Regular Paper

Comparative Study on the Self-Similarity of TCP Reno and TCP Vegas

Peter Ivo Racz,† Takahiro Matsuda† and Miki Yamamoto††

Self-similar traffic patterns have been observed in many measurements of Internet traffic. Self-similarity is very detrimental to the performance of packet networks and recent research has focused on understanding and reducing its amount in Internet traffic. TCP Reno has been identified as being one of the primary sources of self-similarity. We explore the potential of another version of TCP in this paper to reduce the degree of self-similarity in aggregated TCP traffic. We decompose both TCP Reno and TCP Vegas to demonstrate and explain their underlying mechanisms, and separately measure what effects congestion-avoidance and timeouts/exponential backoff mechanisms have on the self-similarity in aggregated TCP flows. We reveal how TCP Vegas reduces the degree of self-similarity and eventually completely eliminates it from aggregated TCP flows at low levels of packet loss. However, at high levels of packet loss we show that TCP Vegas is detrimental, because it increases the degree of aggregated TCP-flow self-similarity.

1. Introduction

Self-similarity is a ubiquitous property of network traffic. It represents a correlation property of traffic volume over a wide range of time scales, i.e., Long Range Dependence (LRD). Traffic of a self-similar nature is detrimental. Its effects range from high queue-buffer overflow rates to long delays and persistent periods of congestion [1]. For example, when traffic is self-similar, excessive packet loss cannot be reduced by increasing the buffer size at the bottleneck link because self-similar traffic is bursty for all time scales. Because of self-similarity's negative effects on the performance of networks, understanding its nature and reducing it has been the focus of attention in the networking community [2]. However, the sources of self-similarity must first be identified to reduce it in network traffic.

Recent research has identified various origins of self-similarity in different protocol layers. Internet applications like HTTP have exhibited self-similar traffic, due to the heavy-tailed file-size distributions of transmitted files [3]. The transport layer – such as the transport control protocol (TCP) – also actively participates in modifying the degree of self-similarity of the traffic arriving from the applications [4),5]. Recent research has shown that both aggregated and single TCP flows have significant self-similar properties [6]. The network itself also plays an important role by amplifying or reducing the degree of self-similarity. The queuing techniques used by routers and switches [2] and the applied traffic filters [7] all affect self-similarity. It has also been demonstrated that self-similar flows induce this phenomenon in other flows, which originally had no self-similar properties when sharing the same queue [8]. That is, the overall degree of self-similarity of traffic results from the interaction of all these causes.

Finding the sources of self-similarity in the TCP protocol can help to reduce its overall presence in network traffic because its TCP-related causes are present regardless of causes on other levels. However, research thus far has solely focused on the TCP Reno version of TCP. It has been shown that TCP-related self-similarity is due to TCP Reno's built-in congestion-avoidance and retransmission schemes. It may be of interest to learn how the congestion-avoidance and retransmission schemes of other versions of TCP protocols perform to reduce the TCP-related self-similarity in network traffic.

We examine the potential of another version of TCP in this paper i.e., TCP Vegas, to reduce the TCP-related causes of self-similar traffic. TCP Vegas is a transport layer protocol introduced by Brakmo, et al. [9]. TCP Vegas has more than 10 new traffic control techniques compared to TCP Reno [10]. Our interest in TCP Vegas lies in its proactive congestion avoidance technique and its capabilities for recovering from multiple packet-loss scenarios using multiple fast retransmissions [9]. Various

† Graduate School of Engineering, Osaka University
†† Department of Electrical Engineering and Computer Science, Kansai University

studies have focused on evaluating TCP Vegas's performance and fairness issues. Research has demonstrated that TCP Vegas can achieve significantly higher throughput compared to TCP Reno [9),10)]. However, TCP Vegas may fail to achieve a fair share of bandwidth among individual flows [11)].

The purpose of this paper was to evaluate the feasibility of TCP Vegas from the viewpoint of self-similarity whereas all other researchers have focused on its feasibility from the viewpoint of throughput and fairness. Because the Reno version of TCP is a popular protocol on the Internet and IP networks, we investigated and modeled what effect introducing TCP Vegas into such networks would have and studied how it would affect the self-similarity of aggregated TCP flows. We identified the underlying mechanisms responsible for both the advantages and disadvantages of introducing TCP Vegas into such networks.

The rest of the paper is organized as follows. Section 2 gives a brief overview of self-similarity and the Hurst parameter. Section 3 describes the simulation environment. Section 4 presents the simulation results. Section 5 analyses what effect the congestion-avoidance and timeout/exponential backoff techniques of TCP Reno and TCP Vegas has on the degree of self-similarity of single and aggregated flows. Section 6 summarizes our findings.

