Computer Communications 33 (2010) 1578–1588

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

TCP-friendly congestion control for the fair streaming of scalable video

Sheng-Shuen Wang *, Hsu-Feng Hsiao

Department of Computer Science, National Chiao Tung University, 1001 University Road, Hsinchu, Taiwan

ARTICLE INFO

Article history:
Available online 24 February 2010

Keywords:
Bandwidth estimation
Congestion control
Packet train
Scalable video coding
TCP-friendly

ABSTRACT

Dynamic bandwidth estimation serves as an important basis for performance optimization of real-time distributed multimedia applications. The objective of this paper is to develop a TCP-friendly and fair congestion control algorithm which regulates the sending rate robustly by inferring the end-to-end available bandwidth. In addition to network stability, we also consider the characteristics of streaming applications, such as the bandwidth resolution in scalable video coding (SVC) which can achieve fine granularity of scalability at bit level to fit the time-vary heterogeneous networks. The congestion control algorithm is mainly composed of two phases: start phase and transmission phase to better utilize the network resource by subscribing SVC layers. In the start phase, we analyze the relationship between the one-way delay and the dispersion of packet trains, and then propose an available bandwidth inference algorithm which makes use of these two features without requiring administrative access to the intermediate routers along the network path. Instead of either binary search or fixed-rate bandwidth adjustment of the probing data as proposed in literature, a *top-down* approach is proposed to infer the initial available bandwidth robustly and much more efficiently. After acquiring the initial available bandwidth, the missions of the transmission phase include the adaptation of the sending rate fairly by progressive probing and also the accommodation of the network resource to TCP flows.

In case of the unavoidable network congestion, we unsubscribe scalable video layers according to the packet loss rate instead of only dropping one layer at a time to rapidly accommodate the streaming service to the channels and also to avoid persecuting the other flows at the same bottleneck. In addition, the probing packets for the estimation of the available bandwidth are encapsulated with RTP/RTCP. The simulations show that the proposed congestion control algorithm for real-time applications fairly utilizes network bandwidth without hampering the performance of the existing TCP applications.

1. Introduction

There has been explosive growth of emerging audio and video streaming applications recently. Many applications of multimedia communication over IP network, such as VoIP, multimedia on demand, IPTV, and video blog, have been integrated into our daily life rapidly. However, efficient delivery of media streams over the Internet is confronted with many challenges. Congestion control plays an important role to avoid congestive collapse by attempting to avoid oversubscription of any network resource of the intermediate nodes in the growing demand of those multimedia services over the Internet. TCP is a connection-oriented unicast protocol that offers reliable data transfer as well as flow and congestion control. However, multimedia streaming protocols have stricter requirement of transfer latency rather than reliable delivery. Many TCP-friendly congestion control protocols which are usually built upon UDP (User Datagram Protocol) with some specified congestion control algorithm are being developed so that the multimedia flows can behave fairly with respect to the coexistent TCP flows that dominate the network traffic so as to avoid the starvation of TCP traffic and to prevent the network from congestive collapse. In addition, the RTT unfairness introduced by the AIMD scheme of TCP leads to unequal bandwidth distribution among the competing flows with different round-trip time under the same congested links also should be taken into consideration. However, most of the existing streaming protocols have no consideration for the fairness and the characteristics of video streams which might affect the quality of streaming services significantly.

According to the general rate distortion curve shown in Fig. 1, the larger bit rate acquired, the better quality displayed. Some works focus on optimizing rate-distortion by multipath routing [28], or by active queue management and receiver feedback [27]. As to various video coding technologies, bit-stream scalability is a desirable feature for many multimedia applications so that graceful adaptation of transmission requirements can be achieved. Scalable Video Coding (SVC) [13], as an amendment G to the H.264/MPEG-4 AVC standard created by Joint Video Team (JVT), intends to encode a video sequence once and the encoded bit stream is able

* Corresponding author.
E-mail addresses: sswang@cs.nctu.edu.tw (S.-S. Wang), hillhsiao@cs.nctu.edu.tw (H.-F. Hsiao).

0140-3664/$ - see front matter Crown Copyright © 2010 Published by Elsevier B.V. All rights reserved.
doi:10.1016/j.comcom.2010.02.022

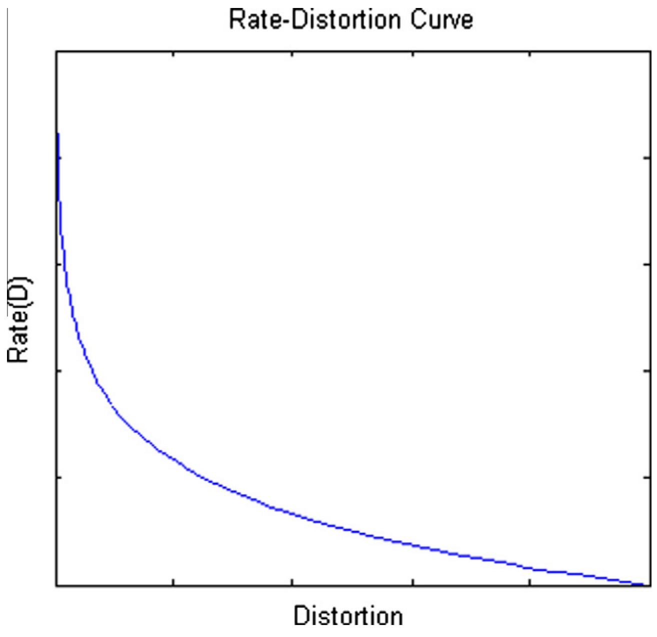

Fig. 1. Rate-distortion relationship.

to allow a diversity of different receivers to acquire and to decode a subset of the encoded bit stream without the need for transcoding. Scalable video coding enables not only efficient distribution of real-time multimedia streaming over heterogeneous networks but also a most promising solution for one-to-many congestion control over multicast networks. To fulfill above requirements, a scalable video bit stream contains a non-scalable base layer which is in compliance with the H.264/MPEG-4 AVC and one or more enhancement layers which may result from spatial, temporal or fidelity scalabilities of the scalable tools. The third octet in a Network Abstraction Layer (NAL) unit header records the layer identifications, including temporal_level (3bits), dependency_id (3bits), and quality_level (4bits). Thus, there can be $2^3 \times 2^3 \times 2^4$ video layers. Besides, the layers are subscribed layer by layer and the bit rate allocation between neighboring layers may vary significantly according to the streaming applications.

Due to the large number of possible scalable video layers and various bit rate allocation between layers, how to quickly converge to the time-varying available bandwidth without violating the existing flows under fairness competition becomes a critical factor in real-time video streaming. As to the information of available bandwidth, Multi-Router Traffic Grapher (MRTG) can use Simple Network Management Protocol (SNMP) to obtain the information from intermediate routers in the past. However, it is often difficult if not impossible due to various technical difficulties and privacy considerations or due to an insufficient level of measurement resolution [12]. One-way delay trend detection is utilized in Pathload [3] to measure the end-to-end available bandwidth by sending periodic packet trains. Since each packet train is used to determine only one decision that if the probed bit rate is greater or smaller than the available bandwidth, usually binary search is adopted to adapt the probing rate to the available bandwidth gradually. In contrast to acquiring initial available bandwidth over unicast networks, layered congestion control algorithm proposed in BIC [14], similar to Pathload, generates periodic burst packet trains from the upper layer over multicast network so that the probing periods of each receiver can be synchronized. Each receiver uses one-way delay trend detection to make the decision of joining one additional layer at a time and leaves a scalable video layer when packet loss rate exceeds a specified threshold. As a result, it is not suitable for receivers that might require joining or leaving several scalable video layers in a short time, due to dramatically fluctuant channels. In [11], a hierarchical sub-layer probing scheme which adopts coarse to fine layer partitioning to improve the efficiency of the probing interval was proposed. It might be helpful to reduce the number of probing periods when compared to BIC, but on the other hand, the probing packets might overshoot easily. Besides, network-layer multicast is still not widely deployed due to cost and management problems. Furthermore, it does not take TCP-friendly into account because only one probing rate is allowed for each synchronization point.

In this paper, we focus on TCP-friendly congestion control of fair end-to-end video streaming by inferring available bandwidth. We regulate the sending rate by using probing packets periodically such that a client running SVC applications can subscribe video layers gradually. In addition, we also consider the fairness property of both intra-protocol and inter-protocols, especially under different RTTs. Furthermore, RTP/RTCP [16] which relies on additional protocols to provide congestion control and to guarantee QoS to real-time multimedia steaming is integrated with the proposed congestion control algorithm. The remainder of this paper is organized as follows. In Section 2, the background and related works about bandwidth estimation and congestion control algorithm are presented. In Section 3, we describe our TCP-friendly congestion control algorithm with the consideration of RTT-fairness. In Section 4, the performance of our proposed algorithm is evaluated and the conclusion of this paper is given in Section 5.

