Assessing Interactions among Legacy and High-Speed TCPs

Hideyuki Shimonishi, M. Y. Sanadidi, and Tutomu Murase

***Abstract*—It has been recognized that TCP throughput deteriorates in high-speed networks with large bandwidth-delay product, and new congestion control algorithms have been proposed to address such deterioration. Assuming that the proposed protocols would be in general used in the Internet, it is imperative to study the interaction among flows of different protocols, in addition to the interaction among flows using the same protocol. In this paper, we discuss a method of assessing the interaction among different protocols. By applying the same experiment setup, including network configuration, flow parameters, and workload of each flow, to multiple experiment runs for the different protocols, we can assess flow-by-flow and file-by-file behavior of different protocols. We provide some numerical results in networks with multiple bottlenecks, a large number of short-lived and long-lived flows, and variety of RTTs. The results show that, while all high-speed TCPs are effective in improving efficiency, they exhibit different detailed characteristics. When compared to previous schemes, CUBIC, Compound-TCP, and TCP-AdaptiveReno improve, to varying degrees, RTT-fairness and Reno-friendliness. Our results also show that, due to slow-start behavior of short flows, delay-based control is not very effective in improving RTT-fairness as expected. Based on the insight above, we have modified the delay-based control part of TCP-AdaptiveReno and the results show its highly balanced efficiency, friendliness, and fairness.**

I. Introduction

TCP has been designed for networks where a packet loss is recognized as a congestion signal and for link speeds lower than what is possible today. It is well known that TCP performance suffers in fast and long-distance networks with non-negligible random losses. To improve the performance of TCP in such networks, a number of new TCP variants, including High Speed TCP [1], Scalable TCP [2], FAST [3], BIC [4], CUBIC [5], Hamilton-TCP [6], Compound-TCP [19], TCP-Westwood [7,8], and TCP-AdaptiveReno [9], to mention a few, have been proposed.

Assuming that these new protocols are likely to be deployed on the Internet, rather than used in private networks, it is imperative to study the interactions among flows of different protocols, in addition to the interactions of flows using the same protocol. TCP-Reno has been successful for more than a decade and it is still widely used. Therefore, when new protocols coexist with Reno, modest negative impacts on TCP-Reno flows performance may be acceptable, but severe damage to TCP-Reno flows would not be tolerated.

A framework for evaluating congestion control algorithms is presented in [10]. Also, a proposal for a standard benchmark suite is given in [11]. There are also a number of comparative evaluations of the high-speed protocols [12-15,21]. In these papers, the effects of RTT diversity, coexisting short-lived and long-lived flows, and so on, have been studied. However, there are few papers that extensively study the behavior of high-speed protocols in networks with multiple bottlenecks, including an evaluation of the coexistence of high speed protocols with legacy protocols. Over paths with multiple bottleneck links, fairness among flows traversing such paths needs to be carefully evaluated. For example, fairness of flows having diverse RTTs, fairness of flows having different hop counts, or friendliness of different protocols under such conditions needs to be investigated. In such a complex network environment, to make the evaluations trustworthy and the results for different protocols comparable, the conditions of each experiment must be recreated with minimal change for the various protocols under study.

In this paper, we discuss a method for assessing interactions among different transport protocols. Three or more sets of identical experiments except for the congestion control algorithm used by each flow are run. Across these runs, network topology, flows parameters, and workload parameters are kept identical. To this end, our simulations rely on a pre-generated set of network configurations, flows, and a workload to be transferred using the TCP protocols. The pre-generated environment is then used in our three set of simulation runs: (1) all flows use TCP-Reno, say year 2005 case, (2) half of the flows use a high-speed protocol and the remaining half use TCP-Reno, say year 2007 case, and (3) all flows use high-speed protocol, say year 2010 case.

In this paper, we analyze statistic behavior of large number of flows coexisting in complex network topologies. We have executed the proposed method using the NS2 simulator and compared various high-speed protocols, including loss-based, delay-based, and combined loss-based and delay-based methods. Regarding delay-based control, we confirmed that its RTT-fairness is not as good as expected. Due to slow-start behavior of short-lived flows, RTT often jumps up suddenly and cause a packet loss, which prevents delay-based control from staying in steady state equilibrium. Based on this observation, we modified the delay-based control part of TCP-AdaptiveReno to improve its RTT-fairness during transient state, as well as in equilibrium state.

