

How to measure available bandwidth on the Internet

Manthos Kazantzidis

Technical Report #010032

Advisor: Dr. Mario Gerla

UCLA CS WAM Lab

Abstract: Available bandwidth is the most useful measurement to network adaptive applications and transports. Unfortunately, it is also very difficult to measure in a network that cannot be well approximated by a weighted fair queuing model, such as the Internet. At the same time, the Internet scalable architecture calls for end-to-end measurements. Current available bandwidth measurement techniques are based in observing packet dispersion in a packet train or pair. The available bandwidth is sampled by using a “bytes divided by dispersion” (or “bytes over time”) calculation and then filtered. In this paper, we deal with sampling the available bandwidth from packet dispersion. We argue that existing techniques, being heuristics, are unintuitive, are not based on any network model, and introduce an error in the process right from the sampling. We propose a different calculation of the available bandwidth from packet dispersion observation, we call *ab-probe*. It is also simple and as robust, and follows a simple network model. It is an improvement to the “bytes over time” calculation because it also captures and deducts a portion of the argued sample calculation error. We study this new model by exploring the differences between the two methods and its robustness to parameter estimation errors. We then apply our model to a well known available bandwidth measurement technique called *cprobe*. We find that our model explains *cprobe*’s behavior as recorded by its researchers’ published real experiments. We further validate the measurement using simulations in some simple wired scenarios.

I. INTRODUCTION

Measuring available bandwidth on the Internet is difficult but particularly interesting. If available bandwidth can be measured with adequate accuracy it can be used to improve network performance. It would do so by improving the flow control in network-adaptive applications e.g. multimedia content adaptation, dynamic server selection and congestion control transports. At the same time it is hard to measure available bandwidth, at least when a network model that is close to Internet reality is required [3]. An active heuristic technique called *cprobe* [1] is now widely used to measure available bandwidth, mainly for server selection purposes. Passive measurements of available bandwidth using packet dispersion techniques are also seen in transports. We indicatively refer to the sender based packet pair equivalent in the faster recovery mechanism proposed in TCP Westwood [6], and Microsoft’s video streaming protocol MSTFP [8]. Packet pair may measure the bottleneck link bandwidth in FIFO queuing networks by using, for example, the filter proposed in [2]. Its result is the available bandwidth only in WFQ networks [7]. Therefore, those measurements are only as accurate as the

weighted fair queuing model approximates the Internet. It is conceivable, however, that the traffic at the bottleneck (or minimum available bandwidth) link statistically approximates the competing traffic from a fair queue (for example, think of a single link connection accepting traffic from many flows). This is the idea behind using sender or receiver based packet pair to measure available bandwidth. In [5] it is shown that such techniques converge to the Asymptotic Dispersion Rate that is lower than the capacity but is not the available bandwidth in the path. Our work is in agreement with this study. We note, however, that the classic bytes-over-dispersion model assumed in previous work can be improved, in fact replaced with a different more intuitive available bandwidth sample calculation. This different calculation affects the filtering part of the measurements too. New filtering mechanisms that take into account the sample behavior will be more successful in estimating available bandwidth from the new samples.

A rather sophisticated filtering mechanism uses packet pair for bottleneck link measurement in the tool *nettimer* [2]. This work has progressed from [3] and has also produced [4]. The latter is a technique that uses a train of packets to measure multiple bandwidths along a path. *Nettimer*, uses a kernel density estimator in order to statistically compute the bottleneck link bandwidth from a collection of samples. It finds the highest density point by effectively dealing with well-known problems arising from constant bin size histograms. The basic assumption under which this estimation is correct, is that more packet pairs end up ‘revealing’ the bottleneck link bandwidth than any other point. While this is mostly true, since the bottleneck link bandwidth is a sustaining quantity, it requires a large number of good samples. Unfortunately, few applications are directly interested in bottleneck link bandwidth. A bottleneck link bandwidth measurement is needed however for all current available bandwidth estimation techniques.

