Multi-channel Live P2P Streaming: Refocusing on Servers

Chuan Wu, Baochun Li
Department of Electrical and Computer Engineering
University of Toronto

Shuqiao Zhao
Multimedia Development Group
UUSee, Inc.

Abstract—**Due to peer instability and time-varying peer upload bandwidth availability in live peer-to-peer (P2P) streaming channels, it is preferable to provision adequate levels of stable upload capacities at dedicated streaming servers, in order to guarantee the streaming quality in all channels. Most commercial P2P streaming systems have resorted to the practice of over-provisioning upload capacities on streaming servers. In this paper, we have performed a detailed analysis on 400 GB and 7 months of run-time traces from UUSee, a commercial P2P streaming system, and observed that available capacities on streaming servers are not able to keep up with the increasing demand imposed by hundreds of channels. We propose a novel online server capacity provisioning algorithm that proactively adjusts the server capacities available to each of the concurrent channels, such that the supply of server bandwidth in each channel dynamically adapts to the forecasted demand, taking into account the number of peers, the streaming quality, and the priorities of channels. The algorithm is able to learn over time, and has full ISP awareness to maximally constrain P2P traffic within ISP boundaries. To evaluate the effectiveness of our solution, our experimental studies are based on an implementation of the algorithm with actual channels of P2P streaming traffic, with real-world traces replayed within a server cluster.**

I. Introduction

With the recent success and commercial deployment of live P2P streaming [1], hundreds of media channels are routinely broadcast to millions of users at any given time. The essence of P2P streaming is the use of peer upload bandwidth to alleviate the load on dedicated streaming servers [2]. Most existing research has thus far focused on peer strategies: Should a mesh or tree topology be constructed? What incentives can be provisioned to encourage peer bandwidth contribution? How do we cope with peer churn and maintain the quality of live streams? We recognize the importance of these open research challenges, as their solutions seek to maximally utilize peer upload bandwidth, leading to minimized server costs.

With this paper, however, we shift our focus to the streaming servers. Such refocusing on servers is motivated by our detailed analysis of 7 months and 400 GB worth of real-world traces from hundreds of streaming channels in UUSee [3], a large-scale commercial P2P live streaming system in China. As all other state-of-the-art live streaming systems (including PPLive), in order to maintain a satisfactory and sustained streaming quality, UUSee has so far resorted to the practice of over-provisioning server capacities to satisfy the streaming demand from peers in each of its channels. Contrary to common belief, we have observed that available capacities on streaming servers are not able to keep up with the increasing demand from hundreds of channels. In response, we advocate to allocate limited server capacities to each of the channels, in order to maximally utilize dedicated servers.

While it is certainly a challenge to determine how much bandwidth to provision on streaming servers to accommodate the streaming demand of all concurrent channels, the challenge is more daunting when we further consider the conflict of interest between P2P solution providers and ISPs. P2P applications have significantly increased the volume of inter-ISP traffic, which in some cases leads to ISP filtering. We seek to design effective provisioning algorithms on servers with the awareness of ISP boundaries to minimize inter-ISP traffic.

In this paper, we present *Ration*, an online server capacity provisioning algorithm to be carried out on a per-ISP basis. *Ration* dynamically computes the minimal amount of server capacity to be provisioned to each channel inside the ISP, in order to guarantee a desired level of streaming quality for each channel. With the analysis of our real-world traces, we have observed that the number of peers and their contributed bandwidth in each channel are dynamically varying over time, and significantly affect the required bandwidth from servers. *Ration* is designed to actively *predict* the bandwidth demand in each channel in an ISP with time series forecasting and dynamic regression techniques, utilizing the number of active peers, the streaming quality, and the server bandwidth usage within a limited window of recent history. It then proactively allocates server bandwidth to each channel, respecting the predicted demand and priority of channels. To show the effectiveness of *Ration*, it has been implemented in streaming servers serving a mesh-based P2P streaming system. In a cluster of dual-CPU servers, the system emulates real-world P2P streaming by replaying the scenarios captured by traces.

The remainder of this paper is organized as follows. In Sec. II, we motivate our focus on servers by showing our analysis of 7 months of traces from UUSee. In Sec. III, we present the design of *Ration*, and discuss how it may be deployed with ISP awareness to serve real-world P2P streaming systems. Sec. IV presents our experimental results evaluating *Ration* by replaying traces in a P2P streaming system running in a server cluster. We discuss related work and conclude the paper in Sec. V and Sec. VI, respectively.

II. Refocusing on Servers: Evidence from Real-world Traces

Why shall we refocus our attention to dedicated streaming servers in P2P live streaming systems? Starting Sep. 2006, we have continuously monitored the performance statistics of a real-world commercial P2P streaming platform, offered by UUSee Inc., a leading P2P streaming solution provider with legal contractual rights with mainstream content providers in China. As other such systems such as PPLive, UUSee maintains a sizable array of 150 dedicated streaming servers, to support its P2P streaming topologies with hundreds of channels to millions of users, mostly in 400 Kbps media streams. UUSee utilizes the "pull-based" design on mesh P2P topologies, that allows peers to serve other peers ("partners") by exchanging media blocks in a sliding window of the stream.

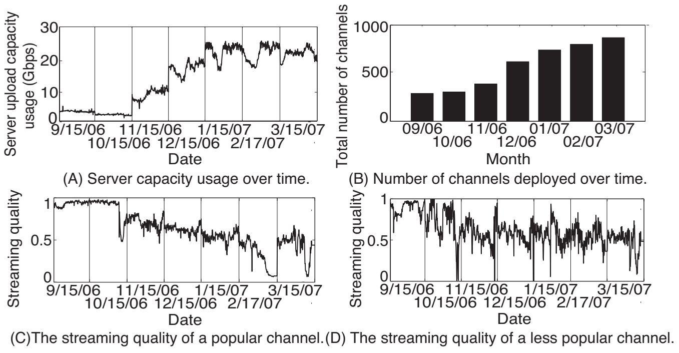
To maximally utilize peer upload bandwidth and alleviate server load, UUSee incorporates a number of algorithms in peer selection. Each peer applies an algorithm to estimate its maximum upload capacity, and continuously estimates its aggregate instantaneous sending throughput to its partners. If its estimated sending throughput is lower than its upload capacity for 30 seconds, it will inform one of the tracking servers that it is able to receive new connections. The tracking servers keep a list of such peers, and assign them upon requests of partners from other peers. During the streaming process, neighboring peers also recommend known peers to each other based on their current streaming quality, represented by the number of available blocks in their current playback buffers. A peer may contact a tracking server again to obtain additional peers with better qualities, once it has experienced low buffering levels for a sustained period of time.

