Handling two-way TCP traffic in asymmetric networks

Fatma Louati, Chadi Barakat[⋆], Walid Dabbous

Planète research group, INRIA Sophia Antipolis, France

{flouati, cbarakat, dabbous}@sophia.inria.fr

Abstract. The TCP congestion control protocol is mainly designed for bandwidth symmetric paths. As two-way asymmetric connections will probably become common case in the future with the widespread use of ADSL, satellites and other high-speed technologies, it is important to make sure that congestion will be properly handled in these environments. To this end, we propose in this paper a new Adaptive Class-based Queuing mechanism called ACQ for handling two-way TCP traffic over links that exhibit bandwidth asymmetry. ACQ runs at the entry of the slow link and relies on two separate classes, one for ACK packets and one for Data packets. ACQ proposes to adapt the weights of both classes according to the crossing traffic in order to maximize some utility function defined by the user or the network operator. We show by simulations that our mechanism is able to reach a good utilization of the available resources, managing then to maximize the satisfaction of the user of such asymmetric connections.

Key words: Bandwidth Asymmetry, Two-Way traffic, TCP performance

1 Introduction

The huge success of the Internet and its transformation into an important commercial infrastructure is in fact a double edge weapon. It is now necessary for Internet designers to consider new consumer expectations in term of performance, services and bandwidth demand [7,2]. Satellite networks and Asymmetric Digital Subscriber Lines (ADSL [13]) are promising technologies since they offer significant bandwidth increases over download paths of existing Internet traffics. So we have seen in the last years the growth of asymmetric and bidirectional access methods. Unfortunately TCP [3] is not prepared yet to face such cases [6].

In the TCP protocol, acknowledgments (ACKs) serve among other as a reliable clock for packet transmission: upon every ACK arrival at the source, there is one or more packet transmissions into the network. This clocking is based on the fact that, when arriving at the source, ACK packets are separated in time by at least the transmission time of a data packet at the bottleneck router on

[⋆] Corresponding author

the Data path. When the bandwidth available on one side of a link is considerably different that the one available on the reverse side of the link, the TCP connection suffers from congestion on the ACK path, yielding delays and losses of ACKs. This leads to a bad behavior of TCP/IP protocol, based essentially on the ACK-Clocked property. The problem is called *ACK-Compression* and it results in an increase in delays, further burstiness in the TCP traffic, difficulties to recover from losses, slowness of the congestion window increase and poor throughput. We refer to [10] for further details on the the ACK compression problem.

We study in this paper the problem of asymmetric links in the context of two-way TCP traffic. Consider a user that downloads data from the Internet via a high speed link while uploading data to the Internet via a low speed link, see Figure 1. The objective is to maximize the link utilization and the user satisfaction. The problem is very simple when both traffics are independent of each other. On asymmetric links the problem is challenging since traffics are dependent on each other. The dependence comes from the flow of acknowledgments (ACKs) that share the reverse path with the uploaded data, when the downloaded data is carried by TCP. The available bandwidth on the slow reverse link has to be shared by ACKs and data. The simultaneous presence of ACKs and data is known to cause several problems. The major problem is that ACKs are not responsive to congestion, so they will finish by monopolizing the available bandwidth and the rate of uploaded data will drop to zero. This may not correspond to the optimal allocation of the scarce resource on the reverse path.

In this context, we propose a class-based scheduling called **ACQ** (Adaptive Class-based Queuing) that adapts the available bandwidth between opposite direction traffics, and that manages to obtain the best utilization of a shared asymmetric link. We first present our model and the ACQ architecture. Then we report simulations showing efficiency of ACQ compared to existing approaches. In the last section we demonstrate through simulations that ACQ is a robust scheduling mechanism when confronted to changes in the network settings.

Related Work

Some solutions that require a change on the sender's and receiver's side have been proposed, such as Sender Adaptation or ACK Congestion Control[10]. Balakrishnan et al. [1,10], Kalampoukas [8] and the PILC group [14] show that performance can be substantially improved by making two operations on the ACK flow: simply suppressing ACKs on reverse channel (ACK Filtering [8,5]) and regenerating them after the slow reverse link has been crossed (ACK Reconstruction [4]). We retain these last two mechanisms since they do not change the TCP stack and are transparent to TCP peers. We use them for comparison in some of our simulations later in this paper.

