Bandwidth-Aware Scaling for Internet Video Streaming

Turhan Tunalı[1], Nükhet Özbek[1], Koray Anar[1], and Aylin Kantarcı[2]

[1] International Computer Institute, Ege University
35100, Bornova, İzmir, Turkey
{tunali,ozbek,kanar}@ube.ege.edu.tr
[2] Department of Computer Engineering, Ege University
35100, Bornova, İzmir, Turkey
kantarci@bornova.ege.edu.tr

Abstract. In this study, we propose a bandwidth-aware scaling mechanism for rate adaptive video streaming. This mechanism involves estimation of the capacity of the network dynamically by measuring *bottleneck bandwidth* and *available bandwidth* values. By taking the available bandwidth as an upper limit, the sender adjusts its output rate accordingly. While increasing the quality of the video, using a bandwidth estimator instead of probing prevents the congestion generated by the streaming application itself. The results of the bandwidth-aware algorithm are compared with that of a similar algorithm with no bandwidth-aware scaling and the improvement is demonstrated with measurements taken over WAN.

1 Introduction

In Internet video streaming applications, compressed video streams need to be transmitted over networks that have varying bandwidth conditions. At any time, making best use of available network resources and guaranteeing maximum level of perceptual video quality from the end-user's perspective require utilization of rate control mechanisms in video streaming systems [1]. Over-rating the output of a video streamer can cause an undesirable traffic explosion and can lead to congestion in the network. On the other hand, uncontrolled reduction of the output bit rate of a video streamer leads to unnecessary quality degradation and inefficient use of available bandwidth resources. Therefore, to achieve the best trade-off between quality and congestion, adaptive strategies have to be developed [2, 3].

In many of the studies, rate adaptation algorithms are based on observed variables such as packet loss rate and delay. These variables provide some information about congestion in classical wired networks. However, particularly loss rate can not be a good indicator of congestion in wireless networks. On the other hand, to eliminate jitter, efficient receiver buffer management policies should be developed [4]. Another important parameter that can efficiently be used is available bandwidth. A good estimate of available bandwidth can provide preventive congestion control. However, integration of a bandwidth estimation algorithm into an adaptive video streamer is not an easy task. Firstly, bandwidth estimation requires sending extra burst packets that brings a considerable overhead into the system. Secondly, these burst packets have to be transmitted over the path of the streamer. Finally, to meet real-time limitations of the streaming system, as opposed to the proposed methods in the literature, the bandwidth estimator should be very fast.

In this paper, we propose a bandwidth-aware rate control algorithm which is modified from our previously developed adaptive rate control algorithm that did not contain any bandwidth estimator [5]. In [5], quality increases until loss occurs which means that congestion must have already started. Embedding a bandwidth estimator into our rate control algorithm avoids this congestion that is merely caused by our streamer. Another improvement is optimal selection of video quality during initial buffer filling period. This minimizes number of changes of video parameters which, in turn, improves perceived video quality.

The paper is organized as follows: In Section 2, a brief review of our adaptation algorithm is given. In Section 3, the bandwidth estimator module developed for our algorithm is introduced. In Section 4, our bandwidth-aware scaling algorithm and initial quality detection method are examined in detail. In Section 5, performance results on WAN environment are given. Finally, in Section 6, concluding remarks are made.

2 Rate Adaptive Streaming Algorithm

In this section, we briefly review our previously developed streaming algorithm [5]. Rate adaptation is achieved by video scaling in a seamless manner via frame dropping or switching to another encoding rate or changing the packet interval. The video is encoded in multiple encoding rates and stored in the database. A metadata file is prepared by packetization module for every encoding rate of the video. Packetization module determines the number of packets to be sent per video file for each encoding rate and frame discard level pair. The transmission interval between consecutive packets is calculated as video duration divided by the number of packets for each pair. Packet interval values are also stored in the metafile.

Frame dropping is performed in levels. Considering the dependency among frame types, the adaptation module drops first B frames then P frames if necessary from the current GOP pattern. Each encoding rate (ER) and frame discard level (FDL) pair corresponds to a different transmission rate in the network. Encoding rates have values as 1000, 500, 200, 100 kbps. Frame rates have values as 30, 20, 10, 5, 1 fps. A grid is formed by using ER and FDL combinations. On this grid, depending on the receiver buffer and the network congestion status, the appropriate ER-FDL pair is chosen for adaptation that follows an AIMD (Additive Increase Multiplicative Decrease) strategy to preserve TCP-friendliness.

