Homeostatic Congestion Control

Luiz Claudio Schara Magalhães[1], Marcos V. S. Monteiro[1], Ricardo C. Carrano[1,2]

[1]Laboratório MídiaCom, [2]DCT - PURO
Universidade Federal Fluminense
Rua Passo da Pátria 156 – Niterói – RJ – Brazil

schara,mvsmonteiro,carrano@midiacom.uff.br

Abstract. *This paper contains the description and evaluation of the congestion control algorithm for rate-based protocols called Homeostatic Congestion Control (HCC). The name homeostatic comes from the use of two mechanisms with opposing bias to achieve a dynamic equilibrium. HCC uses timing information to infer congestion. Packet pairs are sent periodically to probe the network, which may overestimate the available bandwidth. HCC also measures available bandwidth measuring the variation of the arrival time (jitter) of packets in the sequence of evenly spaced packets given by the rate-based approach. The error in the current rate measured by the jitter is fed back into the control loop. The rate is then balanced between a probing mechanism that raises the rate, and an error feedback, which lowers it. HCC is evaluated not only on its ability to avoid congestion, but also on its convergence to available bandwidth, its stability by itself and in the face of competing traffic, and its fairness.*

1. Background

Congestion control is important for the well-being of the network infrastructure. The earliest version of the Transmission Control Protocol did not have congestion control [Nagle 1984]. Packets were sent in bursts of one flow control window at a time. Although this worked when the network was lightly loaded, it resulted in poor throughput as network usage increased. Lost packets caused retransmissions that in turn further congested the link, generating more lost packets in a vicious circle, until network capacity was completely taken by retransmissions. Congestion control was added to TCP by Jacobson [Jacobson 1988], with variations in the algorithm accounting for the numerous flavors of TCP. Today, new congestion control algorithms must be designed to co-exist with TCP, as it is the transport protocol most widely used in the Internet [Qian 2009]. The concept of maintaining the same throughput that TCP would under the same network conditions is called TCP-friendliness [Floyd 2000].

The main contribution of this paper is the definition of the Homeostatic Congestion Control, which is suitable for rate-based protocols. HCC uses packet pairs and inter-arrival times for bandwidth measurement that set the rate. In contrast, RAP [Rejaie1999] uses additive increase, multiplicative decrease (AIMD) for congestion

This work was made possible with support from FAPERJ and TBE

control. AIMD has been shown to converge [Zhang 2004] and it is a good alternative to HCC, although a secondary constraint of dividing a flow into multiple paths for use in mobile multi-homed hosts led to the use of bandwidth measurements, which can be used for both congestion control and for load-balancing.

The idea of using a pacing algorithm to smooth the burstiness of TCP and for achieving higher throughput has been through a revival since TCP Pacing was found wanting [Wei INFOCOM 2006]. Protocols such as TCP-AP (Adaptive Pacing) [ElRakabawy 2005] [ElRakabawy 2009], tailored for mesh networks, implements rate-based scheduling of transmissions within the TCP congestion window.

The need for achieving higher throughputs due to the introduction of last mile links capable of gigabit speeds, which show the limitation of TCP's current congestion control algorithm, has brought a number of new protocols for high speed links, such as MulTCP [Nabeshima 2005], which uses multiple simultaneous TCP links to achieve high throughput, Cubic-tcp [Ha 2008], which uses a cubic increase function to be able to increase bandwidth use more rapidly, and FAST TCP [Wei 2006]. FAST TCP advocates the use of delay as a congestion measure, which is an approach similar to what is done in HCC and TCP Vegas [Brakmo 1995]. The main argument is that delay has a richer information content than the one-bit given by packet loss. However, HCC does the processing on the receiver end, to get more reliable one-way estimates, and FAST keeps the receiver simple by making congestion estimation at the sender.

All protocols have to deal with the problem of TCP-friendliness. Some do it directly, *e.g.* the TCP Friendly Rate Control Protocol (TFRCP [Handley 2003]) uses the actual analytic formula for TCP throughput to limit the sending rate of the protocol, which can guarantee that it will not get more bandwidth than TCP under similar delay conditions. Others set their behavior to mimic TCP under loss conditions.

This paper begins describing the general problem of congestion control in Section 2, presents the rate-based mechanisms in Section 3, how congestion can be inferred by the algorithm in Section 4, how to adapt to varying network conditions in Section 5, the central idea of Homeostatic Control in Section 6, the Homeostatic Congestion Control algorithm in Section 7, validation in Section 8 and ends with conclusions in Section 9.

2. Bandwidth Estimation, Congestion Detection and Control

The central question of a congestion control algorithm is how to detect if congestion is happening. Due to the reliability of most physical links in the Internet, lost packets can be used as a congestion indication. The receiver can infer the sender is overwhelming the buffer space of a router in the path when a packet is lost. That loss can then be reported back to the sender, which can lower the sending rate to stop the congestion. This heuristic is not devoid of problems. The most obvious is that not all packet drops are caused by congestion, especially when wireless links without link layer reliability are being used. Another problem is the time lag between congestion and prevention. The receiver must notice the loss, which normally entails receiving one or more packets after the loss event. The information must then be reported back to the sender, which requires half an RTT, or the expiration of a timer. An approach to minimize the lag is to use Explicit Congestion Notification (ECN [Ramakrishnan 2001]), in which congested routers warn hosts that packets are being dropped.

