Kernel Mechanism for End-to-End Traffic Shaping

Felipe Maya, Luiz Magalhães
Mestrado em Engenharia de Telecomunicações – Universidade Federal Fluminense (UFF)
Rua Passo da Pátria 156, Sala 421 Bloco E, Niterói, RJ
{fmay, schara}@midiacom.uff.br

***Abstract*- This work is an investigation on how end-to-end traffic shaping affects communication. While studying rate-based congestion control, we developed a Linux module capable of measuring available bandwidth between two end points, and of pacing packets according to the available bandwidth. The module can be used for congestion control of non congestion-controlled protocols, such as UDP. This paper describes the module and preliminary results.**

I. Introduction

The transmission of data in networks where bandwidth is dynamically allocated, e.g. the Internet, can result in an unstable behavior when there is significant traffic, leading to overflowing queues on routers and to packet loss. The conventional FIFO policy on routers will discard packets that arrive last when buffers are full. On the other hand, if it were possible to obtain information on measurements such as bandwidth, link capacity and network usage, it would be possible to use the Internet in a more efficient manner, by dynamically controlling flows at the source. This would be a form of congestion avoidance, where packets violating the measured bandwidth would be dropped at the source, not wasting network bandwidth needlessly on the path to the choke point.

This work describes a Linux module that measures available bandwidth in a path [6], and sends packets at regular intervals, with the constraint of timer granularity, to match the available end-to-end bandwidth, joining the concepts of congestion control and pacing in a traffic shaping module.

II. Flow Control Mechanisms

The proposed mechanism for congestion control (or traffic shaping) is composed of two separate phases. The first is the active probing [5], where probe packets are sent in the network to gather delay measurements [3], and the second, the rate change phase, is charged with producing the estimates to control data transmission.

A. Active Probing

The calculation of the available bandwidth on the traffic shaping module is based on a series of techniques that actively measure the number of bits that can be sent in an end-to-end path for a determined time period.

The technique used for active probing is the Train of Packet Pairs (ToPP) [7], which is a friendly (non-aggressive) measurement technique to estimate the available bandwidth in a congested link. It sends a sequence of ICMP [8], [9] packet pairs of the type Timestamp and Timestamp Reply Message (according to RFC 792 [9]), types 13 and 14 [4], where the header carries the time when the packet left the source, the time when the packet arrived at the destination and the time the packet left the destination. Each timestamp corresponds to the time in milliseconds since midnight, used as the epoch. Besides these fields, information using microsecond accuracy is added by our module if present, because it is needed for fine grained measurements (for use with high speed links).

The ToPP sequence used in this traffic shaping module is composed of a series of packet pairs with different time spacing, as shown in (figure 1)

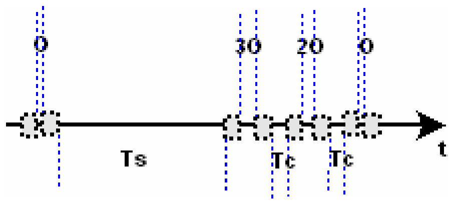

Figure 1. ToPP Sequence

B. Variation on the Transmission Rate

When the timestamps from the ToPP sequence are received, the transmission rate is estimated based on calculations of error on the predicted delay [2]. When the rate is estimated, the module identifies which queue is associated with that end-to-end path, and uses this as the pacing rate [1] for that queue. The traffic shaping uses a dequeue function, which examines each queue in turn, looking for packets that should be sent at the current time, as shown in (figure 2).

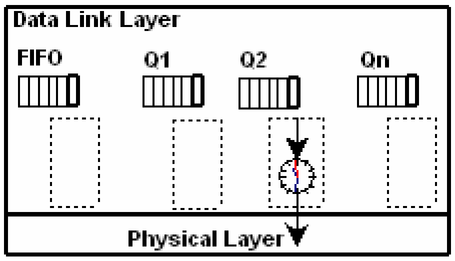

Figure 2. Dequeueing Mechanism

The first queue uses a conventional FIFO, and packets from that queue will be chosen whenever a packet is available. The inspection continues on each queue, selecting packet which are ready to be sent.