To inspect the run-time behavior of UUSee P2P streaming, we have implemented extensive measurement and reporting capabilities within its P2P client application. Each peer collects a set of its vital statistics, and reports to dedicated trace servers every 5 minutes via UDP. The statistics include its IP address, the channel it is watching, its buffer availability map, the number of available blocks in its current playback buffer (henceforth referred to as the *buffer count*), as well as a list of all its partners, with their corresponding IP addresses, TCP/UDP ports, and current sending/receiving throughput to/from each partner. Each dedicated streaming server in UUSee utilizes a similar P2P protocol as deployed on regular peers, is routinely selected to serve the peers, and reports its related statistics periodically as well.

A. *Insufficient "supply" of server bandwidth*

What have we discovered from the traces, that represent snapshots of the system every 5 minutes throughout the 7 months? The first observation we made is related to the insufficient "supply" of server bandwidth, as more channels are added over time. Such insufficiency has gradually affected the streaming quality, in both popular and less popular channels.

In order to show bandwidth usage over 7 months and at different times of a day within one figure, we choose to show all our 5-minute measurements on representative dates in each month. One such date, February 17 2007, is intentionally chosen to coincide with the Chinese New Year event, with typical flash crowds due to the broadcast of a celebration show on a number of the channels. Fig. 1(A) shows the total server bandwidth usage on 150 streaming servers. We may observe that an increasing amount of server bandwidth has been consumed over time, but stabilizing in January 2007. This rising trend can be explained by the rapidly increasing number of channels deployed during this period, as shown in Fig. 1(B). The interesting phenomenon that such bandwidth usage has stabilized, even during the Chinese New Year flash crowd, has led to the conjecture that the total uplink capacity of all servers has been reached. The daily variation of server bandwidth usage coincides with the daily pattern of peer population.



(A) Server capacity usage over time. (B) Number of channels deployed over time. (C)The streaming quality of a popular channel. (D) The streaming quality of a less popular channel.

Fig. 1. The evolution of server bandwidth, channels, and streaming quality over a period of 7 months.

Our conjecture that server capacities have saturated is confirmed when we investigate the streaming quality in each channel. The streaming quality in a channel at each time is evaluated as the *percentage of high-quality peers in the channel*, where a high-quality peer has a buffer count of more than 80% of the total size of its playback buffer. Representative results with a popular channel (*CCTV1*, with more than 10,000 concurrent users) and a less popular channel (*CCTV12*, with fewer than 1000 concurrent users) are shown in Fig. 1(C) and (D), respectively. The streaming quality of both channels has been decreasing over time, as server capacities are saturated. During the Chinese New Year flash crowd, the streaming quality of CCTV1 degraded significantly, due to the lack of bandwidth to serve a flash crowd of users in the channel.

Would it be possible that the lack of peer bandwidth contribution has overwhelmed the servers? As we noted, the protocol in UUSee uses optimizing algorithms to maximize peer upload bandwidth utilization, which in our opinion represents one of the state-of-the-art peer strategies in P2P streaming. The following back-of-the-envelope calculation with data from the traces may be convincing: At one time on October 15, 2006, about $100,000$ peers in the entire network have each achieved a streaming rate around 400 Kbps, by consuming a bandwidth level of 2 Gbps from the servers. The upload bandwidth contributed by peers can be computed as $100,000 \times 400 - 2,000,000 = 38,000,000$ Kbps, which is 380 Kbps per peer on average. This represents quite an achievement, as most of the UUSee clientele are ADSL users in China, with a maximum of 500 Kbps upload capacity.

Indeed, server capacities have increasingly become a bottleneck in real-world P2P live streaming solutions.

B. *Increasing volume of inter-ISP traffic*

The current UUSee protocol is not aware of ISPs. We now investigate the volume of inter-ISP traffic during the 7-month period, computed as the throughput sum of all links across ISP boundaries at each time, by mapping IP addresses to the ISPs using a database from UUSee. Fig. 2 reveals that both the inter-ISP peer-to-peer and server-to-peer traffic have been increasing, quadrupled over the 7-month period, due to the increased number of channels and peers.

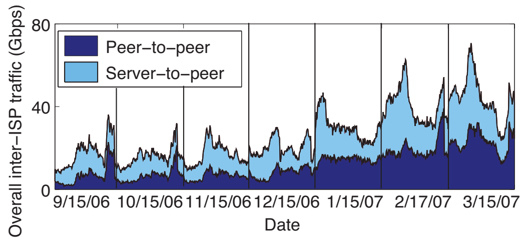


Fig. 2. The volume of inter-ISP traffic increases over time.

In China, the two nation-wide ISPs, *Netcom* and *Telecom*, charge each other based on the difference of inter-ISP traffic volume in both directions, and regional ISPs are charged based on traffic to and from the nation-wide ISPs. Both charging mechanisms have made it important for ISPs to limit inter-ISP traffic. Considering the large and persistent bandwidth consumption for live streaming, we believe that P2P streaming systems should be designed to minimize inter-ISP traffic, which remains one of our objectives in this paper.

C. What is the required server bandwidth for each channel?

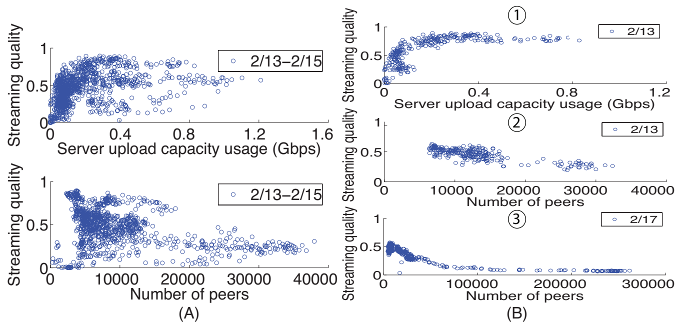


Fig. 3. Relationship among server upload bandwidth, number of peers, and streaming quality for channel CCTV1.

To determine the amount of server bandwidth needed for each channel, we wish to explore the relation among server upload bandwidth usage, the number of peers, and the achieved streaming quality in each channel. Based on a detailed trace analysis, we have identified no strong correlation among the quantities over a longer period of time. For example, Fig. 3(A) plots the correlation of the quantities for channel CCTV1 in a period of three days (February 13-15 2007). There do not exist any evident correlations in this period.

Nevertheless, if we focus on a shorter time scale, the correlation becomes more evident. For example, Fig. 3(B)-1 plots the correlation between server upload bandwidth usage and the streaming quality during a three-hour period (8pm-11pm) on February 13, which exhibits a positive square-root relation between the two quantities. Meanwhile, a negative correlation is shown to exist between the number of peers and the streaming quality, in Fig. 3(B)-2. We have also observed that the shape of such short-term correlation varies from time to time. For example, during the same time period on February 17, the relation between the number of peers and the streaming quality represents a reciprocal curve, as shown in Fig. 3(B)-3. We have observed from the traces that such variations exist in other channels as well, which can be attributed to the time-varying peer upload bandwidth availability in the channels.

All of our observations thus far point to the challenging nature of our problem at hand: how much server bandwidth should we allocate in each channel to assist peers in each ISP?