Table 1. Observed and controlled variables of the algorithm

Observed variables	**Source**	**Controlled variables**	**Source**
Loss rate	RTCP RR feedback	Encoding rate (ER)	Sender
ttp	UDP receiver feedback	GOP pattern (FDL)	Sender
dttp	UDP receiver feedback	Packet interval (PI)	Sender

Table 1 shows the observed and controlled variables of the rate control algorithm. The receiver periodically sends loss rate, current stored video duration, denoted *ttp*, and rate of change of *ttp*, denoted dttp, to the sender. Examining these data and current values of the controlled variables ER, FDL and PI, the algorithm running at the sender decides on the trade off between the video quality and congestion. Further details can be found in [4, 5].

Our algorithm follows a conservative approach by allowing quality degradations after two adaptation requests and quality increases after 5 adaptation requests. We observed that a non-conservative approach that reacts to adaptation requests immediately resulted in frequent rate oscillations, displeasing the viewer. The conservative approach based on a hysteresis model preserved the prevailing quality until an indicator of persistent behaviour in congestion and buffer status is available, thereby eliminating the disturbance of the viewer.

Another property of our algorithm is that it has a content-aware media scaling system. When adaptation is required, quality scaling is used by switching to a version at a lower encoding rate during the transmission of the video segments which contain high motion whereas temporal scaling (i.e. frame dropping) takes precedence over quality scaling during the transmission of the video portions with low motion content.

Since there is no bandwidth estimator in [5], when all goes fine, i.e. the receiver buffer level is within the desired interval and there is no packet loss in the network, the algorithm probes for bandwidth by increasing video quality until it experiences packet loss. Then, it decreases quality but some packets have already been lost in the network. To prevent this negative effect of probing particularly at the initial buffer filling phase, a good estimation of the available bandwidth could be of great help. However, such an estimator may bring in considerable overhead to the streaming algorithm and it may degrade overall performance. In the next section, we will introduce a bandwidth estimator that is suitable for embedding into our algorithm.

3 Estimation of Available Bandwidth

In recent years, there has been significant progress in the area of bandwidth estimation. Several papers related to this area discuss capacity (i.e. bottleneck bandwidth) and available bandwidth estimation methodologies. Among them, Van Jacobson's Pathchar [6], determines per hop based capacity by using variable packet sizes. Another study by Lai and Baker's Nettimer [7], determines end-to-end path capacity via Packet Pairs. These two measure only bottleneck bandwidth. For our purposes, there is another measure called *available bandwidth* which provides better feedback for our streamer. Cprobe [8], Pathload [9], IGI [10] and Pathchirp [11] develop available bandwidth measurement methods.

Cprobe is the first tool to attempt to measure available bandwidth. However, it ignores the fact that bottleneck bandwidth is not same as the available bandwidth. It also requires some privileges on routers such as ICMP messaging. Cprobe is not useful especially nowadays because network administrators are very concerned about attacks based on ICMP messages. For wide area network measurement, it is useful to work with other packet types.

Pathload is based on Self Loading Periodic Streams (SLoPS) methodology. SLoPS consists of K packets of size L, sent to destination at constant rate R. If the stream rate R is higher than the available bandwidth, one way delay of successive packets at receiver shows an increasing trend. By trying different probing speeds, estimation for the available bandwidth can be found. Pathload implementation uses UDP packets that require no privileges on router. Main disadvantage is reported to be low estimation speed [10].

IGI and Patchirp are the new tools that use modified Train of Packet Pairs (TOPP) [11] and SLoPS. They use different packet pair streams and they can achieve similar accuracy in small measurement times [9]. Packet Pair Technique (PPT) is a common approach for bandwidth estimation [12]. Since the approach basically depends on observation of active probing packets' dispersion at receiver, it is also referred to as Packet Pair Dispersion. The active measurement probes are injected to the network by sender for the attention of receiver. By measuring the space changing between consecutive packets at destination, the network path properties can be estimated. The model used in PPT is given in Fig. 1 [13].

