Rescheduling Lengthy Downloads on Bottleneck Links

Odongo Y. G.[1*] and Rai A. I.[2]

[1]Department of Computer Science, Egerton University, P.O. Box 536-20115, Njoro, Kenya. Email: godongo@egerton.ac.ke

[2]Faculty of Computing and IT, Makerere University, P.O. Box 7062, Kampala, Uganda

Received: 5th April, 2011; Revised: 12th Aug., 2012; Accepted: 5th Oct., 2012

Abstract

TCP's ability to share bottleneck bandwidth fairly and efficiently dwindles when faced with an increase in the number of competing flows. Indeed studies have shown that long TCP flows consume more than 90% of the available bandwidth, with short flows consuming a very small portion. It is also known that out of these, a very small percentage of the largest flows carry the majority of the bytes. The utilization pattern of bottleneck links exacerbates this situation. During the daytime (peak period), the network utilization is characteristically high whereas at night (off-peak period), the link is virtually idle. At peak periods, TCP's congestion control mechanisms highly favour long-lived flows at the expense of their short-lived counterparts. We discuss the design and implementation of a flow rescheduling tool. The tool identifies, blocks and reschedules long flows to reconnect during the off-peak period. Network bottlenecks are a commonplace phenomenon in many IP networks just as much as they are unavoidable. TCP's ability to share bottleneck bandwidth fairly and efficiently dwindles when faced with an increase in the number of competing flows. Indeed recent studies have shown that long TCP flows consume more than 90% of the available bandwidth, with short flows consuming a very small portion. It has also been realized that out of these, a very small percentage of the largest flows carry the majority of the bytes. The utilization pattern of bottleneck links does not make matters any better. During the day, the utilization pattern is characteristically high whereas at night, the link is virtually idle. We refer to these times as the peak and off-peak periods respectively. At peak time, TCP's congestion control mechanisms highly favor long-lived flows at the expense of their short-lived counterparts. This work discusses the design and implementation of a flow rescheduling tool, specifically illustrating the techniques used to achieve accuracy and robustness. The tool works by identifying and rescheduling long flows to reconnect during the off-peak period, when link utilization is very low. We present validation results for the tool which demonstrate its reliability in the face of actual Internet conditions.

Key words: rescheduled flow, peak period, off-peak period

Introduction

Unrelenting hasty growth and proliferation of new applications have combined to transform the face of the Internet. Entertainment and real-time applications like medical telemetry, network gaming, voice-over-IP and streaming video have quickly gained ground. These applications not only demand a very diverse set of network performance requirements, but networks themselves are also experiencing rapid growth in terms of the number of users as well as traffic per user (Claffy *et al.,* 1998; Houle *et al.,* 2007). With this growth comes a seemingly insatiable demand for bandwidth with applications now engaged in a stifling competition for network resources. Traditionally there have been two canonical approaches to dealing with this issue. The first is to augment the installed bandwidth to build adequate capacity, and the second to formulate a class-based differentiated service to meet each applications performance requirements. While research on the latter is apparently nonchalant, the former is widely used despite the fact that bandwidth is prohibitively expensive. Over provisioning appears to be an attractive solution but in the actual sense each time the bandwidth is increased, the demand will grow to consume it. Consumer behaviour and expectations tend to change in line with the additional capabilities of the technology thus creating a vicious circle (Claffy, *et al.,* 1998; Rai *et al.,* 2005).

Studies to understand the characteristics of contemporary Internet traffic have shown that a very small percentage of flows consume most of the network bandwidth (Tokuda *et al.,* 2002; Myung-Sup *et al.,* 2004). Internet traffic is known to comprise of two main types of flows the long-lived flows (also known as elephants or heavy-hitters) and the short-lived flows (or mice). Short-lived TCP connections send small documents and spend most of their lifetime in the connection phase while long-lived TCP connections, which transmit large documents, spend most of their lifetime in the established phase. Because of the inherent nature of the ACK-based window flow control of TCP, short-lived flows suffer from a significantly lower throughput compared with long-lived TCP counterparts (Tokuda *et al.,* 2002).

In this work, we discuss the design of a system that intercepts, blocks and reschedules time-consuming file download requests. Later on, we outline results of actual deployment to show the benefits of possible use in contemporary networks. Computer networks as currently deployed do not give network administrators the option to reschedule time-consuming flows

to other periods when the link is idle. The tool we describe here makes the following contribution;

1. Stabilize the link by ensuring that a small number of long flows do not hog bandwidth and other network resources at the expense of already admitted or subsequently admitted flows.
2. Prevent a severe service degradation by monitoring link utilization and preventing undue overloading of an access link by long-lived flows
3. Prevent wastage of resources by connections which are created and aborted midway because of a slow or less than adequate response.
4. Improve the users' perceived service quality by improving the flow completion time

The results are interesting in demonstrating the possibilities of the approach. The immediate benefits of the system in the users perspective is to save him from the agony of having to wait for hours for a download to complete without the guarantee that his many hours of waiting will result in any success. All downloads that are made during the off-peak period are stored in a private directory of the user on a file server. To access the file he will be required to log-in and copy the file from his private directory.

