On ISP-Friendly Rate Allocation for Peer-Assisted VoD

Jiajun Wang
Univ. of California, Berkeley
Berkeley, CA, U.S.A.
junewang@berkeley.com

Cheng Huang
Microsoft Research
Redmond, WA, U.S.A.
chengh@microsoft.com

Jin Li
Microsoft Research
Redmond, WA, U.S.A.
jinl@microsoft.com

ABSTRACT

Peer-to-peer (P2P) content distribution is able to greatly reduce dependence on infrastructure servers and scale up to the demand of the Internet video era. However, the rapid growth of P2P applications has also created immense burden on service providers by generating significant ISP-unfriendly traffic, such as cross-ISP and inter-POP traffic. In this work, we consider the unique properties of peer-assisted Video-on-Demand (VoD) and design a distributed rate allocation algorithm, which can significantly cut down on ISP-unfriendly traffic without much impact on server load. Through extensive packet-level simulation with both synthetic and real-world traces, we show that the rate allocation algorithm can achieve substantial additional gain, on top of previously proposed schemes advocating ISP-friendly topologies.

Categories and Subject Descriptors: C.2.4 [Distributed Systems]: Distributed applications

General Terms: Algorithms, performance

Keywords: Rate allocation, ISP-friendly, peer-to-peer, Video-on-Demand

1. INTRODUCTION

Internet video is poised to be the next big thing. The revenue from Internet video is predicted to grow from $0.9 billion in 2006 to $4.2 billion by 2011, an annual growth rate of 36% [1]. As consumers spend more and more time watching videos online, they are becoming increasingly unsatisfied with being restrained to their computers. Instead, the demand to get Internet video into living room is record-high. This demand fuels the increasing popularity of services (ABC's full-episode streaming, Netflix on-demand, etc.), as well as devices with such capabilities (Microsoft Media Center , Apple TV, TiVo, etc.). Once in the living room, consumers quickly realize that they prefer to go beyond YouTube's limited quality and enjoy SD or even HD video. However, providing high quality Internet video with a traditional client-server model is very costly. In addition, the ever mounting demand is adding significant pressure to existing server-based infrastructures, such as data centers and content distribution networks (CDNs), which are already under heavy burden living up to their current load. As a result, high-profile failures are not uncommon, such as MSNBC's democratic presidential debate webcast mess and the Operah web show crash, not to mention that Internet itself is predicted to melt down if online video becomes mainstream [2].

Fortunately, on the heel of such crisis, comes the help of peer-to-peer (P2P) technology. Indeed, Internet video streaming (both on-demand and live broadcast) using various peer-to-peer or peer-assisted frameworks has been shown to greatly reduce the dependence on infrastructure servers, as well as bypass bottlenecks between content providers and consumers. However, it has also fundamentally altered the relationship among content owners, service providers (ISPs) and consumers. ISPs, in particular, on one hand are spending billions to maintain and upgrade their networks in order to support the ever increasing traffic due largely to P2P. On the other hand, they are also being marginalized by content owners' direct reaching to consumers. As a result, unhappy ISPs start to put up various hurdles for P2P applications by throttling P2P traffic or even taking active measures to deter P2P traffic. As an example, Comcast has recently been exposed to have employed a method that stops BitTorrent traffic by sending reset packets. Practices as such often create huge backlash once they are discovered and made public (even FCC intervened in this incident). ISPs, having learned the lesson in a hard way, now realize that it is in their best interest to work collaboratively with content providers and consumers. For instance, Comcast has announced that it will work closely with BitTorrent. Verizon has also teamed up with Pando Networks to conduct field trials together.

However, to fundamentally incentivize ISPs to embrace P2P, any solution has to address two key aspects: 1) ISPs need to get their share from the booming of Internet video; and 2) P2P applications need to become *ISP-friendly* at the protocol level. While the former aspect is more of a policy or business issue, the latter one poses a concrete technical challenge. A number of recent studies [3, 4, 11, 16, 17] have proposed various schemes at the protocol level to make P2P applications ISP-friendly. The gist of all these proposals is to build P2P overlay topologies in an ISP-friendly fashion, i.e., instead of connecting to neighbors randomly, peers make biased neighbor selection favoring other peers in the same ISP, AS, POP location, and even subnet. Biased neighbor selection has been shown to be effective in reducing ISP-unfriendly traffic. However, neighbor selection is only

MM'08, October 26–31, 2008, Vancouver, British Columbia, Canada.

carried out at a very coarse scale, typically when a peer first joins and when the number of neighbors falls below a threshold, whereas the ideal topology changes constantly as peers join and leave the system. As a result, it is impossible to have an optimal topology at all times. Thus, we propose to study a finer granularity rate allocation problem, in which peers bias transmission rates to their neighbors and dynamically adjust these rates. Our focus in this paper is to design an ISP-friendly rate allocation solution for peer-assisted VoD to compliment biased neighbor selection. Specifically, we make the following contributions.

- We propose and formulate an optimization problem for rate allocation, which unifies the objectives of all three parties: guaranteed QoS for consumers, reduced server load for content owners, and reduced ISP-unfriendly traffic. We derive a *distributed* solution which can be executed by each P2P client independently, while collectively achieving the desired global optimum. (Sec. 3)
- We translate the fluid-level rate allocation scheme into an implementable *packet-level* scheduling algorithm that conforms nicely to the fluid-level rate allocation. (Sec. 4)
- We develop a highly efficient packet-level simulation platform capable of simulating real-world traces at the scale up to 10K concurrent peers. Using this platform, we evaluate the proposed rate allocation solution using both synthetic and real-world traces (collected from a large-scale Internet video service – MSN Video). We confirm the effectiveness of ISP-friendly topology building from earlier studies. More importantly, we quantitatively show that rate allocation can achieve *substantial* additional gain on top of ISP-friendly topologies. (Sec. 5)

2. SYSTEM DESCRIPTION

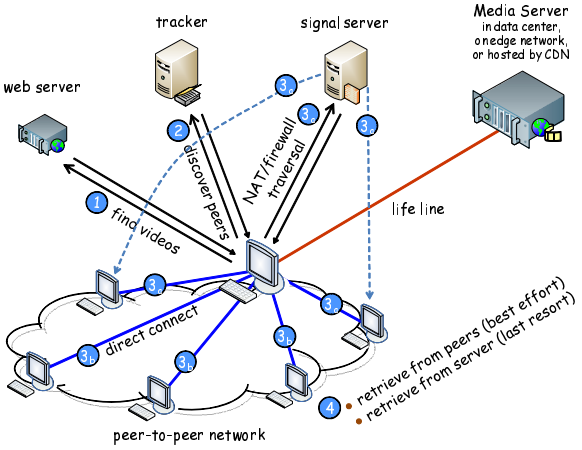


Figure 1: MESH System Architecture.

The proposed rate allocation algorithm is designed for the MESH platform [5] developed at Microsoft. MESH is a peer-assisted distribution platform and supports both video-on-demand, as well as bulk data dissemination. We briefly describe the flow of the peer-assisted VoD scenario. After introducing the proposed algorithm, we will explain where the rate allocation algorithm naturally fits in.

The MESH system architecture is shown in Fig. 1. Similar to many other P2P applications today, a client discovers videos via a web interface to the content library (step 1). Special URLs with embedded information redirect the client to a directory service (or tracker). The client downloads video metadata from the tracker and also obtains a list of peers, who have started watching the same video earlier and are available for sharing (step 2). The client then establishes direct connections with these peers whenever possible (step 3b) or gets help from signal servers if NAT-traversal is required (step 3a). The client retrieves content from the peers in a best effort fashion, which often cannot provide sufficient QoS (e.g., continuous high bitrate for high-def and smooth video playback). Hence, the client actively monitors its QoS and adaptively retrieves content from media servers as needed ("life line" in step 4). Such media servers are located in data centers, on edge networks or hosted by content distribution networks.

