# An Early Bandwidth Notification (EBN) Architecture for Dynamic Bandwidth Environment

Debojyoti Dutta
USC/ISI
4676 Admiralty Way
Marina Del Rey, CA 90292, USA
Email: ddutta@isi.edu

Yongguang Zhang
HRL Laboratories, LLC.
3011 Malibu Canyon Road
Malibu, CA 90265, USA
Email: ygz@hrl.com

***Abstract*—**

**In today's heterogeneous Internet, bandwidth available to TCP flows is often variable. However, current TCP cannot perform optimally under such dynamically varying bandwidth conditions. This paper addresses this problem by introducing a new architecture to improve TCP performance with explicit bandwidth notification (EBN). It uses a normalized bandwidth feedback method to provide accurate and timely bandwidth estimations. Then, a new TCP control algorithm (TCP-EBN) is proposed to promptly respond to any bandwidth changes. Our simulation results have shown that TCP-EBN performs much better than several other variations of TCP.**

## I. Introduction

TODAY'S Internet is inherently heterogeneous. As a result, bandwidth available to a TCP flow is increasingly variable. There are three main reasons why the available bandwidth is increasingly volatile: multiplexing, access control, and mobility. First, the bandwidth available to TCP flows will be affected by other flows sharing the same bottleneck link. In a QoS network where TCP flows are multiplexed in the *best effort* category, the available bandwidth fluctuates when higher priority flows and other TCP flows come and go. Second, in medium access links like wireless network, the bandwidth available to a TCP flow depends on the channel utilization and medium access protocol dynamics. Finally, in mobile networks where frequent mobility leads to hand-offs among paths with different characteristics, both bandwidth and delay can change significantly from an end-to-end [1] perspective.

Research has shown that the current congestion control mechanisms in TCP cannot effectively handle dynamic bandwidth between two TCP end-points, and TCP performance may suffer as a result [2], [3]. One explanation is that TCP does not directly take measure of the bandwidth currently available to the flow, and TCP does not use this knowledge in its control algorithms. We believe that, if accurate information about bandwidth changes is available, TCP can improve its performance under highly variable bandwidth environment. We have verified this hypothesis through a simulation study.

This work was done when Debojyoti was at HRL Labs, LLC during May-August 2001

In this paper, we introduce a new architecture for improving TCP performance in these variable bandwidth environment. This architecture is based on network providing feedback of current available bandwidth to TCP source. Through simulation, we show that our scheme results in better performance compared to TCP NewReno and other related work.

## II. Related Work

There has been few extensive studies on how to use bandwidth estimation in TCP control algorithms. TCP Westwood [4] estimates the available bandwidth by applying a discrete time low pass filter on the ACK stream, and uses this knowledge to do faster recovery on packet losses. Their scheme works well with coarse bandwidth changes where the variation follows a step function.

In an earlier work, TCP Vegas [5] uses fine-grain system timers to achieve more accurate RTT estimates. It then uses these estimates to calculate the expected throughput. If the expected throughput does not agree with the measured throughput TCP Vegas will adjust the congestion window accordingly. (through linearly increase or linear decrease).

Both TCP Vegas and TCP Westwood use tradition ways of estimating bandwidth based on measurements at the network end-points, usually through timing of ACK packet arrivals. The above approaches may have its limitation due to return path congestion and the TCP ACK compression effort.

Another approach is to augment the end-point measurement with feedback from the intermediate nodes. This may produce more accurate bandwidth estimation for the end-point to implement any performance improvement. The idea of sending feedback from intermediate routers is not new; DECbits and ECN [6] uses feedbacks from intermediate routers to pass congestion information. The question is what information should be passed, and how. and where to encode this information. One mechanism is to encode this information in the packet header, like the ECN bits [6]. Another mechanism is to use a ICMP packet or a lightweight signaling protocol or piggyback the bandwidth back to the source [7] which needs the explicit use of the true bandwidth.

## III. Early Bandwidth Notification (EBN)

The basic idea of this approach is as follows. The intermediate router would measure or estimate the current bandwidth available to a TCP flow (or aggregate of flows), and feedback such information to the TCP sender. The TCP sender will respond to this feedback information by adjusting its congestion window size. This concept is analogous to ECN [6], where routers feed network congestion state information back to TCP senders and they respond by invoking congestion control.