While it is possible to simply ask users to schedule large files for download during off-peak hours and provide them space on an FTP server where their downloads are placed for them to pick up in the morning, the practicalities of this can be challenging for two reason; one, there is no mechanism available to the ordinary user by which he will determine that the network is now congested and he needs to schedule. Secondly and of more concern, is that this mode of operation requires total cooperation from network users, something that is more of a mirage since network users are often just as optimistic as the TCP congestion avoidance algorithm itself. They will attempt a download until they fail. But the failure will often come after bandwidth has been wasted and the network has suffered suffer congestion as described earlier. We therefore conclude that there must be a mechanism that makes cooperation mandatory and not optional, for large flows to be rescheduled for later download.

The next section motivates the work by looking at the composition of contemporary Internet traffic.

Objectives and Approach

During the peak period, users making huge file downloads as defined in the context of this work often have to bear the agony of sitting for hours waiting

for a download to complete due to low speeds and congestion. Unfortunately, waiting does not always pay off since TCP gives them no guarantee of success. As a result, the users' perception of quality of service is often low.

Typical campus backbone speeds are 1000 Mb/s and are based on Gigabit Ethernet. The WAN access link where the prototype was tested had an installed capacity of about 42 Mb/s. However as opposed to these numbers, note that typical Internet access link speeds range from 1 Mb/s to about 10 Mb/s, the latter being more expensive and uncommon in developing countries.

From the scenario described above, and with the knowledge that much of the traffic on the Internet is carried by a small number of large flows (elephants) while most of the flows are short in duration and carry a small amount of traffic (mice); it becomes immediately apparent, that the WAN access link has the potential to become a bottleneck resource that needs to be management and controlled. Our approach to management and control works by limiting the number of long flows in a scenario such as that depicted in Figure 1. Our monitoring and control architecture is also shown in Figure 1.

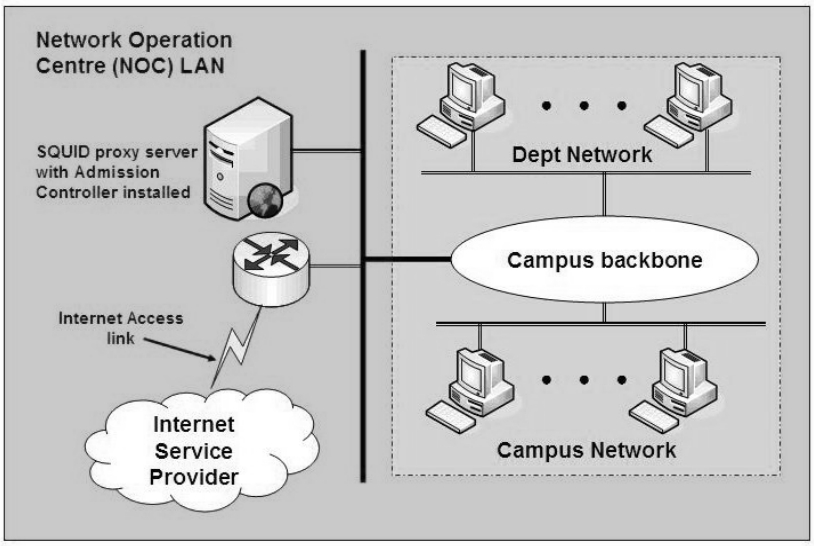

Figure 1: A campus network with an access connection to an ISP

All traffic between the WAN access link and the campus network is made to pass through an Ethernet segment or equivalently, an Ethernet hub. On this segment sits a machine that monitors and controls the admission of all new connection requests. On this machine also sits a proxy server. The proxy server we used is SQUID which we configured such that all new requests it receives are first sent to our tool for classification.

Based on link utilization policies (link utilization and file size threshold) specified by the network manager, the monitoring machine is then able to control TCP connections (i.e. block and reschedule new ones if necessary) so that traffic flowing on the access link conforms to the configured policy. Some sample policies that can be set up are:

- If the access link occupancy exceeds 65% then begin exercising admission control especially for flows whose resource size exceeds 55 MB. Short flows would be allowed since they are not expected to last more than a few seconds.
- Block all new connections whose requested resource exceeds 45 MB, if the link utilization is also greater than 75 %. Long flows typically run for tens of minutes and result in unfairness to short flows.

In order to prevent a situation where long flows are perpetually blocked, we define a period during which the tool is active and another period during which the tool is inactive. We refer to these as the peak and off-peak periods respectively. The peak period refers to the time in between 8 am and 8 pm while the off-peak period lies between 8 pm and 8 am the following morning.