Our Packet Pair Model is similar to IGI/TOPP model except that we send single train of packet pairs in each experiment and use small constant initial gaps rather than increasing amount of initial gaps. This is due to the fact that determining a good value for the initial probing gap is much more difficult. It is possible to determine a better gap value by making several experiments in which a sequence of packet trains with increasing initial probing gaps is sent. However, these experiments take some time, and it may be possible to miss instant available bandwidth. Another important property in our algorithm is that it is 2-5 times faster than IGI method in estimating bandwidth.

In each experiment, the sender injects probe packets into network at regular intervals denoted by ΔTs_n and the destination receives them in intervals denoted by ΔTr_n. To get more accurate results, packet pair train is formed from fixed-size packets represented as P.

If there is no additional traffic on the path, bottleneck bandwidth is also called "path capacity" that is the maximum throughput that the path can provide. We denote it as B_BW. Bottleneck bandwidth should not be confused with the available bandwidth of a path. Available bandwidth is the redundant capacity that is not used by the existing traffic.

We assume that there is no cross traffic in the path and the complete path is formed of K links. We measure the smallest bottleneck bandwidth of the present links [12]:

$$B\_BW = \min_{i=0\ldots K-1} \{B\_BW_i\} . \qquad (1)$$

Amount of competing traffic ratio represented by α is the ratio of increased gaps to received gaps. Motivated by [10], the competing traffic represented by C_BW is given as

$$C\_BW = B\_BW * \alpha . \qquad (2)$$

Finally, we estimate the available bandwidth by subtracting competing traffic throughput from bottleneck bandwidth,

$$A\_BW = B\_BW - C\_BW . \qquad (3)$$

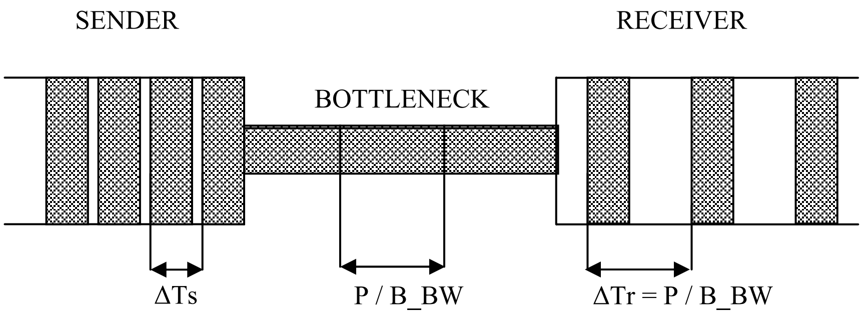


Fig. 1. Packet Pair Model without any competing traffic

4 A Bandwidth-Aware Streaming Algorithm

We developed and implemented a bandwidth-aware streaming algorithm that can efficiently operate on both local and wide area networks. This algorithm is integrated to our previously developed rate adaptive algorithm that utilizes the loss statistics and receiver buffer status.

First, the channel bandwidth estimator module is implemented stand-alone and its performance is optimized to meet our streaming environment requirements. Then the module is embedded into the rate adaptive streaming algorithm. Combining available bandwidth estimation with rate adaptation has two important advantages. First, available bandwidth is used to decide on the initial video quality to be sent during pre-buffering period. The estimated bandwidth value lets the algorithm choose the most appropriate encoding rate to start streaming. Hence, heavy load during initial buffer filling period is avoided. Second, when the quality is to be increased, avaliable bandwidth is very useful to understand whether the current channel capacity meets the new bandwidth requirement. By this way, the unnecessary quality increases in the adaptation algorithm, which may cause packet loss and congestion, are avoided.

Depending on the observed measures of loss rate, ttp, dttp, video dynamics and current values of ER-FDL-PI, the algorithm determines new values for ER-FDL-PI. If conditions impose a decrease in quality, than no further step is taken and new values of ER-FDL-PI are applied to streaming. Since RTCP reports are at least as fast as our bandwidth estimator in informing the congestion to the sender, the estimated bandwidth value is not used when the quality is to be decreased. Hence, no advance information is available in this particular case.

If new adaptation is not in the direction of decreasing data rate, than new proposed data rate (*put_bw*) is compared with estimated available data rate. If *put_bw* is less than the available bandwidth, then the chosen ER-FDL-PI is applied to streaming. If *put_bw* is more than available bandwidth, the current ER-FDL-PI values remain the same. This last case occurs when the streaming application is already using up a hefty bandwidth and there is no more bandwidth available for quality increase. A sketch of the algorithm is given in Fig. 2 where the procedures *down_scale_video* and *up_scale_video* correspond to the original adaptation algorithm [5] with rate decrease and increase respectively.

