QFCP: a Router-Assisted Congestion Control Mechanism for Heterogeneous Networks

Jian Pu and Mounir Hamdi

Department of Computer Science and Engineering
Hong Kong University of Science and Technology
Hong Kong, China
{pujian, hamdi}@cse.ust.hk

Abstract. Traditional end-to-end congestion control measures packet loss or round-trip delay to sense network congestion. However, this mechanism may not work well in heterogeneous networks. Recently some router-assisted congestion control protocols are proposed to address this challenge. Quick Flow Control Protocol (QFCP) is one of them. QFCP allows flows to start with high initial sending rate and converge to the fair-share rate quickly based on feedback from routers. The rate allocation algorithm is quite simple and only needs to be run periodically by routers. We have implemented QFCP in ns-2. Simulations have been done to address the issues such as flow completion time of Poisson-arriving Pareto-distributed-size flows, adaptability to changing flow numbers, fairness on flows with different RTTs, and robustness to non-congestion packet losses. The preliminary results are promising.

1 Introduction

Congestion occurs when the packets injected into the network are more than that it is capable to deliver. In this situation, if the senders do not slow down their sending rate to relieve the traffic load, more and more packets will be dropped by the routers and little useful work can be done, which is called congestion collapse. Therefore, in order to make the network work stably and effectively, congestion control mechanisms have to be employed. Traditionally most of them are designed based on the end-to-end concept, only using the signals that can be measured at the end-system to sense the possible network congestion. Some protocols use packet loss as the signal of network congestion, such as TCP, HighSpeed TCP, and Scalable TCP. Others employ increasing round-trip delay as the implication of congestion, such as TCP Vegas, and FAST TCP. However, the assumption that packet loss or delay increase indicates network congestion does not always hold when the path includes wireless links. Some factors other than congestion such as link bit-error rate, unstable channel characteristics, and user mobility may also contribute to packet loss or round-trip delay variety. For example, studies have shown that TCP performs poorly in wireless networks due to its inability to distinguish packet losses caused by network congestion from those attributed to transmission errors. Much valuable bandwidth resource is wasted since

TCP unnecessarily reduce its congestion window when non-congestion-related loss happens.

Recently some router-assisted congestion control protocols have been proposed to address this challenge, such as XCP [1], VCP [2], and RCP [3]. XCP is the first successfully designed protocol of this kind. It outperforms TCP in both conventional and high bandwidth-delay networks, achieving fair bandwidth allocation, high link utilization, small standing queue size, and low packet drops. VCP can be treated as a simpler version of XCP. It uses only 2 bits to encode the feedback from routers and is easier to deploy in the Internet. RCP proposes to use per-flow rate instead of per-packet window adjustment as the feedback from routers. The advantage of router-assisted congestion control mechanism is that routers are the place where congestion happens and only routers can give precise feedback on the network condition. The disadvantage is the deployment difficulty and the additional workload on routers. So there is a trade-off between the performance and the complexity. If router-assisted scheme does not outperform end-to-end scheme significantly, end-to-end scheme is preferred since it is easier to deploy in the Internet.

Another issue is that Additive Increase Multiplicative Decrease (AIMD) algorithm is too conservative for flows in high bandwidth-delay product networks. This becomes increasingly important as the Internet begins to include more high-bandwidth optical links and large-delay satellite links. Research on the Internet traffic has revealed an important feature at the flow level: most of the flows are very short, while a small number of long flows account for a large portion of the traffic [4], [5]. This is known as the heavy-tailed distribution. But TCP models flows as “long-lived flows”, which means that all flows are assumed to be long enough to reach their fair-share sending rates before finishing. This assumption is acceptable when the Internet is mainly composed of low-speed links. However, if most flows are short as in today’s high-speed networks, can the congestion control algorithm built in TCP continue to work efficiently? The answer is no. On the one hand, a short flow always starts with a very low initial sending rate and very likely finishes before reaching its fair-share rate. The duration of a short flow may be significantly prolonged due to packet loss, which causes timeout and packet retransmission [6]. On the other hand, a large number of short flows also adversely impact the performance of long flows. [7] shows that randomly generated sequence of short flows may reduce the throughput of long flows up to 10%, and some special pattern of short flows can even cause greater reduction (>85%). The reason is that short flows spend most of their lifetime in the Slow Start phase when their congestion windows increase exponentially. Thus, a burst of short flows can rapidly capture a great portion of the bandwidth and driven long flows into timeout and window-halving. But the AIMD algorithm grabs the available bandwidth very slowly and makes the time of converging to the fair-share rate very long for the long flows.