2. Background and related works

In this section, we introduce the bandwidth estimation model based on the one-way delay (OWD) and the packet dispersion by sending probing packets. The selected TCP-friendly congestion control protocols in the literature will also be presented.

2.1. Bandwidth estimation

Among the results of emerging research in bandwidth estimation, link capacity and available bandwidth are of interest. The prior is constrained by the underlying transmission bandwidth. Given that packets are delivered from sender S to its receiver R through a fixed network path P, which consists of a sequence of store-and–forward links, the narrow link of a network path P is defined as the link with minimum capacity along the path. Assuming C_i is the link capacity of link i, and there are H hops in P, the capacity C of the narrow link is:

$$C = \min_{i=1\ldots H} C_i. \tag{1}$$

The technology of packet pair [1] with two back-to-back packets of packet size S is usually used to measure the capacity by observing the dispersion δ ($\delta = S/C$) passing through narrow link if there is no background traffic.

On the other hand, available bandwidth depends on the traffic load of the path and it is typically a time-varying random variable. Assume $\lambda_i(t)$ is the traffic load of link i at time t, the available bandwidth $A_i(t, T)$ of link i is the average unused bandwidth over some time interval T as shown in (2).

$$A_i(t,T) = \frac{1}{T}\int_t^{T+t} (C_i - \lambda_i(t))\,dt. \tag{2}$$

The available bandwidth $A(t,T)$ of the tight link which is defined as the link with minimum available bandwidth along a path is:

$$A(t,T) = \min_{i=1\ldots H} A_i(t,T). \tag{3}$$

Several studies have been devoted to the research of available bandwidth estimation in recent years. Except for the network mathematic model which is based on the specified network behavior or protocol [2], probing-based methods by means of packet train [1] analysis are widely adopted to infer network utilization. A packet train is a sequence of probing packets of equal packet size arranged either back-to-back or with some specified inter-packet dispersion. According to the analysis approaches, there are two major types of packet train based algorithms: one-way delay (OWD) based analysis model and dispersion based analysis model.

2.2. One-way delay based analysis model

Given that a sender transmits K packets of packet size S to its receiver and assumed that the propagation delay can be ignored, the OWD D^k of k-th packet can be modeled as the summation of the transmission delay (S/C_i), processing delay (σ_i), and queuing delay (d_i^k) of each and every link $(i = 1 \ldots H)$ along the path.

$$D^k = \sum_{i=1}^{H}\left(\frac{S}{C_i} + \sigma_i + d_i^k\right). \tag{4}$$

The OWD difference between adjacent packets can be expressed as the contribution from queuing delay as shown in (5).

$$\Delta D^k = D^{k+1} - D^k = \sum_{i=1}^{H}(\Delta d_i^k). \tag{5}$$

The idea of OWD based model is from the following proposition [7]:

$$\text{If } R_p > A, \quad \Delta D^k > 0,$$
$$\text{If } R_p \leq A, \quad \Delta D^k = 0.$$

R_p stands for the probing rate and A is the available bandwidth of a given path. The proposition concludes that if the probing rate is slower than the available bandwidth of the path, the arrival rate at the receiver will match their probing rate at the sender. On the other hand, if the probing rate is faster than the available bandwidth, then network queues will build up and the probing packets will be delayed $(\Delta D^k > 0)$. By observing the delay trend of OWD, many algorithms, such as Pathload [3], pathChirp [4], Pathbw [5], TOPP [6] and SLoPS [7], search for the turning point at which the sending rate and the receiving rate start to match.

2.2.1. Dispersion based analysis model

Dispersion based analysis model exploits the information of the inter-arrival time between two successive probing packets at the receiver. Let δ_{in} and δ_{out} be the time dispersion of a packet pair before and after passing through a single hop, respectively. Assume that the network queue will not be empty between the departure time of the first probing packet of a packet pair and the arrival time of the second probing packet in the joint queuing region (JQR) [8]. Given the network capacity of the tight link C, the available bandwidth $A(= C - \lambda)$ can be estimated by solving the following equation for the traffic load λ [9].

$$\delta_{out} = \frac{S}{C} + \frac{\lambda}{C}\delta_{in}. \tag{6}$$

However, if these two packets do not fall into the same period, (i.e., in the disjoint queuing region (DQR)[8]), the packet dispersion before and after passing through a hop will be equal $(\delta_{out} = \delta_{in})$. Bandwidth estimation tools such as IGI [8] and Spruce [10] are two examples that benefit from this observation. We denote $R_i = S/\delta_i$ as the departure rate after passing through hop i with packet time dispersion δ_i. For the probing packets passing through hop i with the arrival rate R_{i-1} to hop i and departure rate R_i from the same hop, R_{i-1} and R_i will have the following relationship:

$$R_i = R_{i-1} = \frac{C_i}{\lambda_i + \max\{R_{i-1}, A_i\}} \tag{7}$$

λ_i is the traffic load of hop i.

Obviously, R_i is monotonically decreasing because the departure rate will be less than or equal to the arrival rate $(R_{i-1} \geqslant R_i)$, depending on whether the arrival rate (R_{i-1}) is greater than the available bandwidth (A_i). In addition, the available bandwidth A is the minimum of all A_i; thus we can induce that:

$$R_{in} \geqslant R_{out} \geqslant A. \tag{8}$$

2.2.2. One-way delay vs. packet dispersion

The relation between OWD and packet dispersion can be expressed briefly as in Fig. 2. When the probing rate is less than the available bandwidth, we will most likely have $\Delta D^k = 0$ and $\Delta\delta = \delta_{out} - \delta_{in} = 0$. In other words, it means that there is no delay trend of OWD and the packet dispersion of the packet pair measured at the sender and the receiver would be the same. On the other hand, when the probing rate is more than the available bandwidth, the cross traffic can enlarge the dispersion, and will cause the increasing of queuing delay. Assume that the start transmission time for packet i and $i+1$ is s_i and s_{i+1} and the arrival time at receiver is r_i and r_{i+1}. We can infer the dispersion $\Delta\delta_i = (r_{i+1} - r_i) - (s_{i+1} - s_i) = (r_{i+1} - s_{i+1}) - (r_i - s_i) = \Delta D^i$. If ΔD^i is not equal to 0, the OWD for a packet train will have increasing trend. Therefore, $\Delta\delta_i$ can be viewed as an index of queuing delay. In summary, we can conclude that the packet dispersion and OWD are two criteria that can work together to estimate the available bandwidth more precisely.

Available bandwidth can fluctuate dramatically and thus it is very important for the bandwidth measurement to converge fast and accurately. In the previous bandwidth estimation algorithms, such as Pathload [3], which uses *binary search* to adjust the probing rate for the next iteration, and IGI [8], which updates the probing rate by some fixed step size to inspect whether the probing rate matches the available bandwidth, they might be too inefficient to infer the probing rate for the next iteration, especially in real-time distributed applications, in addition to the possible resolution issue of the estimated bandwidth.

2.3. Congestion control

For most unicast flows that require transferring data reliably and as quickly as possible, one of the straightforward options is to use TCP directly. However, TCP whose congestion control is mainly based on AIMD scheme and slow start cannot utilize

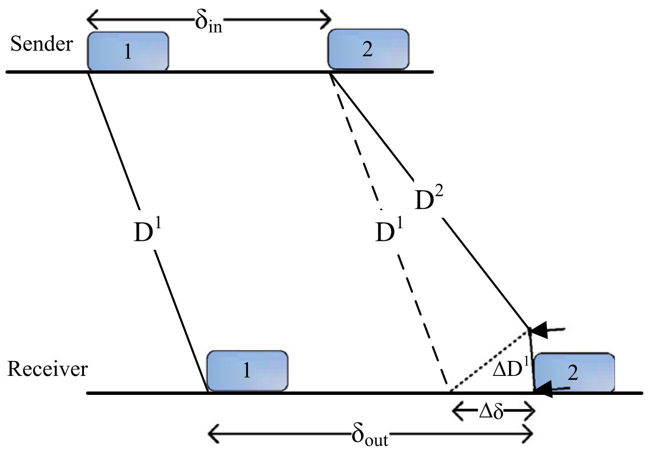

Fig. 2. The relationship between OWD and packet dispersion.

network resource efficiently, especially in the case of networks with high bandwidth-RTT product. It might be too conservative for the AIMD mechanism of TCP to increase the congestion window linearly per RTT, while at the same time the reduction of congestion window by a factor of half tends to be too drastic. This results in inefficient link utilization. For some applications such as multimedia streaming, they can tolerate data loss to some degree and are usually highly delay sensitive.

