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 bring substantial additional gain, on top of previously proposed schemes advocating ISP-friendly topologies.

Categories and Subject Descriptors

C.2.m [Computer - Communication Networks]: Distributed Systems

Keywords

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

1. INTRODUCTION

Internet video is posd to be the next big thing. It is predicted that the revenue from Internet video will grow from $0.9 billion in 2006 to $4.2 billion by 2011, an annual growth rate of 36% [1]. But, as consumers spend more and more time watching videos online, they are becoming increasingly unsatisfied at 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 (e.g., ABC’s full episode streaming, Netflix on-demand, Amazon Unbox, etc.), as well as devices with such capabilities (Media Center by Microsoft, Apple TV, TiVo, etc.). Once in the living room, consumers also quickly realize that they prefer to go beyond YouTube's limited quality and enjoy SD or even HD experience. However, providing high quality Internet video with a traditional client-server model is very costly [2]. More importantly, the ever mounting demand is adding significant pressure onto existing server-based infrastructures (e.g., data centers, content distribution networks (CDNs), etc.), which are already under heavy burden to live up to their current load. As a result, high-profile failures are not uncommon, e.g., MSNBC’s democratic presidential debate webcast mess [3], OperaWeb show crash [4]. Not to mention that Internet itself is predicted to melt down if online video does become mainstream [5].

Fortunately, on the heel of such crisis, peer-to-peer (P2P) technology comes to rescue. 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, e.g., 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, which stops BitTorrent traffic by sending reset packets. Practices as such often create huge backlash once they are discovered and made public (even FCC got intervene 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 [6] 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. In this work, we focus on designing an ISP-friendly rate allocation solution at the protocol level, for
peer-assisted VoD. 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 optimal.
- We translate the fluid-level rate allocation scheme into an implementable packet-level scheduling algorithm that conforms nicely to the fluid-level rate allocation. We also develop a highly efficient packet-level simulation platform, which is able to handle large peer population at the cost of slightly sacrificed fidelity. This platform enables us to simulate real-world traces, at a scale up to 10K concurrent peers.
- 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 bring substantial additional gains on top of ISP-friendly topologies.

2. SYSTEM DESCRIPTION

![MESH System Architecture](image)

The proposed rate allocation algorithm is designed for the MESH platform [7] 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, however, 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 started in closer time stamps (compared to the client’s own start time) are favored. This is based on an early discovery [8] that matching peers in such a matter helps increase upload efficiency.

3. PROBLEM FORMULATION

In a peer-to-peer system, be it streaming or file sharing, each peer connects to multiple other peers (neighbors) simultaneously. The peer downloads, as well as uploads, content to its neighbors. Compared to download capacity, peers’ upload capacity is limited and often times the most prominent constraint of such systems, because most peers are connected via DSL and cable modem (even FiOS hosts have very asymmetric access). When multiple connections to neighbors contend for the limited upload capacity (say via TCP congestion control), each connection gets a fair share of the total capacity. Assuming the round-trip time (RTT) to neighbors are similar, the natural contention will then result in an implicit rate allocation – upload capacity are evenly divided among connections (call it even bandwidth allocation). As shown in the later part of the paper, however, even bandwidth allocation is not sufficient. Indeed, the focus of this paper is to design an explicit rate allocation algorithm, which, at a high level, continuously solves a global optimization problem and dictates accordingly how much bandwidth is allocated on 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. There are many possible rate allocations, which could all result in the same minimum server load. Hence, the second order objective is to minimize ISP-unfriendly traffic, while keeping the minimum server load unaffected.
- Maximize peer prefetching. When both above objectives are met, it is possible that peers still have spare upload bandwidth, which is especially true in a surplus mode. As shown in early work [8], such upload bandwidth should be utilized to allow peers download faster than real-time and cache future content. The so-called prefetching can greatly reduce server load, as well as ISP-unfriendly traffic. Hence, the third order objective is to maximize peer prefetching, while not affecting the first two.