As the Internet keeps evolving into a high bandwidth-delay product (BDP) network, more and more flows are becoming short flows. And the adverse impact of TCP congestion control may become increasingly severe. We need to design a new congestion control protocol for the high-speed networks achieving the following metrics:

- For short flows, they can get high initial sending rates at the startup so that they can finish quickly (“quick start”).

- For long flows, they can converge to the fair-share sending rate quickly and have maintainable high throughput ("quick convergence").

In most currently proposed congestion control protocols, each new flow starts with a low sending rate and then probes the network for the unused bandwidth. And they only show that they can achieve fair rate allocation for long-lived flows. But short flows may suffer since they are not long enough to compete for fair bandwidth allocation.

If the router can find a way to compute the number of active flows and assign the fair-share rate to each flow, there will be no need to let flows wait for many RTTs to probe the network by themselves. Hence, we try to design a router-assisted congestion control protocol similar to XCP and RCP. The sending rate of each flow is controlled by routers along the path. The router senses the degree of congestion periodically and calculates a global fair rate for all flows passing through it. A new flow will start with the same sending rate as the other ongoing flows because they get the same feedback from routers. Additionally we want the routers:

- Do not store any per-flow information.
- Do not maintain any large hash tables or do any hash computation (vs. hash-based flow counting algorithm).
- Do not do complex per-packet calculation on routers.

2 Protocol Design

2.1 Framework

The basic idea is to establish a feedback control system in the network. Periodically a router checks the input traffic load and the queue size to sense the current network condition. Then it uses this information to generate feedback to control the sending rate of all flows through it.

We use a similar framework of Quick-Start [8], but we extend the rate-request-and-grant mechanism to the whole lifetime of a flow: (1) The sender includes a Rate Request field in the header of each outgoing packet and sets the initial value of this field to be the desired sending rate of this sender; (2) When the packet reaches a router, the router compares the value in Rate Request field with the router's own fair-share rate and puts the smaller one back into that field; (3) On receiving the packet, the receiver copies the Rate Request field into the Rate Feedback field of the corresponding ACK packet and sends it back to the sender; (4) When the sender receives the ACK packet, it reads the value in the Rate Feedback field and adjusts its sending rate accordingly.

The sender sets the size of its congestion window according to the rate feedback using the formula:

$$cwnd = rate * RTT\ , \tag{1}$$

where *cwnd* is the congestion window size, *rate* is the rate feedback, and *RTT* is the moving average round-trip time measured by the sender. This is because conges-

tion window governs the flow throughput and its size (*cwnd*) determines the number of packets that can be outstanding within one RTT.

2.2 Rate Allocation Algorithm

The rate allocation algorithm is used by the router to compute the fair-share rate R periodically. One router maintains only one global fair-share rate for each output interface. This rate R is the maximum allowed rate for flows going through this interface during the current control period T. T is set to be the moving average of RTTs of all packets. The current number of flows through this interface is estimated using the aggregate input traffic rate and the fair-share rate assigned in the last control period. And the fair-share rate is updated based on the flow number estimation as follows.

$$N(t) = \frac{y(t)}{R(t-T)}\ , \tag{2}$$

$$R(t) = \frac{C - \beta \cdot \frac{q(t)}{T}}{N(t)}\ , \tag{3}$$

where $N(t)$ is the estimation of the flow number, $y(t)$ is the input traffic rate during the last control period, $R(t)$ is the fair-share rate, C is the bandwidth capacity of the output link, $q(t)$ is the queue size, β is a constant parameter, T is the control period.

