Communication Networks

Generalized Window Advertising for TCP Congestion Control[*]

MARIO GERLA
Computer Science Department – Boelter Hall – UCLA, 405 Hilgard Ave., Los Angeles, CA, 90024, USA
gerla@cs.ucla.edu

RENATO LO CIGNO
Dipartimento di Elettronica – Politecnico di Torino, Corso Duca Degli Abruzzi, 24 – 10129 Torino, Italy
locigno@polito.it

SAVERIO MASCOLO
Dipartimento di Elettrotecnica ed Elettronica – Politecnico di Via Orabona, 4 – 70125 Bari, Italy
mascolo@poliba.it

WENJIE WENG
Motorola, Inc., San Diego, California, USA
W.Weng@motorola.com

Abstract. Congestion in the Internet is a major cause of network performance degradation. The Generalized Window Advertising (GWA) scheme proposed in this paper is a new approach for enhancing the congestion control properties of TCP. GWA requires only minor modifications to the existing protocol stack and is completely backward compatible, allowing GWA-hosts to interact with non-GWA hosts without modifications.

GWA exploits the notion of end-host–network cooperation, with the congestion level notified from the network to end hosts. It is based on solid control theory results that guarantee performance and stable network operation. GWA is able to avoid window oscillations and the related fluctuations in offered load and network performance. This makes it more robust to sustained network overload due to a large number of connections competing for the same bottleneck, a situation where traditional TCP implementations fail to provide satisfactory performance.

GWA-TCP is compared with traditional TCP, TCP with RED and also ECN using the *ns-2* simulator. Results show that in most cases GWA-TCP outperforms the traditional schemes. In particular, when compared with ECN, it provides smoother network operation and increased fairness.

1 INTRODUCTION AND RELATED WORK

The boom of the Internet is stressing the entire TCP/IP protocol suite implementation, as the users grow in number, along with the demand for enhanced service and performance. The majority of data services in the Internet are based on TCP (Transport Control Protocol), with application ranging from bulk data transmission (like `FTP` file transfers), to interactive, delay sensitive services (like `Telnet` remote terminal), to web browsing. TCP provides a connection oriented, reliable communication channel to upper layers. It also provides the functions of receiver flow control and network congestion control. The TCP congestion control mechanism in standard implementations is based on a simple, wired network model, with very reliable nodes and links. Packet loss is mostly caused by buffer overflow and is thus taken as indication of congestion. Several studies have shown that this control approach may fail if losses in the network are due to cases other than congestion, such as in the case of wireless links (see for instance [1, 2, 3]). Moreover, the control does not work well if the congestion is caused by an excessive number of TCP connections allowed to compete for the same bottleneck [4].

Fortunately, routers of the next generation will be more intelligent than the ones used in traditional TCP implementations. They will be able to predict impending congestion because of better environment awareness; however, a major problem consists in correctly elaborating and conveying congestion information from routers to TCP sources.

[*]This work was done when R. Lo Cigno was visiting the UCLA CS Dept. with CNR (Italian National Research Council) grant 203.15.8. The work was partially supported by NSF, DARPA and NASA through varius grants and by the Italian Ministry for University and Scientific Research (MURST) through the project MQOS.

A related issue is to adapt TCP to accept new feedback from the routers, ensuring more stable performance and yet retaining complete backward compatibility. The first approach in the direction of router and TCP cooperation is the Explicit Congestion Notification (ECN) scheme proposed in [5, 6]. ECN is based on a binary feedback from routers to sources that triggers a window size reduction identical to that caused by the fast retransmit/fast recovery mechanism of TCP-Reno, yet without requiring the drop of a packet. ECN is effective in reducing packet loss. However, because of its binary feedback, it cannot avoid window and network oscillations. These can negatively affect the network performance. A recent contribution, named Early Random Marking [7], overcomes partially ECN limitations, at the cost of increasing the TCP transmitter complexity. Binary feedback schemes were also studied in the ATM-ABR (Asynchronous Transfer Mode – Available Bit Rate) context (see for instance [8, 9]), but explicit rate (ER) schemes were finally preferred due to the difficulty of obtaining stable operation with small rate and buffer oscillations without complex parameter optimization that are dependent on the network scenario (propagation delay, number of nodes, etc.). ABR-ER rate schemes, such as those proposed in [10, 11], are related to our work in that they use explicit feedback from the network to the sources; however the networking scenario is so different that comparisons are very hard. An example for all: in ABR the control algorithm is implemented within network nodes, while in TCP/IP the network only performs the measures, while the controlling algorithm is implemented in end nodes. We only notice that the ERICA scheme [10] does not have any control-theoretical framework supporting the stability and performance properties of the system, while [11], which is based on control theory, makes different assumptions and proposes a different control technique.

Two different approaches, also based on explicit congestion notification, are presented in [12, 13]. The second one is based on global network cost optimization; it operates through a distributed algorithm that converges to the optimal operating point if a well-formed cost function is given. This procedure completely disregards transient behavior; this fact arises some concern about its suitability to work in a highly dynamic network scenario like the Internet. The approach in [12] is the more related to the one proposed in this paper. Both approaches propose the use of the advertised receiver window (*rcvwnd*) field in TCP header to *also* convey network congestion indication. Similarities, however, stop here: the congestion control algorithms are completely different, as well as the protocol and implementation details.

TCP Vegas [14], which is often considered "rate based," is indeed not comparable with the scheme we are proposing. The main reason is that TCP Vegas tries to estimate the available bandwidth in the network end-to-end, without requiring the network cooperation. It has been show [15] that the interaction of TCP Vegas with other traffic or the presence of losses not due to congestion may lead to the starvation of TCP Vegas. The authors in [16] conjecture that TCP-Vegas may benefit from the use of AQM schemes and present simulations showing the point. The conjecture is based on a differential equation model.

Charge-sensitive TCP, with milestone works as [17, 18], is also often cited as a way to improve TCP performance. These schemes aims at finding a stable operation point for the network, disregarding the transient behavior (similarly to the work presented in [13]), during which the schemes cannot guarantee a low loss rate; however, Internet traffic is made of short connections, hence a scheme that controls the sources from the very beginning of the transmission is needed.

In [19] we presented a specific implementation of the scheme discussed here that is suited for ad-hoc wireless networks and is not completely backward compatible with previous TCP implementations. In [20, 21] we discussed how explicit rate feedback in the Internet can help in introducing real time services without affecting TCP traffic, as well as forcing fairness among etherogeneous competing TCP flows. None of these papers, however, reports the theoretical analysis which is the focus of this paper. The foundations of this analysis were presented in a simplified scenario and without implementations or results in [22].

This paper presents a comprehensive treatise of the TCP window control mechanism named Generalized Window Advertising (GWA). GWA-TCP is backward compatible, requiring a minimum upgrade to the TCP/IP protocol suite. GWA-TCP is compared with traditional TCP, TCP with RED and also ECN with TCP Reno using *ns-2* [23]. Results show that in most cases GWA-TCP outperforms the traditional schemes. In particular, when compared with ECN, it provides smoother network operation and increased fairness.

2 Generalized Window Advertising TCP

TCP [25, 26, 27] performs two fundamental control tasks: i) end-to-end flow control to avoid overflowing the receiver buffer; ii) connection congestion control, adapting the amount of information injected in the network to the network congestion status. This latter function is based on *congestion recovery*, rather than prevention. TCP increases the traffic offered to the network until the network is forced to drop some packets; then it backs off to give the network time to relieve from congestion, and starts again to increase the offered load to force another congestion episode. The detailed description of the TCP protocol and its dynamic in various versions (Tahoe, Reno etc) is beyond the scope of this paper and can be found in the literature. Below, we will review the features common to all TCP versions. These are the only features needed for the GWA implementation.