In the sequel, we provide some numerical results for networks with multiple bottlenecks, a large number of short-lived and long-lived flows, and RTT diversity, and discuss the different behavior of a set of high-speed protocols. We especially focus on friendliness to Reno, RTT-fairness, and efficiency in multi-hop environment.

II. TCP-ADAPTIVERENO

TCP-AdaptiveReno, or TCP-AReno, has been proposed for higher efficiency in fast long-distance network while still maintaining friendliness to Reno. TCP-AReno is an extension of TCP-Westwood [8]. During congestion avoidance, congestion window size W is adaptively increased based on delay measurement. In addition, TCP-AReno also maintains a window size W^{RENO} that is increased by 1MSS per RTT like Reno. When W becomes smaller than W^{RENO}, W is increased to W^{RENO}. The following model such window size update:

$$W += \left(\alpha \frac{B}{R} RTTe^{c} - \beta Wc \right) / W$$

$$W^{RENO} += 1/W$$

$$W = \max(W, W^{RENO})$$

where c is a delay-based congestion estimation, and $c = (RTT\text{-}RTT_{MIN}) / (RTT_{CONG} - RTT_{MIN})$. RTT_{CONG} and RTT_{MIN} are the RTT value when a packet loss is expected and minimum RTT is observed. B and R are the estimated bottleneck link capacity, and the sending rate ($=W/RTT$), respectively. α and β are control parameter for window increase and decrease, respectively. The window size update functions above implement a modified delay-based control method with the following rational:

1) RTT-fairness in delay-based control is ensured when the control is in equilibrium. However, the equilibrium is easily broken by slow start behavior of other flows. Thus, to improve RTT-fairness during transient state, RTT is multiplied on both first and second term, implicitly and explicitly, compared to regular delay-based control. With this change, the congestion window increase rate is not proportional to the number of round trip times but proportional to time elapsed.

2) To maintain friendliness to Reno, inflated congestion window size has to be quickly reduced down to a Reno compatible size whenever RTT increases. To ensure the behavior, we put e^{c} on the first term and Wc on the second term. Thus, as soon as the path is being congested and queues are building up, W is quickly decreased down to W^{RENO}.

3) B/R is multiplied on the first term to improve scalability to very high-speed networks. If current sending rate is less than the estimated bottleneck link capacity, congestion window is increased faster.

Note that in an equilibrium state, when $\alpha (B/W)RTTe^{c} = \beta Wc$, we have:

$$R = \sqrt{B \frac{\alpha}{\beta} \frac{e^{c}}{c}}$$

Thus, we can confirm that steady state throughput under delay-based control is equal for all flows regardless of RTT.

Upon a packet loss, we tune congestion reduction as in TCPW-BBE and reset W^{RENO}, as follows:

$$W = \frac{1}{1+c} W, \quad W^{RENO} = \frac{1}{1+c} W^{RENO}$$

This means that congestion window size is halved like Reno when packet loss happens during congestion, i.e. $c=1$. On the other hand, if packet loss happens while the congestion estimate is low, the reduction is mitigated to improve efficiency in networks with non-negligible random losses.

III. COMPARATIVE EVALUATION METHOD

A. Configuration Generation

To fairly evaluate TCP protocols, we repeat multiple sets of experiments with the same configuration for different protocols. We rely on an experiment scenario generator, consisting of a topology generator, a flows generator, and a workload generator, which are implemented in a set of *tcl* scripts for NS2 simulator. Below we briefly describe the three generators.

- Topology generator

As shown in Fig. 1, the topology generator builds a file describing a set of links and nodes with their attributes such as capacity and propagation delay. Some links may be configured as wireless links with random packet losses. The topology may be a simple dumbbell topology, but more generally, it will include random network, tree, parking-lot, transit-stub, and artificially generated Internet routers topologies.

- Flow generator

This module generates a set of flows. Flows are created from randomly chosen source to randomly chosen destination unless they are disconnected. In a client-server type application, a small number of intermediate nodes are selected to connect the server and a large number of edge nodes are selected to connect the client hosts. In a peer-to-peer application, source and destination nodes are connected to any of the edge nodes.