There is a dependency between the timer granularity of the system and the rate that packets will be dequeued. The queues are examined at each clock tick, so all rates will be divisors of the basic (maximum) rate, which is to send one packet every clock tick.

For each queue, the maximum bandwidth reachable can vary dynamically depending on the estimates of the active probes, and will follow equation (1), where k is the number of interrupts estimated to give the required rate, HZ is the interrupt frequency and P the size of the packet:

$$\text{MaxRate} = P^* (k * HZ). \tag{1}$$

When k equals 1, the system is working at its maximum rate (which may be below the actual rate in the network), and sends a packet at each interrupt.

III. Experimental Results

Our experiments were carried out in the network shown on figure 3. It consists of two end-systems connected to a 100 Mbps Ethernet network and two routers with a band-limited 28Kbps link (burst 4750 bytes, latency 50 ms). This was achieved using the command *traffic control* (tc), with the *Token Bucket Filter* (tbf). The command line to create the filter is given on the line below:

```
tc qdisc add dev eth0 tbf burst 4750 rate 28kbit latency 5ms
```

The end systems run Linux with kernel 2.6.17 compiled with the variable HZ set to HZ_100, which results in a clock tick of 1ms (timer interrupts occur at 1ms, which gives the minimum period between two events).

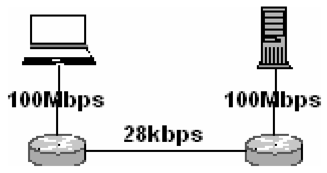

Figure 3. Test Network.

It should be pointed out that the timer granularity is extremely important in these experiments and to rate-based mechanisms in general. The timer granularity sets the minimum interval between the transmission of two packets, and also the maximum rate attainable by a protocol, or in this case, a traffic shaper, that can send only a packet at each timer interrupt. Modern systems allow timer granularity of 1ms (down from 20 ms which was the default in earlier kernels), but for better performance hardware assistance may be necessary.

To test our mechanism we used a modified client/server application TFTP (Trivial File Transfer Protocol), which uses UDP and no congestion control, and can send bursts because the stop and wait mechanism was disabled.

Table 1 shows the results of four runs of the test. The first set has an average of 24,6Kbps, and the second 25,6Kbps. It is a 4% increase in throughput, even though probe packets are being sent to measure available bandwidth.

TABLE I
THROUGHPUT ACHIEVED WITH AND WITHOUT TRAFFIC SHAPING

	Without the module:		With the module:	
Run	time (sec)	rate (bps)	time (sec)	rate (bps)
1	417.2	24730	402.3	25648
2	420.6	24528	404.5	25504
3	415	24858	400.1	25785
4	427.8	24116	406.2	25398
average	420.15	24558	403.28	25583.75

In the second set of tests, we changed the network test-bed for those seen on Figure 4. The second machine sends a competing flow to introduce cross-traffic and make the bandwidth measurement harder.

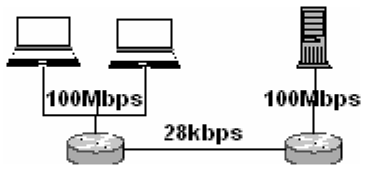

Figure 4. Test Network.

To acquire data, the code for tbf was altered to print a packet count each five seconds, what will allow the measurement of the number of discarded packets. Moreover, the FIFO code was altered to print the packet count at the cross traffic source.

After 10 runs using the modified TFTP server and a TCP cross traffic, we had the following results: 5349 packets were sent, and 3598 packets were received in average (67% arrival rate). It should be pointed out that on the runs without the traffic shaping module, with the lower bandwidth link congested by the TCP flow, in average only 7 packets arrived (less than 1% arrival rate). Figure 5 shows the average of the 10 runs with packets sent being the higher line, and packets received the lower line. At each interval, the difference between those lines is the number of packet lost.