TCP flow control is enforced by means of the *receiver*

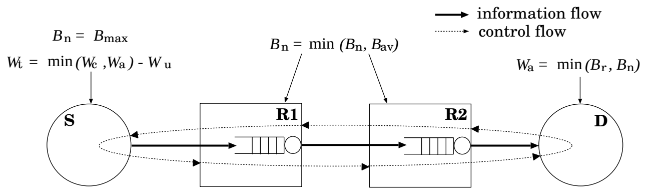

Figure 1: GWA algorithm deployment in a simple network.

advertised window (rcvwnd) field W_a set by the receiver in the header of acknowledgment (ACK) segments. Congestion control is enforced by means of the *congestion window (cnwd)* W_c that is computed by the transmitter following the TCP congestion control algorithm (see [25] for details). Finally the amount of data W_t which TCP is allowed to transmit at any given time is computed as

$$W_t = \min\{W_c, W_a\} - W_u \tag{1}$$

W_u is the amount of outstanding data, i.e., the part of the sliding transmission window that is filled with data already sent, but not yet acknowledged.

The key idea of GWA is to "generalize" the interpretation and function of W_a, using it to convey *both* the receiver available buffer space B_r and the network congestion status B_n. Since network congestion status can be expressed as buffer space available in routers, B_n is defined as the minimum buffer space available in routers along the TCP path.

IP routers cannot access or modify the field W_a since this would be a layer violation. Moreover W_a is protected by TCP CRC. Changing W_a requires the re-computation of the CRC, which is not feasible in a high speed router. The information must be delivered to the TCP transmitter or receiver at the IP level, leaving the task of inter-layer communication to the end hosts.

The information can be sent either to the transmitter (backward notification) or to the receiver (forward notification), with minor implications on performance during transients. Conveying the information to the receiver yields a simpler solution. In fact, the receiver simply assigns

$$W_a = \min\{B_r, B_n\} \tag{2}$$

If IPv6 is used, B_n can be a 2-bytes optional field within the packet header. For the sake of simplicity the following description assumes that this optional field is used by all network hosts. Figure 1 shows how the algorithm is deployed in the network. It features a TCP transmitter (or source) **S**, two routers **R1** and **R2** and a receiver (or destination) **D**.

The transmitter, before transmitting the packet, sets $B_n = B_{\max}$, where $B_{\max}$ is a transmitter chosen target value; for instance it can be the maximum window size negotiated at connection setup. If the window scale option is set by the TCP connections, the same scaling factor applies to B_n.

Routers along the connection modify B_n according to the buffer space B_{av} available for that connection, without ever increasing its value. When the packet arrives at the destination it contains the minimum buffer space along the followed path. The scheme works even if not all the packets follow the same route; however the analysis carried out in section 3 no longer holds. The receiver copies B_n from the IP to the TCP level, and the TCP receiver implements Eq. (2).

B_n is the buffer space that the network can provide to the connection. An implementation with per-flow queuing and Round Robin (RR) serving discipline is an ideal match for GWA scheme. However also FCFS on the aggregate queue can be used. With aggregate FCFS queuing it is necessary to estimate the number of active flows, in order to account for the buffer space available to each one. This can be a source of unfairness, but generally it does not impair the overall performance, as shown in section 6.

Notice that the network feedback is available to the sources from the very beginning of the transmission (i.e., the transmission of the SYN/SYN_ACK pair). The system convergence is monotonic, as demonstrated in [22], so that the transient behavior of connections is controlled too, avoiding overshoots of the transmission rate and the consequent packet losses.

3 THE ALGORITHM FROM A CONTROL THEORY PERSPECTIVE

The theoretical analysis is restricted to the case with per-flow queuing and RR serving discipline. A fluid flow approximation is assumed together with static routing so that all packets follow the same route from the source to the destination. The algorithm is designed exploiting the *Smith predictor* [28], which is a control technique for time-delay systems. The control algorithm is first derived assuming a rate based control scheme. Then, it is modified to fit the TCP window model.

Let $r_i(t)$ be the transmission rate of i-th connection at the source and $c_{i,j}(t)$ be the service rate given to the i-th TCP connection by router j along the connection path. $c_{i,j}(t)$ is generally an unknown, bounded function. It takes into account possible higher priority traffic, as well as the other TCP flows, that interact with the considered one through the Round Robin scheduler.

Assuming infinite buffer capacity[1], at any given router along the path we have the queue level:

$$q^l_{i,j}(t) = \int_0^t \left[r_i(\tau - d'_{i,j}) - c_{i,j}(\tau) \right] d\tau \tag{3}$$

where $d'_{i,j}$ is the delay from source i to router j, $q^l_{i,j}(\cdot)$, $r_i(\cdot)$ and $c_{i,j}(\cdot)$ are non negative; the superscript l refers to the output logical link.

[1]As we show later, the control algorithm guarantees that no packets are lost, hence Eq. (3) is satisfied for any *finite* buffer capacity.

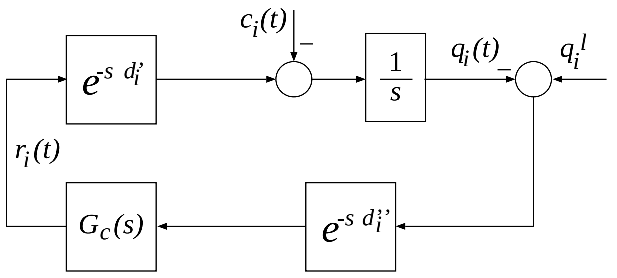

Figure 2: Block diagram of the closed loop model.

Each router j stamps the available buffer $(q_i^l - q_{i,j}^l(t))$ in the IP header field B_n if it is smaller than the one already stored in B_n. The minimum value reaches the receiver, which implements Eq. (2), and sends the value W_a back to the source. q_i^l is a target queue level that can be chosen according to different buffer management disciplines (see Dec. 5 for further details).

$W_a = (q_i^l - q_{i,j}^l(t))$ reaches the source $d_{i,j}''$ seconds after the router measured the available buffer; this terms accounts also for possible elaboration delays at the receiver. Notice that the connection round trip time is $d_i = d_{i,j}' + d_{i,j}''$, regardless of where the bottleneck j is located along the path. Since only the minimum buffer space is propagated at any time, the index j will be omitted from now on without loss of generality.

Figure 2 depicts the block diagram of the system, where $\frac{1}{s}$ is the Laplace transform of the integrator represented by the router buffer and $G_c(s)$ is the transfer function of the controller. The blocks $e^{-sd_i'}$ and $e^{-sd_i''}$ model the delay from the source to the congested router and from the congested router to the destination and back to the source.

The goal is to find a controller $G_c(s)$, which stabilizes the network operation, granting no losses and full link utilization with persistent sources. Mathematically, this can be expressed as (see [22] for a more detailed explanation):

$$q_i^l(t) \le q_i^l \quad \forall t \ge 0 \tag{4}$$

$$q_i^l(t) \ge 0 \quad \forall t \ge T_t \ge d_i \tag{5}$$

Eq. (4) states that the buffer never drops packets provided that $\sum_i q_i^l \le B$ where B is the buffer dimension; Eq. (5) guarantees the full utilization of the bandwidth assigned to flow i, after a transient time T_t, which is necessarily larger than the connection round trip time d_i. Indeed Eq. (5) guarantees the full utilization of the link if any work conserving serving discipline, like RR or FCFS, is used.

In [22] it is shown that a simple controller that satisfies Eq. (4) and (5) with $T_t = d_i$ is the following:

$$G_c(s) = \frac{k}{1 + \frac{k}{s}\left(1 - e^{-sd_i}\right)} \tag{6}$$

Eq. (6) defines the controller in the Laplace domain. In the time domain the controller equation is:

$$r_i(t) = k\left[q_i^l - q_i^l(t - d_i'') - \int_{t-d_i}^{t} r_i(\tau)d\tau\right] \tag{7}$$

Eq. (7) states that the computed rate at the source is proportional to the available buffer space $[q_i^l - q_i^l(t)]$, minus the amount of information that has been sent during the last round trip time, that is $\int_{t-d_i}^{t} r_i(\tau)d\tau$. The time constant of the system is $\tau_c = 1/k$; the transient dynamics are practically exhausted after $4\tau_c$. Setting $k = \frac{1}{d_i}$ provides monotonic convergence in 4 round trip times. Notice that the convergence to steady state is monotonic (without overshoots) since the closed loop dynamic is first order (see [22] for the explanation). This property is important to prove loss free operation, since the absence of overshoots means that the buffer cannot overflow.