The tool we have developed requires that certain variables be set to certain values. The decision as to what level to set these values is dynamic depending on one, the installed bandwidth capacity and two, the desired level of control required to produce an effective effect on the network. For instance, the manager may have established that the maximum speeds during the off-peak period are averagely 13.7 KB/s. The manager would then decide that any file that requires more than two hours to download at 13.7 KB/s is a long flow. He will then set the *filesize.threshold* parameter to reflect that decision. Parameters of the bandwidth management solution that must be set include the following;

1. **available.bandwidth** - This file specifies the maximum available bandwidth or installed capacity in bits per second.
2. **filesize.threshold** - This file specifies the maximum file size that the user is permitted to download during the peak period. During off peak, this restriction is not enforced. Off peak is taken to be the time between 8 pm and 8 am while the peak period is that between 8 am and 8 pm.
3. **utilization.threshold** – This file specifies the maximum utilization permissible on the network beyond which bandwidth management automatically kicks in. Utilization is typically a number between 0 and 1.

4. **consumed.bandwidth** – This file is written into by a script that runs at regular intervals. The script calculates the average utilization and is overwritten every five minutes.

To aid the network manager come up decisions that determine network conditions he may have to consult the organizations ICT Policy document for guidance so that the bandwidth management solution is an implementation of those policies. Such a document would typically take into account the user needs and specifies what the user is allowed to do, what is prohibited, which sites one can access and at what times etc.

Variability of Internet Traffic

At inception and in tandem with applications in use at the time, Internet traffic comprised mainly of short-lived flows. Short-lived flows are considerate bandwidth consumers since they make a modest demand for network resources. With time however, new applications have been introduced and the Internet traffic has seen growth both in diversity and complexity (Myung-Sup et al., 2004).

Table 1: Most Popular Internet Applications by Traffic Analysis

Applications	Bytes (%)	Packets (%)	Flows (%)
Web	75	70	75
DNS	1	3	18
SMTP	5	5	2
FTP	5	3	<1
NNTP	2	< 1	< 1
Telnet	< 1	1	< 1

In describing Internet traffic in terms of protocol and application composition, TCP dominates 75% of the total Internet flows. These flows convey up to 95% of the total bytes transferred and account for 90% of the total packet count. UDP ranks second with about 20% of the total Internet flows ferrying 5% of the bytes and accounting for 10% of the packets. ICMP uses most of what is remaining (Claffy, *et al.,* 1998; Rai *et al.,* 2005; Myung-Sup *et al.,* 2004; Thompson *et al.,* 1997; Ebrahimi-Taghizadeh *et al.,* 2005). In terms of application composition, the web ranks first as seen in Table 1.

Active versus Passive Measurements

Network performance measurements can be classified as being either active or passive. Active measurements are those techniques that carry out measurements by injecting new packets into the network and thereafter make inferences into its state (Prasad *et al.,* 2003). Active measurements

techniques do not require co-operation from network devices, the advantage being that one does not need to have administrative access to network nodes to successfully make measurements (Prasad *et al.,* 2003; Harfoush, *et al.,* 2003; Saroiu, *et al.,* 2002).

Passive measurement techniques have been noted to be accurate and relatively easy to take (Calyam, *et al.,* 2005). Such methods are suitable for environments or networks that are wholly owned and administered by a single organization such as is the case of a corporate or institutional network (Hasib and Schormans, 2003). In such environments, the selection of the measurement or capture point can be freely made from any point on the path from the sender to the receiver. Since passive methods do not create any additional network traffic, they provide an accurate representation of the state of the network. Passive techniques are however vilified because they tend to duplicate and store all the network traffic for analysis, hence creating a need for enormous storage space which may run into hundreds of gigabytes. Besides, because of their duplicate and store tactic, passive techniques also raise serious security and privacy issues (Saroiu, *et al.,* 2002).

Admission Control for TCP Connections

The Internet seamlessly accommodates applications with long and short-lived connections alike. However unknown to regular users is the fact that there is an ongoing "war" between the elephants and the mice (Tokuda *et al.,* 2002; Ebrahimi-Taghizadeh *et al.,* 2005; Guo and Matta, 2001; Laatu *et al.,* 2003). This war is created purely by TCP's standard congestion control mechanisms. Even though this congestion control mechanism is novel, it diminishes TCP's ability to distribute bottleneck bandwidth fairly and efficiently as the number of contending flows increases (Ebrahimi-Taghizadeh *et al.,* 2005). In an effort to address this unpleasant situation, admission control has been advocated for as an effective means of ensuring that active flows have a minimal acceptable throughput (Massoulie and James, 1999).

Admission control consists in refusing a new flow if the addition of its traffic would lead to an unacceptable quality of service level for that or any previously accepted flows. It is also essential in preserving the efficiency of the network and the excellence of services offered to users since there is a minimum acceptable throughput below which no positive utility can be derived. In the absence of such control, the unproductive traffic due to the retransmission of lost packets constitute a major overhead and may lead to congestion collapse in certain configurations.