Datagram Congestion Control Protocol (DCCP) [29] has been proposed by the Internet Engineering Task Force (IETF) to implement connection setup, teardown, ECN, and feature negotiation, and also to provide a framework to allow applications to choose different TCP-friendly congestion control profile such as TCP-like or TFRC by CCID. TCP-friendly congestion control algorithms built upon UDP with suitable rate adaption mechanism for streaming applications are designed to ensure that the coexisting TCP flows will not be treated unfairly by non-TCP flows. Widmer [18] classified unicast TCP-friendly congestion control algorithms into window-based [20] and rate-based [17][21] algorithms. The algorithms in window-based category use a congestion window at the sender or at the receiver to ensure TCP friendliness. On the other hand, rate-based congestion control achieves TCP friendliness by adapting the transmission rate according to either Additive Increase/Multiplicative Decrease (AIMD) scheme or model-based congestion control algorithm. The AIMD scheme mimics the behavior of TCP, while the model-based one follows Padhye's TCP throughput model [19] instead of TCP-like AIMD mechanism. The analytical model of TCP-friendly available bandwidth is shown as follows.

$$f(p) = \frac{S}{t_{RTT}\sqrt{\frac{2bp}{3}} + t_{RTO}3\sqrt{\frac{3bp}{8}}p(1+32p^2)} \tag{9}$$

S is the packet size; p is the packet loss rate; t_{RTT} is the Round Trip Time; t_{RTO} is the TCP RTO (Retransmission Time Out) which is usually set to be $4 \times$ RTT in the experiments, and the number of packets acknowledged by a single TCP acknowledgement is b.

TCP Friendly Rate Control (TFRC) [17], which is rate-based instead of window-based congestion control, adjusts its sending rate based on the equation according to packet loss rate and RTT. The loss event rate p is measured as the inverse of the average loss intervals. According to TCP NewReno, all the lost packets in the same congestion window are treated as a single loss event and the reduction of congestion window is only triggered once. The loss event may consist of several lost packets in the duration of a round-trip time since the first occurred packet loss, and the loss interval is defined as the number of packets between consecutive loss events. A certain number of loss intervals are averaged using exponential decaying weights so that the older loss intervals contribute less to the average. It prevents the loss rate from reacting too strongly to the single loss event. If we take the recent L loss event intervals into account and I_n is the nth loss event interval, which is the number of packets sent between the nth and the $(n+1)$th loss event, the weighted average of loss event intervals I~ can be obtained as follows and the loss event rate equals $1/I$~.

$$\widetilde{I} = \sum_{l=1}^{L} w_l I_{n-1}$$
$$\begin{cases} wl = 1, & \text{if } l < L/2 \\ wl = 1 - \frac{\left(l-\left(\frac{L}{2}-1\right)\right)}{\left(\frac{L}{2}+1\right)}, & \text{otherwise} \end{cases} \tag{10}$$

Slow start is applied to the initialization phase and retransmission timeout. Otherwise, congestion avoidance phase is applied to detecting a loss and calculating the sending rate by the TCP throughput model. However, earlier work [23] shows that the convexity of $1/I$~ ($E[1/I\sim] \neq 1/E[I]$) and different retransmission timeout period (RTO) can be the reasons for TCP and TFRC to experience initially different sending rates. The difference of loss event rates due to the different sending rates greatly amplifies the initial throughput. In addition, since TFRC increases the sending rate per round-trip time, it will lead to RTT unfairness. In other words, the flows with shorter RTT will gain more bandwidth than the flows with larger RTT under the same bottleneck.

3. The proposed method

The queuing delay usually becomes severe before the event of packet loss due to buffer overflow. Therefore, using queuing delay as an indication of congestion can be more accurate to perform estimation than using loss event rate. Different from the traditional TCP which adjusts the congestion window by increasing one packet per RTT (TCP Tahoe, NewReno, ...,etc.) or by observing the change of RTT (TCP Vegas, Fast TCP[26] ...etc.), we determine whether to increase one layer by observing the queuing delay of packet trains.

RTP and RTCP are popular streaming protocols over the Internet. In order to cooperate with SVC and RTP/RTCP, our proposed congestion control algorithm consists of two phases: start phase and transmission phase as illustrated in Fig. 3, compared to the slow start and congestion avoidance status of TCP congestion control algorithm. In the start phase, we focus on the precision aspect of bandwidth estimation algorithm so that the bandwidth can be utilized efficiently especially in the case of networks with high bandwidth-RTT product. TCP-friendly and RTT-fairness will be taken into account during the transmission phase.

3.1. Start phase

Before laying down the proposed algorithm, we use ns2 network simulator to conduct simulations with the topology in Fig. 4 for the observation of our analysis shown in the previous section. The capacities along the path from sender to receiver are 100, 75, 55, 40, 60, and 80 Mbps, respectively. The link delay for each link is 100 ms. Cross traffics are generated from 16 random sources at each link. The inter-arrival time of those cross traffics from each source follows Pareto distribution with exponential factor $\alpha = 1.5$. A packet train transmitted by the sender consists of 10 packets of packet size $S = 1500$ bytes with fixed packet dispersion δ_{in} so that the probing rate $R_{in} (= S/\delta_{in} = 40 \text{ Mbps})$ for these 10 packets is greater than the available bandwidth. The receiver R will record the dispersion δ_{out} of arrival packets under different network utilizations of the bottleneck (i.e., the tight link).

In Fig. 5 we show the dispersion distribution of received packets of the bottleneck link under various network utilizations. It is obvious that the received dispersion δ_{out} is influenced by the utilization of the tight link and has positive correlation with the network utilization when the fixed probing rate (R_{in}) is greater than the available bandwidth (A). The reason is that if the network load gets heavier, there is a higher probability that packets of cross traffic will be placed among the probing packets and they will contribute to the packet dispersion of the probing traffic. In addition, since the received probing rate $R_{out} (= S/\delta_{out})$ is inversely proportional to the received dispersion δ_{out} and the available bandwidth A is equal to $C - \lambda$, the received probing rate at R will have positive correlation with the available bandwidth. Under this condition, the received probing rate is also an *upper bound* of the available bandwidth as shown in Section 2, Part A. Based on the inequality in (8), we can have a better "guess" of the probing rate for the next iteration.

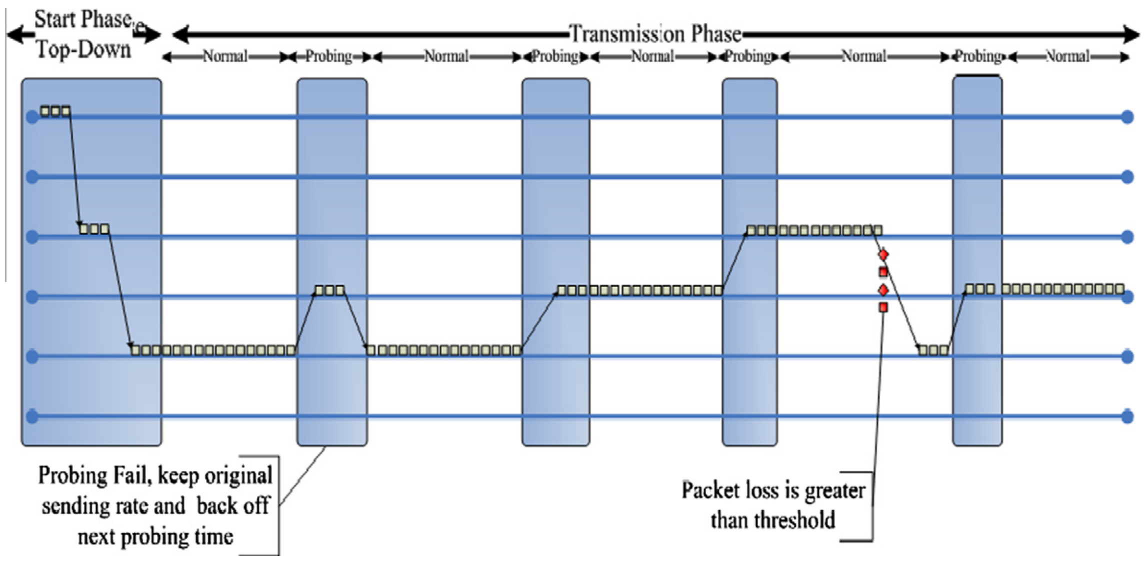
Start Phase
Top-Down
Transmission Phase
Normal
Probing
Probing Fail, keep original sending rate and back off next probing time
Packet loss is greater than threshold

Fig. 3. Schematic diagram of the proposed congestion control algorithm.

