On the Scalability of Fair Queueing

A. Kortebi — France Télécom R&D
L. Muscariello* — Politecnico di Torino
S. Oueslati — France Télécom R&D
J. Roberts — France Télécom R&D

ABSTRACT

The scalability of fair queueing depends on the number of flows that have, or recently have had, packets in the queue. Applying known facts about statistical traffic characteristics at flow level, we demonstrate that this number is typically very small for any link capacity. This suggests fair queueing may be perfectly feasible opening interesting possibilities for the development of the Internet architecture. The evaluation is based on trace simulations and an analytical traffic model.

1. INTRODUCTION

We consider the use of fair queueing to control link bandwidth sharing between flows identified "on the fly" from packet header fields. This is clearly not a new topic and many scheduler designs have been proposed since the pioneering work of Nagle [9]. The potential benefits of imposing fairness in the network remain very important, however, and continue to stimulate a large amount of research on possible realizations.

The main objective of the present paper is to study the scalability of fair queueing in the light of current understanding of the nature of traffic on a high capacity backbone link. It is frequently stated that per-flow scheduling is infeasible since the number of *in-progress* flows can attain tens or hundreds of thousands and that this is too many. In fact, fair queueing schedulers operate on the very small proportion of these flows that are currently *active* in a precise local sense that is made clear later. We show that the number of active flows is typically measured in tens, or at worst hundreds, and does not depend on link capacity.

The number of active flows is a stochastic process whose behaviour depends mainly on the overall traffic intensity and the individual flow rate characteristics. We study the stationary distribution of this process using simulation and some analytical modelling. This evaluation highlights the important distinction between flows that are bottlenecked at the considered link and those, the large majority, that have an exogenous rate limit. Fair queueing is scalable and arguably feasible because the number of bottlenecked flows is typically very small.

Network assured fairness would have important architectural consequences that merit deeper consideration. For example, fair queueing would allow latency critical applications to share resources with bursty data transfers without suffering excessive jitter. Transport protocols, notably for high speed data transfers, could be developed without the current requirement to be "TCP-friendly".

We first recall salient characteristics of IP traffic at flow level before discussing possible realizations of fair queueing. We then present an evaluation of the number of flows that need to be taken into account by the scheduler, using simulation and analysis. Some consequences of the derived results on implementation are then discussed. In the conclusion we suggest a number of outstanding issues for future research.

2. FLOW LEVEL CHARACTERISTICS OF IP TRAFFIC

A flow for present purposes is the set of packets observed at a given link that relate to a particular document transfer or communication. We assume this flow can be identified from header fields (e.g., the usual 5-tuple) and the fact that inter-packet time is less than a certain threshold. We focus in this paper on TCP flows. They have finite size and occur according to a certain arrival process. Traffic intensity, equal to the product of the flow arrival rate and the average flow size, is a global measure of demand. Realized performance (e.g., as measured by flow response times) depends significantly on how this intensity compares to link capacity.

A reasonable model of the arrival process is to assume flows belong to mutually independent sessions that arrive according to a Poisson process. To assume Poisson session arrivals has sound theoretical and empirical justifications and has been recognized as one of the rare invariants of Internet traffic [4]. A session is assumed to consist of an alternating sequence of flows and think times [2]. This sequence can have quite general characteristics, regarding the size of flows and their number,

*Work done while visiting France Télécom R&D.

for instance, depending on the underlying applications. We refer to this as the *Poisson session model*.

The characteristics of individual flows have been catalogued according to a variety of criteria including size, rate, duration and burstiness [3, 16, 12]. The most significant characteristic for present purposes is the exogenous flow rate: the rate the flow would have if the considered link were of infinite capacity. The measurement study by Zhang *et al.* [16] shows that the rate varies widely from flow to flow. Most flows have a relatively low average rate but the fastest flows count for a significant proportion of bytes.

The scalability of fair queueing depends on the number of flows whose (exogenous) rate is greater than the current fair rate since these are the flows that actually need to be scheduled. This is not an intrinsic traffic characteristic but depends on the nature and intensity of demand and on available link capacity. To understand this dependence, we perform trace simulations and some mathematical modelling in Section 4 below.