This parameter β can be set as a policy by the router administer. A large value of β means one wants to drain up the queue more aggressively. The theoretical analysis of the impact of this parameter is left to future work. Currently we set the value of β as 0.5 in our ns-2 implementation.

2.3 Technical Details

Burstiness Control: When designing congestion control protocols for large BDP networks, researchers often find that pure window-based rate control is not enough and additional burstiness control is needed (e.g., FAST TCP [9]). This is because window control is too rough and may trigger a large burst of packets injected into the network all at once (e.g., in the Slow Start phase of TCP). But such violent increase of congestion window is sometimes unavoidable in order to achieve good responsiveness and high throughput, especially in large BDP networks. In QFCP, we use the rate-based pacing to avoid possible burstiness caused by a sudden increase of congestion window. So although the sender may set a large window as approved by the rate feedback in the first ACK packet, it still needs to pace these packets out in one RTT based on the assigned sending rate. Thus, congestion window controls how many packets can be sent in one RTT, while burstiness control paces these packets out in a smooth way.

Rate Stabilization: If the assigned fair-share rate changes too quickly, the formula (2) we use to estimate the number of flows based on the previously assigned rate may fail. The reason is that there may be flows with RTT longer than the control period T using $R(t–2T)$ instead of $R(t–T)$ as the sending rate. So for the sake of the accuracy of

flow number estimation, we don't want the rate assigned for two consecutive intervals to be very different. Thus, in order to stabilize the rate, we use the average of the current computed rate and the most recently assigned rate as the new rate.

Reaction to Packet Loss: If the queue of an interface is overflowed, the router will drop packets and add the number of dropped packet to $q(t)$ as $q(t)$ in formula (3), because this is the real queue size that should be drained during the next interval. The router will use this "virtual queue size" in formula (3) to compute the new rate. The sender just retransmits the dropped packets without any further reaction. This is very different from the loss-based congestion control protocols, which will adjust the congestion window if encountering packet loss. So in QFCP, it is very easy to differentiate the two kinds of packet loss: one is due to the congestion; the other is due to transmission error. Because the rate is totally determined by the routers, the routers will adjust the rate according to the degree of congestion. There is no need for the end-systems to guess whether congestion happens or not when they encounter packet loss.

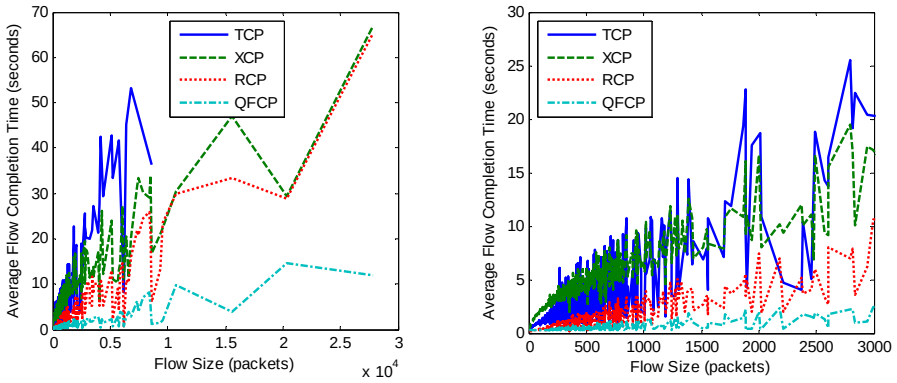
3 Simulation and Evaluation

In this section, we evaluate the performance of QFCP through extensive simulations using the packet-level network simulator ns-2 [10]. Unless specified otherwise, we use the dumb-bell network topology, where senders and receivers reside on each hand side and all flows go through the same bottleneck link. Simulations are run long enough to ensure the system has reached a consistent state. The parameter β of QFCP is set to 0.5. The buffer size on routers is set to be the delay-bandwidth product. The data packet size is 1000 bytes. And we use RED as the queue management scheme for TCP.

