Optimal TCP-friendly Rate Control for P2P Streaming: An Economic Approach

Jinyao Yan[1,2], Martin May[3], and Bernhard Plattner[1]

[1] Computer Engineering and Networks Laboratory, Swiss Federal Institute of Technology, ETH Zurich, CH-8092, Switzerland
jinyao,plattner@tik.ee.ethz.ch
[2] Computer and Network Center, Communication University of China, 100024, Beijing, China
jyan@cuc.edu.cn
[3] Thomson Paris Research Lab, Thomson, France
martin.may@thomson.net

Abstract. TCP and TCP-friendly rate control protocols, designed for unicast, do not take neighbor connections into account in P2P networks. In this paper, we study the topic of distributed and optimal rate control for scalable video streams in P2P streaming applications. First, we propose a fully distributed and TCP-friendly network analytical model for rate control and formulate an optimization problem to maximize the aggregate utility for the P2P streams. In the model, we further extend the definition of TCP-friendliness for P2P network. Second, we propose a shadow price-based distributed algorithm for P2P Streaming that solves the optimization problem. Finally, we evaluate the performance of the proposed algorithm in terms of streaming quality and messaging overhead. Extensive simulations show that the proposed algorithms generate very small overhead and that they are optimal in terms of overall quality for scalable streams.

1 Introduction

Multimedia streaming over Internet has been a hot topic both in academia and in industry for two decades. Since the emergence of peer-to-peer architectures, there has been significant interest in streaming applications over peer-to-peer overlay networks [4] [5] [6]. P2P streaming does not require support from Internet routers compared to IP layer multicast, therefore, it is easy to deploy and also scaleable to very large group sizes.

Rate control is one of key technologies in multimedia communications to deal with the diverse and constantly changing conditions of the Internet. TCP, the dominant congestion protocol designed for client-server unicast communication in the Internet, is also used as rate/congestion control protocol in most of P2P streaming systems. However, using TCP for P2P streaming also has some disadvantages. Streaming applications are usually sensitive to delay. TCP adopts an Additive-Increase Multiplicative-Decrease (AIMD) strategy to react to packet losses and retransmits packets lost in congestion, therefore it introduces long delay and jitters and hence is not well suited for real-time streaming applications. By contrast, UDP is an unreliable and connection-less protocol without integrated rate/congestion control. Without congestion control however, non-TCP traffic can cause starvation or even congestion collapse to TCP traffic [12]. To

overcome the disadvantages of TCP and to handle competing dominant TCP flows in a friendly manner, TCP-Friendly Rate Control (TFRC) was introduced for streaming applications in [1] .

On the other hand, existing P2P streaming systems using rate control send data flows without considering the structure of the overlay tree (e.g., TCP in [5] and TFRC in [4]). TCP and TFRC/UDP, both being client/server (unicast) protocols, prevent applications from either overloading or under-utilizing the available bandwidth of their *local* connections. Moreover, they do not take neighbor connections and the quality of media stream into account.

With the goal to optimize the aggregate utility (video-quality) for P2P streaming application, we develop a fully distributed and optimal TCP-friendly rate control model in Section 2 and propose a shadow price-based distributed algorithm to solve the optimization problem in Section 3. The proposed algorithm is distributed with very small messaging overhead to allow P2P streaming systems to scale up to very large sizes while being TCP-friendly to coexisting traffic outside of the P2P session. We further extend the definition of TCP-friendliness to P2P network. With the help of extensive simulations, we evaluate the performance of the proposed TCP-friendly algorithm in terms of streaming quality and messaging complexity in Section 4. In Section 5, we discuss the implementation issues and conclude the paper.

