Improving Ensemble-TCP Performance on Asymmetric Networks

Qian Wu Carey Williamson
Department of Computer Science
University of Saskatchewan
Email: `carey@cs.usask.ca`

Abstract

For the World Wide Web, the Transmission Control Protocol (TCP) and the HyperText Transfer Protocol (HTTP) are two important protocols. However, interactions between these two protocols, combined with the bandwidth asymmetry of network access technologies such as ADSL, can lead to inefficient HTTP/TCP performance.

This paper investigates the effects of bandwidth asymmetry on Ensemble-TCP, a protocol designed to coordinate multiple HTTP/TCP connections to improve Web document transfer performance. The evaluation is conducted using the ns-2 network simulator. The paper also proposes and evaluates two new schemes, ACC-Ensemble and AF-Ensemble, to improve Ensemble-TCP performance on asymmetric networks. The simulation results show that AF-Ensemble and ACC-Ensemble perform better (often 10-50% faster) than Reno TCP and Ensemble-TCP for typical user-level Web browsing activity on asymmetric networks.

1. Introduction

Web traffic continues to grow at an alarming pace, constituting an ever-increasing fraction of the traffic on the Internet.

The client-server architecture of the Web uses a request-response application-layer protocol called HTTP (HyperText Transfer Protocol). HTTP typically relies on an underlying connection-oriented transport-layer protocol called TCP (Transmission Control Protocol [15]) to provide reliable data transfer for Web documents, though other protocol choices are also possible [9].

Although HTTP uses services provided by TCP to transfer the Web documents, HTTP and TCP do not always "fit together" well. For example, the three-way handshake for connection setup and release in TCP adds extra packets and round-trip time (RTT) delays to HTTP transactions, particularly if a separate TCP connection is required for each Web object. Each TCP connection often undergoes its own slow-start as well.

These unfortunate protocol interactions are well-known, and have been the impetus behind several enhancements to HTTP and TCP protocols. These improvements include: caching and reusing recent TCP connection state information (e.g., Transaction-TCP [4], slow-start threshold estimation [10]); allowing multiple concurrent TCP connections between a Web browser and a server in HTTP/1.0; the use of persistent connections and pipelining in HTTP/1.1 [3, 6, 14]; and coordinating multiple TCP connections as one aggregate connection (with shared RTT and congestion control state information) in Ensemble-TCP [5].

The foregoing approaches have all enhanced HTTP and/or TCP performance. However, the evaluation of these protocols has, for the most part, been conducted assuming *symmetric* network access technologies (i.e., the client-to-server channel and the server-to-client channel have similar properties, in terms of bandwidth, latency, channel access protocol, and error rate).

Symmetry is not always present with today's diverse network access technologies. For example, technologies such as ADSL (Asymmetric Digital Subscriber Line) are explicitly designed to offer asymmetric bandwidth: low capacity for the upstream client-to-server request stream, and high capacity for the traffic-intensive download path from server-to-client. Asymmetry is also possible in wireless networks, satellite-based networks, hybrid fiber-coax networks, and in emerging technologies.

Asymmetric networks in and of themselves can lead to performance problems for TCP [2, 11, 13]. This can reduce the effectiveness of HTTP and TCP for Web data transfer over asymmetric networks [8].

This paper investigates the effects of bandwidth asymmetry on Ensemble-TCP (E-TCP) [5] in particular. Based on the performance degradation observed for E-TCP in the presence of bandwidth asymmetry, two new schemes – AF-Ensemble and ACC-Ensemble – are proposed and evaluated. The *ns*-2 network simulator [1] is used for all of the experiments.

In essence, the contribution of this paper is to combine the features of TCP control block (TCB) sharing from E-TCP [5] with the ACK filtering (AF) and ACK congestion control (ACC) mechanisms from [2], to improve E-TCP performance over asymmetric networks. The simulation results show that AF-Ensemble and ACC-Ensemble perform better than Reno TCP and E-TCP for typical user-level Web browsing activity. Furthermore, the relative performance of the evaluated approaches is robust, even in the presence of packet losses on congested networks or noisy ADSL links.

The remainder of this paper is organized as follows. Section 2 identifies performance issues related to HTTP, TCP, and bandwidth asymmetry, and describes related work to address these performance problems. Section 3 illustrates the impact of bandwidth asymmetry on E-TCP. Section 4 describes the design of the newly proposed schemes (AF-Ensemble and ACC-Ensemble) to improve E-TCP performance. Section 5 presents the experimental methodology to evaluate the proposed schemes. Section 6 presents and analyzes the simulation results. Finally, Section 7 concludes the paper.

2 Background and related work

2.1 Web, HTTP, and TCP performance issues

As indicated earlier, several subtle interactions occur between HTTP and TCP in the context of the Web. A Web page is usually constructed from HTML documents and a number of embedded images. Since HTTP/1.0 creates a new TCP connection for each document, there are typically multiple TCP connections involved in transferring a Web page.

One approach to deal with the multiple TCP connections is to launch them concurrently. Although the concurrent connections can overlap the setup and transfer latencies, there are also side effects due to these parallel independent connections: the connections compete for shared network resources, and can lead to network congestion. Besides the risk of excessive aggregate traffic, creating a new TCP connection for each document also increases the overhead of the Web page transfer.

The short and bursty nature of most Web document transfers is another concern for HTTP and TCP. TCP uses its slow-start phase to probe the network capacity and prevent a connection from overwhelming the network. For short-lived connections, however, the connections may terminate before the slow-start phase is completed. These connections have little opportunity to determine the network capacity and adjust the TCP congestion window size properly. Therefore, the network bandwidth may not be used efficiently [9, 14].

