Improving Perceived Web Performance by Size Based Congestion Control

Thomas Ziegler, Hung Tuan Tran, and Eduard Hasenleithner

ftw. Telecommunications Research Center Vienna*
Donaucitystr.1, 1220 Vienna, Austria
{ziegler, tran, hasenleithner}@ftw.at

Abstract. Flow size based congestion control has the potential to improve user perceived Web performance due to the heavy tailed characteristic of file size distributions in the Web. After discussing the benefits and drawbacks of transport protocol and router based solutions for size based congestion control, guidelines for algorithm design are developed. Using these guidelines we find that size based congestion control needs to incorporate TCP models to avoid undesirable user incentives. Based on this insight we specify enhancements to TCP featuring size based congestion control and provide arguments for parameter settings. It is shown by simulation that our modified version of TCP significantly outperforms NewReno from a user perspective in scenarios using realistic models for Web traffic and topologies with multiple congested links.

1 Introduction

Heavy tailed distributions can be considered as one of the invariants when analyzing Internet performance. The evidence of heavy tails can for instance be found in traffic arrival patterns causing burstiness over multiple time scales and thus a variety of headaches in traffic analysis for researchers and network designers. By exploiting the fact that file size distributions on Web servers are heavy tailed this paper can be seen as an attempt to draw benefits out of the heavy-tail misery.

As shown first in [1] for scenarios having the computational power in web servers as the bottleneck, average response times for Web downloads can be dramatically reduced using process scheduling mechanisms giving priority to short flows. The rationale behind this finding lies in a property of Web traffic we will call the "heavy tailed property of file sizes" for the remainder of this paper: the majority of files is short (the so called web mice) and constitutes only a relatively small portion of the load; long files are less numerously but constitute the major portion of the load. Thus, because short flows are high in number, we may expect to reduce average response times giving preference to short flows. Additionally, we expect that average response time improvements won't negatively affect performance of long flows because of the minor load caused by short flows.

* This work is funded by the Austrian K*plus* research program.

TCP congestion control currently adapts the congestion window independently of the size of the flow to be transferred. Additionally, it can be shown that TCP throughput for short flows is significantly smaller than for long flows. In other words, TCP does the contrary of what the heavy tailed property of file sizes would suggest for maximization of efficiency and additionally causes unfairness among short and long flows.

This paper proposes and investigates TCP Vienna, a new version of TCP employing flow size based congestion control. The basic motivation is to propose straightforward modifications of TCP alleviating unfairness against short flows and thus increasing efficiency. Additionally, it is of major importance to keep congestion control conservative and avoid incentives for misbehaving users to gain an unfair high share of the link capacity due to prioritization of short flows.

The paper is structured as follows. Section 2 reviews related work. The pros and cons of various flow size aware mechanisms to increase Web efficiency and fairness are discussed in Section 3. Section 4 explains the Vienna enhancements to TCP congestion control. After giving an overview on the simulation scenarios, performance evaluation results are shown in Section 5. Finally, Section 6 concludes this paper.

2 Related Work

[1], [2], and [3] have proposed and analyzed size based process scheduling in servers to improve Web performance by exploiting the heavy tailed property of file sizes. Using queueing theory and measurements it has been shown that the performance of overloaded Web servers can be improved by a factor four and more if Shortest Remaining Processing Time (SRPT) scheduling is employed. SRPT naturally favours short jobs having shorter remaining processing times than long jobs. The size of a Web object is not known a-priori in case of dynamic content. Thus [4] shows that similar web server performance improvements can be achieved by SRPT process scheduling of jobs with unknown duration.

Inspired by the work on web server performance improvements [5] aims at achieving similar goals in case the link bandwidth at an Internet router, and not the computational power at a web server, constitutes the bottleneck resource. The basic idea is to keep per flow state at the edge router and mark flows according to their length using a few DiffServ Codepoints. A core router can examine Codepoints and assign packet drop priorities accordingly. As a metric for the flow length a flow's number of bytes received by an edge router is used. For the reminder of the paper we will refer to this idea as "the Router Based Approach (RBA)". In [6] it is shown by analysis and simulation that RBA can improve response times by an order of magnitude in case of a heavy tailed flow size distribution. In case of exponentially distributed flow sizes (light tail) average response times can be improved slightly, however, this improvement comes at the cost of long flows.

