On Dynamic Server Provisioning in Multi-channel P2P Live Streaming

Chuan Wu
Department of Computer Science
The University of Hong Kong
Hong Kong, China
cwu@cs.hku.hk

Baochun Li
Department of Electrical and Computer Engineering
University of Toronto, Canada
bli@eecg.toronto.edu

Shuqiao Zhao
Multimedia Development Group
UUSee, Inc.
China
zhaoshq@uusee.com

***Abstract*—To guarantee the streaming quality in live peer-to-peer (P2P) streaming channels, it is preferable to provision adequate levels of upload capacities at dedicated streaming servers, compensating for peer instability and time-varying peer upload bandwidth availability. Most commercial P2P streaming systems have resorted to the practice of over-provisioning a fixed amount of upload capacity on streaming servers. In this paper, we have performed a detailed analysis on 10 months of run-time traces from UUSee, a commercial P2P streaming system, and observed that available server capacities are not able to keep up with the increasing demand by hundreds of channels. We propose a novel online server capacity provisioning algorithm that proactively adjusts 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 channel priority. The algorithm is able to learn over time, has full ISP awareness to maximally constrain P2P traffic within ISP boundaries, and can provide differentiated streaming qualities to different channels by manipulating their priorities. To evaluate its effectiveness, our experiments are based on an implementation of the algorithm which replays real-world traces.**

***Index Terms*—Distributed applications, peer-to-peer streaming, server bandwidth provisioning, multiple channels**

I. Introduction

Large-scale peer-to-peer (P2P) live streaming has recently been successfully and commercially deployed [1], [2], [3], [4], in which hundreds of media channels are routinely broadcasted to hundreds of thousands 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 [5]. 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.

In this paper, however, we shift our focus to the streaming servers. Such refocusing on servers is motivated by our detailed analysis of 10 months and 800 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 commercial live streaming systems (*e.g.*, PPLive [2], PPStream [4]), in order to maintain a satisfactory and sustained streaming quality, UUSee has so far resorted to the practice of over-provisioning a fixed amount of server capacity to satisfy the streaming demand from peers in all its channels, counteracting the impact of volatile peer dynamics and time-varying peer upload bandwidth availability. Nevertheless, contrary to common belief, we have observed that the deployed capacities on streaming servers are not able to keep up with the increasing demand from hundreds of channels in practice, leading to degraded streaming quality in all channels. In response, we advocate to allocate limited server capacities to each of the channels based on their popularity and priority in order to maximally utilize dedicated servers, and also to dynamically determine the minimum overall amount of server capacity to be deployed in the system.

While it is certainly a challenge to determine the minimum amount of server bandwidth to provision 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 Internet Service Providers (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.

Towards these objectives, this paper presents *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, depending on its popularity and priority. With the analysis of our real-world traces, we have observed that the number of peers and their contributed bandwidth in each channel vary dynamically over time, and significantly affect the required bandwidth from servers. *Ration* is designed to actively *predict* the server 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 UUSee traces.

The remainder of this paper is organized as follows. In Sec. II, we motivate our focus on servers by showing our analysis of 10 months worth of traces from UUSee. In Sec. III, we present the design of *Ration*. In Sec. IV, we discuss how *Ration* may be deployed with ISP awareness to serve real-world P2P streaming systems. Sec. V 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. VI and Sec. VII, respectively.

II. Motivation from Real-world Traces

Why shall we refocus our attention to dedicated streaming servers in P2P live streaming systems? Starting September 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 systems such as PPLive, UUSee maintains a sizable array of about 150 dedicated streaming servers, to support its P2P streaming topologies with hundreds of channels to millions of users, mostly in 400 Kbps media streams. With 80% users in China, UUSee network spans over 20 ISPs in China and around 35 countries in the world. UUSee streaming protocol utilizes the "pull-based" design on mesh P2P topologies, that allows peers to serve other peers ("partners") by exchanging media blocks in their playback buffers, which represent a sliding window of the stream. When a new peer joins a channel in UUSee, the initial set of a number of partners (up to 50) is supplied by one of the tracker servers by randomly selecting from all the existing peers in the channel with available upload bandwidth. The peer establishes TCP connections with these partners, and buffer availability bitmaps (also called "buffer maps") are periodically exchanged. The buffer size at each peer in UUSee is 500 media blocks, and each block represents 1/3 second of media playback (about 10 MB in total).

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 tracker servers that it is able to receive new connections. The tracker servers keep a list of such peers, and assign them upon requests of partners from other peers. In addition, the number of consecutive blocks received and cached in the current playback buffer, starting from the current playback time, is used in UUSee protocol to represent the current streaming quality of each peer, which is referred to as the *buffering level*. During the streaming process, neighboring peers may also recommend partners to each other based on their current streaming quality. A peer may contact a tracker server again to obtain additional peers with better qualities, once it has experienced a low buffering level for a sustained period of time. Besides, UUSee has implemented a number of NAT/firewall traversal techniques based on classification of different types of user connections in its network, in order to maximize peer bandwidth contribution.

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 map, its buffering level, 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 of them. Each dedicated streaming server in UUSee utilizes a similar protocol as deployed on regular peers, is routinely selected to serve the peers, and reports its related statistics periodically as well. A detailed description on UUSee protocol and the measurement methodologies for the above metrics can be found in our previous work [6], [7].

During a 10-month period from September 2006 to July 2007, we have collected more than 800 GB worth of traces with more than 600 million unique IP addresses, representing time-continuous snapshots of the live channels sustained in UUSee every five minutes in this long period of time. Each snapshot captures the information on more than $100,000$ concurrent peers in the entire UUSee network.

A. Insufficient "supply" of server bandwidth

What have we discovered from the traces? 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 10 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; April 2007 is skipped due to lack of traces in the month caused by an upgrade of the trace servers. 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 starting 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.

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 buffering level of more than 80% of the total size of its playback buffer. The criterion of buffering level (*i.e.*, the number of consecutive blocks received and cached in the current playback buffer of a peer), has been extensively used in UUSee system to

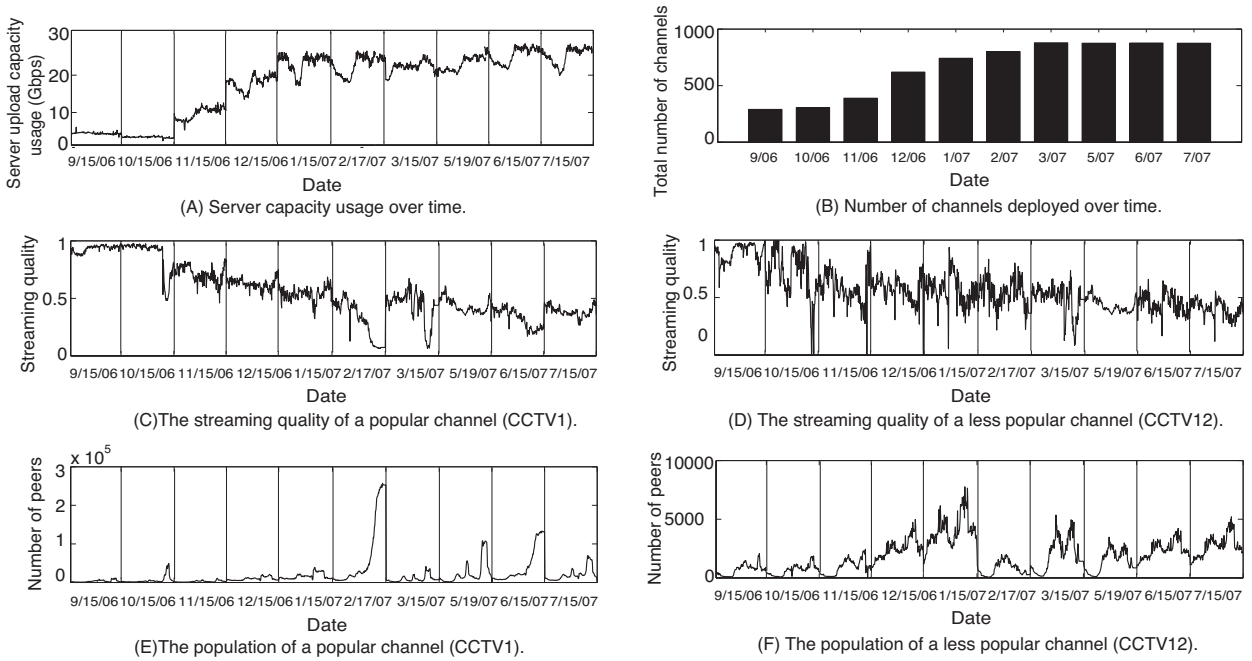

Fig. 1. The evolution of server bandwidth, channels, and streaming quality from September 2006 to July 2007.

