Internet Packet Transport: Traffic Control and Network Engineering*

ANURAG KUMAR
Dept. of Electrical Communication Engineering
Indian Institute of Science
Bangalore, 560 012

Abstract

The Internet can be viewed as the transportation infrastructure for the deployment of Information Technology. The flow of traffic in any transportation network needs to be controlled to avoid congestion, to provide special service to flows that need it, and to protect against overloads. In this paper, we consider the control of store-and-forward (also called *elastic*) traffic in the Internet. We first motivate the need for feedback based congestion controls for elastic traffic. These controls provide for fair sharing of the network bandwidth, and hence determine how bandwidth is shared between elastic flows. We discuss two approaches for implementing such controls: in one, the network provides explicit feedbacks to the traffic sources, and in the other the sources adjust their rates based on implicit feedbacks. Then we discuss models that could help in sizing the Internet for guaranteeing minimum throughputs to elastic flows. These models lead to engineering guidelines and suggest the need for bandwidth management and overload controls. Finally, we present results from an experience in developing and deploying such controls in a campus network.

Keywords: congestion controls, bandwidth sharing, traffic engineering, overload in the Internet

1 Introduction

The Internet is widely regarded as the most powerful emerging technology for transporting information, and is now an integral part of the Information Technology revolution. Internet transport is versatile enough to be able to operate over a variety of physical transmission

*In Proceedings of the Indian National Science Academy Seminar on Emerging Communications Technologies and Society, New Delhi, March 1999.

media, ranging in speeds from a few kilobits per second to more than a billion bits per second. In the Internet, all information (whether it is data, or digitised voice or video) is transported as packets of bits, that are moved from link to link by packet switches that are called routers.

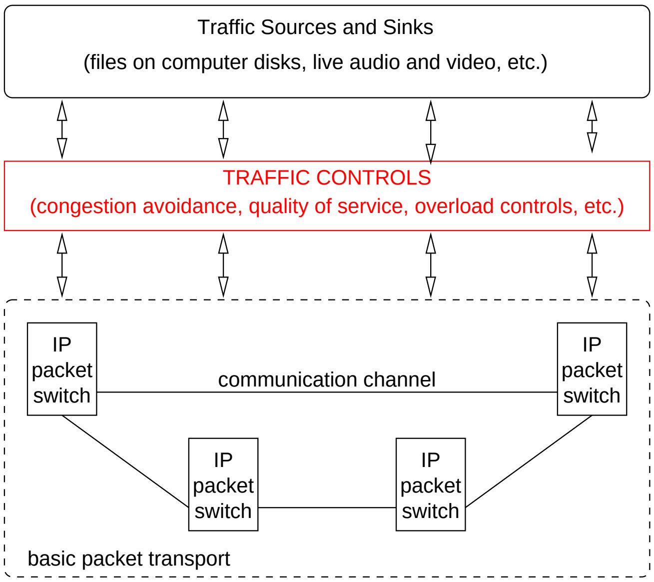

Figure 1: Traffic controls over a basic packet transport.

The digital communication links and the routers provide a basic transport mechanism for moving data around. Figure 1 shows such a transport mechanism being used to carry traffic from various sources. If these sources are permitted to use the basic transport system directly, there will be chaos. Hence traffic controls are always implemented in packet networks. These controls help to prevent congestion, provide service differentiation and quality of service, and take action in case of traffic overload. In Figure 1 these controls are shown conceptually as lying between the traffic sources/sinks and the basic packet transport. The controls are actually implemented as software in the network routers, and in the communicating end-systems. Distributed procedures serve to carry out the various control functions.

At the present time, the overwhelmingly large percentage of use of the Internet is for the transport of, so-called, *elastic* sources of information. Examples of such sources are data files, image and audio files residing on computer disks. Transport of such stored information is distinguished by the fact that there is no natural transfer rate that such a source of information requires. A file on a computer disk, if transported reliably over the network, can be transported at *any* positive rate (bytes per second), and will end up safely on another computer disk. It is in this sense that such transfers are elastic. This is in contrast with *stream* traffic, for example, interactive voice, or live video. Such traffic sources have *intrisic temporal patterns*, and hence cannot be controlled, i.e., cannot be asked to slow down or speed up[1]. This paper is entirely

[1]It is interesting to note here that, in some cases, the coders at such sources can produce high or low bit rate outputs (with varying degree of quality) in response to the bandwidth availability in the network. However, even then these sources cannot be considered to be arbitrarily elastic.

about elastic traffic.

It is clear that elastic sources are eminently suitable for transport over networks in which the available bandwidth is time varying. The Internet transports elastic flows at time varying rates for two reasons: (1) When several elastic connections share a link in the network, the link bandwidth must be *fairly shared* between these connections; hence, as the number of active connections changes, the fair rate for each one of them also changes with time. (2) The Internet is more and more being pressed into the transport of guaranteed bandwidth connections (e.g., Internet telephony) that need to be carried at a higher priority than elastic data; as the number of such connections varies with time, the bandwidth available to the elastic sessions also varies.

Since the bandwidth available to an elastic connection varies with time, there has to be a *feedback control* mechanism in the network by which the sources can vary their sending rates in response to the time varying available bandwidth. In this paper, we begin by discussing two approaches for the implementation of such feedback control in packet networks in general. In ATM (Aysnchronous Transfer Mode) networks, an explicit feedback based approach is used. In the Internet, the ubiquitous TCP protocol uses implicit feedbacks for such control.

As mentioned above, bandwidth sharing for elastic flows is intimately related to some notion of fairness. Given such notions of bandwidth sharing, a model for elastic transfer request arrivals, and a model for the amount of data each such transfer requires, we have a model that can be used to engineer the Internet. The notion of *network engineering* here is much the same as the one that is well known in the context of telephone networks. In telephone networks, given a model for the network, we are interested in the probability that a call is blocked, and the network is engineered (by trunk sizing) so that the call blocking probability is less than, say, 1%. In the same way, using an engineering model for the Internet, we can expect to engineer the Internet to provide a minimum transfer rate for elastic flows.