Ideally, a sender would know how much data the network is able to transfer, that is, the available bandwidth in a network path; however, without reservations, this is almost impossible to know beforehand. Although TCP's congestion control algorithm converges, it does not measure the available bandwidth, but instead the sum of network bandwidth and routers' queues. Furthermore, it requires straining the network (by causing losses) to find network limits. The elasticity of the routers, designed to accept bursty traffic, is in part responsible for the inaccuracy of the measurement, requiring not only the incoming packet rate to be larger than the outgoing packet rate, but also this situation has to last long enough to fill the router's outgoing queues. The result is that TCP overestimates the path bandwidth, and keeps causing losses even in stable conditions. In response, the challenge is to measure the instantaneous available bandwidth without causing congestion on the network.

The first step in designing congestion control algorithms is finding a good congestion indicator. Packet loss is used by TCP, but end-to-end statistics are noisy and do not have a good correlation to loss type [Biaz 1999], often misclassifying transmission losses as congestion losses. Delay measurements are also noisy, but to improve the correlation between interarrival time and network conditions, a rate-based transmission mechanism can be used. Transmitting packets at regular intervals prevents the burstiness inherent to an ack-clocked design, limiting the amount of variance in the transmission of packets and creating a flow with characteristics better suited to analysis.

3. Rate Based Mechanisms

For rate based transmission schemes, packets are transmitted at regular intervals, as opposed to ack-clocked mechanisms that are inherently bursty because the arrival of an ack may trigger the sending of several back-to-back packets. A flow that uses HCC will be a sequence of constant bit rate (CBR) streams, each with a different rate, with boundaries between streams occurring when the rate changes. Given no competing traffic, if the transmission rate of a rate-based stream is below the capacity of the network path, the time dependencies between any two consecutive packets should stay constant, excluding boundary effects when the rate is changed. Moreover, each packet after the first encounters the same conditions in each router in the path, so the total transmit time between source and destination is constant. On the other hand, when the rate is above the capacity of the bottleneck link, a queue starts forming and the rate seen at the receiver becomes the service rate at the bottleneck.

Therefore, in the case of no intervening traffic, feeding back the rate observed at the receiver to the sender, and adjusting the rate at the sender if it is above the rate observed at the receiver, can avoid congestion. Of course this adjustment mechanism is only needed if the rate is overestimated: if the rate used is below that of the bottleneck link, no change will be perceived at the receiver. One way to find the real bottleneck capacity is to send packets at the sender's maximum rate (back-to-back) and measure their interarrival time. The interarrival time is the dispersion caused by the bottleneck link. If packets are sent at the rate that produces this dispersion, no further delays will be caused at the bottleneck link.

The above idea is the packet pair method [Keshav 1992]. With no competing traffic, the queueing policy at the routers is not important, since it will have no effect on the dynamics of the pair. When there is intervening traffic, fair queueing effectively isolates each flow from the effects of burstiness in other traffic and the packet pair method will measure the share of bandwidth allocated to the flow. On the other hand, if a FIFO policy is used (as is common in many Internet routers), a more complex behavior ensues. Two effects come into play: time compression, when there is a queue in a router and the first packet gets delayed; and time expansion, when one or more packets get between the first and second packets in the packet pair. Both effects can affect the same packet pair in different routers along a path, which generates a large dispersal in the interarrival values measured at the receiver.

4. Congestion Indicators

When using packet pair to measure the bottleneck bandwidth, time compression and time expansion must be filtered out. Unfortunately, as shown by [Dovrolis 2004] the distribution of the interarrival times is multi-modal, and the main mode does not necessarily correspond to the bottleneck bandwidth. This lack of correspondence invalidates statistical filtering of the dispersal values to find the bottleneck capacity [Biaz 1998]. Although it is still noisy, the best way to measure timing information is to use information gathered at the receiver, because it is being affected only by the conditions of the one-way path between sender and receiver, and feed back this information to the sender. This introduces a delay between the measurement and its use, which makes the information only a hint of the current network conditions due to the dynamic nature of the network. However, if a mechanism to correct the effects of the low accuracy is present, the tool will be effective. This is, of course, a limitation of all transport protocols: senders always operate on stale data about the network. Therefore, protocols need a heuristic to deal with the unknown varying network conditions, and a way to correct the failed predictions.

5. Congestion Control Mechanisms

The basic mechanism chosen for bandwidth measurements in HCC is packet dispersion, the time difference between arrivals of two packets. A single instance of packet pair does not give reliable information. It can underestimate or overestimate the fair share of bandwidth. To accurately track available bandwidth, two variants of the packet dispersion algorithms are used: the first is the above cited packet pair method, where two packets are sent back-to-back, and the second is the regular measurements of jitter, the difference between the interarrival times of packets at the receiver and the rate with which they were sent. This is more meaningful due to the rate-based sending mechanism, where packets are sent at a known regular rate. In both cases, the packet dispersion is measured. The first technique has been used extensively to measure the minimum capacity link in a path (and not fair share), but requires many repeated experiments and statistical analysis ([Liu 2004], [Park 2009]). The second technique has also been used for capacity, with similar requirements ([Strauss 2003]).

The objective of the dual mechanisms is to achieve homeostasis, where two opposite forces balance out in a dynamic equilibrium. The operational point which must be achieved, called “fair share”, is such that connections with equal characteristics will get the same share in the critical routers, and connections with different characteristics (e.g.

higher RTT) will not be starved out. The critical routers are those that have low available capacity due to actual low link bandwidth and/or heavy traffic.