[7] investigates bandwidth allocation criteria using flow size based differentiation. As a preliminary implementation of these criteria it is shown that an upgraded TCP Reno source setting the increase and decrease parameters of the congestion window during TCP's congestion avoidance phase according to the residual flow size can increase performance in the range of 30-40% compared to standard TCP Reno. With-

out argumentation for parameter dependency on flow size, round trip time and drop probability the multiplicative decrease parameter is varied between 0.01 and 1; the additive increase parameter is varied between 0.25 and 10. For the reminder of the paper we refer to this kind of approach as "the Transport protocol Based Approach (TBA)".

Other examples for the TBA approach to avoid a retransmission timeout and Slowstart in case of small congestion window sizes at the start of a TCP connection have been proposed in [8], [9], and [10]. [8] proposes to set the initial TCP congestion window size to a maximum of 4 segments (see section 4.2 for more details) increasing the performance of short flows. [9] proposes to chop TCP segments into smaller chunks increasing the congestion window size in units of packets and thus avoiding retransmission timeouts in case of small windows. The main idea in [10] is to allow the TCP data sender to transmit new segments already in response to the first and the second duplicated ACK, keeping the ACK clock going and thus avoiding Slowstart in case of small windows.

3 Router vs. Transport Protocol Based Approach

Router and transport protocol based approaches both have their benefits and drawbacks which shall be highlighted in subsequent high-level considerations.

Performance gains: RBA has the potential to provide higher performance gains than TBA. For instance, TBA exhibits limitations in increasing throughput for flows having a size of only a few packets (which is quite common in case of Web traffic [15]). RBA routers could schedule packets of short flows with strict priority queuing, which corresponds to a significantly stronger flow length based differentiation than achievable with TBA without violating the conservativeness of TCP congestion control. However, as shown in [7], strict priority queueing would be an undesirable policy for RBA due to discrimination of long flows in case flow sizes are not heavy tailed and due to undesirable bandwidth allocation effects. Thus a more conservative policy for flow length based discrimination has to be chosen anyway for RBA (see for instance [6]).

Incremental deployment: TBA only requires modification of the TCP data sender. It thus supports incremental deployment and would typically be implemented in Web server transport protocols. RBA does not support incremental deployment, it rather requires standardization as core routers need to be able to correctly interpret codepoints set by edge routers. A full RBA implementation in an Autonomous System would require upgrading all routers potentially subject to congestion.

Scalability: RBA requires per microflow state at edge routers. It is questionable whether the performance gains by flow size based congestion control (although impressive) would balance the cost for per flow state in routers considering the fact the Web and TCP traffic is usually rather low priority traffic. We are not aware of scalability problems with TBA.

Application specific protocol design: Exploiting the heavy tailed property of Web file sizes per definition means violating the important paradigm of designing protocols for a broad spectrum of applications. In case of RBA this is problematic because edge routers are generally not able to distinguish between applications, e.g. if IPsec is used.

Thus RBA would be performed for all kinds of applications using TCP, independently of the characteristic of their flow size distribution. On the contrary, TBA would typically be deployed in Web servers where this problem does not exist. Of course, for RBA and TBA the argument that the Web is not "just another Internet application" holds.

Fairness and avoidance of undesirable incentives for misbehaving users: Another important problem that might come with TBA and RBA is an undesirable incentive for users to chop a long flow into many short flows in order to gain higher throughput. This incentive is created in case TBA or RBA were designed to provide short flows with higher throughput than long flows. A possible solution to this problem is that TBA and RBA like mechanisms need to be designed such that a short TCP flow achieves smaller or equal throughput than a long flow under the same network conditions, i.e. packet loss probability and round trip time (RTT). In other words, fairness should be a design goal not only for congestion control purposes but also to avoid undesirable user incentives. Estimating throughput of a long flow under the same network conditions is hard to achieve in case of RBA because routers have no straightforward possibility to measure path drop probabilities and RTTs. As will be shown in Section 4, estimation of drop probability is feasible with TBA; RTT estimation is already implemented in TCP. Based on drop probability and RTT estimation congestion control parameters can be set such that throughput of short flows approaches, but does not exceed throughput of long flows.