III. Ration: Online Server Capacity Provisioning

Our proposal is *Ration*, an online server capacity provisioning algorithm to be carried out on a per-ISP basis, that dynamically assigns a minimal amount of server capacity to each channel to achieve a desired level of streaming quality.

A. Problem formulation

We consider a P2P live streaming system with multiple channels (such as UUSee). We assume that the tracking server in the system is aware of ISPs: when it supplies any requesting peer with information of new partners, it first assigns peers (or dedicated servers) with available upload bandwidth from the same ISP. Only when no such peers or servers exist, will the tracking server assign peers from other ISPs.

The focus of *Ration* is the dynamic provisioning of server capacity in each ISP, carried out by a designated server in the ISP. In the ISP that we consider, there are a total of M concurrent channels to be deployed, represented as a set $\mathcal{C}$. There are n^c peers in channel $c, \forall c \in \mathcal{C}$. Let s^c denote the server upload bandwidth to be assigned to channel c, and q^c denote the streaming quality of channel c, *i.e.*, the percentage of high-quality peers in the channel that have a buffer count of more than 80% of the size of its playback buffer. Let U be the total amount of server capacity to be deployed in the ISP.[1] We assume a priority level p^c for each channel c, that can be assigned different values by the P2P streaming solution provider to reflect the relative importance of the channels.

At each time t, *Ration proactively* computes the amount of server capacity s^c_{t+1} to be allocated to each channel c for time $t+1$, that achieves optimal utilization of the limited overall server capacity across all the channels, based on their priority and popularity (as defined by the number of peers in the channel) at time $t+1$. Such an objective can be formally represented by the optimization problem **Provision(t+1)** as follows ($\forall t = 1, 2, \ldots$), in which a *streaming quality function* F^c_{t+1} is included to represent the relationship among q^c, s^c and n^c at time $t+1$:

Provision(t+1):

$$\max \sum_{c \in \mathcal{C}} p^c n^c_{t+1} q^c_{t+1} \tag{1}$$

subject to

$$\begin{aligned} &\textstyle\sum_{c\in\mathcal{C}} s^c_{t+1} \le U, \\ &q^c_{t+1} = F^c_{t+1}(s^c_{t+1}, n^c_{t+1}), \quad \forall c \in \mathcal{C}, \\ &0 \le q^c_{t+1} \le 1, s^c_{t+1} \ge 0, \quad \forall c \in \mathcal{C}. \end{aligned} \tag{2}$$

[1] U can be implemented in practice with a number of servers deployed, with the number decided by U and the upload capacity of each server.

Weighting the streaming quality q^c_{t+1} of each channel c with its priority p^c, the objective function in (1) reflects our wish to differentiate channel qualities based on their priorities. With channel popularity n^c_{t+1} in the weights, we aim to provide better streaming qualities for channels with more peers. Noting that $n^c q^c$ represents the number of high-quality peers in channel c, in this way, we guarantee that, overall, more peers in the network can achieve satisfying streaming qualities.

The challenges in solving **Provision(t+1)** at time t to derive the optimal values of $s^{c*}_{t+1}, \forall c \in \mathcal{C}$, lie in (1) the uncertainty of the channel popularity n^c_{t+1}, *i.e.*, the number of peers in each channel in the future, and (2) the dynamic relationship F^c_{t+1} among q^c, s^c, and n^c of each channel c at time $t+1$. In what follows, we present our solutions to both challenges.

B. Active prediction of channel popularity

We first estimate the number of active peers in each channel c at the future time $t+1$, *i.e.*, $n^c_{t+1}, \forall c \in \mathcal{C}$. Existing work has been modeling the evolution of the number of peers in P2P streaming systems based on Poisson arrivals and Pareto life time distributions (*e.g.*, [4]). We argue that these models represent ideal simplifications of real-world P2P live streaming systems, where peer dynamics are actually affected by many random factors. To dynamically and accurately predict the number of peers in a channel, we employ time series forecasting techniques. We treat the number of peers in each channel c, *i.e.*, $n^c_t, t = 1, 2, \ldots$, as an unknown random process evolving over time, and use the recent historical values to forecast the most likely values of the process in the future.

As the time series of channel popularity is generally non-stationary (*i.e.*, its values do not vary around a fixed mean), we utilize the *autoregressive integrated moving average* model, ARIMA(p,d,q), which is a standard linear predictor to tackle non-stationary time series. With ARIMA(p,d,q), a time series, $z_t, t = 1, 2, \ldots$, is differenced d times to derive a stationary series, $w_t, t = 1, 2, \ldots$, and each value of w_t can be expressed as the linear weighted sum of p previous values in the series, $w_{t-1}, \ldots, w_{t-p}$, and q previous random errors, $a_{t-1}, \ldots, a_{t-q}$. The employment of an ARIMA(p,d,q) model involves two steps: (1) model identification, *i.e.*, the decision of model parameters p, d, q, and (2) model estimation, *i.e.*, the estimation of $p+q$ coefficients in the linear weighted summation.

For model identification of time series $n^c_t, t = 1, 2, \ldots$, we have derived $d = 2$ based on the differencing analysis of actual channel popularity time series from the UUSee traces, and have derived $p = 0$ and $q = 1$ with standard model identification techniques using autocorrelation and partial autocorrelation functions for the differenced time series ([5], pp. 187). Due to space constraints, interested readers are referred to our technical report [6] for details. Having identified an ARIMA(0,2,1) model, the channel popularity prediction for time $t+1$, $\bar{n}^c_{t+1}$, can be expressed as follows:

$$\bar{n}^c_{t+1} = 2n^c_t - n^c_{t-1} + a_{t+1} - \theta a_t, \tag{3}$$

where θ is the coefficient for the random error term a_t and can be estimated with a least squares algorithm. When we use (3) for prediction in practice, the random error at future time $t+1$, *i.e.*, a_{t+1}, can be treated as zero, and the random error at time t can be approximated by $a_t = n^c_t - \bar{n}^c_t$ [5]. Therefore, the prediction function is simplified to

$$\bar{n}^c_{t+1} = 2n^c_t - n^c_{t-1} - \theta(n^c_t - \bar{n}^c_t). \tag{4}$$

To dynamically refine the model for accurate prediction of popularity of a channel c over time, we propose to carry out the forecasting in a dynamic fashion: To start, the ARIMA(0,2,1) model is trained with channel popularity statistics in channel c in the most recent N_1 time steps, and the value of coefficient θ is derived. Then at each following time t, $\bar{n}^c_{t+1}$ is predicted using (4), and the confidence interval of the predicted value (at a certain confidence level, *e.g.*, 95%) is computed. When time $t+1$ comes, the actual number of peers, n^c_{t+1}, is collected and tested against the confidence bounds. If the real value lies out of the confidence interval and such prediction errors have occurred T_1 out of T_2 consecutive times, the forecasting model is retrained, and the above process repeats.

