Implementing Shortest Job First Order of Service in the Internet

[1]Ron Addie, [1]Zhi Li and [2]Don McNickle

[1]USQ, Toowoomba [2]Unversity of Canterbury, Christchurch, NZ. E-Mail: addie@usq.edu.au

Keywords: *Differential service; Internet architectures; AQM*

EXTENDED ABSTRACT

The Internet needs protocols and mechanisms to provide guaranteed quality of service. The existing Internet is surprisingly close to providing good quality for a very wide range of services, probably because the TCP protocols aim to achieve, and come to close to achieving, fair queueing, or processor sharing, whenever several flows compete for limited resources. The DiffServ architecture aims to do better than this by providing different performance standards for different classes of service. The obvious way to apply DiffServ is to allocate classes in accordance with the urgency or priority of the requests.

However, another approach is to use DiffServ to allocate service classes according to the "size" of the requests – smaller requests receiving generally better service and longer requests worse.

A Pareto distribution with small shape parameter has been used in a great deal of research, and in this paper, to model the widely accepted heavy-tailed nature of flow lengths. We assume that the starting times of these flows forms a Poisson process.

It was shown in (McNickle and Addie(2005)) by means of a queueing model with this traffic that serving flows in order of job size (in bytes, shorter flows served first) leads to significantly lower mean and standard deviation for response times, for flows of all lengths. This paper also provided evidence that it is unlikely that DiffServ can achieve a significantly better result. This poses the challenge of how to arrange for flows to be served in the shortest-job-first (SJF) order, or as close as possible to it.

We define a simple approach to achieving this which can be implemented locally – in just two routers in the simplest case. Simulations have been used to demonstrate that some of the benefits of the SJF discipline can, indeed, be obtained.

In this paper we propose a protocol for achieving an approximation to the shortest job first order of service at times of congestion. The proposed mechanism is scalable and *local* in the sense that the actions taken are confined to a small number of routers near the site of the congestion. This concept of a *local* protocol modification can be viewed as a generalization of Active Queue Management. Whereas AQM's are generally formed by modifying the queueing discipline and ensuing behavior at the congested node, in our proposal, which we shall term *local QM (LQM)*, some nearby nodes also assist in managing the congestion, with the objective of approaching as close as possible to SJF.

SJF is sometimes not achievable to acceptable accuracy for reasons which have nothing to do with the treatment of packets by the nodes near where congestion is occurring. In such situations we cannot expect our mechanism to do the impossible. For example, if the sending host is unable to deliver a flow at the maximum rate of the bottleneck link, it will not be possible to serve this flow ahead of all others, and so SJF will not be achievable.

Two similar LQM strategies have been tested and compared in the context of a local premises network connected to the Internet via a congested link. In one of these strategies, packets are marked at the edge router are dropped or remarked at the gateway to the premises depending on the traffic conditions there. In the other strategy some packets are remarked and others have their ECN bits left in place.

Simulation results have been able to demonstrate that the strategy is able to produce better response times than AQM's of comparable complexity located at the edge router.

1. INTRODUCTION

Because the fundamental requests for service submitted to the Internet appear to generate demands for resources distributed with a Pareto-like tail (see for example, Crovella, Taqqu and Bestavros (1998)), the shortest-job-first queueing discipline is very close to optimal for all feasible demand profiles, i.e. even if users really wanted their middle sized jobs to achieve the tightest delay constraints it would still be virtually as good if shortest jobs were uniformly served first (McNickle and Addie(2005)).

Identifying the underlying requests by users is very difficult, perhaps impossible, however since we only need to identify these requests when there is a problematic resource constraint, any scheme which behaves in the same way at these times of resources limitation will be equivalent to a system in which the underlying requests have all been tagged and are readily identified. Furthermore, identifying problematic requests is much easier when they happen to be causing a resource shortage. Therefore, we do not necessarily need to be able to identify the underlying requests, except somewhat indirectly and under certain circumstances, in order to achieve performance similar to a system in which all requests have been identified.