2. Definition of Self-Similarity

This section gives a short introduction and points out further references on self-similarity, the Hurst parameter, and the method of measurement that we applied.

Self-similarity is a mathematical property that have been found in the measurement of networks [12)]. Self-similar processes have various mathematical definitions that capture the various properties of self-similarity. The following definition expresses how the variations in self-similar processes decay slowly with aggregation.

Traffic is self-similar with Hurst parameter H $(0 < H < 1)$ if for all $k > 0$ and $t \geq 0$,

$$Y(t) = k^{-H} Y(kt) \qquad (1)$$

where $Y(t)$ is the traffic volume in the function of time and "=" means equality in the sense of distribution.

One of the most important properties of self-similar traffic from the networking point of view is its burstiness for all time scales. Both non-self-similar and self-similar traffic have burstiness on short time scales. Increasing the time scale decreases the burstiness of non-self-similar traffic but leaves self-similar traffic bursty even on the longest time scale. Traffic with such properties will cause persistent congestion, high packet-loss rates, and long delays as described in Section 1.

We used the Hurst parameter (H) in this paper to quantify the degree of traffic self-similarity. It can describe the traffic's degree of self-similarity as a single parameter. H has values of $1 > H > 0$.

Throughout this paper, we focus on if H falls in the range between $1 > H > 0.5$. Traffic with the Hurst parameter within this range is self-similar and long-range dependent causing congestion and high packet loss. The closer H is to one, the stronger the self-similarity and the more negative the properties of the traffic.

When $H = 0.5$, the traffic is non-self-similar and does not have the negative properties associated with self-similarity.

When $0.5 > H > 0$, the traffic is self-similar, but short-range dependent. Short-range dependent self-similar traffic does not have the negative properties associated with long-range dependent self-similar traffic.

Various estimators can be used to provide estimates of self-similarity and the Hurst parameter [13),14)]. Throughout this paper, we have used Veitch and Abry's wavelet-based estimator, which is freely available for academic purposes [15)]. The variance-time method [4),13)] is another popular means of measurement that is prone to the periodicity of the TCP protocol. Our goal with using the wavelet estimator was to eliminate the effect TCP's periodicity had on the value of the measured Hurst parameter.

3. Simulation Environment

This section describes the simulation environment used throughout this paper. We used the ns2 network simulator [16)] to create an aggregated TCP flow consisting of 4–150 single TCP flows that crossed the bottleneck topology in **Fig. 1**. The TCP receivers are attached directly to the bottleneck, and the senders are connected via a feeder link. Although these feeder links impose no bandwidth restrictions, they have different propagation delays making the end-to-end delays uniformly distributed between 100–150 ms.

The buffer size, transmission speed, and link delay of the bottleneck link were set according to **Table 1**; however, our simulations revealed that the findings in this paper are valid for a wide range of these parameters. We have already discussed the effects these parameters have on the aggregated TCP flow in detail [17].

We gradually increased the ratio of TCP Vegas flows in the aggregated flow between 0−90% in consecutive experiments to examine what effect it had on the aggregated flow.

We employed greedy data sources on both the TCP Vegas and TCP Reno flows because Vegas flows carrying data from greedy sources extensively operate in their inventive congestion avoidance mode. Also, the effect of greedy sources on the self-similarity of TCP flows is very limited, because these sources constantly generate packets that must wait for TCP to open its usable window before transmission.

All simulations were run for a duration of 2200 s, where we discarded the first 400 s of data to avoid transients. The flows were commenced randomly during the first 30 s of each simulation. We measured the Hurst parameter for each single and aggregated TCP flow along with packet loss at the bottleneck link for all simulations.

There are currently networks that operate on very low packet-loss rates, i.e., $< 1\%$, due to the ever-increasing performance of networking equipment. However, live measurements taken on the Internet have also revealed packet-loss levels far above 1% [18),19)]. The scope of the simulations discussed in this paper was to examine the self-similar behavior of TCP traffic when TCP flows compete for network resources. That is why we set the simulation parameters so that the TCP flows yielded a packet loss of $1\% < p < 20\%$.

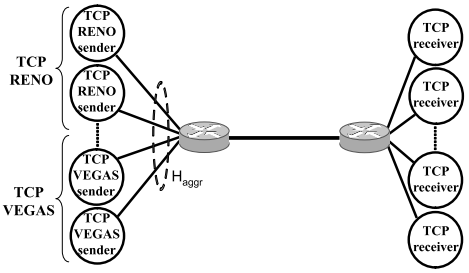

Fig. 1 TCP Reno and TCP Vegas flows share bottleneck link.

Table 1 Parameters and assumptions for simulation.