In the current system, the tracker generates the peer-list based on certain criteria. Roughly speaking, peers within the same AS or ASes with peering relationships with the client's origin AS are favored. This criterion helps to build an ISP-friendly topology, which we will examine in detail later. Additionally, peers that started in closer time stamps (compared to the client's own start time) are favored. This is based on an early discovery [6] that matching peers in such a matter increases upload efficiency.

3. PROBLEM FORMULATION

In a P2P system, be it streaming or file sharing, each peer connects to multiple other peers (neighbors) simultaneously. Compared to download capacity, peers' upload capacity is limited and oftentimes the most prominent constraint of such systems as peers are often connected via DSL and cable modem (even fiber optic service hosts have very asymmetric access). When multiple connections contend for the limited upload capacity, the natural contention will result in (say via TCP congestion control) an implicit rate allocation, in which the upload capacity is evenly divided among all the connections as empirically observed in [4]. Let's name this implicit scheme *even bandwidth allocation*. However, as shown later in the paper, even bandwidth allocation is far from sufficient. Thus, the focus of this paper is to design an explicit rate allocation algorithm, which, at a high level, continuously solves a global optimization problem in a distributed fashion and dictates accordingly how much bandwidth is allocated to each connection. Through extensive simulations, we show that explicit rate allocation *does* significantly outperform even bandwidth allocation, across all the various scenarios we've examined.

There are three objectives that an ideal rate allocation algorithm should achieve:

- Minimize server load. This is the first order objective.
- Minimize ISP-unfriendly traffic, while keeping the minimum server load unaffected.
- Maximize peer prefetching, without affecting the first two objectives. When both above objectives are met, peers may still have spare upload bandwidth, which is especially true when the average peer upload bandwidth is greater than the video bit rate. As shown in early work [6], such spare upload bandwidth should be utilized to allow peers to download faster than real-time and cache future content. The so-called *prefetching* can help peers combat neighbor churning, bandwidth jitter, and thus potentially reduce server load, as well as ISP-unfriendly traffic.

In this section, we first present a centralized optimization problem that achieves these objectives exactly through

a 3-stage optimization. We discuss the drawbacks of this approach and propose an intuitive utility-based optimization that is connected to the 3-stage optimization but can be solved distributedly. Finally, we will introduce some modifications to the optimization problem for practical purposes.

3.1 Notations and assumptions

Suppose there are n peers in the system at any instant of time. Denote them as Peer k ($k = 1, 2, ..., n$), ordered by their arrival time. When Peer j arrives, it connects to a subset of peers already in the system (say including Peer i) and requests data from them. $x_{i,j}$ is the rate Peer i allocates from its upload capacity U_i to serve Peer j. Each peer keeps track of its total upload capacity, which can initially be estimated based on historical values and then measured/updated once data starts to flow to its neighbors. S_j denotes the set of all peers uploading to Peer j (Peer j's *upstream* neighbors). D_i denotes the set of all peers downloading from Peer i (Peer i's *downstream* neighbors). The aggregate rate Peer j receives from all of its upstream neighbors is denoted as $x_j = \sum_{i \in S_j} x_{i,j}$.

Denote R_j as Peer j's desired streaming rate in order to maintain smooth video playback. (In general, R_j is a function of time t as it could vary based on the amount of content in its local cache, which changes over time. For simplicity, let $R_j(t)$ be constant for all j for now, equal to the video bitrate R.) It is clear that smooth video playback requires $x_j \geq R_j = R$. If $x_j < R$, Peer j will request data from the server at rate $R - x_j$ to make up for the deficit. If $x_j > R$, on the other hand, Peer j will download data faster than real-time and cache for future use, i.e. prefetch, so as to combat peer churn, bandwidth jitter, etc. The ability to prefetch is one of the key differentiations between VoD and live streaming. The aggregate rate at which Peer i uploads to all its downstream neighbors is $\sum_{j \in D_i} x_{i,j}$. Clearly, this cannot exceed Peer i's upload bandwidth, i.e., $\sum_{j \in D_i} x_{i,j} \leq U_i$. Furthermore, for unified representation, we denote the server as Peer 0. Thus, $x_{0,j}$ is the rate Peer j obtains from the server, which satisfies $x_{0,j} = \max(0, R - x_j)$.

We make the following assumptions for the analysis:

1. Peers cache all the content they watch and keep them until they leave the system.
2. Peers only upload to their downstream neighbors, not vice versa.
3. Peers have more content than their downstream neighbors and are able to upload as fast as needed.

Assumptions (2) and (3) will be removed in designing a corresponding packet-level algorithm and in the simulations.

3.2 Centralized 3-stage optimization

We now present the centralized 3-stage optimization.

3.2.1 1st-stage – minimize server load

Minimizing the server load can be realized as follows:

$$\min \sum_j x_{0,j} \tag{1}$$

$$\text{s.t.} \sum_{i \in \{0, S_j\}} x_{i,j} \geq R \; \forall \; j \neq 0, \quad \sum_{j \in D_i} x_{i,j} \leq U_i \; \forall \; i \neq 0, \text{ and} \tag{2}$$

$$x_{i,j} \geq 0 \; \forall \; i, j. \tag{3}$$

The minimum server load can then be computed as $U_0^{min} = \sum_{j=1}^{n} x_{0,j}^*$, where $\{x_{0,j}^*\}_{j=1}^n$ is an optimal solution.

3.2.2 2nd-stage – minimize ISP-unfriendly traffic

In this stage, we first add the minimum server load U_0^{min} from the 1st stage as an additional constraint. We then associate a *link cost* with ISP-unfriendly traffic. Denote $g_{i,j}(x_{i,j})$ as the cost for Peer i to upload to Peer j at rate $x_{i,j}$.

Minimizing ISP-unfriendly traffic can be represented as minimizing the total link costs:

$$\min \sum_{i,j} g_{i,j}(x_{i,j}) \tag{4}$$

$$\text{s.t. (2), (3) and } \sum_j x_{0,j} \leq U_0^{min} \text{ (from 1st-stage).} \tag{5}$$

The most intuitive choice for the link cost function is a linear one: $g_{i,j}(x_{i,j}) = c_{i,j} x_{i,j}$. For example, if minimizing cross-ISP traffic is the only concern, we can simply set $c_{i,j} = 0$ if Peer i and j belong to the same ISP and $c_{i,j} = c > 0$ otherwise. This formulation could potentially be generalized to incorporate various levels of ISP-unfriendliness, e.g., using different costs to differentiate intra-POP and inter-POP traffic within the same ISP.

After the optimization, we can compute the minimum ISP-unfriendly traffic cost as $G^{min} = \sum_{i,j} g_{i,j}(x_{i,j}^*)$.

3.2.3 3rd-stage – maximize peer prefetching

Finally, we want to allow peers to download more than their demand. One possible formulation is to minimize the remaining of peers' upload capacity, while keeping the minimum server load and ISP-unfriendly traffic cost limited to U_0^{min} and G^{min}. Then, the rate allocation is as follows:

$$\min \sum_i (U_i - \sum_{j \in D_i} x_{i,j}) \tag{6}$$

$$\text{s.t. (5) and } \sum_{i,j} g_{i,j}(x_{i,j}) \leq G^{min} \text{ (from 2nd-stage).} \tag{7}$$

3.3 Utility-based optimization

While it is clear that this 3-stage optimization achieves all the aforementioned objectives *exactly*, it has two main drawbacks. First, this optimization requires a central oracle and is difficult to solve distributedly in a practical setup. Distributed solutions using iterative algorithms *do* exist for each stage. Thus, given a fixed set of peers and a static topology, each stage can indeed be solved in a distributed manner. However, the optimization problem may *not* converge at all in a highly dynamic environment with peer churns, as it requires the result of an earlier stage to be the constraint of a later stage. Second, this formulation enforces an absolute preference of reducing server load over cutting ISP-unfriendly traffic. As an extreme example, it may yield a solution that incurs 100 Mbps additional ISP-unfriendly traffic just to save 1 Kbps server bandwidth.