To alleviate the problem of bidirectional traffic over asymmetric links, RFC 3449 recommends the use of ACK Filtering (AF)/ACK Reconstruction (AR) combined with an ACKFirst scheduling in the router. In [11], Shekar et al. show that if ACKs are given priority, the rate of packets on the slow reverse channel drops to unacceptable values since the channel will be monopolized by ACKs.

2 ACQ: Adaptive Class-based Queuing

Let x_f be the rate of downloaded data, and x_r the rate of uploaded data. Denote by $U_f(x_f)$ the satisfaction of the user from the downloaded data, and $U_r(x_r)$ his satisfaction from the uploaded data. The problem is then to find x_f and x_r that maximize the global utility function $U(x_f, x_r)= U_f(x_f) + U_r(x_r)$.

The two flows of ACKs and data in the upload direction have to be separated by some kind of two-class CBQ (Class-Based Queuing) buffer [16]. We propose a simple algorithm to adapt the rates of the two-class CBQ buffer at the input of the reverse link. We consider that ACKs and data packets are queued in separate buffers and are served in a fair-queuing way.

We consider that the ACQ buffer is able to measure the rate of data in both directions. This can be done by some signaling between both interfaces or by inferring the rate of data in the forward direction from that of ACKs in the reverse direction. Every time T, our algorithm measures the flow rate of data in both directions, and adapts the rate allocated to the uploaded data flow. The ACK flow will be allocated the rest of the bandwidth.

Let r_n be the rate allocated to the data flow in the upload direction at time nT, $n \in \mathbb{N}$. This rate remains allocated until time $(n+1)T$. Let $x_r(n)$ (resp. $x_f(n)$) be the measured rate of the uploaded data (resp. downloaded data) between nT and $(n+1)T$. Finally, let $U(x_r,x_f)$ be the total satisfaction of the user. We use the gradient projection method to update the value of r_n ,

$$r_{n+1} = x_r(n) + \gamma \frac{dU(x_r(n), x_f(n))}{dx_r} \tag{1}$$

This method assumes the existence of one maximum in the definition region of x_r and x_f, which we assume true. γ is a constant that tradeoffs stability and convergence rate. The experiments in the following will lead us to the optimal value of γ. We write,

$$r_{n+1} = x_r(n) + \gamma \frac{dU(x_r(n))}{dx_r} + \gamma \frac{dU_f(x_f(n))}{dx_f} \cdot \frac{dx_f(n)}{dx_r} \tag{2}$$

Since we don't know the explicit relation between x_r and x_f, we make the following approximation

$$\frac{dx_f(n)}{dx_r} = \frac{x_f(n) - x_f(n-1)}{x_r(n) - x_r(n-1)} \tag{3}$$

We obtain the following rule for setting the rate allocated to the uploaded data flow,

$$r_{n+1} = x_r(n) + \gamma \frac{dU(x_r(n))}{dx_r} + \gamma \frac{dU_f(x_f(n))}{dx_f} \cdot \frac{x_f(n) - x_f(n-1)}{x_r(n) - x_r(n-1)} \tag{4}$$