In this paper we propose a protocol for achieving an approximation to the shortest job first order of service at times of congestion. The proposed mechanism is scalable and *local* in the sense that the actions taken are confined to a small number of routers near the site of the congestion. This concept of a *local* protocol modification can be viewed as a generalization of Active Queue Management. Whereas AQM's are generally formed by modifying the queueing discipline and ensuing behavior at the congested node, in our proposal, which we shall term *local QM (LQM)*, some nearby nodes also assist in managing the congestion, with the objective of approaching as close as possible to SJF.

SJF is sometimes not achievable to acceptable accuracy for reasons which have nothing to do with the treatment of packets by the nodes near where congestion is occurring. In such situations we cannot expect our mechanism to do the impossible. For example, if the sending host is unable to deliver a flow at the maximum rate of the bottleneck link, it will not be possible to serve this flow ahead of all others, and so SJF will not be achievable.

In Section 2 of this paper the real-world problem of performance degradation caused by congestion in bandwidth constrained links is described and existing work on this problem is reviewed. In Section 3, a queueing model of the problem, and the conclusion that Shortest Job First should be a very effective queue management strategy in routers, are presented. In Section 4, an approach to implementing the shortest job first queue management strategy is presented, and in Section 5 a simulation experiment which tests this implementation and compares it to the queueing results are given. Conclusions are drawn in Section 6.

2. BOTTLENECK BEHAVIOUR

The performance of services using the Internet will be affected significantly by the way in which congestion is handled at all of the routers along the end-to-end path traversed by packets used to provide the service.

Broadly speaking, we can distinguish between the links on this path which form a bottleneck and those that do not. It is expected that the majority of links will not form a bottleneck, simply because

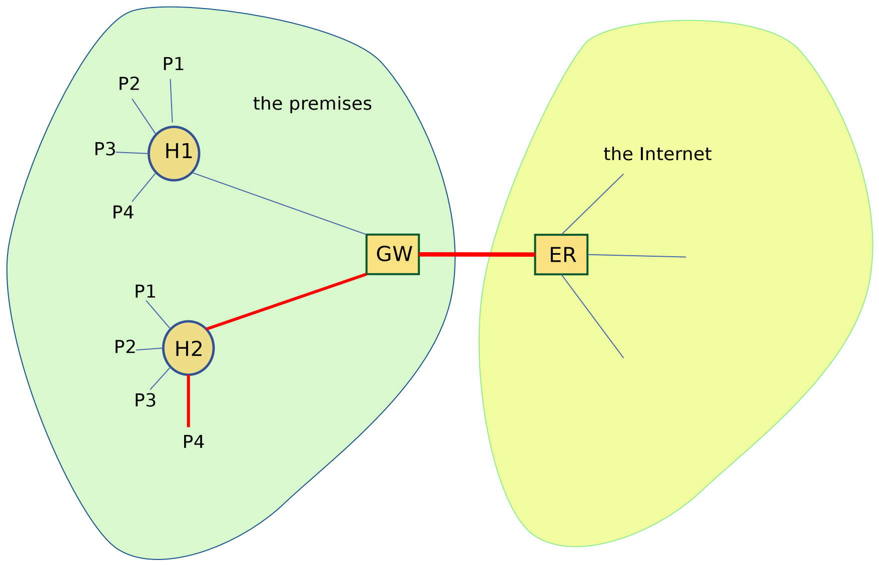

Figure 1. A premises in need of performance protection

the likelihood of the capacity at two links forming such a constraint seems low. We expect the proportion of links running near to capacity in the Internet to be low and because of TCP's congestion avoidance mechanism; even a congested link will not constrain our flow unless it happens to be the most severe constraint on the path.

However, even if a small proportion of links form bottlenecks, it is the bottlenecks which ultimately determine the performance of the whole network and there will usually be a bottleneck because TCP will normally increase the rate at which it delivers packets until congestion is encountered. We therefore need to model these bottleneck links, and that is what we have set out to do in this paper.

Because of the way the TCP protocol senses congestion in the Internet and causes sources to reduce their sending rate, requests for service are effectively held in a virtual queue, the physical implementation of which is at the sources. In order to evaluate strategies for managing congestion, it is this virtual queue we need to model.

