Techniques for Dynamic and Prioritized Bandwidth Allocation On Incoming Links

Saâd Biaz
Computer Science and Software Engineering Department
Auburn University
Auburn, AL 36849-5347, USA
sbiaz@eng.auburn.edu

March 19, 2002

Technical Report # CSSE02-01

Abstract

We consider the problem of prioritizing the *incoming* TCP traffic on a link. The objective is to allocate dynamically a specific share of the incoming link bandwidth to a high priority TCP traffic. However, if the high priority TCP traffic does not fully use this dedicated share, the low priority TCP traffic may take advantage of the unused bandwidth. Up to today, a network administrator has no control on the incoming traffic if the network service does not provide some QoS mechanisms.

In this report, we describe three techniques that would allow a network administrator to prioritize and allocate dynamically the bandwidth without any cooperation from the human, software, or hardware resources of the Internet Service Provider (ISP). No change in the TCP code is required.

1 Introduction

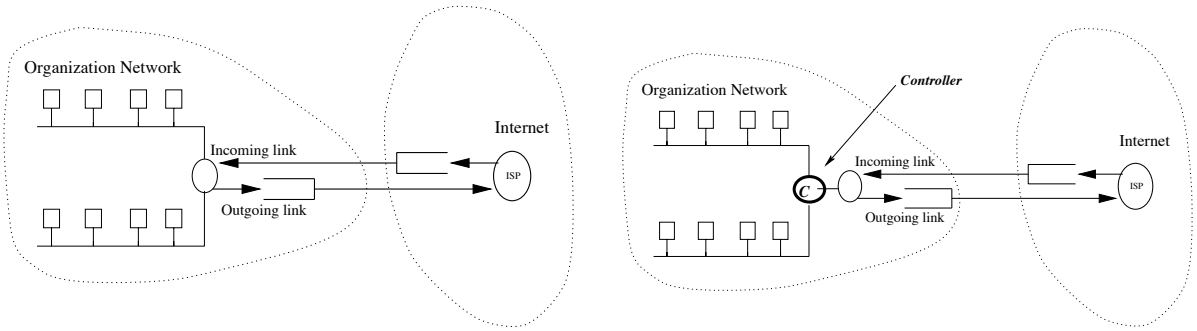
In the recent years, the nature of the traffic poured from the Internet onto network organizations changed dramatically with the increase of multimedia traffic (songs and movies) [6]. This type of traffic appears in general to the network administrator (and to the high management) to be "frivolous" or at least secondary with regard to the main activity of the organization. This secondary traffic may get so heavy that the primary mission of the network is completely jeopardized. For example, on a campus, downloading a paper may just become a nightmare, while students in the dorms are using some automatic downloading tools that are permanently in action congesting the *incoming link*. In the following, we will call this frivolous secondary traffic *low priority* traffic. The traffic supporting the main activity of the organization will be called *high priority* traffic. As of today, a network administrator cannot prioritize dynamically the incoming traffic in order to limit the bandwidth used by the low priority traffic if the network service provider does not offer some QoS mechanisms. In general, the Internet still does not offer these QoS mechanisms.

In this work we want to limit the bandwidth for the low priority TCP traffic only if the high priority traffic is not using the bandwidth. We aim to build such control tool, based on three completely different techniques. This report is organized as follows. Section 2 will describe the problem and the network model. Section 3 will describe the first technique which is based on tampering the window size field in the TCP header. The second technique, based on TCP agents that are responsive to ECN (Explicit Congestion Notification), will be described in Section 4. Finally,

Section 5 will expose the third technique, that consists in inserting a *variable bandwidth* bottleneck on the path of the low priority traffic.

2 The Problem

Figure 1.a shows a network organization connected to the Internet. The organization may be a family house, a campus, or a Navy vessel on mission. For simplicity, we will consider without loss of generality that the organization is a campus. A campus has a limited bandwidth on the incoming



(a) Network Model (b) Position of the Controller

Figure 1: Dynamic Bandwidth Allocator

link. While this incoming link carries in general more traffic than the outgoing link, it is not under the control of the campus network administrator. Users usually generate small requests and get large responses such as data files, or (worse!) multimedia files (songs and movies). Some downloading software tools open multiple TCP connections and may unduly consume all bandwidth on an incoming link. On a campus, the traffic to/from the research labs may be considered of high priority in comparison to the traffic to/from the dorms. For a campus network administrator, the problem is on this incoming link that gets congested with "frivolous", secondary, and low priority traffic.

Our research objective is to allocate and prioritize dynamically the bandwidth on the incoming link without any cooperation from the ISP (Internet Service Provider) network administrator and without any change to the TCP code.

Our research will concern only TCP traffic. Since TCP traffic is predominant on the Internet (about 80% of traffic [6]), we will be controlling most of the traffic on the incoming link.