To overcome these shortcomings, we now present an intuitive utility maximization formulation that has a distributed solution and provides a "knob" to allow flexible tradeoff between the server bandwidth and ISP-unfriendly traffic.

3.3.1 Utility of aggregate received bandwidth

First, we introduce a utility function for each peer in terms of the aggregate rate received from all its upstream neighbors (excluding the server or Peer 0):

$$f(x_j) = \begin{cases} a \cdot x_j & \text{if } x_j \leq R \\ a \cdot R + b \cdot (x_j - R) & \text{if } x_j > R \end{cases}, \tag{8}$$

where a, b are positive constants and $a > b$ (Fig. 2).

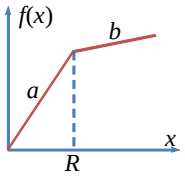


Figure 2: Utility function $f(x_j)$.

The value a can also be interpreted as the cost of server bandwidth. In a peer-assisted VoD session, the server has to supplement all the rate peers cannot obtain from each other. At the same time, each peer has to maintain receiving data at least at the streaming rate R. Hence, up to streaming rate R, every unit of bandwidth that a peer is able to obtain from other peers is a unit of bandwidth the server can save.

A strictly positive b represents the value of prefetching. This is a major difference between live streaming and VoD streaming. In live streaming, peers have roughly synchronous playback times. As a result, they cannot buffer up much ahead of time and there can be very little prefetching, i.e. $b = 0$. In VoD, however, peers' playback times can be sufficiently far apart and there is great value for peers to buffer up whenever possible to save for future use.

The piece-wise linear utility function ensures that, when the total utility is maximized, the server load is minimized. In addition, peer prefetching is maximized simultaneously.

PROPOSITION 1. *The solutions to*

$$\max_{x_{i,j}} \sum_{j\neq 0} f(x_j) \tag{9}$$
$$s.t. \sum_{j\in D_i} x_{i,j} \leq U_i \ \forall\ i \neq 0, \ \textit{and}\ x_{i,j} \geq 0 \ \forall\ i,j.$$

minimize server load and maximize peer prefetching.

Proof sketch: Minimizing server bandwidth can be described as the following optimization problem:

$$\min_{x_{i,j}} \sum_{j\neq 0} a \cdot \max(0, R - x_j) \tag{10}$$
$$\text{s.t.} \sum_{j\in D_i} x_{i,j} \leq U_i \ \forall\ i \neq 0, \ \text{and}\ x_{i,j} \geq 0 \ \forall\ i,\ j,$$

where a is a positive constant.

This is equivalent to

$$\max_{x_{i,j}} \sum_{j\neq 0} a \cdot \min(x_j, R) \tag{11}$$
$$\text{s.t.} \sum_{j\in D_i} x_{i,j} \leq U_i \ \forall\ i \neq 0, \ \text{and}\ x_{i,j} \geq 0 \ \forall\ i,j.$$

Now, we rewrite the utility function (8) as

$$f(x_j) = b \cdot x_j + (a-b) \cdot \min(R, x_j) \tag{12}$$

Since the proposed utility function (8) is strictly increasing in $x_{i,j}$, it can be shown through contradiction that all the solutions to (9) must satisfy $\sum_{j\in D_i} x_{i,j} = U_i \ \forall\ i \neq 0$ and thus maximize peer prefecthing.

Therefore, (9) is equivalent to

$$\max_{x_{i,j}} \sum_{j\neq 0} b \cdot U_j + \sum_{j\neq 0} (a-b) \cdot \min(R, x_j) \tag{13}$$
$$\text{s.t.} \sum_{j\in D_i} x_{i,j} \leq U_i,\ x_{i,j} \geq 0 \ \forall\ i,j.$$

$\sum_j b \cdot U_j$ is a constant and $a - b > 0$, thus (13) is equivalent to

$$\max_{x_{i,j}} \sum_{j\neq 0} a \cdot \min(x_j, R) \tag{14}$$
$$\text{s.t.} \sum_{j\in D_i} x_{i,j} = U_i \ \forall\ i \neq 0, \ \text{and}\ x_{i,j} \geq 0 \ \forall\ i,j.$$

The only difference between (11) and (14) is the constraint on $\sum_{j\in D_i} x_{i,j}$. $\min(x_j, R) = \min(\sum_{i\in S_j} x_{i,j}, R)$ is nondecreasing in $x_{i,j}, \forall\ i,j$. Thus, the solutions to (14) is a subset of those to (11).

In summary, the solutions to (9) is the subset of those to (11) that both minimize server load and maximize peer prefetching. □

3.3.2 Cost of ISP-unfriendly traffic

To incorporate ISP-friendliness, we associate a link cost for each connection, i.e. the cost for Peer i to upload to Peer j at rate $x_{i,j}$ is $g_{i,j}(x_{i,j})$. Taking this into account, the overall utility optimization problem then becomes

$$\max_{x_{i,j}} \sum_{j\neq 0} (f(x_j) - \sum_{i\in S_j} g_{i,j}(x_{i,j})) \tag{15}$$
$$\text{s.t.} \sum_{j\in D_i} x_{i,j} \leq U_i \ \forall\ i \neq 0, \ \text{and}\ x_{i,j} \geq 0 \ \forall\ i,j.$$

As mentioned earlier, a natural choice for $g_{i,j}(\cdot)$ is $g_{i,j}(x_{i,j}) = c_{i,j} \cdot x_{i,j}$, where $c_{i,j}$ is a positive constant that represents the cost of getting each unit of rate from Peer i to j. When positive costs are used for various ISP-unfriendly traffic, it is intuitive that the maximization in (15) will reduce those undesirable traffic, although potentially at the cost of increasing server load. The relationship among $c_{i,j}$, a and b controls the tradeoff between the server load and the ISP-friendliness. To better understand this, let us consider a simple case where $c_{i,j} = 0$ if Peer i and j are within the same ISP, and $c_{i,j} = c > 0$ otherwise.

PROPOSITION 2. *Compared to solutions to (9), solutions to (15) may have higher server rates. For each additional unit of server rate used due to incorporating ISP friendliness, there is at least a reduction of $\frac{a-b}{c}$ units of ISP-unfriendly traffic.*

Proof sketch: Let $\{\overline{x}_{i,j}\}$ be a solution for (9). Let $\{\widetilde{x}_{i,j}\}$ be a solution for (15). Then by definition,

$$\sum_{j\neq 0} (f(\overline{x}_j) - \sum_{i\in S_j} g_{i,j}(\overline{x}_{i,j})) \leq \sum_{j\neq 0} (f(\widetilde{x}_j) - \sum_{i\in S_j} g_{i,j}(\widetilde{x}_{i,j}))$$
$$\Rightarrow \sum_{j\neq 0} (f(\overline{x}_j) - f(\widetilde{x}_j)) \leq \sum_{j\neq 0} (\sum_{i\in S_j} (g_{i,j}(\overline{x}_{i,j}) - g_{i,j}(\widetilde{x}_{i,j})))$$