Besides these protocol-specific issues, performance problems may also arise with network access technologies, such as ADSL that exhibit bandwidth asymmetry. For Web document transfer, this asymmetry means the link carrying acknowledgments (ACKs) from the client towards the Web server is much slower than the link carrying data packets from the server towards the client. Since TCP relies on the acknowledgments from the receiver to ensure reliable data transfer, and the progress of TCP's congestion window is ACK-clocked, the slower acknowledgment channel can constrain TCP performance [2, 11].

2.2 Related work

Two items of related work are central to this paper: E-TCP [5], and Asymmetric TCP [2]. The following subsections provide some background on each of these.

2.2.1 Sharing TCP state information

TCP control block (TCB) sharing allows network state information to be shared amongst TCP connections. Transaction-TCP (T/TCP) proposed by Braden [4] is an example of *temporal sharing*: a new connection can use the cached TCB information from an earlier connection. *Ensemble sharing*, proposed by Touch [16], complements temporal sharing by allowing TCB information to be shared across *concurrent* connections. An implementation example of ensemble sharing is E-TCP, designed by Eggert, Heidemann, and Touch [5].

E-TCP groups related connections together as an *ensemble* based on source and destination host-pairs. In E-TCP, the connections in an ensemble share connection information stored in a structure called an *Ensemble Control Block (ECB)*. E-TCP's congestion control mechanism operates on a per-ensemble basis, with the aggregated connections operating as one (larger, coordinated) TCP flow. A total of four scheduling strategies are proposed in [5]: *Fair-Share, Ticket-Decay, Content-Dependent, and HTML-Priority*. We use these same four strategies in our simulation evaluation of E-TCP.

2.2.2 TCP on asymmetric networks

Several researchers have studied TCP performance on asymmetric networks [2, 8, 11, 13]. To solve the problems caused by bandwidth asymmetry, Balakrishnan et al. proposed such strategies as ACK Congestion Control (ACC), ACK Filtering (AF), Sender Adaptation (SA), and ACK Reconstruction (AR) to reduce the congestion in the upstream direction and improve TCP performance [2].

The main idea of ACC and AF is to alleviate congestion on the upstream link by reducing the frequency of ACKs. ACC determines congestion state by checking the Explicit

Congestion Notification (ECN) bit in packet headers [7]. If this bit is set, ACC decreases the frequency of ACKs. AF can react to congestion more promptly. This approach exploits the cumulative property of TCP ACKs by removing older ACKs for the same connection when the queue becomes too full. SA and AR are used to adapt the sender's behaviour properly to the (intentionally) reduced ACK frequency.

2.2.3 Summary

Although the foregoing strategies have addressed some performance problems, there is no scheme that can simultaneously solve problems induced by both protocol mismatches and bandwidth asymmetry. As can be seen in the next section, a scheme addressing only the mismatches between HTTP and TCP still leaves room for improving Web data transfer performance on asymmetric access networks. This motivates combining strategies to further improve the performance of HTTP transactions on asymmetric networks.

3 Bandwidth asymmetry and E-TCP

This section presents a simple simulation experiment to investigate the effects of bandwidth asymmetry on E-TCP. The simulations are conducted using *ns*-2, and the E-TCP models developed in [5] for *ns*-2. The results in [5] show that E-TCP, with its use of TCB sharing, outperforms Reno TCP on symmetric networks. Our goal is to evaluate performance on asymmetric networks.

3.1 Network topology

Figure 1 shows the simple network topology used in the experiments. A Web client and a Web server are connected via a router at an Internet Service Provider (ISP).

Bandwidth asymmetry exists in this network. The link capacity from the router to the client (1.5 Mbps) is higher than that from the client to the router (settable, ranging from 28.8 kbps to 56 kbps). For comparison, the transfer time on symmetric networks (same capacity in both the upstream and downstream directions) is also considered.

In all experiments, the link capacity between the server and the router matches that from the router to the client, for simplicity. All buffers have room for 10 packets, and all router queues are FIFO (First-In-First-Out). Each link's propagation delay is as shown in Figure 1. The Maximum Segment Size (MSS) for TCP is 512 bytes, and the ACK size is 40 bytes.

3.2 Experimental design

The performance of E-TCP is evaluated by measuring the time to download a Web page. The structure of the Web page in this experiment is identical to the "Text Page" used in [5], for comparison purposes. This page contains one base HTML document (29,058 bytes) and three embedded images (7,097 bytes, 13,329 bytes, and 13,329 bytes, respectively) .

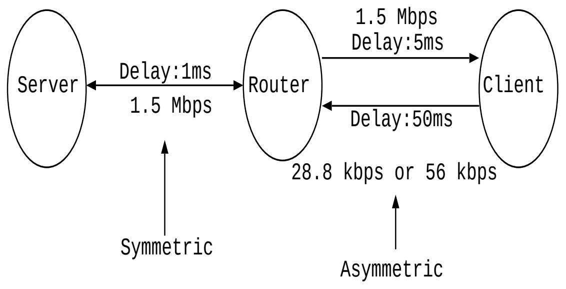


Figure 1. Network topology for the evaluation of E-TCP on asymmetric networks

Two performance metrics are considered: the transfer time for the base HTML document, and the transfer time for the complete Web page. The former value reflects the time at which a Web browser can render the HTML document for the user, while the latter value reflects the time at which the entire Web page (including embedded images) is complete.

The experiments are conducted using a full-factorial design. The factors are the upstream link capacity, and different scheduling strategies within E-TCP. The levels used for these factors are indicated in Table 1.