Control Eq. (7) can be transformed from rate to window formulation by multiplying both sides of Eq. (7) by d_i. Notice that using $k \neq \frac{1}{d_i}$ would make the conversion to a window form of the control less straightforward.

Recall that $B_n \triangleq q_i^l - q_i^l(t)$ and $W_u = \int_{t-d_i}^{t} r_i(\tau)d\tau$. B_n is directly measured by the routers and forwarded to the TCP destination. W_u, the number of outstanding packets (bytes) is known by the TCP source. Thus, the implementation of the GWA scheme through Eqs. (1) and (2) is straightforward.

4 Properties of GWA and modeling approximations

GWA-TCP as outlined in section 3 has several appealing properties. GWA *always* ensures zero loss probability, regardless of the statistical fluctuations of the disturbances $c_i(t)$. This means that there are no losses even if somewhere along the path the bandwidth goes abruptly to zero. This property holds for any buffer capacity (see [22]). The buffer size only determines the link utilization, that is, a buffer capacity equal to the bandwidth-delay product is necessary to ensure full link utilization; in this case the link utilization reaches 100% after one round trip time. For any router in the network the minimum buffer size $S_B^{\min}$ needed to ensure 100% utilization is expressed as:

$$S_B^{\min} \ge \sum_{i=1}^{N} c_i^{\max} \cdot d_i^{\max} < c_l \cdot d^+ \tag{8}$$

where N is the number of connection; $c_i^{\max}$ and $d_i^{\max}$ are upper bounds on disturbances and delays[2]; c_l is the link

[2]Both disturbances and delays are limited in any real network, so that this condition is indeed not a limiting factor.

capacity and d^+ is an upper bound on $d_i^{\max}$ $\forall i$, i.e., the maximum of the round trip delays over all connections. Note that the first inequality of Eq. 8 states the well known condition for full link utilization of traditional TCP implementations. Thus, GWA-TCP does not require additional network resources to work properly. The second inequality gives an upper bound that can be used for router buffer dimensioning. This bound depends only on the local link speed and not on the speed of other links in the network. Hence, the buffers in routers supporting GWA-TCP can be dimensioned with reference to local parameters only.

The above properties can be rigorously verified in our model. However, the real system does not correspond exactly to the model. There are features in the real system which require a reinterpretation of these properties, namely: i) the discrete (instead of continuous) behavior; ii) the effect of transient TCP dynamics[3].

TCP behavior is intrinsically discrete, due to the packetized transmission of information. The difference between the fluid flow model and the real system can be expressed as sampling and quantization noise. Because of this, it can be expected that the real system needs a buffer larger than what predicted by theory in order to reach 100% link utilization. Besides, at steady state, the behavior can be affected by small ripples due to quantization: both the TCP transmission window and the network buffer occupancy may present small oscillations around the equilibrium point (while the theory predicts monotonic convergence).

Regarding slow start and congestion avoidance, both of these features have the effect of reducing the transmission window size (with respect to what is advertised by the network) during the initial phase of a connection. Since GWA guarantees zero loss, slow start and congestion avoidance will not be triggered during steady state transmission (except for infrequent losses due to link errors[4]) and thus will not play a major role. One effect of this "reduced offered load" is that full utilization is not reached after one round trip time; however this is a negligible performance loss.

5 BUFFER MANAGEMENT POLICIES

The control algorithm outlined in the previous sections leaves one degree of freedom in the choice of the target queue level q_i^l. In this section we analyze three different buffer management policies.

[3]Slow Start, Congestion Avoidance, Delayed ACKs and all the other protocol characteristics are in general non-linear, and make it difficult to mathematically analyze the closed loop dynamics. Notice that GWA properties ensure that, apart form the initial transient, Slow Start and Congestion Avoidance phases should be triggered very rarely.

[4]We assume a wired network model, with negligible loss due to link errors. If the path includes wireless segments with interference and error loss which are not locally protected, GWA-TCP proper operation requires that Slow Start and Congestion Avoidance mechanisms be turned off.

Routers with per-flow RR queuing discipline. We assume that GWA IP routers implement per-flow queuing. Let B_t be the total buffer size and n_f be the number of active flows. Let i identify the flow and $q_i(t)$ the queue length of flow i at time t. Then the buffer allocation scheme at the i-th router simply consists in setting $q_i^l = \frac{B_t}{n_f(t)}$ and assigning to the field $B_{n,i}$ of each packet the value

$$B_{n,i} = \min\left\{B_{n,i-1}, \left\lceil \frac{B_t}{n_f(t)} - q_i^l(t) \right\rceil\right\}$$

where $B_{n,i-1}$ is the value assigned by the router one hop before along the path and $n_f(t)$ is the value of n_f at time t and is measured by the router. A simple measurement (which ensures reasonable performance if the number of flows does not vary too quickly in time) is a geometrically decaying time average:

$$n_f(t) = \alpha n_f(t-\tau) + (1-\alpha) n_f^o(t) \tag{9}$$

where τ is an appropriate measure interval, $\alpha < 1$ is the decaying factor and $n_f^o(t)$ is the number of active flows observed during the interval $[t-\tau, t]$. Flow identification can be obtained either through the optional flow label, or by assuming a different flow for each association of source address and destination address. Notice that how flows are identified does not affect the performance of the control. The optimal choices of α and τ depend on link speed, buffer size, and traffic characteristics. A conservative choice, i.e., both α and τ quite large, say $\alpha = 0.9$ and $\tau = 0.1$ s, generally leads to an overestimate the number of active flows (as shown by the results in section 6), hence reducing the advertised buffer size. When the buffer is large enough, this conservative estimate has minor effect on performance.

Routers with FCFS queuing discipline. In FCFS queuing, the router does not keep track of the individual queues $q_i^l(t)$. Taking this into account and using the same notations as above, the buffer allocation policy assigns the available buffer to each flow by using the following estimate

$$B_{n,i} = \min\left\{B_{n,i-1}, \left\lceil \frac{B_t - q_t^l(t)}{n_f(t)} \right\rceil\right\}$$

where $q_t(t)$ is the overall queue length at time t; $n_f(t)$ is estimated through Eq. (9).

Though difficult to prove theoretically, this simpler scheme still provides good aggregate performance. However, there is no way to enforce fairness among flows with different round trip times. It can be expected that the throughput of individual connections is inversely proportional to their round trip time.

Bandwidth Aware *Routers.* Both queueing disciplines described above have the disadvantage of requiring large buffers to ensure full link utilization. One possibility to overcame this impairment is the use of available bandwidth measurements, as shown in [19, 20]. Indeed, letting v_i be the available bandwidth for connection i, we set the q_i^l equal to the bandwidth delay product $v_i \cdot d_i$; we can view $v_i \cdot d_i$ as a "virtual buffer" space assigned to connection i. Thus the resources allocation strategy becomes:

$$B_{n,i} = \min \left\{ B_{n,i-1}, \left\lceil v_i \cdot d_i - \frac{q_t(t)}{n_f(t)} \right\rceil \right\} \qquad (10)$$

We name this version of GWA *Bandwidth Aware* TCP (BA-TCP). BA-TCP has the property of driving the buffer occupancy to zero at steady state.

The implementation of BA-TCP following Eq. (10) is not straightforward, since each router need to know every connection delay d_i, a condition quite hard to realize. A simpler realization can be obtained implementing the controller within TCP transmitters and delivering v_i to sources (see [19, 20] for details). If the round trip propagation delay δ_i is used instead of d_i ($\delta_i \leq d_i$ for physical reasons), then $\delta_i \cdot \min_j(v_{i,j})$ is a lower bound to the transmission window. This last bounding enables to disregard the buffer content, since at steady state the buffer is empty due to the transmission window bounding: the role of the buffer is now only to absorb transients due to changes in the available buffer or in the number of connections. Results labeled "BA-TCP" in section 6 are obtained with this latter and very simple implementation.

6 COMPARISON WITH TRADITIONAL TCP IMPLEMENTATIONS

The performance predicted by theoretical analysis is very promising; however both modeling simplifying assumptions and implementation "shortcuts" are potential sources of performance degradation. Of particular concern is the interaction between the GWA scheme and the other TCP congestion control algorithms: timeouts, slow start, congestion avoidance, etc., that must be preserved for backward compatibility and robustness.