3.1 Flow Completion Time

For fixed-size flows (e.g., FTP, HTTP), the most attractive performance criterion is the flow completion time (FCT). Users always want to download files or web pages as fast as possible especially when they have paid for high-speed access links. Since a large number of flows in the Internet are short web-like flows, it is also important to investigate the impact of such dynamic flows on congestion control. Here we simulate a scenario where a large number of Pareto-distributed-size flows share a single bottleneck link of 150 Mbps. This is a typical mathematical model for the heavy-tail distributed Internet traffic which is composed of many short flows. The total flow number is 60000. The common round-trip propagation delay is 100 ms. Flows arrive as a Poisson process with an average rate of 625 flows per second. Flow sizes are Pareto distributed with an average of 30 packets and a shape parameter of 1.2. Thus, the offered traffic load on the bottleneck link can be estimated as: 8 ∗ packet_size ∗ mean_flow_size ∗ flow_arrival_rate / bandwidth = 1. Such high traffic load is often the situation where differentiates the performance of congestion control protocols. We record the size and completion time for each flow in the simulation, then average the

flow completion time for flows with the same size. Each simulation is conducted for each protocol: TCP-Reno, XCP, RCP, and QFCP. The scenario settings and the input data (i.e., the size and arriving time of each flow) are identical for each simulation. The results show that the Average Flow Completion Time (AFCT) in QFCP is significantly shorter than that in TCP, XCP or RCP.



(a) all flows (b) flows with size < 3000 packets

Fig. 1. Average flow completion time (AFCT) vs. flow sizes for Poisson-arriving Pareto-distributed-size flows. (a) is the global picture for all flows. (b) is a close look at short flows

For TCP, the AFCT is very oscillatory against the flow size. The reason is that although the exponential increase of congestion window in Slow Start does help some short flows finish quickly, the duration of other flows are prolonged due to packet loss. And we should point out that Slow Start is not a good way to shorten the duration of flows, because it actually does not know the proper initial sending rate but just intends to fill up the buffer of routers and cause packet losses, which prolongs the duration of all flows.

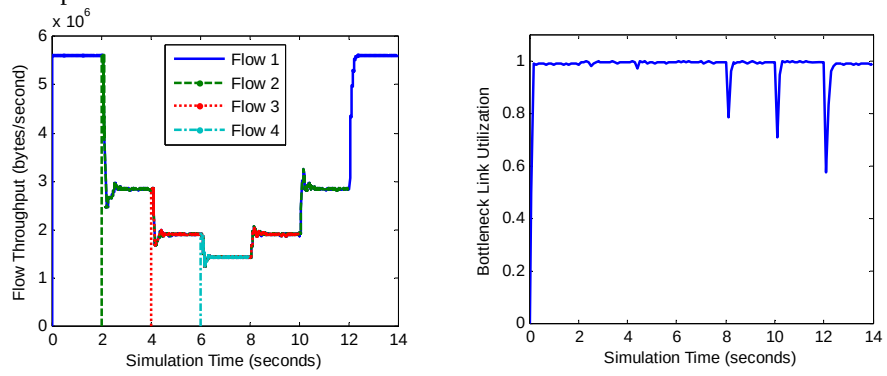
For XCP, it does not use Slow Start. Instead, when new flows join, XCP tries to reclaim the bandwidth from the ongoing flows and reallocate it to the new flows little by little. For short flows they may finish before reaching the fair sending rate. That is why the completion time of short flows in XCP is the longest. However, the AFCT against flow size in XCP is more stable than in TCP because XCP flows experience fewer packet losses.