It is important to note that the bandwidth estimator module is run at the receiver and it operates in full synchronization with other modules of the algorithm. In particular, the observation parameters and estimated available bandwidth are updated every five seconds and the decision process at the sender is not delayed by the bandwidth estimator module.

```
bandwidth_aware_scale( )
{
    observe loss_rate, ttp, dttp, video_dynamics, available_bw;
    if (congestion)
        down_scale_video(loss_rate, ttp, dttp, video_dynamics);
    else
        compute put_bw;
        if (put_bw < available_bw)
            up_scale_video(ttp, dttp, video_dynamics);
        else
            no_scale;
}
```

Fig. 2. Bandwidth-aware video scaling algorithm

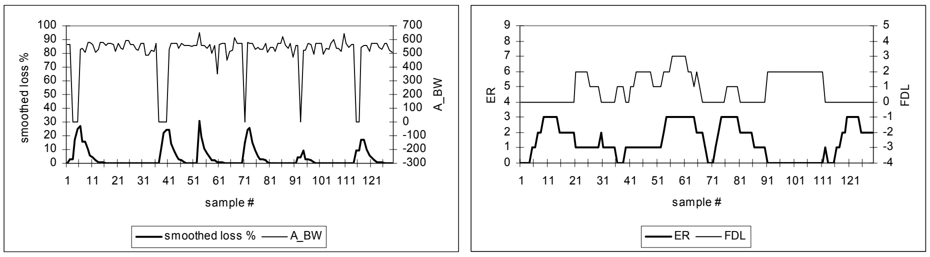
5 Experimental Results

The bandwidth-aware scaling algorithm introduced in the previous section has been tested with the video streaming system that we have implemented. In the system, RTP has been used for data transfer and RTCP has been used to collect network statistics. Control messages are exchanged over UDP. Our streaming system has client-server architecture. The clients request video from the server. Server streams video to the client in a unicast manner. Both the server and client software are multithreaded. Pipelined architecture of the client software further increases the performance of the whole system. Detailed explanation of our testbed can be found in [4, 5].

Experiments have been performed in the actual Internet environment between two Sun Ultra 5 workstations. The workstation which acts as the streaming server is located in Koc University Campus in Istanbul. The client workstation is located in Ege University Campus. *Traceroute* command shows that the number of hops between the two workstations is 9.

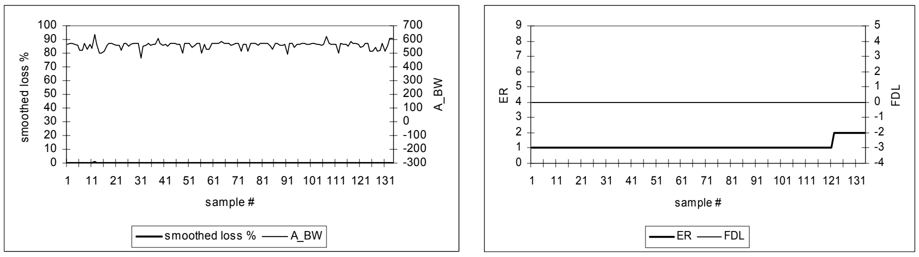
We carried out two sets of experiments to observe the performance of our bandwidth-aware scaling algorithm. The first set of experiments was carried out late at night when the amount of traffic on the network was low, while the second set of experiments was conducted during busy working hours when the load was higher. In both sets of experiments, to be able to evaluate the benefits of our bandwidth-aware scaling algorithm, we collected performance results under two configurations. In the first configuration, bandwidth-aware scaling is turned off, while in the other, it is turned on.

Results presented in Fig. 3 and Fig. 4 which belong to an example in the first set of experiments. Figure 3 shows system variables when adaptation is performed without bandwidth considerations. As seen in Fig. 3.b, during pre-buffering period, transmitted video quality can reach its highest level in which the encoding rate is 1000 kbps (ER = 0) and the frame rate is 30 fps (FDL = 0). However, the estimated available bandwidth is around 550 kbps. Therefore, our system overloads the network, available bandwidth decreases sharply and congestion develops soon after streaming begins. Our adaptation module decreases video quality by following our content-aware scaling algorithm. As the video quality decreases, congestion alleviates and available bandwidth increases. As a consequence, the adaptation module increases video quality by adjusting the encoding rate and frame rate. At sample=35, encoding rate rises to 1000 kbps. This bit rate overloads the network and congestion develops again. In response, the adaptation module decreases video quality again. This cycle repeats itself throughout the transmission, generating a self-similar traffic in the network. Frequent quality switches have been observed as the streaming proceeds.