This section compares the performance of 6 different TCP implementations and network congestion control schemes, running on the simple network depicted in figure 3. Both GWA-TCP and BA-TCP were implemented in the *ns-2* simulator[5], developed at UC Berkeley within the VINT project[6].

The following extensions/modifications to the basic *ns-2* infrastructure were made.

- TCP dynamic window advertisement;

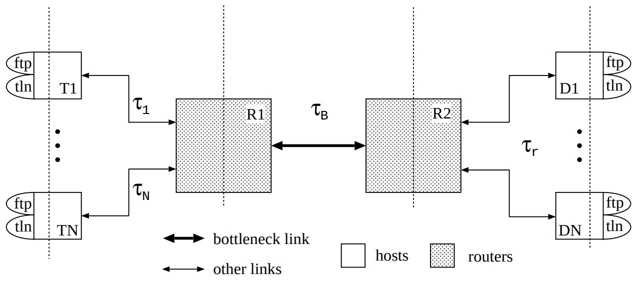

Figure 3: Network topology used for performance comparison.

- Introduction of the B_n optional field in the IPv6 header, plus the fields for BA-TCP as described in [19];
- Implementation of Eq. (2) in GWA-TCP receivers, and the additional control algorithm at the source in BA-TCP;
- Implementation of IP routers which advertise B_n or the available bandwidth v_i.

The 6 TCP implementations are:

- TCP-Reno and standard FCFS queuing IP routers, in the sequel named *Drop-Tail*;
- TCP-Reno and RED IP routers [29], named *RED*;
- TCP-Reno with ECN (Explicit Congestion Notification) capabilities [5, 6] and tagging RED IP routers, named *ECN*;
- TCP-Reno with GWA capabilities and per-flow queuing, RR IP routers, named *GWA-RR*;
- TCP-Reno with GWA capabilities and FCFS queuing IP routers, named *GWA-FCFS*;
- TCP-Reno with *Bandwidth Aware* GWA capabilities; FCFS IP routers capable of measuring the available bandwidth, named *BA-FCFS*.

Simulations are run both in LAN and WAN scenarios, with Telnet-like and greedy FTP applications competing for the bottleneck resources. The performance indices considered here are the FTP throughput at the receiver after discarding the duplicated packets (i.e., the "goodput"); the packet loss probability and the transfer delay of Telnet packets. Telnet throughput is negligible, thus it is not reported.

With reference to figure 3, simulation experiments are run with N hosts, each one running both an FTP and a Telnet application. All links are bidirectional, but the connections are all unidirectional, so that on the backward path the load is due only to ACKs and is very light. Referring again to figure 3, the propagation delay between source and destination i is $\tau = \tau_i + \tau_B + \tau_r$, and the round trip time

[5]*ns-2* is the last version of the simulator; all necessary information and the software can be retrieved at the URL `http://www-mash.cs.berkeley.edu/ns/ns.html`; the additional software for GWA can be obtained from the authors

[6]See the URL `http://netweb.usc.edu/vint` for additional information on VINT

experienced by the transmitter is $d_i = 2\tau + \tau_q$, where τ_q accounts for possible queuing delays both on the forward and the backward path. The TCP MSS (Maximum Segment Size) is set to 1000 bytes. The timer granularity (i.e., the TCP 'tick') is 0.1 s, hence it is somewhat "faster" than present day implementations that use values comprised between 0.2 and 0.5 s.

Telnet applications generate MSS packets with average interarrival time equal to 0.1 s. This is an unrealistic assumption for a Telnet connection, but can be representative of other time-sensitive applications. Indeed running simulations with Telnet connections offering a much lower load would result in a set of results whose statistical properties are not reliable.

The tuning of RED router parameters proved to be a non-trivial task. Parameter values must be adjusted to each specific network configuration in order to get satisfactory performance. In our experiments, the lower and upper thresholds are set to 1/12-th and 3/12-th the buffer size, respectively. The drop probability of RED increases linearly (from 0 to 0.1) with average queue length between the lower and the upper thresholds. Above the upper threshold, RED drops an incoming packet with probability =1. For ECN, routers mark (instead of drop) the packets when average queue length is between the two thresholds. Packets arriving, when average queue length is above the upper threshold, are marked rather than dropped[7]. The other parameters are the same as in the RED experiments.

6.1 LAN ENVIRONMENT

LANs are characterized by negligible propagation delay (i.e., same order or less than the packet transmission time). With reference to figure 3, LAN experiments are run with $N = 10$ hosts, $\tau_B = 40\,\mu$s, $\tau_r = 30\,\mu$s and $\tau_i = 30\,\mu\text{s}\ \forall i$. All link speeds are 100 Mbit/s. The receiver's available buffer space is 64 kbytes. Simulations are run for 25 s. Averaged results are computed after discarding the first 5 s of network operation, in order to avoid biasing due to network transients.

Figure 4 reports the goodput normalized to the link transmission speed, so that the fair share of each connection is 0.1. Each plot has three curves: the middle one is the average among all connections, while the upper and the lower ones are the maximum and the minimum, respectively. The spreading between these two latter curves is indicative of the "unfairness" of the network. The throughput is generally quite good with any scheme. Regarding fairness, all traditional implementations show some degree of unfairness. Any GWA algorithm ensures perfect fairness and full utilization even with small router buffer size.

Figure 5 reports the packet drop rate (plot a) together with the average (plot b) and maximum (plot c) delay experienced by Telnet packets. Each curve corresponds to a different implementation. At steady state, GWA-TCP and ECN yield zero losses. In fact, these three curves are overlapped in figure 5(a). Drop-Tail and RED show losses that decrease with increasing buffer size as expected. The behavior of Telnet packet delays can be explained by considering that the delay measured by the receiver includes retransmissions. This is the reason why TCP implementations that do not avoid loss show very high values of the maximum Telnet transfer delay. With GWA-RR, the delay experienced by Telnet packets is merely the propagation delay, since the individual queues associated with Telnet connections will always be empty. With GWA-FCFS, both the average and the maximum delay increase with buffer size, since the aggregate queue stabilizes to a higher value when the buffer is large. BA-FCFS yields negligible delays since it drives the overall queue length to zero. ECN Telnet delays are also very low. Moreover, from figure 5(c) we note that GWA and ECN maximum delays are in the millisecond range, while Drop Tail and RED maximum delays are in the range of seconds.

Figure 6 plots the transient behavior of an FTP connection window (left plots) and the bottleneck buffer (right plots) during the first 3 seconds of the experiment, for router buffer size = 120 kbytes. The window size is measured and plotted every time a packet is sent and the queue length is averaged over 0.01 second intervals. Small oscillations in GWA are due to quantization effects (residual buffer space is not a multiple of TCP packet size). The inserts in the left plots show a magnification of the first 0.1 seconds, highlighting the initial transient behavior of the network, and offering an insight that is not available from steady state results. In particular, these results clearly reveal the interaction of TCP with the network congestion reaction policy.

Drop-Tail and RED exhibit oscillatory behavior as expected. Somewhat surprising, is the fact that RED yields a very large number of timeouts. We note 8 timeouts in the first 3 seconds of transmission for the FTP under study (only a timeout can halt transmissions for more than 0.2 s). ECN does not suffer packet drops at steady state; however, it incurs multiple drops in a single window followed by timeout at the beginning of the session. This has no impact on steady state FTP throughput, but it may affect the performance of applications exchanging small files.

GWA schemes exhibit very stable behavior, without packet drops even during the initial transient. Steady state is achieved within 0.5 s. This value is much larger than the value indicated by fluid flow analysis. The reason lies essentially in the difficulty of correctly estimating the exact number of active flows during the early transmission phase. Simulation experiments performed assigning the

[7] In the original *ns-2* code we downloaded, the ECN routers mark packets when the buffer is between the thresholds, but drop them when it is above the higher threshold. We were not able to obtain satisfactory results with this layout, while we obtained good results by marking instead of dropping. Good results can probably be obtained also with further adjustments of RED thresholds and drop probability for ECN (which we did not attempt in our study).