We see that packet pairs and trains are in the epicenter of measurements, while measurement method differences lie in the filtering mechanism. For example, when a packet dispersion mechanism is used for available bandwidth estimation, the filtering has an averaging effect [1], rather than discovering modes [5] or densities [3]. In all mechanisms the samples are computed using the simple bytes-over-dispersion formula at the observer side. In this paper, we suggest a different equation to compute the available bandwidth samples. Our equation is not a heuristic and is independent of a specific queuing discipline. The equation is somewhat more

complicated than the simple division, but produces samples that are closer to the actual available bandwidth.

In the next section we analyze the packet pair behavior link by link. In section III we present the new sample calculation and deal with its robustness in IV. In V we present some basic experiments that are in keeping with the theoretical models and conclude in VI.

II. PACKET PAIR BACKGROUND

In this section, we qualitatively analyze the source of errors of an end-to-end packet pair based measurement. An end-to-end available bandwidth measurement method using our available bandwidth sample calculation will have all these errors too. Traditionally advanced filtering is used, to deal with these errors.

We present this, focusing separately on a path's three distinct sub-paths: 1) the path from the source up to right before the bottleneck link, 2) the bottleneck link, 3) the path from right after the bottleneck link to the destination. Let us assume

- A path of N hops, with the bottleneck at hop k
- S_b , the sender bandwidth, determined by the initial packet separation (dispersion) and size
- Pb_i , the potential bandwidth after hop i , determined by the separation after hop i
- B_i , the actual bandwidth of link at hop i

In what follows we look at what may happen at a packet pair on each link in the three sub-paths. As is known, the potential bandwidth (packet size/time) with which the pair arrives at a link defines whether the pair is able to sample the link's bandwidth and available bandwidth (the potential bandwidth needs to be higher than the link bandwidth, otherwise it may not be affected by the link).

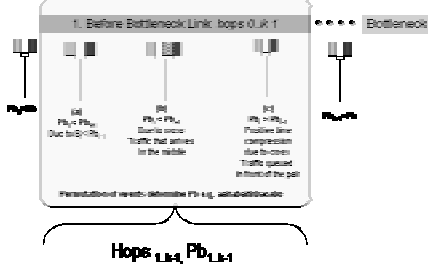


Figure 1. Possible events in links before the bottleneck link.

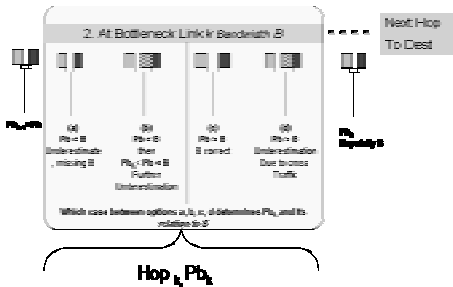


Figure 2. Possible events at the bottleneck link

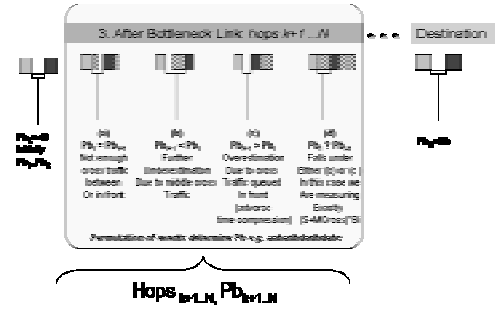


Figure 3 Events occurring after bottleneck link

Figure 1 shows the different events that may occur at a link before the bottleneck link. Note that a combination of events may occur at the same one link, but with no loss of generality we can consider them to occur in separate equal bandwidth links. This part of the path is interesting because it ultimately defines the potential bandwidth (Pb_k) with which the pair will enter the bottleneck link (k). This is determined by a permutation of possible events a, b and c as depicted in Figure 1. After each link the pair may decrease its potential bandwidth when $B_i < Pb_{i-1}$ (event a). The same can happen when cross traffic arrives in the time between the pair arrivals (event b). (Note that the probability of (b) occurring is proportional to $1/Pb_{i-1}$ for a given network load.) When enough in-front queued cross traffic exists at packet pair arrival *useful time compression* may occur. This happens when the its transmission time of already queued packet causes the pair to be queued. We call it useful because in this case the exiting potential bandwidth is increased.