Number of flows	4–150
Ratio of TCP Vegas flows (%)	0–90
Link delay (ms)	100–150
Link speed (Mbps)	3
Buffer size (packets)	35
Packet size (bytes)	1040
Queuing algorithm	FIFO

4. Evaluation of Hurst Parameter for Aggregated and Single TCP Flows

This section presents the results of our simulations and describes our observations, while Section 5 discusses our analyses and explains the underlying mechanisms.

As a result of aggregation, the Hurst parameter for the aggregated TCP flow has lower values than its composite single TCP flows according to **Fig. 2**. While the aggregated TCP flow is mostly short-range dependent ($H < 0.5$), the single TCP flows are mostly long-range ($H > 0.5$) dependent. Short-range dependent traffic is amenable to statistical multiplexing and does not have the detrimental effects of long-range dependent traffic on the performance of networks. Aggregated flow is mostly short-range dependent, because it fully utilizes the capacity of the bottleneck link, which limits the variance of the arrival process. At high packet-loss levels, the aggregated flow reveals a trend to increase, which confirms the findings by Ref. 5) and Ref. 18). At low packet-loss levels, the Hurst parameter demonstrates increased values, which even go beyond $H = 0.5$. This phenomenon will be discussed in more detail in Section 5.1.

Figure 3 plots the Hurst-parameter curves for aggregated TCP flows, which consist of 20%, 40%, and 70% TCP Vegas flows while the remainder of the flows are TCP Reno. **Figure 4** shows the average Hurst parameter for compos-

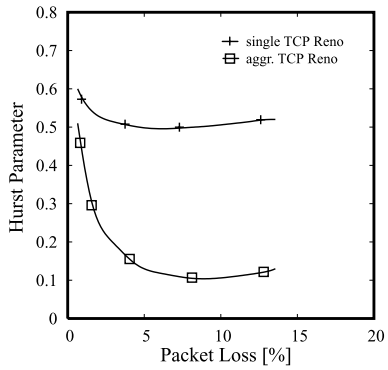

Fig. 2 Hurst parameter for aggregated TCP Reno flow and its composite single flows.

ite single TCP Reno and TCP Vegas flows. The following facts can be observed from Figs. 3 and 4:

(A) For $p < 6\%$, the Hurst-parameter curve for the aggregated flow decreases as the TCP Vegas ratio of the aggregated flow increases, where p stands for packet loss rate throughout this paper. When 20% of the flows were TCP Vegas, no effect was yet noticed by comparing Fig. 3 with Fig. 2. However, at 40% TCP Vegas, the elevated part of the Hurst-parameter curve significantly decreases. At 70%, the TCP Vegas flows completely eliminate the increased levels of the Hurst parameter.

(B) For $p > 6\%$, the Hurst-parameter curve for the aggregated flow slightly increases as the TCP Vegas ratio of the aggregated flow increases according to Fig. 3.

(C) Figure 4 shows that the Hurst parameter for single TCP Reno flows are higher than that for corresponding single TCP Vegas flows for $p < 10\%$. Increasing the TCP Vegas ratio of the aggregated flow decreases the value of the Hurst parameter for both single TCP Reno and TCP Vegas flows for $p < 7\%$.

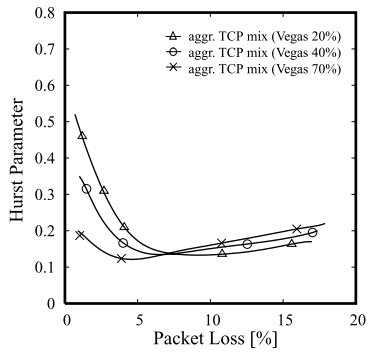

Fig. 3 Hurst parameter for aggregated TCP flow consisting of 20%, 40%, and 70% TCP Vegas flows while remainder is TCP Reno.

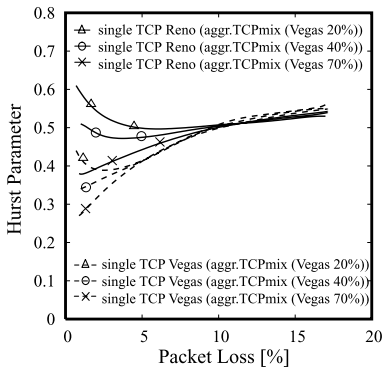

Fig. 4 Hurst parameter for single TCP Reno and single TCP Vegas flows that make up aggregated flows in Fig. 3.

5. In-depth Considerations into Effect of TCP Congestion Control

We focus on traffic-control techniques for single TCP Reno and TCP Vegas flows in this section to discuss how TCP Vegas decreases and increases the self-similarity of an aggregated TCP flow (i.e., facts A and B).