Some natural strategies for managing this queue include: Processor Sharing, which corresponds to fair queueing, First-in-first out, which we can readily see is a poor strategy, priority queueing, where priorities are allocated according to class of service, and shortest-job-first (SJF), which has the advantage of minimizing mean and standard deviation of waiting time.

In McNickle and Addie (2005) it was shown, by means of a queueing model, that the benefits in overall reduction of waiting times (for all classes of traffic) afforded by the SJF protocol are so great, and the marginal advantages (for some classes of traffic) afforded by a priority queue strategy are so small, that the ideal queueing discipline to adopt at a congested link is SJF.

How can SJF be implemented at the virtual queue? This paper is concerned with showing how an approximation to the shortest job first discipline can be implemented and that this approximation achieves a significant proportion of the benefits of the SJF discipline.

First of all, observe that almost all the benefits of the SJF queueing discipline can be obtained so long as the queueing discipline we adopt is close, in some appropriate sense, to SJF. For example, a queueing discipline which is identical to SJF when buffers are heavily loaded and not identical when they are not, will still achieve most of the benefits of SJF, since it is only when buffers are nearly full that the queueing discipline is important.

Secondly, observe that congestion occurs in the Internet primarily in the access networks. At a rough guess, assuming the core part of the Internet is well designed, congestion is most likely to occur in the 3-4 hops closest to the source or the destination of a flow.

We need to distinguish two forms of congestion in an access network – congestion occurring at or near the source and congestion occurring at or near the destination. We restrict our attention to the more important of these, namely congestion at or near the destination (See Figure 1). The same techniques that apply here probably apply to the other case as well. In some respects the case of congestion near the source appears to be somewhat easier since the necessary congestion management actions can be taken in the premises where the traffic is emerging. However, we shall restrict our attention to the case which is of wider interest, namely the one where congestion is occurring at or near the entrance to an access network.

So, given this, let us propose a simple scheme for achieving close to the SJF discipline, which can be achieved by means not greatly different from that used in an AQM such as RED or ARED. Packets at a congested router should be marked using the Explicit Congestion Notification mechanism in basically the same manner as in the RED or ARED AQM. Instead of relying on this mechanism to achieve benefits such as better link utilization and lower queueing delay all by itself, by interacting with the TCP congestion avoidance algorithm at the source, packets with ECN bits set will be treated in a special manner at subsequent routers on the way to their destination. Packets will be differentiated on the basis of the size of their containing flow. Packets in short flows with ECN bits set will have these bits reset, whereas packets in long flows will be dropped.

It is clear that this mechanism will disadvantage flows which show up as unusually long or high in rate according to the token buckets in routers in the access network. It is not obvious that this should result in a discipline similar to SJF for serving the virtual queue of flows sharing access through a bottleneck link. We shall present an argument that this should be the case in Section 4.3.

3. A SIMPLE QUEUE MODEL

For a theoretical model of Internet performance we consider a conventional M/G/1 queue, with unlimited storage, and where the service-time distribution is given by a Pareto distribution of the form:

$$B(t) = 1 - \left(\frac{t}{t+\delta}\right)^{\gamma}, t > 0 \cdot$$

δ is the scale parameter and γ is the shape parameter. We will take $1 < \gamma \leq 2$, thus ensuring heavy-tailed behaviour, and select the scale parameter so that the mean service time is one.

In general the mean and variance of the service time are $\delta/(\gamma-1)$ and $\delta\ \gamma^2/((\gamma-1)^2(\gamma-2))$, for $\gamma > 1$ and 2 respectively. For $\gamma \le 2$ the variance, and all higher moments, are infinite. We take the traffic intensity to be 0.9. While this may seem unrealistic, it is worth considering because real systems stray into high occupancy rates for long periods, due to the long-range-dependent rather than Poisson nature of internet traffic.

We apply this queueing model to the virtual queue which exists among the sources competing for the resources of a bottleneck link which lies nearby, in the local access network.