2 Network Model and Rate Control Problem Formulation

2.1 Network model

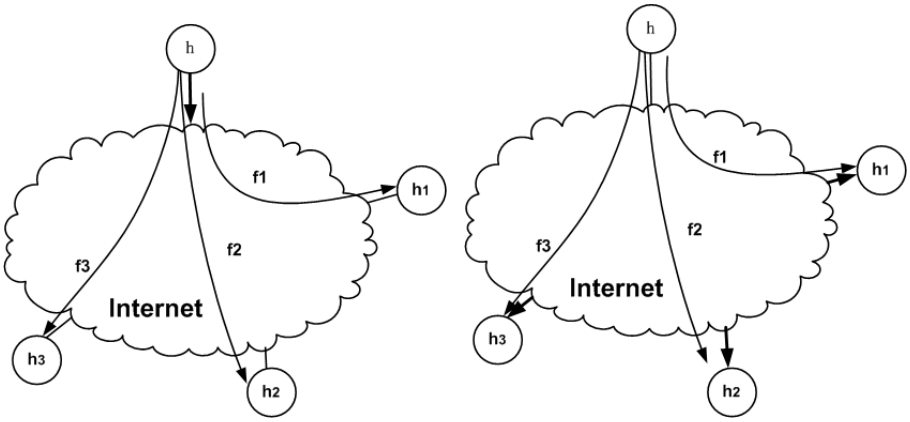
A large number of approaches have emerged in recent years for P2P streaming systems ([6] and its references). The vast majority of systems to date are tree-based P2P streaming, where peers are organized in trees to deliver data. Consider a P2P overlay tree of $n+1$ end hosts, denoted as $H = \{h_0, h_1, \ldots, h_n\}$. End host h_0 is the source of the P2P multicast channel. The structure of the overlay tree is given by the used P2P streaming approach. Non-leaf nodes are forwarding streaming data to its children and are able to scale-down the streams, fulfilling the constraint of the flow data. For our model, we assume that streams are fine-grained scalable [7]. The P2P streaming channel consists of n end-to-end unicast flows, denoted as $F = \{f_1, \ldots, f_n\}$. Flow f_i is the flow that terminates at h_i. Flow $f_i \in F$ has a rate x_i. We collect all the x_i into a rate vector $x = (x_i, i = 1, 2..., n)$. We denote $U(x_i)$ as the utility of flow f_i, when f_i transmits at rate x_i. We assume that $U(x)$ is strictly increasing and concave, and twice continuously differentiable. We measure the utility $U(x)$ for streams in section 4. F'_h is the set of flows sent from h. If a host h_i is the destination of a flow f_i and the source of another flow $f'_i \in F'_{h_i}$, then f'_i is the child flow of f_i, denoted as $f_i \rightarrow f'_i$. We denote h' as the child of h and h^p is the parent node of h, *i.e.*, $h^p \rightarrow h \rightarrow h'$. Let us define [16],

Definition 1. *A rate control algorithm is **TCP-friendly for P2P multicast**, if and only if the coexisting TCP traffic outside of the P2P channel achieves not less throughput than what it would achieve if all flows of the overlay channel were using TCP as rate control algorithm.*

Based on the fact that the backbone links of today's Internet are usually overly provisioned [13], we assume that the bottleneck of a unicast flow f_i only appears at access links, namely upload link $l_u(f_i)$ and download link $l_d(f_i)$.

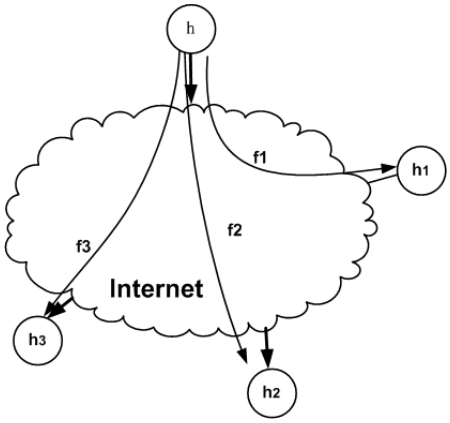
Assumption 1 *Access links (download and upload links) of end hosts are the only bottleneck links of a unicast path.*

Moreover, three possible bottleneck links between host h and its children of a subtree ($h_1, h_2, h_3 \in H'_h$) are presented in Fig. 1.