Table 1. Experimental factors and levels for evaluating E-TCP on asymmetric networks

Factor	Levels
Upstream Link Scheduling	28.8 kbps, 56 kbps, 1.5 Mbps, Fair-Share, Content-Dependent, Ticket-Decay, HTML-Priority

3.3 Experimental results

Table 2 summarizes the simulation results for E-TCP for a 1.5 Mbps downstream link.

As expected, the results show that transfer times are higher in the asymmetric network case, when the upstream link capacity is lower. For example, with an upstream link capacity of 56 kbps, the total transfer time is 38% higher (0.965 seconds vs. 0.699 seconds) than for the symmetric situation, regardless of the scheduling policy used.

To illustrate the effect of the slow ACK channel more clearly, Figure 2 shows TCP sequence number plots for two

Table 2. Transfer times (in seconds) for E-TCP (1.5 Mbps downstream link)

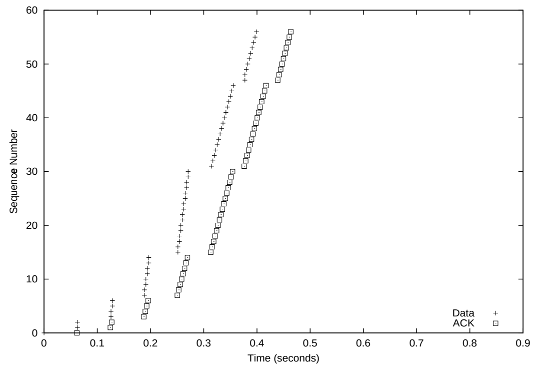
Scheduling Policy	Transfer Time	Upstream Link Capacity		
		28.8 kbps	56 kbps	1.5 Mbps
Fair	HTML	1.451	0.965	0.699
	Total	1.462	0.965	0.699
Ticket	HTML	1.451	0.965	0.699
	Total	1.451	0.965	0.699
Content	HTML	1.351	0.799	0.601
	Total	1.551	0.965	0.699
HTML	HTML	0.806	0.576	0.465
	Total	3.513	0.965	0.699

selected scenarios: a symmetric network (1.5 Mbps each way, in Figure 2 (a)), and an asymmetric network (28.8 kbps upstream, and 1.5 Mbps downstream, in Figure 2 (b)). The HTML-Priority scheduling policy is used for each.

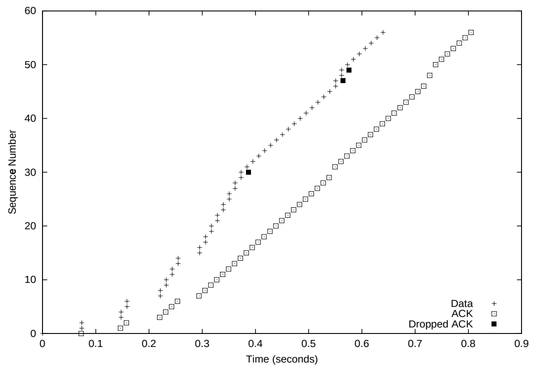
The TCP plots show the times at which data packets (with a specific sequence number) are transmitted, as well as the times at which the corresponding ACK packets are sent. More precisely, for data packets, the horizontal axis shows when a data segment enters the outgoing queue of the Web server. For ACK packets, the horizontal axis shows the time when an ACK reaches the router (i.e., after traversing the constrained upstream link), or is dropped at the Web client's outgoing queue (the bottleneck point). In all cases, the vertical axis shows the sequence number or ACK number carried in the packet.

The graphs in Figure 2 show that the slow upstream link flattens the throughput (packets sent per unit time) for TCP. Furthermore, with continuous arrivals of new ACKs when the congestion window size is large, the gap between the sequence number and ACK number lines becomes wider (see Figure 2 (b)), reflecting greater queueing delay. That is, the slower upstream link delays the arrival of an ACK at the server, and stretches the original spacing of ACKs. The delayed ACK arrivals in turn slow down the growth of the sender's congestion window (compare Figure 2 (a) with Figure 2 (b), noting in particular the timing structure of the ACKs, and the sizes of the window-based data bursts).

Three ACKs are actually dropped in this experiment due to overflow at the client buffer (see the black squares in Figure 2 (b)). The horizontal placement of these ACKs shows the time at which the queue overflowed, causing the loss. The vertical placement of the ACK shows the sequence number involved. The corresponding gaps in the ACK numbers returning to the server are evident. Fortunately, these lost ACKs did not trigger any data packet retransmissions in this example; their main effect is to slow the growth of the sender's congestion window.



(a) Data and ACK Sequence Number Plots on Symmetric Network (1.5 Mbps)



(b) Data and ACK Sequence Number Plots on Aymmetric Network (28.8 kbps Upstream, 1.5 Mbps Downstream)

Figure 2. TCP sequence number plots for HTML document transfer using E-TCP

4 Improving E-TCP

This section describes two new schemes – ACC-Ensemble and AF-Ensemble – to improve Web data transfer performance on asymmetric networks. These schemes combine the features of TCB sharing (from E-TCP [5]) and ACK congestion control (from [2]).

- ACC-Ensemble

  ACC-Ensemble combines features of E-TCP, ACK Congestion Control (ACC), and Sender Adaptation

(SA). It requires modification to both the sender's and the receiver's TCP. The application layer can still use HTTP/1.0 to transfer Web data.

Since only ACC modifies the receiver's TCP stack, an ACC-Ensemble receiver can directly adopt all schemes that are designed for a receiver implementing ACC. That is, ACC uses the Random Early Detection (RED) [7] algorithm for active queue management at the bottleneck link. RED detects the onset of congestion by computing the average queue size, and notifies the receiver of congestion using the ECN bit. Upon receiving a set ECN bit, the receiver reduces the ACK frequency.