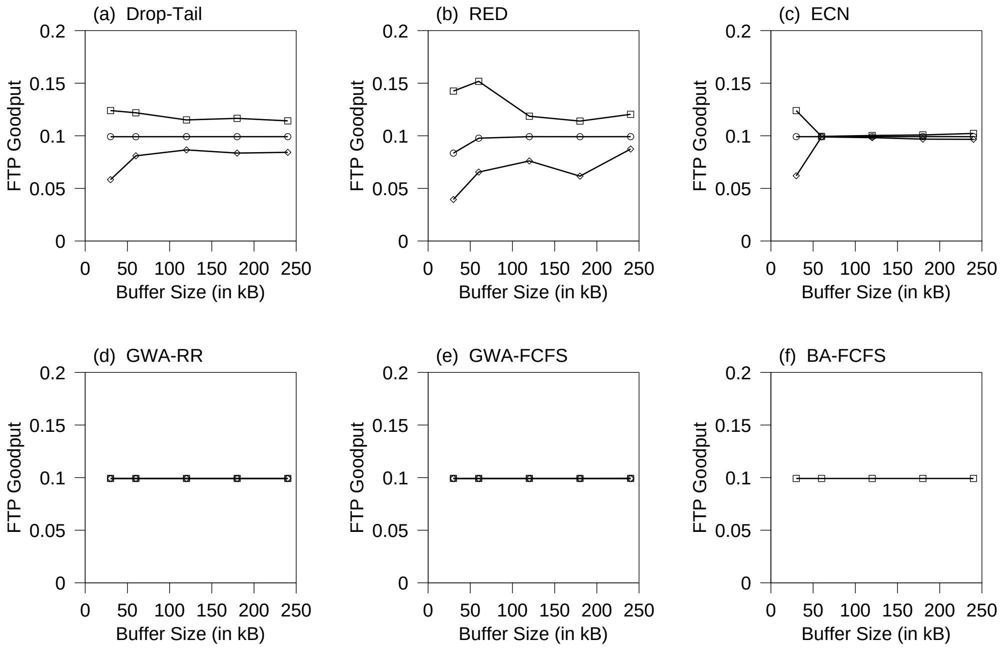

Figure 4: Receiver throughput for the 6 considered implementations versus the router buffer size in LAN environment.

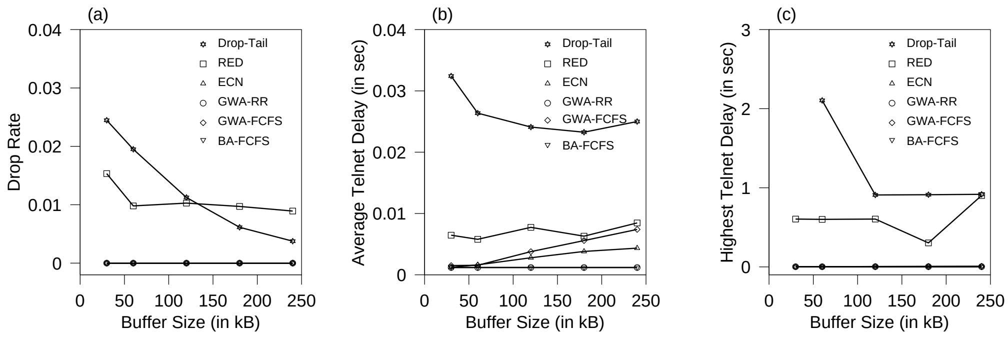

Figure 5: (a) Packet drop probability, (b) average delay of Telnet packets, and (c) maximum delay of Telnet packets versus the buffer size in LAN environment. For Drop-Tail case, the maximum delay is about 12 s when buffer size is 30 kbytes.

exact number of active flows ‘a priori’ show a shorter, smoother transient, but this knowledge is unrealistic and results are not included here.

Consider now the right column of figure 6 (buffer size behavior). The buffer occupancy at steady state is similar in both GWA-RR and GWA-FCFS. If only FTP connections were present, the steady state level would be about 1/2 of the buffer size. This is explained by the fact that the stable operating point for each TCP connection is when the advertised window is equal to the data transmitted but not acknowledged. So, every time a packet is ACKed, a new one is transmitted. Since the propagation delay is negligible, all the non ACKed packets are in the bottleneck queue. The equilibrium is reached when the buffer is half full. The plots in figure 6 show a smaller buffer occupancy. The reason is that Telnet connections offer less traffic than the maximum allowed. Telnet connections detected as being active are counted in n_f and reduce the buffer occupancy. As expected the buffer occupancy with BA-TCP is close to zero.

6.2 WAN ENVIRONMENT

Wide Area Network (WAN) behavior is influenced, if not dominated, by propagation delay. Difference in propagation delay between different connections may lead to unfairness: it is well known that TCP throughput is inversely proportional to the round trip time. When competing for shared resources, longer connections are penalized. With reference to figure 3, WAN experiments are run by setting the bottleneck link speed to 155 Mbit/s (the speed of the other links is 100 Mbit/s). $\tau_B = \tau_r = 5$ ms. The other propagation delays (τ_i in figure 3) are uniformly distributed between 1 an 10 ms. The window scale option for TCP window is activated, setting the maximum window size to 512 kbytes to allow full link utilization.

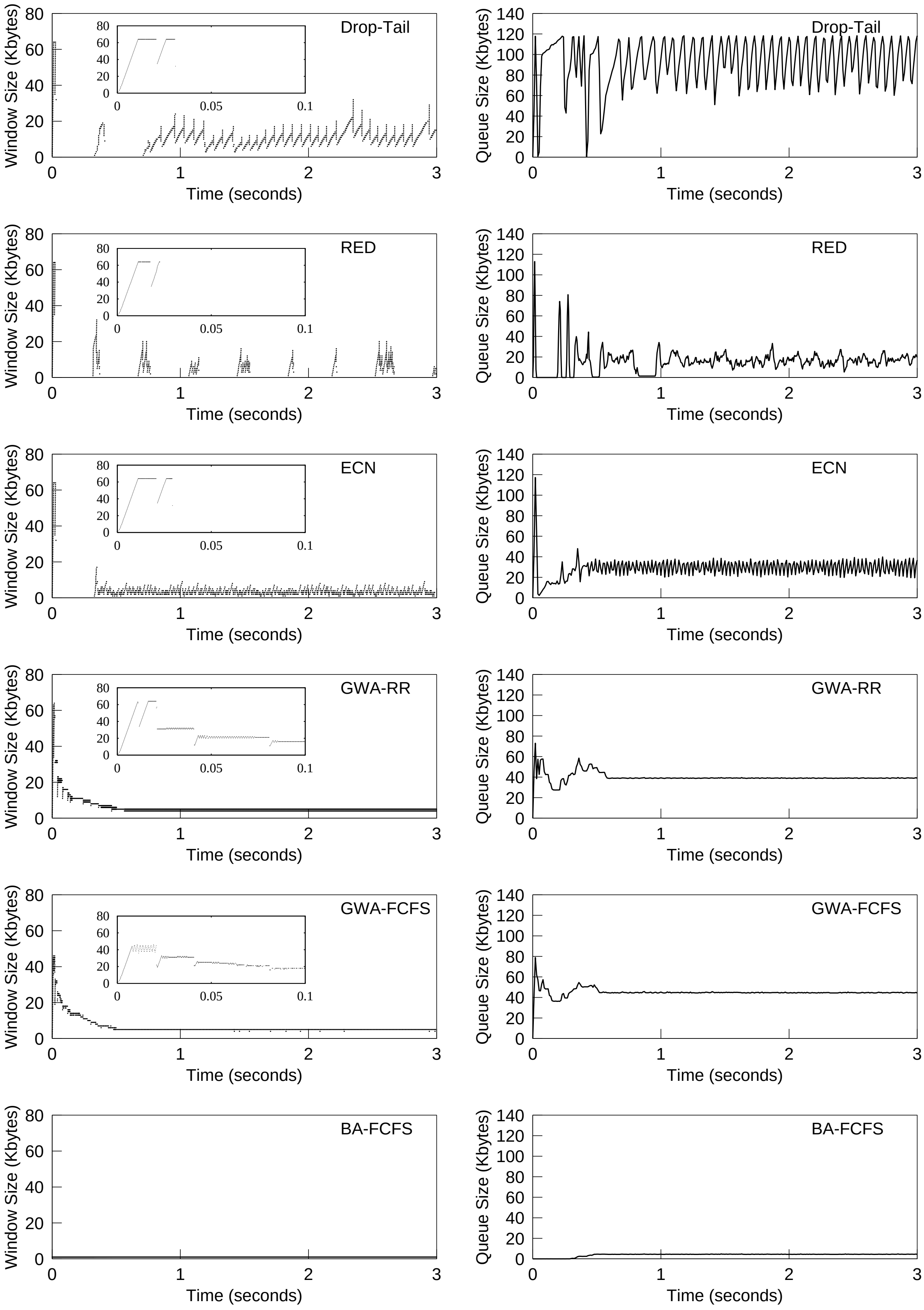

Figure 6: Window dynamic (left column) and bottleneck buffer dynamic (right column) versus time in LAN environment.