The flows may be divided to run under two or more protocols. Flows in the same group use the same congestion control algorithm, but different groups may use different algorithms.

- Workload generator

The workload generator generates a set of data chunks characterized by file size and its initiation time. These files are pre-generated for each flow and the same set of files is supplied to each run to ensure repeated workload among different runs.

To evaluate the important effects of short-lived traffic, the workload may be set up as an appropriate mix of short-lived and long-lived flows. For short-lived flows, Internet file size distribution studied in [17], or Pareto distribution may be used. For long-lived flows, infinite file sizes may be used, but rather large (media) files may also be used because a large portion of the Internet traffic is occupied by large P2P file sharing. In the

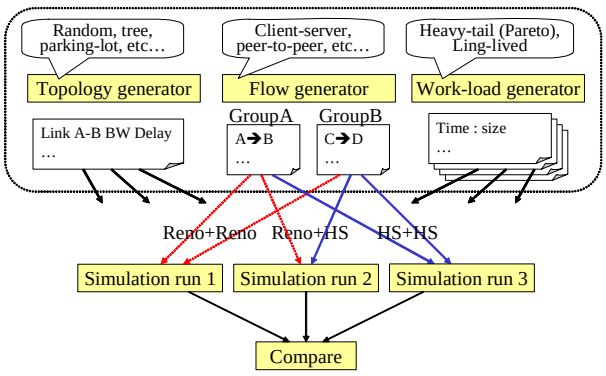

Fig. 1: Comparative evaluation of high-speed

later case, a time varying number of long-lived flows may be specified to create changes in the network load, and thus agility and stability of congestion control algorithms may be tested.

B. Experiments

The generated experiment scenario is exercised in three or more sets of runs applying different protocol to each flow group. Thus, we can compare the behavior of the same flow when it uses Reno and when it uses a high-speed protocol, and assess in what condition performance of a flow is improved by the high-speed protocols. We can also compare the behavior of a Reno flow when it coexists with other Reno flows and when it coexists with a high-speed flow. We assess which high-speed protocol is friendly to such Reno flows, and in what condition a Reno flow is degraded.

IV. Numerical Results

A. Network Model and Traffic Model

As shown in Fig. 2, we used the parking-lot topology as a good representative of the case of multiple bottleneck links. There are 5 routers and 4 uni-directional backbone links whose capacity is 1Gbps. Capacity of access links between routers and terminals is also 1Gbps. Round trip propagation delay (RTD) of each link is exponentially distributed and the average of RTDs is 15msec. The average flow RTD is around 130msec and about 60% of flows have RTD more than 100msec, a reasonable number since the measurement results in [18] pointed out that 40% of actual Internet flows have RTT more than 100msec. Since high-speed routers tend to have smaller buffer sizes than the bandwidth delay product, the buffer size is set at 2MB, which corresponds to 16% of bandwidth delay product for 100msec RTD.

We generated a large number of short-lived and long-lived flows. The source and destination terminals are connected to randomly selected nodes. File size distribution of short-lived flows is Pareto with 1MB average and its inter-arrival time is exponentially distributed with 1sec average. The number of short-lived flows is 100; thus there is 800Mbps traffic load in total over the 8 backbone links. For long-lived flows, the file size is fixed at 4.7GB, which corresponds to 1 video DVD, and its inter-arrival time is 2 min.

We compared RENO, High-speed TCP (HSTCP), Scalable TCP, BIC, CUBIC, Hamilton-TCP, Compound-TCP, TCP-Vegas and TCP-AdaptiveReno. We used a patch that provides Linux TCP congestion control algorithms on NS2 simulator [20]. Thus the results loosely follow their Linux TCP implementation, rather than the original NS2 codes of each protocol. We have to note that, it is reported that CUBIC behavior using this patch is somewhat different from its Linux behavior, and indeed we got some unexpectedly poor performance. For this reason, we removed CUBIC results from this paper until the patch behavior in the future coincides with the Linux implementation for CUBIC.