3.1 Brief primer

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. Denote \( S_j \) as the set of all peers uploading to Peer \( j \) (Peer \( j \)'s upstream neighbors). Denote \( D_i \) as 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 the desirable streaming rate of Peer \( j \) in order to maintain smooth video playback. (In general, \( R_j \) could vary based on the amount of content in its local cache. For simplicity, however, let all \( R_j \)'s be the same for now, equal to the video bitrate \( R \).) It is clear that smooth video playback requires \( x_j \geq R \). Hence, 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. Peer \( j \) could vary based on the amount of content in its local cache. Hence, 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. Peer \( j \) could vary based on the amount of content in its local cache.

After the optimization, the minimum server load can then be computed, as \( U_0^{\text{min}} = \sum_j x_{0,j} \), where \( \{x_{0,j}\}_{j=1}^n \) is an optimal solution.

3.2 3-stage optimization

Now, we are ready to describe a rate allocation scheme, which achieves all the aforementioned objectives through a 3-stage optimization.

3.2.1 1st-stage – minimize server load

The rate allocation minimizing the server load can be realized as follows:

\[
\begin{align*}
\min & \sum_j x_{0,j} \\
\text{s.t.} & \sum_{i \in (0, D_j)} x_{i,j} = R_i, \\
& \sum_{j \in D_i} x_{i,j} \leq U_i, \\
& x_{i,j} \geq 0 \forall i, j (i, j \neq 0).
\end{align*}
\]

After the optimization, the minimum server load can then be computed, as \( U_0^{\text{min}} = \sum_j x_{0,j} \), 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 limit the upload capacity of the server to \( U_0^{\text{min}} \) from the first stage. 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 \( x_{i,j} \). It is clear that the link cost can take forms as simple as the sign function, where the cost is 0 for intra-ISP and 1 for inter-ISP traffic. But it could also be generalized to incorporate various levels of cost, e.g., using different costs to differentiate intra-POP and inter-POP traffic within the same ISP.

The rate allocation can be represented as follows:

\[
\begin{align*}
\min & \sum_{i,j} g_{i,j}(x_{i,j}) \\
\text{s.t.} & (2), (3), (4) \text{ and } \sum_j x_{0,j} \leq U_0^{\text{min}} \text{ (from 1st-stage)}. 
\end{align*}
\]

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

3.2.3 3rd-stage – maximize peer prefetching

In this stage, the key is to allow peers download more than their demand. Peer prefetching can then be maximized when all peers fully utilize their upload capacity. Hence, this objective is translated into minimizing the remaining of peers’ upload capacity. The first two objectives will not be affected, as long as we keep the minimum server load and ISP-unfriendly traffic limited to \( U_0^{\text{min}} \) and \( G^{\text{min}} \), respectively. Then, the rate allocation is as follows:

\[
\begin{align*}
\min & \sum_i (U_i - \sum_j x_{i,j}) \\
\text{s.t.} & (3), (4), (6) \text{ and } \sum_{i \in (0, D_j)} x_{i,j} \geq R \text{ (download could exceed demand)}, \\
& \sum_{i,j} g_{i,j}(x_{i,j}) \leq G^{\text{min}} \text{ (from 2nd-stage)}. 
\end{align*}
\]

3.2.4 Unifying three stages

The 3-stage optimization can easily be unified into a single optimization, as follows:

\[
\begin{align*}
\min & \alpha \sum_j x_{0,j} + \beta \sum_{i,j} g_{i,j}(x_{i,j}) + \gamma \sum_i (U_i - \sum_j x_{i,j}) \\
\text{s.t.} & \sum_{i \in (0, D_j)} x_{i,j} \geq R, \sum_{j \in D_i} x_{i,j} \leq U_i, x_{i,j} \geq 0 \forall i, j (i, j \neq 0).
\end{align*}
\]

It is not difficult to show that, if \( \alpha \gg \beta \gg \gamma \), then the solution to (10) will be equivalent to the 3-stage optimization. Obviously, both optimizations are easy to carry out with the existence of a central oracle, while not so much so if all peers strike distributedly and independently. Next, we will introduce an artificial utility function, which not only bears an intuitive meaning, but can also directly translate the centralized optimization into a utility-based optimization, from which a completely distributed solution is merited.