evaluate the current streaming quality of a peer; and the 80% benchmark has empirically been shown to be effective in reflecting the playback continuity of a peer in the following few minutes, based on an internal performance monitoring system in UUSee. Accordingly, we also use the peer buffering level as our streaming quality criterion.[1] Representative results with a popular channel *CCTV1* and a less popular channel *CCTV12* are shown in Fig. 1(C) and (D), respectively, with their population measurements plotted in Fig. 1(E) and (F), 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, as illustrated in Fig. 1(E).

Would it be possible that the lack of peer bandwidth contribution has led to the overwhelming demand of the servers? As we noted, the protocol in UUSee uses optimizing algorithms to maximize peer upload bandwidth utilization, which 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, considering that most of the UUSee clientele are ADSL users in China with a maximum of 512 Kbps upload capacity, and that many random factors influence the available upload bandwidth at the peers.

In addition, one may doubt if the downgrade of streaming quality during flash crowd scenarios could have been caused by bandwidth bottlenecks within the Internet backbone at those times. Our previous measurement studies in [7] have revealed that there does not exist significant difference between bandwidth availabilities on P2P links, that are decided by Internet backbone bandwidths, at regular times and during flash crowd scenarios, and have confirmed the common belief that bandwidth constraints in P2P streaming mainly lie at the last-mile upload links at the peers and servers in most cases.

All the above observations have led to the conclusion that server capacities have increasingly become a bottleneck in real-world P2P live streaming solutions. When the server capacity usage by different channels is not regulated and is largely random (as in the current UUSee protocol), the results are less than satisfactory: Taking a typical streaming quality result of 0.5 for both CCTV1 and CCTV12, there are many more peers experiencing a low buffering level in the popular channel than in the less popular channel, considering the large difference of their populations. In practice, we may wish to provide a good streaming experience to as many peers as possible in the entire system, and therefore advocate to allocate the limited server capacities to each of the channels based on their popularity and priority, in order to maximally utilize dedicated servers.

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 10-month period, computed as the throughput sum of all links across ISP boundaries at each time. For each IP address in the traces, we derive the AS (Autonomous System) it belongs to using the Whois service provided by Cymru [8], and then map each China AS number to its affiliated ISP by making use of the

[1]Nevertheless, we have evaluated a number of other possible streaming quality metrics, such as the instantaneous streaming download rate of a peer, and still identified the one we use as the most effective in reflecting the streaming quality of peers/channels.

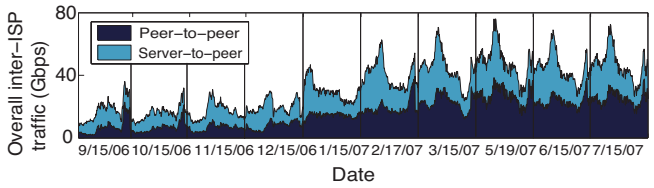

Fig. 2. The volume of inter-ISP traffic increases over time.

official mapping data provided by CERNET, China [9].[2] Fig. 2 reveals that both the inter-ISP peer-to-peer and server-to-peer traffic have been increasing, quadrupled over the 10-month period, due to the increased number of channels and peers.

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 (to avoid the fate of traffic filtering by ISPs), which remains one of our objectives in this paper.

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

To determine the amount of server bandwidth needed to achieve a specific level of streaming quality in each channel, we wish to explore the relation among server upload bandwidth, the number of peers, and the achieved streaming quality in each channel. Fig. 3(A) illustrates the evolution of the three quantities for channel CCTV1 over a period of one week, from February 13 to 19, 2007. We can observe a weak positive correlation between the server bandwidth and the streaming quality, and a weak negative correlation between the peer population and the streaming quality.

To further explore any evident correlation on a shorter time scale, we plot in Fig. 3(B)-1 the correlation between server upload bandwidth usage and the streaming quality on February 13 and that between the number of peers and the streaming quality in Fig. 3(B)-2. We can observe an evident positive relation between the two quantities in the former figure and a negative correlation in the latter. We have extensively explored such correlation in many channels over different dates and have observed that the correlation varies from one time to another even in the same channel, which can be attributed to the time-varying aggregate peer upload bandwidth in the channel over time. For example, Fig. 3(B)-3 plots the relation between the number of peers and the streaming quality on February 17, which approximates a reciprocal curve.

Another interesting observation we have made here is that: contrary to common belief, we have observed a decreasing trend of streaming quality in a streaming channel when the number of peers increases, based on our extensive study of many streaming channels over many time intervals. We believe the reason lies in that many peers cannot contribute up to the level of their required streaming rate (about 400 Kbps) in such a practical P2P system over today's Internet, which

[2]The majority of UUSee users are in China and we focus on trace data of such users in producing Fig 2.

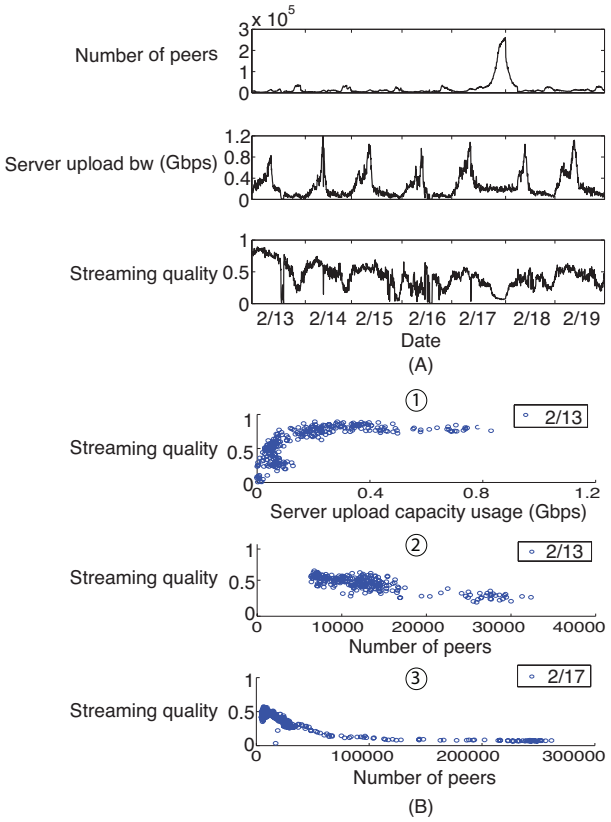

Fig. 3. Relationship among server upload bandwidth, number of peers, and streaming quality in channel CCTV1.

further justifies the necessity of deploying server capacity in the system.

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 in each ISP to assist the peers?

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 tracker 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 tracker server assign peers from other ISPs.

The focus of *Ration* is the dynamic provisioning of server capacity in each ISP,[3] 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

[3]We note that *Ration* can be extended to cases that it is not feasible to deploy servers in each ISP, by having servers in one ISP responsible to serve peers from a number of nearby ISPs.