Figure 2 shows the possible events for a packet pair entering the bottleneck link. The packet pair arrives with a Pb_{k-1} and exits with $Pb_k (= Pb)$, hopefully equal to $B_k (= B$ the bottleneck link). If Pb is less than B then we already have an underestimate of B , which is either sustained (a), or further underestimated from intercepting cross traffic (b). If Pb is more than B then the exiting potential bandwidth is either B or some underestimation due to intercepting cross traffic (d).

Figure 3 shows events occurring after the bottleneck link. As the pair is traveling, enough intercepting cross traffic may cause further underestimation over a link (case (b)) or traffic queued in front of it will cause time compression (if service (transmission) time is more than the current packet separation). This is an adverse time compression that causes overestimation. Event (d) is the combination of (b) and (c).

The packet pair size defines the necessary time separation in order to achieve a potential bandwidth. Cross traffic problems mentioned above (1b, 2b, 2d and 3b) are dependent on the separation of the packets. Therefore the larger the packet size used, the smaller the interference from cross traffic. The time compression events (1c and 3c) depend less on the packet pair size.

III. AVAILABLE BANDWIDTH SAMPLING

A. Why we need a new method to calculate the samples?

For available bandwidth measurement, the most known practical active technique is called *cprobe* [1]. It proposed

measuring available bandwidth by sending a train of packets at a rate higher than the bottleneck link, and then dividing the bytes sent by the time separation between the first and the last packet of the train at the receiver.

But let us consider a train of 3 packets (1,2 and 3) entering the bottleneck link at rates close to B (the bottleneck link bandwidth). Packet 1 denotes the beginning of time in the measurement. It is the time it enters for service. Neglecting packet sizes for now, it will take $1/B$ time to finish service. During that time packets from other flows are arriving and are queued. Exactly after $1/B$ Packet 2 arrives. It is queued, if at least one packet arrived between 1 and 2. After another $1/B$, at time $2/B$, Packet 3 arrives and denotes the end of time measurement.

Now consider a train of N ($N \geq 2$) equally spaced packets each of size S bits. Assume the packets reach the bottleneck link with a potential bandwidth of P_b , that the bottleneck link bandwidth is B and the actual available bandwidth (during the train) at the bottleneck link is A . The interval that this train will sample begins when packet 1 of the train enters for service (transmission) and ends when packet N is received and possibly queued. This interval is

$$\text{Interval Sampled} = \frac{(N-1)S}{P_b}$$

Equation 1

If the available bandwidth during this interval is A then, using Equation 1, the traffic (in bits) we should receive from other flows during this interval is

$$\text{Cross Traffic bits} = \frac{(B-A)(N-1)S}{P_b}$$

Equation 2

Then the final dispersion (separation) between Packet 1 exiting service and Packet N exiting service is the transmission time of the packet train bits and the cross traffic bits through the link.

$$T = T_N - T_1 = \frac{NS}{B} + \frac{(B-A)(N-1)S}{P_b B}$$

Equation 3

If this dispersion is maintained (see section B) then this is the T sampled at the observer side. The observer side will then measure the available bandwidth as $(N-1)*S/T$. However, as we have assumed the available bandwidth is A . Note that there is a diversion between what the bytes-over-time calculation (e.g. *cprobe*) obtains as the available bandwidth sample and what the actual sample A should be. We draw this error for a few cases in Figure 5. It shows the error in the bytes over time calculation with relation to Equation 3. We set A to different values for a 1Mbps, calculate the dispersion from Equation 3 and draw the available bandwidth samples calculated with *cprobe* and its absolute error from the assumed A . We do that for a different number of packets for the packet train. The error grows very large (to a relative error of 100%) as the load is higher. We also note that the error (and the measurement) is different for packet trains consisting of different number of packets. This has an adverse impact in applications and transports that use this model. For these reasons, a new model is needed.



Figure 4. Available bandwidth measurement at the bottleneck link.