(a) Bottleneck link at the upload link and host h (Case $I1$)

(b) Bottleneck links at download links and host h (Case $I2$)



(c) Bottleneck links at the upload link and download links, and host h (Case $I3$)

Fig. 1. Locations of Bottleneck links (Note: Bold lines are bottleneck links in the arrow direction. All links are directed)

Proposition 1. *Only sibling flows in the tree may share bottlenecks.*

Here, sibling flows are flows sent from the same end host h, i.e., all $f \in F'_h$ are sibling flows. Proposition 1 is straightforwardly provable with the locations model of bottleneck links shown in Fig.1. Therefore, non-sibling flows have independent bottleneck links and the overlay tree can be *fully decomposed* into subtrees with independent bottleneck links.

Let t_i be the TCP-friendly available bandwidth for the unicast flow f_i at the bottleneck links determined by end-to-end TFRC algorithm. We measure all t_i for flows $f_i \in F$. Hence, we get the TCP-friendly available bandwidth for P2P multicast channel at bottleneck links, $c_{l_d(f_i)}$ and $c_{l_u(f_i)}$. For cases $I2 \cup I3$ where bottleneck links locate at download links $l_d(f_i)$: $c_{l_d(f_i)} = t_i$. For cases $I1 \cup I3$ where bottleneck links locate at upload links $l_u(f_i)$: $c_{l_u(f_i)} = \sum_{f_i \in F(l_u)} t_i$.

For each bottleneck link l, $F(l) = \{f \in F \mid l(f) = l\}$ is the set of flows in the channel that pass through it and $l(f)$ is the bottleneck link through which f goes.

We define the constraints for rate control as follows: Flow rate of f_i should not exceed the TCP-friendly available bandwidth $c_{l_d(f_i)} = t_i$ when the bottleneck link locates at download link of h_i. On the other hand, the sum of all flow rates in one direction and the same channel that go through the upload link of h_i should not exceed $c_{l_u(f_i)} = \sum_{f_i \in F(c_{l_u})} t_i$, when the bottleneck link is at the upload-link. Therefore, coexisting TCP traffic outside of P2P channel obtain no less throughput than what they would achieve if all streams would use TFRC. Formally, such TCP-friendly available bandwidth constraint for P2P streaming rate control is expressed as follows:

$$\sum_{h_i^p \to h_i} x_i \le c_{l_u(f_i)} = \sum_{h_i^p \to h_i} t_i, \quad \forall h_i^p \in I1 \cup I3. \tag{1}$$

$$x_i \le c_{l_d(f_i)} = t_i, \quad \forall h_i^p \in I2 \cup I3 \tag{2}$$

Moreover, the downstream rate is constrained by the upstream rate, namely, if $f_i \to f_j$ then $x_j \le x_i$. We define the data constraint or flow preservation $F \times F$ matrix B. $B_{f1,f2} = -1$,if $f2 \to f1$,*i.e.*, $f1 = f2'$; $B_{f1,f2} = 1$, if $f1 = f2$, and $f1$ has a parent flow; Otherwise $B_{f_1 f_2} = 0$. Hence, given the P2P distribution tree, the data constraint can be formalized as follows:

$$B \cdot x \le 0 \tag{3}$$

A summary of the notations used in the model can be found in Table 1.