Since both E-TCP and SA modify the sending TCP, the sender of ACC-Ensemble is implemented by combining E-TCP and SA. E-TCP's congestion control mechanism is retained in ACC-Ensemble, since it treats connections within an ensemble as one aggregate TCP connection. To adapt to the receiver sending infrequent ACKs, the following features implemented in SA are added to the sender of ACC-Ensemble: (1) The congestion window is increased by counting how many segments are acknowledged in an ACK, rather than counting the number of received ACKs; (2) An upper bound is placed on the number of data packets sent back-to-back. If the number of data packets to be sent exceeds the upper bound, the extra data is scheduled using *burstsnd_timer* [2]; (3) An improved heuristic is added for handling delayed ACKs. E-TCP does not have an appropriate scheme to deal with delayed ACKs. Upon an arrival of an ACK for a connection (such as Connection 1), if there is an earlier outstanding segment of another connection (such as Connection 2), E-TCP will increment the duplicate ACK counter of Connection 2. Obviously, this is not suitable for ACC's possibly reduced ACK frequency. It no longer exists in ACC-Ensemble. To efficiently and accurately trigger the fast retransmit in case of loss, the strategy proposed in [2] is used: when the receiver detects segment loss, a bit identifying if the fast retransmit is required is set for all duplicate packets. When the sender receives an ACK with the bit set, it retransmits the lost segment and modifies the congestion window size [2].

- AF-Ensemble

As described earlier, the ACK Filtering (AF) algorithm is also designed to reduce the frequency of ACKs, but it can react to congestion more promptly than ACC. Therefore, AF-Ensemble, which combines features of E-TCP, AF, and SA, is designed to improve upon the performance of ACC-Ensemble.

Table 3. Experimental factors and levels for lossless asymmetric networks

Factor	Levels
Downstream Link	1.5 Mbps, 4 Mbps, 8 Mbps
TCP Version	Reno/ACC/SA, Reno/AF/SA, Reno, E-TCP, ACC-Ensemble, AF-Ensemble

To form AF-Ensemble, the receiver of AF-Ensemble uses the AF algorithm to reduce the frequency of ACKs. That is, when an ACK enters the queue at the upstream bottleneck link, the queue is checked for any ACK for the same connection. If any, AF removes some or all of these previous ACKs from the queue based on the degree of queue fullness.

As in ACC-Ensemble, the sender in AF-Ensemble combines E-TCP and SA so that connections within an ensemble behave as one TCP connection, and adapt to infrequent ACKs. Also, AF-Ensemble uses the same strategy as ACC-Ensemble to trigger fast retransmit.

5 Experimental methodology

This section provides details on the network model, workload model, and experimental design for the simulation experiments.

5.1 Experimental design

The simulation experiments fall into two main categories: those for lossless asymmetric networks, and those for lossy asymmetric networks. In all experiments, the two performance metrics are the transfer time to deliver the base HTML component of each Web page, and the transfer time for delivering each Web page in its entirety.

The experimental design for the first (lossless) set of experiments is summarized in Table 3. The main factors are downstream link capacity, and version of TCP. Six TCP strategies are tested. To simplify the experiment, only the "HTML-Priority" scheduling policy is tested in all ensemble cases. HTML-Priority is chosen since it usually produces the fastest HTML document transfer time.

These experiments are conducted using the network topology shown in Figure 3. The bottleneck queue (from client to router) is a RED queue. The client-to-router link capacity is fixed at 56 kbps.

These experiments assume lossless transfers. That is, there is no packet loss in the downstream direction. The buffer size for the downstream direction is 50 packets, large

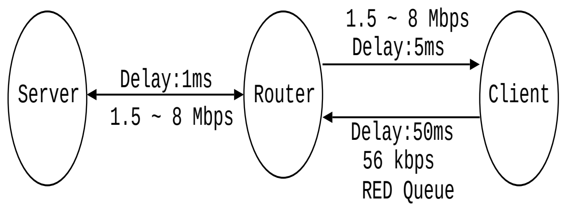


Figure 3. Network topology for experiments on lossless asymmetric networks

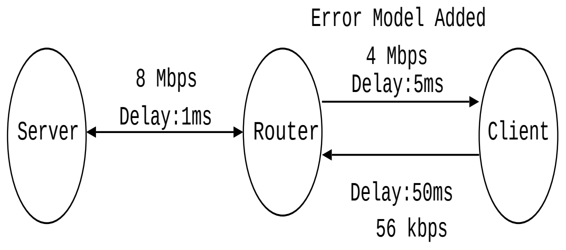


Figure 4. Network topology for experiments on lossy asymmetric networks

Table 4. Experimental factors and levels for lossy asymmetric networks

Factor	Levels
Packet Error Rate	0.0001 to 0.1
TCP Version	Reno/ACC/SA, Reno/AF/SA, Reno, E-TCP, ACC-Ensemble, AF-Ensemble

enough to hold any data burst. The buffer size from the client to the router is 10 packets.

The second (lossy) set of experiments uses the experimental design summarized in Table 4. A new factor, the packet error rate, is added to the experiments to assess the robustness of the proposed schemes to lost packets.

These experiments are conducted using the network topology shown in Figure 4. To simplify the experiments, all link capacities are fixed as shown. The buffer size for the link from router to client is small (10 packets); thus, congestion losses are possible. The buffer size from the client to the router is 10 packets.