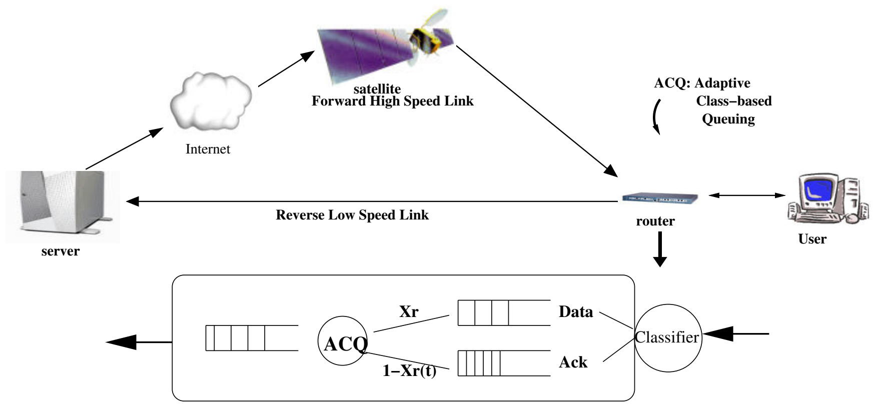

Fig. 1. ACQ architecture

We consider in this paper a particular utility function, which is equal to the bandwidth utilization. The user wants to maximize the sum of utilization in both directions. Let C_f be the available bandwidth in the downstream direction, and C_r the available bandwidth in the reverse direction. Hence, $U_r(x_r)= \frac{x_r}{C_r}$ and $U_f(x_f)= \frac{x_f}{C_f}$. The rule of updating rates becomes,

$$r_{n+1} = x_r(n) + \gamma(\frac{1}{C_r} + \frac{1}{Cf}\frac{x_f(n) - x_f(n-1)}{x_r(n) - x_r(n-1)}). \tag{5}$$

In order to achieve the user satisfaction and maximize the utility function described above, we propose to use a CBQ-like scheduler with two queues, one for ACK packets and one for Data Packets. Both queues are served according to the Weighted Fair Queuing algorithm. Each queue is allocated a fraction of the bandwidth C_r. As represented in Figure 1, ACQ runs at the entry of the slow link and proposes to adapt the weights of both classes according to the crossing traffic.

The aim of ACQ is to find in a minimum number of intervals T, the allocation scheme that allows the optimal user satisfaction, i.e. that maximizes the total utility function. We define the weight of the Data class as the variable $X_r(n) = \frac{r_n}{C_r}$, $X_r(n) \in [0,1]$. The weight of the ACK class is then equal to 1-$X_r(n)$. At each interval T we update the weight $X_r(n)$ as formulated in equation (5).

T represents the bandwidth allocation refreshment interval, its value is a tradeoff between stability and responsiveness of the system. T must be long enough to permit the traffic to react to a change in the bandwidth allocation. At the same time, T cannot be very long since this alters the stability of the system and slows its reaction to any change in traffic conditions. Since TCP adapts its window size every two Round-Trip Times (RTT), the minimum value of T is obviously twice the bigger round trip time of the connections.

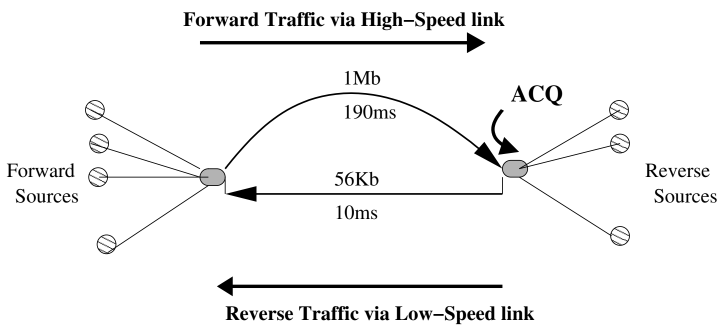
Forward Traffic via High−Speed link
1Mb
190ms
ACQ
Forward Sources
56Kb
10ms
Reverse Sources
Reverse Traffic via Low−Speed link

Fig. 2. Topology of the simulations

Concerning γ, this factor decides on the amount by which the rates of CBQ are updated every T. Giving a big value for γ will rapidly lead us to an unstable state, and a too small value of γ will necessitate very long time to converge. γ is a variable that must help the system to avoid big oscillations and to converge to the stable state as quick as possible. The choice of γ involves then a clear tradeoff. The unit of γ is $Kbps^2$.

3 Simulations and results

We use ns-2 [9] to simulate the network topology in Figure 2. N TCP flows start simultaneously on each side of the asymmetric link. All flows are long-lived, using TCP Reno. Packets are 1500 bytes for Data and 40 for ACKs. All routers are FIFO with 40 packets buffers. In the reminder of this paper, we call T_f the forward traffic and T_r the reverse one. The receiver windows of all connections are set to very large values.