TABLE I
NOTATION IN *Ration*

Symbol	Definition
U	the total amount of server capacity
$\mathcal{C}$	the set of channels
M	the number of channels in $\mathcal{C}$
F^c	streaming quality function of channel c
s^c	server upload bandwidth assigned to channel c
q^c	streaming quality of channel c
p^c	priority level for channel c
n^c	the number of peers in channel c
$\bar{n}^c$	the estimated number of peers in channel c
θ^c	coefficient of the random error term in the ARIMA model for channel c
α^c	exponent of s^c in streaming quality function
β^c	exponent of n^c in streaming quality function
γ^c	weight parameter in streaming quality function
G	the objective function in **Provision(t+1)'**
B^c	the maximal server capacity required for channel c to achieve $q^c = 1$

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 buffering level 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. We assume a priority level p^c for each channel c, that can be assigned different values by the P2P streaming solution provider to achieve service differentiation across the channels. We list important notations in this paper in Table I for ease of reference.

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

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

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 $s^{c*}_{t+1}, \forall c \in \mathcal{C}$ are: (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.

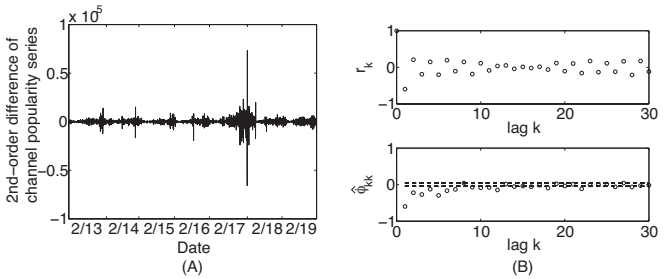

Fig. 4. ARIMA model identification for channel popularity series of CCTV1 in Fig. 3(A)-1.

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.*, [10]). We argue that these models represent 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 high accuracy [11]. 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.

In model identification, to determine the degree of differencing, d, a standard technique is to difference the time series as many times as is needed to produce stationary time series. We have therefore derived $d = 2$ for our time series $n^c_t, t = 1, 2, \ldots$, based on the observations that the 2nd-order difference of the original time series for all the channels is largely stationary. For example, Fig. 4(A) shows the 2nd-order difference of the channel popularity time series of CCTV1, as given in Fig. 3(A)-1, which is stationary with the mean of zero. To identify the values of p and q, the standard technique is to study the general appearance of the estimated autocorrelation and partial autocorrelation functions of the differenced time series ([11], pp. 187). For example, Fig. 4(B)

plots the autocorrelation r_k and partial autocorrelation $\hat{\phi}_{kk}$ function values of the differenced channel popularity series in Fig. 4(A) up to lag $k = 30$. We observe that only r_1 is non-zero and $\hat{\phi}_{kk}$ tails off, which identifies $p = 0$ and $q = 1$ ([11], pp. 186). As we have generally made similar observations regarding channel popularity series for other channels, we derive an ARIMA$(0, 2, 1)$ model to use in our prediction in each streaming channel.

Having identified an ARIMA$(0, 2, 1)$ model, the channel popularity prediction in channel c at time $t + 1$, $\bar{n}^c_{t+1}$ can be expressed as follows:

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

where θ^c is the coefficient for the random error term a^c_t and can be estimated with a least squares algorithm. When we use (3) for prediction in practice, the random error at the future time $t + 1$, *i.e.*, a^c_{t+1}, can be treated as zero, and the random error at time t can be approximated by $a^c_t = n^c_t - \bar{n}^c_t$ [11]. Therefore, the prediction function is simplified to the following. We note that the derived model is desirably simple, with only one coefficient θ^c to be estimated:

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

To dynamically refine the model for an accurate prediction of the popularity of channel c over time, we propose to carry out the forecasting in a dynamic fashion: To start, the ARIMA$(0, 2, 1)$ model for channel c is trained with its channel popularity statistics in the most recent N_1 time steps, and the value of the coefficient θ^c is derived. Then at each subsequent 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. We note that the values of T_1 and T_2 represent a tradeoff between the accuracy of the model and the computational overhead incurred. The empirical values of $T_1 = 8$ and $T_2 = 10$ work well in our experiments to be presented in Sec. V.

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 trace study in Sec II-C, we have observed $q^c \propto (s^c)^{\alpha^c}$ in each specific channel c, where α^c is the exponent of s^c, *e.g.*, $q^c \propto (s^c)^{0.4}$ in Fig. 3(B)-1. [4] 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 different 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.

In order to accurately capture the relationship among the quantities over time, we dynamically re-learn the regression model in (6) for each channel c in the following manner. To start, the designated server trains the regression model for channel c 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.

In summary, we emphasize that the goal for the modeling of the streaming quality function is to accurately capture the relationship among the streaming quality, server capacity usage, and the number of peers in a channel at each time. While one may consider using peer upload bandwidths as variables, we choose to employ the number of peers instead whose values can be much easier collected in practical systems, and to represent the influence of the supply/demand of peer bandwidths on the streaming quality using both the number of peers n^c and the dynamically learned parameters γ^c and β^c in (5). We further 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 should 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 could vary over time, depending on the relationship between the demand for streaming bandwidth at peers and the supply of peer upload bandwidth at different times: on one hand, the streaming quality can be improved with more high-contribution peers (*e.g.*, Ethernet peers) in the channel (the case of $\beta^c > 0$); on the other hand, if more peers join the channel with an average upload bandwidth lower than the required streaming rate, a downgrade of the streaming quality would occur (the case of $\beta^c < 0$). The different cases of α^c and β^c are to be further investigated in our experiments using

[4] A more accurate model for the relation in Fig. 3(B)-1 (derived using our algorithm that follows in the section) is $q^c = 1.15(s^c)^{0.37}$. The goodness of fit tests applied over the converted linear regression model, $\log q^c = \log 1.15 + 0.37 \log s^c$, validated the significance of the coefficients 1.15 and 0.37.

the traces in Sec. V-A2.

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)':

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

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

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

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

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 $\frac{\mathrm{d}G}{\mathrm{d}s^c_{t+1}}$, as long as the upper bound of s^c_{t+1} indicated in (9) has not been reached. The marginal utility, $\frac{\mathrm{d}G}{\mathrm{d}s^c_{t+1}}$, represents how much the streaming quality in a channel c can be enhanced by allocating one unit more server capacity to this channel, which is decided by the priority, the number of peers, and the server capacity already allocated into the channel.

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 incremental water-filling approach is given in Table II. To better illustrate the idea of *water filling*, we utilize the reciprocal of marginal utility $\frac{\mathrm{d}G}{\mathrm{d}s^c_{t+1}}$, *i.e.*, $(\frac{\mathrm{d}G}{\mathrm{d}s^c_{t+1}})^{-1}$, in our algorithm description, and maximally assign server capacity to the channels with the current smallest value of $(\frac{\mathrm{d}G}{\mathrm{d}s^c_{t+1}})^{-1}$, equivalently.