Summarizing above bullets, we observe the dominant advantages of the transport protocol based approach in terms of deployment, scalability, application specific protocol design and potential for avoidance of undesirable user incentives.

4 Enhancements to TCP Congestion Control

4.1 Design Guidelines and Basic Idea

Section 3 provides a design guideline for size based congestion control. Adapting TCP congestion control parameters as a function of the flow size such that throughput of short flows approaches but does not exceed throughput of long flows enables exploitation of the heavy tailed property while maximizing fairness and avoiding undesirable user incentives[1].

The basic idea behind TCP Vienna is to estimate the throughput a "long flow exhibiting typical TCP behavior" would have under the network conditions a flow of arbitrary size currently experiences. By network conditions we mean drop probability and RTT; a "long flow with typical TCP behavior" means a TCP Reno flow with delayed ACKs enabled having infinite flow length. Having estimated drop probability and RTT the throughput of such a typical long flow can be computed using an approved model

1. As shown in figure 1, Section 4, throughput as a function of the flow size is a monotonically increasing function in case of current versions of TCP. Thus long flows achieve a disproportionally high share of the bottleneck capacity compared to short flows.

of TCP [11]. Using the same model, congestion control parameters can be adjusted such that throughput of an arbitrary-length flow approaches throughput of a long flow under the same network conditions as closely as possible.

Model based adoption of TCP parameters enabling the improvement of Web performance while avoiding undesirable user incentives is the main innovation of TCP Vienna compared to the RBA and TBA solutions proposed so far in [6] - [10].

4.2 What TCP Parameters to Adjust

Having defined flow length independent fairness as a design guideline there remains an important constraint to be considered when increasing throughput for short flows. TCP congestion control must stay conservative in order to avoid congestion collapse in certain scenarios. Keeping this constraint in mind the TCP parameters to be adjusted according to the flow length and their bounds may be identified.

Additive increase, multiplicative decrease: the additive increase (α) and multiplicative decrease (β) parameters may be adapted according to the flow size during the congestion avoidance phase. Without argumentation the decrease factor is varied between 0.01 and 1 and the increase factor is varied between 0.25 and 10 in [7], dependent on the residual size of a flow. In TCP Vienna parameter adoption happens based on TCP models, more conservatively, and taking above design guidelines into account. An arbitrary sized flow should exhibit congestion control behavior at least as aggressively as a typical TCP Reno flow having infinite length to achieve a fair share of the throughput. A typical TCP Reno flow increases the congestion window (cwnd) by 1/cwnd at the receipt of an ACK and halves the congestion window at fast retransmit, fast recovery in case a packet is lost. Thus the decrease parameter is lower bounded by $\beta_{min} = 0.5$; the increase parameter is lower bounded by $\alpha_{min} = 1$. In order to stay conservative, and independently of a flow's size, we always want to decrease the window somewhat in case a packet is lost. Thus we use $\beta_{max} = 0.9$ as an preliminary upper bound for the multiplicative decrease. We set the upper bound of the increase parameter to $\alpha_{max} = 8$ in our simulations, a similar value as proposed in [7]. Note that while lower bounds for additive increase and multiplicative decrease are well argued in case of TCP Vienna, upper bounds require further substantiation by simulation experiments. Thus, as a first step, it is shown in Section 5 that the above parameter settings improve Web performance without exhaustively increasing loss rates at congested router output ports.