2.2 Problem Formulation

Our objective is to devise a distributed rate control algorithm that maximizes the aggregate utility, *i.e.*, the overall video quality of all streams in the P2P streaming channel:

$$\max \sum_{i=1,2...n} U(x_i) \tag{4}$$

Table 1. Summary of Notations in the Model

Notation	Definition
$h \in H = \{h0, h1, \ldots, hn\}$	End Host
$h^p \rightarrow h \rightarrow h' \in H'_h$	h^p is the parent node of h, h' is a child of h
H'_h	Set of child of h
$f \in F = \{f1, f2, \ldots, fn\}$	Unicast flow in P2P streaming channel
$f_i \rightarrow h_i$	Flow f_i terminated at h_i
f_h	Flow terminated at h
$x = (x_i, i = 1, 2, ..., n)$	Flow rate set of $fi \in F$
$l \in \Gamma = 1, 2, \ldots, L$	Bottleneck Link l (download link or upload link)
$c_l \in C, l \in \Gamma$	TCP-friendly available bandwidth for the channel at bottleneck link l
$fi \rightarrow fi' \in F'_{hi}$	fi' is a child flow of fi
F'_{hi}	Set of flow sent from hi in the channel
$l_u(fi) \in \Gamma$	The upload link that fi goes through
$l_d(fi) \in \Gamma$	The download link that fi goes through
$F(l)$	Set of siblings flows that go through bottleneck link l
$B = (B_{fi,fj})_{F \times F}$	Data constraint matrix
t_i	TCP-friendly available bandwidth for unicast for fi at bottleneck
$U(x_i)$	Utility Function of streams at rate x_i

fulfilling the following constraints:

$$\begin{cases} \sum_{h_i^p \rightarrow h_i} x_i \leq c_{l_u(f_i)} = \sum_{h_i^p \rightarrow h_i} t_i, & \forall h_i^p \in I1 \cup I3 \\ x_i \leq c_{l_d(f_i)} = t_i, & \forall h_i^p \in I2 \cup I3 \\ B \cdot x \leq 0 \end{cases}$$

3 Algorithm

In this section, we propose a distributed rate control algorithm for P2P streaming based on a shadow price concept that solves the convex optimization problem (4). Compared with the dual approaches proposed in [3][16][2], our primal algorithm is a feasible direction algorithm [9] which is applied to the original problem (primal problem) directly by searching the feasible region in the direction of improving the aggregate utility for an optimal solution. Please note that the proposed primal algorithm is different from the primal algorithm introduced by Kelly' in [8] or other penalty algorithms. Thanks to our fully distributed model, solving the optimization program (4) directly requires the coordination among those sibling flows only sharing bottleneck links. In order to find the direction for improving the aggregate utility, we define,

Definition 2. *The **Data shadow price** of a flow is the change in the aggregate utility of the flow itself and its subtree by relaxing the data constraint by one unit (a small move).*

By moving the bandwidth of the bottleneck link from children flows with lower shadow prices to children flows with higher shadow prices, the aggregate utility is improved.

We call a flow a data constrained flow when it is actively constrained by its parent flow, *i.e.*, $x_i = x_j$ (where $f_j = f'_i$); otherwise it is a data unconstrained flow, *i.e.*, $x_i > x_j$ (where $f_j = f'_i$) and actively constrained by the bottleneck link.

For a data constrained leaf flow f_i, its data shadow price is:

$$p_{f_i} = \Delta U(x_i)/\Delta x_i = U'(x_i) \tag{5}$$

For a data constrained intermediate flow f_i:

$$p_{f_i} = U'(x_i) + \sum_{f_j \in F'_{h_i}} p_{f_j} \tag{6}$$

When a flow is not constrained by its parent flow, its data shadow price is zero ($p_{f_i} = 0$). For example, all dashed flows in Fig. 2 have data shadow price zero. We call a node data constrained node (see Fig. 2(b)) when its incoming flow is a data constrained flow, otherwise it is a data unconstrained node (as in Fig. 2(a)). Each end host h_i is assumed to be capable of communicating with neighbors, to determine the locations of bottleneck links, t_j (where $f_j \in F'_{h_i}$) and to compute and adapt the sending rate for each flow f'_h (*i.e.*, sender-based flow).