The average number of MSS packets (or ACKs) that are outstanding in the network at any time is about 450. Both GWA theory and heuristic considerations (see [24] for example) tell that the minimum buffer space for satisfactory network operation is around 500 kbytes.

Figure 7 shows the average, maximum and minimum throughput achieved by FTP connections for the 5 TCP implementations. Experiments last 25 s, and results exclude the first 5 s. The average throughput performance over all connections is generally satisfactory for all implementations. Perfect fairness is achieved with GWA-RR and BA-FCFS. Unfortunately GWA-RR requires a large buffer to support full link utilization, as predicted by the theory. BA-FCFS, on the other hand, exploits the virtual buffer provided by the propagation pipe and is does not suffer small buffers. Indeed the same performance can still be achieved with buffers as small as 64 or 128 kbytes. The fairness improvement as the buffer size increases for GWA-FCFS is explained by the increased buffer occupancy and therefore increased round trip time (RTT) seen by TCP sources. Recall that the throughput difference is proportional to the *relative* RTT difference among connections. ECN fairness behavior is due to the same reason, although less evident due to the buffer oscillations (see figure 8).

The throughput performance is consistent with the packet loss and delay results reported in figure 9. The loss-free properties of GWA and ECN implementations lead to good delay performance of real time applications like Telnet. This is a definite advantage of both GWA and ECN algorithms. Note, however, that while GWA *guarantees* loss-free operation, the same can not be said for ECN. Due to the large buffer size and propagation delays, the drop rate for Drop-Tail and RED is quite low in this case, never exceeding 10^{-3}.

Figure 8 show the window and buffer dynamic behavior in a WAN environment for 2 Mbyte router buffer. GWA transient behavior is quite good, although a high frequency ripple is noted, especially with the RR discipline. This is due to measurement errors and jitters, both in n_f and in the available buffer space. For this reason the RR implementation is more "noisy", since the buffer measure is not averaged over the connections. In addition, as discussed in section 4, the discrete nature of the system can cause small oscillations that add up to those due to measurement errors. BA-FCFS shows once more the most stable and smooth behavior; notice that the buffer scale is 4 times larger than in other cases to allow appreciating its behavior.

Drop-Tail, RED and ECN behave as expected. Notice that there are fewer timeouts here than in the LAN scenario: In this case the packet drop rate for RED and Drop Tail is less than 10^{-3}, while it was 10^{-2} in the LAN experiment.

6.3 Behavior with many flows

One of the concerns in traditional TCP implementations is the degradation of performance when the number of connections competing for the same bottleneck becomes large. Two recent papers [4, 30] highlight the problem and give good explanation of this phenomenon. Essentially, TCP connections oscillate between timeout state and active state. Besides, if a large number of flows are simultaneously active, their windows are small. This implies that the load offered to the network increases very fast even during congestion avoidance, since the window is increased by 1 packet for every ACK returned, up to the congestion threshold.

The definition of what is a "large number of connections" (or flows) is somewhat vague. However, both [4] and [30] agree in defining the critical number roughly equal to the total number of packets that can be stored in the bottleneck buffer plus the number of packet and ACKs that are "in flight" between source and destination.

Unfortunately, due to the limited scalability of *ns-2* and the insufficient computer resources, it was impossible to run true "many flow" experiments in a WAN scenario with 155 Mbit/s bottleneck link. To overcome in part this problem, and using again the network in figure 3, all link speeds are reduced to 10 Mbit/s. Propagation delays are equal to those given in section 6.2. The number of "in flight" packets and ACKs is around 50; the number of FTP connections is $N = 100$. Telnet connections are not simulated in this case since the main focus is on throughput behavior.

The router buffer space ranges from 200 to 800 kbytes. Thus, the sum of packet buffers and packets in flight is always larger than the number of connections. Strictly speaking, we are below the "many flows" threshold. However, the experiments exhibit the very pronounced unfairness trends typical of the "many flows" behavior. Simulation experiments are run for 100 s and the first 10 s are discarded as transient. Simulation runs are longer in this case to allow collecting statistically meaningful results for all the connections.

Figure 10 reports the steady state results for the 5 considered implementations. Plots (a)–(e) reports the average, maximum, and minimum throughput; plot (f) reports the drop rate for all cases. This latter plot shows that GWA and ECN still ensure loss-free operation, while the drop rate for Drop-Tail and RED are nearly three orders of magnitude higher than when only 10 FTP connections were present.

Observing the throughput performance in plots (a)–(e) of figure 10, the limitations of binary feedback schemes based on statistical principles become evident. On one hand, the average throughput performance is excellent for any TCP implementation. In fact, full bottleneck utilization is achieved in all schemes. On the other hand, when fairness is considered, neither Drop-Tail, nor RED, nor ECN provide satisfactory performance. Indeed plot (c) shows that ECN works properly when the buffer can hold several packets per connection. It becomes grossly unfair when the buffer size is not large enough. Basically, some of the connections never get a chance to transmit. We did not attempt a careful optimization of the ECN and RED pa-

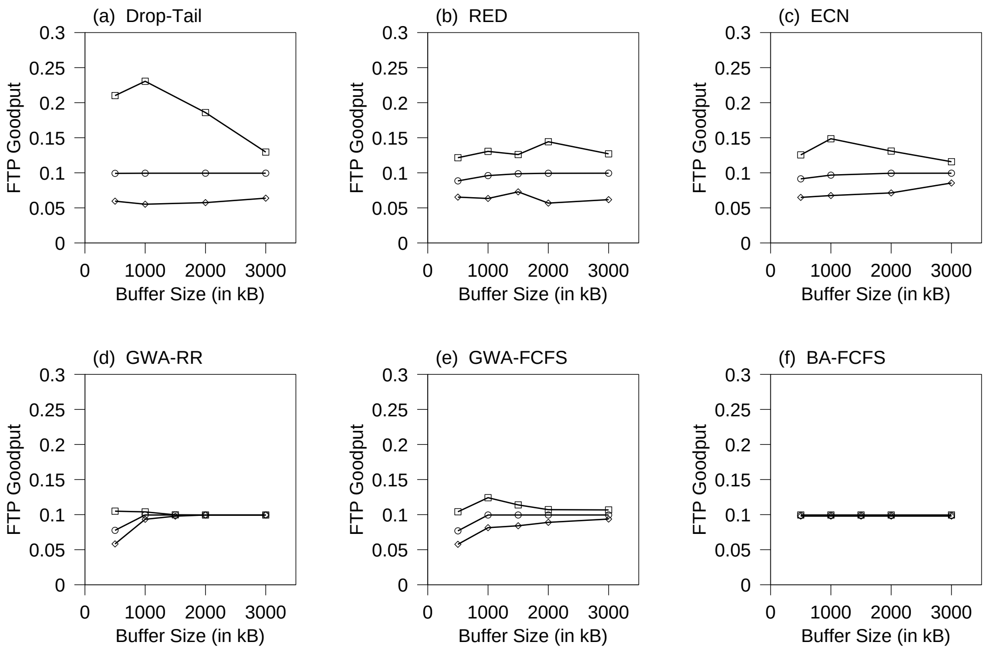

Figure 7: Receiver throughput for the 5 considered implementations versus the router buffer size in WAN environment.

rameters. Thus, we are not claiming that these schemes do never work properly. However, we wish to point out that the results suggest at the very least the need of rather sophisticated, configuration dependent, tuning of the RED router parameters.