Also, as the data reception rate becomes very low, users tend to abandon their transactions due to impatience. Additionally, higher layer protocols may deduce that excessive acknowledgement delays are a sign of link failure, thereby interrupting the connection. Both situations result in unnecessary and wasteful commitment of scarce network resources (Roberts and Massoulie, 2000). By using admission control, we ensure that any accepted flow receives a minimum acceptable throughput and avoids wasting network resources on retransmissions and incomplete transfers (Massoulie and James, 1999).

Traffic Engineering and Load Control for Elastic Flows

Studies have shown that TCP-controlled elastic traffic, is predominant (Roberts and Massoulie, 2000). Traffic that is elastic in character includes those produced by applications such as e-mail, FTP, HTTP and SMTP. All these applications essentially involve the transfer of files. Elastic flows are primarily established for the transfer of digital objects which can be conveyed at any rate up to the maximum value imposed by the link and system capacity. The digital object referred to might be a file, a web page or a video clip transferred for local playback. We generically refer to such objects as documents.

Requests for transfer of these documents are triggered mainly by the act of a user at a client clicking on a URL, and then the TCP protocol controls the sharing of the access link bandwidth between the ongoing file transfers from the Internet to campus clients. Since the campus backbone is typically of a much higher speed than the access link, the file transfers are logically bottlenecked at the access link.

For an elastic flow, quality of service is manifested essentially by the time it takes to complete the document transfer. This time depends both on the way bandwidth is shared and on the random fluctuation in the number of flows in progress as flows begin and end. For example, taking account of random traffic, it has been shown in (Roberts and Massoulie, 2000) that throughput can be greatly improved by actively discriminating in favor of shorter transactions. To ensure quality of service in case of traffic overload, it appears necessary to additionally employ admission control, with flow blocking appearing as a more acceptable quality degradation than diminishing throughput.

Structural and Control Design

The flow-control system described here was implemented on a Linux-based gateway and comprised of the components shown in Figure 2.

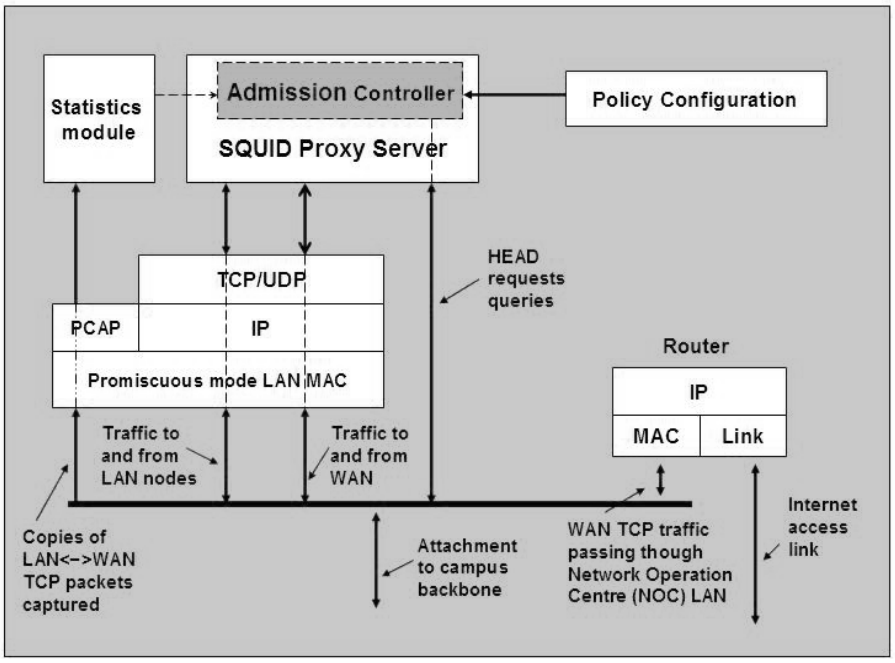

Figure 2: A campus network with an access connection to an ISP

- Statistics Module: Link utilization was monitored using IPTraf which is a console-based network statistics utility for Linux. IPTraf was configured to log the interface statistics every minute.
- Admission Controller: The admission controller implements policy and contains the logic modules. It distinctively comprises two main units;
    1. Control and Logic Module: This module uses the utilization statistics and the configured control policy to determine whether or not a stream of packets or connections needs to be blocked. We used an admission algorithm to trigger the controls and enforce the policy settings for a connection.
    2. Policy Configuration: The configuration of the control policy is done by the network administrator or manager by editing start-up files used by the system. Regular network users cannot configure the system.