The description of the two mechanisms that follows assumes that FIFO queueing is being used at the routers. The first mechanism, packet pair, produces a bandwidth measurement at the receiver. This measurement goes through a weighted average to smooth out transitions and prevent oscillations. A randomizer is used to prevent synchronization, which can be a problem for rate based protocols [Aggarwal 2000]. The resulting value is sent back to the sender and used as the new rate. If the new rate value is below the "fair share", the interarrival times will tend to be dispersed, and positive and negative values of jitter will be received. This happens because the critical router is not working at maximum capacity. Sometimes the packet pair will be queued together, and the resulting value of jitter will be negative as they arrive at a lower interval than the sending rate; sometimes the packets will be separated, and if this time separation is greater than the rate the jitter will be positive. However, if the new rate is above the "fair share", there will be a bias towards positive jitter values, because after the critical router the packets will tend to be spaced farther apart than they were sent. When a threshold of consecutive positive jitter values is seen, it serves as an indication that the current rate has overshot the accepted "fair share" rate. The receiver requests a rate decrease, using the average value of the jitter as the amount of overshoot in the rate.

It is important to notice that the "fair share" concept does not guarantee that protocols with different congestion control mechanisms will get equal throughput. Even among TCP connections, if connections have different RTTs, they will not get the same bandwidth on the links they share [Klein 2004]. Connections with lower RTTs will tend to get a greater share of bandwidth, everything else being the same.

5.1 Adapting to dynamic conditions

As will be shown below, three parameters can be altered to change the behavior of protocols using the proposed mechanism. The first is the parameter of the weighted average used on the rate calculation after a packet pair is received. The weighting factor regulates how much history is kept when a new rate is calculated and can be adjusted so the protocol adapts slower or faster to the current network conditions. While it would be nice to adapt as fast as possible, every rate adjustment has influence on all traffic on that path, including the adapting flow. If the value overshoots the desired operational point, it is possible that the next adjustment will try to correct this, possibly creating oscillations. The second parameter is in the jitter monitoring mechanism. If the rate overshoots the operational point, this mechanism notices positive jitters, and adapts the rate. A value proportional to the average jitter is used to correct the rate. The threshold (how many consecutive positive jitters) can be adjusted, and how much of the jitter is used to correct the rate can be adjusted so that the protocol backs-off faster or slower, and more or less in case of errors in the "fair share" rate calculation. The third parameter is the amount of back off due to congestion, defining how much the rate has to be decreased in case of packet losses. TCP uses a multiplicative decrease, halving its window in case of losses. A similar approach is used, and the rate is halved in case of losses. Although halving the window and halving the rate are not exactly equal, the effect is the same, namely, to back off enough so the queues that have been building up in the network have a chance to clear.

6. Homeostatic Control

In this section the details of the congestion control heuristics are given. The name homeostatic control comes from the two mechanisms used for achieving balance. The heuristic tries to keep balance by overestimating the available bandwidth (due to the bias of the packet pair to measure link capacity and not available bandwidth), and correcting the estimate by slowing down the rate using jitter averages.

6.1 Rate based sending mechanism and jitter monitoring

Using a rate-based sending mechanism means that packets are sent at a certain *rate*, or conversely, that packets are sent with a certain *period*. Due to flow conservation, the number of packets that enter the network must be equal to the number of packets that leave the network. Packets may leave the network either by arriving at the destination or by being dropped at one of the routers in the path. In a best effort network, packets are only dropped because of errors in the header field (caused for example by bit errors in transmission) or because there are no buffers available at the router output queue.

The existence of buffer queues in the routers means that it is possible to violate the maximum service rate for a limited time without packet loss. In fact, this is the common case for TCP. Large bursts in the network can cause packet loss even if the average packet rate is below the service rate of all routers in the path, by causing temporary buffer overflows. If evenly spaced packets are sent, this possibility can be ruled out.

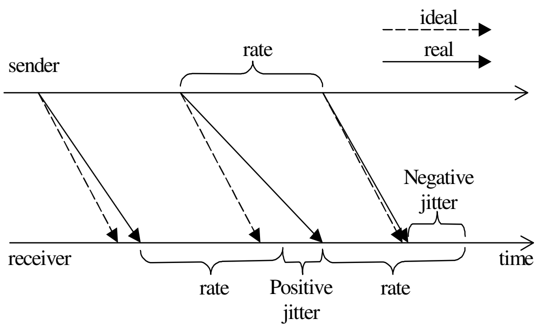


Figure 1: positive and negative jitter

If the transit times of all packets were the same, they should keep the same timing relation with which they were sent. Unfortunately, due to the presence of cross traffic, even packets sent at a rate below the maximum service rate of a path will have different transit times (packets sent above the maximum service rate will certainly be delayed, and if the rate is kept for long enough, some of them will be lost). Let jitter be defined as the difference between the interarrival time of two packets and the period with which they were sent. Using this definition, packets can experience both positive and negative jitter. Negative jitter occurs when the first packet of a pair is delayed, but the second packet is not, positive jitter occurs when the first packet is delayed less than the second (see Figure 1). Positive and negative jitters should alternate. When a series of positive jitters is noticed, the network service rate is being violated. When using homeostatic control, two consecutive positive jitters are taken as an indication of rate violation, and the period is corrected, given the conditions observed by those two packets.