3.3 Utility-based optimization

3.3.1 Single utility function
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):

\[
f(x_j) = \begin{cases} 
  a \cdot x_j & \text{if } x_j \leq R \\
  a \cdot R + b \cdot (x_j - R) & \text{if } x_j > R 
\end{cases}
\]  

(11)

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

\[f(x) = \begin{cases} 
  a & \text{if } x \leq R \\
  b & \text{if } x > R
\end{cases}
\]

**Figure 2: Single Utility Function \(f(x_j)\).**

Here, \(a\) represents the value of getting rate from other peers up to streaming rate \(R\): every unit of rate from other peers increases the utility by \(a\). \(a\) 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 at least 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 hence peers 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 when possible to save for future use.

As a matter of fact, 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

\[
\max_{x_{i,j}} \sum_{j} f(x_j)
\]

(12)

s.t. \(\sum_{j \in D_i} x_{i,j} \leq U_i, \ x_{i,j} \geq 0 \ \forall \ i, j.\)

minimize server load and maximize peer prefetching.

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

\[
\min_{x_{i,j}} \sum_{j} a \cdot \max(0, R - x_j)
\]

(13)

s.t. \(\sum_{j \in D_i} x_{i,j} \leq U_i, \ x_{i,j} \geq 0 \ \forall \ i, j,\)

since \(a\) is a positive constant.

This is equivalent to

\[
\max_{x_{i,j}} \sum_{j} a \cdot \min(x_j, R)
\]

(14)

s.t. \(\sum_{j \in D_i} x_{i,j} \leq U_i, \ x_{i,j} \geq 0 \ \forall \ i, j.\)

Now, we rewrite the utility function (11) as

\[
f(x_j) = a \cdot \min(R, x_j) + b \cdot \max(0, x_j - R) = b \cdot x_j + (a - b) \cdot \min(R, x_j)
\]

(15)

It is clear that all the solutions to (12) using the proposed utility function (11) must satisfy \(\sum_{j \in D_i} x_{i,j} = U_i\) (thus maximize peer prefetching). This can be shown through contradiction. Hence (12) is equivalent to

\[
\max_{x_{i,j}} \sum_{j} b \cdot U_j + \sum_{j} (a - b) \cdot \min(R, x_j)
\]

(16)

s.t. \(\sum_{j \in D_i} x_{i,j} = U_i, \ x_{i,j} \geq 0 \ \forall \ i, j.\)

\[
\sum_{j} b \cdot U_j \text{ is a constant and } a - b > 0, \text{ thus (16) is equivalent to}
\]

\[
\max_{x_{i,j}} \sum_{j} a \cdot \min(x_j, R)
\]

(17)

s.t. \(\sum_{j \in D_i} x_{i,j} = U_i, \ x_{i,j} \geq 0 \ \forall \ i, j.\)

The only difference between (14) and (17) is the constraint on \(\sum_{j \in D_i} x_{i,j} \cdot \min(x_j, R) = \min(\sum_{j \in D_i} x_{i,j}, R)\) is nondecreasing in \(x_{i,j}, \forall \ i, j.\) Thus, the solutions to (17) is a subset of those to (14).

In summary, the solutions to (12) is a subset of those to (14) and minimize server load and maximize peer prefetching.

Now we are ready to explain the connection between the utility optimization (12) and the unified 3-stage optimization (10). In fact, ISP-unfriendly link cost aside (or letting \(g_{i,j}(x_{i,j}) = 0\)), it is not difficult to show that these two optimizations are equivalent and, in addition, \(a = \alpha\) and \(b = \gamma\). Compared to the unified 3-stage optimization (10), the utility function introduced here bears a much more intuitive meaning. It is straightforward to see that with the piece-wise utility function, a peer would not help a neighbor prefetch if it could instead use the upload bandwidth to satisfy other neighbors’ basic streaming demand first. Hence, minimizing server load will have a higher priority than prefetching.