Admissibility Criteria

Our approach consisted in blocking and rescheduling the heavy hitters during periods of high utilization while allowing such flows during times of low utilization. The peak period has defined to begin at 8:00 am to 8:00 pm and off-peak from 8:00 pm to 8:00 am. In order to aid discussion, the variables *res_Size* are defined as the actual size of the user requested file while *curr_Util* represents the existing bandwidth utilization. The variable

ut_Thresh is defined to hold the utilization threshold and *fs_Thresh* to hold the resource size threshold. *Ut_Thresh* and *fs_Thresh* are set when the system reads the configuration files while *curr_Util* and *res_Size* are dynamically written during system operation. Modifying the configuration files changes the values of *ut_Thresh* and *fs_Thresh* and alters the admissibility criteria. Each new connection request that arrives during the peak hours is subjected to the algorithm below. Any flow that fails to meet the set criteria is logged and the user informed of the event.

if ((*res Size* <= *fs Thresh*) **and** (*curr Util* <= *ut Thresh*)) **then**
admit the flow
else if ((*res Size* <= *fs Thresh*) **and** (*curr Util* > *ut Thresh*)) **then**
admit the flow
else if ((*res Size* > *fs Thresh*) **and** (*curr Util* <= *ut Thresh*)) **then**
reschedule the flow
else if ((*res Size* > *fs Thresh*) **and** (*curr Util* > *ut Thresh*)) **then**
reschedule the flow

Algorithm 1: Flow Admissibility Criteria

Results for the Variation of Utilization

In the first experiment, the utilization threshold was set to 0.45 and the test run with various figures for the resource size. This was done with the file size threshold set to 50MB, 75MB and 100MB respectively and a count kept of the number of flows that were blocked. Figure 3 plots the number of flows blocked versus time. It also plots the real-time utilization versus time on the secondary y-axis.

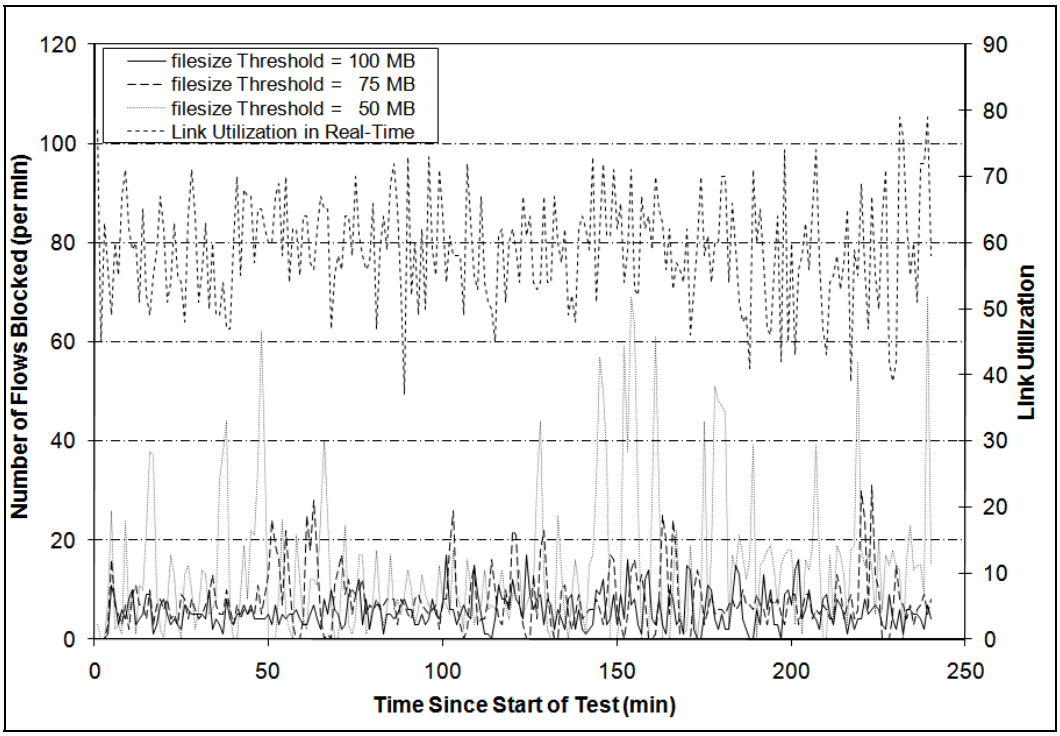

Figure 3: Number of blocked flows with utilization threshold held at 0.45

It is seen that as the file size threshold is varied upwards, there is a reduction in the number of blocked flows. Additionally, it is seen that the number of flows blocked per minute was highest at 69 in the 154th and 239th minutes. This can be attributed to the fact that both the real-time utilization and the requested file size were well above the defined thresholds values of 0.45 and 50MB respectively.

In the second experiment, the utilization threshold was raised to 0.65. The file size was again varied at 50 MB, 75MB and 100MB respectively and the test run again. As before, a count was kept of the number of flows that were blocked. Figure 4 plots the number of flows blocked versus time on the primary y-axis as well as the real time utilization versus time on the secondary y-axis.