We present the algorithm of an intermediate peer in Table 2. We choose the TCP-friendly available rate of unicast flows as the initial rate, *i.e.*,$x_j(0) = t_j$.The algorithm purely depends on the coordination of end nodes. Each node receives the data shadow prices from its children nodes for children flows at each step. The algorithm ***(i)*** reallocates the bandwidth of the bottleneck link with stepsize γ ($\gamma > 0$) from children flows with lower shadow prices to children flows with higher shadow prices such that the TCP-friendly available bandwidth constraints are not violated and flows with higher data shadow price get more bandwidth; and ***(ii)*** the algorithm obtains a better rate allocation after each step with an improved aggregate streaming quality. Thus, we have the following theorem,

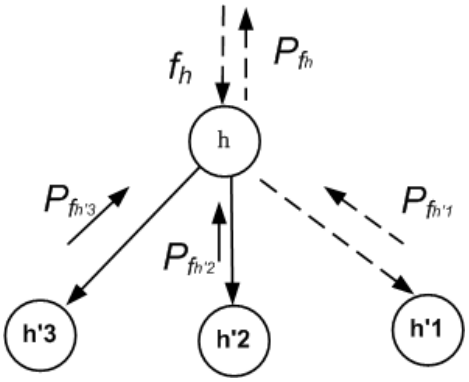
Theorem 1. *For any P2P multicast streaming session, the rate allocation by the algorithm in Table 2 with sufficient small stepsize γ($\gamma > 0$) converges to the optimal allocation.*

Proof. For the subtree rooted at end host h_i, each allocation generated in the algorithm process is feasible and flows with higher data shadow price get more bandwidth. Therefore, the value of the aggregate utility of the subtree $\sum U(x_j(t)) < \sum U(x_j(t+1))$ improves constantly. Given the receiving rate f_i, as there is a limit for the aggregate utility of the subtree, the algorithm will finally converge to a maximum. For a convex optimization problem, the convergent rate allocation is the global maximum (the optimality) of the subtree(Chapter 11.1 in [9]).By each subtree converging to the optimal allocation for a given receiving rate iteratively, the optimal allocation of the entire multicast tree with its root at h_0 will be eventually reached. □

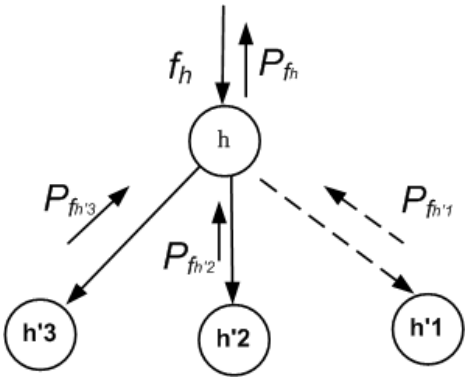
Unlike the fluctuating convergence procedure in dual approaches [3][16], the feasible direction algorithm steadily converges to the optimal allocation .

Table 2. Algorithm of End Host h_i

Initialization
Sending data with the TFRC unicast rate for each flow.
Update the data shadow price from children
Get shadow price for children flows f_i': $p_{f_j}(t), f_j \in F'_{h_i}$
Compute the median shadow price of children flows: $pj_{med}(t)$
Update information from the parent node
Get the flow rate $x_i(t)$ and the data constraint information
Re-allocate the rate among the children flows for $f_j \in F'_{h_i}$
for $h_i \in I1 \cup I3$
if $p_{f_j}(t) > pj_{med}(t)$
$x_j(t+1) = x_j(t) + \gamma$
else if $p_{f_j} < pj_{med}$
$x_j(t+1) = x_j(t) - \gamma$
end if
end if
Update Data Shadow Price to parent node
For data constrained node h_i : $p_{f_i}(t+1) = U'(x_i)$
for $f_j \in F'_{h_i}$:=1 to n do
if $x_j(t) \geq x_i(t)$
$x_i(t+1) = x_j(t); p_{f_i}(t+1) = p_{f_i}(t+1) + p_{f_j}(t)$
else $x_{f_j}(t+1) = x_{f_j}(t)$
end if
For data unconstrained node h_i: p_{f_i}=0
Send $p_{f_i}(t+1)$ up to parent node h_i^p
Update stream rates and inform the children
for $f_j \in F'_{h_i}$
Stream media to child j with updated rate $x_j(t+1)$
Update the data constraint information and $x_j(t+1)$ to h_j