We will discuss results from the analysis of such a model for a simple network, such as that would be present at a small Internet service provider (ISP). The results immediately imply the need for bandwidth management controls if the load exceeds the engineered levels, and overload controls if the offered load exceeds the network capacity. Finally, we will present some results from actually implementing such controls at the Internet access link to an academic campus in India.

2 Feedback Controls for Elastic Traffic

In Figure 2, we show a simple scenario in which several hosts need to download files (say, using the World Wide Web (WWW) protocols; formally called *http*) from one of the web servers. Typically, the hosts would be on a high speed local area network (LAN), and would be connected to the web site by a wide area network (WAN). The WAN is shown here simply as a single link, depicted as a *bit pipe*. Let the bit rate of this link be R bits per sec (bps) (typically links have the same rate in both directions; but we are concerned here in the direction from

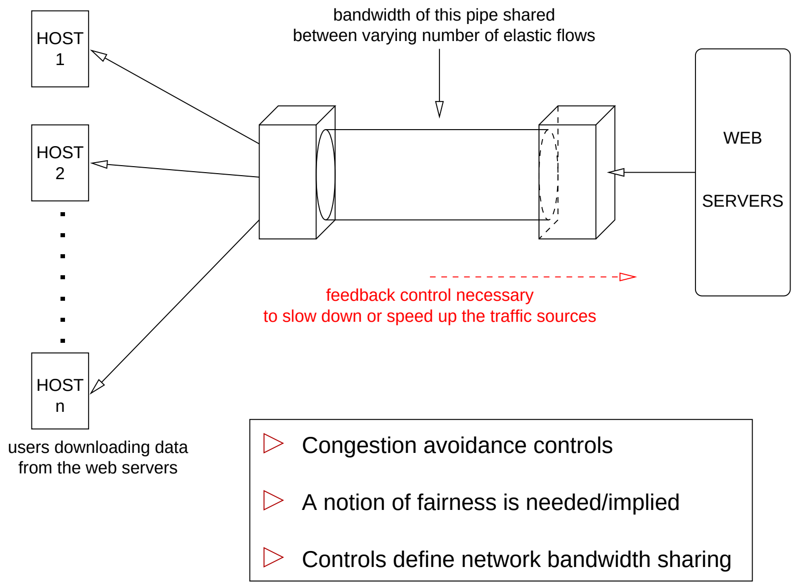

Figure 2: A simple illustration of the need for feedback congestion controls for elastic traffic.

the server to the *client* hosts).

Suppose Host 1 begins to download a file. Since the host is on a high speed LAN, the bottleneck for this transfer will be the WAN link. The host can then expect to obtain a file transfer rate of R bps. Now, suppose that before Host 1's transfer finishes, Host 2 requests a download (for the same or different web server). Suppose that nothing is done, then even if Host 2 gets a small positive transfer rate ϵ, the link will need to carry $R + \epsilon$. This is not possible, and hence there will be data loss at the router between the web server and the link. Hence, clearly, when Host 2 starts its transfer, the rate given to the transfer to Host 1 should reduce. Furthermore, a natural requirement is that the rates should be fairly allocated to the two hosts. Thus when n hosts are each downloading data, the average rate that each should get should be $\frac{R}{n}$.

In general the servers would not know of each other's transfer requests (in fact, the servers would be geographically separated, unlike the simple example here). Hence, in order for the transfer rates to adapt, some kind of feedback needs to be provided to the traffic sources. Such feedback should result in the sources adjusting their rates to share the bandwidth fairly. We notice that bandwidth sharing between elastic flows is intimately linked to these feedback based congestion controls, and the notion of fairness they implement. This should be seen in contrast to the telephone network, where bandwidth sharing between various sessions simply entails the end-to-end allocation of a fixed amount of bandwidth to each session (e.g., 64Kbps (Kilo bits per second) to each voice call).

3 Fair Allocation of Bandwidth: Concepts and Techniques

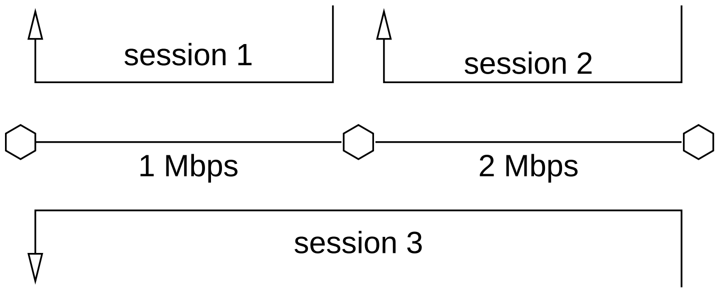

Figure 3: An example demonstrating max-min fair bandwidth sharing in a network.

In the example of Figure 2, owing to the simple topology, it was very clear what fairness meant; fairness meant equal allocation. Consider, however, the scenario in Figure 3. Three sessions, 1,2, and 3, share the bandwidth in a two link network. The link speeds are 1Mbps (Mega bits per second) and 2Mbps. Session 3 transits over both the links, Session 2 over the 2Mbps link, and Session 1 over the 1Mbps link.

If equal allocation was the objective, then we can obtain an allocation as follows. Starting with all flows at zero rate, increase the rates of all the flows equally, until some link saturates. In this example, this procedure will result in each session getting a rate of 0.5Mbps, and the 1Mbps link will saturate. It is easy to see that what we have is the *max-min* rate; i.e., among all the feasible rate vectors (for the three sessions), the minimum rate in any rate vector is no more than the max-min rate.

But clearly, it is inefficient to stop at this allocation. There is still spare bandwidth on the 2Mbps link. We can allocate it to Session 2, thus obtaining the feasible allocation: Session 1 = 0.5Mbps = Session 3, and Session 2 = 1.5Mbps.