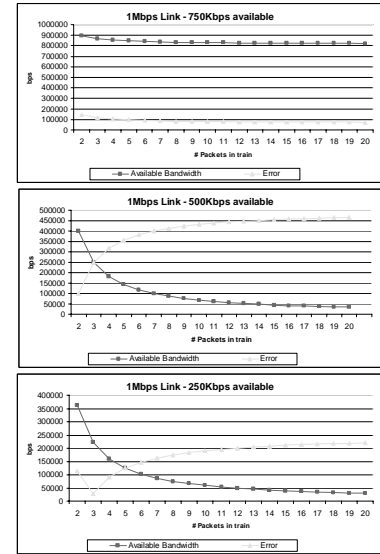


Figure 5. Cprobe's available bandwidth estimation and error for a 1Mbps and 1-top:750Kbps available, 2-middle: 500Kbps, 3-bottom: 250Kbps available

B. The *ab-probe* model

The *ab-probe* model uses Equation 3 to estimate the available bandwidth. Available bandwidth is the desired unknown quantity, while T is the observed dispersion. Therefore we solve for available bandwidth, and find that we should calculate it as:

$$A = B - \frac{P_b(B * T - N * S)}{(N-1)S}$$

Equation 4

It follows that Equation 4 should be used for available bandwidth sample estimation based on the observed packet dispersion. It replaces the simple form $N*S/T$.

Note that our calculations are in keeping with the findings of the *cprobe* researchers' experiments. In low loads the measurement has a relative error up to 20% (see Figure 5 top). In the real testbed experiments they performed, they observed an unexplained attribute of their measurement. As the packet train consisted of an increasing number of packets, *cprobe* would measure a lesser available bandwidth even though the traffic load remained the same. This can be seen in all graphs in Figure 5. This means that our model can capture this *cprobe* behavior and explain it. We draw this effect clearer in Figure 6. This fact provides an additional level of confidence that our model is correct and has captured a part of the sampling error

in available bandwidth techniques. In Figure 7 we draw the diversion between the two models for a 1Mbps link.

Last, let us show a generalized form of Equation 4 that is appropriate for passive measurements (when packets in the packet train have different Pb_i s and sizes S_i)

$$A = B - \frac{T - \frac{1}{B} \sum_{i=1}^N S_i}{\sum_{i=1}^{N-1} \frac{S_i}{Pb_i}}$$

Equation 5

As the packet train travels towards its destination it undergoes the distortions we described in the packet pair section and appropriate filtering of the resulting samples should be used.

IV. ROBUSTNESS

A potential problem of the new sampling method could be its robustness in face of errors in potential bandwidth and bottleneck link bandwidth estimation. In this section, we show that the method is in most cases at least as robust as the bytes-over-time method.

In our measurement both the potential bandwidth entering the bottleneck link and the bottleneck link bandwidth, take part in the actual calculation of the available bandwidth. In the bytes-over-time model they are only used to ensure that the packet train pairs are sent at rates that exceed the bottleneck link bandwidth. In practice, one can use the sender's potential bandwidth, or the bottleneck link bandwidth, in place of the potential bandwidth entering the bottleneck link. Therefore we should examine our method's robustness to estimation errors of the potential bandwidth.

Let us for example look at the case of a measurement over a 1Mbps bottleneck link with 750K available that uses a sender potential bandwidth of 1.1Mbps with 8 packets of 160 bytes. We investigate a relative error in the actual potential bandwidth right before the bottleneck link from -99% to +200%. For each case we calculate the actual interval to be sampled at the bottleneck link (by taking into account the potential bandwidth error) as given in Equation 1. Then, we go on calculating the cross traffic bits (Equation 2) and the packet dispersion seen at the exit of the bottleneck link (Equation 3). Assuming this will be retained until observation, we estimate the absolute difference between the actual available bandwidth (750Kbps) and the estimated according to the two models. Figure 8 shows that the new model is even more robust than the bytes-over-time model for a relative error of -20% to +200%. We notice that with severe underestimation of the potential bandwidth our method's error is not bounded whereas the bytes-over-time is. Practically, an underestimation of less than 20% may be avoided and/or filtered out.