(a) data unconstrained node h



(b) data constrained node h

Fig. 2. Nodes and flows in the algorithm (dashed line means data unconstrained flow, constrained flow otherwise. $h'_1, h'_2, h'_3 \in H'_h$)

4 Performance Evaluation

4.1 The utility function of scalable streams

The utility function used in [3] was $U(x_i) = \ln(x_i)$, which did not reflect the application quality of video streams. To tailor the utility function to the application quality, we use the rate-distortion function as the utility of our algorithm for each flow. The classic rate-distortion function for Gaussian distribution video source with mean $\mu = 0$ and variance σ^2 [15] is ,

$$D(x_i) = \sigma^2 \cdot 2^{-\alpha x_i} \tag{7}$$

We decided to use MPEG-4 fine-grained Scalable video (FGS) steams [7] in our performance evaluation, due to its ability to be sent at any given rate determined by a rate control algorithm at server side or any intermediate peer in the tree.

Table 3. Measurement of Rate-distortion function for streams

Video streams	a	σ^2	**Fitting Goodness(**$SSE/sum(D)$**)**
Forman	-0.8625	100.915	0.0355
Akiyo	-1.728	38.827	0.0970
Mobile	-0.4917	256.711	0.0656
Highway	-1.514	41.041	0.0611
Tempete	-0.6177	167.963	0.0728
Container	-1.098	82.196	0.0967

To measure the quality function of FGS coded P2P streams, we first use the Microsoft MPEG-4 software encoder with FGS functionality to encode the stored raw video streams. Then, we cut the corresponding FGS enhancement layer at the increasing and equally spaced bit rates (step size = 100kbps). For each compressed and cut bit-stream, we specify the distortion D after decoding. Subsequently, we generate the rate-distortion curve of the FGS video stream using these sample points and finally we estimate the parameters in the classic video rate-distortion function that fit the rate-distortion traces [14]. We compute these parameters values for video sequences and keep them constant throughout the entire streaming process. Parameters we measured for some typical streams are presented in Table 3. All streams measured are CIF format, 30 fps and 300 frames in length. A value of $SSE/sum(D)$ closer to 0 indicates a better fit, where SSE is the sum of squares due to error and distortion D is measured by the average MSE of a truncated video sequence.

We use the utility (video quality) function for *Forman*(CIF, 30fps, 300frames) in the experiments:

$$U(x_i) = -D(x_i) = -100.915 * 2^{-0.8625x_i} \tag{8}$$

where $D(x_i)$ stands for the distortion of the stream and *mbit/s* is used as unit for streaming rate x_i. The utility function (8) is strictly increasing and concave, and twice continuously differentiable. It follows that solving problem (4) is equivalent to maximizing the overall video quality or minimizing the overall video distortion.

The primary concept of incorporating the rate-distortion function of a video encoding scheme into congestion control is directly applicable to other video-encoding schemes beyond FGS. As a matter of fact, we can use the same model with a different utility function (namely the utility function of TCP [10] or TFRC) for any other TCP-like P2P application.

4.2 Simulation Setting

While we have carried out simulations on various network topologies, we present here only the representative results of simulations on a topology generated with Brite [11] in the router level topology model with 1000 routers. The average time interval of shadow price updates and constraint information updates is 10ms. The bandwidths of all links are randomly distributed between 100Mbps and 1000Mbps with 0.6ms average delay. To investigate the message overhead of the algorithm in difference size of network, we set up other two smaller topologies with 20 and 100 routers of the same average link

delay and bandwidth properties (part of the 1000 routers topology). We build the P2P streaming sessions consisting of various number of peers, each with a random access link bandwidth from 1Mbps to 100Mbps.