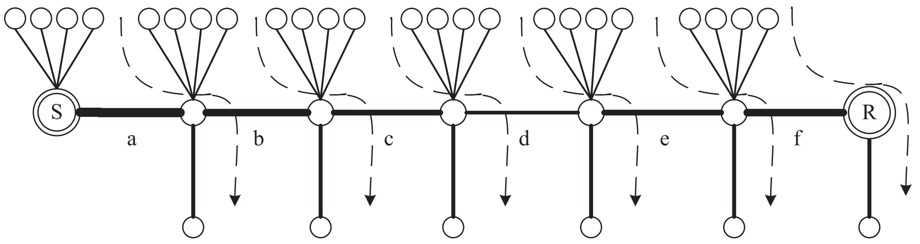

Fig. 4. Network topology used in the ns2 simulations.

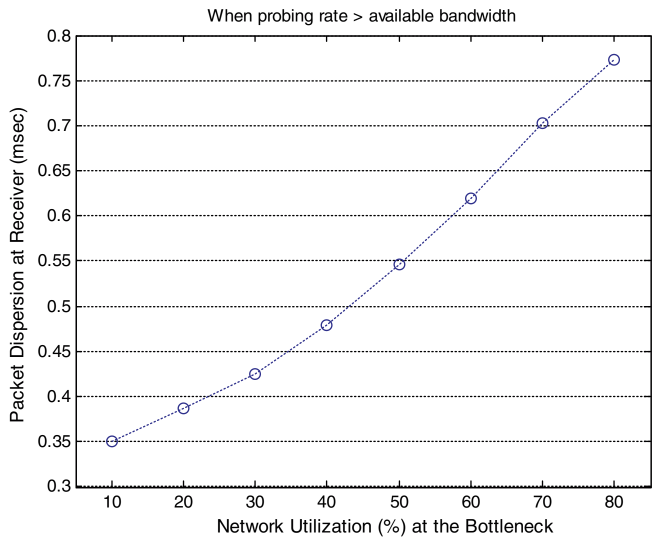

Fig. 5. The received packet dispersion of probing traffic at various traffic loads, given that the probing rate is greater than the available bandwidth.

In the beginning of the start phase, we rarely have a clue about the information of available bandwidth, the probing sender transmits a packet train with probing rate (R_{in}) which is equal to the capacity of tight link. Or, the probing rate can also simply be a packet train with back-to-back packets when the capacity of tight link is not available, or the initial probing rate can be the bit rate of the largest layer in SVC. After passing through the links, its receiver observes the dispersions of the packet train and also the corresponding rate ($R_{out=}S/\delta_{out}$), and feeds back the information R_{out} to the sender as the next probing rate. We continue above steps iteratively until the probing rate and the available bandwidth start to match by means of performing the delay trend detection on the OWD. The *full search algorithm* [11] is modified to detect the delay trend. It is shown in [11] that full search algorithm is better than the Pairwise Comparison Test (PCT) and Pairwise Difference Test (PDT) of the delay trend detection in Pathload[3]. The block diagram of the proposed algorithm for available bandwidth estimation is showed in Fig. 6.

Instead of binary search in Pathload, we adapt the top-down approach to use the received probing rate as the next probing rate for the sender due to the analysis that the received probing rate has positive correlation with the available bandwidth and it is also an upper bound of the available bandwidth. The estimation process can converge faster as shown in the next section. In addition, we use OWD as a criterion to decide whether the probing rate is less than or equal to the available bandwidth. In other words, a packet train is sent to examine whether the probing rate is greater than the available bandwidth and the received rate is used as the next probing rate iteratively until no delay trend is detected by (11). Eq. (11) is used to calculate the probability whether the OWD of the latter packets (D^k) is greater than the previous ones (D^l) for a packet train of length M. Therefore, we can acquire the residual bandwidth faster than the slow start of TCP and TFRC. As to threshold$_{FS}$, if the probing rate is greater than 2 Mbps, the heuristic value

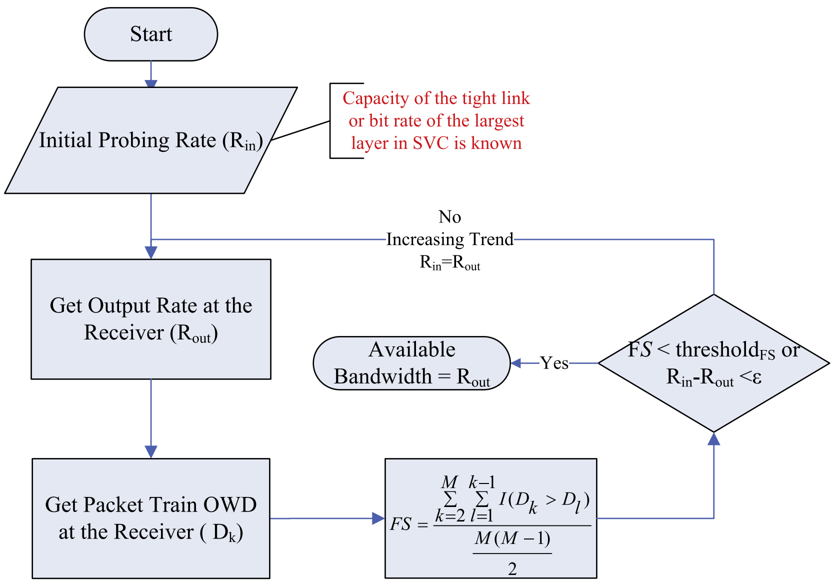
Start
Initial Probing Rate (R_{in})
Capacity of the tight link or bit rate of the largest layer in SVC is known
No
Increasing Trend
$R_{in}=R_{out}$
Get Output Rate at the Receiver (R_{out})
Available Bandwidth = R_{out}
Yes
$FS <$ threshold$_{FS}$ or $R_{in}-R_{out} < \varepsilon$
Get Packet Train OWD at the Receiver (D_k)
$FS = \frac{\sum_{k=2}^{M}\sum_{l=1}^{k-1} I(D_k > D_l)}{\frac{M(M-1)}{2}}$

Fig. 6. System diagram for available bandwidth estimation in start phase.

of the threshold is 0.7. Otherwise, the range of the threshold is 0.6–0.7.

$$FS = \frac{\sum_{k=2}^{M}\sum_{l=1}^{k-1} I(D^k > D^l)}{\frac{M(M-1)}{2}}$$
$$\begin{cases} I(D^k > D^l) = 1 & \text{if } D^k > D^l, \\ I(D^k > D^l) = 0, & \text{otherwise.} \end{cases} \tag{11}$$

Comparing with the other algorithms in literature, such as Pathload [3] or IGI [8], our algorithm is more robust. For example, if the available bandwidth suddenly drops down during the estimation period, the Pathload or IGI may over estimate the available bandwidth easily. Therefore, it is not appropriate for multimedia streaming. However, our probing rate is based on the previous receiving rate; it will not be influenced by the sudden decrease of available bandwidth. On the other hand, when the available bandwidth increases during the probing, the estimation might be underestimated but it will not induce packets loss.

V	P	*1*	CSRC count	*1*	Payload type	Sequence number (16 bits)
Timestamp (32 bits)						
Synchronization source (SSRC) id. (32 bits)						
Contributing source (CSRC) id. (0~15 items, 32 bits each)						
defined by profile						extension header length
System clock						

Fig. 7. Modified RTP header for probing packet.

V	P	subtype	Payload type=APP	length(16 bits)
SSRC/CSRC				
Probe				
Received Rate / Application Specified Data				

Fig. 8. Modified RTCP APP packet header for feedback packet.

3.2. Transmission phase

Once the initial sending rate is determined, we periodically send probing packets to detect whether there is extra available bandwidth to enhance the QoS of video streaming. Basically there are normal periods and probing periods in this phase as shown in Fig. 3. Therefore, we have to determine how long to perform the probing and how to drop a video layer whenever congestion is detected in this phase. In order to compete with TCP flows for the network resource friendly, the probing period is dynamically adapted based on the RTT and the bit rate between two layers. Let current sending bit rate for layer i be $rate_i$ and the algorithm will probe $rate_{i+1}$ next. The time interval t between two probing events is determined as follows:

$$t = \frac{rate_{i+1} - rate_i}{s}\mathrm{RTT}_{\min} \tag{12}$$

The $\mathrm{RTT}_{\min}$ is the minimal RTT recorded so far and it is updated whenever smaller RTT is measured from the RTCP receiver report.