Jitter monitoring prevents the period from straying from the network operational point. Unfortunately, this is not enough to guarantee that the network will operate with no losses. It is possible that small differences will build up eventually, even when positive and negative jitters are perfectly intermingled. The correction mechanism should decrease the rate to a point below the operational point, allowing any queues that have formed to shrink. This is also a good place to add some randomness. If different rate-base flows become synchronized, they will increase and decrease their rates in tandem. While this may increase network utilization for a short time, if all flows violate the network service rate at the same time, they may all experience simultaneous losses. If those losses are coupled with a backoff mechanism, then all flows will backoff at the same time, leading to a large waste in bandwidth [Klein 2004]. By varying the amount of rate decrease using a random function, different flows avoid becoming synchronized, at the cost of never achieving full link bandwidth.

7. Algorithm

The HCC algorithm consists of three phases: exponential increase, congestion avoidance and congestion control. The objective of the exponential increase phase is to converge quickly to the available bandwidth. Packet pairs are sent once every 5 packets. Let P_0 be the initial period estimated by sending one packet pair and using twice their interarrival time as its value. The period in which packets are sent is then adjusted for each measurement according to

$$\mathbf{P_{n+1} = (1 - \alpha) * P_n + \alpha * P_{Measured}, \alpha \in [0,1].}$$

The error, or difference between the optimal period and the current period is given by

$$\mathbf{Error = ((1 - \alpha)^n) * (P_0\text{-}P_{Optimal})}$$

if the bandwidth is stable. Thus, the error decreases exponentially for a static scenario.

The congestion avoidance phase is signaled by a sequence of positive jitters, showing that the network is getting loaded, and queues are increasing. In that phase, the period is corrected by the error between the current period and the optimal period. The magnitude of the error is given by the sum of the jitters, because the jitter is caused by the difference between what was measured and what packets experienced while in the network, yielding a new period $\mathbf{P_{n+1}= P_n + (jitter_1+...+jitter_n)/n}$, where n is the number of measurements and $\mathbf{jitter_n}$ is the difference between the rate the packets were sent and the actual inter-arrival time of packets..

In the congestion control phase, triggered by congestion related losses, the protocol starts multiplicative decrease, doubling the period every RTT if:

current_time > time_last_loss + RTT + 2* $\mathbf{P_n}$.

Let $\mathbf{P_c}$ be the current period, $\mathbf{P_{Measured}}$ the measured period and $\mathbf{P_{Optimal}}$ the true optimal period. We have 3 cases:

Too small, $\mathbf{P_{Measured} < (P_{Optimal} - (1-\alpha) * P_c) / \alpha;}$

Sweet spot, $\mathbf{P_{Optimal} > P_{Measured} > (P_{Optimal} - (1-\alpha) * P_c) / \alpha;}$

Too large, $\mathbf{P_{Measured} > P_{Optimal}}$

If the period is too small, the rate will be too high, and may cause congestion. If the period is too large, the protocol will not achieve maximum throughput. If the period falls in the sweet spot, the new period will be close to the one that will result in using all the available bandwidth of the link. The homeostatic nature of the congestion control acts to correct the errors in measurement. Because there is a bias to overestimate available bandwidth, normally the measurements will vary between optimal and too small. Undersized measurements will generate positive jitters, and jitter correction will come into play, slowing down the rate. Oversized measurements are rarer, and only temporarily slow down the rate, until the next measurement. The jitter can be only as large as the sum of the time a packet can expend at the router queues in a path, giving a bound on its maximum absolute value.

8. Simulations

In this section, the results of ns2 simulations are presented. These simulations use the HCC-protocol module called XMTP, and were designed to test six aspects of HCC: convergence to link bandwidth, convergence to available bandwidth, fairness, stability, TCP friendliness, and performance. Previous work ([Magalhaes ICNP 2001], [Magalhaes 2001]) presents the description and implementation of a HCC in real world environments. The protocols were constrained to low bandwidth links due to low timer granularity, such as the then current 2 Mbps wireless links. Recently, timer granularity has improved significantly. The current “tickless” design of the Linux kernel allows for workable protocols on high bandwidth links, and research has begun on this direction.

8.1 Convergence to Bottleneck Bandwidth

This experiment consists of a single flow with no cross traffic going through a bottleneck link. The delay of the bottleneck link is changed with each run. The results shown are for a single run for each delay. The number of packets that arrive at destination in each second is recorded and compared to a TCP New Reno flow.

TCP New Reno has problems with high delays due to its inherent burstiness. Therefore, for a certain bandwidth/delay product, if the queues at the routers are not large enough, TCP New Reno cannot achieve the path throughput. At higher RTTs TCP cannot achieve the maximum throughput of 124 packets per second, even though the only variable changed in each run was the link delay on the bottleneck link. Although scaling the queues to the bandwidth delay product of the path would allow TCP to achieve link bandwidth for all delays, this was not done to illustrate that XMTP does not heavily load the routers in the path due to its rate-based mechanism.

During the handshake phase of XMTP, an estimate of the bottleneck bandwidth is acquired by using the packet pair method. To be conservative, the value obtained is doubled, so the initial value of the rate should be below that of the bottleneck link, allowing room for refinement as more information is gathered. In this experiment, because there is no cross traffic, the value obtained should accurately measure the bottleneck bandwidth. Because the value is doubled, the rate will be half of the maximum rate available and each subsequent measurement should increase the rate, until it converges to the available bandwidth. The rate of increase is directly connected to the amount of history kept in each iteration of the probing packet pair.