We used a trace from the NLANR site[1] collected on August 14 2002 on the westbound IPLS-CLEV OC48 Abilene link. We extracted a 1 minute sequence (9:20:40–9:21:40) comprising more than 4 million packets in 111000 5-tuple flows. Average link utilization is a low 16%. The packet size distribution is approximately tri-modal with 46% around 40 bytes, 6% around 576 bytes and 48% around 1500 bytes. We observe approximately the same distribution of flow rates as in [16] with a maximum of around 10 Mb/s. Assuming a flow is in progress from the arrival epoch of the first packet to the expiration of a 20s time out after the last, the average number is around 20000. Some 98% of bytes are in TCP flows.

3. COMPLEXITY OF FAIR QUEUEING

We discuss mechanisms and algorithms for achieving approximate fairness between flows identified "on the fly", distinguishing fair queueing schedulers and more approximate active queue management (AQM) mechanisms.

3.1 Scheduling

Low complexity schedulers can ensure that the amount of data transmitted by a backlogged flow never differs from the ideal amount by more than one maximum sized packet. We choose to exemplify this class of schedulers by the Start-time fair queueing (SFQ) algorithm [5]. Deficit round robin provides similar guarantees [13].

An adaptation of SFQ to on the fly flow detection is summarized in the pseudocode of Figure 1. The complexity of this algorithm is determined by the operations that add and remove flows from ActiveList and by the sort operation implicit in line 9. We define flows included in ActiveList to be *active flows*. The complexity

[1]http://pma.nlanr.net/Traces/long/ipls1.html

```
1.   on arrival of l-byte packet p of flow f:
2.   if f ∈ ActiveList do
3.      TimeStamp.p = FinishTag.f
4.      FinishTag.f += l
5.   else
6.      add f to ActiveList
7.      TimeStamp.p = VirtualTime
8.      FinishTag.f = VirtualTime + l

9.   transmit packets in increasing TimeStamp order

10.  at the start of transmission of packet p:
11.     VirtualTime = TimeStamp.p
12.     for all flows f ∈ ActiveList
13.        if (FinishTag.f ≤ VirtualTime) remove f
```

Figure 1: Pseudocode of SFQ with a dynamic list of active flows.

of the sort in fact depends only on the number of flows in ActiveList whose first packet has a time stamp different to VirtualTime. This is the number of *bottlenecked flows* whose packets are transmitted at the current fair rate.

In any realization, ActiveList has finite capacity and may already be saturated when a new flow should be added at line 6. It is important therefore to specify the default treatment for a packet that cannot be scheduled correctly. We assume the packet is emitted as if it acquired VirtualTime as time stamp. This has no consequence if the flow is not bottlenecked. A bottlenecked flow will be incorrectly regulated until one of its packets does trigger its inclusion in ActiveList.

3.2 Approximate fairness

A number of AQM mechanisms have been proposed with the objective of realizing approximate fair sharing. These mechanisms are based on dropping packets with a certain probability determined from an estimate of the current rate of the flow to which the packets belong.

With CSFQ [14], the rate is assumed to be measured precisely at an edge router and included in a specific header field. Complexity is thus minimal at the core router where sharing is performed. However, implementation requires changes to the IP header and must be performed for an entire network. FRED [7], AFD [11] and RED-PD [8] all require a list of flows and their complexity depends on the size of this list. AFD and RED-PD seek to identify just the bottlenecked flows. There is however no discussion in the cited papers of how accurately this set can be identified among the changing population of in-progress flows.

3.3 Pros and cons

A major advantage of AQM is that packets that are not dropped are transmitted via a simple FIFO. On the other hand, the complexity of maintaining virtual scheduling queues with linked lists is not necessarily an obstacle if the number of active flows is small.

The precise fairness of schedulers is not crucial for elastic flows. However, a significant side benefit is that fair queueing also allows useful guarantees for latency critical flows (see Section 5.2).

The performance of AQM mechanisms is less clearly understood than that of schedulers and tends to depend critically on chosen parameter values.