Using (12),

$$\begin{aligned} \text{LHS} &= \left(b \sum_{j\neq 0} \overline{x}_j + (a-b) \cdot \sum_{j\neq 0} \min(R, \overline{x}_j)\right) - \\ &\quad \left(b \sum_{j\neq 0} \widetilde{x}_j + (a-b) \cdot \sum_{j\neq 0} \min(R, \widetilde{x}_j)\right) \\ &= b \cdot \left(\sum_{j\neq 0} \overline{x}_j - \sum_{j\neq 0} \widetilde{x}_j\right) + \\ &\quad (a-b) \cdot \left(\sum_{j\neq 0} \min(R, \overline{x}_j) - \sum_{j\neq 0} \min(R, \widetilde{x}_j)\right). \end{aligned}$$

As mentioned earlier, $\sum_{j\neq 0}\overline{x}_j = \sum_{j\neq 0} U_j \geq \sum_{j\neq 0}\widetilde{x}_j$. $\sum_{j\neq 0}\min(R,\overline{x}_j) - \sum_{j\neq 0}\min(R,\widetilde{x}_j)$ is the increase in server load due to incorporating ISP-friendliness. In short,

$$\text{LHS} \geq (a-b)\cdot(\text{additional server load}). \tag{16}$$

On the other hand,

$$\text{RHS} = c\cdot\sum_{j\neq 0}(\sum_{\substack{i\in S_j\\ i\notin I_j}}\overline{x}_{i,j}) - c\cdot\sum_{j\neq 0}(\sum_{\substack{i\in S_j\\ i\notin I_j}}\widetilde{x}_{i,j}) \tag{17}$$

where I_j denotes the subset of Peer j's neighbors in the same ISP as j.

Here, $\sum_{j\neq 0}(\sum_{\substack{i\in S_j\\ i\notin I_j}} x_{i,j})$ is nothing but the total ISP-unfriendly rate using allocation $\{x_{i,j}\}$. Hence

$$\text{RHS} = c\cdot(\text{saving in ISP-unfriendly rate}). \tag{18}$$

Combining (16) and (18), we get

$$\text{saving in ISP-unfriendly rate} \geq \frac{a-b}{c}(\text{additional server load}).$$

Intuitively, setting $c > a > b$ will result in a pure ISP-friendly solution that eliminates ISP-unfriendly traffic completely. Setting $a > c > b$ will eliminate cross-ISP prefetching. Setting $a > b > c$ will guarantee full utilization of peers' upload bandwidth at all time, which may cause unnecessary ISP-unfriendly traffic. We typically do not choose $c > a > b$.

3.4 Distributed solution

Since $f(\cdot)$ is concave, the problem at hand (15) is concave as long as $g_{i,j}(\cdot)$ is convex $\forall\ i,j$. It is straightforward to apply classical distributed solutions [8] to solve such a convex optimization problem with linear constraints. In other words, by using convex $g_{i,j}(\cdot)\ \forall\ i,j$, we are able to obtain a distributed solution to (15). In particular, we adopt the feasible steepest descent algorithm: at Peer i, $x_{i,j}$ is initialized to $\frac{U_i}{|D_i|}$ for $j\in D_i$ and 0 otherwise, where $|D_i|$ is the number of downstream neighbors Peer i has. $x_{i,j}$ is updated at each step as follows:

$$\dot{x}_{i,j} = \Delta\cdot(\frac{\partial}{\partial x_{i,j}}f(x_j) - \frac{\partial}{\partial x_{i,j}}g_{i,j}(x_{i,j})), \tag{19}$$

$$x_{i,j} = [x_{i,j} + \dot{x}_{i,j}]^+ \tag{20}$$

where $[\cdot]^+$ means l_2 projection onto a feasible set, which guarantees convergence for this problem [8].

Due to the symmetry of the linear constraints, projection onto feasible set (20) can be easily implemented as follows:

```
while(Σ x_{i,j} > U_i) {
  N_i = number of neighbors with x_{i,j} > 0;
  foreach(j)
    if(x_{i,j} > 0)   x_{i,j} = x_{i,j} − min(x_{i,j}, (Σ x_{i,j} − U_i)/N_i); }
```

Note that in order for Peer i to carry out the update step, the only external information required is the aggregate received bandwidth at its downstream neighbors, i.e. $x_j,\ j\in D_i$. This information can be easily piggybacked in Peer j's packet requests.

3.4.1 Connection to 3-stage optimization

It is in fact straightforward to show that the utility-based optimization problem (15) is equivalent to

$$\min(a-b)\sum_{j\neq 0}x_{0,j} + b\sum_{i\neq 0}(U_i - \sum_{j\neq 0}x_{i,j}) + \sum_{i,j}g_{i,j}(x_{i,j}) \tag{21}$$

$$\text{s.t.}\ \sum_{i\in\{0,S_j\}}x_{i,j}\geq R\ \forall\ j\neq 0,\ \ \sum_{j\in D_i}x_{i,j}\leq U_i\ \forall\ i\neq 0,\ \text{and}$$

$$x_{i,j}\geq 0\ \forall\ i,j.$$

(21) is simply a weighted sum of the 3 objective functions in the 3-stage optimization. By choosing the weights $a-b$, b and functions $g_{i,j}(\cdot)$, we can set a relative priority and consequently a flexible tradeoff among server bandwidth, various kinds of ISP-unfriendly traffic, and pre-fetching. The utility-maximization formulation is introduced as it bears a more intuitive meaning and facilitates understanding of the following modifications we make for practical purposes.

3.5 Practical considerations

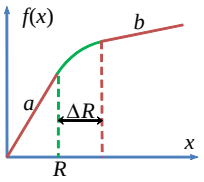


Figure 3: Modified Utility Function.

The piece-wise linear utility function $f(\cdot)$ and linear link cost functions $g_{i,j}(\cdot)$ have shortcomings in practice. First, $f(\cdot)$ has a sudden slope change at exactly video rate R. This causes unsteady convergence behavior. Further, it eliminates certain ISP-unfriendly prefetching completely if b is smaller than the corresponding link cost. As a matter of fact, moderate prefetching is always beneficial in reducing the server load and improving peers' uplink utilization in the long run, especially in the presence of peer churning, bandwidth jitter or flash crowd (when neighboring peers' playback points are very close to each other). Hence we modified $f(\cdot)$ slightly, by connecting the two linear components with a concave smooth curve of width ΔR, as qualitatively shown in Fig. 3. The implication is that we potentially allow moderate ISP-unfriendly prefetching, but bound the amount by ΔR per peer in the worst case. We use $\Delta R = 0.1R$ and a Deg. 3 polynomial for the curve. This modification leads to a small increase in ISP-unfriendly traffic, in exchange for steadier convergence, and lower server bandwidth. Second, we change the link cost function $g(\cdot)$ from linear to flat quadratic such that the optimization is strictly concave. This enables convergence to a unique global optimum and thus better performance in a highly dynamic environment. We set $g(\cdot)$ such that $\frac{d}{dx}g(x) = c+\varepsilon x$ with $\varepsilon \ll \frac{c}{R}$. By making these modifications, the final solution will deviate from the optimal one directly derived from (21). We will examine the issue of sub-optimality in detail through simulations.

Additionally, to take peers' evolving buffer level into consideration, peers also convert their buffer level into an equivalent received rate. We use a simple linear conversion where the aggregate received rate is incremented by $\frac{\text{buffer level}}{\text{buffer length}}\cdot R$. For instance, suppose Peer j's buffer can hold 160 packets. Then if Peer j has 40 packets in the buffer, it will consider its aggregate received rate to be $\frac{1}{4}R + \sum_{i\in S_j}x_{i,j}$.

4. PACKET-LEVEL ALGORITHM

So far, the rate allocation stays at an ideal network flow level. In other words, upload bandwidth is treated as fluid, which can be divided and utilized as finely as desired. We now translate the flow-level rate allocation algorithm into a corresponding packet-level algorithm, which, as demonstrated in Sec. 5.1, conforms nicely to the flow-level one.