The rates that result from the above procedure are called *max-min fair*. For details, see the textbook treatment in [Bertsekas and Gallager '92]. A recursive procedure for obtaining the max-min fair rate vector in a given network and session topology is evident:

Step 0 Start with all the sessions and the entire network; call this the Network.

Step 1 Increase the rates of all the sessions in the Network until some link saturates. Allocate the resulting rate to all the sessions that pass through the saturated links.

Step 2 Form a new network by removing all the saturated links and all the bottlenecked sessions, and by reducing the capacity of each unsaturated link by the amount of flow allocated to the bottlenecked sessions passing through that link. Replace Network with this network, and repeat Step 1.

The algorithm is guaranteed to stop in at most as many steps as there are links in the network, since at each step at least one link saturates or there are no more unbottlenecked sessions left. We have described, however, a centralised algorithm; i.e., the session and network topology information has to be available at some central place that computes the fair rates. This is clearly not practical in real networks considering their large extent, high link speeds, and propagation delays. A distributed algorithm is essential. In the next section we describe two approaches for implementing such distributed feedback controls.

Recently, other notions of fair bandwidth sharing have been introduced and studied. One of these, that arises from an economic formulation (maximising user utility subject to flow feasibility) is called *proportional fairness*; see [Kelly et al. '98]. In some network toplogies, proportional fairness is equivalent to max-min fairness.

3.1 Network Algorithms: Explicit Feedback

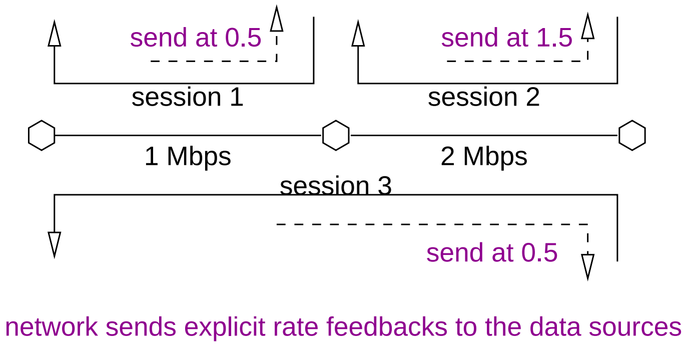

Figure 4: Fair rate allocation by explicit feedback from the network.

Figure 4 shows the same network and session topology as in Figure 3. Notice that we have shown that the source of each session is being provided with an *explicit* feedback of the fair rate of that session. Evidently, in this approach the network explicitly participates in the computation of the fair rate. Also this computation should be done in a distributed fashion, by the mutual cooperation of software running in the switches and the end systems. We describe the conceptual basis for one technique for carrying out such a distributed computation.

Figure 5 shows associated with each link l a number η_l. The values shown are $\eta_1 =$ 0.5Mbps, and $\eta_2 = 1.5$Mbps. The specific values depend on the problem instance. There is an η for each direction of flow on each link; since in the example all the flows are from "right" to "left", the η's here correspond to this direction of each link. Notice that the fair rate r_s for session s is given by

$$r_s = \min_{(j \in \text{links used by } s)} \eta_j$$

Let us now suppose that, in the network, somehow these link parameters have been computed. Each source can then determine its fair rate in a distributed way as follows. Each source sends

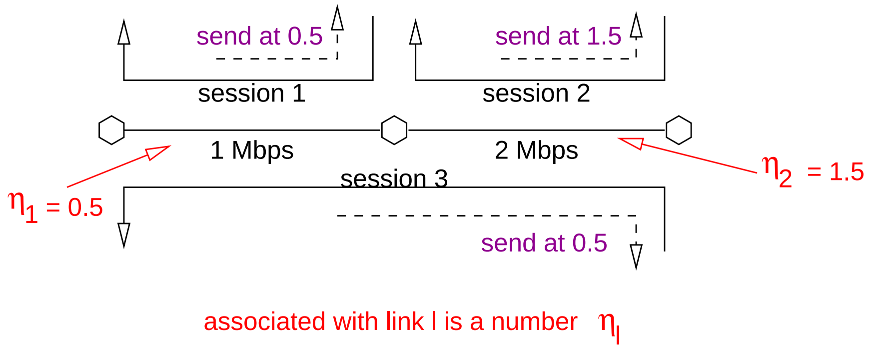

Figure 5: Each link has an associated link parameter η_l, from which the max-min fair rates for each session can be computed.

a control packet in which one field is set to a large value. As the packet passes through each switch in its path, that switch replaces the value in this field with the minimum of the local η and the existing value in the field. When the packet reaches the destination node, it is looped back. Notice that when the packet reaches the source again, it carries the minimum of the η values in the path of the source, and hence the fair rate for the source.

It remains to produce distributed algorithms that compute the η values for the links. This is an important research problem. A survey is provided in [Bonomi and Fendick '95]; an important seminal algorithm is presented in [Charny '94]; an algorithm that has seen a lot of real implementation and experimentation is reported in [Jain et al. '95]. Our work, which addresses the important issues of the effects of propagation delays, and time varying available link capacities (owing to high priority stream traffic) is documented in [Abraham and Kumar '97a, Abraham and Kumar '98a, Abraham and Kumar '98b, Abraham '98]. Our approach yields algorithms that operate at each switch and use only local information; i.e., no information needs to be explicitly exchanged between between the various switches in order to compute the η values. Each switch only needs to measure the aggregate elastic flow through each exit link, and estimate the available capacity (for elastic flows) at each such link. A simple asynchronous algorithm working independently at each switch then succeeds in iteratively calculating the link η values. Note that the coupling between the algorithms running at the switches is by the way the η values are used to obtain the max-min fair rates (see above), and by the sources actually sending at these rates. Our approach has the advantage of not requiring per flow computations at each link, or per flow updates on the control packets that pass through a link.