### A. Architecture

Consider the network given in Figure 1. The router at C has implemented mechanisms to estimate the bandwidth available to TCP flows (which will be further explained in the next section). For every packet passing the link, the bandwidth information is then encoded into certain reserved bits in the packet header. We call these bits *Early Bandwidth Notification (EBN)* bits. Currently we are using 8 bits to encode bandwidth information.

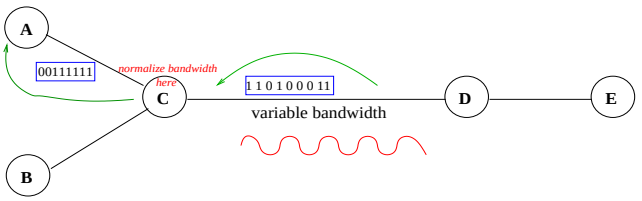



Fig. 1. Basic EBN Architecture

Suppose a packet travels from $A$ to $C$ and toward $D$. The router at $C$ will look at this packet and from the bits will try to ascertain the bandwidth available to the flow till C. Then it will try to estimate the bandwidth from $C$ to $D$. After that it will normalize this estimate with the encoding that is present in the packet header and send it to $D$. Fig. 1 illustrates the procedure.

Say, the TCP agent is getting acknowledgments from the recipient at $E$. Each such packet has a bandwidth encoding in it. The source looks at this and tries to estimate the available bandwidth. It then checks whether the bandwidth is going down or up. If the current bandwidth shows a downward trend, it will try to behave very much like TCP. It has been shown that multiplicative decrease leads to network stability [8]. On the other hand, if the bandwidth increases, then there is scope for improvement. A normal TCP agent would actually wait for a couple of ACKs before its congestion window is increased significantly. In our work, we try to make the TCP respond faster by increasing the congestion window size. Thus TCP can send more data. Since the available bandwidth is greater, the network can handle this increased window.

### B. Estimating Available Bandwidth

This can be done by several ways. The most common method is to have the router keep track of all the input and the output interfaces. Now the feedback can be flow based or it can be based on the aggregate. For flow based feedback, the router keeps track of all the flows and the bandwidth they use. Now, this does not add significant overhead since WRR and fair queuing is done anyway. So these measurements can be done at a per flow basis. This method has scalability issues at the core but it will work fine at the edges. This is where bandwidth estimation is important. We assume that most of the congestion is due to the edge and the core is over-provisioned. For the core routers, we have to estimate the bandwidth used by a flow. We can only approximate it. We use the *Fair Share Approximation*. In this, we simply find the aggregate bandwidth and divide it equally amongst all the flows. Sometimes this is bound to be erroneous especially with the over-provisioning. But we will see later that such errors can be taken care of due to our novel scheme of using normalized available bandwidth.

### C. Changing the TCP Algorithm

Now that TCP has accurate information about bandwidth changes, we can take advantage of this information to do better flow control that matches the changing bandwidth condition. One obvious approach is to increase the congestion window size (`cwnd`) when more bandwidth becomes available, and to reduce it when the bandwidth decreases. While there can be many ways to achieve this goal, we have developed the following empirical formula. We have proved through simulation (see subsequent sections) that this formula has achieved our objective.

Our mechanism is based on utilizing the EBN feedback described in the previous subsections. Let's denote the value of instant bandwidth encoded in the current EBN feedback to be *currebn*, and let's denote the previous instant bandwidth value to be *lastebn*. Our algorithm is as follows:

1) $currebn = lastebn$: we do nothing.
2) $currebn < lastebn$: we will reduce the current congestion window (`cwnd`), by multiplying it with $\beta_1 \times currebn/lastebn$. Here $\beta_1$ is a correction factor that we set to 1.
3) $currebn > lastebn$: we will increase the congestion window (`cwnd`), by multiplying it with $\beta_2 \times currebn/lastebn$. Here $\beta_2$ is a correction factor that we set to 2.

Here, $\beta_1$ and $\beta_2$ are smoothing factors that determine how much should `cwnd` be tuned to follow the bandwidth changes. Their values are empirically chosen through simulation study.

## IV. Experimental Results

To evaluate our EBN strategy, we have implemented it in NS2 [9] (version 2.1b7a) and conducted simulations to compare its performance with several other TCP variations, namely, TCP-NewReno, TCP-Vegas and TCP-Westwood. The modification to TCP algorithm is implemented as a derivative of TCP-NewReno to add bandwidth estimation. We call it *TCP-EBN*.