We want to allow the campus network administrator to allocate and to secure a fraction of the bandwidth to the high priority TCP traffic. However, if the high priority TCP traffic does not use all its dedicated share, the low priority TCP traffic may take advantage of this unused bandwidth. The position of the traffic controller is shown on Figure 1.b. To our knowledge, there is no hardware or software tool that allows a network network administrator to allocate dynamically a fraction of the bandwidth on an *incoming* link to a high priority TCP traffic. Incoming traffic is just like rain pouring onto the network organization without any possible control from the campus network administrator. Today, a network administrator may only adopt the following techniques to provide good service to the high priority traffic:

1. Cut off Internet access to low priority traffic during critical hours.
2. Install a second line completely dedicated to low priority traffic
3. Ban access to certain sites on the Internet for a set of low priority users.
4. Filter, delay, or kill some low priority traffic packets.
5. Request from the ISP network administrator to prioritize the incoming traffic.

The first solution is quite drastic: low priority users may need to use the Internet during "critical hours" for basic services such as email, news, or http. Moreover, the incoming link from the Internet may be underutilized when low priority traffic is cut off. For the second solution, if the network administrator installs a second dedicated link to low priority traffic, or statically allocates to it some bandwidth, there will be a waste of resources. For example, on a campus, during daytime, the low priority link will be underutilized, as most of the students are in class or in the labs. During night, the low priority link may get heavily congested while the high priority link is underutilized, as most research labs, offices, and classes are deserted. The third solution that consists in banning the access to some sites is far from a good solution as new sites appear every day. The fourth solution that has the objective of affecting adversely the low priority incoming traffic is not efficient, because the incoming traffic has already used the bandwidth link. Such solution will only affect the quality of service provided to the low priority users without any effect on the bandwidth consumed. This policy is just a kind of *punishment* of low priority users. Finally, the network administrator may request from the ISP network administrator to prioritize the incoming traffic by configuring the ISP router. The ISP may not be willing to provide such service. It is also possible that the organization has a network address translator (NAT). Therefore, the ISP network administrator may not be able to distinguish the two kinds of traffic.

Our report proposes three techniques that address this problem of dynamic and prioritized bandwidth allocation of the *incoming* link without getting any cooperation from the network human, software, or hardware resources and without any change to the TCP code.

The following sections will describe the three techniques. The first technique below takes advantage of the *flow control* mechanism of TCP.

3 Fake Flow Controller

The first technique consists in tricking the low priority sender by tampering the window size field in the TCP header of all acknowledgements returned by the receiver.

TCP adjusts its sending *rate* by adjusting the amount of outstanding data (data sent and not yet acknowledged) in the network. The outstanding data is also called the window size. Two mechanisms are used to limit this window size: flow control and congestion control. The flow control allows a receiver to control and pace the sending rate of the sender such that the receiver buffers cannot get overflooded. The congestion control is achieved at the sender using congestion control algorithms [5]. We are not interested in this section by the congestion control mechanism. Flow control is achieved by TCP through the window size W_a advertized by a receiver. The TCP header bears a 16-bit field that is used by the receiver to advertize the window size W_a. When a sender receives the value W_a, it cannot have an amount of outstanding data larger than W_a.

The basis of our first technique is to tamper the field containing the advertized window size for every acknowledgement coming from the low priority TCP receiver. By tampering the window size field, the controller can adjust the sending rate of the sender of low priority traffic. This idea of tampering the window size field is due to my student Amiry Samir [8]. In his Master Thesis, the author implemented on Linux a controller that gave encouraging results on an experimental test bed.

The controller maintains variables that measure the incoming link utilization and the bandwidth used by each type of traffic. Measurements are performed by monitoring all incoming packets. The link utilization is measured to detect if there is contention between low priority traffic and high priority traffic. The controller periodically updates a coefficient of reduction R_w ($0 \leq R_w \leq 1$). R_w is used to decrease the window size in the TCP header field of low priority traffic acknowledgments. Every tick (period of time to be tuned), the algorithm shown in Figure 2 is invoked to update R_w.

```
At each tick:
if (Contention on Link) {
    if (Bandwidth Share of High Priority Traffic < Bandwidth Allocated) {
        Decrease Rw;
    }
    else Increase Rw;
}
else Increase Rw;
```

Figure 2: Periodical Update of the Coefficient of Reduction

For every acknowledgement for low priority traffic that is received by the controller, we invoke a second algorithm shown on Figure 3.

```
At the reception of each low priority traffic ACK:
if (Contention on Link) {
    if (Bandwidth Share of High Priority Traffic < Bandwidth Allocated) {
        Window Size Field <--- Rw * Window Size Field
    }
}
```

Figure 3: Tampering Window Size Field on Acknowledgements

The use of a multiplicative-decrease and additive increase [1] on the reduction coefficient R_w gives extremely encouraging preliminary results.

However, with this technique, it may happen that the window size is made so small that "tinygrams" start flowing over the network with a high overhead for the low priority traffic. To avoid this, we limit the smallest window size to be one mss (maximum segment size). This technique is currently under exhaustive simulation tests using *ns-2* [2].

The second technique assumes that a TCP sender reacts to the ECN (Explicit Congestion Notification) [3, 7]. This technique is described below.