Window increase during Slowstart: a TCP flow having delayed ACKs disabled doubles the congestion window every RTT during Slowstart. A TCP flow having delayed ACKs enabled does so roughly every two RTTs due to the reduced ACK frequency. As TCP Slowstart behavior without Delayed ACKs is well known and has been identified as sufficiently conservative we may increase the congestion window more aggressively in case our TCP flow has delayed ACKs enabled in order to increase throughput of short flows. Consequently, we define the "Exponential Increase" parameter (γ), determining the congestion window increase in units of segments at the receipt of an acknowledgement during the Slowstart phase. In case of delayed ACKs disabled γ is fixed to 1*MSS (usual TCP without delayed ACKs; MSS denotes the Maximum Segment Size). In case of delayed ACKs enabled γ_{max} = 2MSS (imitating TCP behavior with delayed ACKs disabled) and γ_{min} = 1MSS (TCP with delayed ACKs enabled) define upper and lower bounds for γ.

Initial window: RFC 3390 [8] recommends min(4*MSS,max(2*MSS,4380bytes)) as an upper bound for the initial window size of TCP and discusses the benefits and drawbacks of increasing the initial window. Obviously, an initial window higher than 1*MSS favors short flows and is thus in-line with our goal. Independently of the flow length the initial window is set as recommended in [8] for TCP Vienna.

4.3 Modeling TCP Throughput

Taking constraints on upper and lower bounds for the TCP parameters identified in Section 4.2 into account, these parameters need to be adapted dynamically such that throughput B of a flow with size d approaches throughput of an infinite-length flow having delayed ACKs enabled and experiencing the same network conditions in terms of drop probability p and RTT. Note that B is a function of α, β, γ, d, p, and RTT.

For computation of B the approved TCP model in [12] which is based on the model in [11] is employed. This model computes the expected steady-state TCP transfertime for a flow as $E(T) = E(T_f) + E(T_{ss}) + E(T_{ca}) + E(T_d)$; throughput $B = d / E(T)$. T_f models the time during connection setup, T_{ss} models the time spent during initial Slowstart, T_{ca} models the time spent in congestion avoidance, and T_d models additional delay introduced by delayed ACKs.

We generalize this model to incorporate arbitrary values for α and β instead of fixed parameter settings, as for instance $\beta = 0.5$. This generalization requires a modified derivation of the expression for the expected congestion window size $E(W)$ which is used to compute T_{ca} (see eq. 22 in [12]). Due to space limitations we only roughly sketch this derivation, based on the analysis in [11]. Let TDP_i denote the Triple Duplicate Ack Period i (the time period between two consecutive packet losses), W_i denote the congestion window size at the end of TDP_i, and X_i the length of a TDP_i in units of RTTs. We observe that during TDP_i the congestion window increases linearly with a slope of α/b between βW_{i-1} and W_i. Thus, modifying eq. 2.8 in [11] to incorporate arbitrary values for α and β, W_i can be written as follows.

$$W_i = \beta W_{i-1} + \alpha \frac{X_i}{b} \qquad (1)$$

In eq. 1, b equals 2 in case of delayed ACKs, else it equals 1. Eq. 1 needs to substituted into eq. 2.9 in [11] and simplified resulting in an expression for Y_i, the number of packets transmitted in TDP_i.

$$Y_i = \frac{X_i}{2}\left(W_{i-1}\left(2\beta - \frac{\alpha}{2}\right) + \alpha(W_i - 1)\right) + \kappa_i \qquad (2)$$

κ_i denotes the number of packets sent in the last RTT of TDP_i. From eq. 2 and equation 2.6 in [11] it follows an expression for $E(W)$ by solving the resulting quadratic equation (compare to eq. 2.14 in [11] or eq. 23 in [12], respectively):

$$E(W) = \frac{1+\alpha\beta b}{db(4d+\alpha)} + \sqrt{\left(\frac{1+\alpha\beta b}{db(4d+\alpha)}\right)^2 + \frac{2(1-p)}{pb\beta(2\beta+\alpha/4)}} \qquad (3)$$

Figure 1 shows throughput B as a function of flow length calculated by the updated model for two settings of α and β. Other parameters: $\gamma = 1$, initial window = 1 MSS, p = 0.1, RTT = 0.2s.