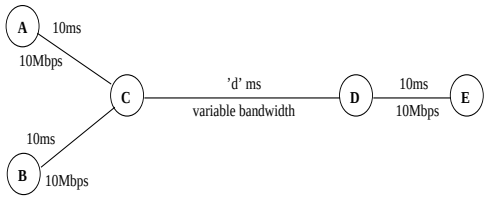



Fig. 2. The Simulation Network with Changing Bandwidth

Fig. 2 illustrates the simulation network topology. Here the links between $A$ and $C$, $D$ and $E$, and between $B$ and $C$ simulate the local access networks from $A$ or $B$ to the Internet. The bandwidth for both links is fixed at 10Mbps and the delay is fixed at 10ms. The link between $C$ and $D$ simulates a heterogeneous inter-network (the Internet) with dynamic changing bandwidth characteristics. We make this link a predominant and bottleneck part of the whole network so that its link characteristics can be manifested in the end-to-end performance between $A$ and $E$ or between $B$ and $E$. To implement this link in NS2, we have introduced a new type of link called variable-bandwidth-link – where the instantaneous bandwidth changes over time according to a pre-defined bandwidth function.

We have implemented a bandwidth function in which the bandwidth changes according to *sine* wave: $F_t(b, \alpha, \omega, \phi) = b \cdot (1 + \alpha^{-1} \cdot \sin(\omega \cdot t + \phi))$ where $b$, $\alpha$, $\omega$, and $\phi$ are tunable control parameters. Here $b$ defines the bandwidth average, $\alpha$ (the amplitude ratio no less than 1) controls how drastic is the change, $\omega$ (the angular frequency) controls how frequent is the change (note that the period of change is $2\pi/\omega$), and $\pi$ defines the phase of the oscillation (which is usually zero in our simulations). While a sine function may not be the most suitable approximation for studying the bandwidth dynamics in the Internet, its versatility allows us to see quickly how TCP responds to slow or fast changing bandwidth.

In addition to the bandwidth characteristics, the delay parameter ($d$) for the link between $C$ and $D$ is also configurable to facility simulation.

During the simulation experiments, there are TCP flows from $A$ to $E$ and from $B$ to $E$. The flow from $B$ to $E$ is a small 20Kbps flow to provide competing background traffic. This is to ensure that we avoid the synchronizations due to the inherently deterministic nature of packet level simulations.

The data sets are denoted using a simple naming schema. For example, the label *tcp-100-200-0.1-3* means that it is for *tcp* with the average bottleneck bandwidth ($b$) set to *100 Kbps*, the delay of the bottleneck bandwidth link set to *200 ms*, the angular frequency ($\omega$) to be *0.1*, and the amplitude ratio of the variance ($\alpha$) to be *3* (i.e., the bandwidth would vary within $\pm 1/3$ around the average). If the last parameter is not included, we assume a default value of 3.

### *A. Comparison of TCP-EBN with TCP-NewReno*

In this section, we compare the performance of TCP-EBN with TCP-NewReno. The results are plotted in Fig. 3. We first consider cases (a) to (d). In all these cases, TCP-EBN outperforms TCP-NewReno. It is interesting to note that EBN delivers better performance gains when the delays are smaller. This is because smaller delays implies smaller the buffer buildup, hence the drop probabilities are smaller. According to Padhye et al [10], throughput can be written as $T = k/(RTT \cdot \sqrt{3p/8})$. Then, a shorter delay will ensure a better throughput. Our scheme is aggressive and ensures that the sender shall try its best to send as much as the bandwidth can allow. The interesting conclusion is that buffer sizes at routers should be chosen appropriately to tackle variable bandwidth (the buffer size was 50 packets in our simulation).

Next we look at case (e). In this case, we simply varied $\omega$, i.e., the frequency of the bandwidth oscillation, while fixed other parameters. In this graph, we can see that EBN had much better performance than TCP-NewReno under all oscillation frequencies. Ideally, we had expected that the faster we vary the bandwidth, the better EBN should perform. But the simulation result shows that not to be the case. We believe that buffers at the router and the the standard max-receiver window and the maximum congestion window have contributed to this smoothing effect.

In case (f), we varied the delay and fixed the other parameters. EBN again delivered better performance. We believe the reason is that EBN helps to alleviate TCP slow start. This can be verified by comparing the curves of 1000ms delay.