Guo, et al. [4] have shown through analyses that the self-similarity of a single TCP flow is a result of the exponential-backoff algorithm. Figueiredo, et al. [6] added that the congestion avoidance technique also contributes to self-similarity in aggregated TCP flows.

To understand the roles of congestion-avoidance and exponential backoff/timeout techniques in the self-similarity of aggregated TCP traffic, we separated their effects on single TCP Reno and TCP Vegas flows. We determined their level of influence and the packet-loss range where these transport-control techniques were dominant.

To eliminate the interaction between single flows in a bottleneck link, we ran a single Reno/Vegas flow across a bottleneck link in a separate simulation (**Fig. 5**). Packet loss was randomly generated at the aggregating router. All other parameters remained unchanged from the simulation described in Fig. 1 and Table 1.

To measure what effect the timeout/exponential backoff technique had, we derived modified TCP flows where the effects of congestion-avoidance and slow-start techniques were eliminated. After identifying the timeouts in the congestion window trace of the single flows, we modified the throughput trace so that the derived TCP flow generated packets with a constant bit rate between two adjacent timeouts. The constant bitrate was calculated so that the original and the modified flow would transfer the same amount of traffic during the simulation period. **Figure 6** plots the congestion window for the reference TCP flow and the transmission rate for the modified TCP flow.

To measure what effects congestion avoidance

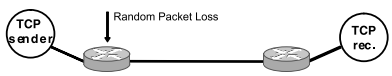

Fig. 5 Single TCP Reno/Vegas flow crosses bottleneck link with random packet loss.

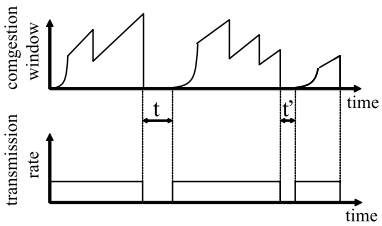

Fig. 6 Congestion window for reference TCP Reno flow (above) and transmission rate of modified TCP Reno flow (below) with effects of congestion-avoidance technique eliminated.

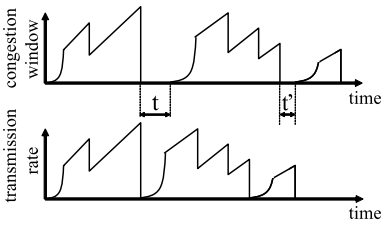

Fig. 7 Congestion window for reference TCP Reno flow (above) and transmission rate of modified TCP Reno flow (below) with effects of timeout and exponential backoff techniques eliminated.

had, we derived another type of TCP flow that had the effects of timeouts and the exponential backoff technique eliminated. We modified the original flow so that the modified TCP flow would enter slow-start immediately after encountering a timeout (see **Fig. 7**). We assumed that the effects of slow-start, fast retransmission, and fast recovery would be marginal compared to the effect of congestion avoidance.

The "t" and "t'" in Figs. 6 and 7 mark two consecutive timeouts in the original TCP flow. Timeouts were identified as the time between the departure of the packet transmitted from the TCP sender, which its retransmission timer eventually expired, and the departure of the first packet in the slow-start phase after the timer had expired. Any packets retransmitted during this time interval were ignored for the sake of simplicity.

Figures 8 and **9** plot the Hurst-parameter curves for the original TCP Reno and TCP Vegas flows, and also the two modified flows reflecting the Hurst parameters of their congestion-avoidance and exponential backoff techniques. We can see that both the congestion-avoidance and exponential backoff/timeout techniques contribute to the self-similarity of both types of TCP flows. When the Vegas flows decrease the degree of self-similarity in the aggregated flows (i.e., fact A),

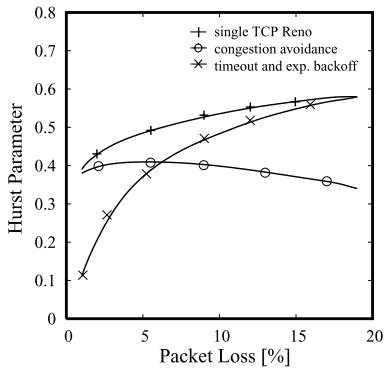

Fig. 8 Contribution of congestion-avoidance and exponential backoff techniques to self-similarity of single TCP Reno flow.