Tree construction mechanism: Since streaming applications are very time sensitive, we design a delay-based tree construction mechanism for P2P streaming systems. A new peer selects the closest peer in the tree as its parent node in terms of end to end delay. We further constrain that each peer has at most four children.

Assumption 1 holds in our experiments. By examining all bottleneck link constraint matrixes in our experiments, it was confirmed that "only sibling flows in the tree may share bottlenecks", i.e, namely on-sibling flows are with independent bottlenecks. Therefore the TCP-friendly bandwidth constraints at bottleneck links are fully decomposed for each subtree (1)(2), i.e., the network model and algorithm are fully distributed.

4.3 Rate Allocation

First, we compare the rate allocation results of our proposed algorithms with a standard unicast algorithm. We generate various P2P streaming systems sizes from 5 to 200 peers. In our simulations, the stepsize γ is set to 0.0001. Fig. 3 shows that the proposed algorithm is optimal in terms of average utility for various number of peers. If we first allocate the rates independently as unicast flows using the TCP/TFRC algorithm and then apply the data constraint at the same time, we get a set of rates with average/aggregate utility lower than the average/aggregate utility allocated by our algorithm.

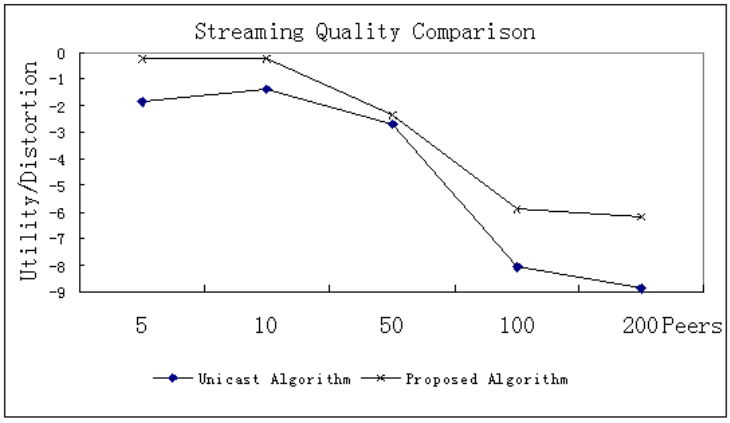


Fig. 3. Comparison of Average Streaming Quality

4.4 Messaging Overhead

Next, we investigate the messaging overhead of the algorithm in various size of network topology. Fig. 4 shows the average number of messages sent by all peers per time interval. The results show that the larger P2P session the more messages are produced. Moreover, the number of messages increases with the number of peers in the session linearly, i.e., each peer produces the same and small amount of message no matter the number of simultaneous P2P sessions. Therefore, our proposed algorithm can scale up to very large sizes and produces a small messaging overhead. Hence, we conclude that our algorithm is a fully distributed algorithm with small messaging overhead while maximizing the aggregate utility of P2P multicast tree.