We explain the incremental water-filling approach with a 5-channel example in Fig. 5. In this figure, each channel is represented by one bin. The volume of water in bin c is $(s^c)^{(1-\alpha^c)}$, the width of bin c is $w_c = p^c\gamma^c\alpha^c(n^c_{t+1})^{(1+\beta^c)}$, and thus the water level of the bin represents $(\frac{\mathrm{d}G}{\mathrm{d}s^c})^{-1} = \frac{(s^c)^{(1-\alpha^c)}}{w_c}$ for channel c. As $0 < \alpha^c < 1$, each bin has a maximal height, $\frac{(B^c_{t+1})^{(1-\alpha^c)}}{w_c}$, which is represented by the dashed line in each bin. The incremental water-filling approach starts with the optimal server capacity allocation at the current time t, *i.e.*, $s^c = s^{c*}_t, \forall c \in \mathcal{C}$ (Step 1 in Table II). 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.*, $U - \sum_{c\in\mathcal{C}} s^c > 0$, or the allocated capacity of some channel c exceeds its maximal server capacity requirement for time $t+1$, *i.e.*, $s^c > B^c_{t+1}$ (*e.g.*, the case that the water level goes above the maximal bin height for bin 1 in Fig. 5(A).) If so, the overall surplus is computed (Step 2 in Table II) and allocated to the channels whose maximal server capacity requirement has not been reached, starting from the channel with the current maximum marginal utility $\frac{\mathrm{d}G}{\mathrm{d}s^c}$, *i.e.*, lowest water level (Step 3 in Table II). For the example in Fig. 5, the surplus portion of the water in bin 1 in (A) is reallocated to bin 2 and bin 4, with the results shown in (B). After all the surplus has been allocated, the server capacity assignment is further adjusted among the channels towards a same marginal utility (water level), 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, *i.e.*, the water in the bin with the highest water level (largest value of $(\frac{\mathrm{d}G}{\mathrm{d}s^c})^{-1}$) is flooded into the bin that has the lowest water level (smallest value of $(\frac{\mathrm{d}G}{\mathrm{d}s^c})^{-1}$) and has not reached its maximal bin height yet (Step 4 in Table II). This process repeats until all channels have reached the same marginal utility or have reached their

TABLE II
INCREMENTAL WATER-FILLING APPROACH

1. Initialize
   $s^c \leftarrow s^{c*}_t, \forall c \in \mathcal{C}$.
   $B^c_{t+1} \leftarrow (\gamma^c (n^c_{t+1})^{\beta^c})^{-\frac{1}{\alpha^c}}, \forall c \in \mathcal{C}$.
2. Compute current surplus of server capacity
   surplus $= U - \sum_{c\in\mathcal{C}} s^c$.
   for all $c \in \mathcal{C}$
     if $s^c > B^c_{t+1}$
       surplus = surplus + $(s^c - B^c_{t+1})$
     end if
   end for
3. Allocate surplus to channels
   Compute $(\frac{\mathrm{d}G}{\mathrm{d}s^c})^{-1} = \frac{(s^c)^{1-\alpha^c}}{p^c\gamma^c\alpha^c(n^c_{t+1})^{(1+\beta^c)}}, \forall c \in \mathcal{C}$.
   while surplus > 0 and not all s^c has reached B^c_{t+1}, $\forall c \in \mathcal{C}$
     find the channel c_0 with the smallest value of $(\frac{\mathrm{d}G}{\mathrm{d}s^c})^{-1}$ and $s^{c_0} < B^{c_0}_{t+1}$.
     $s^{c_0} \leftarrow s^{c_0} + \delta$, surplus $\leftarrow$ surplus $- \delta$.
     update $(\frac{\mathrm{d}G}{\mathrm{d}s^{c_0}})^{-1}$.
   end while
4. Adjust channel capacity assignment towards the same marginal utility
   if not all s^c has reached B^c_{t+1}, $\forall c \in \mathcal{C}$
     find channel $c_{\min}$ with the current smallest value of $(\frac{\mathrm{d}G}{\mathrm{d}s^c})^{-1}$ and $s^c < B^c_{t+1}$.
     find channel $c_{\max}$ with the current largest value of $(\frac{\mathrm{d}G}{\mathrm{d}s^c})^{-1}$.
     while $\| (\frac{\mathrm{d}G}{\mathrm{d}s^{c_{\min}}})^{-1} - (\frac{\mathrm{d}G}{\mathrm{d}s^{c_{\max}}})^{-1} \| > \epsilon$
       $s^{c_{\min}} \leftarrow s^{c_{\min}} + \delta$, $s^{c_{\max}} \leftarrow s^{c_{\max}} - \delta$.
       update $(\frac{\mathrm{d}G}{\mathrm{d}s^c})^{-1}$ for channels $c_{\min}$ and $c_{\max}$.
       find channel $c_{\min}$ with the current smallest $(\frac{\mathrm{d}G}{\mathrm{d}s^c})^{-1}$ and $s^c < B^c_{t+1}$.
       find channel $c_{\max}$ with the current largest $(\frac{\mathrm{d}G}{\mathrm{d}s^c})^{-1}$.
     end while
   end if
5. $s^{c*}_{t+1} \leftarrow s^c, \forall c \in \mathcal{C}$.

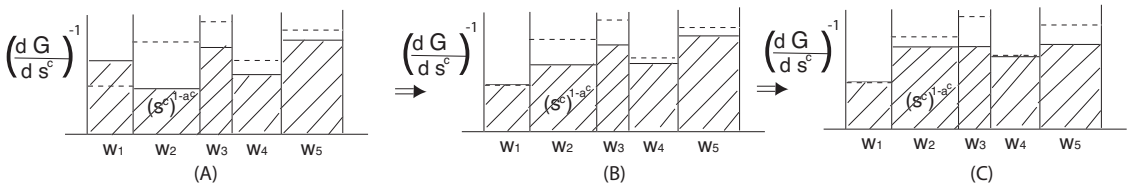

Fig. 5. An illustration of the incremental water-filling approach with 5 channels.

respective maximum server bandwidth requirement. For the example in Fig. 5(B), the water from bin 3 and 5 are filled into bin 2 and 4, until bin 4 reaches its maximal height and bins 2, 3, and 5 achieve the same water level, as shown in Fig. 5(C).

Such an incremental water-filling approach derives the optimal server capacity provisioning for all channels at time $t+1$, as established by the following theorem:

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

We postpone the proof of the theorem to Appendix A.

E. Ration: the complete algorithm

Our complete algorithm is summarized in Table III, 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 periodic heartbeat messages to the server, each of which includes its current buffering level.

We note that in practice, the allocation interval is determined 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.

We further remark on the computational complexity of the algorithm in Table III. The main computation for steps 2 and 3 lies in the training of ARIMA or the regression model, with least squares algorithms at $O(N^3)$ where N is the size of the training sample set. As both steps are carried out for each of the M channels, their complexity are at most $O(M{N_1}^3)$ and $O(M{N_2}^3)$, respectively. We generally need no more than $30-50$ samples to train an ARIMA$(0,2,1)$ model, *i.e.*, $N_1 < 50$, and even fewer to derive the regression model [11], [12] (thus only a small amount of historical data needs to be maintained at the server for the execution of *Ration*). Further considering that the training is only carried out when necessary (*i.e.*, when the accuracy of the models has fallen), we conclude that the two steps incur low computational overhead in reality. At step 4, we have designed the incremental water-filling approach, which only involves local adjustments for channels that are affected. In summary, the algorithm involves low computational overhead, and can be carried out in a completely online fashion to adapt to the dynamics of P2P systems.

TABLE III
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 buffering level 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 for channel c with the most recent N_1 peer number statistics using least squares algorithm.
Predict the channel popularity at time $t+1$ by $\bar{n}_{t+1}^c = 2n_t^c - n_{t-1}^c - \theta^c(n_t^c - \bar{n}_t^c)$, where θ^c is the parameter in ARIMA(0,2,1).
$\rightarrow$ Channel popularity predictions for all channels derived.

3. Learning 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 outside 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, using the least squares algorithm.
$\rightarrow$ Current streaming quality functions for all channels 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 Table II.
$\rightarrow$ Optimal server capacity provisioning, $s_{t+1}^{c*}, \forall c \in \mathcal{C}$, derived.

IV. PRACTICAL IMPLICATIONS OF *Ration*

We now discuss the practical application of *Ration* in real-world P2P live streaming systems. In practical systems with unknown demand for server capacity in each ISP, *Ration* is able to fully utilize 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.