4.1 Packet-level algorithm

Sliding window: We use a sliding buffer for packet management. Peers maintain a buffer of interest and only download packets within the range. They slide the buffer forward every fixed interval (1 second in our case). Peers cache all the content they have watched and make it available to their neighbors (both upstream and downstream). Peers advertise about their packet availability once per second.

Packet request: A peer ranks all the missing packets in its buffer by their importance. From each neighbor, it requests the most important packet possessed by the neighbor, as long as the number of pending requests at that neighbor is below a certain threshold. The importance of a packet is determined as follows. We treat one second's worth of packets as one segment, and a segment is more important if it's closer to the playback deadline. Within a segment, the rarer a packet is in the peer's one-hop neighborhood, the more important that packet is. We also define an urgency buffer, which is the first u second(s) of the buffer, of which the peers will simply request all the missing packets from the server (we use $u = 1$ in all our experiments). Unlike in the flow-level framework, peers are *not* prohibited from requesting packets from their downstream neighbors. Removing this restriction facilitates sharing among peers when their playback points are close.

Rate allocation and request response: The more important part of the packet-level scheduling algorithm is how serving peers determine which neighbor's request to satisfy first. We adopt a token system to implement the rate allocation algorithm at a packet level. We define a rate-allocation interval T_{RA} and a token-allocation interval T_{TA}. Namely, every T_{RA} second, Peer i carries out rate update as in (19) and computes the appropriate rate to allocate to each of its neighbors. This computation needs not be synchronized among peers. Every T_{TA} second, Peer i gets a certain number of tokens to give out. Again, peers need not synchronize their intervals. The number of tokens a peer gets is proportional to its upload bandwidth. For instance, if Peer i has U_i Kbps upload bandwidth, it gets U_i tokens each round. It then allocates $x_{i,j}$ tokens to neighbor Peer j as computed in (19). Note that U_i and $x_{i,j}$ need not be integers, as partial token will simply be left with peers for use in future rounds. In deciding which neighbor's request to satisfy, Peer i chooses the neighbor with the highest level of tokens and deduct $\frac{\text{packet size}}{T_{TA}}$ tokens from the neighbor.

Request rejection: Peers turn down requests that they are unlikely to fulfill, so that the requesters do not wait unnecessarily. For this purpose, peers include *time to playback deadline* (D_m) in the request of Piece m. Peer i knows how much token its neighbor Peer j has accumulated (L_j). Peer i will turn down the request for Piece m from j if by the time the packet expires, Peer j still would not have received enough tokens given the current token allocation rate, i.e.

$$L_j + x_{i,j} \cdot \lfloor \frac{D_m}{T_{TA}} \rfloor < \frac{\text{packet size}}{T_{TA}}.$$

4.2 Computation complexity and overhead

Each iteration of the rate allocation at Peer i involves computing the update step for each of its neighbor j. Since the derivatives of the utility and cost functions are closed form expressions and pre-stored, this is a fairly straightforward computation with a few multiplications and additions. Each peer typically has no more than 20 neighbors including both upstream and downstream neighbors. In our simulations, we carry out the rate allocation once per second ($T_{RA} = 1$ second). For today's personal computers with GHz processors, the computation overhead is marginal.

There is almost no communication overhead involved either. The only additional information that peers have to transmit to their neighbors is their aggregate received bandwidth and the playback deadline for each packet. Peers can easily piggyback this information in their packet requests.

5. EVALUATION

To evaluate the proposed packet scheduling algorithm, we implemented a discrete-time packet-level simulator that can simulate the peer-assisted VoD system, as described in Sec. 2, with up to 10k concurrent users. We first demonstrate that the proposed packet-level algorithm converges closely to the solution of the fluid-level optimization problem (9). We then evaluate its performance using both synthetic and real-world traces. In particular, we demonstrate the effectiveness of the proposed algorithm in reducing ISP-unfriendly traffic and server load compared to a simple even rate allocation scheme. We showcase the flexible tradeoff between server load and ISP-unfriendly traffic enabled by the proposed framework. We also mimic the MESH platform and implement a heuristic-based ISP-friendly topology building scheme and demonstrate that the proposed algorithm can bring significant additional performance gain.

We use VoD traces collected from MSN Video service in July, 2007. The 10 most popular videos' traces logged around 10 million users. These traces contain clients' public IP addresses and download bandwidth. We map these IP addresses to AS numbers [7] and find the relationship among ASes using CAIDA's data [9]. We say traffic between Peer i and Peer j is ISP-friendly if i and j are in the same AS or in ASes with a peering relationship, otherwise it is ISP-unfriendly. We infer peers' upload bandwidth from their download bandwidth as in [6], and quantize it to 128, 384, 512 and 768 Kbps for the discrete-time simulator[1]. For all our simulations, data packet size is 4KB[2].

5.1 Convergence behavior

We first demonstrate that the packet-level algorithm converges closely to the solution of the optimization problem (9).

[1]Though we have only used a single bandwidth constraint per peer, this needs not be the case in practice. For example, peers within the same local area network may have higher bandwidth among them. As long as all the bandwidth constraints are node constraints instead of link constraints, the proposed framework will work directly. We only need to add the additional bandwidth constraints to the formulation and modify the specific implementation of projection onto feasible set accordingly. In this section, we continue to use one bandwidth constraint per node as it is the only information we can derive using the data at hand.

[2]Each packet equals to multiple network packets. The size is chosen to strike a balance between communication overhead and flexibility and mimics that used in the real system.

Using statistics obtained from the MSN traces, we simulated peers from the 5 ASes with the most users. At Time 0, peers start joining the system following a Poisson arrival process with an arrival rate of 1.2 peers per second. The video is 300 seconds long at 512 Kbps. Peers stay in the system until the end of the video and leave immediately after watching the entire video. In other words, peers arrive and stay for exactly 300 seconds. In steady state, there are on average $1.2 \times 300 = 360$ peers in the system.

Every second, we log the current topology and run centralized global optimization to obtain the solution of (9) and the corresponding server load and amount of ISP-unfriendly traffic. We refer to this as the *centralized fluid-level* solution. We also log the server load and amount of ISP-unfriendly traffic using our packet-level discrete-time simulator every second and plot it against the centralized solution as shown in Fig. 4. Fig. 4(a) plots the number of peers in the system over time. Fig. 4(b) and (c) plot the server load and amount of ISP-unfriendly traffic respectively. We see that both curves remain very close to the centralized fluid-level solution even though peers constantly join and leave.

We then vary the peer arrival rate to see how the proposed algorithm responds to much higher peer arrival rates, i.e. more dynamic topology. We compute the average percentage difference between the packet-level simulation result and the centralized fluid-level solution in the following way. Every second, we compute the (absolute) percentage difference as $\frac{|\text{packet-level result - fluid-level centralized result}|}{\text{fluid-level centralized result}}$. We compute the average percentage difference (over 15 minutes) for each peer arrival rate and plot it in Fig. 4(d). It shows that regardless of the peer arrival rate and fast evolving topology, the packet-level algorithm converges very closely to the centralized fluid-level solution.