Both GWA implementations perform remarkably well in a "many flows" situation, with full link utilization and perfect fairness. Moreover, in comparing figure 10(e) with figure 7(e), one is surprised to find that GWA-FCFS fairness actually improves when increasing the number of active connections and reducing the available bandwidth. The reason is the following. When few connections share a high bandwidth link, like in figure 7(e), several packets per connection are "in flight" within the network. If the buffer is not very large, the number of queued packets is smaller or equal to the number of packets "in flight" (see for instance figure 8). In this case the throughput obtained by each connection is inversely proportional to RTT, leading to unfairness. When there are many connections sharing a smaller amount of bandwidth, each connection has at most 1 packet in flight in the network: in the case considered here there are about 50 in flight packets and 100 connections. The number of queued packets is larger than the number of in flight packets. The FCFS queuing delay has an equalizing effect of RTT and therefore on throughput.

7 Concluding remarks

The Generalized Window Advertising (GWA) algorithm presented in this paper is a new approach to TCP congestion control. The scheme relies on the cooperation between IP network entities (routers) and host based TCP entities. The window control algorithm is based on a classical control theory approach known as Smith Predictor. While the original Smith Predictor scheme requires the knowledge of the round trip time, the paper shows that such explicit knowledge is not needed in TCP, since it is sufficient to monitor the amount of data that has been transmitted, but not yet acknowledged. This value is normally monitored by TCP. The theoretical analysis requires per flow queuing, but the paper shows that even FCFS routers can achieve satisfactory performance, greatly reducing complexity (and cost) of GWA implementation. A modified version of GWA, based on available bandwidth measures and named *Bandwidth Aware* TCP, overcomes what is probably the only drawback of GWA: the large buffer requirements within routers. BA-TCP ensures smooth and fair network operation under any circumstance, while maintaining buffers almost empty.

All GWA schemes presented were implemented in the *ns-2* simulator, and they were compared with traditional TCP implementations and buffer management schemes. Results show that GWA in general attains a more stable network operation as well as a higher degree of fairness. It guarantees a loss free operation as predicted by theory. Throughput performance depends on propagation delays and router buffer size; in typical operating conditions, GWA ensures full link utilization.

GWA is backward compatible with all TCP versions. Namely, GWA- and non-GWA TCP hosts can always communicate with each other. In addition it does not require extensive modifications to the existing protocols.

Manuscript received on February 10, 2000

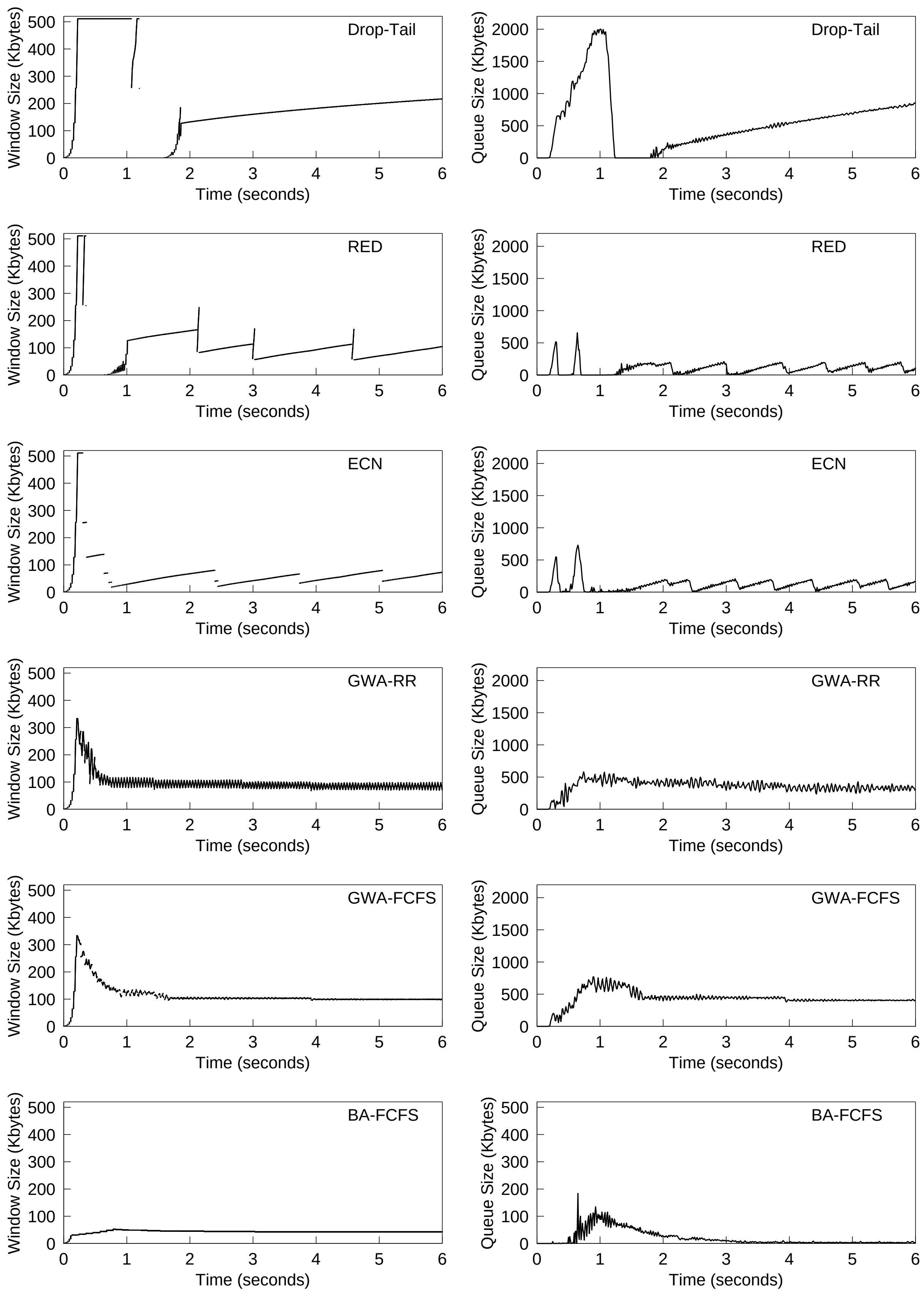

Figure 8: Window dynamic (left column) and bottleneck buffer dynamic (right column) versus time in WAN environment.

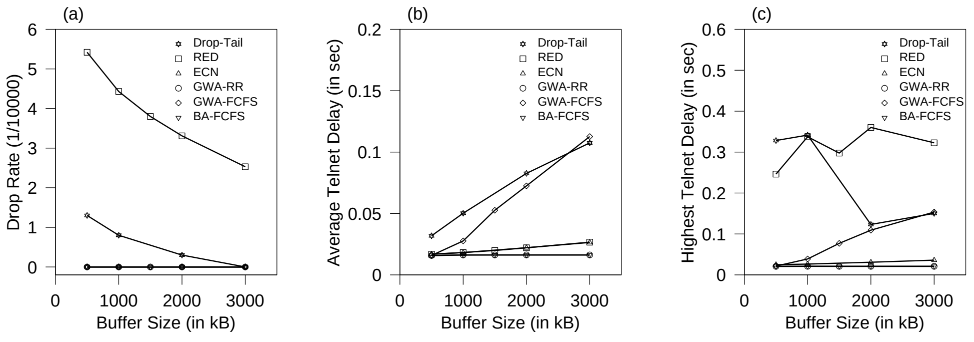

Figure 9: (a) Packet drop probability, (b) average delay of Telnet packets, and (c) maximum delay of Telnet packets versus the buffer size in WAN environment.