3.2 Network Algorithms: Implicit Feedback

In Figure 6 we show another approach to controlling the sending rates of elastic sessions. The network does not explicitly participate in the feedback control. The boxes marked "TCP" in Figure 6 represent software running in each source device that autonomously adjusts the rate

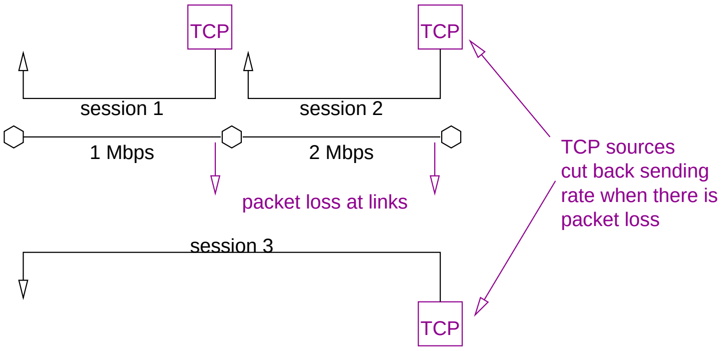

Figure 6: Fair rate allocation by implicit feedback, caused by acknowledgements and packet losses.

at which source sends, in response to *implicit* feedbacks. Normally when a packet succeeds in getting through the network, the destination sends an acknowledgement back to the transmitter. The TCP software interprets an acknowledgement of successful data as an indication of bandwidth availability in the network, and slightly increases its sending rate. If the aggregate data rate at any link is more than the link's transmission rate, then data loss occurs. This results in the lack of acknowledgement for the lost data, which causes the TCP software to drastically reduce its sending rate.

The term TCP used above is the short form for Transmission Control Protocol, a portmanteau protocol in the Internet, that takes care of the following functions:

- Recovery of lost data, and resequencing: The basic packet transport in the Internet can lose or reorder packets. TCP recovers and sequences the packets before delivering them to an application that uses TCP.
- Sender-Receiver flow control: TCP allows a slow receiver (e.g., a mechanical printer with a small memory) to slow down a fast transmitter to avoid data loss.
- Congestion control: This is the topic that we have been discussing at length in this paper.

TCP carries out its functions by means of a window based transmission protocol. All data is sequence numbered, and at any time there is a sequence number (the left window edge) upto which the transmitter knows that all data it sent has been received. There is also a window upto which the transmitter can send new data beyond the left window edge. The flow control and congestion control functions are carried out by making this window time varying. The rate of data transmission is reduced by reducing the window size. Since new data is generated in response to acknowledgements, roughly, the effective data transmission rate is given by the window size (in bytes) divided by the round trip delay in the connection. Thus window adaptation indirectly results in rate adaptation. For details on the TCP protocol see [Stevens '94].

The design approach behind TCP is in keeping with the general Internet philosophy of a simple network connecting intelligent terminals. TCP expects little from the network, and is itself a very complex protocol that runs in the end-systems. Owing to its simplicity and its lack of assumptions, TCP basically hunts around for the optimal sending rate, and keeps overshooting (resulting in data loss) or undershooting (resulting in link starvation). In a sense, it implements a kind of "bang-bang" control. Since TCP does not know what rate to send at, it builds up its window slowly; if there is loss, the rate is drastically cut and then rebuilt slowly. Since the rate build up is driven by acknowledgements, the rate can only build up over round trip times. These factors contribute to TCP being conservative, and inefficient, and *unfair* to sessions that have large round trip delays. The latter effect occurs since connections with longer round trip delays need larger windows, and take longer to rebuild these large windows when there is data loss. For various experiments and analyses that demonstrate these performance aspects of TCP, see [Kumar '98a, Lakshman and Madhow '97, Padhye et al. '98].

4 An Engineering Model

A packet data network is a traffic handling system, and in this context "engineering the system" means to design and size the system to handle a certain amount of load (offered traffic) so as to meet some quality of service (QoS) requirement for what is being transported. For example, in the Internet context, assuming that only store and forward applications (e-mail, file transfers, and WWW) are carried, the offered load may be expressed as the rate of arrival of transfer requests between all pairs of sources and sinks, and the distribution of the volume of the transfer at each request. For example, λ_{ij} may denote the rate at which files are requested to be transferred from node j to node i. The volume of each transfer may be represented by the random variable X with distribution function $F(\cdot)$. This then specifies the network offered load. The QoS may be specified simply as: the average throughput of a transfer should be at least τ_{min} Bps (Bytes per second). Given an Internet topology, and routing within the network, the problem then is to determine if the offered load can be carried with the required QoS.

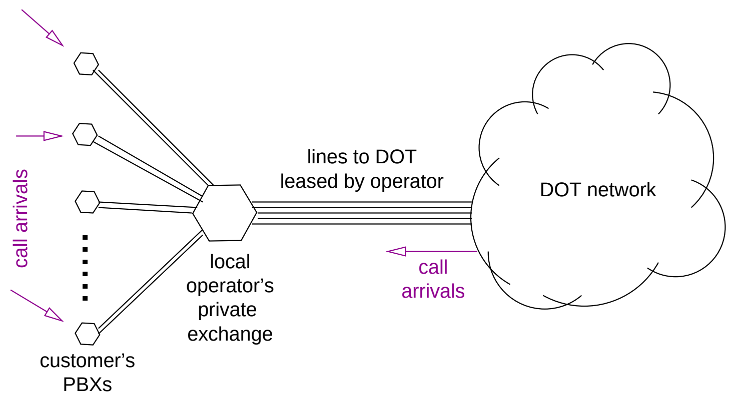

Figure 7: A network model for a private access provider for telephone service.