(a) (b)

Fig. 3. System variables in lightly loaded network without bandwidth aware scaling



(a) (b)

Fig. 4. System variables in lightly loaded network with bandwidth aware scaling.

Fig. 4 shows system variables when our bandwidth-aware scaling algorithm is applied. Prior to streaming, initial network bandwidth is estimated and proper quality level is determined by comparing available bandwidth with encoding rates of pre-encoded streams. According to Fig. 4a, available bandwidth is between 500 and 600 kbps and the highest encoding rate that is closest to the available bandwidth is 500 kbps. Therefore, transmission starts with this encoding rate (500 kbps) with a frame rate of 30 fps (FDL = 0). Before switching to a higher quality video, our streaming system calculates the bit rate requirement and compares this value with the available bandwidth. Since the available bandwidth does not exceed 600 kbps, we observed that the adaptation module does not increase video quality. Since this experiment is conducted late at night when the Internet is lightly loaded and our system transmits video packets at a rate in compliance with the available bandwidth, no loss has been conveyed through RTCP reports and transmission proceeds at the initial video quality throughout the experiment, without invoking the adaptation module. At sample=120, buffer level has fallen below a predetermined threshold. The decrease in buffer level has been compensated by decreasing the video quality to 200 kbps. Low quality video results in less number of packets in the network leading to fewer packets with smaller end-to-end delays which increases input rate to the buffer, increasing the buffer occupancy in turn.

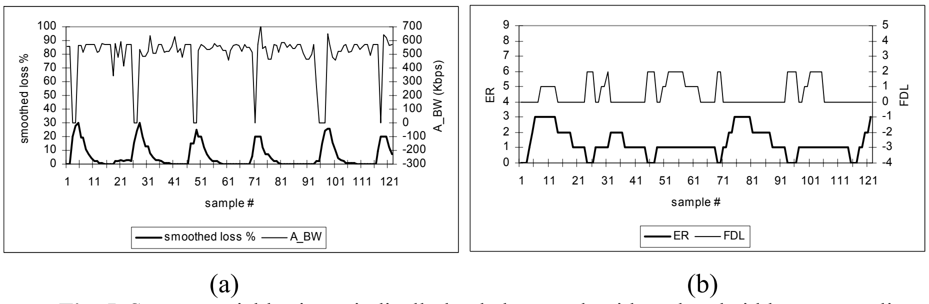


Fig. 5. System variables in periodically loaded network without bandwidth aware scaling.

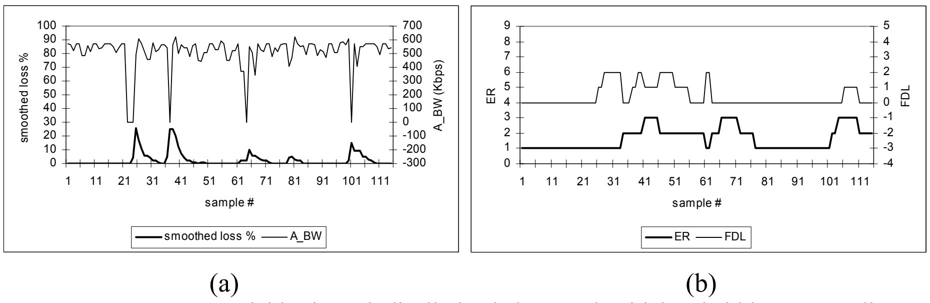


Fig. 6. System variables in periodically loaded network with bandwidth aware scaling.

Figure 5 and Fig. 6 demonstrate the behavior of our system in a periodically loaded network. When these figures are examined, it is seen that the available bandwidth has more variation than it had in lightly loaded network scenarios. This is because of the fact that, compared to the lightly loaded case, the competing traffic on the Internet is varying considerably in this case. Figure 5 shows the status of system variables when the bandwidth-aware scaling algorithm is switched off. Behavior of the system is very similar to the case given in Fig. 3. Figure 6 presents the performance results when bandwidth-aware scaling is applied. We observed that the number of congested intervals is reduced when bandwidth is taken into account in rate control decisions. When quality changes given in Fig. 6.b are examined, it is seen that congestion is caused by the traffic generated by the other sources on the Internet rather than by our streaming software. This is best observed during the initial buffer filling periods.