C. Dynamic learning of the streaming quality function

Next, we dynamically derive the relationship among streaming quality, server bandwidth usage, and the number of peers in each channel c, denoted as the streaming quality function F^c in (2), with a statistical regression approach.

From the traces, we have observed $q^c \propto (s^c)^{\alpha^c}$ at short time scales, where α^c is the exponent of s^c, *e.g.*, $q^c \propto (s^c)^{0.5}$ in Fig. 3(B)-1. We also observed $q^c \propto (n^c)^{\beta^c}$, where β^c is the exponent of n^c, *e.g.*, $q^c \propto (n^c)^{-1}$ in Fig. 3(B)-3. As we have made similar relationship observations from a broad trace analysis of channels over different times, we model the streaming quality function as

$$q^c = \gamma^c (s^c)^{\alpha^c} (n^c)^{\beta^c}, \tag{5}$$

where $\gamma^c > 0$ is a weight parameter. Such a function model is advantageous in that it can be transformed into a multiple linear regression problem, by taking logarithm at both sides:

$$\log(q^c) = \log(\gamma^c) + \alpha^c \log(s^c) + \beta^c \log(n^c).$$

Let $Q^c = \log(q^c)$, $S^c = \log(s^c)$, $N^c = \log(n^c)$, $\Gamma^c = \log(\gamma^c)$. We derive the following multiple linear regression problem

$$Q^c = \Gamma^c + \alpha^c S^c + \beta^c N^c + \epsilon^c, \tag{6}$$

where S^c and N^c are regressors, Q^c is the response variable, and ϵ^c is the error term. Γ^c, α^c, and β^c are regression parameters, which can be estimated with least squares algorithms.

As we have observed in trace analysis that the relationship in (5) is evident on short time scales but varies over a longer term, we dynamically re-learn the regression model in (6) for each channel c in the following fashion: To start, the designated server trains the regression model with collected channel popularity statistics, server bandwidth usage and channel streaming quality during the most recent N_2 time steps, and derives the values of regression parameters. At each following time t, it uses the model to estimate the streaming quality based on the used server bandwidth and the collected number of peers in the channel at t, and examines the fitness of the current regression model by comparing the estimated value with the collected actual streaming quality. If the actual value exceeds the confidence interval of the predicted value for T_1 out of T_2 consecutive times, the regression model is retrained with the most recent historical data.

We note that the signs of exponents α^c and β^c in (5) reflect positive or negative correlations between the streaming quality and its two deciding variables, respectively. Intuitively, we should always have $0 < \alpha^c < 1$, as the streaming quality could not be worse when more server capacity is provisioned, and its improvement slows down with more and more server capacity provided, until it finally reaches the upper bound of 1. On the other hand, the sign of β^c may be uncertain, depending on the peer upload bandwidth availability at different times: if more peers with high upload capacities (*e.g.*, Ethernet peers) are present, the streaming quality can be improved with more peers in the channel ($\beta^c > 0$); otherwise, more peer joining the channel could lead to a downgrade of the streaming quality ($\beta^c < 0$).

D. Optimal allocation of server capacity

Based on the predicted channel popularity and the most recently derived streaming quality function for each channel, we are now ready to proactively assign the optimal amount of server capacity to each channel for time $t+1$, by solving problem **Provision(t+1)** in (1). Replacing q^c with its function model in (5), we transform the problem in (1) into:

Provision(t+1)':

$$\max G \tag{7}$$

subject to

$$\textstyle\sum_{c\in\mathcal{C}} s^c_{t+1} \le U, \tag{8}$$

$$s^c_{t+1} \le B^c_{t+1}, \qquad \forall c \in \mathcal{C}, \tag{9}$$

$$s^c_{t+1} \ge 0, \qquad \forall c \in \mathcal{C}, \tag{10}$$

where the objective function $G = \sum_{c\in\mathcal{C}} p^c n^c_{t+1} q^c_{t+1} = \sum_{c\in\mathcal{C}} p^c \gamma^c (n^c_{t+1})^{(1+\beta^c)} (s^c_{t+1})^{\alpha^c}$, and $B^c_{t+1} = (\gamma^c (n^c_{t+1})^{\beta^c})^{-\frac{1}{\alpha^c}}$, denoting the maximal server capacity requirement for channel c at time $t+1$, that achieves $q^c_{t+1} = 1$.

The optimal server bandwidth provisioning for each channel, $s^{c*}_{t+1}, \forall c \in \mathcal{C}$, can be obtained with a water-filling approach. The implication of the approach is to maximally allocate the server capacity, at the total amount of U, to the channels with the current largest marginal utility, as computed with $\frac{dG}{ds^c_{t+1}}$, as long as the upper bound of s^c_{t+1} indicated in (9) has not been reached.

In *Ration*, the server capacity assignment is periodically carried out to adapt to the changing demand in each of the channels over time. To minimize the computation overhead, we propose an *incremental water-filling approach*, that adjusts server capacity shares among the channels from their previous values, instead of a complete re-computation from the very beginning:

The approach starts with $s^c_{t+1} = s^{c*}_t, \forall c \in \mathcal{C}$. It first computes whether there exists any surplus of the overall provisioned server capacity, that occurs when not all the server capacity has been used with respect to the current allocation, *i.e.*, $U - \sum_{c\in\mathcal{C}} s^c_{t+1} > 0$, or the allocated capacity of some channel c exceeds its maximal server capacity requirement for time $t+1$, *i.e.*, $s^c_{t+1} > B^c_{t+1}$. If so, it adds up the surpluses and allocates them to the channels whose maximal server capacity requirement has not been reached, starting from the channel with the current maximal marginal utility ($\frac{dG}{ds^c_{t+1}}$). After this, it further adjusts the server capacity assignment towards the achievement of a same marginal utility (water-level) across the channels, by repeatedly identifying the channel with the current smallest marginal utility and the channel with the current largest marginal utility, and moving bandwidth from the former to the latter. This process repeats until all channels have reached the same marginal utility, or have reached their respective maximum server bandwidth requirement.

In our accompanying technical report [6], we have included detailed steps of the incremental water-filling approach and more discussions based on a graphical illustration. Interested readers are referred to [6] due to space constraints.

Theorem 1. *Given the channel popularity prediction $n^c_{t+1}, \forall c \in \mathcal{C}$, and the most recent streaming quality function $q^c_{t+1} = \gamma^c (s^c_{t+1})^{\alpha^c} (n^c_{t+1})^{\beta^c}$, $\forall c \in \mathcal{C}$, the incremental water-filling approach obtains an optimal server capacity provisioning across all the channels for time $t+1$,* i.e., *$s^{c*}_{t+1}, \forall c \in \mathcal{C}$, which solves the problem **Provision(t+1)** in (1).*

Again, interested readers are referred to [6] for the proof.

E. Ration: the complete algorithm

Our complete algorithm is summarized in Table I, which is periodically carried out on a designated server in each ISP. The only peer participation required is to have each peer in the ISP send periodical heartbeat messages to the server, each of which includes its current playback buffer count.