A very similar problem arises in the telephony context. Consider Figure 7 where we show the scenario of a small basic phone service provider's network. The provider sets up a small telephone exchange, and leases some telephone channels to the public phone network (in India the DOT (Dept. of Telecom) network). Some telephone channels then connect the provider's customers to this small exchange. The customers in turn distribute the service within their campuses using PBXs. The problem now is to determine the number of channels connecting the provider's exchange to the phone network and to its customers. Phone calls can be between the phone network and the subscribers, or between the campuses of the subscribers. A mathematical model is now needed to proceed with this network engineering problem. The following are the elements of the model:

- Call arrival model: The arrival rates of calls between the various pairs of points in the network are needed. Typically a Poisson arrival model is used for each of these arrival processes.
- Bandwidth sharing model: Each call between a pair of points is allocated one channel in each link along the route between that pair of points.
- Call holding time model: Each call holds its allocated channels for a certain random amount of time and then terminates. Typically, it is assumed that the call holding times are exponentially distributed; this assumption is, however, not even mathematically necessary for solving the fixed routing model.

The QoS objective is to keep the probability of call blocking below some small number, say 1%. Given a network topology, the number of channels on each link, and the routing plan, the above is a well formulated problem whose solution has been known for decades. Hence, by and large, the above problem is considered to be a well studied and tractable one (see, for example, [Bertsekas and Gallager '92] and [Ross '95]).

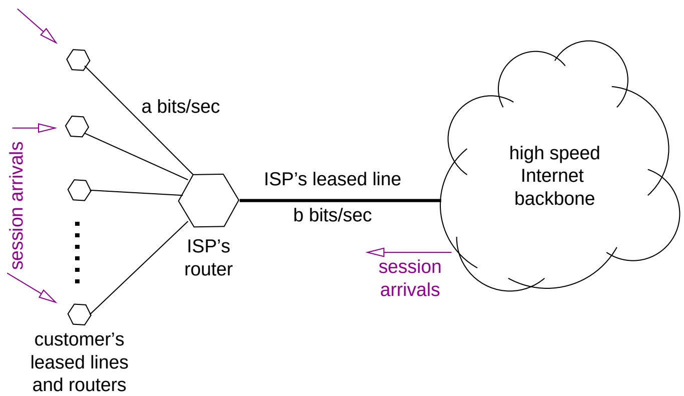

Figure 8: A model for a small internet service provider (ISP).

Let us now turn to the problem of engineering a small Internet Service Provider (ISP). A typical small ISP is depicted in Figure 8. The ISP leases a high speed link (say 2Mbps, or

34Mbps) to a higher level (bulk) ISP, who presumably has a very high speed backbone. Then our small ISP offers service to its customers via local leased lines or dial up. Requests for transfer of files arrive from the local customers, who want to download files from the Internet, and from the users on the wide area Internet, who want to download data from the sites hosted by the ISP's customers. A certain minimum average throughput is required for such transfers. We now need to analyse the performance of the ISP's network. The problem is similar to the one seen above for the phone network, with some important differences. The elements of the engineering model now are:

- Session arrival model: E-mails and file transfers (via ftp) result in arrivals of requests to transfer individual files. The way in which the WWW is used is, however, somewhat different. Users start a WWW session, during the course of which they may download several short files, looking at them and thinking in between, and then occasionally download large files. To obtain a first cut model, we can simplify things and model the process of arrivals of all individual file transfer requests (including those within a WWW session). The model for arrival epochs of transfer requests between each source and sink can be taken to be Poisson; for more on this approach, see [Paxson and Floyd '95].
- Bandwidth sharing model: Assuming fixed routing, and an ideal protocol that enforces perfect max-min fair bandwidth sharing, we can assume that given any configuration of sessions at an instant, the rate that each session gets is the max-min fair rate at that instant. When there are large propagation delays, a large number of sessions with small amounts of data to transfer, and the protocols use slow-start, this model could be quite inaccurate.
- Session volume model: Each transfer request is modeled as having a certain file size, which is sampled from a distribution, independently of any other transfer. Note that this model does not translate to a session holding time model, as the rate at which the the file is transfered is randomly varying.

The QoS objective may be to ensure that the average transfer rate over any route is more than, say, τ_{min} Bps. This model, which has been the subject of some recent research activity, is known to be hard to analyse. In [De Veciana et al. '99] the authors have proved that (for exponentially distributed session volumes) the process of the number of active sessions is stable if the offered load on each link is less than the link rate. In [Kherani '99] some approximations have been developed for analysing session throughputs in small example networks, and the effect of propagation delays and slow start have been studied by simulations. For a single link, a different arrival model involving a finite number of users alternating between file transfers and "think" periods, has been studied by [Heyman et al. '98] and [Berger and Kogan '98]; they also assume the ideal fair bandwidth sharing in the network.

For the simple network shown in Figure 8, however, calculations can be easily done if the number of customers is allowed to grow to infinity. It is as if each new session is started by a customer on an access link that is not already carrying a session. Let us assume that all transfers are from the Internet to the ISP's subscribers (this is usually the direction in which the ISP's access link will congested). We denote the ISP's Internet access link speed by b bps, and the customers' access link speed to the ISP by a bps.

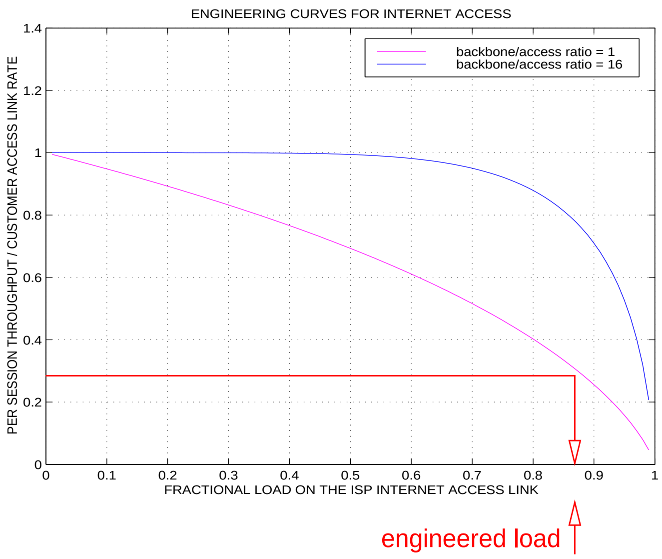

Figure 9: Session throughput vs. access link load; curves obtained from the simple engineering model. The bent line with an arrow shows an example of maximum engineered load for given minimum session throughput.