A. Service differentiation across channels in tight supply-demand relations

In most typical scenarios, 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. In this case, *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 the marginal utility of each channel, $\frac{\mathrm{d}G}{\mathrm{d}s^c} = p^c n^c \frac{\mathrm{d}q^c}{\mathrm{d}s^c}$, as determined by the priority and popularity of the channel, and the marginal improvement of its streaming quality upon a unit increase of server capacity. Given the

limited server capacity deployed in the ISP, the streaming solution provider can differentiate the streaming quality of the channels by manipulating the priority levels assigned to the channels. The following theorem states the proportional service differentiation across channels provided by *Ration*.

Theorem 2. *The optimal server capacity provisioning in* Ration *in (1) provides the following streaming quality differentiation between any two channels c and $\bar{c}$, which do not achieve the best streaming quality of* 1 *in the allocation:*

$$\frac{(q_{t+1}^c)^{\frac{1-\alpha^c}{\alpha^c}}}{(q_{t+1}^{\bar{c}})^{\frac{1-\alpha^{\bar{c}}}{\alpha^{\bar{c}}}}} = \frac{p^c\alpha^c(\gamma^c)^{\frac{1}{\alpha^c}}(n_{t+1}^c)^{\frac{\alpha^c+\beta^c}{\alpha^c}}}{p^{\bar{c}}\alpha^{\bar{c}}(\gamma^{\bar{c}})^{\frac{1}{\alpha^{\bar{c}}}}(n_{t+1}^{\bar{c}})^{\frac{\alpha^{\bar{c}}+\beta^{\bar{c}}}{\alpha^{\bar{c}}}}},$$
$$\forall c, \bar{c} \in \mathcal{C} \text{ where } q_{t+1}^c < 1 \text{ and } q_{t+1}^{\bar{c}} < 1. \quad (11)$$

We postpone the proof to Appendix B. Priority levels of different channels can be set according to Eq. (11) to achieve desired relative streaming quality levels across the channels. More specifically, if $\alpha^c \approx \alpha^{\bar{c}} \approx \alpha$ (we have observed similar α^c values for many different channels in our trace studies), we have the following approximation:

$$\frac{q_{t+1}^c}{q_{t+1}^{\bar{c}}} = \frac{(p^c)^{\frac{\alpha}{1-\alpha}}(\gamma^c)^{\frac{1}{1-\alpha}}(n_{t+1}^c)^{\frac{\alpha+\beta^c}{1-\alpha}}}{(p^{\bar{c}})^{\frac{\alpha}{1-\alpha}}(\gamma^{\bar{c}})^{\frac{1}{1-\alpha}}(n_{t+1}^{\bar{c}})^{\frac{\alpha+\beta^{\bar{c}}}{1-\alpha}}}.$$

In this case, if the streaming solution provider wishes to achieve a better streaming quality in channel c than that in channel $\bar{c}$, *i.e.*, $q_{t+1}^c > q_{t+1}^{\bar{c}}$, they may set the priorities for the channels, p^c and $p^{\bar{c}}$, to satisfy the following conditions: $\frac{p^c}{p^{\bar{c}}} > (\frac{\gamma^{\bar{c}}}{\gamma^c})^{\frac{1}{\alpha}} \frac{(n_{t+1}^{\bar{c}})^{\frac{\alpha+\beta^{\bar{c}}}{\alpha}}}{(n_{t+1}^c)^{\frac{\alpha+\beta^c}{\alpha}}}$. A similar priority assignment method can be applied to determine the priority levels of multiple channels in the system.

B. Total server capacity adjustment in all supply-demand relations

Ration is not only useful in optimal utilization of deployed server capacity; the channel popularity prediction and streaming function learning modules in *Ration* can further be used to determine the minimum overall amount of server capacity to be deployed in each ISP, guaranteeing desired levels of streaming qualities in all the channels.

If the P2P streaming system is operating at the tight supply-demand mode in an ISP and if the streaming solution provider wishes to boost the streaming quality of all the channels to a best value around 1, they may compute how much more server capacity to be added, by comparing the current provisioning with the overall required server capacity to achieve $q_{t+1}^c = 1, \forall c \in \mathcal{C}$. The latter amount is $\sum_{c\in\mathcal{C}} B_{t+1}^c$, where B_{t+1}^c is the maximum server capacity needed for channel c as in (9) in **Provision(t+1)'**, derived using the streaming quality function in (5) by setting $q_{t+1}^c = 1$.

Similarly, if the system is operating with extremely tight supply-demand relations, *e.g.*, the flash crowd scenario, most server capacity is assigned to 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. The channel popularity prediction in *Ration* can be used to detect such a flash crowd, and to trigger the deployment of backup server resources. The extra amount to add can be computed with the current streaming quality function derived for the respective channels, according to the targeted streaming quality.

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

C. Channel deployment in each ISP

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.

D. Server capacity implementation

Finally, we note that the server capacity provisioning in each ISP can be implemented in practice with a number of streaming servers deployed, with the number decided by the total capacity to be provisioned and the upload capacity of each server. Inside each ISP, the servers can be further deployed in different geographical areas, and the derived server capacity provisioning for each channel can be distributed among several servers as well, in order to achieve load balancing and streaming delay minimization.

V. 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++ over a P2P emulation platform on a high-performance cluster of 50 Dell 1425SC and Sun v20z dual-CPU servers, deployed in the networking research laboratory at the University of Toronto [13]. On this platform, we are able to emulate hundreds of concurrent peers on each server, and emulate all network parameters, such as node capacities, link bandwidth bottlenecks, messaging delays, etc. 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,

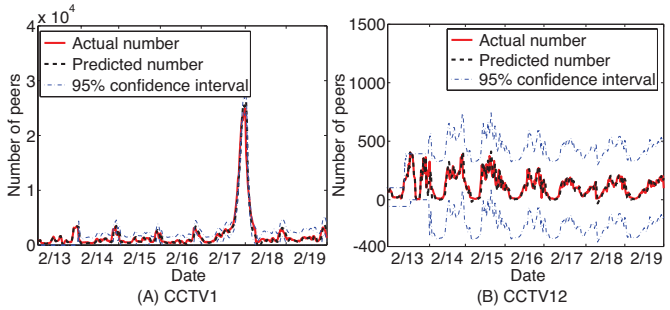

Fig. 6. Prediction of the number of peers with ARIMA$(0, 2, 1)$.

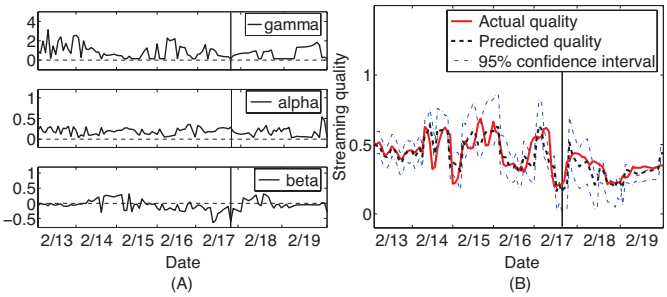

Fig. 7. Dynamic learning of the streaming quality function for CCTV1.

we deploy one server for each ISP, implementing both the tracker 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 employs. 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 the playback buffer on the peers is set to 30 seconds. Each peer reports its buffering level to the server in its ISP every 20 seconds with heartbeat messages, and the server processes them and adjusts the capacity allocation every 60 seconds. We note that this represents much expedited settings, as these intervals can be much longer in real-world systems. With this setting, we are not only able to accelerate our experiments that emulate real-world streaming over a long period of time, but also to demonstrate the speed and efficiency of our algorithm even when it is executed in very short time scales.

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[5]. 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.

1) Prediction of the number of peers: Fig. 6 presents the results of prediction with ARIMA$(0, 2, 1)$ for the popular channel CCTV1 and the less popular 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.

2) Dynamic streaming quality function: Fig. 7(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.

Fig. 7(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.

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.