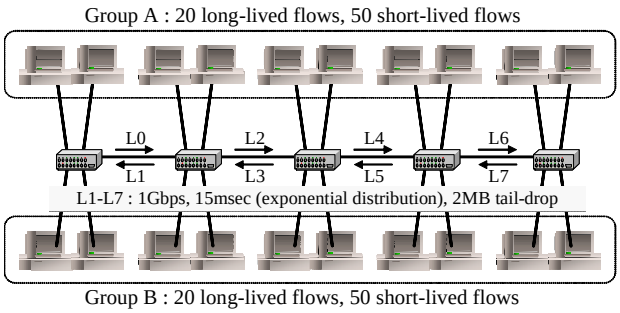

Fig. 2: Network model

We tested three cases 1) all flows use Reno, 2) half of the flows use Reno and the rest of the flows use the high-speed protocol, and 3) all flows use the high-speed protocol. We generated 12 different configuration sets, which are run for 10 minutes each.

B. Efficiency Improvement

In this section, we first confirm the performance improvement of high-speed protocols. Then, we investigate flow-by-flow behavior and discuss the characteristics of high-speed protocols. The following results are obtained with all flows using the high speed protocol.

Fig. 3 shows the average utilization of 8 backbone links. The number of long-lived flows is set to 1, 10, and 40, successively. As shown in this figure, all high-speed protocols improve the average link utilization especially when traffic load is light. Compound-TCP achieves slightly lower utilization than others, but the difference is not that significant. Vegas, as it is not designed as a high-speed version, shows the lowest efficiency, even worse than Reno. This appears to be due to the frequent slow start behavior of short-lived flows, the presence of small buffer sizes, as well as multiple bottleneck links. In such environment there are lots of packet losses even when the link is not fully utilized and thus delay-based protocol like Vegas tend to result in poor performance.

Although all high-speed protocols perform well in terms of average link utilization, they are very different in detailed behaviors. Figure 4 focuses on the per-flow throughput improvement achieved by the high-speed protocols. The number of long-lived flow is 40; and thus we have fairly loaded paths. The improvement is defined here as:

$$\textit{Relative throughput of flow } i = T_i^{HS+HS} / T_i^{Reno+Reno}$$

where T_i^{HS+HS} and $T_i^{Reno+Reno}$ are the throughput of flow i in the third run where all flow use high-speed protocols and the first run where all flow use Reno, respectively. In Fig. 4, the relative throughput of each flow is shown for various base RTT on x-axis. In Fig.5, relative throughput of each flow is plotted against the number of hops the flow traverses on x-axis.

Figures 4 and 5 illustrate the different character of the high-speed protocols. Under protocols having progressive congestion window increase like Scalable-TCP, flows with short RTT and short hop count quickly increase their throughput relative to the longer RTT flows, and occupy the links capacities. As a result, there would be no space for long RTT flows when they arise. On the other hand, newer protocols, like Hamilton-TCP, Compound-TCP and TCP-AReno, exhibit different behavior. They improve throughput of long RTT

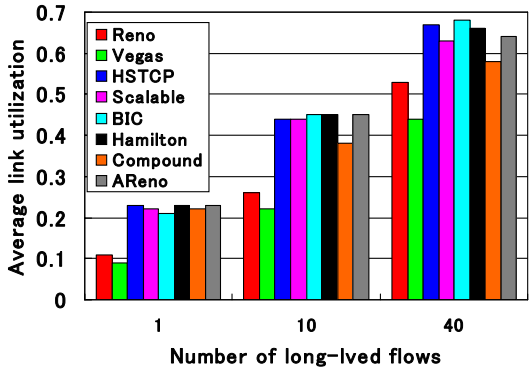

Fig. 3: Average link utilization

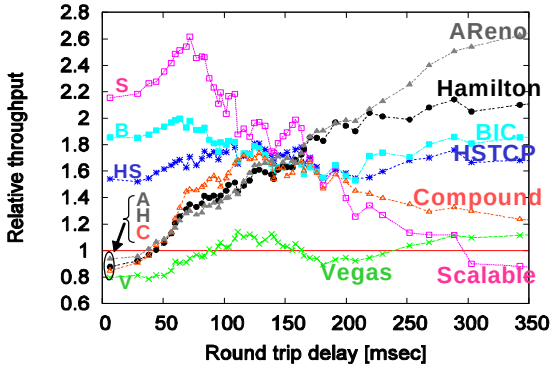

Fig. 4: RTT v.s. throughput improvement (40 flows)