Because XMTP is rate-based, it does not suffer from the problem of overflowing queues that prevent TCP from achieving the maximum throughput possible. Although XMTP does not achieve the maximum available bandwidth of 124 packets per second, the only influence on the link delay is the ramp-up time. Flows with different bottleneck delays still achieve the same throughput. The effect of randomness can be seen in Figure 2 as the throughput fluctuation during "steady-state." This random variance was introduced to avoid synchronization.

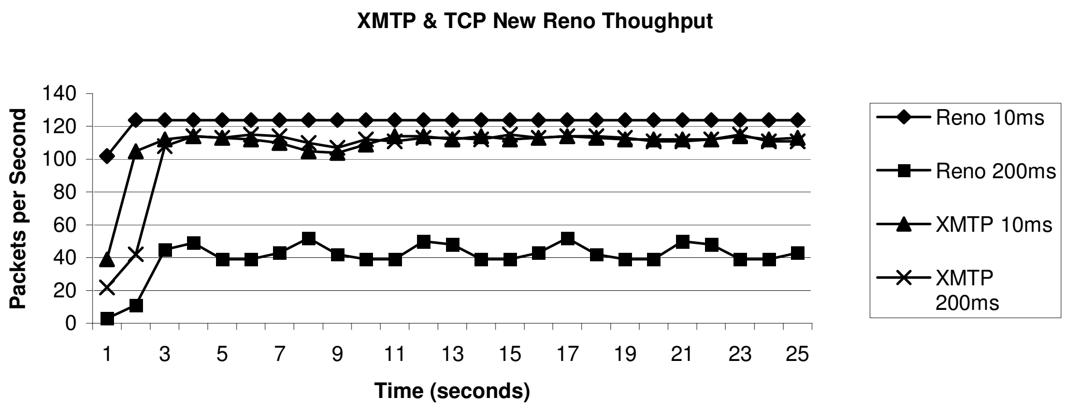


Figure 2: XMTP and TCP New Reno Throughput

Figure 2 shows results for XMTP and TCP New Reno for the lowest and highest values of RTT. XMTP is not dependant on router queues to compensate for different bandwidth/delay products, so it can achieve maximum bandwidth with any delay, only limited by its synchronization avoidance mechanism to stay below the link maximum. XMTP does not achieve the maximum bandwidth by design. If data were transmitted at the maximum available bandwidth, it would be impossible for queues forming at routers to drain, because there would be no bandwidth left over. Although counter-intuitive, this leads to greater link utilization under most scenarios by avoiding congestion losses.

8.2 Convergence to available bandwidth

A non-responsive flow by definition will not change its rate due to congestion. Multimedia flows are often non-responsive either because they lack the mechanism to measure network conditions, or because they cannot change the source encoding rate. In this case, most applications assume that it is better to suffer some losses than to fail in transmission, especially because most multimedia streams will degrade gracefully in the presence of a small percentage of losses. This set of experiments depicts the reaction of an established flow (converged to the bottleneck bandwidth) to a non-responsive flow.

Under the same scenario, The unresponsive flow uses 1.5Mbps of the 2Mbps bottleneck link. The link delay on the bottleneck link is changed and throughput is measured. The ideal responsive flow should show an inverted square wave. The objective is to measure how long it takes a flow to decrease its throughput, which minimizes losses, and how long it takes to regain its previous throughput once the non-responsive flow is gone.

XMTP reacts as expected; dropping its throughput in response to the CBR flow and returning to the same level after the flow is gone. It takes two seconds for XMTP to drop to the new available bandwidth, and two to three seconds to return to the initial level once the unresponsive flow is gone, depending on the bottleneck link delay. In Figure 3 XMTP and TCP are compared, and it is shown that they have similar behavior. TCP is faster than XMTP to regain bandwidth once the CBR flow stops, but XMTP is less affected by differences in path delay.

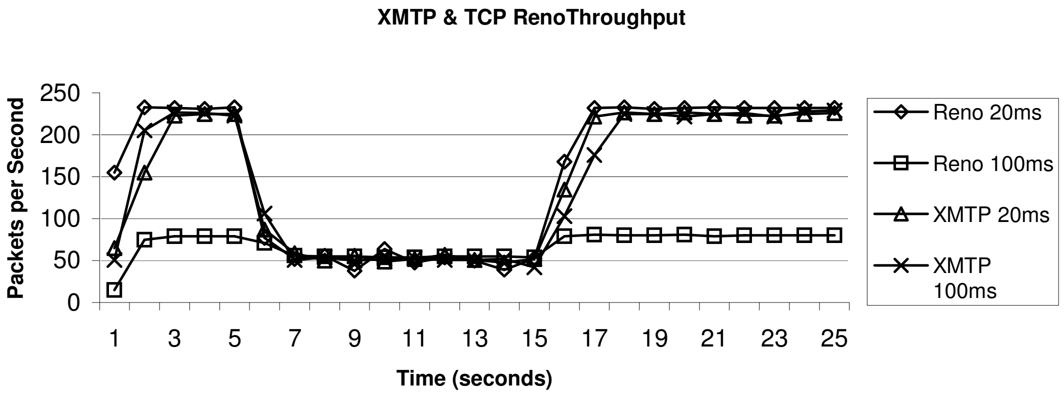


Figure 3: TCP and XMTP response to a CBR Flow

8.3 Fairness and Bandwidth sharing

A congestion control algorithm should be fair in the sense that when multiple flows are present, each gets an equal share of bandwidth. This is hard, and even TCP will not be fair if conditions are not the same for all flows. If flows with larger RTTs are sharing the same link, they tend to get a smaller share because they take longer to recover from losses. In this experiment, simulations are run with five flows, changing the mix from 5 TCP flows to 5 XMTP flows. The total number of packets that was received for each flow is recorded. Then the usage, or total number of packets is calculated, along with the fairness. To calculate fairness, we use Jain's fairness index:$(\sum_i^n x_i)^2 / n(\sum_i^n x_i^2)$. Where n is the number of flows and xi is the number of packets for each flow.