### 3.3.2 Costs on ISP-unfriendly traffic

Recall that, to reduce ISP-unfriendly traffic, we associate a link cost with ISP-unfriendly traffic, i.e. the cost for Peer \(i\) to upload to Peer \(j\) at rate \(x_{i,j}\) is \(g_{i,j}(x_{i,j})\). The overall utility optimization problem then becomes

\[
\max_{x_{i,j}} \sum_{1 \leq j \leq n} \left( f(x_j) - \sum_{i \in D_j} g_{i,j}(x_{i,j}) \right)
\]

(18)

s.t. \(\sum_{j \in D_i} x_{i,j} \leq U_i, \ x_{i,j} \geq 0 \ \forall \ i, j.\)

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 (18) 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 (12), solutions to (18) 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: Among all the ISP-unaware solutions that minimize server load, a subset of them requires the least ISP-unfriendly traffic. Let $\{\bar{x}_{i,j}\}$ be one of these solutions. Let $\{\bar{x}_{i,j}\}$ be a solution for (18). Then by definition,

$$\sum_{1 \leq j \leq n} (f(x_j) - \sum_{i \in S_j} g_{i,j}(\bar{x}_{i,j})) \leq \sum_{1 \leq j \leq n} (f(x_j) - \sum_{i \in S_j} g_{i,j}(\bar{x}_{i,j}))$$

$$\Rightarrow \sum_{1 \leq j \leq n} (f(x_j) - f(\bar{x}_j)) \leq \sum_{1 \leq j \leq n} \left( \sum_{i \in S_j} (g_{i,j}(\bar{x}_{i,j}) - g_{i,j}(\bar{x}_{i,j})) \right)$$

Using (15),

$$\text{LHS} = \left( b \sum_j x_j + (a-b) \cdot \min(R, x_j) \right) - \left( b \sum_j \bar{x}_j + (a-b) \cdot \min(R, \bar{x}_j) \right)$$

$$= b \cdot \left( \sum_j x_j - \sum_j \bar{x}_j \right) + (a-b) \cdot \left( \min(R, x_j) - \min(R, \bar{x}_j) \right)$$

As stated earlier, $\sum_j x_j = \sum_j U_j$, thus $\sum_j x_j \geq \sum_j \bar{x}_j$. $\min(R, x_j) - \min(R, \bar{x}_j)$ is the increase in server load due to incorporating ISP-friendliness. Thus

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

Due to incorporating ISP-friendliness.

$$\text{RHS} = c \cdot \sum_{1 \leq j \leq n} \left( \sum_{i \in S_j \cap I_j} x_{i,j} \right) - c \cdot \sum_{1 \leq j \leq n} \left( \sum_{i \in S_j \cap I_j} \bar{x}_{i,j} \right)$$

where $I_j$ denotes the subset of Peer $j$’s neighbors in the same ISP as $j$. Here, $\sum_{1 \leq j \leq n} (\sum_{i \in S_j \cap I_j} x_{i,j})$ is nothing but the total ISP-unfriendly rate using allocation $\{x_{i,j}\}$. Hence

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

Combining (19) and (20), we get

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

Intuitively, setting $a < c$ will result in a pure ISP-friendly solution that eliminates ISP-unfriendly traffic completely. On the other hand, setting $b > c$ will result in full utilization of peers’ upload bandwidth at all time.

3.4 Distributed solution

Now that we have formulated the rate allocation problem as a convex optimization problem with linear constraints, it is straightforward to apply classical distributed solutions [10]. In particular, we adopt the feasible steepest descent algorithm: $x_{i,j}$ is first initialized to 0 and then updated at each step as follows:

$$\begin{align*}
\dot{x}_{i,j} &= \Delta \cdot \left( \frac{\partial}{\partial x_{i,j}} f(x_j) - \frac{\partial}{\partial x_{i,j}} g_{i,j}(x_{i,j}) \right), \\
x_{i,j} &= [x_{i,j} + \dot{x}_{i,j}]^+ 
\end{align*}$$

where $[\cdot]^+$ means $l_2$ projection onto a feasible set, which guarantees convergence for this problem [10].

Due to the simple symmetric linear constraints, this step can be easily implemented as follows:

$$\text{while} (\sum_{1 \leq j \leq n} x_{i,j} > U_i) \{$$

$$N_i = \text{number of neighbors with } x_{i,j} > 0;$$

$$\text{foreach} (j)$$

$$\text{if} (x_{i,j} > 0)$$

$$x_{i,j} = x_{i,j} - \min(x_{i,j}, U_i - \sum_{1 \leq j \leq n} x_{i,j}) / N_i;$$

$$\}$$

Note that in order for Peer $i$ to carry out the update step, the only external information required is the aggregate received bandwidth at Peer $j$. This information can be easily piggybacked in Peer $j$’s packet requests.