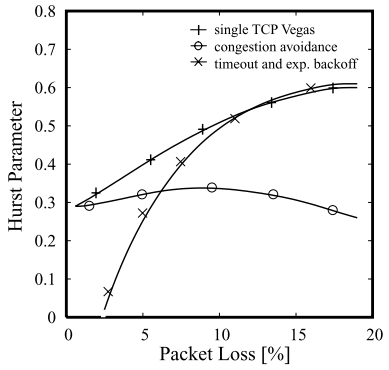

Fig. 9 Contribution of congestion-avoidance and exponential backoff techniques to self-similarity of single TCP Vegas flow.

the influence of the congestion-avoidance technique is dominant. Whereas in the packet-loss range where the Vegas flows increase the degree of self-similarity in aggregated flows (i.e., fact B), the timeout and exponential backoff techniques are dominant.

5.1 Effect of Congestion Avoidance

This section focuses on how TCP Reno's and TCP Vegas's congestion-avoidance technique affects the Hurst parameter of the aggregated TCP flow at the low packet-loss level (i.e., fact A).

The single TCP Reno flows in Fig. 4 show an increased value for $p < 6\%$ (Vegas ratio $< 70\%$) compared to the single TCP Reno flow in Fig. 8. This is due to the difference in the packet-loss mechanism. Packets from the Reno flow in Fig. 8 were dropped randomly while the single TCP flows in Fig. 4 suffered packet loss due to overflow in the FIFO queue at the bottleneck link as seen in Fig. 1. **Figure 10** plots the time distribution between two con-

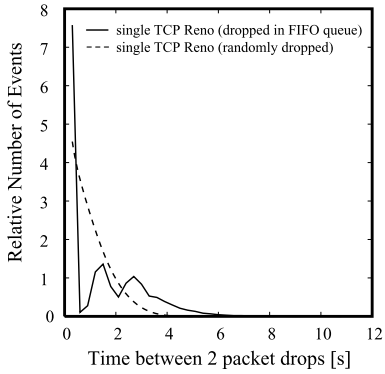

Fig. 10 Distribution of time between two consecutive packet drops that occur in single TCP flows for different packet-dropping schemes.

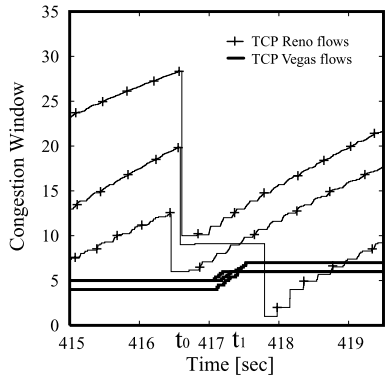

Fig. 11 Congestion window for TCP Reno and Vegas flows crossing bottleneck in Fig. 1.

secutive packet losses for the single TCP Reno flows in Fig. 4 (Vegas ratio 20%) and Fig. 8 for $p = 2.5\%$ packet loss. The distribution curve for the single TCP Reno flow in Fig. 10, which was part of the aggregated flow, shows frequent short ($dt < 0.2\,s$) and long ($dt > 2\,s$) time intervals between consecutive lost packets compared to the single TCP flow, whose packets were dropped randomly. This indicates multiple packet losses followed by longer packet-loss-free periods, which are typical of synchronized TCP Reno flows.

Synchronization is a phenomenon where individual TCP flows encounter packet loss, and as a result they decrease their throughput at the same time. Synchronization is a result of TCP Reno's congestion-avoidance technique [20]. **Figure 11** shows a trace of the synchronized congestion windows of the single TCP Reno flows captured during the simulation described by Fig. 1. At time t_0 the TCP Reno flows over flooded the queue, and cut their congestion windows causing a sudden drop in the volume of aggregated traffic. This synchronization in TCP flows is the cause of the elevated Hurst-parameter curve for the single TCP Reno flows in Fig. 4 and also the aggregated TCP flow in Fig. 3 for $p < 6\%$. This is because when long-range-dependent single Reno flows increase and cut their throughput at the same time they become synchronized, and the aggregated Reno flow will also demonstrate long-range dependence.

We will next discuss how adding TCP Vegas flows to the aggregated flow decreases its Hurst parameter. Comparing the Hurst parameters of single Vegas flows in Figs. 4 and 9 reveals that the Vegas flows remain short-range-dependent ($H < 0.5$) for $p < 6\%$ even when they cross the bottleneck link with Reno flows. This is because the TCP Vegas flows do not synchronize with the TCP Reno flows. Due to their intuitive congestion-avoidance technique, they do not have to keep increasing their throughput to detect congestion in the network. Instead TCP Vegas measures the change in the round-trip-time to detect network congestion and linearly increases or decreases its throughput accordingly [9]. The synchronized packet loss of the Reno flows does not affect the TCP Vegas flows, because the packet bursts of TCP Vegas do not arrive at the queue synchronized with the packet bursts of TCP Reno flows. Also, the TCP Vegas flow's packet bursts are smaller than the bursts of the TCP Reno flows just before the queue is flooded. The smaller bursts contribute to reducing the probability of packet loss. Figure 11 also shows that after the TCP Reno flows cut their throughput at t_0, the TCP Vegas flows detect the new network resources that have become available and increase their throughput at time t_1, thus actively preventing synchronization from occurring and reducing the synchronization-related fluctuations in aggregated traffic. Therefore, increasing the TCP Vegas ratio of the aggregated TCP flow reduces the increased Hurst-parameter value of the single TCP flows in Fig. 4 for $p < 6\%$. It also reduces the degree and eventually eliminates the aggregated TCP flow's self-similarity for $p < 6\%$ (i.e., fact A).