Table 1: Fairness for XMTP & TCP flows

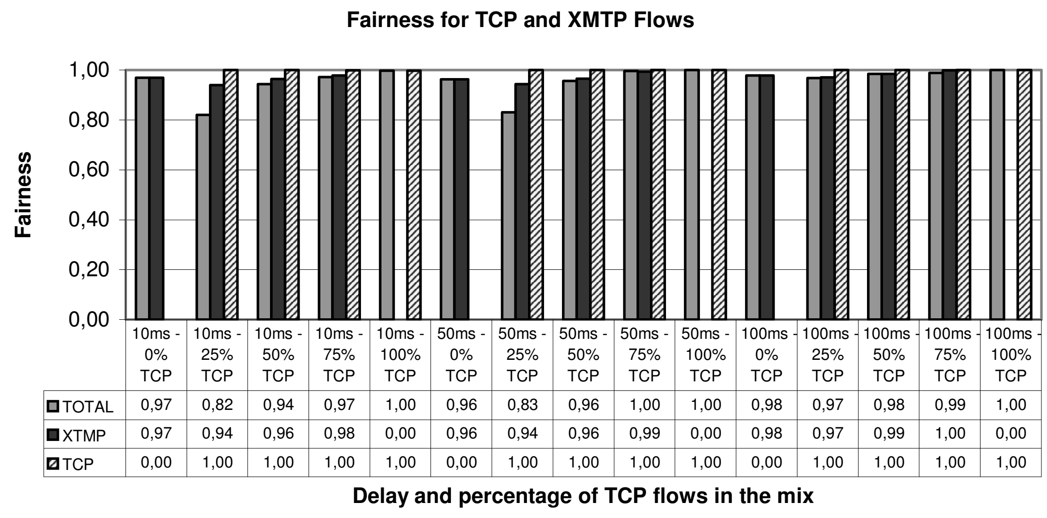
	Flow 1	Flow 2	Flow 3	Flow 4	Flow 5	Usage	Fairness
5 TCP	6274	6188	6188	6100	6193	30943	0.999
1XMTP/4TCP	2930	7137	7137	6858	6866	30928	0.935
2XMTP/3TCP	5480	6541	6541	5998	5935	30495	0.996
3XMTP/2TCP	5384	5473	5473	6265	5884	28479	0.997
4XMTP/1TCP	2204	5712	5712	6831	6192	26651	0.916
5 XMTP	4277	5238	5238	5682	5471	25906	0.990

The experiment shows XMTP's behavior in presence of reactive traffic. The questions are if XMTP flows will be stable, each flow converging to a fair share of bandwidth without oscillations, and if new flows will be able to get bandwidth from established flows. This is compared with the performance of TCP flows under the same conditions.

A scenario similar to the previous runs was used with five flows, each with its own starting and ending node, and shared a bottleneck link. The bandwidth on each link is 10Mbps, (2Mbps per flow). The delay on the bottleneck link was changed on each run.

Table 1 shows the results obtained for a 50ms delay on the bottleneck link. Each line represents one experiment with darker cells indicating TCP flows (*i.e.*,. line 1 depicts an experiment with 5 TCP flows). For the test with only TCP the fairness is almost 1. In line 2, the single XMTP flow is less aggressive than TCP and therefore captures a smaller portion of the bandwidth. This demonstrates that XMTP is not interfering with TCP, and although it is able to use only half of the bandwidth of the other flows, it is not completely stopped by TCP. A better result is obtained on lines 3 and 4, where TCP and XMTP share the link equally. In line 5, because one flow did not perform well, both usage and fairness dropped, but no flow was starved, and TCP's performance is not hampered. In the case with only XMTP flows, each flow is near 17% below the maximum (by design), and though there are differences in the bandwidth experienced by each flow, the fairness is still high. In general, TCP performs better at smaller RTTs, while at larger RTTs XMTP tends to outperform TCP. In both cases neither is starved.

Table 2: Fairness for TCP and XMTP flows for different bottleneck delays



	10ms - 0% TCP	10ms - 25% TCP	10ms - 50% TCP	10ms - 75% TCP	10ms - 100% TCP	50ms - 0% TCP	50ms - 25% TCP	50ms - 50% TCP	50ms - 75% TCP	50ms - 100% TCP	100ms - 0% TCP	100ms - 25% TCP	100ms - 50% TCP	100ms - 75% TCP	100ms - 100% TCP
▦ TOTAL	0,97	0,82	0,94	0,97	1,00	0,96	0,83	0,96	1,00	1,00	0,98	0,97	0,98	0,99	1,00
■ XTMP	0,97	0,94	0,96	0,98	0,00	0,96	0,94	0,96	0,99	0,00	0,98	0,97	0,99	1,00	0,00
▨ TCP	0,00	1,00	1,00	1,00	1,00	0,00	1,00	1,00	1,00	1,00	0,00	1,00	1,00	1,00	1,00

8.4 Stability

This experiment was designed to test the stability of the congestion control mechanism when multiple flows are sharing the same link. A dumbbell-shaped scenario was created, where 100 nodes are connected to a single node though links with 1Mbps throughput and 10ms delay each. This node is connected to another using a single link with a 50Mbps bandwidth (half the aggregated bandwidth) and variable delay. Finally, the last node has another 100 nodes connected to it. Each node in one extreme is paired with another in the other extreme, and a reliable connection, using either TCP New Reno or XMTP is established. The simulation is run for 25 seconds, and the number of packets received is recorded. This number is then input into Jain's Fairness Index.