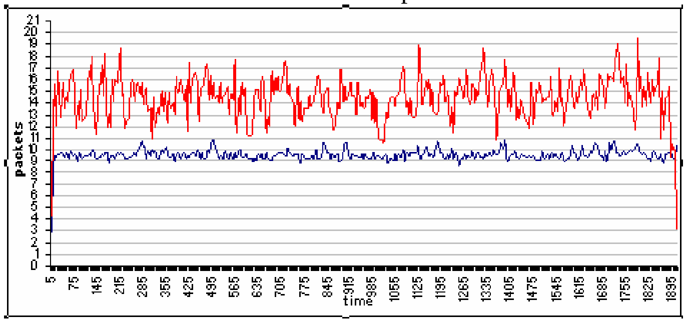

Figure 5. UDP packets sent and received with module.

The average time to send 2736340 bytes was 1811 seconds. In a sense, because TFTP was changed to send packets without waiting for confirmation, all file transfers, with or without the module, failed. The file used for testing never arrived whole. But the objective was to have a non-congestion-controlled source, and to this intent the test was successful. With traffic shaping the number of packets discarded by the routers were much lower than without it. In Figure 6 the number of packets discarded is shown in the lower line, and the number of packets transmitted in the higher line. This periodic behavior can be explained by the TCP cross traffic, which tends to use more and more bandwidth, chocking the UDP paced traffic.

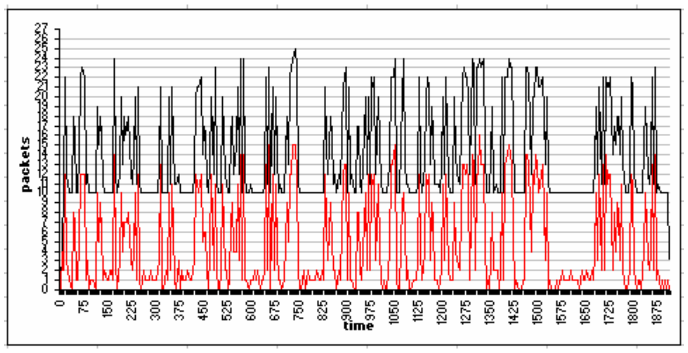

Figure 6. UDP packets sent and lost with module.

It was seen with TCP Vegas [10], that congestion avoidance has problems when used with traffic that reacts to congestion. The traffic with congestion avoidance tends to give up bandwidth to the other traffic, because it reacts to impending congestion, while the other traffic only reacts after congestion happens. While we intend to investigate further, the current algorithm sets a maximum interval (in this case 50 HZ) which will is the cut-off value to the delay imposed on each packet. This can be seen on Figure 7, which compares the TCP cross traffic (higher line) with the congestion controlled UDP traffic (lower line). TCP forces the UDP traffic to operate most of the time at the cut-off value.

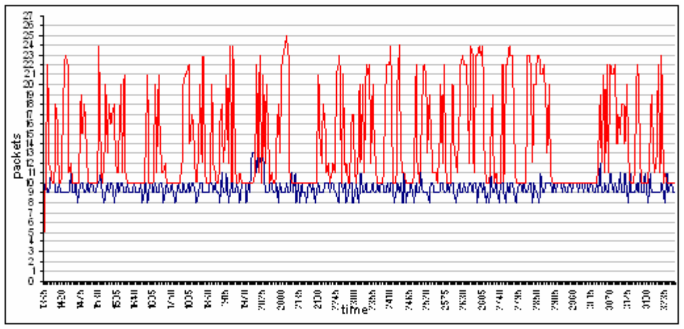

Figure 7. TCP cross traffic and UDP packets sent.

We are also interested on the behavior of TCP when subjected to our traffic shaping module. We altered TCP's code to print the events occurred when using the module, so we could track the number of packets sent.

To have a baseline measurement, the first test was conducted without the traffic shaping module. The result can be seen on Figure 8. The time taken to send our test file was 2465 seconds, with 2748 packets sent.

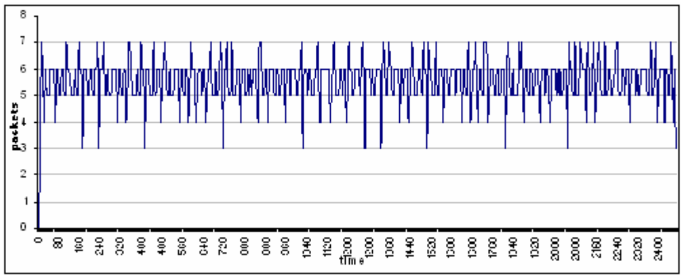

Figure 8. Baseline TCP measurement without module.