Upon detecting the congestion, our adaptation module reacted properly by decreasing video quality. When Fig. 5.b and Fig. 6.b are compared, it is seen that inclusion of the bandwidth estimation module decreased the number of quality changes, resulting in more stable video quality settings. Additionally, overall quality was higher when bandwidth aware algorithm was applied. For example, the encoding rate was decreased to the worst level (ER = 100 kbps) at sample = 6 in Fig. 5.b. On the other hand, prevailing encoding rate was 500 kbps until sample = 33 in Fig. 6.b, thanks to the employment of initial bandwidth estimation procedure.

To summarize, experimental results justify our hypothesis that bandwidth awareness may prevent the occurrence of congestion due to probing and reduces the effects of congestion due to the traffic generated by the external sources. It also reduces the quality fluctuations by avoiding unnecessary invocations of the adaptation module. Similarly, initial bandwidth detection eliminated congestions at the start of the streaming process. Finally, overall perceptual quality was higher with bandwidth-awareness, resulting in a positive effect on the disturbance of the viewer.

6 Conclusions

In this study, we developed a bandwidth-aware streaming algorithm and reported performance results. A mechanism is proposed to measure available bandwidth in the network. By using Pair Packet Dispersion technique, the channel bandwidth is estimated. Our adaptive streaming system checks available bandwidth before performing any increase in quality and/or transmission rate. We have particularly observed that our algorithm is effective during initial buffer filling period. We compared test results of the algorithms with and without bandwidth estimator, which are both taken in WAN environment. It has been shown that the bandwidth-aware streaming algorithm does not allow packet loss and congestion due to the quality increase. However, it can robustly react to the congestion caused by other factors while maintaining acceptable and interrupt-free perceptual quality.

Acknowledgement

The authors are indebted to Koc University for allowing remote use of their workstations while onducting streaming experiments.

References

1. Wang, L.: Rate Control for MPEG Video Coding. Signal Processing: Image Communications **15** (2000) 493–511
2. Girod, B.: Scalable Video for Multimedia Systems. Computers & Graphics **17**(3) (1993) 269–276
3. Sadka, A.H.: Compressed Video Communications. John Wiley & Sons, Ltd. (2002)
4. Tunalı, T., Kantarcı, A., Özbek, N.: Robust Quality Adaptation for Internet Video Streaming. Accepted to be published in Journal of Multimedia Tools and Applications. Kluwer Academics.

5. Kantarcı, A., Özbek, N., Tunalı, T.: Rate Adaptive Video Streaming Under Lossy Network Conditions. Elsevier Science Image Communications **19**(6) (2004) 479–497
6. Jacobson, V.: Pathchar-a Tool to Infer Characteristics of Internet Paths. Presented at the Mathematical Sciences Research Institute (MSR1). (1997)
7. Lai, K., Baker, M.: Nettimer: A Tool For Measuring Bottleneck Link Bandwidth. In: Proceedings of the USENIX Symposium on Internet Technologies and Systems. (2001)
8. Carter, R.L., Crovella, M.E.: Measuring Bottleneck Link Speed in Packet-Switched Networks. Performance Evaluation **27**(28) (1996) 297-318
9. Jain, M., Dovrolis, C.: Pathload: A Measurement Tool for End-to-end Available Bandwidth. In: Proceedings of Passive and Active Measurements (PAM) Workshop. (2002)
10. Hu, N., Steenkiste, P.: Evaluation and Characterization of Available Bandwidth Probing Techniques. IEEE Journal on Selected Areas in Communications **21**(6) (2003)
11. Ribeiro, V., Riedi, R., Baraniuk, R., Navratil, J., Cottrell, L.: PathChirp: Efficient Available Bandwidth Estimation for Network Paths. In: Proceedings of Passive and Active Measurements (PAM) Workshop. (2003)
12. Dovrolis, C., Ramanathan, P., Moore, D.: What Do Packet Dispersion Techniques Measure? In: Proceedings of IEEE INFOCOM. (2001)
13. Jacobson, V., Karels, M.J.: Congestion Avoidance and Control. In Proceedings of the ACM SIGCOMM Conference. (1988)