Fig. 8(A) and (B) show the server capacity allocated to each of the 5 channels, and their actually achieved streaming quality at different times. 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*, $\frac{\mathrm{d}G}{\mathrm{d}s^c} = p^c n^c \frac{\mathrm{d}q^c}{\mathrm{d}s^c} = \frac{p^c \gamma^c \alpha^c (n^c)^{(1+\beta^c)}}{(s^c)^{1-\alpha^c}}$. As $\beta^c > -1$ is observed in our previous experiment, the marginal utility is positively

[5]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.

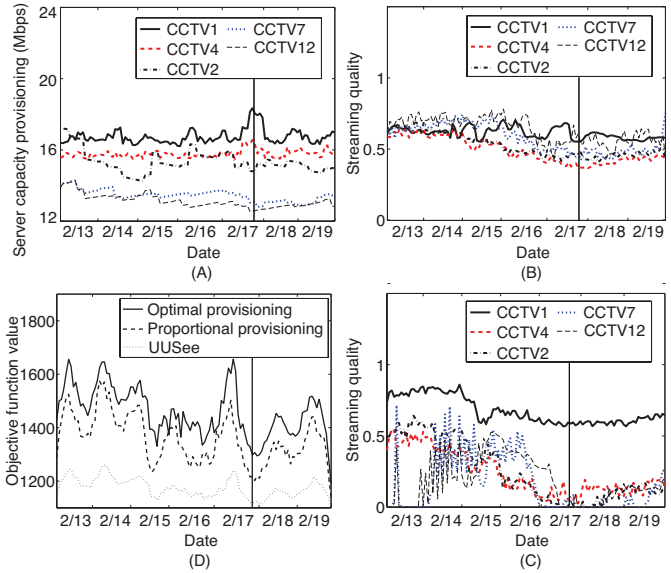

Fig. 8. Server capacity provisioning for 5 non-prioritized channels: (A) Server capacity provisioning with *Ration*, (B) Streaming quality achieved with *Ration*, (C) Streaming quality achieved with proportional allocation, (D) Comparison of objective function values.

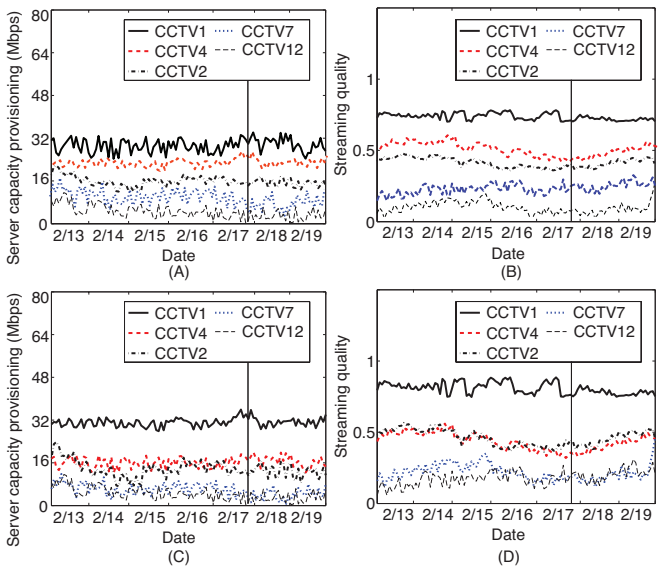

Fig. 9. Server capacity provisioning for 5 prioritized channels with *Ration*: (A)(B) 5 streaming quality levels; (C)(D) 3 streaming quality levels.

correlated with the number of peers, and thus the more popular channel is assigned more server capacity.

On the other hand, in Fig. 8(B), we do not observe evident correlation between the channel popularity and its achieved streaming quality, as the latter is decided by both 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 implements the same protocol as employed in UUSee, except its server capacity allocation, which goes as follows: At each time t, the server capacity is proportionally allocated to the channels, based only on their predicted number of peers for time $t+1$. Fig. 8(C) shows that the most popular channel, CCTV1, achieves better streaming quality with this proportional allocation as compared to that in Fig. 8(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. 8(B) and (C), we compute the values of the objective function of **Provision(t+1)** in (1), and plot them in Fig. 8(D). Given the 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 buffering level 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*.

In our experiments, we have also compared *Ration* with the original UUSee protocol, in which no allocation across channels is done at all. We have observed more unstable and lower streaming qualities in all the channels than those shown in Fig. 8(B) and (C), and lower objective function values than those achieved by the proportional provisioning approach, as shown in Fig. 8(D).

4) Effectiveness of channel prioritization: In the next experiment, we study the effect of channel prioritization on server capacity provisioning with *Ration*. We investigate two cases: (1) We wish to achieve differentiated streaming qualities across all 5 channels, with $q^{cctv1} > q^{cctv4} > q^{cctv2} > q^{cctv7} > q^{cctv12}$; (2) We wish to achieve three streaming quality levels, with $q^{cctv1} > q^{cctv4} \approx q^{cctv2} > q^{cctv7} \approx q^{cctv12}$. We set the following channel priority levels in the two cases, respectively: (1) $p^{cctv1} = 500, p^{cctv4} = 250, p^{cctv2} = 200, p^{cctv7} = 75, p^{cctv12} = 100$; (2) $p^{cctv1} = 500, p^{cctv4} = 220, p^{cctv2} = 200, p^{cctv7} = 75, p^{cctv12} = 110$. These priority values are derived based on Eqn. (11) in Sec. IV to guarantee the targeted streaming quality differentiation, using the popularity, γ^c, α^c and β^c derived for respective channels. The experimental results are plotted in Fig. 9.

Comparing Fig. 9(A) and (C) to Fig. 8(A), we observe further differentiated server capacities among the channels, where the channels with higher priority and popularity are allocated more capacity. In Fig. 9(B) and (D), we observe differentiated streaming quality levels across the channels, which meet our expectations: we can observe 5 different streaming quality levels in Fig. 9(B) and 3 in Fig. 9(D). 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

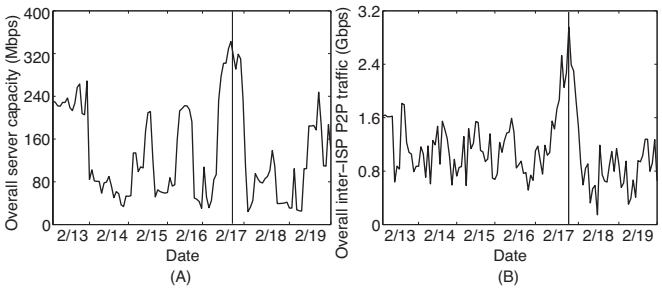

Fig. 10. P2P live streaming for 5 channels in 4 ISPs: without ISP awareness.

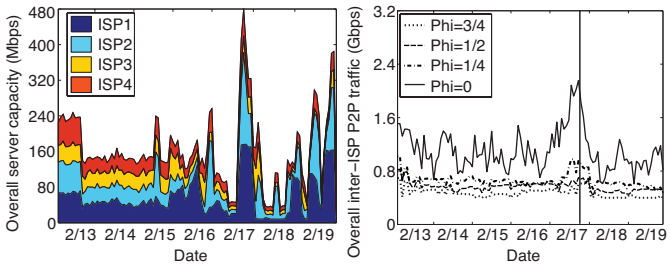

Fig. 11. P2P live streaming for 5 channels in 4 ISPs: with full ISP awareness.

Fig. 12. Server capacity provisioning vs. inter-ISP traffic: a tradeoff.

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, B^c_{t+1}, 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.

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, *i.e.*, each peer is assigned partners that can be any other peers in the entire network. The overall server capacity U_B used on the server over time is shown in Fig. 10(A), and the total inter-ISP P2P traffic in the system is plotted in Fig. 10(B).