For RCP and QFCP, both of them give a high initial sending rate to new flows based on the feedback from routers and help short flows finish quickly. However, the formula used in RCP to estimate the number of flows holds only when the input traffic just fills up the link capacity C, otherwise it leads to wrong estimation of the flow number. This wrong estimation of flow number makes the rate allocation in RCP under-optimum and thus prolongs the FCT in general compared with QFCP.

3.2 Adaptability to Changing Flow Numbers

For fixed-time flows or long-lived flows, we are more interested in the fairness of link bandwidth sharing among flows, especially when flow number changes causing

bandwidth reallocation. In this simulation, four flows share a common bottleneck link of 45 Mbps. The common round-trip propagation delay of each flow is 40 ms. Flow 1-4 start at time 0, 2, 4, 6 seconds and stop at 14, 12, 10, 8 seconds respectively. The results show that QFCP can converge quickly to the fair-share sending rate as flows join and leave, and can maintain a high utilization of the bottleneck link. This simulation also shows a significant difference between QFCP and XCP: the high initial sending rate. In QFCP, any new flow starts with the same rate as the other ongoing flows, and then converges to the fair rate if this new flow is long enough. While in XCP, a new flow starts with a low sending rate and then converges to the fair rate. That's why QFCP can help short flows finish much faster than XCP as demonstrated in the previous simulation.