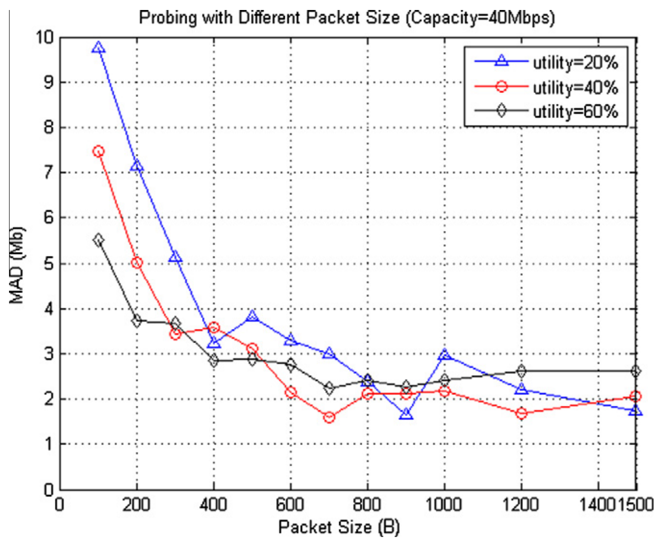

Fig. 9. The estimation error with different packet sizes under different network utilizations.

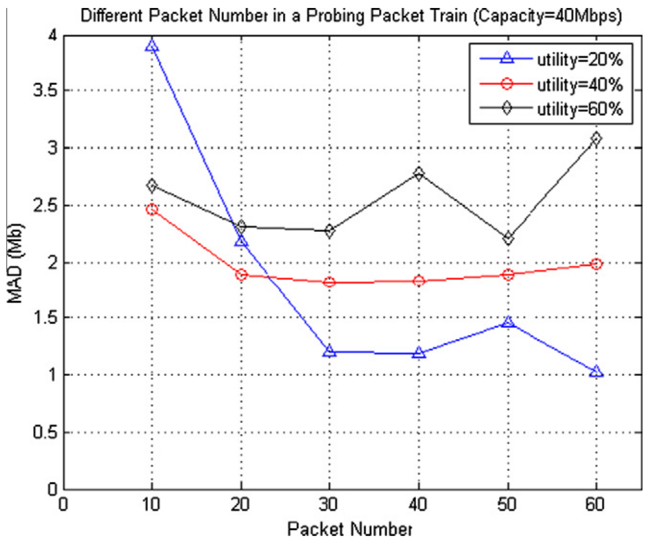

Fig. 10. The estimation error with different packet numbers in a packet train under different network utilizations.

In addition, whenever the probing fails to increase a video layer, the next probing period will be postponed by time interval t as indicated in (13). Assuming there are N video layers and $layer_i$ is the bit rate of current layer i, threshold_t is the upper bound of time interval between two probing. According to the experiments, $\text{threshold}_t = 10$ s is selected. In this way, the sending rate of the lower layer has better opportunities to catch up with the higher layer because of the shorter probing period when there is extra available bandwidth. Furthermore, this rule can also reduce the influence of different RTTs under the same bottleneck on the receiving rate.

$$t = \min\left(\text{threshold}_t,\ t \times \left(1 + \frac{layer_i}{layer_N}\right)\right) \tag{13}$$

As usual, the packet loss rate (p) is also the indicator to detect congestion occurrence during normal periods. Due to the characteristic of multimedia streaming which allows tolerable degree of quality deterioration, if the packet loss rate is greater than $\text{threshold}_{\text{PLR}}$, we drop layers and reset the time interval of two probing periods, according to (12) immediately. In order to respond to the packet

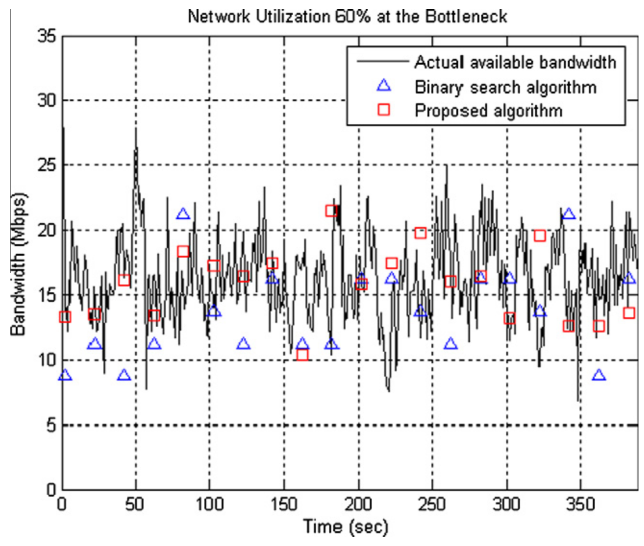

Fig. 11. The estimated available bandwidth at network utilization 60% (average available bandwidth is 16 Mbps).

Table 1
The MAD and probing number under different network conditions.

Network utilization (%)	Binary search		Proposed algorithm	
	MAD (Mbps)	Probing number	MAD (Mbps)	Probing number
20	1.98	4	1.19	2.25
40	2.231	4	1.82	2.7
60	3.62	4	2.27	3.05

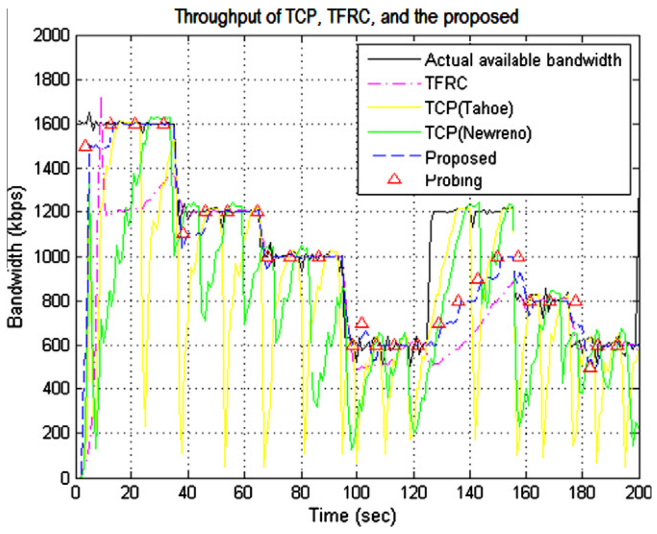

Fig. 12. The throughput of TCP, TFRC, and the proposed algorithm under different background traffics.

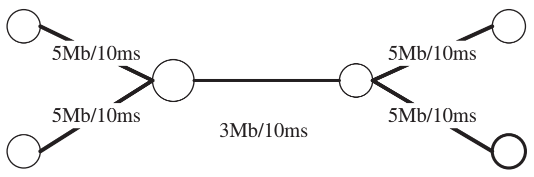

Fig. 13. Dumbbell network topology for TCP-friendly simulation.

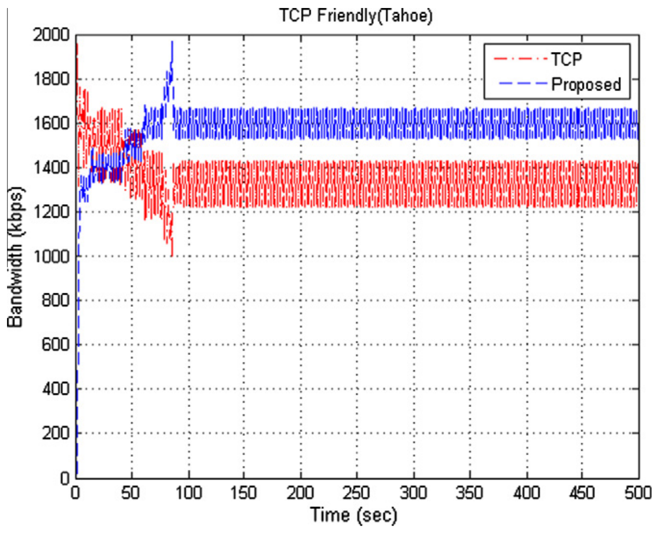

Fig. 14. Throughput for TCP (Tahoe) and the proposed algorithm to compete under the same network condition.

loss rate, we also take sending rate multiplied by the factor $(1-\sqrt{p})$ into account, since the throughput is roughly inverse proportion to $\sqrt{p}$ according to the equation model of TCP throughput. Therefore, our transmission rate will be adjusted to the corresponding layer of $rate_i \times (1-\sqrt{p})$.