Next we draw the potential bandwidth relative error values above which our method is more robust than the bytes-over-time method. Figure 9 shows that in all cases only with severe underestimation of bottleneck link and potential bandwidth we get larger errors in our method as compared to the bytes-over-time method. These errors can potentially be filtered out (see Figure 8).

A. Stability and TCP Friendliness

In this paragraph we study the use of the measurement in a transport protocol. We assume per RTT updates and minimum available bandwidth reporting on the round-trip path in order to have the potential to be TCP-friendly. We show how a multimedia transport based on the measurement to perform flow control can be TCP friendly, and that such a transport is stable.

A flow is TCP friendly if its throughput is conformant with a TCP connection throughput in similar circumstances. This means that its throughput should be less than $\frac{1.5\sqrt{2/3} * B}{R * \sqrt{p}}$

where B is the segment size, R the roundtrip time and p the loss rate. An equivalent expression is that the throughput of a steady state TCP connection should be $.75 * W * B / R$. This is actually obtained before replacing the upper bound of the window as a function of p .

Recent important work ([Bansal-Infocom2001]) introduces the class of window-based binomial algorithms as those with increase/decrease functions such as

$$I : w_{t+R} \leftarrow w_t + a / w_t^k, a > 0$$

$$D : w_{t+\delta R} \leftarrow w_t - \beta w_t^l, 0 < \beta < 1$$

In fact TCP's AIMD falls into this category when $k=0$ and $l=1$. All algorithms with $k+l=1$ converge to fair allocation and are known to be TCP-friendly.

Using the above result we can choose a k and l that results in less abrupt flow control and is therefore suitable for multimedia. In our experimentation we use the SQRT algorithm that uses $k=0.5$ and $l=0.5$.

Our problem then is twofold. First, we need to have a rate-based equivalent of the above equation and second, use a TCP friendly adaptation mechanism that does not rely on loss rate, but on our available bandwidth measurement. Such an algorithm will satisfy our requirements for delay and delay-jitter sensitivity and performance over heterogeneous networks. We develop the following algorithm based on the notion that in TCP steady state the window is the delay-bandwidth product. Our target TCP-friendly throughput then should be

$$\frac{0.75 * W * B}{R} \approx 0.75 * AB, \text{ when } W * B = R * AB$$

where AB is the available bandwidth measured. If our current average sending rate is L then we may use the following equation on each RTT (since we do get a new sample on each RTT)

$$I : w_{t+R} \leftarrow w_t + a / w_t^k \quad \text{when } L < .75 * AB$$

$$D : w_{t+\delta R} \leftarrow w_t - \beta w_t^l \quad \text{when } L > .75 * AB$$

Now assume our stream is encoded into N layers of L_i average throughput each. Then those define N points, used as 'watermarks' for the value of w

$$(L_1, L_2 \dots L_N) \text{ maps to}$$

$$(W_1 = L_1 * R, W_2 = L_2 * R \dots W_N = L_N * R)$$

The above 'watermark' points allow us to choose a layer by keeping a 'fluid' value of w . When w becomes closer to a new watermark, a layer switch is attempted at the server. Note, that the above algorithm may result in layer add/drop oscillation if hysteresis is not applied. Hysteresis effectively inserts a zone of

no action. We do not deal with this type of oscillation in this paper.

V. EXPERIMENTS

We now perform simulation experiments in order to see whether the theoretical results presented actually agree with a 7-layer vertical simulation (using NS). The results presented here should be cross-referenced with Figure 7, which summarizes the two models errors and diversion.

In the simple topology of Figure 10 the central link is the bottleneck link with 1.1Mbps (10ms propagation). Other links are 5Mbps (5ms prop.). All queues are FIFO. The experiments last for 66 seconds. Audio/CBR connections of 45,863 bps, from S3 to R3 and from S2 to R2, are initiated after 20.6 seconds and last up to 46.6 sec. Each starts at a random time within 21msec after the start time to avoid synchronization effects. S1 throws probing traffic to R1 and samples the bottleneck link bandwidth from the packet pairs and the available bandwidth using packet trains of 8 packets of 480 bits. The latter is calculated using both the bytes over time method and ours, *ab-probe*. The probing traffic is sent at rates of 900Kbps and for the purposes of this experiment is sent continually.