4. IN-PROGRESS, ACTIVE AND BOTTLENECKED FLOWS

In this section we evaluate the distributions of the numbers of in-progress, active and bottlenecked flows using trace simulation and analysis.

4.1 Trace driven simulations

Using the Abilene trace described in Section 2, we have performed simulations of a link handling this traffic using the SFQ algorithm of Fig. 1. We vary the capacity of the link to account for a range of loads.

With a capacity of 2.5 Gb/s, there is no real queueing but the SFQ algorithm identifies flows contributing to busy periods in the FIFO queue of the Abilene router and populates the active list appropriately. However, given the low link utilization of 0.16, the number of packets in the trace busy periods is very small, never exceeding 7. When link capacity is reduced, certain trace busy periods coalesce leading to increased scheduling activity. The exogenous rate of some flows is then greater than the current fair rate leading to rate reduction by the scheduler. Buffers are sized to avoid loss due to this induced congestion.

We have performed simulations with link capacities of 2.5 Gb/s, 1 Gb/s, 500 Mb/s and 450 Mb/s corresponding to utilizations of 0.16, 0.41, 0.84 and 0.93, respectively. Figure 2 presents the empirical complementary distribution of the active list size for each simulation. Even under a heavy load of 0.93, the number of active flows never exceeds 215, a value much smaller than the number of in-progress flows of around 20000.

4.2 A traffic model for homogeneous flows

To explain the difference in the numbers of in-progress, active and bottlenecked flows, we introduce the following simple model. Flows are generated according to the general Poisson session model described in Section 2 and share a link of capacity C. We assume all flows emit data at the same constant rate and, to simplify the presentation, we choose this rate to be C/m with m an integer. Let the number of in-progress flows be

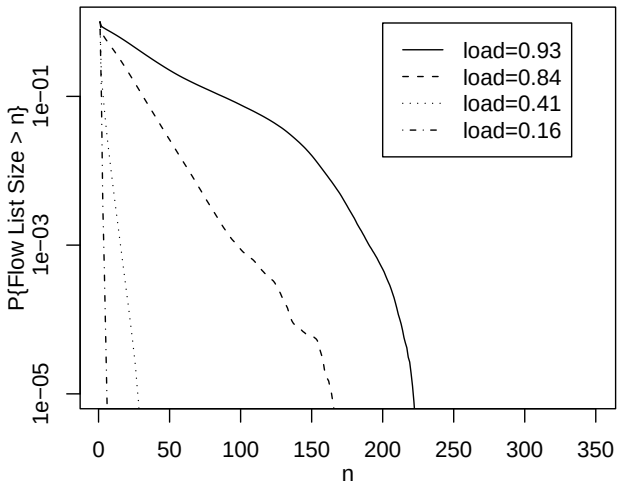


Figure 2: Complementary distribution of the active list size - trace simulation, link utilizations: 0.16, 0.41, 0.84, 0.93.

N_m. When $N_m \geq m$, the SFQ scheduler imposes that each flow realizes the current fair rate on output. We assume the transport protocol is perfectly efficient in ensuring each flow is able to emit at this rate.

The above assumptions correspond to the fluid traffic model considered by Ben Fredj *et al.* [2]. The distribution of the number of in-progress flows depends only on the link load ρ (flow arrival rate × mean size / C) and the rate parameter m. Denoting this distribution by $\pi_m(n)$, we have:

$$\pi_m(n) = (1-\rho)f(\rho) \times \begin{cases} \frac{m!}{n!}(m\rho)^{n-m}, & \text{for } n < m, \\ \rho^{n-m}, & \text{for } n \geq m, \end{cases} \tag{1}$$

where

$$f(\rho) = \frac{(m\rho)^m/m!}{(1-\rho)\sum_{k=0}^{m-1}(m\rho)^k/k! + (m\rho)^m/m!}.$$

Let A_m represent the number of active flows when a new entry needs to be added to this list (triggered by a packet arrival when $n < m$ or a new flow arrival when $n \geq m$). To derive the distribution of A_m, we assume all packets have the same size. Considering the operation of SFQ, when N_m is less than m, the number of active flows is non-zero only in a busy period of the packet queue. During the busy period, A_m increases by one on each packet arrival, these packets coming necessarily from distinct flows, and is reset to zero at the end of the busy period. When N_m is greater than or equal to m, the active list always contains exactly N_m flows.