The average results when using the module can be seen on Figure 9. The minimum time was 2430 seconds with 2638 packets sent, and the maximum as 2495 with a total of 2665 packets. The average was 2457 seconds with 2651 packets sent. Although the time taken to transmit the data was the same, less packets were discarded by the network, which resulted in a decrease of 3.5% of the total of packets sent.

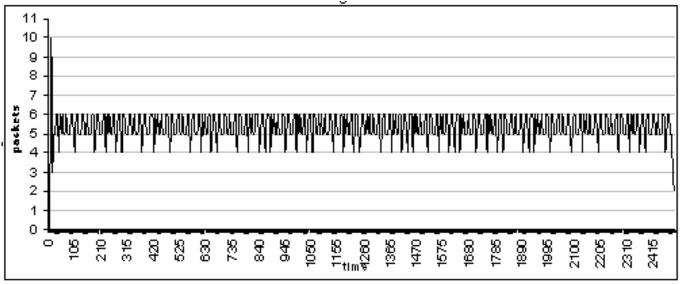

Figure 9. Average TCP performance with the module.

In the test above the cross traffic was TCP. The next test, both flows use the module, so both the cross traffic and the main transmission are going through traffic shaping.

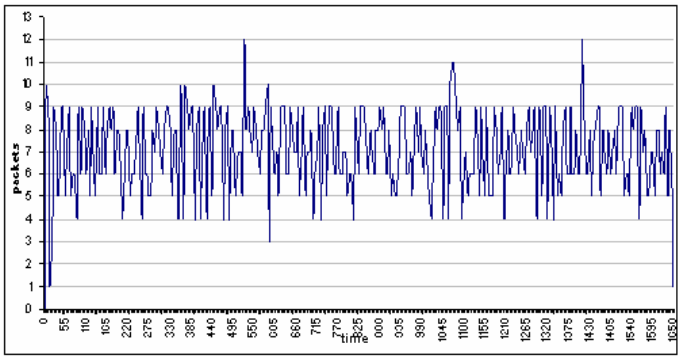

Figure 10. TCP behavior with two controlled flows.

Figure 10 shows the results. The minimum time was 1625 seconds with 2339 packets sent, and the maximum time was 1690 seconds with 2366 packets transmitted. The average value was 1661 with 2348 packets transmitted. This is a remarkable improvement of 32% less time and 14% less packets transmitted.

The behavior of the controlled cross traffic can be seen on Figure 11. Initially, TCP sends a large number of packets, which is reduced when the second flow is turned on, and returns to its original value when the file transmission ends. Although this is typical TCP behavior, we want to show that the traffic shaping module does not keep a flow from using all the available bandwidth when it is present.

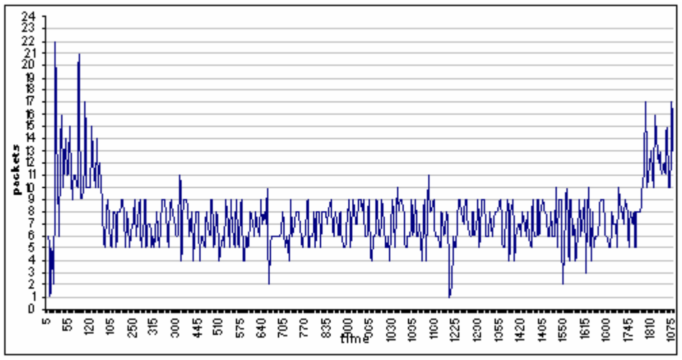

Figure 11. TCP cross traffic using the module.

Figure 12 shows both flows superimposed. Although both flows show the characteristic bandwidth fluctuations of TCP, both flows get their fair share of bandwidth, even considering the limitations of working with a real system implementation, where timer granularity and time spend doing other operating system functions influence on the outcome.