Figure 9 shows two example curves obtained from the analysis of the model. We plot the average throughput per active session (normalised to a), defined by ($N(t)$ is the number of active sessions at time t, and $I_{\{N(t)>0\}} = 1$ if $N(t) > 0$, and $= 0$ otherwise)

$$\lim_{t\to\infty} \frac{1}{t} \int_0^t \frac{I_{\{N(u)>0\}}}{N(u)} du,$$

versus the normalised load on the backbone link. Note that the normalised load (ρ) on the backbone link is simply

$$\rho = \frac{1}{b} \times \text{session arrival rate (per second)} \times \text{mean session volume (bits)}.$$

The ratio b/a is seen to be important; shown are curves for $b/a = 1$ and $b/a = 16$; $b/a = 1$ applies to all cases in which the access link cannot be the bottleneck, as would be the case when all the users are on a high speed LAN attached to the ISP router. The formula for $b/a = 1$ is obtained from the well known processor sharing model (see [Gelenbe and Mitrani '80]), and is given by

$$\frac{1-\rho}{\rho} \ln \frac{1}{1-\rho}$$

Notice that for $b/a > 1$ the curve is flat upto a larger value of ρ. Hence if an ISP has a large Internet access bandwidth (say 2Mbps) and its customers access it at low speeds (say 64Kbps), then the ISP can operate its access link at high occupancies and yet offer its customers a high average transfer rate. Note, however, that for large b/a an operating point close to 1 is more sensitive to small variations in the load.

We make the following observations from Figure 9. The average throughput per active session decreases as ρ increases, dropping to 0 when $\rho \to 1$. For $\rho \geq 1$ the system is unstable, in the sense that the number of active sessions will grow to infinity. In practice, of course, the number of users is finite; if more transfer rate is requested than the bottleneck rate b, the users will simply get backlogged, and will not be able to generate any more requests. It follows that if a certain minimum transfer rate is required, then the offered load cannot exceed some maximum value (see Figure 9). Further, if the offered load is allowed to approach or exceed 1, the users will experience very poor throughput, and certain overload behaviour may manifest itself: users abandoning partially transferred files, TCP connections timing out, etc.; see [Massoulie and Roberts '98]. These arguments motivate the need for some form of traffic control at the TCP connection level and at the aggregate flow level.

5 Edge Based Connection Control: An Experience

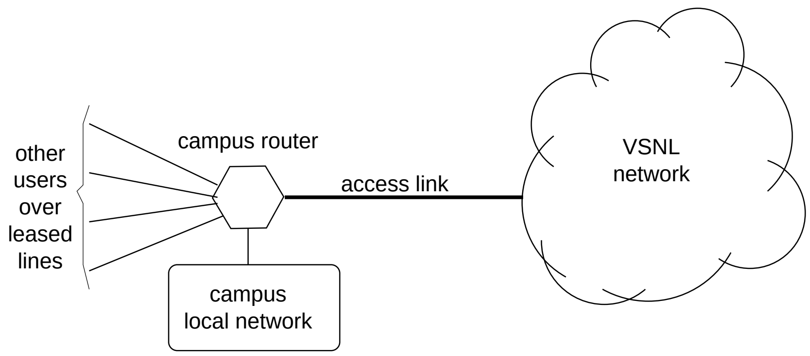

Figure 10: Local network and wide area Internet access in a campus.

Figure 10 show the Internet connectivity at the Indian Institute of Science campus. An access link connects the campus, via a router, to a major ISP (in the figure, VSNL is shown). Users access this bandwidth either via the campus LAN, or via serial links (dial-up or leased). Notice that the network is conceptually very similar to the one shown in Figure 8. There are several 100s of active users on the campus, and unless adequately sized, the Internet access link could become a bottleneck.

The access link into the IISc campus is at present just 128Kbps. Figure 11 shows the 24 hour occupancy of the access link (on February 17, 1999), in the direction into the campus. Observe that, except for an hour or two between 5am and 10am, and a period in the evening (meal times in the campus messes!), the link occupancy is close to 100% (the fact that it does not show 100% is probably because our measurement is not counting some low level link header bytes that are attached to every packet sent on the link). Clearly, the link experiences

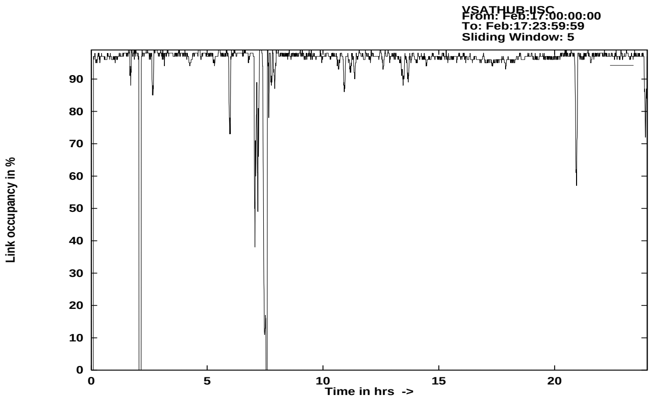

Figure 11: Internet access link occupancy without overload controls.

sustained overload during much of the day. Thus one can expect that, for much of the day, the detrimental effects of operating near $\rho = 1$, as pointed out at the end of Section 4, should be evident. This is indeed the case, as we will show from actual measurements presented below.

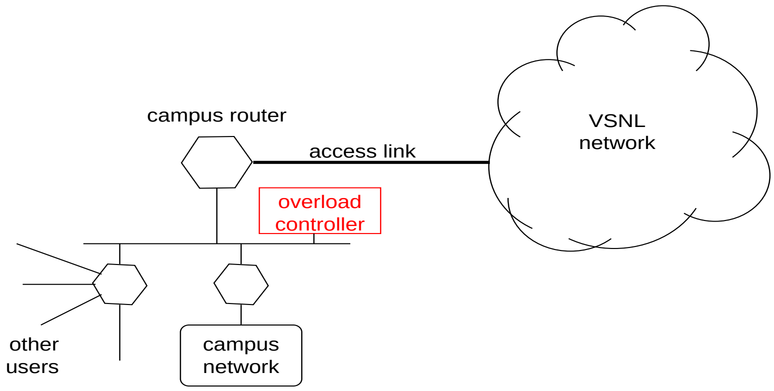

Figure 12: A nonintrusive overload control device.