### *B. Comparison of TCP-EBN with different TCP variations*

In this section, we compare the performance of our TCP-EBN with the different variations of TCP. We add TCP-Vegas because we believe that it is the first to use RTT measurement to estimate bandwidth. We also compare our approach with TCP-Westwood [2] because it is a significant piece of related work. The results of this comparison are in Fig. 4.

We first look at case (a) to (d). These four cases have the same network parameters except $\omega$. In all four cases, TCP-EBN outperforms the other variations and the performance gap increases as $\omega$ (oscillation frequency) increases. This matches our predictions. With a faster variation, explicit feedbacks from intermediate routers can give a much more accurate estimate of the bandwidth.

Another interesting observation is that TCP-Newreno outperforms TCP-Vegas and TCP-Westwood in all cases. This is contradicting to the previous published results that TCP-Westwood outperforms TCP [2]. This is because, the bandwidth variation in their study was in the form of sudden steps, while in our study, we use continuous variation. We believe that their bandwidth estimation algorithms are not being able to detect the continuous changes prevalent our scenarios. Finally, the reason why TCP-Vegas does very badly in all the cases is probably due to the lack of fast retransmits and fast recovery in TCP-Vegas.

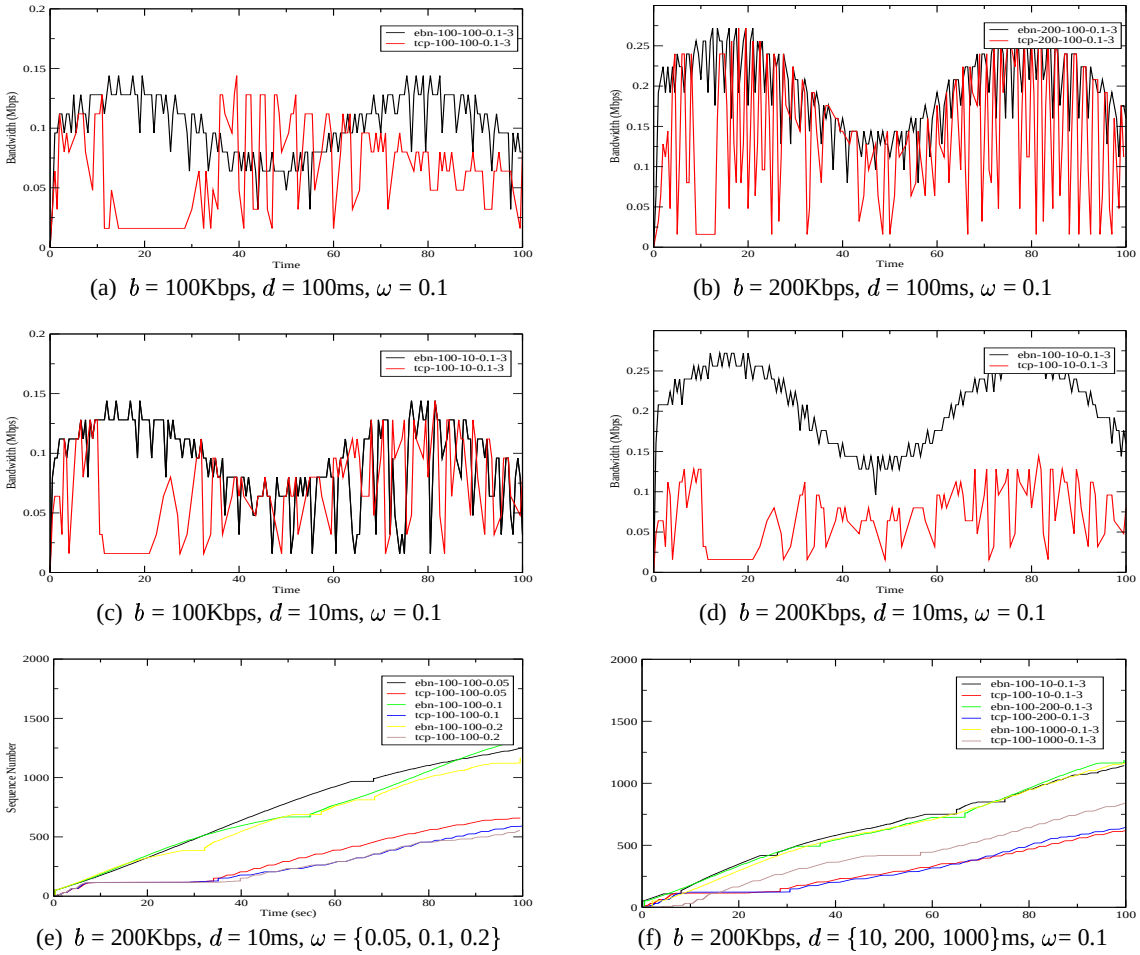



Fig. 3. Comparisons between TCP-EBN and TCP-NewReno ($\alpha$ = 3)