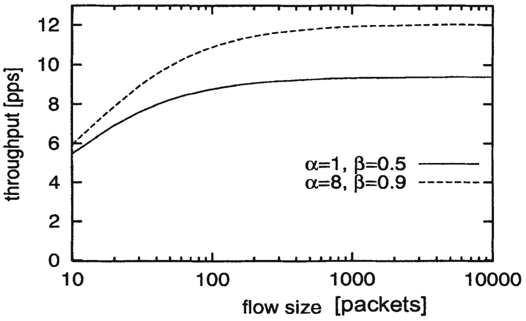

Fig. 1. TCP throughput according to model

4.4 The TCP Vienna Algorithm

Let α^* and β^* define k-dimensional vectors of increase and decrease values, and γ^* be a two dimensional vector. In correspondence with the guidelines presented in section 4.2 we set k = 5, α^*= (1,2,4,6,8), β^*= (0.5,0.6,0.7,0.8,0.9), and γ^*= (1,2). Additionally, let B_{max} denote the throughput of a typical TCP flow with delayed ACKs enabled and infinite length, where "infinity" equals a flow size of 10000 packets[2]. Given above definitions, the algorithm to compute size based congestion control parameters for a delayed acks enabled TCP data sender can be outlined as follows.

```
Initialization:
   Set $\alpha = \alpha_{max}$; $\beta = \beta_{max}$; $\gamma = \gamma_{max}$

Every m segments sent to the data receiver:
   Estimate p and RTT
   $B_{max} = B(1,0.5,1,10000,p,RTT)$
   Search for i,j:
      $B_{max} - B(\alpha_i^*,\beta_i^*,\gamma_j^*,d,p,RTT)$ is minimal and strictly positive
   Set $\alpha = \alpha_i^*$; $\beta = \beta_i^*$; $\gamma = \gamma_j^*$
```

Fig. 2. Simplified TCP Vienna pseudo code

2. Figure 1 illustrates that TCP throughput stays constant for flow lengths above a certain limit. Using the model and simulations we have verified that 10000 packets is above this limit for all relevant settings of p and RTT. Thus it is reasonable to consider 10000 packets as infinite flow length.

The search operation consists of a simple loop scanning α^*,β^*, and γ^*. *RTT* equals the smoothed round trip time as computed in all standard versions of TCP. The distance *m* in terms of number of packets to be transmitted between executions of the computations shown in figure 2 is set as an exponentially increasing function of N, the total number of packets transmitted so far. The rationale behind this choice is the need of frequent throughput estimations for short lived flows to enable correct parameter settings. Flows having already transmitted many packets can base their estimations on a high number of samples thus *m* may be higher. The exponentially increasing function is bounded by a constant *M*, which is set to 600 packets in our simulations.

$$m = min(M, 2^N) \qquad (4)$$

For estimation of the drop probability *p* we employ a simple heuristic. Let *L* denote the number of triple duplicate ACK events plus the number of partial ACKs seen so far[3]. *L* can be considered as a lower bound for the actual number of losses because it does not take lost packets into account which have been transmitted directly after a packet for which a partial ACK has been received during Slowstart. Let *R* denote the total number of retransmitted packets as counted by TCP. *R* overestimates the actual number of losses because packets which have already been delivered correctly at their first transmission are counted twice in case they are retransmitted during Slowstart. We consider underestimation of the loss probability by *L* as less erroneous than overestimation by *R*, thus the drop probability *p* is computed as the weighted average of *L* and *R* divided by *N*, the total number of packets sent so far:

$$p=(a*L+(1-a)*R)/N,\ a > 0.5 \qquad (5)$$

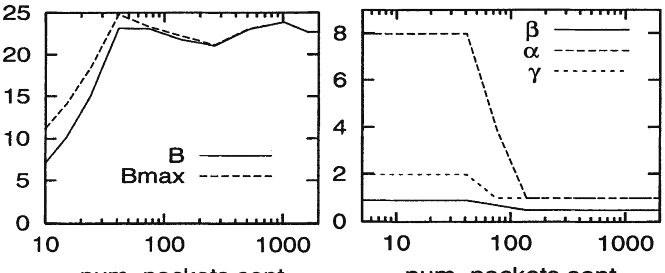

Fig. 3. Throughput and TCP parameter adoption

Figure 3 shows an example of our ns simulation [14] results with a single TCP Vienna flow over a single link topology to illustrate and evaluate the basic dynamics of the algorithm. The link enforces a uniformly distributed packet drop probability of 0.025; link propagation delay equals 100ms; the link bandwidth equals 10Mbps; initial