5.2 Effect of Timeout and Exponential Backoff

This section discusses how the timeout and exponential backoff mechanisms of TCP Vegas are responsible for the elevated level of the Hurst parameter of the aggregated TCP flow

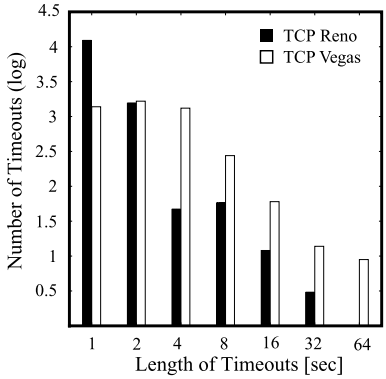

Fig. 12 Distribution of OFF periods in TCP Reno and TCP Vegas flows at $p = 12\%$ packet loss.

when $p > 6\%$ in Fig. 3 (i.e., fact B).

As discussed in Section 5, the timeouts and the exponential backoff mechanisms for both TCP Reno and TCP Vegas are dominant when $p > 6\%$, according to Figs. 8 and 9. Under these conditions, the TCP flow can be modeled with the ON/OFF process model used previously in Section 5 and described in Fig. 6. The OFF times correspond to the timeout periods, and the ON times correspond to those time intervals where the TCP is in congestion-avoidance mode [18]. The TCP in this model transmits with a constant average bit rate during ON times, while the TCP flow does not transmit during OFF times.

Figure 12 plots the distribution in OFF-period length in TCP Vegas and TCP Reno flows at $p = 12\%$ packet-loss rate, where the ON/OFF model can be applied according to Figs. 8 and 9. We can see from Fig. 12 that TCP Vegas has about 25% fewer OFF periods, i.e., $t_{OFF} < 1\,\mathrm{s}$. This is due to the ability of TCP Vegas to recover from multiple packet-loss scenarios with greater efficiency [9]. TCP Vegas, however, has more frequent, longer lasting ($t_{OFF} > 2\,\mathrm{s}$) timeouts than TCP Reno according to Fig. 12. Long-tailed distribution and self-similarity are strongly related. Aggregated ON/OFF processes are where the ON or OFF periods are heavy-tailed and form aggregated flows with long-range-dependent properties [18]. That is, when the TCP Vegas ratio of the aggregated flow is increased, the more frequent, longer lasting timeouts and exponential backoffs of TCP Vegas increase the Hurst parameter for the aggregated TCP flow (i.e., fact B) as shown in Fig. 3.

The source of TCP Vegas's increased number

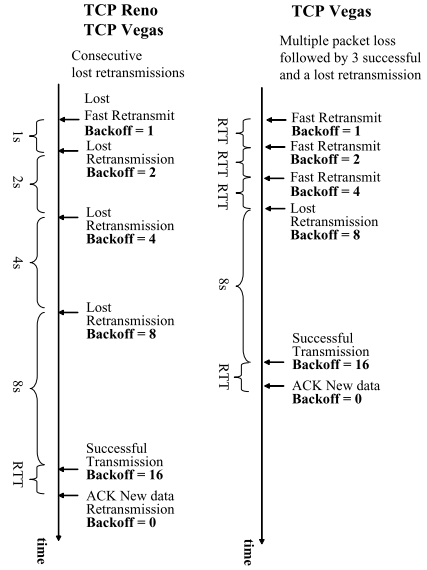

Fig. 13 Left: Exponential backoff mechanisms in TCP Reno and TCP Vegas. Right: Exponential backoff mechanism only in TCP Vegas.

of exponential backoffs is that it activates its exponential-backoff mechanism more frequently than TCP Reno due to its new traffic-control technique.

When consecutive retransmitted packets are lost, both TCP Reno and TCP Vegas function similarly. They activate their exponential-backoff traffic-control techniques. The left side of **Fig. 13** illustrates a scenario that both TCP Reno and TCP Vegas could encounter. When the RTO of a retransmitted packet expires, both TCP Reno and TCP Vegas retransmit the lost packet and exponentially increase the length of the retransmitted packet's RTO. The RTO of the retransmitted packet and every following lost packet are exponentially increased until the segment is successfully transmitted. Consecutive losses of retransmitted packets result in timeouts with exponentially increasing lengths of up to 64 s.