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. There is no inter-ISP traffic in this case. The server capacity used on the server in each ISP is illustrated with the area plot in Fig. 11. Comparing Fig. 11 to Fig. 10(A), we can see that more overall server capacity is needed in the system when the traffic is completely restricted inside each ISP with per-ISP server capacity deployment, as peers now have fewer choices of supplying neighbors and may have to resort more to the server in their respective ISPs. However, the increase in the total server capacity usage is non-significant. The difference is only larger during the flash crowd, when it becomes very difficult for peers to identify enough supplying peers with available bandwidth inside the ISP.

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. 10(A) be $U_{B\min}$ and that shown in Fig. 11 be $U_{B\max}$. 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 $U_{B\min} + \phi(U_{B\max} - U_{B\min})$ 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. 12 show an increase of inter-ISP traffic with the decrease of server capacity provisioning. Further comparing the $\phi = 0$ case in Fig. 12 to Fig. 10(B), we observe that while the total server capacity is the same $U_{B\min}$ in both cases, a smaller amount of inter-ISP P2P traffic is involved with the ISP-aware peer selection than without any ISP awareness.

VI. Related Work

With the successful Internet deployment of mesh-based P2P live streaming systems [14], [2], [3], [4], significant research efforts have been devoted to their measurements and improvements. With respect to measurements, existing studies [15], [16], [17], [18], [19], [20] 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 seminal work of Coolstreaming [14], various improvements of peer strategies in such mesh-based P2P live streaming have been proposed, *e.g.*, the enhancement of the block pulling mechanism [21], the optimization of peer connectivity for content swarming [22], and the exploration of inter-overlay cooperation [23]. 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. A preliminary report of this work appeared in INFOCOM 2008 [24]. This paper represents a substantial revision and extension, with solid studies of the UUSee server capacity utilization over a longer trace period and complete discussions on the design, analysis and application of the online server capacity provisioning algorithm.

With respect to analytical work related to the subject of server capacity, Das *et al.* [25] 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.* [10] modeled streaming quality in a mesh-based P2P streaming system in terms of both server and peer upload capacities. As compared to these studies, our work focuses entirely on the *practicality* of a dynamic server capacity provisioning mechanism. Other than using simplified modeling assumptions such as Poisson arrivals, 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 on the large amount of inter-ISP traffic brought by P2P applications, with respect to BitTorrent file sharing [26], [27], P2P Video on Demand [5], and P2P software update distribution [28]. Approaches for the localization of P2P traffic inside ISP boundaries have been proposed, which mostly focus on ISP-aware peer selection strategies[27], [29], [30]. 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 at the server side on a per-ISP basis to maximally guarantee the success of ISP-aware P2P streaming.

VII. 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 10 months worth 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 expectations to effectively provision server capacities according to the demand and channel priority over time.

Appendix A
Proof of Theorem 1

Proof: Let $s_{t+1}^{c*}, \forall c \in \mathcal{C}$, be an optimal solution to the optimization problem in (7). Introducing Lagrangian multiplier λ for the constraint in (8), $\nu = (\nu^c, \forall c \in \mathcal{C})$ for the constraints in (9) and $\mu = (\mu^c, \forall c \in \mathcal{C})$ for the constraints in (10), we obtain the KKT conditions for the problem as follows (pp. 244, [31]):

$$\sum_{c\in\mathcal{C}} s_{t+1}^{c*} \le U, \tag{12}$$

$$\lambda^* \ge 0, \tag{13}$$

$$0 \le s_{t+1}^{c*} \le B_{t+1}^c, \nu^{c*} \ge 0, \mu^{c*} \ge 0, \forall c \in \mathcal{C}, \tag{14}$$

$$\lambda^*(\sum_{c\in\mathcal{C}} s_{t+1}^{c*} - U) = 0, \tag{15}$$

$$\mu^{c*} s_{t+1}^{c*} = 0, \forall c \in \mathcal{C}, \tag{16}$$

$$\nu^{c*}(s_{t+1}^{c*} - B_{t+1}^c) = 0, , \forall c \in \mathcal{C}, \tag{17}$$

$$-\frac{\mathrm{d}G}{\mathrm{d}s_{t+1}^c} + \lambda^* - \mu^{c*} + \nu^{c*} = 0, \forall c \in \mathcal{C}. \tag{18}$$

For $s_{t+1}^{c*} > 0$, we have $\mu^{c*} = 0$ from (16), and then $\frac{\mathrm{d}G}{\mathrm{d}s_{t+1}^c} = \lambda^* + \nu^{c*}$ from (18). Therefore, if further we have $s_{t+1}^{c*} < B_{t+1}^c$, we derive $\nu^{c*} = 0$ from (17) and then $\frac{\mathrm{d}G}{\mathrm{d}s_{t+1}^c} = \lambda^*$. As $\frac{\mathrm{d}G}{\mathrm{d}s_{t+1}^c} = \frac{p^c\gamma^c\alpha^c(n_{t+1}^c)^{(1+\beta^c)}}{(s_{t+1}^c)^{1-\alpha^c}}$, $\gamma^c > 0$ and $0 < \alpha^c < 1$, we can derive, $\forall c \in \mathcal{C}$,

$$s_{t+1}^{c*} = \begin{cases} B_{t+1}^c & \text{if } \frac{1}{\lambda^*} \ge \frac{(B_{t+1}^c)^{(1-\alpha^c)}}{p^c\gamma^c\alpha^c(n_{t+1}^c)^{(1+\beta^c)}}, \\ (\frac{p^c\gamma^c\alpha^c(n_{t+1}^c)^{1+\beta^c}}{\lambda^*})^{\frac{1}{1-\alpha^c}} & \text{if } \frac{1}{\lambda^*} < \frac{(B_{t+1}^c)^{(1-\alpha^c)}}{p^c\gamma^c\alpha^c(n_{t+1}^c)^{(1+\beta^c)}}. \end{cases} \tag{19}$$

Notice that for all the channels with $0 < s_{t+1}^{c*} < B_{t+1}^c$, we have $\frac{1}{\lambda^*} = (\frac{\mathrm{d}G}{\mathrm{d}s_{t+1}^c})^{-1}$, which is the final water level for those bins whose maximal heights are not achieved, as illustrated in Fig. 5(C)[6]. We also notice that $\frac{(B_{t+1}^c)^{(1-\alpha^c)}}{p^c\gamma^c\alpha^c(n_{t+1}^c)^{(1+\beta^c)}}$ is the maximal height for bin c. Therefore, the key to derive s_{t+1}^{c*} is to derive the optimal water level $\frac{1}{\lambda^*}$. If a bin's maximal height is below the optimal water level, the optimal server capacity share for the corresponding channel is its maximal server capacity requirement, *i.e.*, $s_{t+1}^{c*} = B_{t+1}^c$; otherwise, its allocated server capacity is what achieves $\frac{(s_{t+1}^{c*})^{(1-\alpha^c)}}{p^c\gamma^c\alpha^c(n_{t+1}^c)^{(1+\beta^c)}} = \frac{1}{\lambda^*}$.

To derive the optimal water level, from the starting water levels in the bins decided by the server capacity assignment at time t, we first make sure the overall server capacity at the amount of U is maximally filled into the bins, while no bin's maximal height is exceeded. Then, we decrease the high water levels by decreasing the server capacity assigned to the corresponding bins (as $\alpha^c < 1$, $(\frac{\mathrm{d}G}{\mathrm{d}s_{t+1}^c})^{-1}$ decreases with the decrease of s_{t+1}^c), and increase the low water levels by increasing the server capacity assigned to the corresponding bins, while guaranteeing the maximal height of each bin is never exceeded. When all bins reach the same water level, except those whose maximal heights have been reached, we have derived the optimal server capacity allocation for all channels for time $t+1$, as given in (19). □

Appendix B
Proof of Theorem 2

Proof: Both the incremental water-filling algorithm in Table II and the proof of theorem 1 in Appendix I show that at optimality, the marginal utility for any channels, that are not allocated their maximum server capacity requirement B_{t+1}^c (*i.e.*, $\forall c, q_{t+1}^c < 1$), is the same, *i.e.*, $\frac{\mathrm{d}G}{\mathrm{d}s_{t+1}^c} = \frac{\mathrm{d}G}{\mathrm{d}s_{t+1}^{\bar{c}}}, \forall c, \bar{c} \in \mathcal{C}$, where $q_{t+1}^c < 1$ and $q_{t+1}^{\bar{c}} < 1$.