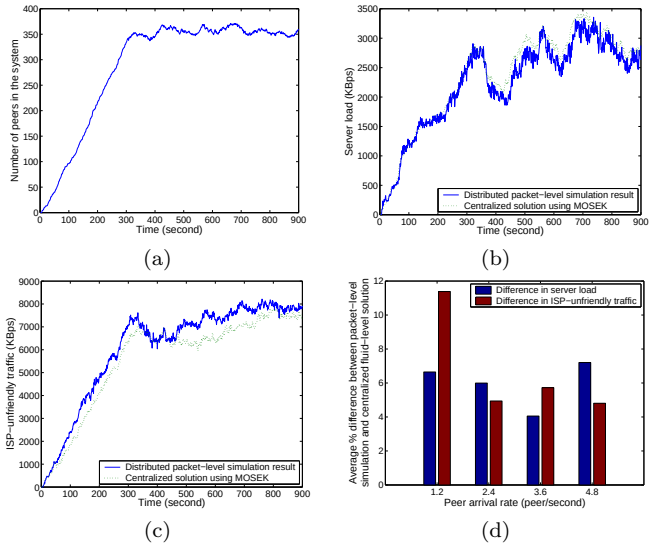


Figure 4: Convergence behavior of proposed packet-level algorithm. Peers have Poisson arrival process of rate 1.2/sec. Peers watch the entire video (5 min) then leave. (a) Number of concurrent peers over time. (b) Server load over time (KBps). (c) ISP-unfriendly traffic over time (KBps). (d) Percentage discrepancy between packet-level simulation and centralized fluid-level solution as peer arrival rate varies from 1.2 to 4.8, i.e. avg. number of concurrent peers varies from 360 to 1440.

5.2 Synthetic trace results

In this part of the result, we demonstrate:

1. the proposed framework enables a flexible tradeoff between server load and ISP-unfriendly traffic;
2. the proposed framework outperforms even bandwidth allocation[3] in reducing both server load and ISP-unfriendly traffic;
3. given an ISP-friendly topology, the proposed algorithm provides significant additional reduction in ISP-unfriendly traffic.

We use synthetic traces to control a relatively stable number of peers in the system to make certain comparisons. We simulated peers using statistics of the Top 5 ASes. Peers have a Poisson arrival process at the rate of 2 peers per second. Peers stay until the end of the video. The video is 300 second long at 512 Kbps. In steady state, there are 600 peers in the system. We compare results using two different topology building strategies. In the case with ISP-unaware topology, peers simply connect to a certain number of upstream peers (8 in our case) with the closest playback points. In the case with ISP-friendly topology, we mimic the MESH platform and generate ISP-friendly topologies in the following way. Among the certain number of upstream peers a peer tries to connect to, it tries to make 70% of the connections with peers in the same or peering ASes[4]. If this cannot be satisfied right away, the peer will make as many connections with peers in the same or peering ASes as possible and ensure a minimum total number of connections (6 in our case) by connecting to peers in non-peering ASes. The peer then queries the tracker periodically to see if additional upstream peers in the same or peering ASes become available[5].

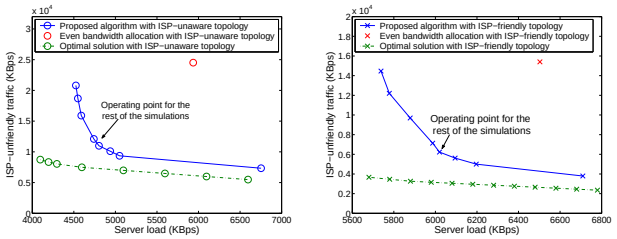
To study the tradeoff between server load and ISP-unfriendly traffic, we vary the coefficient of the linear term in the link cost function (c) and plot the average server load versus the average ISP-unfriendly traffic using both ISP-unaware and ISP-friendly topologies (solid lines in Fig. 5 (a) and (b) respectively). As expected, when we increase c, ISP-unfriendly traffic is reduced at the cost of increased server load.

The single data point in Fig. 5 (a) and (b) is the result of using even bandwidth allocation. All but one point on the performance curve of the proposed algorithm lie to the lower left of the even bandwidth allocation data point, indicating both lower server load and lower ISP-unfriendly traffic.

Comparing Fig. 5 (a) and (b), we see that an ISP-friendly topology helps reducing ISP-unfriendly traffic. Using the same even bandwidth allocation scheme, the amount of ISP-unfriendly traffic is reduced from 250,000 KBps to 150,000 KBps, an reduction of 40%, at the cost of approximately 500 KBps increase in server load. However, the proposed algorithm can still provide a significant amount of additional reduction in ISP-unfriendly traffic.

We acknowledge that, due to the strictly convex link cost function (to ensure a unique global optimum), there will be a performance gap between the proposed algorithm and the ideal 3-stage optimization. Fig. 5 also plots the optimal solution of the 3-step optimization. The gap between the

[3]To simulate even bandwidth allocation, each peer serves its neighbors' packet requests in a round-robin fashion, thus almost evenly splitting its upload bandwidth among them.
[4]This can be done either by the tracker or by peers themselves using the IP-AS map followed by AS relationship table
[5]This could happen if an upstream peer in the same or peering AS has freed up an upload slot due to peer leaving.



(a) (b)

Figure 5: ISP-unfriendly traffic vs. server load for proposed algorithm with different link costs and even bandwidth allocation with (a) ISP-unaware topology and (b) ISP-friendly topology.

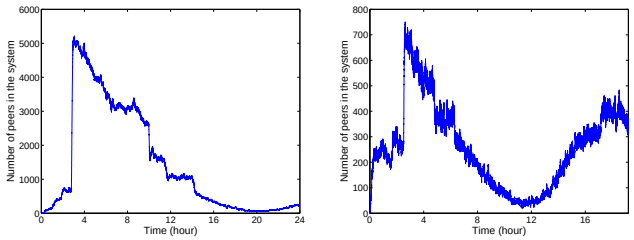
proposed algorithm and the 3-stage optimization is larger when c is small, i.e., when the server load needs to be minimized. This is because the quadratic term of the link cost function ($\varepsilon x^2 + cx$) becomes more dominant for small c's. In those cases, as the transmission rate increases, it becomes almost as expensive to transmit within and across ASes. This suppresses traffic within the same (or peering) ASes to some extent, thereby increasing ISP-unfriendly traffic. This, however, cannot be avoided unless the quadratic term of the link cost is dropped, which then causes convergence issues. In fact, without the quadratic term, the iterative algorithm may never converge under peer dynamics. It is interesting future work to reduce the gap when the server load needs to be optimized while maintaining good convergence behavior.

From Fig. 5 (a) and (b), we see that even though the topologies are quite different, the shape of the tradeoff curve of the proposed algorithm remains very similar. We find the tradeoff behavior from such small-scale simulations to be a helpful indicator for choosing an appropriate operating point. For the rest of our simulations, we fix $c = \frac{a}{2}$, which corresponds to the points indicated by the arrows.

In practice, we do *not* believe that a single set of parameters can fit all scenarios. It is important to allow flexible tradeoffs for different ISPs and at different times. For instance, say the server load is charged based on 95^{th} percentile rule, it makes perfect sense to set a small link cost during peak time to keep server load small. While during trough period, when the server could afford more bandwidth, the link cost can be set to a relatively large value. In this way, over traffic peak and valley, the value of link cost should change slowly through the day. While the specific values of the parameters will need online tuning[6], a possible strategy for picking the initial values is to follow the results of small-scale simulations.

5.3 MSN trace results

In addition to peers' public IP address and their download bandwidth, the MSN traces also log peers' joining time and the amount of time they stay in the streaming session. We follow the traces in simulating peers' joining and departing. We use two traces, each *24 hours long*, with very different populations. Fig. 6 shows the number of peers over the 24-hour period for each trace. From here on, we refer to the first trace (Fig. 6 (a)) as the gold trace and the second trace (Fig. 6 (b)) as the silver trace. We extract peers in the *20 most popular ASes* to make the number of peers to simulate manageable. Even so, at peak time, the gold trace still logged more than 5000 concurrent users. As stated before, we look up AS relationships using CAIDA's data.