The exponential-backoff mechanism of TCP Vegas is not only activated when consecutive retransmitted packets are lost but also in multiple packet-loss scenarios. As previously mentioned, TCP Vegas is capable of recovery without timing out when multiple packets are lost in a congestion window of data using multiple fast retransmissions. The right side of Fig. 13 illustrates a scenario that only TCP Vegas can encounter. Each retransmission of a lost packet further increases the value of the exponential-

backoff mechanism's backoff parameter. When Vegas succeeds in recovering all the lost packets and acknowledges new data, no timeouts occur. If Vegas manages to recover some of these, but not all lost packets in a round, Vegas times out. The length of this timeout is exponentially proportional to the number of retransmitted segments since TCP Vegas has entered the fast retransmission stage. The exponential-backoff mechanism is only reset when new data are acknowledged.

Contrary to TCP Vegas, TCP Reno has no traffic-control mechanism to handle multiple packet losses. TCP Reno is mostly capable of recovering the loss of single packets [21] and in multiple-loss scenarios, like the one illustrated on the right of Fig. 13, TCP Reno times out.

6. Conclusions

We investigated the benefits and drawbacks of introducing TCP Vegas into contemporary IP networks in this paper. We demonstrated that adding TCP Vegas flows to aggregated TCP flows can both increase and decrease the degree of self-similarity in aggregated TCP traffic.

To find the underlying reasons, we decomposed individual TCP Reno and TCP Vegas flows and discovered that TCP Vegas's intuitive congestion avoidance is responsible for its positive effects while its timeout/exponential backoff techniques are responsible for its negative effects.

The positive effects appear at low levels of packet loss, where TCP Vegas decreases the Hurst parameter of aggregated TCP flows, by crossing bottleneck links unaffected, while synchronized long-range-dependent TCP Reno flows over flood the queue. The TCP Vegas flows also decrease the Hurst parameter of the aggregated traffic by immediately utilizing the bandwidth left behind by the TCP Reno flows when they simultaneously halve their throughput or enter timeout.

At higher packet-loss rates, TCP Vegas increases the degree of self-similarity in aggregated TCP Reno flows. When TCP Vegas is recovering from multiple packet-loss scenarios, its exponential-backoff mechanism introduces less frequent but longer timeouts than TCP Reno. These timeouts are responsible for the increased Hurst parameters of both single TCP Vegas flows and aggregated TCP flows containing a TCP Reno and Vegas mix.

In future research, we intend to improve the retransmission-timeout mechanism for TCP Vegas by modifying the TCP Vegas implementation of the ns2 [16] network simulator.

Unfortunately, TCP Vegas is not always a practical transport protocol because it may fail to achieve a fair share of bandwidth when Vegas and Reno flows coexist. However, practical transport protocols, e.g., TCP Veno [22], which is a combination of TCP Reno and Vegas mainly used in wireless networks, have been proposed. Since these protocols are partially based on TCP Vegas, we are confident that our findings can be applied to such protocols as well.

Acknowledgments The authors would like to thank Biplap Sikdar Ph.D of the Rensselaer Polytechnic Institute, Troy, NY, USA for his valuable comments.

References

1) Paxson, V. and Floyd, S.: Wide Area Traffic: The Failure of Poisson Modeling, *IEEE/ACM Trans. on Networking*, pp.71–86 (1997).
2) Sikdar, B., Chandrayana, K., Vastola, K.S. and Kalyanaraman, S.: On Reducing the Degree of Second-Order Scaling in Network Traffic, *Proc. IEEE GLOBECOM*, pp.2594–2598 (2002).
3) Crovella, M. and Bestavros, A.: Self-similarity in World Wide Web traffic: Evidence and possible causes, *IEEE/ACM Trans. Networking*, Vol.5, No.6, pp.835–846 (1997).
4) Guo, L., Crovella, M. and Matta, I.: How does TCP generate Pseudo-self-similarity?, *Proc. Int. Workshop on Modeling, Analysis and Simulation of Computer and Telecommunications Systems* (2001).
5) Veres, A. and Boda, M.: The chaotic nature of TCP congestion control, *Proc. IEEE INFOCOM* (2000).
6) Figueiredo, D.R., Liu, B., Misra, V. and Towsley, D.: On the Autocorrelation Structure of TCP Traffic, *Tech. Rep., Computer Science Technical Report*, 00-55, University of Massachusetts (2000).
7) Procissi, G., Gerla, M., Kim, J., Lee, S. and Sanadidi, M.: On Long Range Dependence and Token Buckets, *SPECTS* (2001).
8) Veres, A., Kenesi, Z., Molnar S. and Vattay, G.: On the propagation of long-range dependence in the Internet, *Proc. ACM SIGCOMM*, pp.243–254 (2000).
9) Brakmo, L.S., O'Malley, S.W. and Peterson, L.: TCP Vegas: New Techniques for Congestion Detection and Avoidance, *Proc. ACM SIG-*