We note that in practice, the allocation interval is decided by the P2P streaming solution provider based on need, *e.g.*, every 30 minutes, and peer heartbeat intervals can be shorter, *e.g.*, every 5 minutes. To train ARIMA(0,2,1), generally no more than $30 - 50$ samples are required, *i.e.*, $N_1 < 50$, and even less samples are needed to learn the streaming quality function. Therefore, only a small amount of historical data needs to be maintained at the server for the execution of *Ration*.

F. Practical implications

Finally, we discuss the practical application of *Ration* in real-world P2P live streaming systems. In such systems with unknown demand for server capacity in each ISP, *Ration* can make full utilization of the currently provisioned server capacity, U, and meanwhile provide excellent guidelines for the adjustment of U, based on different relationships between the supply and demand for server capacity.

If the P2P streaming system is operating at the over-provisioning mode in an ISP, *i.e.*, the total deployed server capacity exceeds the overall demand from all channels to achieve the required streaming rate at their peers, *Ration*

TABLE I
RATION: THE ONLINE SERVER CAPACITY PROVISIONING ALGORITHM

```
At time $t$, the designated server in each ISP

1. Peer statistics collection
   Collect the number of active peers in each channel, $n_t^c, \forall c \in \mathcal{C}$, with peer heartbeat messages.
   Collect per-peer buffer count statistics from the heartbeat messages, and derive the streaming quality for each channel, $q_t^c, \forall c \in \mathcal{C}$.

2. Channel popularity prediction for each channel $c \in \mathcal{C}$
   Test if $n_t^c$ is within the confidence interval of $\bar{n}_t^c$, the value predicted at time $t-1$.
   If $n_t^c$ lies out of the confidence interval for $T_1$ out of $T_2$ consecutive times, retrain the ARIMA(0,2,1) model with the most recent $N_1$ peer number statistics.
   Predict the channel popularity at time $t+1$ by $\bar{n}_{t+1}^c = 2n_t^c - n_{t-1}^c - \theta(n_t^c - \bar{n}_t^c)$, where $\theta$ is the parameter in ARIMA(0,2,1).
$\rightarrow$ Channel popularity predictions for all channels are derived.

3. Learning the streaming quality function for each channel $c \in \mathcal{C}$
   Estimate the streaming quality for time $t$ with the current streaming quality function model: $\bar{q}_t^c = \gamma^c (s_t^c)^{\alpha^c} (n_t^c)^{\beta^c}$.
   Test if the actual $q_t^c$ is within the confidence interval of $\bar{q}_t^c$.
   If $q_t^c$ lies out of the confidence interval for $T_1$ out of $T_2$ consecutive times, retrain the regression model in Eq. (6) with statistics in the most recent $N_2$ times.
$\rightarrow$ The current streaming quality functions for all channels are derived.

4. Proactive server capacity provisioning for all the channels
   Adjust server capacity assignment among all the channels with the incremental water-filling approach in Sec. III-D.
$\rightarrow$ Optimal server capacity provisioning,$s_{t+1}^{c*}, \forall c \in \mathcal{C}$,is derived.
```

derives the minimal amount of server capacity needed for each channel c to achieve its best streaming quality, represented as $q^c = 1$. This is guaranteed by (9) in **Provision(t+1)'**, as the server capacity provisioned to each channel may not exceed the amount that achieves $q^c = 1$. When the P2P streaming solution provider discovers that the system is always operating at the over-provisioning mode, they may consider to reduce their total server capacity deployment in the ISP.

If the system is operating in a mode with tight supply-demand relations, *i.e.*, the total server capacity can barely meet the demand from all channels to achieve the best streaming qualities, *Ration* guarantees the limited server capacity is most efficiently utilized across the channels, respecting their demand and priority. With its water-filling approach, the preference in capacity assignment is based on marginal utility of each channel, $\frac{dG}{ds^c} = p^c n^c \frac{dq^c}{ds^c}$, as determined by the popularity and priority of the channel, and the marginal improvement of its streaming quality upon unit increase of server capacity. If the streaming solution provider wishes to improve the streaming quality of the channels in the ISP, they may further compute how much more server capacity to be added, using the derived streaming quality function in (5).

If the system is operating with extremely tight supply-demand relations, *e.g.*, the flash crowd scenario, most server capacity is assigned to the one or few channels that are involved in the flash crowd, and most of the other channels are starving with no or very little server bandwidth. Upon detecting this, our algorithm can trigger the deployment of backup server resources. Similarly, the extra amount to add can be computed with the current streaming quality function derived for the respective channels.

In addition, with *Ration*, the P2P streaming solution provider can dynamically make decisions on channel deployment in each ISP, when it is not possible or necessary to deploy every one of the hundreds or thousands of channels in each ISP. When a channel is not allocated any server capacity due to very low popularity or priority during a period of time, the channel is not to be deployed in the ISP during this time.

IV. EXPERIMENTAL EVALUATIONS WITH TRACE REPLAY

Our evaluation of *Ration* is based on its implementation in a multi-ISP mesh-based P2P streaming system, which replays real-world streaming scenarios captured by the traces.

The P2P streaming system is implemented in C++ on a high-performance cluster of 50 Dell 1425SC and Sun v20z dual-CPU servers, interconnected by Gigabit Ethernet. On this platform, we are able to emulate hundreds of concurrent peers on each cluster server, and emulate all network parameters, such as node/link capacities. Actual media streams are delivered over TCP connections among the peers, and control messages are sent by UDP. The platform supports multiple event-driven asynchronous timeout mechanisms with different timeout periods, and peer joins and departures are emulated with events scheduled at their respective times.

The P2P streaming protocol we implemented includes both the standard pull protocol and the unique algorithms employed by UUSee, as introduced in Sec. II. Without loss of generality, we deploy one server for each ISP, implementing both the tracking server and streaming server functions. *Ration* is also implemented on each of the servers, with 800 lines of C++ code. The server capacity allocation for each channel is implemented by limiting the total number of bytes sent over the outgoing connections from the server for the channel in each unit time.

Our experiments are carried out on realistic replays of the traces. We emulate peer dynamics based on the evolution of the number of peers in each channel from the traces: when the number of peers rises between two consecutive time intervals, we schedule a corresponding number of peer join events during the interval; when the number of peers decreases, peer departure events are scheduled for a corresponding number of randomly selected peers. Upon arrival, each peer acquires 30 initial upstream peers, and the P2P topology evolves based on the same peer selection protocol as UUSee's. The node upload capacities are emulated using values from the traces, which follow a heavy-tail distribution in the major range of 50 Kbps to 10 Mbps. The streaming rate of each channel is 400 Kbps, with the streams divided into 1-second blocks for distribution. The size of playback buffer on the peers is set to 30 seconds. Each peer reports its buffer count to the server in its ISP every 20 seconds, and the server processes them and adjusts capacity allocation every 60 seconds.