The distribution $b_m(i,n)$ of the number of packets in a busy period can be derived from known results for the nD/D/1 queue [15]. We have, for $1 \leq n < m$ and

$1 \le i \le n$:

$$b_m(i,n) = \binom{n-1}{i-1} \frac{m-n}{m-n+1} \frac{i^{i-2}(m-i)^{n-i-1}}{m^{n-2}} \quad (2)$$

Let $d_m(j,n)$ denote the probability of the event $A_m = j$, given $N_m = n$. We have:

$$\begin{aligned} d_m(j,n) &= (1 - \frac{n-1}{m}) \sum_{i=j+1}^{n} b_m(i,n), 0 \le j < n < m \\ & \quad 1, n \ge m \text{ and } j = n, \text{ or } j = n = 0 \\ & \quad 0, \text{ otherwise.} \end{aligned}$$

The first line derives from the fact that an arriving packet sees an active list size equal to the number of packets that have already contributed to the nD/D/1 busy period to which that packet belongs. By a classical renewal theory result, the probability this number is equal to j is Pr[busy period $> j$] / E[busy period length]. Finally, the distribution of the number of active flows $a_m(j)$ is derived by deconditioning with respect to the distribution (1):

$$a_m(j) = \sum_{n \ge 0} d_m(j,n) \pi_m(n). \quad (3)$$

Let B_m denote the number of bottlenecked flows. B_m is equal to n with probability $\pi_m(n)$, for $n \ge m+1$, and is 0 with probability $\sum_{0 \le i \le m} \pi_m(i)$.

Noting that A_m is unbounded, any finite active list size leads to some probability of overflow. We would like to know how big the list needs to be to satisfy some upper limit on this probability. We therefore choose to represent the performance of this traffic model in terms of a remote percentile of the distribution.

Figure 3 plots the 99.9% percentile of the distributions of N_m and A_m against link load ρ for a range of rate parameters m. When $m = 1$, all in-progress flows are active and bottlenecked and N_1, A_1 and B_1 have the same geometric distribution. The percentile of this distribution is represented as the continuous line in Fig. 3 (a) and (b). We do not plot results for B_m since the percentile is near zero except when ρ is very close to 1.

The results illustrate two important points about fair queueing:
1) Scheduling is *scalable* in that the numbers of active and bottlenecked flows do not increase with link capacity C. For a fixed exogenous flow rate and a given link utilization, the percentile for bottlenecked flows decreases as C increases while that of active flows tends to a limit.
2) Scheduling is *feasible* up to high load (utilization $< 90\%$, say) since the number of active flows remains small. However, this number explodes as utilization approaches 100% under demand overload.

4.3 Bottlenecked and non-bottlenecked flows

To more closely model real traffic, it would be necessary to account for the wide range of exogenous flow rates. Modelling such a system is extremely difficult, however, and to derive useful formulas we must make some simplifications. Specifically, we assume there are just two classes of traffic: flows of the first class are always bottlenecked and are served at the fair rate; flows of the second class are never bottlenecked and are represented as a stream of variable sized packets. To further simplify, we suppose this stream is Poisson.

Let the traffic intensity of class i flows be A_i for $i = 1, 2$. A reasonable approximation for the distribution of the number of class 1 in-progress flows is derived on assuming they fairly share a constant residual bandwidth equal to $C - A_2$. The distribution is then:

$$\pi(n) = (1 - \rho_{\text{eff}}) \rho_{\text{eff}}^n \quad (4)$$

where $\rho_{\text{eff}} = A_1/(C - A_2)$. This follows on setting $m = 1$ in (1).