3.1 Stability with the values of T and γ

It is important to choose the pair of T and γ that enables ACQ to give the maximum utilization of the link. In Figure 3, we plot the total utility function for different values of T and γ. Simulations last 10000 seconds to let the system the time to converge to its optimal point. We note that for every value of T, there exists a maximum value for the utility function. The higher the T, the smaller the value of γ that gives the maximum utility function. Figure 3 shows three optimal pairs : $T = 20$ seconds / $\gamma = 50$ Kbps^2, $T = 10$seconds / $\gamma = 100$ Kbps^2 and $T = 5$seconds / $\gamma = 150$ Kbps^2.

In fact giving a value of T, γ must help the system to be reactive to a change in bandwidth allocation. So if we have a small value of T, γ must be large. We see in Figure 3 that for T equal to 2 seconds, the utility function given by ACQ is really bad. This is quite normal because the RTT of our topology is around 1.2

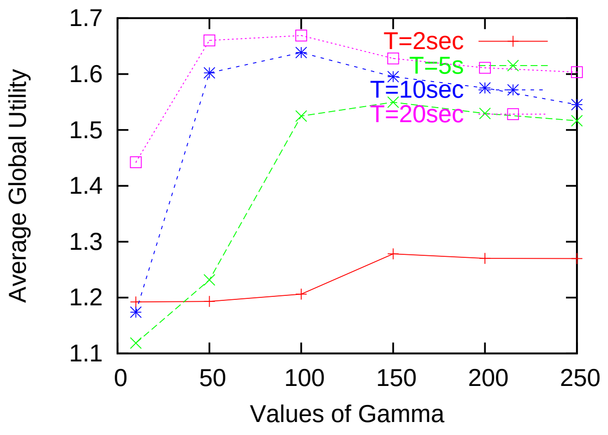

Fig. 3. Variation of the Utility Function with different values of γ and T

seconds (we mean by RTT the sum of links propagation times plus the waiting times in routers). T equal to 2 seconds is then too short and the system cannot react. In the contrary, if the value of T is sufficient to make the system converge (5, 10 or 20 seconds), γ must not be too large otherwise the bandwidth allocation will oscillate. On the other hand, we see that small values of γ lead to very poor results since the bandwidth allocation changes by small values.

We have also done simulations involving other scenarios, especially changing the number of connections and the values of RTTs. We don't report the results here for lack of space but as a general rule, we can claim that T and γ have different optimal values according to the topology and to the network load:

- If the network is under-loaded (small number of TCP connections) or average RTT is high, both the refreshment interval T and γ must be large so that sources can react.
- If the network is sufficiently loaded and if T is large, γ must be small to avoid important oscillations. However, if T is small, γ must be large to converge rapidly to the optimal utility function.

3.2 Bandwidth sharing with ACQ

In Figure 4 we plot the variation of Data and ACK queue weights versus the simulation time. We have 10 connections in each direction that start all randomly between 0 ms and 5 ms and stop at 10000 seconds. We chose an important duration of the simulations to let ACQ the time to adapt to the best utility function, but we notice that it converges quite rapidly. We use 10 seconds for T and 50 for γ. As we can see, the Data class obtain 80 per cent of the available bandwidth, leaving the other 20 per cent to the ACK class. We also remark that ACQ converges rapidly to these allocation ratios, our scheduler is then stable.