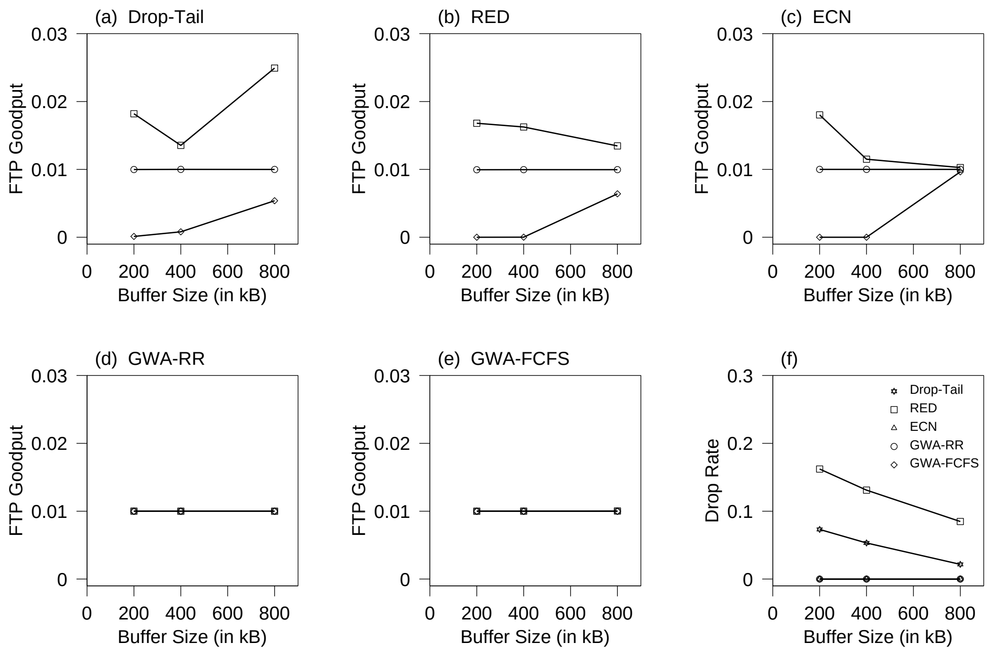

Figure 10: Throughput and loss probability with Many Flows.

REFERENCES

[1] T. V. Lakshman, U. Madhow. The Performance of TCP/IP for Networks with High Bandwidth-Delay Product and Random Loss. *IEEE/ACM Transactions on Networking*, Vol. 3, No. 3, Pages 336–350, June 1997.

[2] M. Gerla, R. Bagrodia, L. Zhang, K. Tang, L. Wang. TCP over Wireless Multihop Protocols: Simulation and Experiments. In *IEEE ICC'99*, Vancouver, Canada, June 1999.

[3] M. Gerla, K Tang, R. Bagrodia. TCP Performance in Wireless Multihop Networks. In *IEEE WMCSA'99*, New Orleans, LA, Feb. 1999.

[4] R. Morris. TCP Behavior with Many Flows. In *IEEE ICNP'97*, Atlanta, GE, USA, Oct. 28 – 31, 1997.

[5] S. Floyd. TCP and Explicit Congestion Notification. *ACM Computer Communication Review*, Vol. 24 No. 5, Pages 10–23, Oct. 1994.

[6] K. K. Ramakrishnan, S. Floyd. A Proposal to add Explicit Congestion Notification (ECN) to IP. RFC 2481, IETF, Jan. 1999.

[7] D. Lapsey, S. Low. Random Early Marking for Internet Congestion Control. In *IEEE Globecom'99* Rio de Janeiro, Brasil, Dec. 5–9, 1999.

[8] R. Jain. Congestion Control and Traffic Management in ATM Networks: Recent Advances and A Survey. *Computer Networks and ISDN Systems*, Vol. 28, No. 13, Pages 1723-1738, Oct. 1996.

[9] M. Ajmone Marsan, A. Bianco, R. Lo Cigno, M. Munafò. Four Standard Control Theory Approaches for the Implementation of RRM ABR Services. In: D. Kouvatsos (editor), *Performance Modelling and Evaluation of ATM Networks, Vol. 3* Chapman and Hall, London, 1997.

[10] S. Kalyanaraman, R. Jain, S. Fahmy, R. Goyal, B. Vandalore. “The ERICA Switch Algorithm for ABR Traffic Management in ATM Networks. *IEEE/ACM Transactions on Networking*, Vol. 8, No. 1, Feb. 2000.

[11] A. Kolarov, G. Ramamurthy. A Control Theoretic Approach to the Design of Explicit Rate Controller for ABR Service. *IEEE/ACM Transactions on Networking*, Vol. 7, no. 5, Pages 781–753, Oct. 1999.

[12] L. Kalampoukas, A. Varma, K. K. Ramakrishnan. Explicit Window Adaptation: A Method to Enhance TCP Performance. In *IEEE Infocom'98*, San Francesco, CA, USA, March 29 – April 2, 1998.

[13] J. Golestani, S. Bhattacharyya. A Class of End-to-End Congestion Control Algorithms for the Internet. In *IEEE ICNP'98*, Oct. 1998.

[14] L. Brakmo, L. Peterson. TCP Vegas: End to End Congestion Avoidance on a Global Internet. *IEEE Journal on Selected Areas in Communications*, 13(8), Oct. 1995.

[15] J.S. Ahn, P.B. Danzig, Z. Liu, L. Yan. Evaluation of TCP Vegas: Emulation and Experiment. *IEEE Transactions on Communications*, 25(4), Oct. 1995.

[16] J. Mo, R.J. La, V. Antharam, J. Walrand. Analysis and Comparison of TCP Reno and Vegas. In *IEEE INFOCOM 1999*, New York, NY, USA, March 1999.

[17] F.P. Kelly. Charging and Rate Control for Elastic Traffic. *European Transactions on Telecommunications*, Vol. 8(1), Jan. 1997.

[18] R.J. La, V. Ananthram. Charge-Sensitive TCP and Rate Control in the Internet. In *IEEE INFOCOM 2000*, Tel Aviv, Israel, March 2000.

[19] M. Gerla, W. Weng, R Lo Cigno. BA-TCP: A Bandwidth Aware TCP for Satellite Networks. In *IEEE ICCCN'99*, Boston-Natick, MA, USA, Oct. 11-13, 1999.

[20] M. Gerla, W. Weng, R Lo Cigno. Enforcing Fairness with Active Network Feedback in the Internet. In D. Skellern, A. Guha, F. Neri Ed. *Evolving Access and Networking Techniques*, selected papers of the *10th IEEE Workshop on Local and Metropolitan Area Networks (LANMAN'99)*, IEEE press, 2001.

[21] M. Gerla, W. Weng, R. Lo Cigno. Bandwidth feedback control of TCP and real time sources in the Internet. In *IEEE Globecom 2000*, San Francisco, CA, USA, Nov. 27 – Dec. 1 2000.

[22] S. Mascolo. Congestion Control in High-speed Communication Networks using the Smith Principle. *Automatica Journal, Special Issue on 'Control Methods for Communication Networks'*, Eds. J. Walrand and V. Anantharam, Dec. 1999.

[23] ns-2 Network Simulator. http://www-mash.cs. berkeley.edu/ns/, 1999.

[24] C. Villazamir, C. Song. High Performance TCP in ANSNET. *ACM Computer Communication Review*, Vol. 4, No. 5, Oct. 1995

[25] R. Stevens. *TCP/IP Illustrated*, Vols. 1 & 2 Addison Wesley, 1994.

[26] J. Postel. Transmission Control Protocol – DARPA Internet Program Protocol Specification. RFC 793, DARPA, Sept. 1981.

[27] W. Stevens. TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms. RFC 2001, IETF, Jan. 1997.

[28] O. Smith. A Controller to Overcome Dead Time. *ISA Journal*, Vol. 6, No. 2, Pages 28–33, 1959.

[29] S. Floyd, V. Jacobson. Random Early Detection Gateways for Congestion Avoidance. *IEEE/ACM Transaction on Networking*, Vol. 1, No. 4, Pages 397–413, Aug. 1993

[30] C. M. Pazos, J. C. Sanchez Agrelo, M. Gerla. Using Back-Pressure to Improve TCP Performance with Many Flows. In *IEEE INFOCOM'99*, New York, NY, USA, 21st - 25th March 1999.