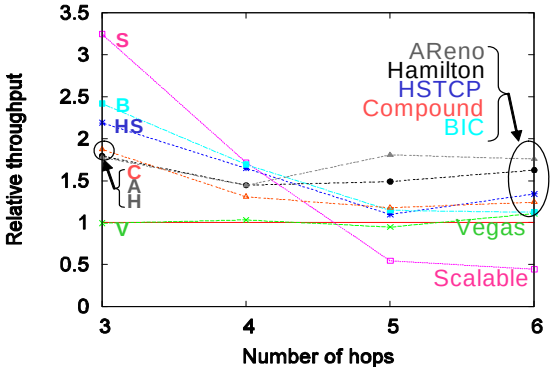

Fig. 5: Hop-count v.s. throughput improvement (40 flows)

flows relative to shorter ones. That is they improve the throughput of long RTT flows that suffer poor performance under Reno. Regarding Compound-TCP, long RTT flows, whose behavior is dominated by its delay-based function, do not perform very well. This is because such flows takes longer time to reach steady state, and as it is said for Vegas, the equilibrium can easily be broken due to slow start behavior of short-lived flows. Among these high-speed protocols, TCP-AReno has the best RTT-fairness. It has little performance improvement for flows whose RTT is less than 100msec, but it greatly improves performance of long RTT flows. Delay-based behavior of TCP-AReno is designed so that long RTT flows can increase congestion window faster.

As shown in Fig. 5, TCP-AReno and Hamilton-TCP are also good at improving performance of flows traversing multiple bottleneck links. We have to note here that, multi-hop flows consume more network resource than single-hop flows, thus allocating large bandwidth to multi-hop flows means inefficient resource usage. If average throughput is the concern, we would allocate as much resources as possible to single hop-flows. But doing so would result in lower fairness. In this paper, we do not address such trade-off between efficiency and fairness.

C. Fairness among flows

Figure 6 shows per-flow throughput in heavily loaded condition as a function of RTT. Scalable-TCP gets highest throughput for short RTT flows and lowest throughput for long RTT flows, i.e. its RTT-fairness is significantly worse than even Reno. BIC, HS-TCP, and Compound-TCP improve throughput regard less of RTT but their RTT-fairness is similar to that of Reno. Since flows with varying RTT are competing at multiple bottleneck links and slow start behavior of short-lived flows results in frequent and unpredictable packet loss, it is not easy for high-speed protocols to behave ideally as they are designed in terms of fairness and friendliness. Only TCP-AReno and Hamilton-TCP can improve RTT-fairness in such significantly challenging conditions. Although they have little throughput improvement for short RTT flows as expected, throughput of long RTT flows is significantly improved. The throughput ratio of 30msec RTT flow and 300msec RTT flow of Reno, HSTCP, and Compound is 9.7, 9.7, 11.3, and 7.3, respectively, while that of Scalable TCP is 20.3. Regarding Hamilton-TCP and TCP-AReno, the ratio is only 5.5 and 4.9, respectively.

To investigate fairness among flows a bit closer, we plot cumulative distribution of per-flow throughput in Fig. 7. As it is expected from the above results, Scalable-TCP has widest distribution of per-flow throughput, while Hamilton-TCP and TCP-AReno has small variation. For example, the fraction of flows achieving throughput less than 20Mbps is 36%, 27%, 20%, 18%, 16%, 6%, and 4%, for Scalable, Vegas, Compound, BIC, HS-TCP, Hamilton and AReno. On the other hand, 37% of Scalable flows, 32% of BIC flows, 29% of HSTCP flows, 22% of Compound flows, 20% of Hamilton flows, 18% of AReno flows, 12% of Reno flows and 11% of Vegas flows have throughput more than 200Mbps.

If the network is lightly loaded, on the other hand, we observe different results. Although we do not show figures in this paper due to space limitation, we have confirmed that all high-speed protocols have good RTT-fairness property, as well as achieve high throughput. When the load is low, even Scalable-TCP has significant improvement on the throughput of long RTT flows. Since long RTT flows and short RTT flows are not competing at backbone links and thus long RTT flows are not disturbed by short RTT flows, long RTT flows achieve high throughput as the high speed protocols are designed.