An error model with a certain packet error rate is added to the router-to-client link. Thus it is possible that a data packet is dropped[1] with a certain probability due to errors. The packet error rate is one of the factors. Ten levels are used for the error rate, ranging from 10^{-4} to 10^{-1}. Error rates of 10^{-7} to 10^{-5} were also tested. Since their performance is similar to the error rate of 10^{-4}, their results are omitted to save space.

The use of random packet drops adds significant variability to the simulation results, since the document transfer performance depends not only on how many packets are dropped, but also which packets are dropped, and the size of the TCP congestion window at the time of the drop (since this can affect whether "fast retransmit" or a "coarse timeout" will be needed to recover from the loss). To compensate for this, simulation replications are used. That is, for each setting of the error rate, 32 independent runs are made (i.e., with different random number seeds), and 95% confidence intervals are calculated for the mean transfer times.

[1]TCP cannot distinguish packets lost due to congestion from packets lost due to transmission errors.

Table 5. Web workload characteristics

Web Page	HTML (bytes)	Num Images	Min (bytes)	Mean (bytes)	Max (bytes)
1	27,047	24	42	3,941	48,561
2	5,414	14	42	1,171	9,202
3	14,651	17	42	1,741	14,651
4	36,323	4	130	10,486	36,323
5	3,094	3	3,094	4,627	5,393
6	2,226	1	815	1,521	2,226
7	3,061	1	815	1,938	3,061
8	2,663	10	289	2,043	14,109
9	2,660	10	277	1,659	5,176
10	2,503	0	2,503	2,503	2,503

5.2 Workload assumptions

A simple user-level Web browsing session is modeled. This session contains ten Web pages accessed by a single user, over the span of several minutes. Table 5 summarizes the characteristics of the Web content accessed. The think times occurring in navigating Web pages are modeled using the cumulative distribution functions for user think times (based on [12]) in *ns*-2. Screen shots and URLs of the chosen Web pages are provided in [17].

6 Simulation results

6.1 Lossless asymmetric networks

The transfer times for the base HTML documents of the Web pages in the workload are shown in Figure 5. The X-axis represents each page contained in the workload[2] and the Y-axis represents the HTML file transfer time of each

[2]Technically, the points on these graphs should not be connected together with lines, since the Web pages represented along the horizontal axis are independent samples, accessed in sequence. However, the lines

page. Figures 5 (a) to (c) are the results with different downstream link capacities.

Figure 5 shows that AF-Ensemble outperforms other strategies in all cases, and ACC-Ensemble outperforms others (excluding AF-Ensemble) in most cases. In many cases, the transfer time with AF-Ensemble is 50% lower than with other TCP strategies. E-TCP has performance close to ACC-Ensemble, especially for transferring Page 5 to Page 10. Since AF-Ensemble, ACC-Ensemble, and E-TCP implement temporal sharing and use the *HTML-Priority* scheduling policy, the results illustrate that temporal sharing and giving the HTML document higher priority make it possible to transfer the HTML file faster.

Figure 6 shows the simulation results for the total transfer times. The X-axis represents each page contained in the workload and the Y-axis represents the total transfer time of each page. Figures 6 (a) to (c) are the results with different downstream link capacities.

The results show that AF-Ensemble outperforms all other strategies on transfers of Page 2 to Page 10, although it does not outperform Reno/AF/SA for transferring Page 1. For most cases, AF-Ensemble is 10-50% faster than other TCP strategies.

The relatively poor performance for AF-Ensemble on the first page is because most objects contained in Page 1 are small. For small documents, contention for network resources among concurrent connections is not serious, and the gains achievable with TCB state sharing are minimal. For the subsequent pages, however, temporal sharing helps AF-Ensemble outperform other strategies. Recall that temporal sharing allows a new connection to use the cached TCB information from an earlier connection. Therefore, the Page 2 transfer can use the cached TCB information from the transfer of Page 1. If the cached connection information is more accurate than the default TCP values, temporal sharing allows AF-Ensemble to utilize the network bandwidth more efficiently, outperforming the traditional TCP schemes, such as Reno/AF/SA, on subsequent page transfers.

ACC-Ensemble and E-TCP benefit less on subsequent page transmission, despite the use of temporal sharing. This illustrates that besides temporal sharing, an efficient ACK congestion control mechanism is another important factor affecting the total transfer time. ACC-Ensemble's ACK congestion control relies on the ECN bit. It needs time to inform a receiver of the congestion and let the receiver take an appropriate action. AF-Ensemble and Reno/AF/SA, on the other hand, do not need the feedback (ECN bit) coming from the TCP sender to perform their ACK congestion control. They simply drop ACKs belonging to the same connection according to the fullness of the buffer. Therefore, they can react to congestion more promptly than ACC-Ensemble does.

The simulation results demonstrate that for lossless asymmetric networks, temporal sharing and the ACK congestion control mechanism are the important factors affecting the total transfer time. For the HTML file transfer, temporal sharing, ACK congestion control, and the scheduling policy affect the performance.

6.2 Lossy asymmetric networks

Figure 7 shows the HTML file transfer times for a selected subset (the first three) of the Web pages used in the previous experiments (Complete simulation results are available in [17].) The X-axis represents the packet error rate and the Y-axis represents the transfer time for the base HTML document of each page. In the figures, each vertical bar represents the confidence interval (with the confidence level of 95%) surrounding each point, the mean from 32 replications.

Compared to the experimental results for lossless links, the experimental results shown here are complicated. Basically, there is no consistent order for the six strategies across the whole error rate scale and the whole workload. For low error rates (less than 0.003), the strategies using temporal sharing and the *HTML-Priority* scheduling policy (AF-Ensemble, ACC-Ensemble, and E-TCP) usually have better performance. At higher error rates, it is hard to say which strategy has better performance, since the confidence intervals overlap.