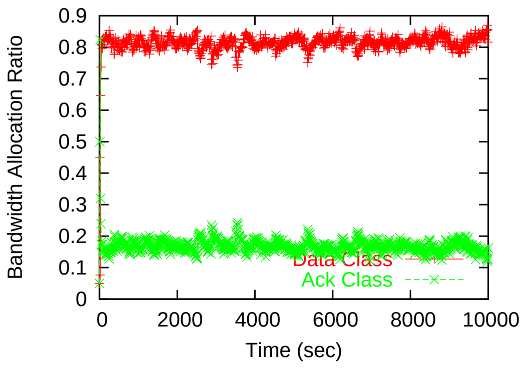

Fig. 4. Bandwidth share between ACK and Data class

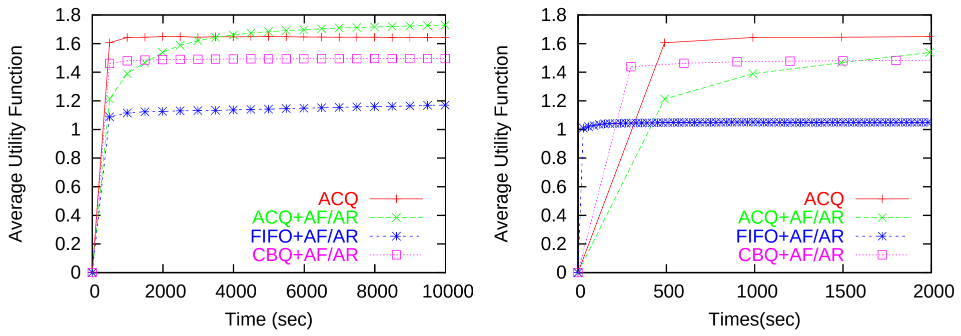

Fig. 5. Variation of the Utility Function values; zoom on the first 2000th seconds

3.3 Variation of the Utility function

In this section, we compare the average utilization of the link realized with three scheduling schemes: CBQ scheduling with equal sharing of the available bandwidth between ACK and Data class and with AF/AR mechanism in the ACK queue, ACQ with and without AF/AR in the ACK queue, and simple Drop-Tail (FIFO) with AF/AR mechanism.

ACQ manages to reach 1.62 as average utility function after around 500 seconds of simulations, see Figure 5 [1]. However, the use of AF/AR mechanism in ACQ has slowed the convergence to the optimal utility function. The explanation of this behavior is that AF and AR mechanisms cheat on the real forward traffic by removing some ACKs at the entry of the slow link and regenerating them at the exit. ACQ begins by giving non optimal values for classes weights and after

[1] We have a very small confidence interval, in fact there is 95 per cent of chance that the average utilization is in the confidence interval of [1.621481 - 0.005186, 1.621481 + 0.005186].

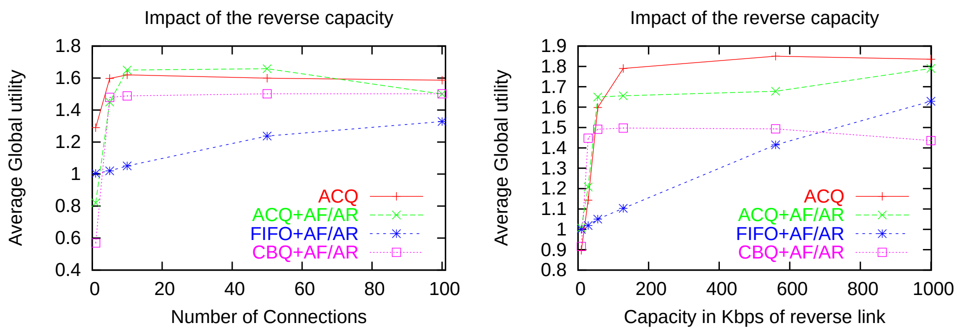

Fig. 6. Impact of the number of connections and the asymmetry degree on ACQ

some intervals T (after 2500 seconds of simulation exactly), it manages to find out the optimal solution. ACQ adds then a protection to Data class against ACK packets when filtering and reconstruction of ACKs are applied. FIFO scheduler gives advantage to ACK Class and hence to the forward traffic, and as seen in Figure 5, does not manage to reach the maximum utility function. We can see that applying ACQ with AF/AR in the ACK queue allows to reach a maximum utility function i.e 1.72 compared to 1.5 for CBQ with AF/AR and 1.28 for FIFO with AF/AR. It just then necessitates longer time to converge.

3.4 Changing some network settings

In this section we show that ACQ is able to adapt to many changes in the traffic, and to stabilize itself quite rapidly, giving then a good value for the utility function. Certainly the amount of scenarios that we can imagine is infinite but we have reported some of them that are the most insightful.