Given n bottlenecked flows, the number of flows in the active list can be deduced from an analysis of the busy period of the M/G/1 queue [10]. A worst case assumption for the number of active flows is that backlogged packets have maximum size MTU. In this case, VirtualTime in SFQ (Fig. 1) evolves as a multiple of MTU and each distinct value defines a "busy cycle" as illustrated in Fig. 4: the active flow list contains n flows at the start of the cycle and increases by one for every class 2 packet arrival. These flows are all effaced from the list at the end of the cycle.

The number of such class 2 flows is equal to the number of customers joining the busy period of an M/G/1 queue that starts with an exceptional customer of size $n \times$MTU. Formulas for evaluating this system are given in [10] and can be used to derive the distribution of the number of active flows, as in Section 4.2.

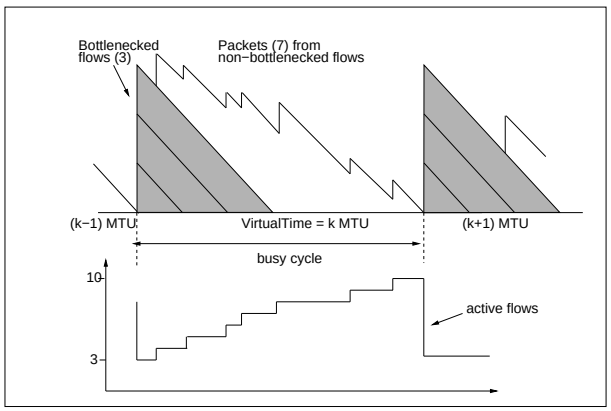
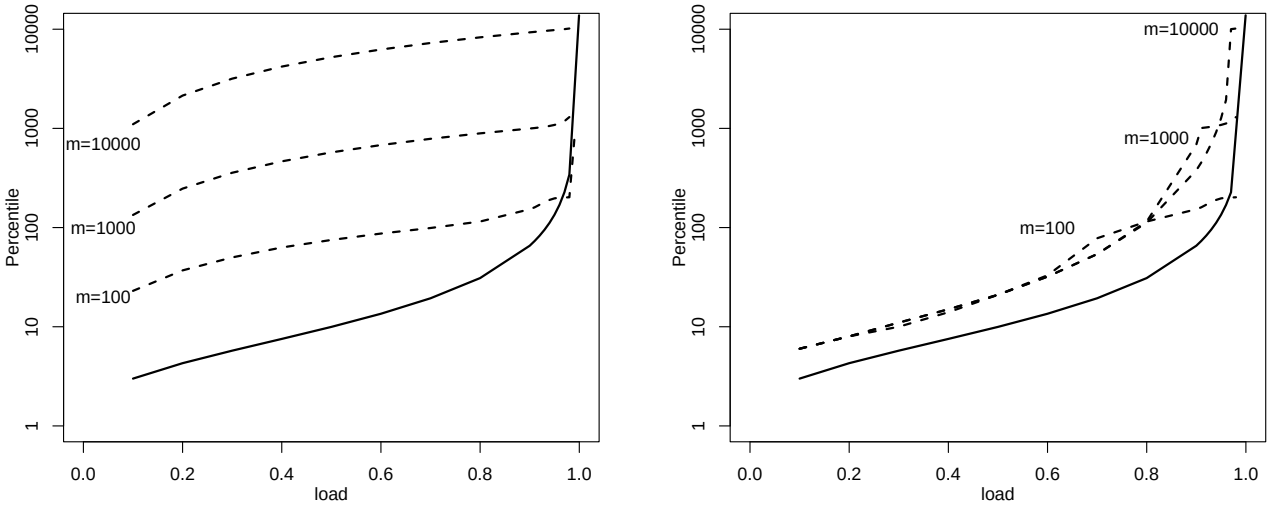


Figure 4: Illustration of the notion of busy cycle.

Using the distribution of packet sizes derived from the trace, we have evaluated this probability for the link



(a) In-progress flows N_m

(b) Active flows A_m

Figure 3: 99.9 percentiles of flow number distributions against link load: $m = 1, 100, 1000, 10000$**.**

loads considered in Section 4.1. Proportions of bottlenecked and non-bottlenecked traffic are those measured in the simulations. Fig. 5 compares analytical and simulation results. Agreement is reasonable suggesting the model provides the basis for a qualitative understanding of how bottlenecked and non-bottlenecked flows combine to determine the complexity of fair queueing.