Since $\frac{\mathrm{d}G}{\mathrm{d}s_{t+1}^c} = \frac{p^c\gamma^c\alpha^c(n_{t+1}^c)^{(1+\beta^c)}}{(s_{t+1}^c)^{1-\alpha^c}}$ and $\frac{\mathrm{d}G}{\mathrm{d}s_{t+1}^{\bar{c}}} = \frac{p^{\bar{c}}\gamma^{\bar{c}}\alpha^{\bar{c}}(n_{t+1}^{\bar{c}})^{(1+\beta^{\bar{c}})}}{(s_{t+1}^{\bar{c}})^{1-\alpha^{\bar{c}}}}$, we have

$$\frac{p^c\gamma^c\alpha^c(n_{t+1}^c)^{(1+\beta^c)}}{(s_{t+1}^c)^{1-\alpha^c}} = \frac{p^{\bar{c}}\gamma^{\bar{c}}\alpha^{\bar{c}}(n_{t+1}^{\bar{c}})^{(1+\beta^{\bar{c}})}}{(s_{t+1}^{\bar{c}})^{1-\alpha^{\bar{c}}}}. \tag{19}$$

[6]Note that $s_{t+1}^{c*} = 0$ only occurs at very special cases, such as $n_{t+1}^c \to 0$ or $\alpha^c \to 0$. In this case, the width of the bin corresponding to channel c is zero, and thus no water (bandwidth) will be assigned to the bin. We omit this special case in our discussions.

From $q^c_{t+1} = \gamma^c (s^c_{t+1})^{\alpha^c} (n^c_{t+1})^{\beta^c}$, we derive $s^c_{t+1} = (\frac{q^c_{t+1}}{\gamma^c (n^c_{t+1})^{\beta^c}})^{\frac{1}{\alpha^c}}$. Substitute s^c_{t+1} with this formula in Eq. (19), we can obtain

$$\frac{p^c \gamma^c \alpha^c (n^c_{t+1})^{(1+\beta^c)}}{(\frac{q^c_{t+1}}{\gamma^c (n^c_{t+1})^{\beta^c}})^{\frac{1-\alpha^c}{\alpha^c}}} = \frac{p^{\bar{c}} \gamma^{\bar{c}} \alpha^{\bar{c}} (n^{\bar{c}}_{t+1})^{(1+\beta^{\bar{c}})}}{(\frac{q^{\bar{c}}_{t+1}}{\gamma^{\bar{c}} (n^{\bar{c}}_{t+1})^{\beta^{\bar{c}}}})^{\frac{1-\alpha^{\bar{c}}}{\alpha^{\bar{c}}}}}. \quad (20)$$

A transformation of (20) will give Eq. (11). □

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] *PPLive*, http://www.pplive.com/.

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

[4] *ppStream*, http://www.ppstream.com/.

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

[6] C. Wu, B. Li, and S. Zhao, "Exploring Large-Scale Peer-to-Peer Live Streaming Topologies," *ACM Transactions on Multimedia Computing, Communications and Applications*, vol. 4, no. 3, August 2008.

[7] ——, "Characterizing Peer-to-Peer Streaming Flows," *IEEE Journal on Selected Areas in Communications*, vol. 25, no. 9, pp. 1612–1626, December 2007.

[8] Cymru Whois service, http://www.cymru.com/BGP/asnlookup.html.

[9] Cernet BGP View, http://bgpview.6test.edu.cn/bgpview/curana/ipv4cn /chinaasnlist.shtml.

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

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

[12] D. C. Montgomery, e. A. Peck, and G. G. Vining, *Introduction to Linear Regression Analysis, Third Edition*. John Wiley and Sons, Inc., 2001.

[13] M. Wang, H. Shojania, and B. Li, "Crystal: An Emulation Framework for Practical Peer-to-Peer Multimedia Streaming Systems," in *Proc. of the 28th International Conference on Distributed Computing Systems (ICDCS 2008)*, June 2008.

[14] 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.

[15] 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.

[16] X. Hei, Y. Liu, and K. W. Ross, "Inferring Network-Wide Quality in P2P Live Streaming Systems," *IEEE Journal on Selected Areas in Communications, Special Issue on Advances in Peer-to-Peer Streaming Systems*, vol. 25, no. 9, pp. 1640–1654, December 2007.

[17] 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.

[18] 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.

[19] B. Li, S. Xie, G. Y. Keung, J. Liu, I. Stoica, H. Zhang, and X. Zhang, "An Empirical Study of the CoolStreaming+ System," *IEEE Journal on Selected Areas in Communications, Special Issue on Advances in Peer-to-Peer Streaming Systems*, vol. 25, no. 9, pp. 1627–1639, December 2007.

[20] B. Li, Y. Qu, Y. Keung, S. Xie, C. Lin, J. Liu, and X. Zhang, "Inside the New Coolstreaming: Principles, Measurements and Performance Implications," in *Proc. of IEEE INFOCOM*, April 2008.

[21] 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.

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

[23] 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.

[24] C. Wu, B. Li, and S. Zhao, "Multi-channel Live P2P Streaming: Refocusing on Servers," in *Proc. of IEEE INFOCOM 2008*, April 2008.

[25] 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.

[26] 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.

[27] 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.

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

[29] V. Aggarwal, A. Feldmann, and C. Scheideler, "Can ISPs and P2P systems cooperate for improved performance?" in *Proc. of ACM CCR*, July 2007.

[30] H. Xie, Y. R. Yang, A. Krishnamurthy, Y. Liu, and A. Silberschatz, "P4P: Provider Portal for Applications," in *Proc. of ACM SIGCOMM*, August 2008.

[31] S. Boyd, *Convex Optimization*. Cambridge University Press, 2004.

Chuan Wu. Chuan Wu received her B.Engr. and M.Engr. degrees in 2000 and 2002 from Department of Computer Science and Technology, Tsinghua University, China, and her Ph.D. degree in 2008 from the Department of Electrical and Computer Engineering, University of Toronto, Canada. She is currently an assistant professor in the Department of Computer Science, the University of Hong Kong, China. Her research interests include measurement, modeling, and optimization of large-scale peer-to-peer systems and online/mobile social networks. She is a member of IEEE and ACM.

Baochun Li. Baochun Li received the B.Engr. degree from the Department of Computer Science and Technology, Tsinghua University, China, in 1995 and the M.S. and Ph.D. degrees from the Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, in 1997 and 2000. Since 2000, he has been with the Department of Electrical and Computer Engineering at the University of Toronto, where he is currently a Professor. He holds the Nortel Networks Junior Chair in Network Architecture and Services from October 2003 to June 2005, and the Bell University Laboratories Endowed Chair in Computer Engineering since August 2005. His research interests include large-scale multimedia systems, cloud computing, peer-to-peer networks, applications of network coding, and wireless networks. Dr. Li was the recipient of the IEEE Communications Society Leonard G. Abraham Award in the Field of Communications Systems in 2000. In 2009, he was a recipient of the Multimedia Communications Best Paper Award from the IEEE Communications Society, and a recipient of the University of Toronto McLean Award. He is a member of ACM and a senior member of IEEE.

Shuqiao Zhao. Shuqiao Zhao received his B.Engr. degree from Department of Computer Science, Shandong University, China, in 2001. He is currently the managing director in the Multimedia Development group in UUSee Inc., Beijing, China, where he is in charge of the development of the core techniques of UUSee's peer-to-peer live streaming solution.