This simulation is parameterized so the following can be varied: the delay of the bottleneck link, how many of the hundred flows is TCP or XMTP and the duration of the run. Delays of 10ms, 50ms and 100ms were chosen for the bottleneck link, and the mix of TCP and XMTP was varied in five steps: 0/100, 25/75, 50/50, 75/25 and 100/0.

These results are summarized in Table 2. It can be observed that TCP flows are very fair to each other, as expected, and XMTP flows are also very fair, because fairness never drops below 0.9. But the combined TCP/XMTP flows are not as fair, although the fairness never drops below 0.8. Under the conditions chosen, TCP gets more bandwidth.

8.5 TCP Friendliness

Fairness is a very close concept to "friendliness". Both measure the concept of a network "good neighbor". In the last set of experiments, the objective is to investigate how TCP and XMTP interact. Two flows, one XMTP and one TCP share the same path, and the throughput in packets per second is measured.

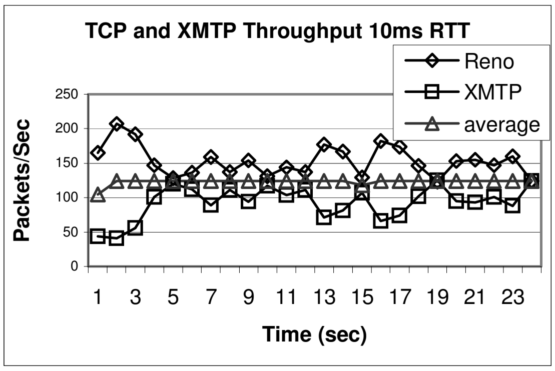


Figure 4: TCP and XMTP Throughput (10ms delay)

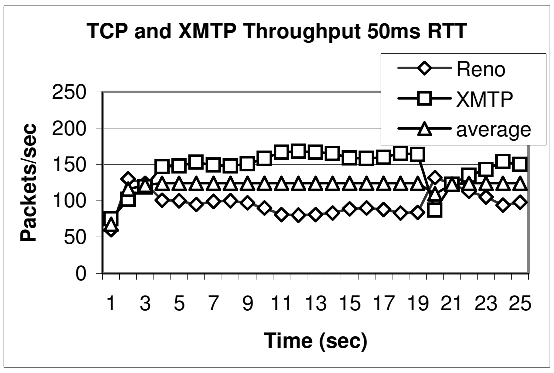


Figure 5: TCP and XMTP Throughput (50ms delay)

Figure 4, shows how both flows share a 10ms delay link. With low delay, TCP is more aggressive, and tends to get more bandwidth than XMTP. The converse is true for larger RTTs. With a 50 ms delay, Figure 5 shows that XMTP dominates. Nevertheless, in both cases neither TCP nor XMTP starve out the other. Even with larger delays (not shown), when TCP cannot achieve full link bandwidth, XMTP does not starve it out.

9. Conclusions and Future Research

This paper presented the Homeostatic Congestion Control algorithm, which is suitable for congestion control of rate-based transport protocols. HCC is a congestion avoidance algorithm, because it can react to changing network conditions before congestion takes place, by measuring available bandwidth and dropping the protocol transmission rate if it exceeds the perceived available bandwidth of the network. HCC measures available bandwidth using two strategies: packet pair and jitter correction. To measure increases in available bandwidth, it uses the packet-pair method. This allows it to converge faster to available bandwidth than an AIMD algorithm, but may lead to an overestimation of available bandwidth. To measure decreases in available bandwidth, and to compensate for the overestimation of available bandwidth by the packet-pair measurement, HCC corrects the rate by using the error information, which is the difference between the expected rate and the inter-arrival times of the packets. By adding this quantity (jitter) to the rate, both congestion avoidance and rate correction are achieved. HCC has an additional mechanism if congestion avoidance is not enough and losses still occur. If losses are detected, the rate is halved once for each RTT that contains a loss event. Finally, HCC uses a novel strategy to avoid synchronized losses, which can affect rate-based protocols. A randomizer is used to add up to 10% to the value calculated for the rate. Because the value is random, different flows will use different values for the rate, which will fluctuate slightly, preventing synchronization. The performance of HCC was tested using a protocol (XMTP) implemented in ns2. XMTP was shown to converge to link bandwidth and respond well to both CBR and TCP flows, regaining bandwidth when competing flows are shut off. XMTP was shown to be stable and TCP friendly.

HCC can be augmented with a loss discrimination heuristic, which was shown in another paper [Magalhaes 2003]. One of the weaknesses of HCC is its need of good timers. User space implementations of HCC, (described in [Magalhaes ICNP 2001] and [Magalhaes 2001]) are limited to the timer resolution offered to user programs. Current high resolution timers provide milliseconds resolution [Vasudevan 2009], and enable higher throughputs than possible at the time, which motivates further investigation. Additionally, HCC could be improved by adapting the probing interval to the path RTT.