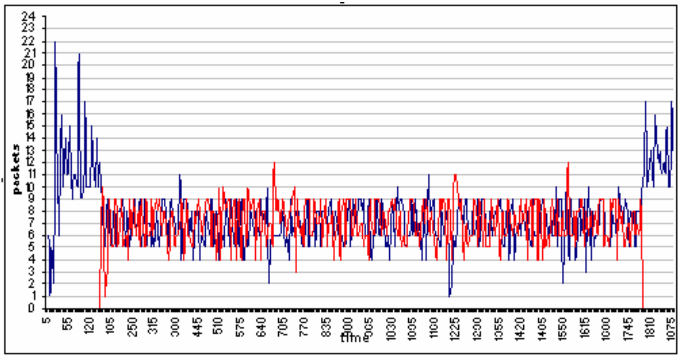

Figure 12. Bandwidth sharing of two TCP flows using the module.

IV. Conclusions and Future Work

While it is in its early stages, this investigation shows promise, specially in using traffic shaping techniques at the end-point to work as a congestion control mechanism for applications that may be badly behaved and cause congestion, specially those that use UDP.

We also obtained good results when the module was used with TCP. The biggest gain is better utilization of network resources, as packets that would be lost at the routers are stopped at the source. This leads to better network utilization, as seen by the decrease of the number of packets discarded.

The module works with any protocol, and measures the available bandwidth for all active destinations. We plan to study the interactions between TCP's congestion control and the pacing algorithm, and see how to better set the minimum threshold to fight the bias towards losing bandwidth to non-congestion avoidance traffic. Both congestion avoidance and TCP pacing showed promise, but did not behave well in an uncontrolled environment. Our better results were realized only when all flows were paced, which makes sense, but will not normally be seen on the network at large.

The need for better timers may also be a hurdle to be overtaken. It is expected that timer granularity will be finer as processor clock speeds keep on improving, and the current tendency of making smarter network cards [8], offloading network protocol processing from the main CPU to the NIC can also help this effort.

As the work develops, we will redo all measurements using common tools such as IPERF [11]. This will make the results easier to reproduce for other groups interested in using the module. We also plan to run tests on sending voice and video flows, to see how those perform under the module.

References

[1] Visweswaraiah, V., Heidemann, J., “Rate Based Pacing for TCP”, 1997 http://www.isi.edu/lsam/publications/rate_based_pacing/index.html

[2] Luiz C. Schara Magalhães. PhD thesis: A Transport Layer Approach to Host Mobility., University of Illinois at Urbana-Champaign, 2005.

[3] Kevin Lai, Mary Baker, “Measuring Link Bandwidths Using Deterministic Model of Packet Delay”, http://www.sigcomm.org/sigcomm2000/conf/paper/sigcomm2000-8-3.ps.gz, 2000.

[4] Prasad, R.S., Murray M., Dovrolis, C., Clay K.: Bandwidth estimation: metrics, measurement techniques, and tools. IEEE Network Magazine, Volume: 17, Issue: 6 pages: 27- 35, 2003.

[5] Hu, N., Steenkiste: Evaluation and Characterization of Available Bandwidth Probing Techniques. IEEE Journal on Selected Areas in Communication 21 Volume:21, Issue:6 pages: 879- 894, 2003.

[6] Strauss, J., Katabi, D., Kaashoek, F.: “A Measurement Study of Available Bandwidth Estimation Tools”. In: ACM SIGCOMM Internet Measurement Workshop. Pages: 39 - 44 , 2003.

[7] Johnsson, A., “On the Comparison of Packet Pair and Packet Train Measurements”. Proc ICCC, Las Vegas, June 2004.

[8] Fabien Viger, “Active Probing with ICMP Packets”, http://www.eleves.ens.fr/home/viger/docs/Active_Probing_with_ICMP.pdf, 2003.

[9] Postel, J., RFC 792, “Internet Control Message Protocol”, September 1981.

[10] L. Brakmo and L. Peterson. TCP Vegas: End to End Congestion Avoidance on a Global Internet. IEEE Journal on Selected Areas in Communication, Vol 13, No. 8 (October 1995) pages 1465-1480.

[11] A Tirumala, F Qin, J Dugan, J Ferguson, K Gibbs, “Iperf – The TCP/UDP bandwidth measurement tool”, http://dast.nlanr.net/Projects/Iperf/, 2005.