For case (e) and (f), we stepped up the average bandwidth ($b$) to 200Kbps. And for case (g) and (h), we further stepped up the delay (of the bottleneck link) to 200ms The simulation data clearly indicate that for these cases TCP-EBN still performs much better than the other TCP variations. We should also note that the performance gap is slightly bigger when the delay is lower.

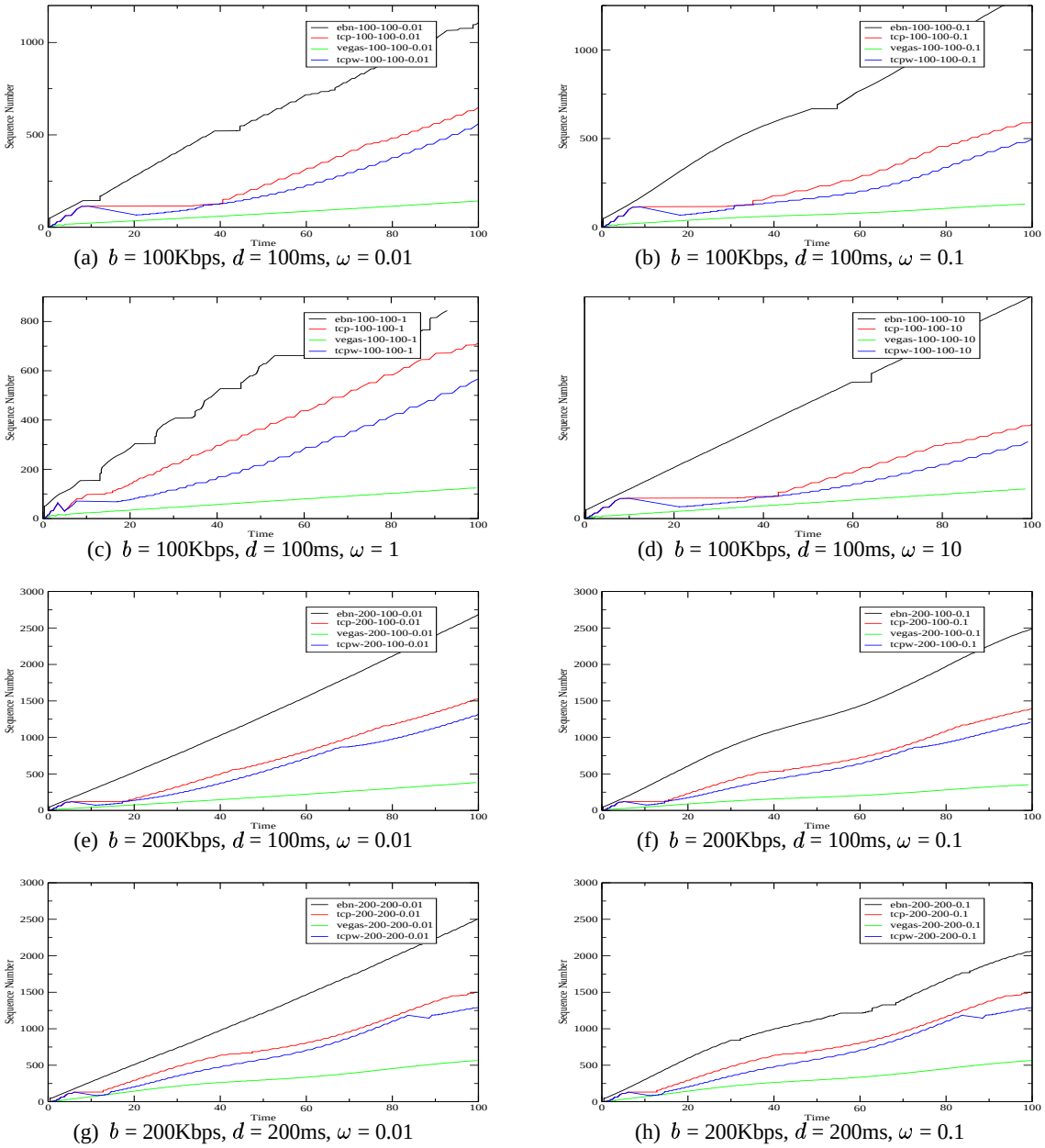
## V. Discussions, Future Work and Conclusions

The results show that TCP-EBN has better throughput than TCP-NewReno and TCP-Westwood. This has been possible due to the more accurate feedback that was available from the intermediate routers. We believe that the such feedback will become more popular to give the extra support to the already steady congestion control algorithms present in TCP. TCP-Westwood does a nice job of estimating bandwidth when the variations are not very quick.