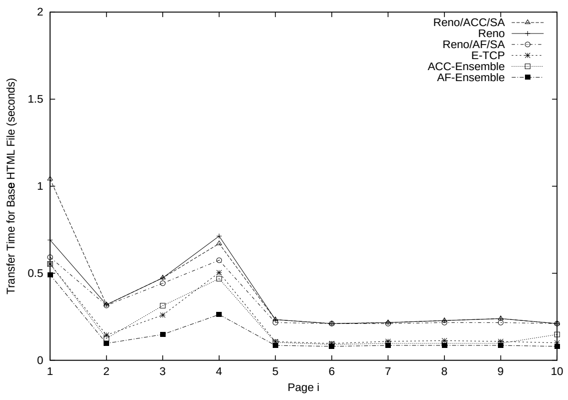
The confidence intervals become wider with the higher error rates, showing there is larger variation in the transfer time at these points.

Figure 8 shows the total transfer time results for the selected subset of Web pages. In each graph, the X-axis represents the error rate and the Y-axis represents the total transfer time of each page. Each vertical bar represents the confidence interval (with the confidence level of 95%) surrounding each point, the mean from 32 replications.

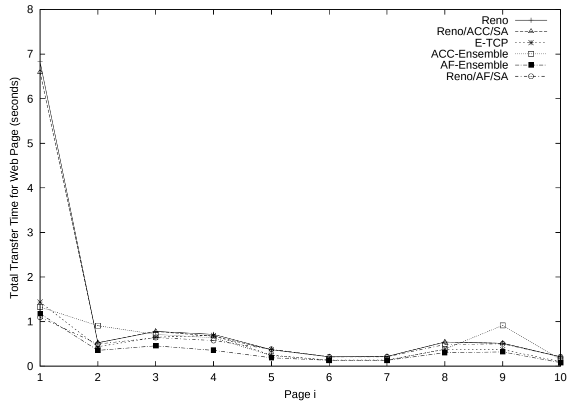
Similar to the HTML file transfer time, the results can be separated into two parts: lower error rates and higher error rates. At low error rates, Reno/AF/SA outperforms other strategies in Page 1 transmission. This also happened in the previous lossless experiments. For subsequent page transfers, the strategies implementing temporal sharing, such as AF-Ensemble, ACC-Ensemble, and E-TCP, usually have better performance than others, with AF-Ensemble often the best. At higher error rates, the confidence intervals overlap, so it is not possible to conclude which strategy is the best.

There is large variation in the total transfer time for all strategies when the error rate is high.

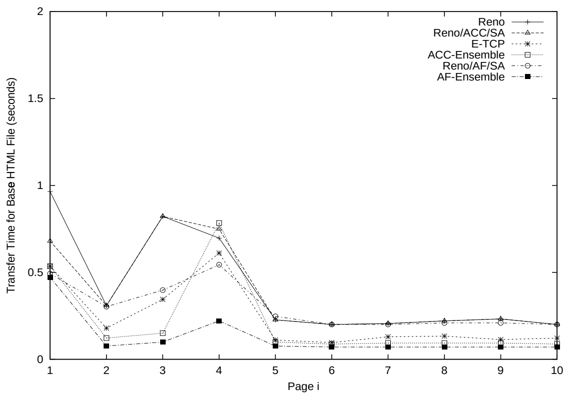
prove helpful in tracking the performance of each TCP algorithm, indicating the relative performance of each algorithm, and highlighting anomalous behaviours that occur.



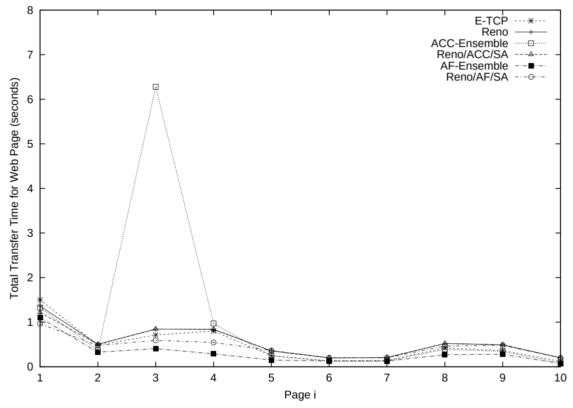
(a) Upstream: 56 kbps; Downstream: 1.5 Mbps



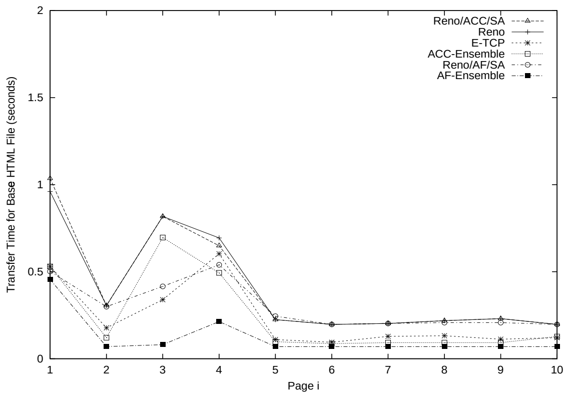
(a) Upstream: 56 kbps; Downstream: 1.5 Mbps