D. Friendliness to TCP-Reno

We examine the extent of Reno throughput degradation due

to the introduction of a coexisting high-speed protocol. We run a first simulation set with all Reno flows, and then run a second simulation set in which half of the Reno flows are replaced by

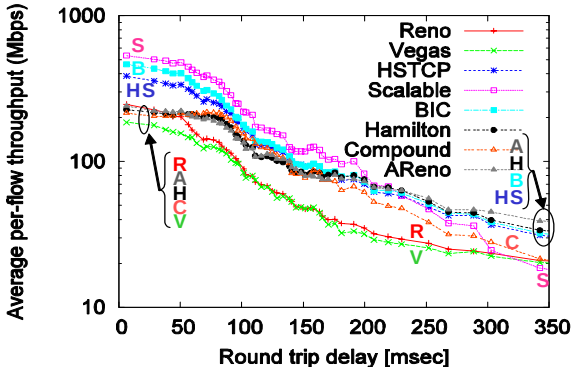

Fig. 6: RTT v.s. per-flow throughput (40 flows)

the high-speed protocols. Friendliness of high-speed protocols to Reno is evaluated by the throughput degradation of Reno flows due to such replacement. Thus, the degradation is defined here as:

$$\textit{Throughput degradation of flow } i = T_{Reno\_i}^{HS+Reno} / T_{Reno\_i}^{Reno+Reno}$$

where $T_{Reno\_i}^{HS+Reno}$ is the throughput of Reno flow i in the second run in which half of the flows use high-speed protocols. $T_{Reno\_i}^{Reno+Reno}$ is the throughput of Reno flow i in the first run where all flow use Reno.

Figure 8 shows throughput degradation of Reno flows and the lines are labeled by coexisting protocols. The number of long-lived flows is 40 thus the network is fairly loaded. There are three groups of high-speed protocols. Compound-TCP and AReno are most friendly to coexisting Reno flows because their delay-based control is not in effect in such congested condition and thus they behave just like Reno. They degrade the throughput of coexisting short lived Reno flows by roughly 25%. Due to the improved overall network utilization, the number of congested link increases. Longer flows traversing multiple links are more likely to encounter multiple congested links, and thus longer flows experience more degradation especially due to coexisting AReno flows. The second group consists of HS-TCP, Hamilton-TCP and BIC. They roughly halve the throughput of coexisting Reno flows. Scalable TCP severely degrades Reno throughput. Comparing to the case where Reno flows compete with other Reno flows, throughput of these Reno flows are degraded by 60%-85%, depending on RTT, when they coexists with Scalable-TCP flows. Since Scalable-TCP aggressively increases congestion window very quickly, flows experience more packet losses which significantly damages Reno flows.

On the other hand, when the network is lightly loaded, high-speed protocols are sufficiently friendly to Reno flows because, off cause, there is less contention in the network. Although graphs are not shown in this paper, we found that throughput degradation of Reno flows is less than 20% regardless of RTT when the number of long-lived flows is 10.

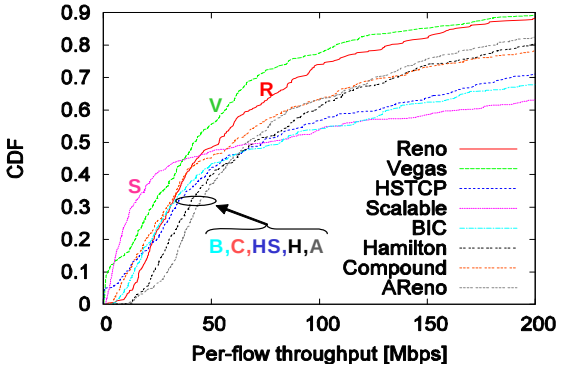

Fig. 7: CDF of per-flow throughput (40 flows)

E. File transfer time of short-lived flows

Figure 9 and 10 show average file transfer times of high-speed flows and Reno flows. The transfer time starts when the sender sends out the first packet and ends when the sender receives an acknowledgement for its last packet. As shown in Fig. 11, transfer of smaller files, e.g. less than 1MB, is completed during slow-start and there is no significant difference among the different protocols. On the other hand, transfer time of mid-sized files, for which congestion control is in effect, is improved by AReno, Hamilton, and BIC. Scalable TCP is not effective for mid-sized files because of its aggressive behavior described above. Its aggressive behavior increases packet loss probability and only a single packet loss affects much for these small and mid-sized transfers.