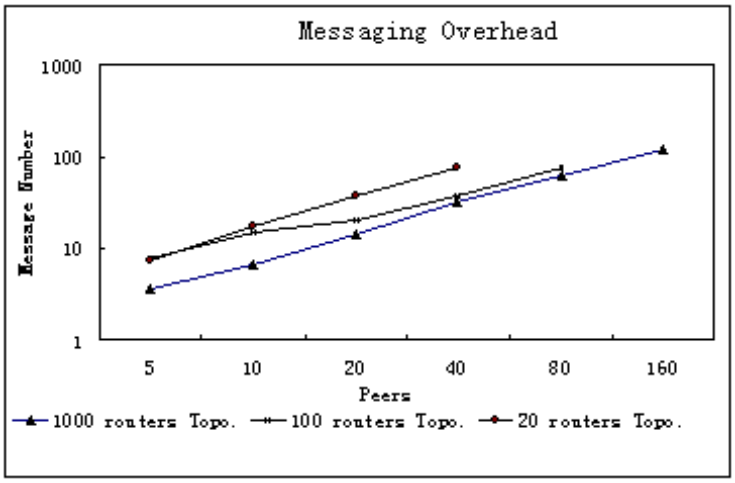


Fig. 4. Comparison of Messaging Overhead

5 Concluding Remarks

In this paper, we have proposed a fully distributed and TCP-friendly rate control model which maximizes the social utility for the P2P streams. The proposed algorithm works very well when bottleneck links are not access links. It is TCP-friendly to cross traffic outside the P2P session, while the rate allocation is proportionally fair in the P2P distribution tree [3]. In particular, the average time interval of the data shadow price updates and rate updates in the algorithm are much smaller than that of the TCP-friendly available bandwidth measurement so that the algorithm converges fast while the TCP-friendly available bandwidth measurement overhead is very small. Concerning future work, we are about to implement the algorithm in a real, large-scale P2P streaming system and will present more measurement results in upcoming publications.

6 Acknowledgments

The first author is supported in part by Swiss National Science Foundation under grant No.200020-121753 and National Science Foundation of China under grant No.60970127.

References

1. S. Floyd, M. Handley, and J. Padhye “Equation-Based Congestion Control for Unicast Application”, ACM SIGCOMM’00, September 2000.
2. S. Low, D. E. Lapsley, “Optimization Flow Control, I: Basic Algorithm and Convergence”, IEEE/ACM Trans. on Networking, vol. 7, no. 6, Dec. 1999.
3. Y.Cui, Y.Xue and K.Nahrstedt, “Optimal Resource Allocation in Overlay Multicast”, Parallel and Distributed Systems, IEEE Transactions on, 17(8),2006
4. Y. Chu, S. G. Rao, and H. Zhang,“A case for End System Multicast” in Proc. ACM Sigmetrics, June 2000
5. PPlive,http://www.pplive.com/
6. J. Liu, S. G. Rao, B. Li, and H. Zhang,“Opportunities and Challenges of Peer-to-Peer Internet Video Broadcast”, IEEE JSAC, Vol. 96, No. 1, January 2008.
7. W. Li, “Overview of Fine Granularity Scalability in MPEG-4 Video Standard”, IEEE Trans.on CSVT, 11(3), March 2001.
8. F. P. Kelly, A.K. Maulloo D.K.H. Tan, “Rate Control for Communication Networks:Shadow Prices, Proportional Fairness and Stability” Vol.49, Journal of the Operational Research Society, 1998.
9. David Luenberger, “Linear and Nonlinear Programming”,Addison-Wesley, 1984.
10. S.H. Low, F. Paganini,J.C. Doyle, “Internet congestion control”IEEE Control Systems Magazine, Vol.22, Issue 1, Feb. 2002.
11. www.cs.bu.edu/BRITE/
12. Sally Floyd, Kevin Fall, ”Promoting the Use of End-to-End Congestion Control in the Internet”, IEEE/ACM Transactions on Networking,1999
13. C. Barakat , P. Thiran, G. Iannaccone, C. Diot, P. Owezarski,“Modeling Internet backbone traffic at the flow level”, IEEE Trans. on Signal Processing, 51(8), Aug. 2003.
14. Jinyao Yan,M. May, B. Plattner, “Media and TCP-Friendly Congestion Control for Scalable Video Streams”, IEEE Trans. on Multimedia,8(2),2006.
15. T.M Cover and J.A.Thomas, ”Elements of Information Theory”, Wiley, New York, NY, 1991.
16. Jinyao Yan,M. May, B. Plattner, ”Distributed and Optimal Congestion Control for Application-layer Multicast: A Synchronous Dual Algorithm”, IEEE CCNC,Las Vegas, US.,2008