A. *Performance of Ration components*

We first examine the effectiveness of each composing algorithm in *Ration*. In this set of experiments, we focus on the streaming inside one ISP, with one server of 80 Mbps upload capacity and 5 channels. We use the peer number statistics of 5 channels from the traces, *CCTV1*, *CCTV4*, *CCTV2*, *CCTV7*, and *CCTV12*, in one ISP (China *Telecom*) during the week of February 13 – 19, 2007.[2] The 5 channels have a regular instantaneous number of peers at the scale of 2000, 500, 400, 150 and 100, respectively. The statistics of CCTV1 and CCTV4 also captured the flash crowd scenario on February 17, when the Chinese New Year celebration show was broadcast on the two channels.

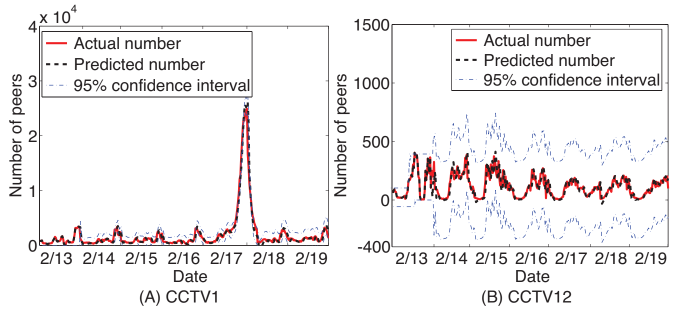


Fig. 4. Prediction of the number of peers with ARIMA(0,2,1).

1) Prediction of the number of peers: Fig. 4 presents the results of prediction with ARIMA(0,2,1) for the popular channel CCTV1 and the unpopular channel CCTV12, respectively. In the dynamic prediction, the training set size is $N_1 = 30$, and the error count parameters are $T_1 = 8$ and $T_2 = 10$. The predicted numbers for both channels largely coincide with the actually collected number of peers, both at regular times and during the flash crowd, no matter whether the prediction confidence interval is large or small at different times. This validates the correctness of our model identification, as well as the accuracy of our dynamic prediction.

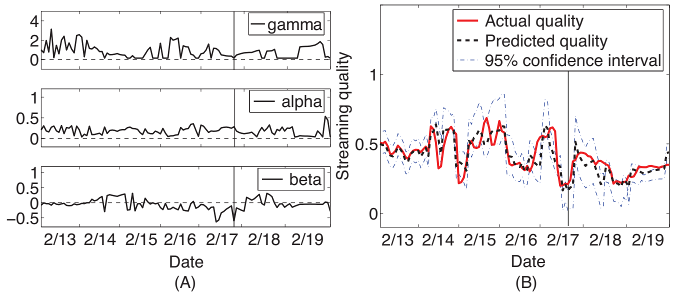


Fig. 5. Dynamic learning of the streaming quality function for CCTV1.

2) Dynamic streaming quality function: Fig. 5(A) plots the derived parameter values for the dynamic streaming quality function of CCTV1. In the dynamic regression, the training set size is $N_2 = 20$, the error count parameters are $T_1 = 8$ and $T_2 = 10$. We see that γ^c is all positive, the values of α^c are always within the range of $0-1$, and β^c may be positive or negative at different times. We have observed similar results with the derived streaming quality functions of other channels. This validates our analysis in the last paragraph of Sec. III-C. During the flash crowd scenario, which hereinafter is marked with a vertical line in the figures, β^c is significantly below zero, revealing a negative impact on the streaming quality with a rapidly increasing number of peers in the channel.

[2]To expedite our experiments, each peer number time series from the traces is sampled, such that the evolution of the P2P system in each day is emulated within half an hour.

Fig. 5(B) plots the actually measured streaming quality in the channel against its estimated value, calculated with the derived streaming quality function at each time. The actual streaming quality closely follows the predicted trajectory at most times, including the flash crowd scenario, which exhibits the effectiveness of our dynamic regression.

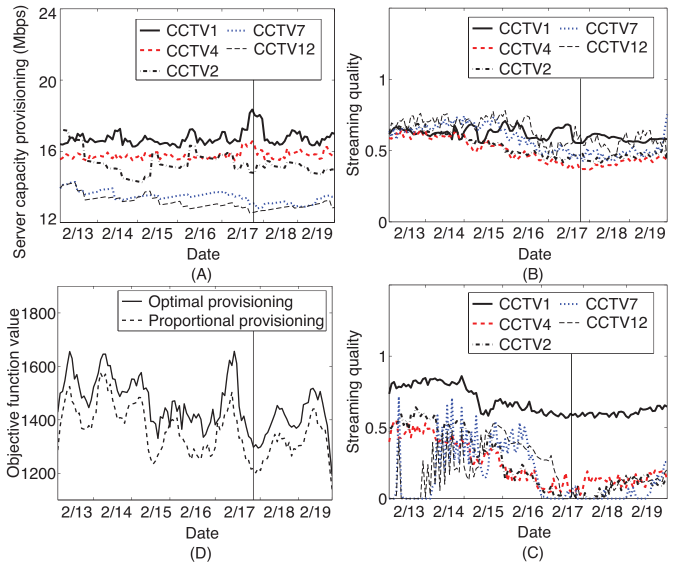


Fig. 6. Server capacity provisioning for 5 non-prioritized channels: (A) Server capacity provisioning with *Ration*, (B) Streaming quality achieved with *Ration*, (C) Streaming quality achieved with proportional allocation, (D) Comparison of objective function values.

3) Optimal provisioning among all channels: We now investigate the optimal server capacity provisioned to different channels over time. In this experiment, we focus on examining the effects of channel popularity on capacity allocation, and set the priorities for all 5 channels to the same value of 1.

In Fig. 6(A), we observe that, generally speaking, the higher the channel's popularity is, the more server capacity it is assigned. This can be explained by the marginal utility of the channels used in the water-filling allocation of *Ration*, $\frac{dG}{ds^c} = p^c n^c \frac{dq^c}{ds^c} = \frac{p^c \gamma^c \alpha^c (n^c)^{(1+\beta^c)}}{(s^c)^{1-\alpha^c}}$. As $\beta^c > -1$ is observed in our previous experiment, the marginal utility is positively correlated with the number of peers, and thus the more popular channel is assigned more server capacity.

On the other hand, in Fig. 6(B), we do not observe evident correlation between the channel popularity and its achieved streaming quality, as the latter is decided by both the allocated server capacity (positively) and the number of peers (positively or negatively at different times). Nevertheless, we show that our water-filling assignment achieves the best utilization of the limited overall server capacity at all times, with a comparison study to a proportional allocation approach.