3.5 Practical considerations

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 and, more importantly, eliminates ISP-unfriendly prefetching completely. 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 $\Delta R$, as qualitatively shown in Fig. 3. The implication is that we potentially allow moderate ISP-unfriendly prefetching, but bound the amount by $\Delta R$ per peer in the worst case. We use $\Delta R = 0.1R$ and a Deg. 3 polynomial for the curve. Second, we change the link cost function $g(\cdot)$ from linear to a flat quadratic one such that the optimization is strictly concave, which allows faster and steadier convergence. We set $g(\cdot)$ such that $\frac{\Delta}{R} g(x) = c + ex$ with $e \ll R$. Note that, by making above modifications, the final solution will deviate from the optimal one directly derived from (10). We will examine the issue of sub-optimality in detail in a later section.

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 $\text{buffer level} \cdot R$. 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 $\frac{1}{4} R + \sum_{i \in S_j} x_{i,j}$. 

![Figure 3: Modified Utility Function.](image-url)
4. PACKET-LEVEL ALGORITHM

So far, the rate allocation stays at an ideal network flow level, i.e. upload bandwidth is treated as fluid and can be divided/utilized as finely as desired. In this section, we describe a practical packet-level algorithm that conforms nicely to the flow-level one (see Sec. 5.1 for detailed results).

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. 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 (23) and compute 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 (peers need not synchronize their intervals). The number of tokens it 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 (23). Note that $U_i$ and $x_{i,j}$ need not be integers, as partial tokens 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 $x_{i,j}$ tokens from the neighbor.

Request rejection: Peer $i$ should turn down requests that it is unlikely to fulfill, so that the requester does not wait unnecessarily. For this purpose, peers include time to playback deadline in the request of each packet ($D_m$). Suppose a neighbor Peer $j$ has accumulated token level $L_j$ (Peer $i$ has this information). Peer $i$ will turn down the request for Piece $m$ from $j$ if by the time the packet expires, Peer $j$ still wouldn’t have accumulated enough token, i.e.

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

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. This is because the only additional information that peers have to transmit to each other is 1) their aggregate received bandwidth and 2) the playback deadline for each packet. Peers piggyback this information in their packet requests with almost no overhead.

5. EVALUATION

In this section, we evaluate the proposed packet scheduling algorithm using a discrete-time packet-level simulator that simulates the peer-assisted VoD system as described in Sec. 2. We first demonstrate that the proposed packet-level algorithm converges closely to the solution of the fluid-level optimization problem (12). 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 the even bandwidth allocation. 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 a little more than 10 million users total. These traces contain clients’ public IP addresses, as well as their download bandwidth. We map these IP addresses to AS numbers [9] and find the relationship among ASes using CAIDA’s data [11]. 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 [8], and then quantize it to 128, 384, 512 and 768 kbps for the discrete-time simulator. For all our simulations, data packet size is 4KB.

5.1 Convergence behavior

We first demonstrate that the packet-level algorithm converges closely to the solution of the optimization problem (12). Using distributions obtained from the MSN traces, we simulated peers from the 5 ASes with the most users. At Time 0, there are no peers in the system. Peers start joining the system at Time 0 following a Poisson arrival process with arrival rate 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 (12) 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

\[
\text{percentage difference} = \frac{\text{packet-level result} - \text{fluid-level centralized result}}{\text{fluid-level centralized result}} \times 100\%.
\]

We compute the average percentage difference over time 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.

![Convergence behavior of proposed packet-level algorithm](image)

**Figure 4:** Convergence behavior of proposed packet-level algorithm. Peers have a Poisson arrival process of rate 1.2 peers/second. Peers watch the entire video (300 second) and then leave right away. (a) Number of peers in the system over time. (b) Server load over time. (c) Amount of ISP-unfriendly traffic over time. (d) Percentage discrepancy between the packet-level simulation and the centralized fluid-level solution as peer arrival rate varies between 1.2 and 4.8, i.e., the average number of peers in the system varies between 360 and 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\(^1\) in reducing both server load and ISP-unfriendly traffic;
3. given an ISP-friendly topology, the proposed algorithm can provide significant additional reduction in ISP-unfriendly traffic.