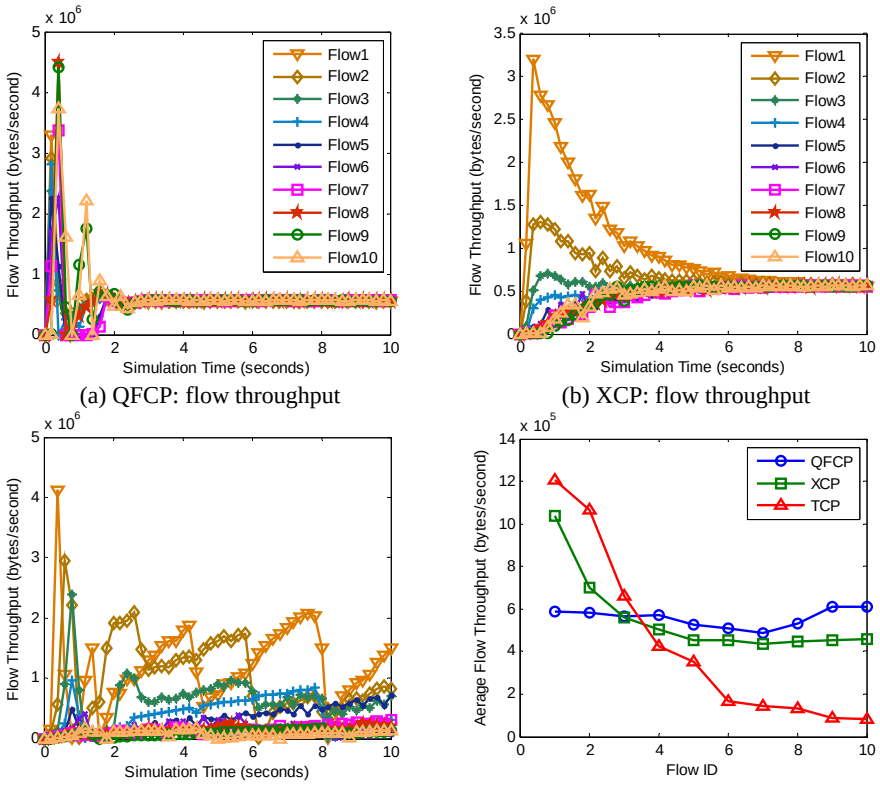


(a) QFCP: flow throughput (b) QFCP: bottleneck link utilization

Fig. 2. Flow throughput and bottleneck link utilization

3.3 Fairness on Variant RTTs

In this scenario we have 10 flows with different round-trip times sharing a common bottleneck link of 45 Mbps. The flow RTTs range from 40 ms to 220 ms in a step of 20 ms. All flows start at time 0 s and we wait until they converge to the steady state. The results in Fig. 3 show that although the RTTs of the flows are very different, QFCP fairly allocates the bottleneck bandwidth among the flows and converges to the fair-share rate even faster than XCP. For TCP, Fig. 3(c) (the legend is not shown for the sake of visibility) shows that some flows occupy most of the bandwidth while other flows keep sending at low rate. We then calculate the average throughput for each flow with different protocols. Fig. 3(d) shows that QFCP obtained the best fairness. It also confirms that TCP penalizes against flows with higher RTTs. Router-assisted protocols (QFCP, XCP) provide relatively fair bandwidth allocation for all flows although they use the average RTT as the control interval. However, for XCP, when the available bandwidth is large, flows with short RTTs increase their window much faster than flows with long RTTs, which causes transient unfairness as you can see in Fig. 3(b). This is because a flow with short RTT updates its window more frequently and gains more benefit from the positive feedbacks. While in QFCP, all flows get the same feedback on rate if they go through the same bottleneck no matter what RTTs they have. Thus, QFCP can converge to the fair-share rate faster than XCP.



(a) QFCP: flow throughput (b) XCP: flow throughput

(c) TCP: flow throughput (d) Fairness

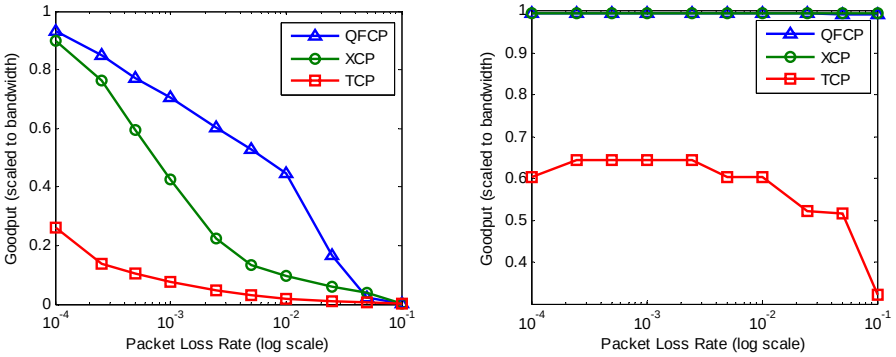
Fig. 3. Fairness on flows with different RTTs

3.4 Robustness on Lossy Link

This simulation tests for the sensitivity to non-congestion loss. The bottleneck link is 100 Mbps and can randomly generate packet losses at a fixed probability. The common RTT is 40 ms. Fig. 4(a) tests for data packet losses on the forward direction and Fig. 4(b) tests for ACK packet losses on the reverse direction. The loss rate ranges from 0.0001 to 0.1 and covers most typical loss ratios seen in a wireless network. Each simulation runs for 60 seconds and we record the highest acknowledged packet number seen at the sender to calculate the goodput. Goodput is the data packets that are successfully transferred (ACKed). It is necessary to differentiate goodput from throughput in this scenario since lossy link may cause massive packet retransmission but only goodput (highest ACKed packet number) is the work actually done from the user's vision.

The simulation results in Fig. 4 show that generally router-assisted congestion control performs better than TCP. ACK losses do not impact the goodput of XCP or QFCP very much because ACKs are cumulative and ACK loss can be recovered by subsequent ACKs. In contrast, data packet losses require retransmission. QFCP is

more robust to data packet losses than XCP because the feedback in QFCP has more redundancy. For XCP, each ACK carries unique feedback on congestion window adjustment and the whole ACK packets in one RTT together give the correct value of aggregate feedback. Packet loss may cause difference between the actual congestion window size and the one expected by routers, especially when the lost ACK packet carries a large value of feedback. This may happen when the current window is small while the target window is large (e.g., the situation after a timeout). But for QFCP, since the feedback is directly the flow rate and this value is only updated once every control period, any ACK in the current control period can give the correct window size to the sender. This kind of information redundancy gives high chance to prevent senders from starvation (keep sending at unnecessary low rate for a long time) in a lossy network.