The proportional allocation approach goes as follows: At each time t, instead of using water-filling, the server capacity is proportionally allocated to the channels, based only on their predicted number of peers for time $t+1$. Fig. 6(C) shows that the most popular channel, CCTV1, achieves better streaming quality with this proportional allocation as compared to that in Fig. 6(B), at the price of downgraded quality for the other channels, especially during the flash crowd. This is because CCTV1 now obtains more than half of the total server capacity at regular times, and almost all during the flash crowd scenario.

With the streaming quality results in Fig. 6(B) and (C), we compute the values of the objective function of **Provision(t+1)** in (1), and plot them in Fig. 6(D). Given a same priority for all the channels, the value of the objective function at each time represents the total number of peers in all the channels that achieve satisfying streaming rates at the time. The values from the proportional allocation are consistently lower than those achieved with our water-filling approach, exhibiting the optimality of the server capacity utilization with *Ration*.

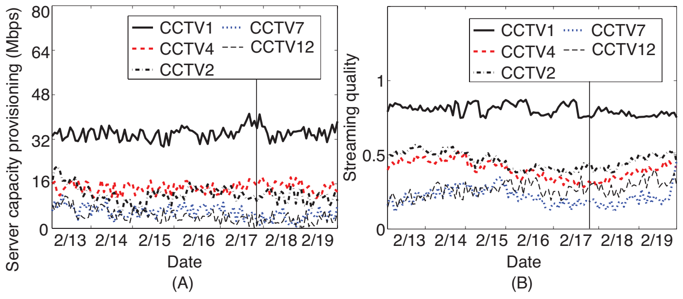


Fig. 7. Server capacity provisioning for 5 prioritized channels with *Ration*.

4) Effectiveness of channel prioritization: In the next experiment, we investigate the effect of channel prioritization on server capacity provisioning with *Ration*, by setting 3 priority levels: $p^c = 500$ for CCTV1, $p^c = 200$ for CCTV4 and CCTV2, and $p^c = 50$ for CCTV7 and CCTV12.

Comparing its results in Fig. 7(A) to those in Fig. 6(A), we observe further differentiated server capacities among the channels, where the channel with the highest priority and popularity, CCTV1, is allocated much more capacity than the others. In Fig. 7(B), we also observe differentiated streaming qualities among channels based on their priority levels. These demonstrate the effectiveness of channel prioritization in *Ration*, which facilitates the streaming solution provider to differentiate services across channels, when the supply-demand relation of server capacity is tight in the system.

B. Effectiveness of ISP-aware server capacity provisioning

Next, we evaluate *Ration* in multi-ISP streaming scenarios. 4 ISPs are emulated by tagging servers and peers with their ISP IDs. Again, 5 channels, *CCTV1*, *CCTV4*, *CCTV2*, *CCTV7*, *CCTV12*, are deployed in the ISPs, with peer number statistics in each ISP extracted from those in 4 China ISPs, *Telecom*, *Netcom*, *Unicom* and *Tietong*, from the traces. While a fixed overall server capacity is used in the previous experiments, in the following experiments, we do not cap the server capacity, but derive with *Ration* the minimal amount of overall server capacity needed to achieve the best streaming qualities for all the channels in the system (*i.e.*, $q^c = 1, \forall c \in \mathcal{C}$), which is referred to as U_B hereinafter. At each time during the dynamic provisioning, U_B is derived by summing up the upper bound of server capacity required for each of the channels, as given in (9), at the time. Our focus is to compare the total server capacity U_B required when ISP awareness is in place and not, and the inter-ISP traffic that is caused. The channels are not prioritized in this set of experiments.

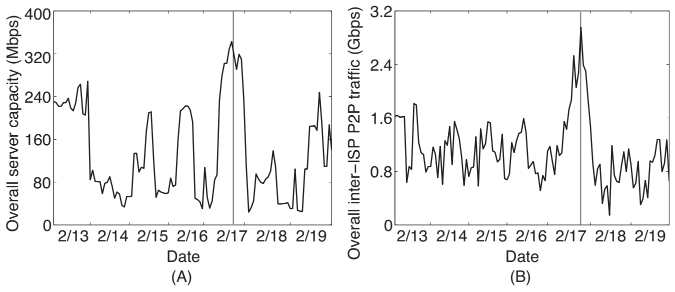
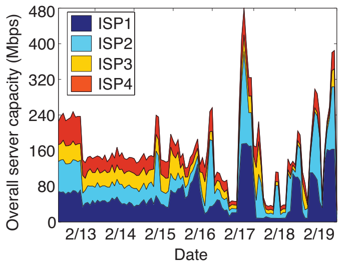


Fig. 8. P2P live streaming for 5 channels in 4 ISPs: without ISP awareness.

1) Without ISP awareness: In the first experiment, we deploy one server in the system, and stream with a peer selection protocol that is not ISP-aware. The overall server capacity U_B used on the server over time is shown in Fig. 8(A), and the total inter-ISP P2P traffic in the system is plotted in Fig. 8(B).



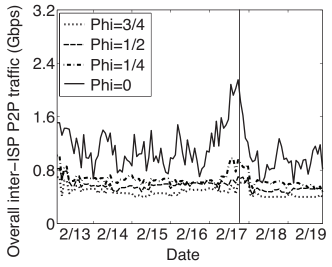


Fig. 9. P2P live streaming for 5 channels in 4 ISPs: with full ISP awareness.

Fig. 10. Server capacity provisioning vs. inter-ISP traffic: a tradeoff.

2) With full ISP awareness: In the second experiment, we deploy one server in each ISP, and constrain all streaming traffic inside each ISP by fully ISP-aware peer selection, *i.e.*, peers are only assigned partners inside the ISP. The server capacity used on the server in each ISP is illustrated with the area plot in Fig. 9. Comparing Fig. 9 to Fig. 8(A), we can see that the overall server capacity usage in the system does not increase much when the traffic is completed restricted inside each ISP with per-ISP server capacity deployment. The difference is only larger during the flash crowd, the reason for which, we believe, is that it becomes hard for peers to identify enough supplying peers inside the ISP when the total number of peers soars, and thus they have to resort to the server.

3) Tradeoff between server capacity and inter-ISP traffic: In the final experiment, we provision a total server capacity in the system that is between the amount used in 1) and that used in 2), and examine the resulting inter-ISP traffic. Specifically, let the overall server capacity usage shown in Fig. 8(A) be U_{Bmin} and that shown in Fig. 9 be U_{Bmax}. We reduce the server capacity provisioned on each server in each ISP, such that the overall server capacity at each time is at the value of

$U_{Bmin} + \phi(U_{Bmax} - U_{Bmin})$ at the time. In this case, peers are allowed to connect to servers/peers across ISPs if they fail to acquire sufficient streaming bandwidth within the ISP.

The experiment is repeated by setting ϕ to $\frac{3}{4}, \frac{1}{2}, \frac{1}{4}$ or 0, that represent different levels of the total server capacity. The results in Fig. 10 show an increase of inter-ISP traffic with the decrease of server capacity provisioning. Further comparing the $\phi = 0$ case in Fig. 10 to Fig. 8(B), we observe that while the total server capacity is the same U_{Bmin} in both cases, a smaller amount of inter-ISP P2P traffic is involved with the ISP-aware peer selection than without any ISP awareness.