COMM, pp.24–35 (1994).
10) Hengartner, U., Bollinger, J. and Gross, T.: TCP Vegas Revisited, *Proc. IEEE INFOCOM* (2000).
11) Mo, J., La, R.J., Anantharam, V. and Walrand, J.: Analysis and Comparison of TCP Reno and Vegas, *Proc. IEEE INFOCOM* (1999).
12) Willinger, W., Taqqu, M., Sherman, R. and Wilson, D.: Self-similarity through high-variability: Statistical analysis of Ethernet LAN traffic at the source level, *Proc.ACM SIGCOMM*, pp.100–113 (1995).
13) Karagiannis, T., Faloutsos, M. and Molle, M.: A User-Friendly Self-Similarity Analysis Tool, *ACM SIGCOMM Comp. Comm. Rev.*, Vol.33, pp.81–93 (2003).
14) Abry, P. and Veitch, D.: Wavelet Analysis of Long-Range Dependence Traffic, *IEEE Trans. Inf. Theory*, Vol.4, No.1, pp.2–15 (1998).
15) Veitch, D. and Abry, P.: A Wavelet-based Joint Estimator for the Parameters of Long-Range Dependence, *IEEE Trans. Inf.* (1999).
16) ns2 simulator. http://www.isi.edu/nsnam/ns
17) Racz, P.I., Matsuda, T. and Yamamoto, M.: Contribution of the Application, Transport, and Network Layers to the Self-Similarity of Internet Traffic, *IPSJ Journal*, Vol.46, No.12, pp.3109–3122 (2005)
18) Sikdar, B. and Vastola, K.S.: On the Contribution of TCP to the Self-Similarity of Network Traffic, *Lecture Notes in computer Science* (*Proc. IWDC*), Vol.2170, pp.596–613 (2001).
19) Padhye, J., Firoiu, V., Towsley, D. and Kurose, J.: Modeling TCP Throughput: A Simple Model and its Empirical Validation, *Proc. ACM SIGCOMM*, pp.303–314 (1988).
20) Floyd, S. and Jacobson, V.: Random Early Detection Gateways for Congestion Avoidance, *IEEE/ACM Trans. on Networking*, Vol.1, No.4, pp.397–413 (1993).
21) Fall, K. and Floyd, S.: Simulation-based Comparisons of Tahoe, Reno and SACK TCP, *Comp. Commun. Rev.* (1996).
22) Fu, C.P. and Liew, S.C.: TCP Veno: TCP Enhancement for Transmission Over Wireless Access Networks, *IEEE J. on Select. Areas in Comm.*, Vol.21, No.2 (2003).

(Received March 30, 2006)
(Accepted September 14, 2006)
(Online version of this article can be found in the IPSJ Digital Courier, Vol.2, pp.783–791.)

Peter Ivo Racz received his B.E. and M.E. in electrical and communication engineering from the Technical University of Budapest (TUB) in Hungary in 1996 and 1999. He joined Cisco Systems Japan K.K. in 2005, where he is presently a systems engineer. He is concurrently working toward his Ph.D. at the Department of Communications Engineering of Osaka University. His research interests include QoS and analysis of Internet traffic.

Takahiro Matsuda received his B.E. with honors, M.E., and Ph.D. in communications engineering from Osaka University in 1996, 1997, and 1999. He joined the Department of Communications Engineering at the Graduate School of Engineering of Osaka University in 1999. He has been a lecturer in the Division of Electrical, Electronic, and Information Engineering at the Graduate School of Engineering of Osaka University since May 2005. His research interests include performance analysis and the design of communication networks and wireless communications. He is a member of IEICE and IEEE.

Miki Yamamoto received his B.E., M.E., and Ph.D. in communications engineering from Osaka University in 1983, 1985, and 1988. He joined the Department of Communications Engineering at Osaka University in 1988. He moved to the Department of Electrical Engineering and Computer Science of Kansai University in 2005, where he is a professor. He visited the University of Massachusetts at Amherst in 1995 and 1996 as a visiting professor. His research interests include multicast communications, high-speed networks, wireless networks, and the evaluation of performance of these systems. He is a member of IEEE, ACM, and IEICE.