(a) Data packet loss (b) ACK packet loss

Fig. 4. Goodput on a lossy link

As mentioned before, packet loss is not treated as congestion signal in our scheme. For example, if 3 duplicate ACKs are received by the sender, TCP will halve the congestion window and retransmit the lost packet indicated by the dup-ACKs. This window shrinking may significantly reduce the flow rate and may be unnecessary if the packet loss is due to transmission media error instead of congestion. While in QFCP, upon 3 dup-ACKs, the sender will only retransmit the lost packet without shrinking the congestion window, unless it is told to do so explicitly by the feedback field of the ACK. This prevents unnecessary reduction on the flow rate. But for packet loss indicated by timeout event, QFCP still shrinks the congestion window as TCP does. This is a conservative procedure because no feedback is available at this point. But if the network is not congested (i.e., a non-congsetion loss), any subsequent ACK will restore the congestion window size to a proper value and the flow rate will be recovered immediately.

4 Conclusions

Pure end-to-end congestion control scheme may unnecessarily reduce the sending rate when encountering non-congestion-related packet loss and underutilize the wireless

networks. Router-assisted mechanism may help us design more robust congestion control protocols for better utilizing the heterogeneous networks. But we should note that there is a trade-off between the performance and the complexity. In this paper, we design QFCP of this kind of protocol. It gives a high initial sending rate for any new flow as approved by routers along the path. The scheme can significantly shorten the completion time of both short flows and long flows. All flows can converge to the fair-share sending rate quickly whenever new flows join or old flows leave. And it also shows fairness for flows with different RTTs. It is easy for QFCP to differentiate non-congestion losses from congestion losses and thus prevent unnecessary window shrinking. The computation overhead on routers is acceptable and most calculations only need to do periodically. Future work may include implementing QFCP in Linux and deploying it in the real networks to test its performance. Establishing some mathematical models of QFCP and doing theoretical analysis on it are also desirable.

References

1. Katabi, D., Handley, M., Rohrs, C.: Congestion control for high bandwidth-delay product networks. Proceedings of ACM SIGCOMM '02 the conference on applications, technologies, architectures, and protocols for computer communications. ACM Press, Pittsburgh, Pennsylvania, USA (2002) 89-102
2. Xia, Y., Subramanian, L., Stoica, I., Kalyanaraman, S.: One more bit is enough. Proceedings of SIGCOMM '05 the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, Philadelphia, Pennsylvania, USA (2005)
3. Dukkipati, N., Kobayashi, M., Zhang-Shen, R., McKeown, N.: Processor Sharing Flows in the Internet. International Workshop on Quality of Service (2005)
4. Crovella, M.E., Bestavros, A.: Self-similarity in World Wide Web traffic: evidence and possible causes. IEEE/ACM Transactions on Networking **5** (1997) 835-846
5. Claffy, K., Miller, G., Thompson, K.: The Nature of the Beast: Recent Traffic Measurements from an Internet Backbone. Proceedings of INET '98 (1998)
6. Liang, G., Matta, I.: The war between mice and elephants. Proceedings of ICNP (2001) 180-188
7. Ebrahimi-Taghizadeh, S., Helmy, A., Gupta, S.: TCP vs. TCP: a systematic study of adverse impact of short-lived TCP flows on long-lived TCP flows. Proceedings of INFOCOM 2005, Vol. 2 (2005) 926-937
8. Jain, A., Floyd, S., Allman, M., Sarolahti, P.: Quick-Start for TCP and IP. IETF Internet-draft, work in progress (2005)
9. Jin, C., Wei, D.X., Low, S.H.: FAST TCP: motivation, architecture, algorithms, performance. Proceedings of INFOCOM 2004 the Twenty-third Annual Joint Conference of the IEEE Computer and Communications Societies, Vol. 4 (2004) 2490-2501
10. The network simulator ns-2. http://www.isi.edu/nsnam/ns