V. Related Work

With the successful Internet deployment of mesh-based P2P streaming systems [7], [8], [9], significant research efforts have been devoted to their measurement and improvement. With respect to measurements, existing studies [8], [9], [10] mostly focus on the behavior of peers, with little attention devoted to the streaming servers, which nevertheless contribute significantly to the stability of P2P live streaming.

Since the seminar work of Coolstreaming [7], various improvements of peer strategies in such mesh-based P2P streaming have been proposed, *e.g.*, the enhancement of the block pulling mechanism [11], the optimization of peer connectivity for content swarming [12], the exploration of inter-overlay cooperation [13]. To the best of our knowledge, this paper presents the first detailed measurements of server capacity utilization in a live P2P streaming system, and the first online *server capacity* provisioning mechanism to address the dynamic demand in multiple concurrent channels.

With respect to analytical work touching on the subject of server capacity, Das *et al.* [14] have shown with a fluid model the effects of server upload capacities on the average peer download time in BitTorrent-like P2P file sharing applications. Also based on fluid theory, Kumar *et al.* [4] have modeled the streaming quality in a mesh-based P2P streaming system in terms of both server and peer upload capacities. As compared to these analytical studies, our work focuses entirely on the *practicality* of a dynamic server capacity provisioning mechanism. Other than using simplified modeling assumptions, we employ time series forecasting techniques to derive the evolution of the number of peers, and use dynamic regression approaches to learn the relation among the streaming quality, server capacity and the number of peers at different times.

There have recently emerged a number of discussions about the large amount of inter-ISP traffic brought by P2P applications, with respect to BitTorrent file sharing [15], [16], P2P Video on Demand [2], and P2P software update distribution [17]. Approaches for the localization of P2P traffic inside ISP boundaries have been proposed, which mostly focus on ISP-aware peer selection. In contrast, our study is the first to investigate the impact and evolution of inter-ISP P2P live streaming traffic, and our proposal emphasizes on the dynamic provisioning of server capacity on a per-ISP basis to maximally guarantee the success of ISP-aware P2P streaming.

VI. Concluding Remarks

This paper focuses on dynamic server capacity provisioning in multi-ISP multi-channel P2P live streaming systems. In practice, we believe that it is important to refocus our attention on dedicated streaming servers: based on our detailed analysis of 7 months of traces from a large-scale P2P streaming system, available server capacities are not able to keep up with the increasing demand in such real-world commercial systems, leading to a downgrade of peer streaming quality. Emphasizing on practicality, our proposed algorithm, *Ration*, is able to dynamically predict the demand in each channel, using an array of dynamic learning techniques, and to proactively provision optimal server capacities across different channels. With full ISP awareness, *Ration* is carried out on a per-ISP basis, and is able to guide the deployment of server capacities and channels in each ISP to maximally constrain P2P traffic inside ISP boundaries. Our performance evaluation of *Ration* is highlighted with the replay of real-world streaming traffic from our traces. We show that *Ration* lives up to our full expectations to effectively provision server capacities according to the demand over time.

References

[1] J. Liu, S. G. Rao, B. Li, and H. Zhang, "Opportunities and Challenges of Peer-to-Peer Internet Video Broadcast," *Proceedings of the IEEE, Special Issue on Recent Advances in Distributed Multimedia Communications*, 2007.

[2] C. Huang, J. Li, and K. W. Ross, "Can Internet Video-on-Demand Be Profitable?" in *Proc. of ACM SIGCOMM 2007*, August 2007.

[3] *UUSee Inc.*, http://www.uusee.com/.

[4] R. Kumar, Y. Liu, and K. W. Ross, "Stochastic Fluid Theory for P2P Streaming Systems," in *Proc. of IEEE INFOCOM*, May 2007.

[5] G. E. P. Box, G. M. Jenkins, and G. C. Reinsel, *Time Series Analysis: Forecasting and Control (Third Edition)*. Prentice Hall, 1994.

[6] C. Wu, B. Li, and S. Zhao, "Multi-channel Live P2P Streaming: Refocusing on Servers," http://iqua.ece.toronto.edu/papers/refocus.pdf, ECE, University of Toronto, Tech. Rep., January 2008.

[7] X. Zhang, J. Liu, B. Li, and T. P. Yum, "CoolStreaming/DONet: A Data-Driven Overlay Network for Live Media Streaming," in *Proc. of IEEE INFOCOM 2005*, March 2005.

[8] X. Hei, C. Liang, J. Liang, Y. Liu, and K. W. Ross, "A Measurement Study of a Large-Scale P2P IPTV System," *IEEE Trans. on Multimedia*, vol. 9, no. 8, pp. 1672–1687, December 2007.

[9] A. Ali, A. Mathur, and H. Zhang, "Measurement of Commercial Peer-To-Peer Live Video Streaming," in *Proc. of Workshop in Recent Advances in Peer-to-Peer Streaming*, August 2006.

[10] T. Silverston and O. Fourmaux, "Measuring P2P IPTV Systems," in *Proc. of the 17th International workshop on Network and Operating Systems Support for Digital Audio & Video (NOSSDAV'07)*, June 2007.

[11] M. Zhang, J. Luo, L. Zhao, and S. Yang, "A Peer-to-Peer Network for Live Media Streaming - Using a Push-Pull Approach," in *Proc. of ACM Multimedia 2005*, November 2005.

[12] N. Magharei and R. Rejaie, "PRIME: Peer-to-Peer Receiver-drIven MEsh-based Streaming," in *Proc. of IEEE INFOCOM 2007*, May 2007.

[13] X. Liao, H. Jin, Y. Liu, L. M. Ni, and D. Deng, "AnySee: Peer-to-Peer Live Streaming," in *Proc. of IEEE INFOCOM 2006*, March 2006.

[14] S. Das, S. Tewari, and L. Kleinrock, "The Case for Servers in a Peer-to-Peer World," in *Proc. of IEEE International Conference on Communications (ICC 2006)*, June 2006.

[15] T. Karagiannis, P. Rodriguez, and K. Papagiannaki, "Should Internet Service Providers Fear Peer-Assisted Content Distribution," in *Proc. of the Internet Measurement Conference (IMC'2005)*, October 2005.

[16] R. Bindal, P. Cao, W. Chan, J. Medval, G. Suwala, T. Bates, and A. Zhang, "Improving Traffic Locality in BitTorrent via Biased Neighbor Selection," in *Proc. of the 26th IEEE International Conference on Distributed Computing Systems (ICDCS 2006)*, July 2006.

[17] C. Gkantsidis, T. Karagiannis, P. Rodriguez, and M. Vojnovic, "Planet Scale Software Updates," in *Proc. of ACM SIGCOMM*, September 2006.