3. Triple duplicate ACK events (causing fast retransmit/recovery) and partial acks indicate packet losses. A partial ACK acknowledges new data but not the highest-sent packet before invocation of Slowstart or fast retransmit, respectively.

window = 4 MSS. The left part figure shows the typical evolution of B and B_{max} in units of packets per second; the right part figure shows α, β, γ parameters during the lifetime of a TCP Vienna flow. Comparing the initial phase in the left and right part of figure 3 we observe the effect of the upper bounds α_{max}, β_{max}, γ_{max} on α, β, γ and thus on throughput B. Although TCP Vienna parameters equal their upper bounds ($\alpha = 8$, $\beta = 0.9$, $\gamma = 2$), B is smaller than B_{max} for $N < 50$. In the course of the simulation B converges to B_{max} and α, β, γ are decreased to $\alpha = 1$, $\beta = 0.5$, $\gamma = 1$. In case of constant RTTs and drop probabilities B and B_{max} would both converge to a constant value. This is, however, not the case in our simulation due to RTT and drop probability variations.

Further simulations with different settings for drop probability and propagation delays show similar results. In case of shorter propagation delays the decrease of α, β, γ parameters is less steep and happens at a later point in time.

5 Performance Evaluations

Simulations are performed with ns-2 using the topology shown in figure 4 with n set to 1, 5, and 9. Links employ Drop Tail queue management; buffer sizes are set to 100 packets; packet sizes equal 500 octets. To ensure tight confidence intervals, sufficiently long simulation times (8000s) are selected.

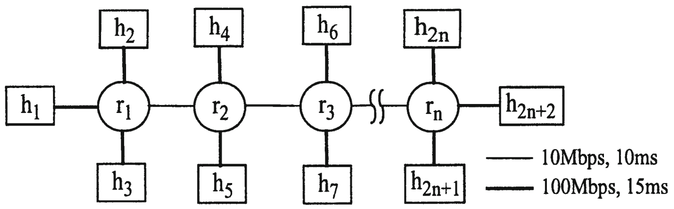

Fig. 4. Simulated network

Several aggregates of Web traffic generating load according to SURGE [15], a state of the art model for Web traffic based on real traffic traces, are simulated. SURGE models the object size as a mixed lognormal/pareto distribution (see [15] for details). We evaluate performance of an aggregate having Web servers at host h_{2n+2} and clients at h_1. Crosstraffic is created in forward direction by aggregates having clients at h_{2i-1} and servers at h_{2i+1} $(2<=i<=n)$ and in reverse direction by aggregates having clients at h_{2j+2} and servers at h_{2j}, where $1<=j<=n-1$. Load is varied by changing the number of users per aggregate. Considering a link between router r_i and r_{i+1}, the number of users in the evaluated aggregate equals the number of users in both crosstraffic aggregates sharing the link.

TCP Vienna has been implemented as a modified version of TCP NewReno; see Section 4 for details on the algorithm and parameter settings. The *a* parameter for averaging the drop probability is set to 0.75. Delayed ACKs are enabled. The flow length *d* is not known a-priori in case of dynamic Web content or HTTP1.1. Thus two versions of the algorithm are investigated. In TCP-V1 *d* corresponds to the total size of the flow (assuming a-priori knowledge); in TCP-V2 *d* corresponds to the number of bytes sent so far. These two versions are compared against TCP NewReno having an initial congestion window of 1 MSS (TCP-R1) and NewReno having an initial congestion window of 4 MSS (TCP-R2) according to [8].