4 Fake Window "Congestion" Controller

The idea of this technique is to trick the sender of low priority traffic by making it "think" that the incoming link is congested. When there is no contention on the incoming link, no action is taken by the controller. If there is contention, and the high priority traffic cannot use the allocated bandwidth, the controller starts setting the ECN bit on the acknowledgements of the low priority traffic. If a sender receives a packet with ECN flag set, it will throttle its sending rate. Therefore, low priority senders will send less data and the high priority traffic wil get more bandwidth. However, to make the control smooth, the ECN bit will be set using a probabilistic heuristic similar to the one used on RED routers [4]. The probability P_{ECN} of setting the ECN bit on low priority acks will be computed based on the "deficit" of bandwidth experienced by the high priority traffic. The deficit is the positive difference between the bandwidth allocated to the high priority traffic and the bandwidth effectively used by this traffic. For this technique, we will be using two algorithms. The first algorithm will be invoked periodically to update P_{ECN} as follows:

$$P_{ECN} = f(Link\,Utilization,\ Bandwidth\,Deficit\,of\,High\,Priority\,Traffic)$$

The second algorithm will be invoked for each low priority acknowledgement. This algorithm is shown in Figure 4.

At the reception of each low priority traffic ACK:

if (HIGH Contention on Link) {
Set ECN bit with probability 1;
}

if (MILD Contention on Link) {
Set ECN bit with probability P_{ECN}
}

Figure 4: Setting the ECN Bit on Low Priority Acknowledgements

This is just a draft on how this technique should be implemented. This technique will be first simulated in order to identify key factors of the control loop.

In the following section, we describe the technique that inserts a "fake" bottleneck on the path of low priority traffic.

5 Fake Bottleneck

The idea is to insert a simple forwarding agent on the path of the incoming low priority traffic. Figure 5 shows the position of the "fake" bottleneck (Circle labelled with the letter "F"). The

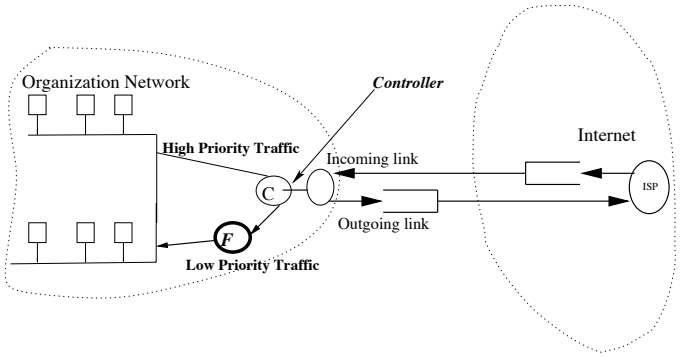


Figure 5: "Fake" Bottleneck

incoming low priority traffic will go through the forwarding agent. This forwarding agent must have a variable sending rate (this can be achieved). The controller will periodically monitor the bandwidth used by the high priority traffic. Any left over (unused) bandwidth B_L can be allocated to the low priority traffic. The sending rate of the "fake" bottleneck will then be set to B_L. The direct effect of such limitation on the forwarding agent will force the TCP sender of low priority traffic to *ADAPT* to the "fake" bottleneck instead of adapting to the ISP link characteristics. By periodically updating the sending rate of the forwarding agent, we force the low priority traffic to use only the bandwidth we want to allocate to it.

However, one of the major drawbacks of this technique is that packets for the low priority traffic will be dropped after they have already used the bandwidth on the incoming link. It is of interest to study the performance of this technique including the losses of packets that already used the incoming link.

6 Conclusion

We plan to study, analyze, and evaluate these three completely different techniques that can achieve dynamic and prioritized bandwidth allocation. Key parameters of these techniques are not well known or understood. Extensive simulation tests will allow us to identify these factors and tune each technique. Our investigation may also unveil new different techniques during the progress of the research.

After the simulation, we plan to evaluate the three techniques by setting up an experimental test bed.

References

[1] D.-M. Chiu and R. Jain, "Analysis of the increase and decrease algorithms for congestion avoidance in computer networks," *Journal of Computer Networks and ISDN*, vol. 17, pp. 1–14, June 1989.

[2] K. Fall and K. V. editors, "VINT project U.C. berkeley/LBNL, ns2:network simulator." http://www-mash.cs.berkeley.edu/ns/.

[3] S. Floyd, "TCP and explicit congestion notification," *ACM Computer Communication Review*, vol. 24, pp. 20–35, Oct. 1994.

[4] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Transactions on Networking*, vol. 1, pp. 397–413, Aug. 1993.

[5] V. Jacobson, "Congestion avoidance and control," in *ACM SIGCOMM'88*, pp. 314–329, Aug. 1988.

[6] S. McCreary and K. Claffy, "Trends in wide area ip traffic patterns," 2000. http://www.caida.org/outreach/papers/2000/AIX0005/AIX0005.pdf.

[7] K. Ramakrishnan, S. Floyd, and D. Black, "RFC 3168: The addition of explicit congestion (ECN) to IP," Sept. 2001.

[8] A. Samir, "Dynamic bandwidth allocation," Master's thesis, Al Akhawayn University, Ifrane, Morocco, Dec. 2001.