3.1. Service Disciplines

We consider three service disciplines, FIFO (first-in first-out), PS (processor sharing) and SJF (shortest-job first.) We consider the expected response time (sojourn time) *T(x)* of a job with a service requirement of *x* (Of course it is well known (Schrage (1968)) that the shortest-remaining-processing-time is optimal for M/G/1 with respect to average response time, but this appears impractical for Internet applications, although Harchol-Balter, Crovella and Park (1998) make a persuasive argument for it).

FIFO: Because the variance of the service time is infinite it immediately follows from the Pollaczek-Khinchine formula (Gross and Harris (1998) p.212) that *T(x)* will be infinite regardless of the service requirement. Even if the service-time distribution is truncated at, say, 100, then *T(x)* = 6.68 + x for a traffic intensity of ½ and $\gamma = 1.5$. This is in spite of the fact that the mean service time is one, and that 50% of the jobs are not delayed. Thus FIFO is a very unsatisfactory and unfair discipline.

PS: A well-known, but still surprising result is that $T(x) = x/(1-\rho)$ where ρ is the traffic intensity. That is, processor sharing gives linear discrimination in response time regardless of the service time distribution. Processor-sharing type disciplines have long been proposed as suitable "fair" models for Internet traffic (see, for example Parekh and Gallager, (1993)). Generalized Processor Sharing, where the rate for each job is a function of the number in the system, has been shown to accurately describe the flow-level characteristics of traffic on Internet access lines (Beckers, Hendrawan, Kooij and van der Mei, (2001))

SJF: We consider a *preemptive-resume* discipline. That is, when a job with shorter processing time arrives it interrupts the job in process. After the higher priority job is finished the interrupted job needs only to complete the balance of its service time.

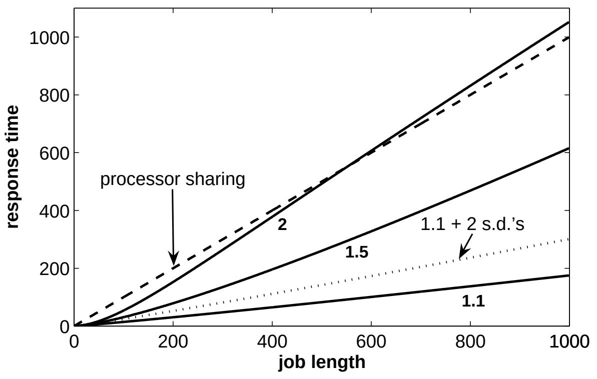

Figure 2. Processor Sharing vs SJF for γ = 2, 1.5, 1.1

3.2. Comparing SJF and PS for the queue model

Our objective here is to show that SJF is very hard to beat, and in fact is uniformly superior to PS for very heavy-tailed traffic. Taking the limit of the expressions given in Takagi (p.346) for the preemptive-resume M/G/1 priority queue gives:

$$T_{SJF}(x) = \frac{\lambda \int_0^x t^2 dB(t)}{2\left(1-\lambda\int_0^x t dB(t)\right)^2} + \frac{x}{1-\lambda\int_0^x t dB(t)}$$

A similar but more complicated expression follows for the variance of the response times

With the assistance of Maple we can plot the mean response times for a particular service requirement. In Figure 2 these are compared with the mean response times for processor sharing for shape parameters $\gamma = 2$, 1.5, and 1.1.

The basic conclusion is that the smaller the shape parameter (i.e. the more heavy-tailed the job distribution) the greater the advantage of shortest job first over processor sharing. For values of the shape parameter below 1.687 the expected response times for SJF are uniformly smaller than those for PS.

So SJF offers a substantial advantage over PS for almost all jobs, with the advantage increasing with the degree of heavy-tailedness of job lengths. In fact for $\gamma = 1.1$ the expected wait for the SJF discipline is lower than that for PS even when a "design margin" of two standard deviations is included in the SJF values. This is graphed in Figure 2 as the dotted line labelled "1.1+2 s.d.'s". The advantage of SJF over PS also increases with traffic intensity.