For the avoidance of the introduction of unnecessary noise and also for better robustness over a wide range of time scales, threshold$_{\text{PLR}}$ used in normal periods needs to be determined carefully. When the damage of a layer is greater than what we can tolerate, we have to consider dropping layers. The determination of the threshold of packet loss rate (threshold$_{\text{PLR}}$) is related to the bit rate of the current sending layer as shown in (14) so that the transmission of lower layer would have higher probability than the transmission of the higher one to increase one additional layer.

$$\begin{aligned}&\text{threshold PLR} = \alpha \times ((rate_i - rate_{i-1})/rate_i)\\&\alpha \in (0,1].\end{aligned} \tag{14}$$

In additon, if the packet loss rate in normal periods is less than the threshold$_{\text{PLR}}$ and the RTT demonstrates obvious decrement and it is close to RTT$_{\min}$, it implies that there can be extra available bandwidth. We also launch probing mechanism immediately to examine if one more layer is appropriate. The condition for additional probing according to RTT is as follows:

$$\text{RTT}_{\text{measure}} \approx \text{RTT}_{\min}. \tag{15}$$

3.3. Integration of RTP/RTCP and congestion control algorithm

Realtime Transport Protocol (RTP) provides end-to-end delivery services for real-time traffic and RTCP (Realtime Transport Control Protocol) conveys the statistic information about the participants and also the QoS-related information. In RTP, the field *sequence number* can be used to detect packet loss and *timestamp* is used to reflect sampling instant of the first byte of data where clock frequency is specified by the profile of payload format documents of the application. In order to use existing RTP data packets as probing packets so that the probing overhead can be kept at minimal, the header extension of RTP is updated to include the system clock information. Besides, the marker bit is also set so as to bypass the non-probing data efficiently. Fig. 7 shows the values of the corresponding fields. RTCP provides feedback on the quality of data distribution by five packet types, including Sender Report (SR), Receiver Report (RR), Source Description (SDES), BYE, and Application (APP), to generate compound packets. The APP packet is selected as the control message for the probing results. The control message contains the received rate and the corresponding values are shown in Fig. 8.

4. Simulations and experimental results

We use ns2 to simulate and evaluate the performance of the proposed algorithms. As mentioned in the previous section, part of the normal data packets are utilized as probing packets so as to eliminate the cost of probing overhead. At first, we examine the precision and the efficiency of the proposed bandwidth estimation algorithm in the start phase. Furthermore, we present experiments and discussions from the aspects of available bandwidth utilization, TCP friendliness, and fairness, respectively. Besides, slow start in TCP or TFRC is inefficient especially for high speed networks. In order to verify whether the proposed algorithms can work well in various bottleneck capacities, simulations with different capacities are conducted in bandwidth estimation and congestion control.

4.1. Bandwidth estimation

We consider the simulation in case of large bandwidth-delay product topology in Fig. 4 where the capacities from the sender to its receivers are 100, 75, 55, 40, 60, 80 Mbps with 100 ms link delay for each link. For all the following simulations, if the parameters are not mentioned explicitly, the default settings are as flows. The length of a packet train is 30 packets and the probing packet size is 1500 bytes. The link with capacity 40 Mbps is the tight and narrow link and the queue length is 20 packets. Cross traffic is generated from 16 random sources at each link with Pareto distribution as stated earlier. To mimic the traffic in the real world, the flows are set as follows: 40% are 40 bytes, 50% are 550 bytes, and 10% are 1500 bytes. The threshold$_{\text{FS}}$ in Fig. 6 is 0.7 and ε is 2 Mbps. We discuss the parameters of probing packet train, size and length and compare with binary-search as follows.

4.1.1. Packet size

In this section, we observe the influence of the packet size under different network utilizations because the packet size has

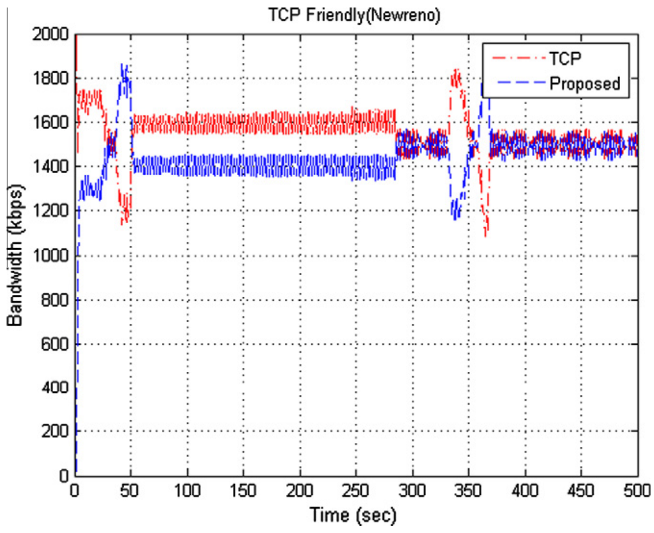

Fig. 15. Throughput for TCP (NewReno) and the proposed algorithm to compete under the same network condition.

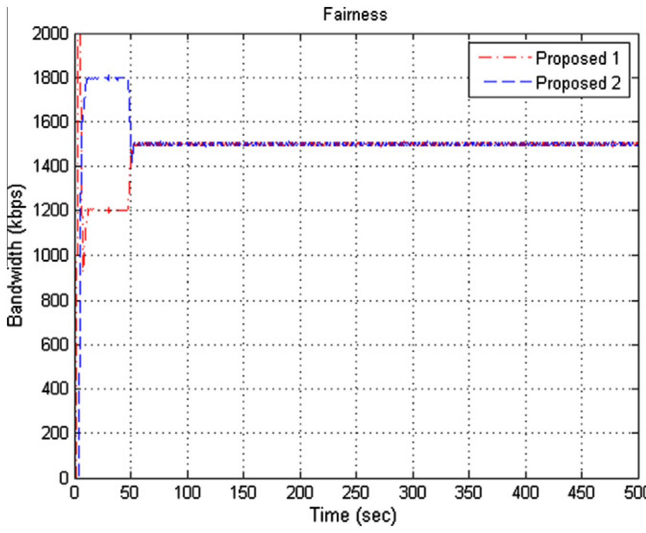

Fig. 16. Throughput for two flows with the proposed algorithm to compete under the same network condition.

impact on transmission delay and packet dispersion according to (6). If the packet size is larger, the time interval between two consecutive packets in a packet train will increase at a fixed probing rate. Therefore, the delay trend detection will not be disturbed easily by the transient variation of cross traffic. The mean absolute difference (MAD) between the average real available bandwidth and the estimated bandwidth is used to measure the estimation error. From Fig. 9, when the packet size is greater than 800 bytes, the estimation error is smaller as well as the MAD variation.

4.1.2. The length of packet train

The length of packet train means the sampling rate to detect delay trend. If the sample rate is not frequent enough like 10 packets under 20% utility in Fig. 10, the estimation error will be larger. However, too many probing packets with higher sending rate than the available bandwidth will easily lead to queue overflow and it is intrusive to the existing cross traffic like the result of 60 packets under 60% utility. It is a tradeoff between the precision of estimation and the amount of network traffic causing by probing packets. Therefore, the length of packet train has to be sufficiently large to accommodate various situations and it needs to be not intrusive. As shown in Fig. 10, 30 or 50 packets in a packet train is the better choice. In order to reduce the burden of the network load, we choose 30 packets in the following simulations.

4.1.3. Comparison

We compare our algorithm to binary-search based algorithm in [11]. The available bandwidth is estimated every 20 s, and the spots in Fig. 11 show the starting time of each estimation and the corresponding estimated available bandwidth. From Fig. 10, it is shown that our algorithm keeps much closer to the curve of real available bandwidth.

Table 1 shows the MAD between the estimated and the true bandwidths. The required average number of packet trains to complete bandwidth estimation under different network utilizations is also shown. Our algorithm outperforms binary-search based algorithm, especially in the case of heavy traffic load.

Using binary-search to infer the available bandwidth is unstable with only one threshold, because once the probing rate falls in the gray region [3], the results may be inconsistent. Even though Pathload uses two thresholds to detect delay trend, it is still easy to misjudge in the fast fluctuant network. Our proposed algorithm utilizes the features of OWD and dispersion to avoid above problems and the accuracy and convergence speed of probing can be improved.

4.2. Congestion control

As to the performance evaluation of congestion control algorithms, we present the discussion from the aspects of available bandwidth utilization, friendliness, and fairness, respectively. Simulations over Brite-generated network topology are also shown. Our simulation parameters are set as follows if they are not especially mentioned. The threshold of full search algorithm is 0.63. As to the various scalabilities of SVC, we assume the maximum bit rate is 2 Mbps and the maximum bit rate is equally divided into 20 layers. The accumulative bit rates for all layers are 100, 200, 300, ..., 2000 kbps. The parameter α of threshold$_{\mathrm{PLR}}$ is set to 0.3 heuristically. The version of TCP flows is Tahoe and the packet sizes for TFRC and TCP flows are also 1000 bytes.

4.2.1. Available bandwidth utilization

In this section we evaluate the available bandwidth utilization of the proposed congestion control algorithm compared to the algorithms in TFRC [22] and TCP. We use ns2 network simulator to conduct simulations with the topology shown in Fig. 4. The length of a packet train is 30 packets of packet size 1000 bytes. The capacities along the path are 10, 7.5, 5.5, 2, 6, and 8 Mbps, respectively. The link with capacity 2 Mbps is the tight and narrow link and the queue length is 30 packets. The cross traffic consisting of constant-bit-rate flows (CBR) of packet size 550 bytes is generated for each link to alter the available bandwidth so that the available bandwidth can exhibit large fluctuation over a period of time.