At first, 3 audio connections are used. Figure 11 shows the *ab-probe* samples for a few seconds. Some samples have measured the bottleneck link bandwidth, some arrived unaffected at their potential bandwidth, and others have measured some effect of the cross traffic. (Note that the bottleneck link bandwidth of 1.1Mbps could be measured with a 900Kbps potential bandwidth due to queuing right before the bottleneck point i.e. useful time compression as in section II). We then simply arithmetically average these samples over different, large enough intervals and report typical values. We do that in order to directly compare the sampling methods, not affected by filtering. The available bandwidth with *ab-probe* samples is approx. 953,173 bps. The bytes-over-time method has a more canonical sample distribution and gives 1,013,229 bps. Note that the correct value is 1.1Mbps minus 137,589 bps, which is 962,411 bps. In this low load case *ab-probe* performs better and *c-probe* gives reasonable results, as our theoretical models agree.

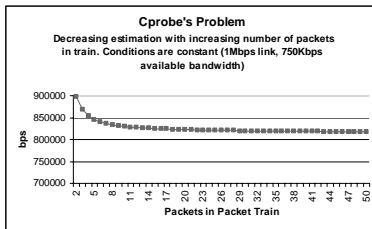


Figure 6 Cprobe's theoretical error (absolute) and decreasing measurement problem.

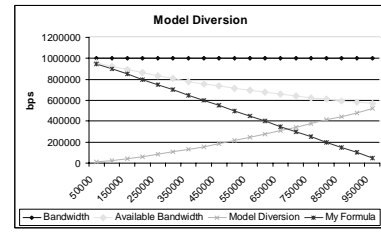


Figure 7 The diversion between the bytes/time and ab-probe for different consumed bandwidths (x axis).

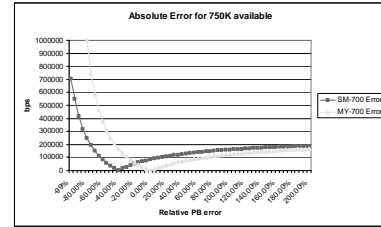


Figure 8. How does overestimation/underestimation of the potential bandwidth entering the bottleneck link affects the two models in the case of 1Mbps link and 750K available.

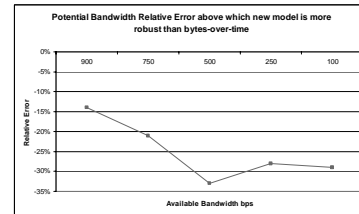


Figure 9. Potential Bandwidth relative errors above which the new method is more robust than the bytes-over-time

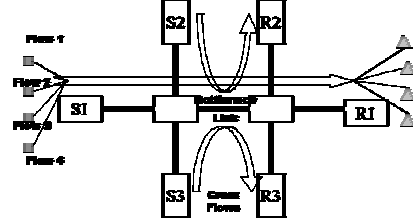


Figure 10. Basic topology for wired experiments

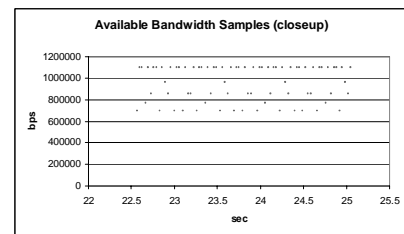


Figure 11. *ab-probe* samples for the first experiment.

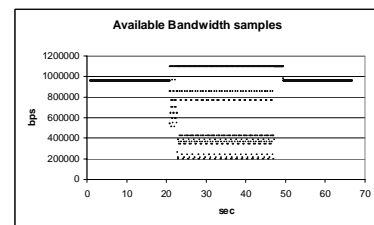


Figure 12. *ab-probe* samples for experiment 2.