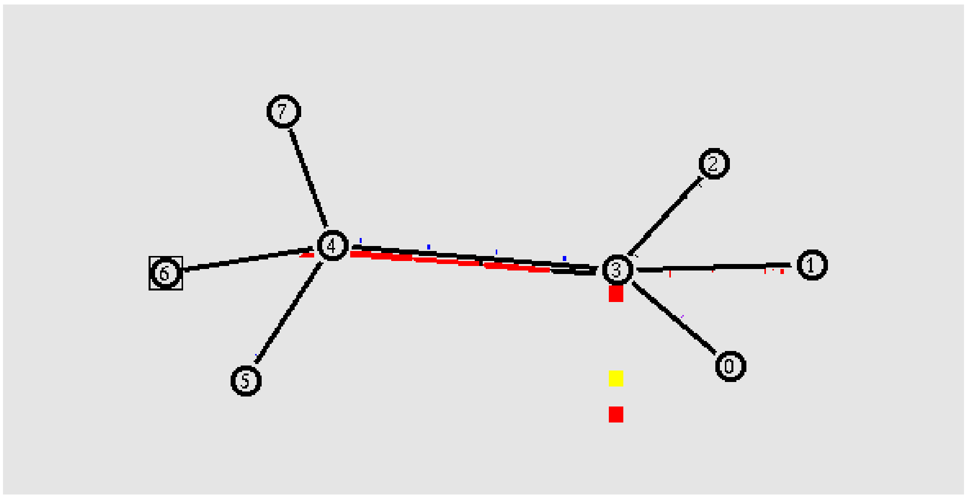

Figure 3. The experimental setup.

An interesting feature of the DiffServ proposal (Blake et al (1998)) is the ability to give preferential service to one class over all others, for example to delay-sensitive traffic. We have investigated the M/G/1 model under a variety of other priority disciplines. These all suggest that any discipline other than SJF results in serious deterioration of service for the short jobs, which constitute the majority of Internet traffic. The gain in performance for longer jobs cannot compensate for this.

4. IMPLEMENTATION

We would like to demonstrate that mechanisms for implementing an approximation to the shortest-job-first (SJF) queueing discipline are feasible. Since we will not attempt to implement SJF exactly, we need to argue that the protocol we implement is sufficiently close to SJF that it gains a significant proportion of the benefits of SJF. Simulation, using Network Simulator Version 2 (Fall and Varadhan (1997), McCanne and Floyd. (2005)), will be used to demonstrate this.

Since the objective of this work is to develop a strategy which achieves as much or more than a change of Internet architecture, like DiffServ, by means of relatively minor adjustments to the behaviour of routers, we have attempted to implement an approximation to SJF in a purely local manner. This protocol will be referred to as ECN with Dropping (ECND). The other variation relies on hosts implementing a protocol stack which responds to ECN. It is known as ECN with Censorship (ECNC).

4.1. ECN with Censorship (ECNC)

The ECNC protocol works as follows. Consider the network depicted in Figure 1. The hosts in the premises on the left suffer poor performance because the link between the premises and the Internet is congested. This congestion is caused by communication between one process, P4, at host H2. The fact that this particular flow is causing congestion can be identified at the gateway (GW), but action needs to be take at the Edge Router (ER). We assume ER has implemented RED with random marking of packets by ECN bits whenever the buffer level exceeds a certain level. These ECN bits are then reset at GW if they are not in the problem flow. Thus, in the end, the task of marking packets to indicated congestion to sending hosts is carried out jointly by ER and GW.

4.2. ECN with Dropping (ECND)

ECN with dropping is not necessarily better than ECNC, but it does not rely on sources responding to congestion indication bits, and it is *safer* than ECNC in that packets in problem flows are dropped by one of the routers in the path of the flow and near the congestion. The desired effect of ECNC can be bypassed by hosts which fail to implement an appropriate TCP/IP stack, but ECND cannot be bypassed because packets in the problem flow are dropped.

Instead of packets being dropped at ER, which is what would happen with today's Internet protocols, we assume that ER implements RED with ECN marking and the packets which are marked in this way are dropped at GW. The dropping at GW occurs only for packets in the problem flow. In this way, the two locations where the existence of the problem and the identity of the problem flow can be identified join forces to take action. Two locations might not be sufficient to identify the problem flow(s). However, the idea of resetting ECN bits in non-problem flows, or dropping them in problem flows, is still valid if a larger number of routers are involved.