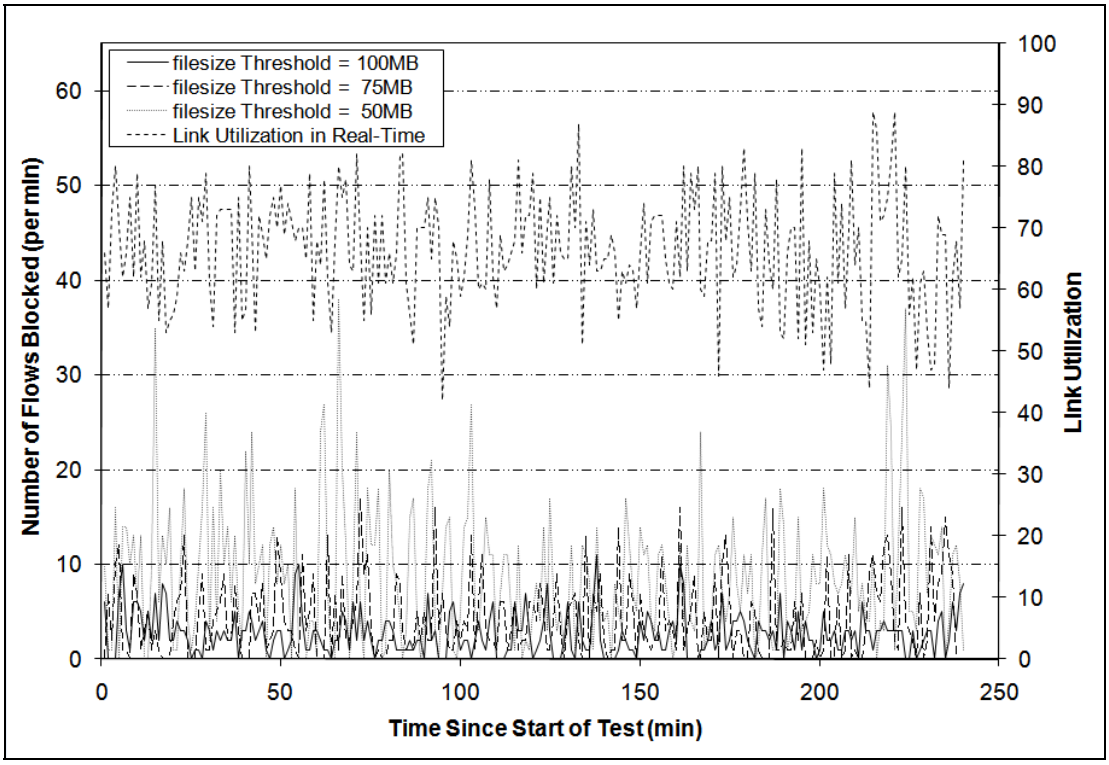

Figure 4: Number of blocked flows with utilization threshold held at 0.65

It is observed from Figure 4, that as the file size threshold is adjusted upwards, there is a pattern of reduction in the number of blocked flows. Additionally, it can be seen that during the 15th, 66th and 225th minutes, the number of flows blocked per min was high at 35, 38 and 37 respectively. This can be attributed to the fact that at that time, both the real-time utilization and the requested file size were well above the defined thresholds of 0.65 and 50MB respectively and provided a sufficient trigger for flow blockage.