We use synthetic traces here 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 vary the coefficient of the linear term in the link cost function (c) and plot the server load versus the amount of ISP-unfriendly traffic using both ISP-unaware and ISP-friendly topology (blue solid lines in Fig. 5 (a) and (b) respectively). As expected, as we increase c, ISP-unfriendly traffic is reduced at the cost of increased server load. If we are able to obtain knowledge of the desirable server load and ISP-unfriendly traffic ratio or their relative cost, we could further choose which operating point to operate at by adjusting the link cost parameter.

The red single data point in Fig. 5 (a) and (b) is the result of using even bandwidth allocation. We can see that all but one operating 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.

We mimic the MESH platform and generate ISP-friendly topologies in the following way. As described earlier, when a peer joins a streaming session, it tries to make a certain number of connections (8 in our case) with upstream peers. Among them, it tries to make 70% of the connections with peers in the same or peering ASes\(^2\). 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 will then query the tracker periodically to see if additional upstream peers in the same or peering ASes become available\(^3\).

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 KB to 150,000 KB, an reduction of 40%, at the cost of approximately 500 KB 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 the strictly convex link cost function we use to ensure a unique global optimum, there is 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 proposed algorithm and the 3-stage optimization is larger when the server load needs to be optimized, i.e., c is small. This is because when the ISP-unfriendly link cost is small,\(^1\)

---

\(^1\)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.

\(^2\)This can be done either by the tracker or by peers themselves using the IP-AS map followed by AS relationship tables.

\(^3\)This could happen if an upstream peer in the same or peer-ing AS has freed up an upload slot due to peer leaving.
the quadratic term of the link cost function \((cx^2 + cx)\) becomes more dominant. This means that traffic within the same or peering ASes becomes as expensive as ISP-unfriendly traffic as this amount of traffic grows. This will suppress traffic within the same or peering ASes to some extend, thereby increasing ISP-unfriendly traffic. This cannot be avoided unless we drop the quadratic term of the link cost, which causes fluctuation of convergence. In fact, without the quadratic term, when there is peer dynamic, the iterative algorithm may never converge. It is an interesting future topic to reduce the gap when server load needs to be optimized while maintaining good convergence behavior. In the rest of the section, we fix \(c\) to be half of the slope of the steeper part of the utility function, i.e. \(c = \frac{a}{2}\), which corresponds to the fourth right most point on the curve.

### 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 with very different population distribution over a period of 24 hours as shown in Fig. 6. 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. As stated before, we look up AS relationships using CAIDA’s data.

For the gold trace, the video is 456 seconds long. We quantize its video bit rate to 512 kbps. Fig. 7 (a) and (b) show the plots the server load (in KB) and the amount of ISP-unfriendly traffic (in KB) over time for all four systems. Fig. 7 (c) and (d) show the the corresponding CDFs over the whole day.

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 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.

### 6. Can Topology Building Be 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. Up to now, we have adopted a simple heuristic
to build ISP-friendly topologies, which appears to be quite effective. In this section, we present some preliminary investigation on the benefit of topology building.

In particular, we would like to answer the following question: if we are allowed to 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 question yet, our study did provide some intuition on why a centralized topology building guide may not be sufficient.

### 6.1 The iTacker approach

We adapt the iTacker approach from the recently proposed P4P framework [13] and evaluate a similar topology building mechanism in our VoD scenario. The P4P framework advocates a flexible and light-weight portal, through which ISPs can provide explicit information and guidelines to P2P apps. For example, ISPs can offer network topology information so that locality becomes explicit and extensive network probing/inference can be avoided. ISPs can also define traffic guidelines, when respected by P2P apps, can control traffic both inside ISP (between POPs) and cross ISP boundaries. To be specific, ISPs can define a bandwidth cap on inter-ISP links and instruct P2P apps not to exceed the cap. At the center of the proposed P4P framework, iTacker is the key co-ordinating unit. In short, iTacker takes information from both ISPs (e.g., network topology, bandwidth cap, etc.) and P2P apps (e.g., peers’ upload/download bandwidth, IP address, etc.) as input. It solves a global optimization problem to strike a balance between ISPs’ objective (e.g., limit cross-ISP traffic) and P2P apps’ objective (e.g., speed up download). The outcome of the optimization is then used to guide topology building.