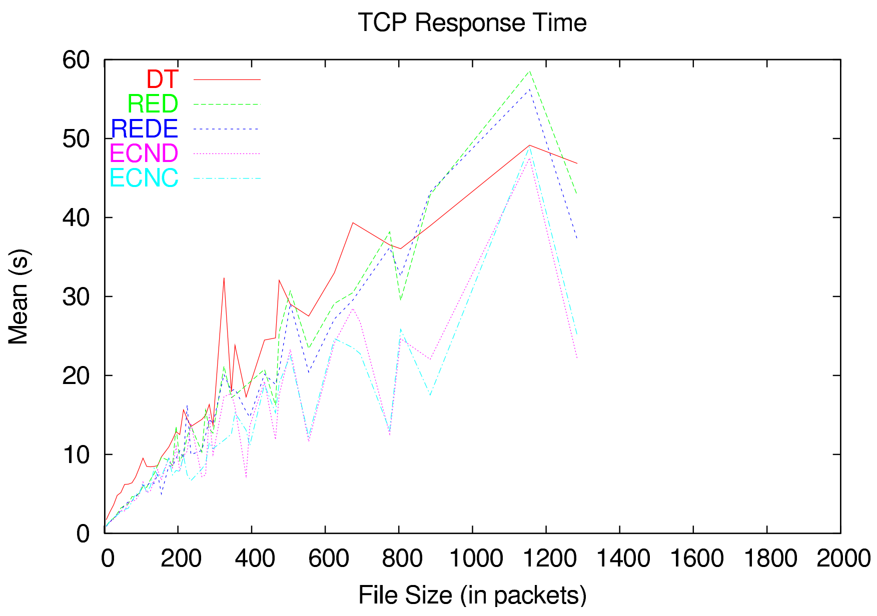

Figure 4. Mean Response Time for Droptail (DT), RED, RED with ECN (REDE), ECNC and ECND.

4.3. Why ECNC and ECND are similar to SJF

If these protocols are able to maintain a short buffer at ER and ensure that hosts generating short flows receive no packets with congestion indicated, and no packets are dropped at ER or GW, then the total delay experienced by packets in these flows will be close to the minimum possible by any strategy only effected at these nodes. The fact that no packet losses are generated ensures that the flow is achieving the greatest rate sustainable, given all other network conditions. The fact that the buffer at ER is kept at a low level ensures that competition for resources with other flows does not impact significantly on this flow.

It is to be expected that the boundary between short and long flows will fluctuate over time, depending upon the total level of traffic, and the traffic mix. Precisely how to distinguish between long flows and short flows has not yet been established.

5. A SIMULATION EXPERIMENT

A series of experiments have been carried out with the objective of testing the feasibility of the hypothesis that a satisfactory approximation to SJF can be achieved by means of Local Queue Management.

5.1. A Simulation Experiment

In the simulations, the task of distinguishing between short and long flows was achieved by foreknowledge – cheating. The experimental setup is shown in Figure 3 (which is a snapshot from an NS2 simulation).

Congestion is occurring in the middle link. This can be addressed in a variety of ways – by dropping packets at the node at the head of this link, by marking packets so that they can be dropped later, and so on. Instead of dealing with this problem purely at Node 3, we intend to deal with it at Node 3 *and* Node 4 – a local subnetwork.

This has been achieved in this test network by using RED with marking at Node 3 and at Node 4, we take one of three actions: (i) leave the packet as is (which will cause a congestion indication flag to be sent to the source) (ii) drop the packet (which will cause TCP to back off from what it perceives as congestion); or (iii) reset the marking on the packet (allowing the source to continue at full rate).

5.2. Results

The simulations were monitored in a variety of ways. The statistic of most importance in the present case is *response time*, i.e. the delay between a flow starting, at the host, and being fully received, at the destination. A plot of response times using various different approaches is shown in Figure 4. It is clear that the proposed approach, in its two forms, ECNC and ECND, is able to deliver better response times than RED with or without the use of Explicit Congestion Notification (ECN) bits.