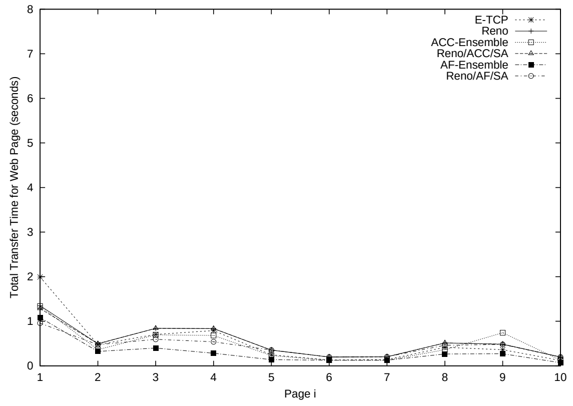
(b) Upstream: 56 kbps; Downstream: 4 Mbps



(b) Upstream: 56 kbps; Downstream: 4 Mbps



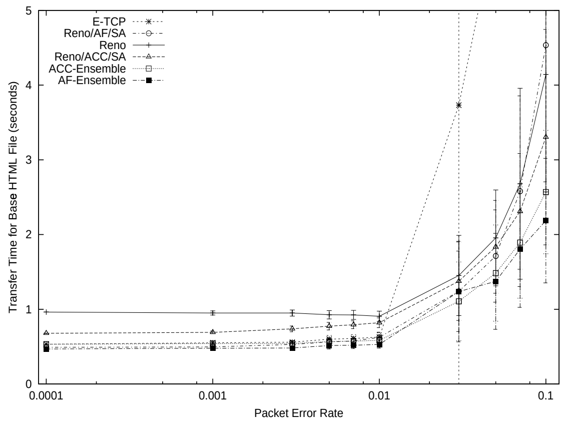
(c) Upstream: 56 kbps; Downstream: 8 Mbps



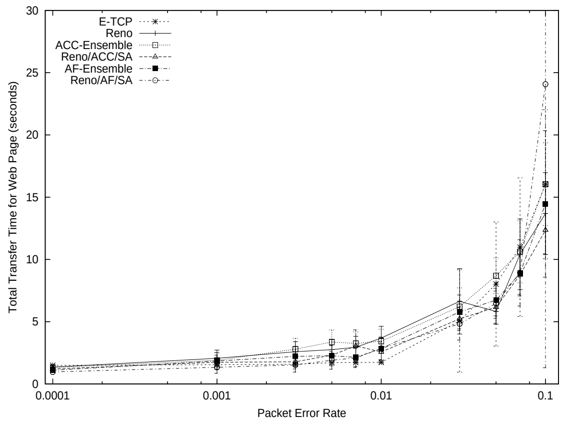
(c) Upstream: 56 kbps; Downstream: 8 Mbps

Figure 5. Transfer times for HTML files of Web pages on lossless asymmetric networks

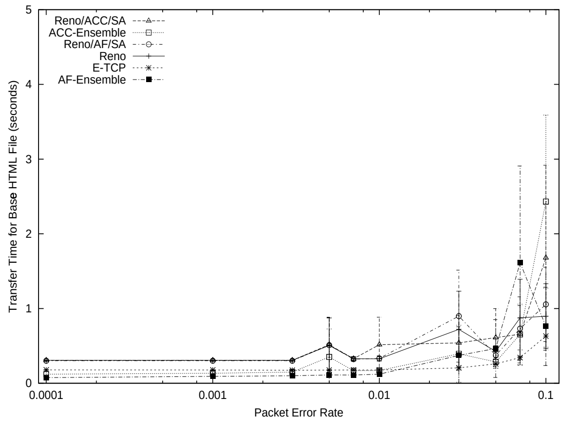
Figure 6. Total transfer times for Web pages on lossless asymmetric networks



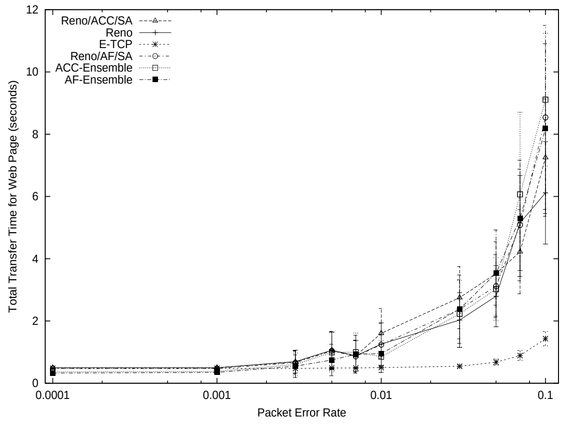
(a) Page 1: Upstream: 56 kbps; Downstream: 4 Mbps



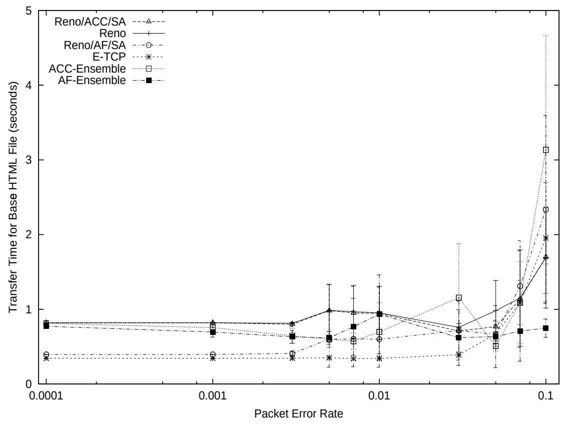
(a) Page 1: Upstream: 56 kbps; Downstream: 4 Mbps