In order to control the traffic, we adopted a nonintrusive approach shown in Figure 12. We have developed a software package ([Kumar et al. '98b]) that monitors the access link occupancy, the amount of access bandwidth used by various services and IP address groups (as configured by us), and takes control actions if certain levels of utilisation are exceeded. This software runs on a computer labelled "Overload Controller" in the figure. Notice from Figure 12 that the traffic between the Internet and the campus users does not pass through the controller device. Instead, the controller sits on a broadcast network (simply an Ethernet hub), and can listen to all the packets flowing between the users and the Internet. The controller sends poll packets to the routers at either end of the access link; the responses to these packets carry traffic statistics collected at these routers. By picking up packets off the broadcast local network, the controller can look at individual packets (passing between the users and the access link) and classify them according to whatever categories we set (e.g., one category can be: "e-mail packets arriving from outside the campus for a user who is on the campus LAN"). One function of the software is to record and display the statistics collected in this way.

We also set policies based on access link utilisation. For example, we can set the maximum utilisation of the access link to be 90%; further, we can say that users on the campus LAN will get a minimum of 60Kbps (in the inbound direction, out of the 128Kbps) during the day. Out of this, 25Kbps will be guaranteed for e-mail, and the rest can be used by WWW and file transfers. When the actual traffic flow violates these allocations then the connections are controlled. New TCP connections are disallowed, and existing connections are made to slow down their transfers.

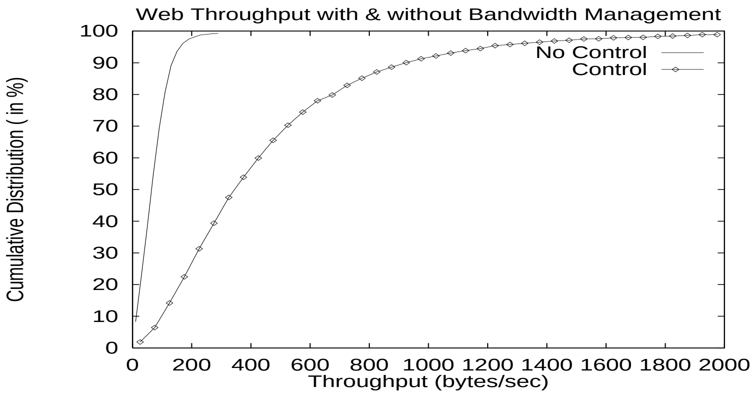

Figure 13: Cumulative frequency distribution plot of web transfer throughputs with and without overload controls.

In Figure 13 we show a result of implementing our controls on WWW throughputs. The figure shows the cumulative distribution of web throughputs with and without controls. The data was obtained from a web proxy[2] through which all web transfers take place. Notice that without control more than 95% of the web transfers get a throughput of less than 0.2KBps (note that 128Kbps = 16KBps), whereas with control the throughputs range upto 2KBps. Without control the average throughput is about 75Bps, whereas with control the average is about 450Bps. Further observations from the web proxy show that, if there are no controls, then although the link carries more traffic, much of the additional traffic it carries does not contribute to "good througput". This is mainly because there are a lot of partial transfers, either because of users abandoning their transfer requests or the connections timing out.

Another important observation that we made is that, without control, users on the campus LAN are able to use more of the access link bandwidth than users that are attached via the low speed leased or dial-up lines (see Figure 10). With our bandwidth management controls, we are able to allocate any desired portion of the access link bandwidth to any subset of users.

[2]A *web proxy* is a machine that obtains data from a web server on behalf of a requesting client. An important advantage of such an approach is that the data can be copied into a *cache* in the proxy, just in case anyone else requests for the same data. Proxies are also an important point where access controls can be exercised.

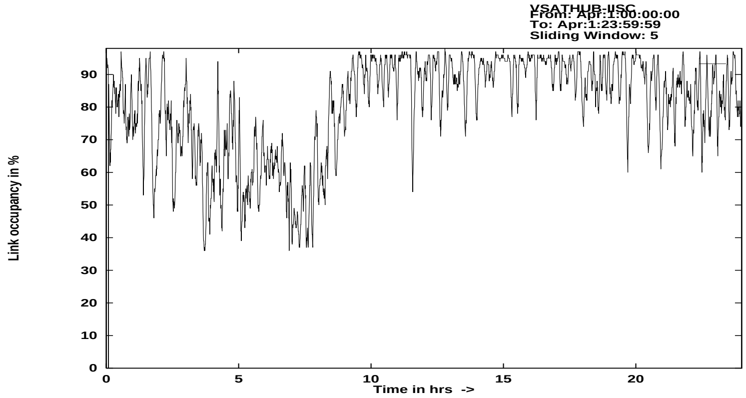

Figure 14: Internet access link occupancy with overload controls in place.

In Figure 14 we show the occupancy of the link when the controls are implemented. The occupancy now varies between more reasonable values.

As mentioned above, connection control is exercised by connection blocking and by slowing down ongoing connections. With the current level of offered load, and the 128Kbps link, during peak hours on weekdays, we need to block upto 60% of the connections. There is thus a tradeoff. Without overload control, there is no connection blocking, and users get the satisfaction of always “getting through”, but then suffer very low throughputs or stalled connections. Also the overloaded link carries a lot of useless data. On the other hand, if connection controls are exercised, depending on the level of overload, some percentage of connections get blocked, but those that get through experience good throughputs. Further, the link is efficiently utilised. Obviously, no network should need to operate for long with a high percentage of blocking. If an operator makes bandwidth allocations, and then finds that the blocking probabilities begin to approach, say, 1% then it should be taken as an indication that bandwidth is insufficient, and either bandwidth reallocation or increase in the basic access bandwidth is needed. Our observations, however, clearly demonstrate the performance of elastic transfers in the Internet in a situation of severe overload, and the success of our approach in mitigating the situation.