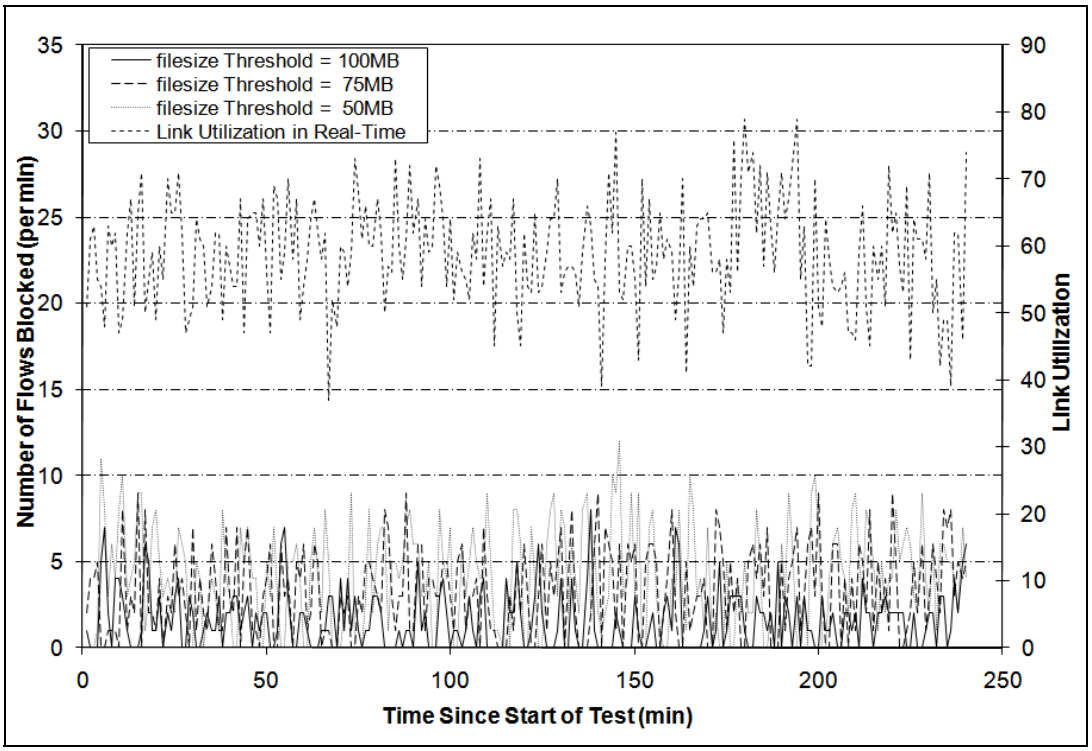

Figure 5: Number of blocked flows with utilization threshold held at 0.85

In the third experiment, the utilization threshold was increased further to 0.85 and as before, the resource size varied at 50MB, 75MB and 100MB respectively. Figure 5 plots the number of flows blocked versus time on the primary y-axis as well as utilization in real time versus time on the secondary y-axis. It is once again observed that as the file size threshold got smaller, there is a marked reduction in the number of blocked flows. In addition, it is noticeable that the number of blocked flows per minute went above 10 only during the 5^{th} and 146^{th} minutes. This is low as compared to the values that were obtained during other experiments and is probably due to the fact that real-time utilization never really went beyond 80 %. This means that the only parameter that triggered the rescheduling mechanism was the file size.

Automatic Reconnection of Blocked Flows

During the off-peak period, the system automatically re-connected the blocked flows. It did this by reading the log of flows it blocked on the particular day. The system reads the first fifty flows and reconnected them simultaneously. Once these fifty were running, it kept on checking to see that

when one flow completes, a new one is launched in its place. This mechanism ensured that at any one particular time, there were fifty flows running for as long as it had pending flows to initiate. It is seen from Figure 6 that though first 50 flows are reconnected within the first minute, subsequent reconnections do not all happen at the same time. This is due to the fact that different flows have different completion times since each such request involves the transfer of a random number of bytes of data.

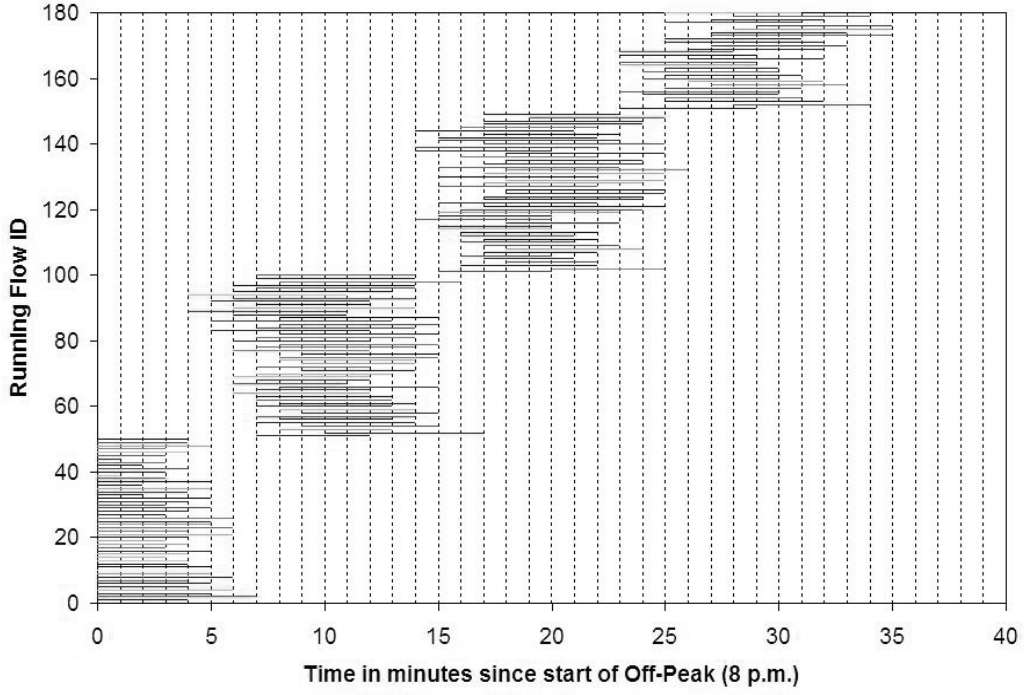

Figure 6: The resumption of flows during the off-peak period

Figure 6 shows how blocked flows were re-established over the first forty minute period. From the graph we see that though the first bunch of flows was resumed in the 1st minute, they were completed at different times. The next set of fifty flows were reconnected somewhere in between the 7th minute to 13th minute. The third lot of fifty flows were reconnected at around the 15th minute to 23rd minute. The last bunch of fifty flows in the experiment were reconnected somewhere in between the 25th minute to 37th minute. This method of reconnection ensures that the link is kept engaged during most of the off-peak time. With 50 flows running concurrently, the tool was able to download data at a rate of about 12.24MB per second and 34.4GB per hour. As compared to the peak time file download speeds, it is seen that the maximum download speed attainable was around 13.7KB/s which is very low. Put in perspective, during the peak periods it would take about 14 hours to download a 700MB file at a speed of 13.7 KB/s whereas

during the off-peak period, with speeds reaching 250KB/s, one only needs 47 minutes to retrieve the same file.