Figure 10 shows file transfer times when half of the flows use a high-speed protocol. The lines are labeled by the coexisting high-speed protocols. Except in the case of the aggressive behavior of Scalable-TCP resulting in increasing file transfer time of coexisting Reno flow, other high-speed TCPs are do not seriously degrade Reno throughput.

V. Conclusion

In this paper, we discussed a method of assessing interactions among different protocols. Using a pre-generated set of configurations repeatedly for different mixes of protocols, we can assess flow-by-flow and file-by-file behavior under different protocols. We provided simulation results of networks with multiple bottlenecks, a large number of short-lived and long-lived flows, and variety of RTTs.

The results generally show the effectiveness of most high-speed TCPs, and their differing characteristics in per-flow behavior. Earlier protocols like High Speed TCP and Scalable TCP achieve good efficiency and average throughput. The more recent generation of protocols improve RTT-fairness and friendliness to coexisting Reno flows. Scalable TCP shows degraded RTT-fairness and damage to coexisting Reno flows, especially in loaded networks. BIC has very similar character with High Speed TCP but has slightly improved per-flow throughput. In contrast, newer ones like Hamilton-TCP, Compound TCP and TCP-AReno have greatly improved fairness and friendliness. They achieve link utilization as high as the previous generation, but improvement of average per-flow throughput is limited because they allocate larger bandwidth to flows using multiple links and consume more

network resources. Hamilton-TCP has good fairness among

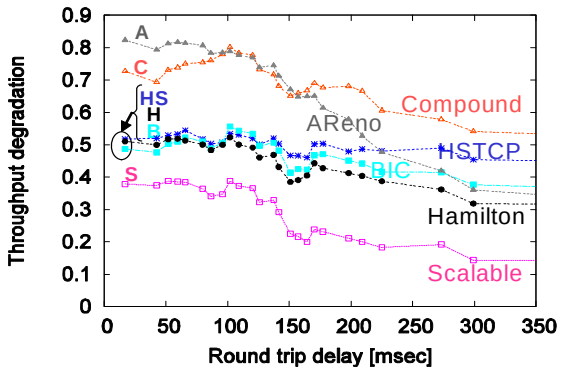

Fig. 8: Throughput degradation of Reno flows; indexed by coexisting flows (40 flows)

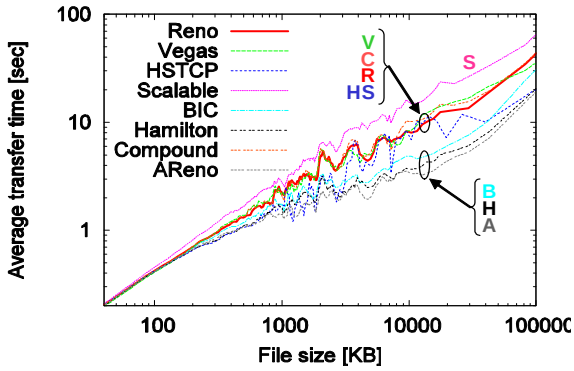

Fig. 9: File transfer time of high-speed flows (40 flows)

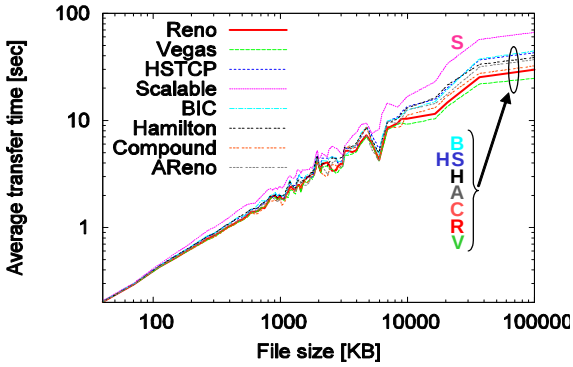

Fig. 10: File transfer time of Reno flows; indexed by coexisting flows (40 flows)

flows but its friendliness to coexisting long-lived Reno flows is comparable to previous generation protocols. While Compound-TCP has best friendliness to Reno even in highly loaded conditions, its throughput improvement is limited especially for long RTT flows. TCP-AReno maintains high link utilization and has best RTT-fairness, while maintaining good friendliness to Reno.