From the experiment results shown in Fig. 12, the sawtooth-like rate shape of TCP shows the worst utilization. The proposed congestion control algorithm has better performance than TFRC not only in the start phase but also in the transmission phase. During the start phase, our algorithm can converge to the available bandwidth by top-down approach faster than the slow-start in the TFRC/TCP. During the transmission phase, our proposed algorithm fast converges to the available bandwidth under TCP-friendly condition because our proposed algorithm estimates the available bandwidth by probing instead of TCP throughput model. In our proposed algorithm, the efficiency to converge to the available bandwidth depends on the time interval between two probing periods. In order to be friendly with TCP, we mimic the congestion avoidance status in TCP to dynamically adapt the time scale between two probing periods. The simulation results of the proposed algorithms show steady throughput without overestimating the shared bandwidth which is a critical property of video streaming.

4.2.2. Friendliness

We simulate the proposed algorithm with different versions of TCP protocols over a basic dumbbell network topology depicted in Fig. 13. The capacity of the bottleneck is 3 Mbps, and the proposed algorithm starts behind the TCP flows by 2 s. As for calculating the average throughput, the first 200 s are considered as a transient phase and they are not taken into account. In Fig. 14, the average throughputs of TCP Tahoe and the proposed algorithm are 1327.63 and 1599.5 kbps, respectively. In Fig. 15, the average throughputs of TCP NewReno and the proposed algorithm are 1537.24 and 1462.75 kbps. Tahoe and NewReno are aggressive congestion control algorithms that send packet continuously until pocket loss occurs. According to Jain's fairness index [24], the corresponding indices of Tahoe and NewReno with our proposed algorithm are 0.991 and 0.999. The main difference between Tahoe and NewReno is the fast retransmission mechanism where NewReno sets the congestion window to one half of the previous window

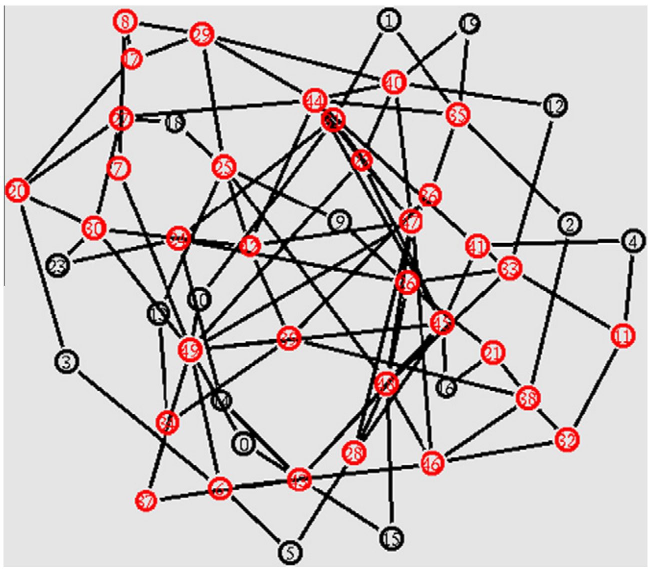

Fig. 17. The large scale network topology generated by Brite.

Table 2
Peer size and fairness index under different protocols.

	4 peers	8 peers	16 peers	16 peers (20 Mbps)
Proposed	0.941021	0.919904	0.916501	0.994578466
TFRC	0.437532	0.610675	0.999736	0.520080428

size and Tachoe sets to one after receiving three duplicate acknowledgements. Therefore, NewReno has better throughput than Tahoe. The throughput of the proposed algorithm depends on threshold$_{PLR}$ because our proposed algorithm only considers dropping video layers when the packet loss rate is greater than threshold$_{PLR}$. Due to the imitation of the TCP behavior to adapt the probing period and also the usage of queuing delay trend as congestion index in the transmission period, our proposed algorithm will not lead to congestion collapse and starvation of TCP traffic. In addition, the step size between video layers is also one of the factors affecting the throughput. If TCP and the proposed algorithm are in the congestion avoidance status and compete for the residual bandwidth at the same time, our proposed algorithm will occupy enough bandwidth for a layer by probing. If the residual bandwidth is not enough for a layer, the proposed algorithm will not change the sending rate. On the other hand, the average throughputs of TCP Tahoe/TFRC and TCP NewReno/TFRC are 1189.26/1755.06 and 1411.63/1587.99, respectively. The corresponding fairness indices are 0.964 and 0.997, respectively.

We run two instances of the proposed congestion control algorithm over topology as shown in Fig. 13 and the difference of the start time between two flows is 2 s. The average throughputs for each are 1500.06 and 1499.93. Form Fig. 16, the throughput of two instances will converge after 50 s, because our congestion control algorithm takes the residual bandwidth and esteems the existing traffic until packet loss reaches the threshold.

4.2.3. Fairness over larger network topology

In order to be close to the real network topology further, we use Brite [25] topology generator to generate the network topology as shown in Fig. 17 which is composed of 50 nodes. 33 nodes (the red ones in Fig. 17) among those 50 nodes are in the core network. The minimum degree for each node is 2 and the capacity for each link is 10 Mbps. We observe the fairness index between the number of peers and different protocols. From Table 2, our proposed algorithm seems to have better performance in 4 and 8 peers. However,

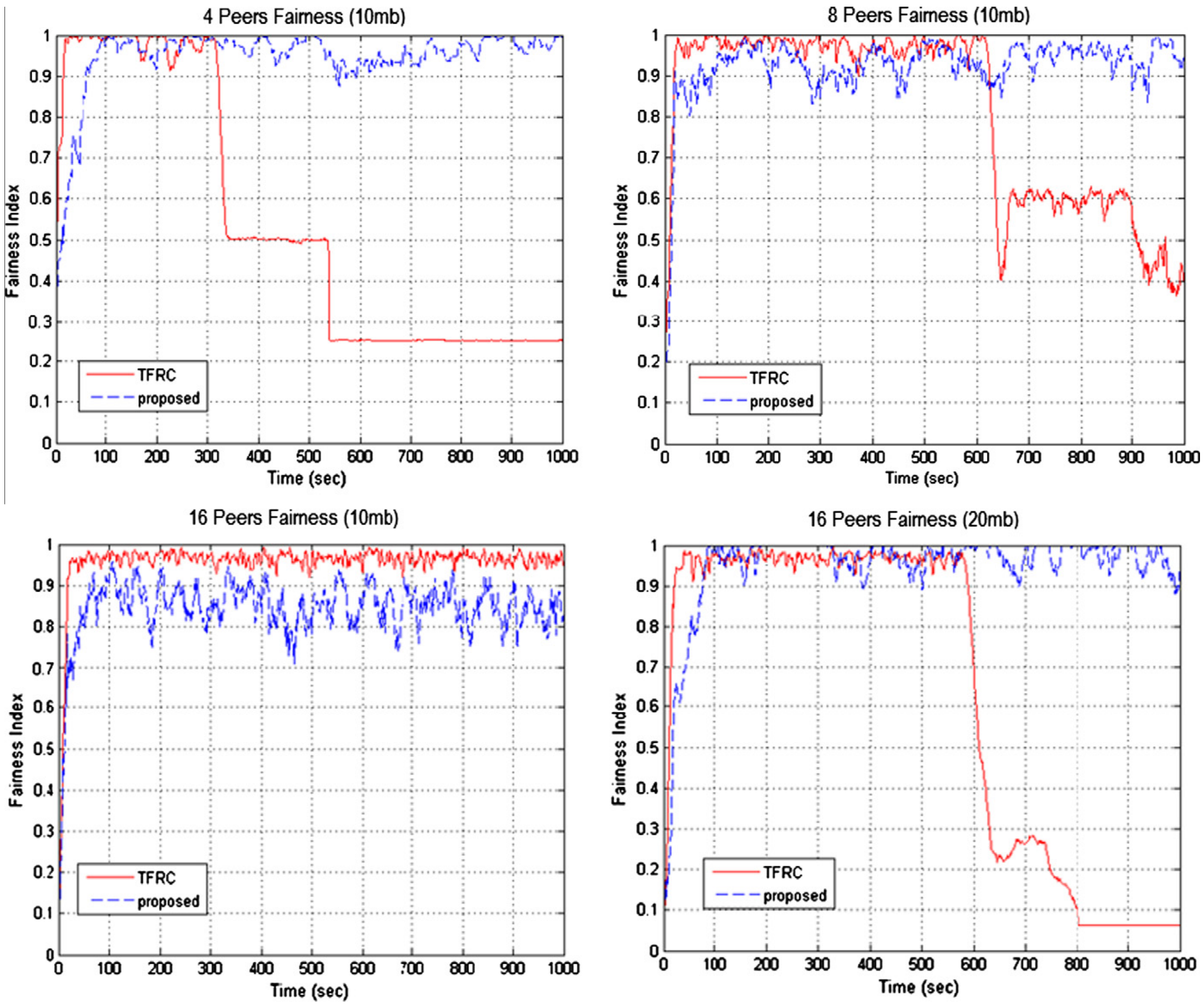

Fig. 18. Fairness index over time with respect to different peer numbers and shared bandwidths.