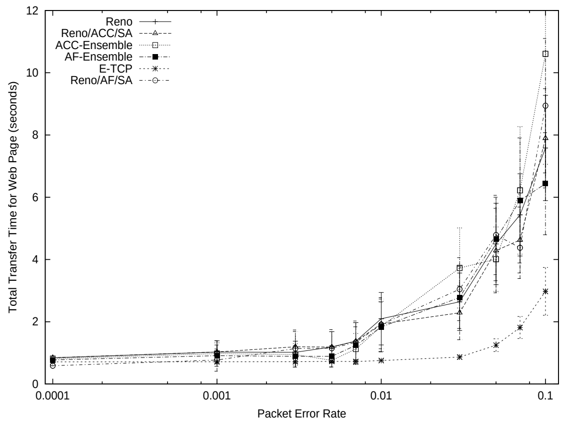
(b) Page 2: Upstream: 56 kbps; Downstream: 4 Mbps



(b) Page 2: Upstream: 56 kbps; Downstream: 4 Mbps



(c) Page 3: Upstream: 56 kbps; Downstream: 4 Mbps



(c) Page 3: Upstream: 56 kbps; Downstream: 4 Mbps

Figure 7. Transfer times for HTML files of Web pages on lossy asymmetric networks

Figure 8. Total transfer times for Web pages on lossy asymmetric networks

7 Conclusions

This paper studies the effects of bandwidth asymmetry on Web data transfer performance. Instead of using a traditional TCP scheme, a recently developed TCP strategy called E-TCP is tested. The experimental results identify that a scheme addressing only the mismatches between HTTP and TCP still leaves room for improving Web data transfer performance on asymmetric networks.

Based on the observation of performance degradation of E-TCP over asymmetric networks, we proposed two schemes – AF-Ensemble and ACC-Ensemble – which combine features of TCB sharing and ACK congestion control. These two schemes are evaluated on both lossless asymmetric networks and lossy asymmetric networks using a modeled Web browsing session as the workload.

The simulation results illustrate significant benefits of our approaches on lossless transfers: HTML document transfers that are often 50% faster, and total document transfer times that are often 10-50% faster. On lossy links, the performance of all schemes degrade, but the relative performance of strategies seems to remain the same. The proposed schemes were also evaluated on a client-to-router link with a smaller link delay (5 ms) and higher link capacity (256 kbps). The evaluation results are available in [17].

Future work will involve a more complete Web page workload, and a more careful study of the relationships between page structure, packet losses, and the performance of each scheme. An experimental implementation of AF-Ensemble in an ADSL or wireless environment is also planned.

Acknowledgements

The authors thank the anonymous reviewers for their comments and feedback on an earlier version of this paper. Financial support for this research was provided by TR*Labs* (Telecommunications Research Laboratories) in Saskatoon, and by NSERC research grant OGP0121969.

References

[1] *UCB/LBNL/VINT Network Simulator — ns (version 2) (See http://www-mash.cs.berkeley.edu/ns/).*

[2] H. Balakrishnan, V. Padmanabhan, and R. Katz. The effect of asymmetry on TCP performance. *ACM Journal of Mobile Networks and Applications (MONET)*, 4(3):219–241, October 1999.

[3] P. Barford and M. Crovella. A performance evaluation of Hyper-text Transfer Protocols. In *Proceedings of ACM SIGMETRICS'99*, pages 188–197, Atlanta, GA, May 1999.

[4] R. Braden. *RFC1644: T/TCP — TCP Extensions for Transactions. Functional Specification (See http://www.cis.ohio-state.edu/htbin/rfc/rfc1644.html).*

[5] L. Eggert, J. Heidemann, and J. Touch. Effects of Ensemble-TCP. *ACM Computer Communication Review*, 30(1):15–29, January 2000.

[6] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, and T. Berners-Lee. *RFC2068: Hypertext Transfer Protocol — HTTP/1.1 (See http://www.cis.ohio-state.edu/htbin/rfc/rfc2068.html).*

[7] S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking*, 1(4):397–413, August 1993.

[8] G. Hasegawa, M. Murata, and H. Miyahara. Performance evaluation of HTTP/TCP on asymmetric networks. *International Journal of Communication Systems*, 12(4):281–296, July-August 1999.

[9] J. Heidemann, K. Obraczka, and J. Touch. Modeling the performance of HTTP over several transport protocols. *ACM/IEEE Transactions on Networking*, 5(5):613–630, October 1997.

[10] J. Hoe. Improving the start-up behavior of a congestion control scheme for TCP. In *Proceedings of ACM SIGCOMM'96*, pages 270–280, Stanford, CA, August 1996.

[11] T. Lakshman, U. Madhow, and B. Suter. Window-based error recovery and flow control with a slow acknowledgment channel: A study of TCP/IP performance. In *Proceedings of IEEE INFOCOMM'97*, pages 1199–1209, Kobe, Japan, April 1997.

[12] B. Mah. An empirical model of HTTP network traffic. In *Proceedings of IEEE INFOCOMM'97*, pages 592–600, Kobe, Japan, April 1997.

[13] I. Ming-Chit, D. Jinsong, and W. Wang. Improving TCP performance over asymmetric networks. *ACM Computer Communication Review*, 30(3):45–54, July 2000.

[14] V. Padmanabhan and J. Mogul. Improving HTTP latency. In *Proceedings of 2nd International World Wide Web Conference*, Chicago, IL, October 1994.

[15] W. Stevens. *TCP/IP Illustrated, Volume 1.* Addison-Wesley, 1994.

[16] J. Touch. *RFC2140: TCP Control Block Interdependence (See http://info.broker.isi.edu/in-notes/rfc/files/rfc2140.txt).*

[17] Q. Wu. *Improving Web Data Transfer Performance on Asymmetric Networks*, May 2001.