This paper assumes that the routers can estimate the available bandwidth and send this as feedback. A complete discussion of this issue is beyond the scope of this paper. Most routers have some AQM or QoS policies that operate on a per-flow basis. We believe that such routers can be modified to incorporate our *EBN* scheme without any increase in complexity.

We are currently investigating means to make this architecture more robust and stable. Key issues are TCP-friendliness and the dynamics of TCP-EBN in failure conditions. We are also looking into effects of routing failures, transients on TCP-EBN.

In this paper, we have demonstrated that TCP NewReno is incapable of handling network conditions where the available bandwidth varies to a great extent. It was also shown that there were inherent limitations to the performance improvement that can be obtained with end-2-end feedback. This was primarily due to the phenomena like ACK compression. Then we introduced a novel approach to help improve TCP performance in variable bandwidth heterogeneous networks. This was based on the intermediate routers sending a few bits of information that encoded the available bandwidth at that router. Using this feedback, we have modified the TCP window size to give much better performance compared to TCP NewReno, TCP Vegas and




(a) $b$ = 100Kbps, $d$ = 100ms, $\omega$ = 0.01

(b) $b$ = 100Kbps, $d$ = 100ms, $\omega$ = 0.1

(c) $b$ = 100Kbps, $d$ = 100ms, $\omega$ = 1

(d) $b$ = 100Kbps, $d$ = 100ms, $\omega$ = 10

(e) $b$ = 200Kbps, $d$ = 100ms, $\omega$ = 0.01

(f) $b$ = 200Kbps, $d$ = 100ms, $\omega$ = 0.1

(g) $b$ = 200Kbps, $d$ = 200ms, $\omega$ = 0.01

(h) $b$ = 200Kbps, $d$ = 200ms, $\omega$ = 0.1

Fig. 4. Comparisons between TCP-EBN, TCP-NewReno, TCP-Vegas, and TCP-Westwood

TCP Westwood [2].

## REFERENCES


[1] J.H. Saltzer, D.P. Reed, and D.D. Clark, "End-To-End arguments in system design," in *the 2nd International Conference on Distributed Systems*, April 1981, pp. 509–512.

[2] S. Mascolo, C. Casetti, M. Gerla, M. Y. Sanadidi, and R. Yang, "TCP Westwood: Bandwidth estimation for enhanced transport over wireless links," in *ACM MOBICOM*, Aug. 2001, pp. 287–297.

[3] D. Dutta and Y. Zhang, "An active proxy based architecture for TCP in heterogeneous variable bandwidth networks," in *IEEE Globecom*, Nov. 2001.

[4] S. Mascolo, C. Casetti, M. Gerla, S. Lee, and M. Sanadini, "TCP Westwood: congestion control with faster recovery," 2000, UCLA CS Technical Report 200017, Univ. California at Los Angeles.

[5] L.S. Brakmo, S.W. O'Malley, and L.L. Peterson, "TCP Vegas: New techniques for congestion detection and avoidance," in *ACM SIGCOMM'94*, May 1994, pp. 24–35.

[6] S. Floyd, "TCP and explicit congestion notification," *ACM Computer Communication Review*, vol. 24, no. 5, pp. 10–23, Oct. 1994.

[7] M. Gerla, W. Weng, and R. Signo, "Bandwidth feedback control and real time sources in the internet," in *IEEE Globecom*, Nov. 2000.

[8] V. Jacobson, "Congestion avoidance and control," in *ACM SIGCOMM*, Aug. 1988, pp. 314–329.

[9] L. Breslau et.al., "Advances in network simulation," *IEEE Computer*, vol. 33, no. 5, pp. 59–67, May 2000.

[10] S. Floyd, M. Handley, J. Padhye, and J. Widmer, "Equation-based congestion control for unicast applications," in *ACM SIGCOMM*, Aug. 2000.