Table 3
Throughput and fairness index under different RTT.

	Proposed	TFRC
1RTT	0.983293973	0.781547055
2RTT	0.948974893	0.686864937
3RTT	0.945894628	0.626919711
4RTT	0.931168674	0.594973221
5RTT	0.928647422	0.569126743

the fairness performance of our algorithm is a little bit worse than the performance of TFRC when there are 16 peers, due to the bandwidth resolution (100 kbps) of the video layers which also determines the sending rate resolution in the proposed algorithm. The fair throughput is 625kbps and the sending rate just can be 600 or 700kbps for decision so as to unfavorable in calculating fairness index. However, when the link capacity becomes larger (20 Mbps as shown in Table 2), the fairness index of TFRC suffers a lot.

Fig. 18 shows the fairness index over time with respect to different peer numbers and different shared bandwidths. Obviously, the proposed algorithm is more stable than TFRC no matter in which condition. When one of TFRC flows is suppressed by the other flows, it is rather difficult for it to grow up again.

4.2.4. Fairness over various RTT values

TCP flows usually show different end-to-end throughputs at different RTTs even under the same bottleneck. In the following simulations, we change the link delay of the last segment of one path in Fig. 13 so that there will be different RTTs between two flows and compare the proposed algorithm with TFRC. Each simulation lasts for 1000 s and the RTT of the flow varies from 1 to 5 times with respect to the other one, respectively. The average performance of 10 times of simulations with random start time is shown in Table 3. The proposed congestion control is based on the queuing delay, and the back-off time of the probing periods is related to the transmitted video layers according to (13) whenever a probing fails. The lower layer has better opportunities to catch up with the higher layer because of the shorter probing period when there is extra available bandwidth. Therefore, the unfairness of different RTT on our proposed algorithm is much less than that on the TFRC, obviously. Because TFRC does not emphasize on the inter-session fairness; as a result, one of the TFRC flows usually suppresses the other one and dominates the whole bandwidth resource after a long period of time.

5. Conclusion

In this paper, we propose a bandwidth estimation algorithm using top-down scheme that combines the features of OWD and packet dispersion, and further present a congestion control algorithm for SVC based streaming through the tool of bandwidth inference by periodical probing. In order to compete with TCP congestion control algorithm and not to hamper the existing TCP flows, we dynamically adapt probing periods according to the RTT and the bit rate of video layers to mimic TCP congestion avoidance status. When the packet loss rate is greater than $threshold_{PLR}$, we drop transmission rate according to the sending rate and packet loss rate. In addition, we also observe the changes of RTT to determine whether probing one layer is beneficial. As shown in the simulation results, the proposed algorithm demonstrates better performance than TFRC and it can coexist with TCP flows friendly.

References

[1] C. Dovrolis, P. Ramanathan, D. Moore, Packet-dispersion techniques and a capacity – estimation methodology, IEEE/ACM Transaction on Networking 12 (6) (2004).
[2] R.S. Prasad, M. Murray, C. Dovrolis, K. Claffy, Bandwidth estimation: metrics, measurement techniques, and tools, Network IEEE 17 (6) (2003) 27–35.
[3] M. Jain, C. Dovrolis, Pathload: a measurement tool for end-to-end available bandwidth, in: Proc. Passive Active measurements, Fort Collins, CO, March 2002.
[4] V. Ribeiro, R. Riedi, R. Baraniuk, J. Navratil, L. Cottrell, PathChirp: efficient available bandwidth estimation for network paths, Passive and Active Measurement Workshop 2003.
[5] Q. Liu, J.N Hwang, End-to-end available bandwidth estimation and time measurement adjustment for multimedia QoS, in: International Conference on Multimedia and Expo (ICME), July 2003.
[6] B. Melander, M. Bjorkman, P. Gunningberg, A new end-to-end probing and analysis method for estimating bandwidt bootlenecks, in: Global internet Symposium, 2000.
[7] M. Jain, C. Dovrolis, End-to-end available bandwidth: measurement methodology, dynamics, and relation with tcp throughput, IEEE/ACM Transaction on Networking 11 (4) (2003).
[8] N. Hu, P. Steenkiste, Evaluation and characterization of available bandwidth probing techniques, IEEE Journal on Selected Areas in Communications 21 (6) (2003).
[9] X. Liu, K. Ravindran, D. Longuinov, What signals do packet-pair dispersion carry?, in: Proceedings of IEEE INFOCOM, vol. 1, March 2005, pp. 281–292.
[10] J. Strauss, D. Katabi, F. Kaashoek, A measurement study of available bandwidth estimation tools, in: Proceedings of the 3rd ACM SIGCOMM Conference on Internet Measurement, October 2003, pp. 39–44.
[11] J.L. Lin, S.C. Pei, J.N. Hwang, Fine-grain layered multicast based on hierarchical bandwidth inference congestion control, in: International Symposium on Circuits and Systems (ISCAS), May 2005.
[12] S.S. Wang, H.F. Hsiao, Fast end-to-end available bandwidth estimation for real-time multimedia networking, in: International Workshop on Multimedia Signal Process (MMSP), October 2006.
[13] Joint Video Team, ITU-T Recommendation H.264: Advanced Video Voding for Generic Audiovisual Services, ITU-T Rec. H.264 and ISO/IEC 14496-10 AVC, March 2009.
[14] Q. Liu, J.N. Hwang, A new congestion control algorithm for layered multicast in heterogeneous multimedia dissemination, in: International Conference on Multimedia and Expo (ICME), July 2003.
[16] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, A transport protocol for real-time applications, RFC 3550, July 2003.
[17] S. Floyd, M. Handley, J. Padhye, J. Widmer, Equation-based congestion control for unicast application, in: Proceedings of ACM Special Interest Group on Data Communication (SIGCOMM), May 2000.
[18] J. Widmer, R. Denada, M. Mauve, A survey on TCP-friendly congestion control, in: IEEE Network, vol.15, issue 3, May 2001, pp. 28–37.
[19] J. Padhy, V. Firoiu, D. Towsely, J. Kurpose, Modeling TCP throughput: a simple model and its empirical validation, ACM SIGCOMM 28 (4) (1998).
[20] S.-G. Na, J.-S. Ahn, TCP-like flow control algorithm for real-time applications, in: IEEE International Conference on Networks (ICON), 2000.
[21] I. Rhee, V. Ozdemir, Y. Yi, TEAR: TCP emulation at receivers-flow control for multimedia streaming, Dept. of Camp. Sci., NCSU, Tech. Rep., April 2000.
[22] S. Floyd, M. Handley, J. Padhye, J. Widmer, TCP Friendly Rate Control (TFRC): Protocol Specification, RFC 3448, January 2003.
[23] I. Rhee, L. Xu, Limitations of equation-based congestion control, in: IEEE/ACM Transaction on Networking, vol. 15, no. 4, August 2007.
[24] D-M. Chiu, R. Jain, Analysis of increase and decrease algorithms for congestion avoidance in computer networks, Computer Networks and ISDN Systems 17 (1) (1989) 1–14.
[25] A. Medina, A. Lakhina, I. Matta, J. Byers, Brite, <http://www.cs.bu.edu/brite/>.
[26] D.X. Wei, J. Cheng, S.H. low, S. Hegde, FAST TCP: motivation, architecture, algorithms, performance, IEEE/ACM Transaction on Networking 14 (6) (2006) 1246–1259.
[27] Yingsong Huang, Shiwen Mao, Scott F. Midkiff, A control-theoretic approach to rate control for streaming videos, in: IEEE Transactions on Multimedia, Special Issue on Quality-Driven Cross-Layer Design for Multimedia Communications, vol. 11, no.6, October 2009, pp. 1072–1081.
[28] L. Zhou, B. Geller, B. Zheng, A. Wei, J. Cui, System scheduling for multi-description video streaming over wireless multi-hop networks, IEEE Transactions on Broadcasting 55 (4) (2009) 731–741.
[29] E. Kohler, M. Handley, S. Floyd, Datagram Congestion Control Protocol (DCCP), March 2006.

Further reading

[15] J.-R. Ohm, M. van der Schaar, J.W. Woods, Interframe wavelet coding-motion picture representation for universal scalability, Signal Processing: Image Communication 19 (9) (2004) 877–908.