In this experiment, a subset of flows sufficient to load the link to 50% of its capacity was selected for favourable treatment. These flows were of length from 1 to 2000 packets. When the ECNC protocol was in use, any packets in the favoured flows tagged with an ECN flag were un-flagged. Other ECN packets were allowed to pass in the usual way. When ECND was in use, the favoured flows were treated in the same way, but packets in the other flows would be dropped at the GW node (node 4).

In a "proper" implementation of ECNC or ECND it would be necessary to identify which flows should be favoured dynamically. The desired proportion of favoured flows might need to change dynamically, and also the scheme by means of which flows are distinguished (the flow rate and depth of a leaky bucket). In the present instance, the purpose of the experiment is to identify if it is feasible to achieve the gains potentially offered by SJF order of service.

The simulations required to obtain the results shown in Figure 4 required a considerable amount of computation time on a high performance computer, even though we have not attempted to generate confidence intervals at this stage. The improvement in response time shown in Figure 4 is significant, but not as dramatic as predicted from the queueing theory. Before further simulations are undertaken it is necessary to develop a theory which can predict more precisely how much response time can be improved, under a variety of circumstances, a which can provide guidance concerning how to set the parameters for the queue management method.

6. CONCLUSIONS

We have demonstrated on the basis of both simulation and queueing models that the Shortest Job First queueing discipline among competing flows offers considerable advantages over simple Processor Sharing for the kinds of traffic that can be expected to be encountered in the Internet and that if traffic continues to become even more heavy-tailed, this advantage will increase. We have, furthermore, investigated how something approximating SJF can be implemented.

On the other hand it appears that priority queue strategies are risky, in that the marginal advantages gained by those jobs that benefit from the priority scheme are very small, and the majority of jobs can expect to receive worse service. This suggests that not only is it the case that a more global architecture for differential service is difficult, but also there is a significant risk that the cost in reduced performance for subsets of traffic is not warranted.

7. REFERENCES

Beckers, J.,I. Hendrawan, R. Kooij, R. van der Mei, (2001) Generalized processor sharing models for Internet access lines, *9th IFIP Conference on Performance Modelling and Evaluation of ATM and IP Networks*, Budapest.

Blake, S, D. Black, M. Carlson, E. Davies, Z. Wang and W. Weiss, (1998) *An architecture for differentiated services*, IETF, RFC 2475.

Crovella, M., M, Taqqu and A. Bestavros (1998). Heavy-tailed probability distributions in the World Wide Web. Robert J. Adler, Raisa E. Feldman, Murad S. Taqqu (eds.), *A Practical Guide To Heavy Tails*. 1, 3--26. Chapman and Hall, New York.

Fall, K. and Varadhan, K. (1997) ns Notes and Documentation. Technical report, UC Berkeley, LBL, USC/ISI, and Xerox PARC.

Gross, D. and C. Harris (1998) *Fundamentals of Queueing Theory*, 3rd ed. John Wiley, New York.

Harchol-Balter, M., M. Crovella and S. Park (1998) The case for SRPT scheduling in web servers. *Technical Report No.. MIT-LCS-TR-767*, MIT Lab for Computer Science.

S. McCanne and S. Floyd. (2005) *ns Network Simulator*. [Online]. Available: http://www.isi.edu/nsnam/ns/.

McNickle, D. and Addie, R. G. (2005) *Comparing Protocols for Differential Service in the Internet*, IEEE TENCON, Melbourne.

Parekh, A. and G. Gallager (1993) A generalized processor sharing approach to flow control in integrated services networks: the single node case. *IEEE/ACM Trans. on Networking*, 2, 137-150.

Schrage, L (1968) A proof of the optimality of the shortest remaining processing time discipline *Operations Research*, 16, 678-690.

Takagi, H. (1991) *Queueing Analysis: A Foundation of Performance Evaluation, Vol 1, Vacation and Priority Systems Part 1*, North Holland.