/Public/P2PTVSim.
[6] S. Shen and A. Iosup. The xfire online meta-gaming network: Observation and high-level analysis. In *Workshop MMVE*, 2011.
[7] R. Sweha, V. Ishakian, and A. Bestavros. Angelcast: cloud-based peer-assisted live streaming using optimized multi-tree construction. In *ACM MMSys*, 2012.
[8] A. L. Traud, P. J. Mucha, and M. A. Porter. Social structure of facebook networks. *CoRR*, abs/1102.2166, 2011.
[9] TwitchTV. http://www.twitch.tv/.
[10] M. Wang, L. Xu, and B. Ramamurthy. A flexible divide-and-conquer protocol for multi-view peer-to-peer live streaming. In *IEEE P2P*, 2009.
[11] M. Wang, L. Xu, and B. Ramamurthy. Improving multi-view peer-to-peer live streaming systems with the divide-and-conquer strategy. *Computer Networks*, 55(18):4069–4085, 2011.
[12] C. Wu, B. Li, and Z. Li. Dynamic bandwidth auctions in multioverlay p2p streaming with network coding. *IEEE Trans. Parallel Distrib. Syst.*, 19(6):806–820, 2008.
[13] C. Wu, B. Li, and S. Zhao. On dynamic server provisioning in multichannel p2p live streaming. *IEEE/ACM Trans. Netw.*, 19(5):1317–1330, 2011.
[14] Xfire. http://www.xfire.com/.