In accordance with related work in [1]-[7], we measure mean slowdown, defined as the mean of response times divided by Web object sizes. Note that slowdown is best to measure user perception of Web performance because it takes into account that absolute reponsetime improvements are perceived more dramatically in case of small objects.

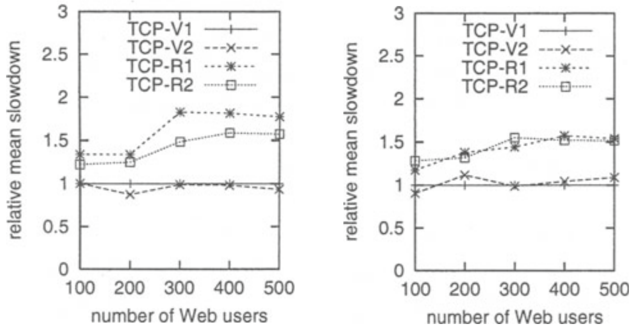

Fig. 5. Mean slowdown as a function of load for n = 1 (left part figure) and n = 9 (right part figure)

Figure 5 shows the mean slowdown of TCP-V2, TCP-R1, TCP-R2 normalized by the slowdown of TCP-V1. TCP-V2 exhibits similar performance to TCP-V1, thus we can conclude that there exists no need for a-priori information of the flow size in TCP Vienna. The figure shows dramatic improvements in user perceived performance with TCP Vienna. With slight dependency on the load we observe improvements of a factor 1.3 up to 1.8.

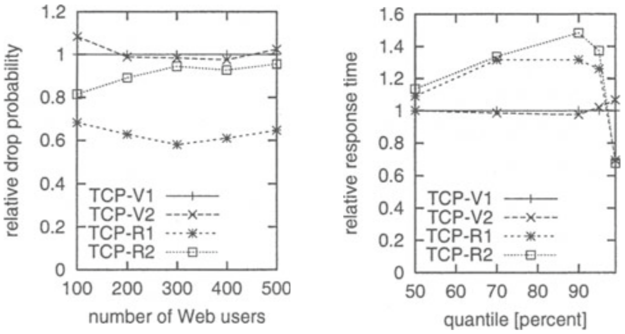

Fig. 6. Drop probabilities and quantiles of response time

The left part of figure 6 shows the drop probability as a function of load for n = 1. The dramatic performance improvements with TCP Vienna shown in figure 5 come to the expense of a slightly increased drop probability compared to TCP-R2, i.e.

NewReno with an initial congestion window set as recommended in [8]. TCP-R1 shows significantly smaller drop probabilities than the other versions of TCP as its congestion control is less aggressive than TCP-R2 and TCP Vienna.

The right part of figure 6 shows a typical example for the quantiles of the response time distribution of TCP-V2, TCP-R1, TCP-R2 normalized by the responsetime quantiles of TCP-V1. The selected scenario is n = 9 with a load of 300 Web users. As expected, both versions of TCP Vienna cause significantly shorter response times for the majority of short flows. This is indicated in the figure by showing that TCP Vienna causes smaller response times up to the 95% quantile. For the longest flows TCP Vienna causes higher responsetimes because short TCP Vienna flows grab a higher portion of the link capacity than TCP-NewReno in comparable scenarios. Thus for quantiles greater than 95% NewReno drops below TCP Vienna. Again, both versions of TCP Vienna exhibit similar performance.

Due to space limitations only a small subset of the simulations results shown in [13] can be presented in this paper. Simulations in [13] investigate the dependency of TCP Vienna performance on parameter setting. Furthermore, we find that it is beneficial for TCP Vienna if the TCP max_burst parameter, limiting the number of back-to-back segments sent in response to a single acknowledgement, is set to a value between 4 and 8. Additionally, comparing scenarios with active queue management and drop-tail in the routers, show that active queue management does not have a significant impact on the performance of TCP-Vienna in relation to TCP NewReno. Extensive simulations using topologies with multiple congested hops loaded with a realistic traffic mix of peer-to-peer and Web traffic show significant performance improvements of TCP-V1 and TCP-V2 compared to TCP-R2 in terms of response time quantiles and slowdown without significantly increasing drop probabilities or decreasing the goodput (throughput at user level) of the peer-to-peer traffic. This finding is in accordance with the "heavy tailed property of file sizes" (see section 1).