Changing the number of connections In a first step, we change the number of connections involved in the simulations. However we always take the same number at each side in order to keep more fairness between forward and reverse traffic. Obviously varying the number of connections will vary the charge of the network. Similarly to what has been done in Section 3.1, at each simulation scenario, we set the parameters T and γ to the optimal values according to the number of connections. For example, we set T to 20 seconds and γ to 400 $kbps^2$ for the case of 1 connections on each side. Results are reported in Figure 6. The figure shows that as soon as the number of connections exceeds five connections in each direction, the average utility function is stabilized around 1.5. So, when the network is sufficiently charged, ACQ is robust and can adapt to any change in the number of connections.

Changing the degree of asymmetry of the link We set again the number of connections to ten on each side and the values of T and γ to 10s and 50

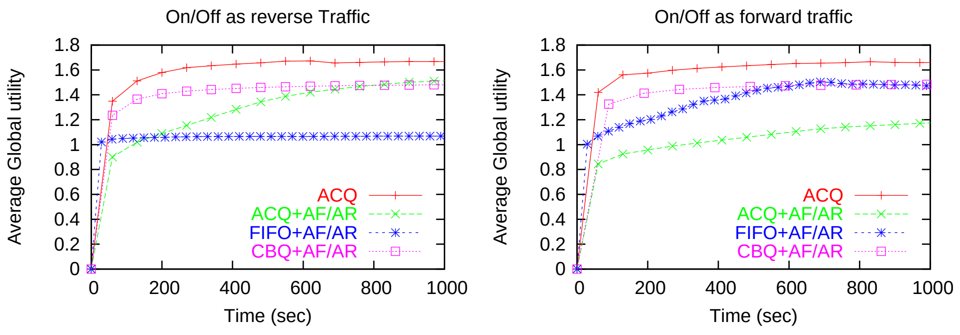

Fig. 7. Short-lived flows as reverse and forward traffic

$kbps^2$ respectively. We change now the capacity of the reverse link from a strong asymmetric case to the symmetric case. Simulation results are plotted on the right hand-side of Figure 6. For a reverse capacity equal to 28.8kbps, ACQ gives a utility function around 1.2. This is an important asymmetric scenario where the reverse link is more than 30 times slower than the forward link. We may expect this scenario to limit the performance of ACQ, but it still behaves very well. We also remark that for large values of the reverse capacity, it is preferable not to use AF/AR mechanism. In fact, when the asymmetry is not very important, the time spent in filtering and reconstructing the ACK packets has worse effect than the asymmetry itself. In this case, ACQ alone gives better results.

Changing the type of TCP connections We change the TCP connections type in our simulations and consider the case where T_r or T_f are On/Off flows with an exponential distribution for the ON and OFF periods. By considering such kind of connections, we aim to emulate short-lived TCP flows where a finite number of packets is transferred by a TCP connection before it stops and transmits again another finite number of packets. The On/Off sources have 0.5 seconds as On and Off average periods, 210 bytes as packet size and 64 Kbps as transmission rate in the ON period. Results are reported in Figure 7. Having On/Off traffic as T_r does not alter the behavior of ACQ, which still gives the best values for the utility function. If we have On/Off traffic in the forward path, ACQ also gives the best results concerning the utility function value. We can then say that ACQ manages to optimally handle the bandwidth sharing between data and ACK packets on the slow reverse link even if the TCP traffic is of the short-lived type.

Involving non-responsive traffic We extend our study to the case of a traffic composed by TCP and other non responsive protocol, like UDP. We have a reverse and forward traffic composed of 5 long-lived TCP traffic (FTP) and 5 constant rate UDP traffic (CBR). Parameters of CBR traffic are: 210 bytes for the packet size and 448 Kbps for the transmission rate. Figure 8 shows the

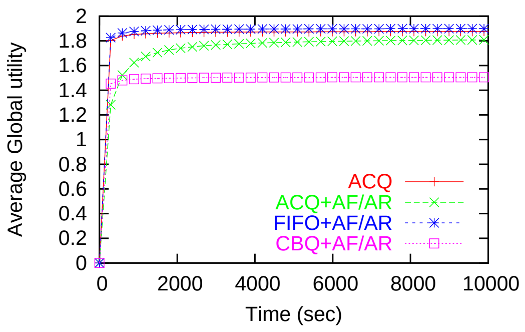

Fig. 8. T_f and T_r are UDP/CBR + TCP/FTP

variations of the value of the utility function. We see in the plot that ACQ and FIFO give a utility function value around 1.8. So, even if we have greedy traffic like CBR over UDP, our scheduler ACQ reacts in a good manner.