References

A. Aggarwal, S. Savage, and T. Anderson (2000) “Understanding the performance of TCP pacing”. In IEEE INFOCOM, pages 1157--1165, Tel-Aviv, Israel

S. Biaz, N. Vaidya (1998) "Distinguishing congestion losses from wireless transmission losses: a negative result," in Proc. 7th Int. Conf. Comp. Com. and Net.

S. Biaz, N. H. Vaidya (1999) "Discriminating Congestion Losses from Wireless Losses using Inter-Arrival Times at the Receiver," IEEE Symposium ASSET'99.

L. Brakmo, L. Peterson (1995) “TCP Vegas: end-to-end congestion avoidance on a global internet,” IEEE J. Sel. Area Comm., vol. 13, no. 8, pp. 1465–1480.

W. Cleveland, D. Sun, “Internet Traffic Data” (2000) Journal of the American Statistical Association, 95, 979-985.

C. Dovrolis, P. Ramanathan, D. Moore (2004) "Packet Dispersion Techniques and a Capacity Estimation Methodology", IEEE/ACM Transactions on Net 12(6):963–977.

S. ElRakabawy, A.Klemm, C. Lindemann (2005) “TCP with adaptive pacing for multihop wireless networks”, Proc. ACM MobiHoc, Urbana-Champaign, IL,USA.

S. ElRakabawy, C. Lindemann (2009) “Practical Rate-based Congestion Control for Wireless Mesh Networks”. http://rvs.informatik.uni-leipzig.de/de/publikationen/.

S. Floyd, M. Handley, J. Padhye, J. Widmer (2000) "Equation-Based Congestion Control for Unicast Applications", ACM SIGCOMM 2000, Stockholm.

S. Ha, I. Rhee, L. Xu (2008) CUBIC: a new TCP-friendly high-speed TCP variant. SIGOPS Oper. Syst. Rev. 42, 5 (Jul. 2008), 64-74.

M. Handley, S. Floyd, J. Padhye, J. Widmer (2003) "TCP Friendly Rate Control (TFRC): Protocol Specification", RFC 3448.

V. Jacobson (1988) “Congestion avoidance and control”, In Proceedings of the SIGCOMM ’88 Symposium, pages 314-329.

J. Ke and C. Williamson (2000) "Towards a Rate-Based TCP Protocol for the Web", Proc. of 8th Int. Symp. on Modeling, Analysis and Sim. of Comp. and Telecom. Syst.

S. Keshav (1992) "A Control-Theoretic Approach to Flow Control," presented at Proceedings of the SIGCOMM '92 Symposium.

T. Klein, K. Leung, H. Zheng (2004) "Enhanced Scheduling Algorithms for Improved TCP Performance in Wireless IP Networks", IEEE Globecom.

X. Liu, K. Ravindran, B. Liu, and D. Loguinov(2004) "Single-Hop Probing Asymptotics in Available Bandwidth Estimation: A Sample-Path Analysis, " ACM IMC.

L. Magalhaes, R. Kravets (2001) “MMTP: Multimedia Multiplexing Transport Protocol”. in SIGCOMM-LA 2001, San Jose, Costa Rica.ACM, v. 31. p. 220-243.

L. Magalhaes, R. Kravets, A. Harris (2003) “Improving Performance of Rate-Based Transport Protocols in Wireless Environments”, In: XX SBrT, Rio de Janeiro.

L. Magalhaes, R. Kravets (2001) “Transport Level Mechanisms for Bandwidth Aggregation on Mobile Hosts”. In: ICNP 2001, Riverside, CA.

M. Nabeshima (2005) “Performance Evaluation of MulTCP in High-Speed Wide Area Networks”, IEICE Trans. Commun. E88-B, Nº 1.

J. Nagle (1984) "Congestion Control in IP/TCP Internetworks", RFC 896.

K. Park, H. Lim, J. Hou, C. Choi (2009) "Feedback-assisted robust estimation of available bandwidth": The Int. J. of Comp. and Telecom. Net, v.53 n.7, p.896-912.

F. Qian, A. Gerber, Z. Mao, S. Sen, ,O. Spatscheck, W. Willinger. “TCP Revisited: A Fresh Look at TCP in the Wild”, ACM SIGCOMM, pages 76-89, Chicago, IL, 2009

K. Ramakrishnan, S. Floyd, D. Black (2001) "The Addition of Explicit Congestion Notification (ECN) to IP", Request for Comments: 3168.

R. Rejaie, M. Handely, D. Estrin (1999) "RAP: An End-to-end Rate-based Congestion Control Mechanism for Realtime Streams in the Internet", Proc. IEEE Infocom'99.

J. Strauss, D. Katabi, F. Kaashoek (2003) "A measurement study of available bandwidth estimation tools", In 3rd ACM SIGCOMM, Miami Beach, FL pages 39-44.

V. Vasudevan, et al (2009) “Safe and Effective Fine-grained TCP Retransmissions for Datacenter Communication”, ACM SIGCOMM, pages 303-314.

D. Wei, P. Cao, and S. Low (2006) “TCP Pacing Revisited,” in IEEE INFOCOM.

D. Wei, C. Jin, S. Low, S. Hegde (2006) "FAST TCP: Motivation, Architecture, Algorithms, Performance" IEEE/ACMTransactionson Net,vol.14,no.6,pp.1246-1259.

Y. Zhang , S. Kang , D. Loguinov (2004) “Delayed stability and performance of distributed congestion control”, ACM SIGCOMM Comp. Com. Review,v.34 n.4.