Future work related to this coexistence study would cover other prominent protocols like FAST, CUBIC, TCP-Libra, and variants of TCP Westwood, as well as expand the range of parameters used in this study. More importantly, we would like to carry measurements of actual implementations on emulated or existing Internet II paths to confirm the simulation results.

ACKNOWLEDGEMENTS

The authors would like to thank Prof. Lachlan Andrew, Prof. Steven Low, Dr. David Wei, Prof. Injong Rhee, and Mr. Takayuki Hama, for their helpful advice.

REFERENCES

[1] S. Floyd, “High Speed TCP for large congestion windows”, RFC 3649, 2003.
[2] T. Kelly, “Scalable TCP: Improving performance in high-speed wide area networks”, In Proc. of PFDnet, 2003.
[3] C. Jin, D. X. Wei, S. H. Low, “FAST TCP: motivation, architecture, algorithms, performance”, in Proc of IEEE INFOCOM, 2004.
[4] Lisong Xu, Khaled Harfoush, and Injong Rhee, "Binary Increase Congestion Control for Fast Long-Distance Networks", In Proc. of INFOCOM 2004
[5] Injong Rhee and Lisong Xu, “CUBIC: A New TCP-Friendly High-Speed TCP Variant”, In Proc. of PDLFnet, 2005
[6] R.N.Shorten, D.J.Leith, “H-TCP: TCP for high-speed and long-distance networks”, In Proc. of PDLFnet, 2004
[7] R. Wang, M. Valla, M. Y. Sanadidi, and M. Gerla, "Adaptive Bandwidth Share Estimation in TCP Westwood", In Proc. of Globecom 2002.
[8] H. Shimonishi, M. Y. Sanadidi, and M. Gerla, “Improving Efficiency-Friendliness Tradeoffs of TCP in Wired-Wireless Combined Networks”, In Proc. of ICC, 2005.
[9] H. Shimonishi, T. Hama, and T. Murase, “TCP-AdaptiveReno: Improving Efficiency-Friendliness Tradeoffs of TCP Congestion Control Algorithm”, In Proc. of PDLFnet, 2005
[10] S. Floyd, “Metrics for the evaluation of Congestion Control Mechanisms”
[11] D. Wei, “Time for a TCP Benchmark Suite”
[12] H. Bullot, “Evaluation of Advanced TCP Stacks on Fast Long-Distance Production Networks”
[13] Y. Li, “Experimental Evaluation of TCP Protocols for High-Speed Networks”
[14] S. Ha, Y. Kim, L. Le, I. Rhee, and Lisong Xu, “A Step toward Realistic Performance Evaluation of High-Speed TCP Variants”, In Proc. of PDLFnet, 2006
[15] K. Kumazoe, K. Kouyama, Y. Hori, M. Tsuru, and Y. Oie, “Can high-speed transport protocols be deployed on the Internet? : Evaluation through experiments on JGNII” , In Proc. of PDLFnet, 2006
[16] C. Marcondes, M. Y. Sanadidi, M. Gerla, H. Shimonishi. T. Hama, and T. Murase, “Inline Path Characteristic Estimation to Improve TCP Performance in High Bandwidth-Delay Networks”, In Proc. of PDLFnet2006, 2006
[17] P. Bardord and M. Crovella. Generating representative web workloads for network and server performance evaluation. In SIGMETRICS'98
[18] J. Aikat J. Kaur F. Donelson S. K. Jeffay , “Variability in TCP Roundtrip Times” ACM IMC 2003
[19] K. Tan, J Song, Q Zhang, and M. Sridharan, “Compound TCP: A Scalable and TCP-Friendly Congestion Control for High-speed Networks “, In Proc. of PFLDnet, 2006
[20] D. Wei and P. Cao, “A Linux TCP implementation for NS2”, available at http://www.cs.caltech.edu/~weixl/technical/ns2linux/index.html, May. 2006.
[21] R. Wang, M. Valla, M.Y. Sanadidi, B. K. F. Ng, M. Gerla, "Efficiency/Friendliness Tradeoffs in TCP Westwood", Seventh IEEE Symposium on Computers and Communications, Taormina, Italy, 1 - 4 July, 2002.