6 Conclusions

Based on high level considerations we find that transport protocol based solutions for size based congestion control provide significant advantages compared to router based solutions in terms of scalability, ease of deployment, and avoidance of undesirable user incentives. We emphasize that flow size based congestion control needs to stay conservative to avoid congestion collapse and must not have a bias against long flows but rather converge to a state of flow length independent fairness in order to avoid undesirable user incentives. The latter design guideline motivates us to enhance a well known TCP model for the requirements of size based congestion control. Subsequently, we identify the TCP parameters to be adapted as a function of the flow size, provide arguments for their parameter setting, and define flow size based enhancements to congestion control in TCP Vienna incorporating the enhanced model of TCP. Simulations illustrate the dynamics of parameter adaptations during the lifetime of a TCP Vienna flow.

Performance evaluations with realistic Web traffic over topologies with several congested hops show that TCP Vienna with and without a-priori information on the flow length outperforms TCP NewReno by a factor 1.3-1.8 in terms of user perceived performance. Additionally, we investigate the effect of TCP Vienna on drop probability and response time.

As a promising future research topic the investigation of TCP-Vienna in combination with TCP-SACK can be mentioned. Additionally, the proposals of [9] or [10] could easily be integrated into TCP Vienna. Subsequently to implementing TCP models based on table lookups for a variety of drop probability and RTT settings, a Web server implementation of TCP Vienna is planned.

References

[1] M.E. Crovella, R. Frangioso, M. Harchol-Balter, "Connection Scheduling in Web Servers", USENIX Symposium on Internet Technologies and Systems (USITS '99), Boulder, Colorado, October '99

[2] M. Harchol-Balter, M.E. Crovella, S. Park, "The Case for SRPT Scheduling in Web Servers", MIT-LCS-TR-767, October 1998.

[3] N. Bansal and M. Harchol-Balter, "Analysis of SRPT Scheduling: Investigating Unfairness", Proceedings of ACM Sigmetrics 2001, Conference on Measurement and Modeling of Computer Systems, 2001.

[4] M. Harchol-Balter, "Task Assignment with unknown Duration", Proceedings of ICDCS 2000

[5] L. Guo and I. Matta, "The War between Mice and Elephants", Proc. 9th IEEE International Conference on Network Protocols (ICNP'01) , Riverside, CA, November, 2001.

[6] L. Guo and I. Matta, "Scheduling Flows with Unknown Sizes: Approximate Analysis", Proceedings of ACM SIGMETRICS'02, poster session.

[7] S.J. Yang and G.D. Veciana, `` Size Based Adaptive Bandwidth Allocation: Optimizing the Average QoS for Elastic Users", Proceedings of IEEE INFOCOM 2002.

[8] M. Allman, S. Floyd, C. Partrige, "Increasing TCP's initial Window", RFC 3390, Oct. 2002

[9] M. Mellia, M. Meo, C. CAsetti, "TCP Smart Framing: using smart segments to enhance the performance of TCP", Proceedings of Globecom 2001

[10] M. Allman, H. Balakrishnan, S. Floyd, "Enhancing TCP's Loss Recovery Using Limited Transmit", RFC 3042, January 2001

[11] J. Padhye et al.,"Modeling TCP Troughput: A simple Model and its empirical Validation", Proceedings of ACM SIGCOMM 1998, August 1998

[12] N. Cardwell, S. Savage, T. Anderson, "Modeling TCP Latency", Proceedings of IEEE Infocom 2000, Tel Aviv, Israel, 2000

[13] T. Ziegler, "Investigating size Based Congestion Control", techn. Report, Oct. 2003, www.userver.ftw.at/~ziegler

[14] NS Simulator Homepage, http://www.isi.edu/nsnam/ns/

[15] P. Barford, M.E. Crovella, "Generating Representative Web Workloads for Network and Server Performance Evaluation", Sigmetrics 1998.