Conclusion and Further Research

An admission control approach that deals with the problem of resource hoarding by lengthy TCP flows has been presented. At the heart of this approach is a flow monitoring tool that sits on the network gateway. It determines the resource size by sending a simple HEAD method to the remote host with the desired resource. This information is then parsed to an admission control algorithm whose parameters have been preset by the network manager. By comparing the results of the HEAD request to the preset values, a decision is made on whether the flow is blocked or allowed. Since only a small number of bytes are sent and received to obtain this information, this approach to admission control does not allow a flow to connect and needlessly consume scarce network resources, only for it to be discovered later that the request is unable to complete successfully due to persistent high utilization. The blockage of long flows corrects the situation of hoarding of network resources by long flows especially when the network utilization is remarkably high. Additionally, we see that the system is able to make effective use of idle capacity during the off-peak period by automatically reconnect blocked flows at a minimum rate of about 25GB worth of data per hour.

In the context of the problem of a high speed campus network connected to the Internet by a relatively low-speed WAN access link, we have experimented with a TCP Flow-admission control-based strategy to correct unfairness to short flows. Flow admission control appears to be a feasible way to guarantee some TCP throughput performance on an overloaded Internet access link.

In view of new Internet services and applications, it would be interesting to carry out further research on how such control can be applied to streaming applications and services.

References

Calyam, P., Krymskiy, D., Sridharan, M. and Schopis, P., (2005). Active and Passive Measurements on Campus, Regional and National Network Backbone Paths, in Proceedings of the 14th International Conference on Computer Communications and Networks (ICCCN), pp. 537–542.

Claffy, K.C., Miller, G. and Thompson, K., (1998). The Nature of the Beast: Recent Traffic Measurements from an Internet Backbone," International Networking Conference (INET) '98, pp 71-82.

Ebrahimi-Taghizadeh, S., Helmy, A. and Gupta, S., (2005) TCP vs. TCP: A systematic study of adverse impact of short-lived TCP flows on long-lived TCP flows," INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE, vol. 2, pp. 926–937.

Guo, L. and Matta, I., (2001). The War between Mice and Elephants, in 9th IEEE International Conference on Network Protocols (ICNP), Riverside, California.

Harfoush, K., Bestavros, A. and Byers, J., (2003). Measuring Bottleneck Bandwidth of Targeted Path Segments, in Proceedings of IEEE INFOCOM.

Hasib, M., and Schormans, J. A. Limitations of Passive and Active Measurement Methods in Packet Networks, in Proceedings of the London Communications Symposium, 2003.

Houle, J.D., Ramakrishnan, K.K., Sadhvani, R., Yuksel, M. and Kalyanaraman, S., (2007) The Evolving Internet - Traffic, Engineering, and Roles, in 35th Research Conference on Communication, Information and Internet Policy (TPRC), pp. 23–36.

Laatu, V., Harju, J. and Loula, P., (2003). Evaluating Performance among different TCP flows in a Differentiated Services enabled network, 10th International Conference on Telecommunications (ICT), Vol. 1, pp. 709–715.

Massoulie, L. and James, R., (1999). Arguments in favour of Admission Control for TCP flows, *Teletraffic science and engineering, vol. 3a, pp.* 33–44.

Myung-Sup, K., Won, Y.J., Hyung-Jo, L.J., Hong, W. and Boutaba, R., (2004). Flow-based Characteristic Analysis of Internet Application Traffic, in IEEE International Workshop on End-to-End Monitoring Techniques and Services (IEEE E2EMON).

Prasad, R.S., Murray, M., Dovrolis, C. and Claffy, K., (2003). Bandwidth Estimation: Metrics, Measurement Techniques, and Tools. *IEEE Network., vol. 17, no. 6, pp. 27–35.*

Rai, I.A., Biersack, E.W. and Urvoy-Keller, G., (2005). :Size-based Scheduling to Improve the Performance of Short TCP Flows. *IEEE Network, vol. 19, no. 1, pp. 12–17.*

Roberts, J. W., and Massoulie, L., (2000). Bandwidth Sharing and Admission Control for Elastic Traffic, in Telecommunication Systems, vol. 15. Springer, pp. 185–201.

Saroiu, S., Gummadi, P. K. and Gribble, S. D. (2002). SProbe - A Fast Technique for Measuring Bottleneck Bandwidth in Uncooperative Environments, in Proceedings of IEEE INFOCOM.

Thompson, K.G., Miller, J. and Wilder, R., (1997). Wide-area Internet Traffic Patterns and Characteristics, Network, *IEEE, Vol. 11, no. 6, pp. 10–23, Nov/Dec 1997.*

Tokuda, K., Hasegawa, G. and Murata, M., (2002). Analysis and Improvement of the Fairness between Long-lived and Short-lived TCP Connections, in Seventh International Workshop on Protocols for High-Speed Networks (PfHSN2002).