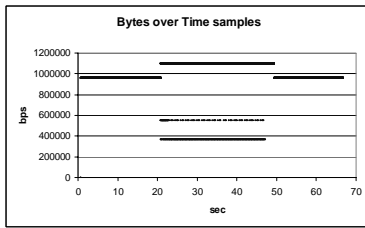


Figure 13. bytes-over-time samples for experiment 2.

Next we move to middle loads, throwing in 7 audio connections consuming now 321,011 bps and leaving 778,959 bps available. The *ab-probe* samples throughout the experiment are shown in Figure 12 and the bytes-over-time in Figure 13. The averaging this time gives 774,339 bps for *ab-probe* and 1,043,258 bps for *c-probe*. This is in accordance to our theoretical models. In higher loads the error of *c-probe* is higher (see also Figure 7).

Last, we throw in 20 audio connections, which should consume 917,260 bps. In this case however, we have packet drops before the bottleneck link (consider also the intense probing traffic). We monitor those packet drops and find that the consumed bandwidth in the bottleneck link by the 20 connections is 725,236 bps leaving 374,764 bps available. *ab-probe* measured typically a 334,355 bps and bytes-over-time measured 651,286 bps, which is very much in keeping with our model.

VI. CONCLUSIONS

We have presented a new way to calculate available bandwidth samples from packet dispersion techniques. The sampling can be used in active and passive measurements, server selection applications, routing, end-to-end transports, adaptive multimedia etc. The motivation has been the unintuitive current methods for measuring available bandwidth as used in the above. We compared our calculation with the common bytes-over-time calculation and found that the latter reports reasonably only at low to low-middle loads. Our calculation performs accurate sampling throughout the range of available and bottleneck link bandwidths. It behaves differently in face of errors in bottleneck link bandwidth estimation but is actually more robust than the bytes-over-time when the relative error in bottleneck link bandwidth is over -20% . The sample calculation is derived from a model that captures and explains *c-probe* behavior as seen by its researchers in measurements over a real network. Basic simulation results have been obtained that further validate our theory. The simulation results have been presented using simple averaging over different starting and ending points and intervals so as to leave out the effects of advanced filtering. The fact that our sample calculation reacts strongly to very large underestimation of bottleneck link calculation makes it easily filterable and calls for new filtering techniques.

Accurate available bandwidth measurements can be very useful to network stability and congestion control, especially for multimedia transports. Combining the sample calculation with filtering techniques developed based on their observation in the Internet is our immediate future direction.

REFERENCES

- [1] R. L. Carter, M. E. Crovella, "Server selection using dynamic path characterization in wide-area networks," *Proceedings of IEEE INFOCOM '97*
- [2] K. Lai, M. Baker, "Nettimer: A tool for measuring bottleneck link bandwidth", *Proceedings of the USENIX Symposium on Internet Technologies and Systems*, March 2001.
- [3] K. Lai, M. Baker, "Measuring bandwidth", *Proceedings of IEEE INFOCOM '99*, New York, NY, USA, pp. 235-245, March 1999.
- [4] K. Lai, M. Baker, "Measuring link bandwidths using a deterministic model of packet delay", *Proceedings of ACM SIGCOMM 2000*
- [5] Constantinos Dovrolis, Parameswaran Ramanathan, and David Moore. What do packet dispersion techniques measure? In *Proceedings of IEEE INFOCOM*, April 2001.
- [6] C. Casetti, M. Gerla, S. S. Lee, S. Mascolo, and M. Sanadidi, "TCP with Faster Recovery," *Proceedings of IEEE Military Communications Conference MILCOM 2000*, Los Angeles, USA, October 2000.
- [7] S. Keshav, "Packet-pair flow control," 1994. <http://www.cs.cornell.edu/skeshav/papers.html>. To appear, *IEEE/ACM Transactions on Networking*.
- [8] Qian Zhang, Ya-Qin Zhang, and Wenwu Zhu, "Resource Allocation for Audio and Video Streaming over the Internet" - ISCAS 2000