(a) (b)

Figure 6: Number of peers over a period of 24 hours in (a) the gold trace and (b) the silver trace.

For each trace, we compare the server load and the amount of ISP-unfriendly traffic for the following four systems:

1. proposed algorithm with ISP-unaware topology;
2. even allocation with ISP-unaware topology;
3. proposed algorithm with ISP-friendly topology;
4. even allocation with ISP-friendly topology.

The gold trace video is 456 seconds long. We quantize the streaming rate to 512 Kbps. Fig. 7 shows the server load (in KBps) and the amount of ISP-unfriendly traffic (in KBps) over time for all four systems. Fig. 7 shows the the corresponding CDFs over the whole day[7].

Consistent with the previous result, we see that ISP-friendly topology is able to effectively reduce the amount of ISP-unfriendly traffic when using the same even rate allocation scheme. However, the proposed algorithm is able to provide significant additional performance gain. Compared to the baseline system with ISP-unaware topology and even rate allocation, the best scheme (ISP-friendly topology with proposed algorithm) can achieve an overall reduction in ISP-unfriendly traffic of 40% on average. Using the proposed algorithm also provides significant reduction in server load. In fact, the 95 percentile server load is reduced almost by half compared to even bandwidth allocation schemes (Fig. 7 (c)).

For the silver trace, the video is 217 seconds long. The bit rate is 503 Kbps, which we quantize to 512 Kbps. Fig. 8 (a) and (b) show the CDFs of the server load and ISP-unfriendly traffic. Here we observe very similar behavior to the gold trace though the popularity of the trace is very different. In summary, the proposed algorithm combined with ISP-friendly topology is able to reduce the amount of ISP-unfriendly traffic by up to 50% compared to the baseline system.

5.4 Is topology building enough?

The evaluations so far have clearly demonstrated: 1) building ISP-friendly topology can help reduce both server load and ISP-unfriendly traffic; 2) once a topology is established, the proposed rate allocation can provide substantial additional gain. In this section, we study the following question: if we use a more sophisticated topology building mechanism, will it achieve most of the gain such that an explicit rate allocation becomes unnecessary? While we cannot completely answer the question yet, our study did provide some intu-

[6]A practical solution is to collect feedback from peers and gradually change the parameter assignment.

[7]The CDF curves have an unusual shape due to the fluctuation of the number of peers in the system over time.

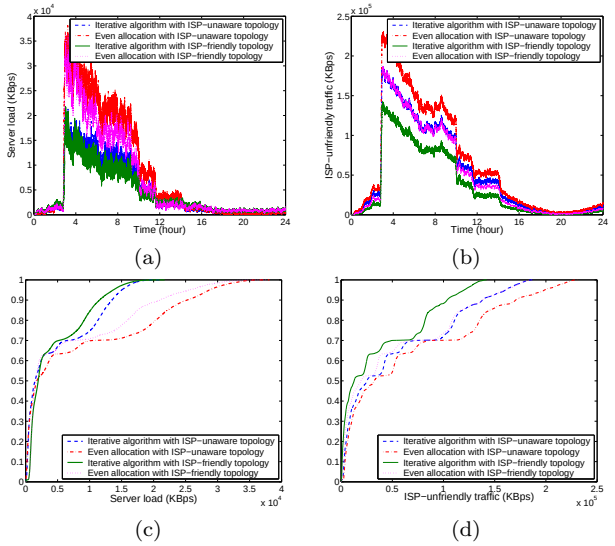


Figure 7: Gold trace: (a) server load over time, (b) ISP-unfriendly traffic over time, (c) server load CDF, (d) ISP-unfriendly traffic CDF .

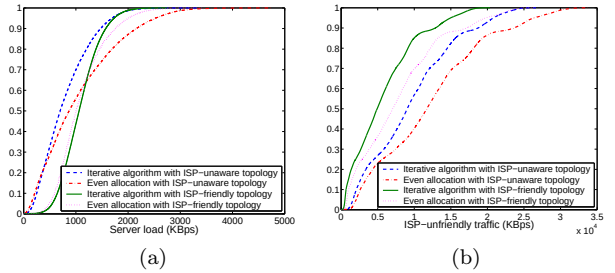


Figure 8: Silver trace: (a) server load CDF, (b) ISP-unfriendly traffic CDF.

ition on why a centralized topology building guide may not be sufficient in practice.

5.4.1 The iTracker approach

We adapt the *iTracker* approach from the recently proposed P4P framework [11] and evaluate a similar topology building mechanism in the VoD scenario. The P4P framework advocates a flexible and light-weight portal, through which ISPs provide explicit information and guidelines to P2P apps. The centerpiece of the proposed P4P framework is a coordinating unit called iTracker. In short, iTracker takes information from both ISPs (e.g., network topology, bandwidth cap, etc.) and P2P apps (e.g., peers' upload and download bandwidth, IP addresses, etc.) as input and solves a global optimization problem to strike a balance between ISPs' objectives (e.g., limit cross-ISP traffic) and P2P apps' objectives (e.g., speed up download). The outcome is used to guide topology building. Specifically, the iTracker optimization consists of two parts: 1) modeling peer sharing efficiency, following the method proposed by Qiu et al. [10]; 2) solving the global optimization problem and allocating rates on each individual link. Total traffic within ISP and between ISPs are then used to compute ratios, which guides topology building for future peers. For example, say there are three ISPs (A, B and C). Suppose the outcome of the global optimization is that the ratio among total traffic inside ISP A, total traffic between A-B, and total traffic between A-C is 7:2:1. Hence, when a new peer from ISP A joins, it will establish connections based on this ratio: 70% connections to peers within A, 20% to B and 10% to C.

When we adapt the iTracker approach to the VoD scenario, we make two changes to single out the effect of topology building. First, we remove the first part of efficiency modeling and compute the efficiency exactly, by taking a snapshot of the entire existing topology. The 3-stage global optimization is executed on this topology. Once optimal rates are allocated, for each ISP, we compute the total internal/external traffic and calculate the corresponding ratio, which is then used to guide topology building in a similar way. Second, the global optimization is executed whenever a new peer joins. Though neither complete snapshot nor frequent optimization is scalable in practice, we did it in an attempt to create a scenario that allows us to focus on the impact of sophisticated topology building alone.

5.4.2 Evaulation

We simulated 24hr of the silver MSN trace (see Sec. 5.3 for more details on the trace). Fig. 9 shows the CDF of server load and ISP-unfriendly traffic using the following systems:

1. proposed algorithm with ISP-unaware topology;
2. even allocation with ISP-unaware topology;
3. proposed algorithm with iTracker-guided topology;
4. even allocation with iTracker-guided topology.

Though the iTracker-guided topology provides gain in reducing ISP-unfriendly traffic, the proposed iterative algorithm can be used on top of the iTracker-guided topology to achieve significant additional gain. Compared to Fig. 8, the iTracker-guided topology does not seem to provide better result than the simple heuristic-based ISP-friendly topology building scheme. While we acknowledge that our adaptation of the iTracker approach may not be the best one, the following reasons also play a role in the result.

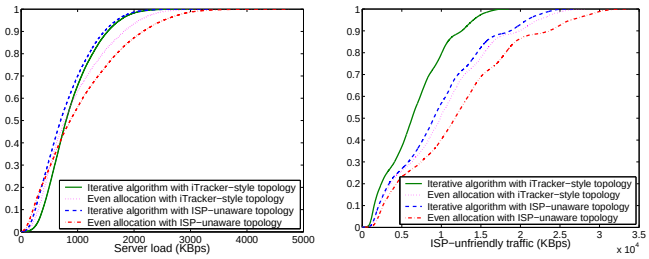
First, the iTracker guides the peers on topology building by providing a ratio of peers from each ISP that they should connect to. This ratio is an *average* among all peers from the same ISP, not for each *individual* peer. Thus, if a new peer in this ISP makes connections according to this ratio, the resulting topology is typically suboptimal.