4 Conclusion

In this paper, we present a class-based scheduler called ACQ for bandwidth sharing between Data and ACK flows in an asymmetric bandwidth environment. The scheduler uses two queues: one for Data and one for ACK packets and adapts the weights of both queues according to an efficient dynamic method that relies on two parameters T and γ and that uses the gradient projection method. We explain the tradeoff in choosing these two parameters according to traffic and network settings. ACQ allows to reach a good satisfaction of a user generating two-way traffic in asymmetric networks. We compare ACQ with existing mechanisms and we demonstrate by different simulations that ACQ gives the best results. We also test the robustness of ACQ faced to a change in the number of connections involved, in the capacity of the reverse link and in the traffic protocol. Our simulations show indeed that ACQ is a robust scheduling mechanism. In the future, we will be working on the implementation of this mechanism and on its test with real network traffic.

References

1. H. Balakrishnan, V.N. Padmanabhan, and R.H. Katz. "The Effects of Asymmetry on TCP Performance" in Proc. 3rd.
2. T.V. Lakshman, B. Suter. "TCP/IP Performance with Random Loss and Bidirectional Congestion" , IEEE/ACM transactions on networking, Vol8, no5, Oct2000.
3. V. Jacobson. "Congestion avoidance and control", ACM SIGCOMM, Aug1998.

4. T.V. Lakshman, U. Madhow, and B. Suter. “Window-based error recovery and flow control with a slow acknowledgment channel: a study of TCP/IP performance”, IEEE INFOCOM, Apr1997.
5. C. Barakat, and E. Altman. “On ACK Filtering on a Slow Reverse Channel”, proceedings of the first international workshop on Quality of future Internet Services (QofIS), Berlin, Germany, Sept2000.
6. L. Zhang, S. Shenker, and D.D. Clark. “Observations on the dynamics of a congestion control algorithm: The effects of two-way traffic”, In Proc. SIGCOMM ’91 Symposium on Communications Architectures and Protocols, pages 133–147, Zurich, Sept1991.
7. U. Madhow, “Dynamic congestion control and error recovery over a heterogeneous Internet” (invited paper), IEEE CDC, 1997.
8. L. Kalampoukas, A. Varma, and K. K. Ramakrishnan. “Improving TCP throughput on two-way asymmetric links: analysis and solutions”, In Proc. of Sigmetrics, 1998.
9. Ns network simulator, available via http://www-nrg.ee.lbl.gov/ns/
10. H. Balakrishnan, V. N. Padmanabhan, G.Fairhurst, M. Sooriyabandara “TCP Performance Implications of network Path Asymmetry” IETF RFC 3449 Dec2002 .
11. D. Shekhar, H. Qin, S. Kalyanaraman, K. Kidambi, “Performance Optimization of TCP/IP over Asymmetric Wired and Wireless Links,” Invited paper at European Wireless 2002, February 2002.
12. V. Jacobson. “Compressing TCP/IP Headers for Low-Speed Serial Links”, RFC 1144, Feb 1990
13. S Kalyanaraman, D. Shekhar, K. Kidambi “TCP/IP Performance Optimization over ADSL” GI2000.
14. PILC: Performance Implications of Link Characteristics Working Group, URL: http://www.ietf.org/html.charters/pilc-charter.html.
15. S. Kunniyur, R. Srikant “End-to-End Congestion Control Schemes: Utility Functions, Random Losses and ECN Marks” INFOCOM 2000.
16. S. Floyd, V. Jacobson “Link-Sharing and Resource Management Models for Packet Networs” IEEE/ACM Transactions on Networking, Vol3, No4, August 1995.