Specifically, the iTacker optimization consists of two parts: 1) modeling peer sharing efficiency, following the method proposed by Qiu et al. [12]; 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 eventually guides topology building for future peers. Here is a concrete example. Say there are three ISPs (A, B and C). All peers in ISP A download either from other peers in A or from B/C. Suppose the outcome of the global optimization is that the ratio among total traffic inside ISP A, total traffic between A-B, total traffic between A-C is 7:2:1. The ratio suggests that each peer in ISP A download 70% from other peers in A, 20% from B and 10% from C. 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 iTacker 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. Instead, we compute the efficiency exactly, by taking a snapshot of the complete topology. The 3-stage global optimization is executed on the complete topology (the peer population is manageable in our simulation). 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.

### 6.2 Evaluation

We simulated one day of the silver MSN trace (see Sec. for more details about the trace). In Fig. 9, we show the CDF of server load and ISP-unfriendly traffic using the following 4 systems:

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

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

The iTacker 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. For example, if the ratio among total traffic inside ISP A, total traffic between A-B, total traffic between A-C is 7:2:1, then on average, peers in ISP A are getting 20% rate from ISP B. This does not mean that every peer in ISP A gets exactly 20% rate from ISP B. Therefore if every peer in ISP makes 20% connections with peers in ISP B, the resulting topology is typically suboptimal.

Further, the 3-stage optimization does not have a unique global optimum. As peers come and go, the optimization result may fluctuate among different optimums that correspond to completely different rate allocations on each link and yield different connection ratios. In fact, we did observe large fluctuations in the connection ratio as new peers join. While the new peer can follow the newest iTacker result in making the connections, it is impossible for all previous peers to also change their connections to conform to the new result. This is another reason for the sub-optimality of the resulting topology.

In practice, it is clearly not scalable for the iTacker to run the 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. It is consistent with what we showed earlier that given a suboptimal topology, the proposed algorithm is very effective in improving the performance.

![Figure 9: Effect of using a sophisticated topology-building mechanism. (a) and (b) show the server load and ISP-unfriendly traffic CDF over a day using the silver MSN trace.](image)

7. RELATED WORK

**P2P and ISP:** Many measurement-based studies have argued that peer-to-peer or peer-assisted applications can indeed become ISP-friendly. Saroiu et al. [14] collected in-bound/outbound traffic of a large university and showed that there are lots of redundancy in P2P traffic. Gummadi et al. [15] studied Kazaa traffic and showed that ISPs’ external traffic can be greatly reduced by incorporating locality-awareness. Karagiannis et al. [16] studied BitTorrent traffic and showed that locality-aware P2P delivery solutions can significantly alleviate the induced cost at the ISPs. Huang et al. [8] 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 [17] estimated the cost of deployment and operation of a VoD service in a CDN with several hierarchical levels, and figured out the minimal cost and 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. It be an explicit mechanism, as simple as biased neighbor selection [18], or as sophisticated as the iTracker approach [13]. It can be extended to take into account multinet topology, from subnets, POPs to ISPs [19]. The topology building mechanism can also be implicit. As shown in the recent study of UUSee [20], 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 [21, 22, 23], has been applied to P2P applications recently. All the studies [24, 25, 26] 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 pre-fetching, besides QoS and ISP-friendliness.

8. CONCLUSIONS

We have presented and analyzed a distributed packet-level rate allocation algorithm using an optimization framework that can significantly reduce ISP-unfriendly traffic without much impact on the server load. The proposed algorithm is evaluated using both synthetic and real-world traces. We have shown that, even though an ISP-friendly topology can be quite effective, using the proposed algorithm on top will bring significant additional gain. In future, we would like to integrate the rate allocation algorithm into the MESH platform and understand its behavior, as well as ultimate benefits, in the real-world.

9. REFERENCES