Second, as peers churn, the optimization result may fluctuate among different optimums that correspond to completely different rate allocations on each link and yield different connection ratios. While new peers can follow the newest iTracker result in making the connections, it is infeasible for existing peers to change their connections to conform to the new result. This is because video streaming applications are much more sensitive to breaking and establishing new links compared to file download due to the strict delivery deadline for every packet.

In practice, it is clearly not scalable to run the iTracker optimization every time a new peer joins a VoD session, nor is it possible to optimize over the entire topology. This will only cause the resulting topology to be of lower quality and the proposed algorithm can be used to improve performance.

6. RELATED WORK

P2P and ISP: Many measurement-based studies have argued that peer-to-peer or peer-assisted appliations can indeed become ISP-friendly. Saroiu et al. [12] collected in-



(a) (b)

Figure 9: Effect of sophisticated topology-building: (a) and (b) show the server load and ISP-unfriendly traffic CDF over 24 hr using the silver MSN trace.

bound/outbound traffic of a large university and showed that there are lots of redundancy in P2P traffic. Gummadi et al. [13] studied Kazaa traffic and showed that ISPs' external traffic can be greatly reduced by incorporating locality-awareness. Karagiannis et al. [14] studied BitTorrent traffic and showed that locality-aware P2P delivery solutions can significantly alleviate the induced cost at the ISPs. Huang et al. [6] studied video-on-demand traffic from a large portal and showed there is a trade-off to be explored between server load and ISP-unfriendly traffic. Griwodz [15] estimated the cost of deployment and operation of a VoD service in a CDN with several hierarchical levels, and studied the placement of VoD content that achieves the minimal cost.

Topology building: A number of studies have shown that ISP-friendliness can be incorporated into topology building [3, 4, 11, 16, 17]. It can be an explicit mechanism, as simple as biased neighbor selection [16], or more sophisticated schemes requiring a dedicated oracle [3], drafting behind CDNs' infrastructure [4], or executing optimizations in an iTracker [11]. It can be extended to take into account multi-tier network topology, from subnets, POPs to ISPs [17]. The topology building mechanism can also be implicit. As shown in the recent study of UUSee [18], peers tend to form ISP-based clusters as the topology evolves naturally. In our study, we confirm the benefit from topology building and further demonstrate that there is substantial additional gain from rate allocation.

Rate allocation: Distributed optimization framework, a well-explored direction in network flow problems [19, 20, 21], has been applied to P2P applications recently. All the studies [22, 23, 24] so far have been focusing on either application layer multicast or P2P live streaming. Instead, we study peer-assisted VoD, which deals with additional challenge of peer prefetching, besides QoS and ISP-friendliness.

7. CONCLUSIONS

Making P2P applications ISP-friendly has been a topic of great interest. Instead of tackling the problem from a topology-building angle, we proposed to study a distributed bandwidth allocation algorithm for video-on-demand that minimizes ISP-unfriendly traffic without much impact on the server load. We formulated the problem using an optimization framework that allows a flexible tradeoff between server load and ISP-unfriendly traffic. We presented a distributed solution and translated it into a light-weight packet-level scheduling algorithm. We evaluated the performance of the proposed solution with a discrete-time packet-level simulator using both synthetic and real-world traces. We showed that the proposed algorithm can bring significant additional reduction (up to 40%) in ISP-unfriendly traffic on top of an ISP-friendly topology without incurring additional server load. In the future, we would like to integrate the proposed work into the MESH platform and understand its behavior in the real-world.

8. REFERENCES

[1] M. Goodman, Internet Video Forcast: Broadband Emerges as an Alternative Channel for Video Distribution. *Yankee group*, 2006.

[2] Global Internet Geography 2006. *TeleGeography Research*, 2006.

[3] V. Aggarwal, et al., Can ISPs and P2P Users Cooperate for Improved Performance? *ACM SIGCOMM Computer Communication Review*, vol 37, pp. 29–40, Jul. 2007.

[4] D. Choffnes, and F. Bsutamante, Taming the Torrent: A Practical Approach to Reducing Cross-ISP Traffic in Peer-to-Peer Systems, In *Proc. of SIGCOMM*, Aug. 2008.

[5] MESH: Peer-assisted Distribution Platform. Microsoft, http://meshv1.edge.msn.com.

[6] C. Huang, et al., Can Internet Video-on-Demand be Profitable? In *Proc. of SIGCOMM*, Aug. 2007.

[7] N. Spring, et al., Measuring ISP Topologies with Rocketfuel. *IEEE/ACM Trans. on Networking*, vol 12, pp. 2–16, Feb. 2004.

[8] D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific.

[9] The Cooperative Association for Internet Data Analysis. http://www.caida.org.

[10] D. Qiu, and R. Srikant, Modeling and Performance Analysis of BitTorrent-like Peer-to-Peer Networks. In *Proc. of SIGCOMM*, Aug. 2004.

[11] H. Xie, et al., P4P: Proactive Provider Participation for P2P. In *Proc. of SIGCOMM*, Aug. 2008.

[12] S. Saroiu, et al., An Analysis of Internet Content Delivery Systems. In *Proc. of Symposium on Operating Systems Design and Implementation*, Dec. 2002.

[13] K. Gummadi, et al., Measurement, Modeling and Analysis of a Peer-to-Peer File-Sharing Workload. In *Proc. of ACM Symposium on Operating Systems Principles*, Oct. 2003.

[14] T. Karagiannis, et al., Should Internet Service Providers Fear Peer-Assisted Content Distribution. In *Proc. of ACM Internet Measurement Conference*, Oct. 2005.

[15] C. Griwodz, Movie Placement in a Hierarchical CDN with Stream Merging Mechanisms. In *Proc. of SPIE/ACM International Conference on Multimedia Computing and Networking*, Jan. 2004.

[16] R. Bindal, et al., Improving Traffic Locality in BitTorrent via Biased Neighbor Selection. In *Proc. of IEEE International Conference on Distributed Computing Systems*, Jul. 2006.

[17] J. Li, Locality Aware Peer Assisted Delivery: the Way to Scale Internet Video to the World. *Packat Video*, 2007.

[18] C. Wu, et al., Magellan: Charting Large-Scale Peer-to-Peer Live Streaming Topologies. In *Proc. of International Conference on Distributed Computing Systems*, Jun. 2007.

[19] F. Kelly, et al. Rate Control in Communication Networks: Shadow Prices, Proportional Fairness and Stability. *Journal of the Operational Research Society*, vol 49, pp. 237–252, Mar. 1998.

[20] D. Wei, et al., FAST TCP: Motivation, Architecture, Algorithms, Performance. *IEEE/ACM Trans. on Networking*, vol 14, pp. 1246–1259, Dec. 2006.

[21] H. Han, et al., Multi-Path TCP: A Joint Congestion Control and Routing Scheme to Exploit Path Diversity in the Internet. *IEEE/ACM Trans. on Networking*, vol 14, pp. 1260–1271, Dec. 2006.

[22] D. Lun, et al., Achieving Minimum-cost Multicast: A Decentralized Approach Based on Network Coding. In *Proc. of INFOCOM*, Mar. 2005.

[23] C. Wu, and B. Li, Strategies of Conflict in Coexisting Streaming Overlays. In *Proc. of INFOCOM*, May. 2007.

[24] C. Wu, and B. Li, On Meeting P2P Streaming Bandwidth Demand with Limited Supplies. In *Proc. of SPIE/ACM International Conference on Multimedia Computing and Networking*, Jan. 2008.