6 Conclusion

In this paper we have considered the Internet as primarily transporting elastic traffic. We have traced through a complete understanding of the issues involved in controlling elastic traffic in the Internet, and engineering the network for such traffic. Starting from congestion control of elastic traffic, and its implications for bandwidth sharing, we have described network engineering models that incorporate such bandwidth sharing and that demonstrate the possibility of network overload. We have shown by actual experience the effects of network overload, and the results of our attempts to mitigate network overload in our campus.

Unlike the phone network, very little work seems to have been done on large scale modelling and control problems in the Internet. The framework presented in this paper should help to put all the associated problems and issues in context.

References

[Abraham and Kumar '97a] Santosh P. Abraham and Anurag Kumar, "Max-Min Fair Rate Control of ABR Connections with Nonzero MCRs", Proceedings IEEE *Globecom'97*, pp 498-502

[Abraham and Kumar '98a] Santosh P. Abraham and Anurag Kumar, "A Stochastic Approximation Approach for Max-Min Fair Adaptive Rate Control of ABR Sessions with MCRs", *IEEE Infocom'98*, San Francisco, Mar-Apr, 1998.

[Abraham and Kumar '98b] Santosh P. Abraham and Anurag Kumar, "A Simulation Study of an Adaptive Distributed Algorithm for Max-Min Fair Rate Control of ABR Sessions" *Proc. CCBR'98*, June 1998, Ottawa, Canada.

[Abraham '98] Santosh P. Abraham, "Asynchronous Distributed Rate Control Algorithms for Best-Effort Sessions in Integrated Services Networks with Minimum Rate Guarantees", *Ph.D. Thesis*, ECE Department, Indian Institute of Science, Bangalore, India, October 1998.

[Berger and Kogan '98] A.W. Berger and Yaakov Kogan, "Dimensioning Bandwidth for Elastic Traffic in High-Speed Data Networks," submitted to *IEEE Trans. on Networking*.

[Bertsekas and Gallager '92] D. Bertsekas and R.G. Gallager, *Data Networks*, Second Edition, Prentice Hall of India, New Delhi, 1992.

[Bonomi and Fendick '95] F. Bonomi, K.W. Fendick, "The Rate-Based Flow Control Framework for the Available Bit Rate ATM Service" *IEEE Network* March/April 1995, pp 25-39

[Charny '94] Anna Charny, "An algorithm for rate allocation in a packet-switching, network with feedback", *Master's thesis*, MIT, Cambridge, May 1994

[De Veciana et al. '99] G. De Veciana, T-J Lin, T. Konstantopoulos, "Stability and Performance Analysis of Networks Supporting Services with Rate Control - Could the Internet be Unstable?," *Prof. IEEE Infocom 1999*, New York, March 1999.

[Gelenbe and Mitrani '80] E. Gelenbe and I. Mitrani, *Analysis and Synthesis of Computer Systems*, Academic Press, London, 1980.

[Heyman et al. '98] D.P. Heyman, T.V. Lakshman, and Arnold Neidhardt, "A New Method for Analysing Feedback-Based Protocols with Applications to Engineering Web Traffic over the Internet," *Performance Evaluation Review*, Vol. 25, No. 1, 1998.

[Jain et al. '95] R. Jain, S. Kalyanraman, R. Viswanathan and R. Goyal, " A Sample Switch Algorithm", *ATM Forum/95-0178,* February 1995.

[Kelly et al. '98] F.P. Kelly, A.K. Maulloo, D.K. Tan, "Rate Control for Communication Networks: Shadow Prices, Proportional Fairness and Stability," *Journal of the Operational Research Sociiety*, Vol. 49, 1998.

[Kherani '99] A.A. Kherani, "Rate Based Congestion Control for Ephemeral Best-Effort Sessions in Packet Networks," Master of Engineering Thesis, ECE Department, Indian Institute of Science, Bangalore, January 1999.

[Kumar '98a] Anurag Kumar. Comparative performance analysis of versions of TCP in local network with a lossy link. *IEEE/ACM Transactions on Networking*, 6(4), August 1998.

[Kumar et al. '98b] Anurag Kumar, Malati Hegde, S.V.R. Anand, Gopi Garge, B.N. Bindu, "NETMASTER: A Nonintrusive Approach for Bandwidth Management and Service Control at a WAN Access Link to a Campus Network," Technical Report, Number ERNET-IISc-1998.2, ERNET Project, Indian Institute of Science, Bangalore, May 1998.

[Lakshman and Madhow '97] T.V. Lakshman and Upamanyu Madhow. The performance of TCP/IP for networkswith high bandwidth delay products and random loss. *IEEE/ACM Transactions on Networking*, 5(3):336–350, June 1997.

[Padhye et al. '98] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling TCP throughput: A simple model and its empirical validation. In *Proceedings ACM SIGCOMM'98*, 1998.

[Paxson and Floyd '95] V. Paxson and Sally Floyd, "Wide Area Traffic: The Failure of Poisson Modelling," *IEEE/ACM Transactions on Networking*, Vol. 3, No. 3, pp. 226-244, June 1995.

[Massoulie and Roberts '98] L. Massoulie and J. Roberts, "Arguments in Favour of Admission Control for TCP Flows," *Proc. International Teletraffic Congress 16*, Edinburgh, June 1999.

[Ross '95] K.W. Ross, *Multiservice Loss Models for Broadband Telecommunication Networks*, Springer-Verlag, London, 1995.

[Stevens '94] W.R. Stevens, *TCP/IP Illustrated, Volume 1*, Addison-Wesley, Reading, Mass., Nov. 1994.