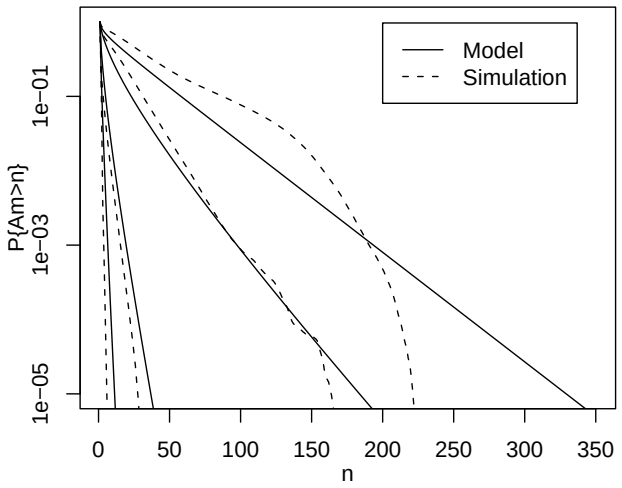


Figure 5: Complementary distribution of the active list size - analysis and simulation.

5. IMPLEMENTATION CONSIDERATIONS

In this section we attempt to draw the consequences of the above results on the implementation of fair queueing in core routers.

5.1 Sizing the active list

The results in Fig. 5 suggest that, if link utilization can be maintained less than around 90%, an active list capacity of some 200 flows could be sufficient. If it were possible to directly identify bottlenecked flows, the list capacity could be considerably smaller: by (4), assuming half the traffic intensity comes from bottlenecked flows, the number of bottlenecked flows at 90% utilization is less than 33 for 99.9% of the time.

To reduce the required active list size, the inclusion of a new flow might be based on a randomized decision. If the probability of adding a flow were 0.1, say, only 10% of the non-bottlenecked flows would be included. The bottlenecked flows, on the other hand, would be identified as such after the emission of 10 packets on average. The list size requirement for the example above would be reduced from 200 to around 50 $(33+(200-33)\times 10\%)$.

5.2 Integrating latency critical flows

The SFQ algorithm does not specify the order in which packets with the same time stamp should be served. If packets of new flows (i.e., flows not yet in ActiveList) are given priority over packets of backlogged flows with the same time stamp, their queueing delay is reduced. Noting that most audio and video flows have a relatively low rate and would not be bottlenecked, this device realizes an *implicit* service differentiation [6].

5.3 Dealing with overload

Fair queueing is scalable and arguably feasible as long as link utilization is maintained less than some threshold (90%, say). If a sustained overload occurs, however, the number of active flows will largely exceed the list capacity and the scheduler will fail. Note, however, that no AQM can satisfactorily control queueing at an over-

loaded link either. All systems tend to revert to FIFO with flow level performance deteriorating significantly.

One possible solution is to avoid overload at all cost by preventive or reactive traffic engineering. An alternative is to perform admission control at flow level [6].

5.4 Buffer requirements

It has recently been observed that the rule of thumb that router buffers should be able to store some 200 ms of data at line rate is excessive and unsustainable as link bandwidth increases [1]. The present discussion on the numbers of in-progress, active and bottlenecked flows throws further elucidates typical buffer requirements.

It is shown in [1] that buffer requirements decrease substantially as the number of in-progress flows increases, under the assumption that these flows are all bottlenecked. On the other hand, our analysis demonstrates that, at normal utilization levels, the number of non-bottlenecked flows in progress may be large, but the number of bottlenecked flows remains small.

According to the model of Section 4.3, it is quite probable that only one flow is bottlenecked (i.e., with probability $\rho_{\text{eff}}(1-\rho_{\text{eff}})$) and that it then has a very high rate. In this case, as shown by the analysis of [1], TCP would indeed require the router to store a round trip's worth of data to fully utilize the available rate.

In fact, if there are flows that can attain rates comparable to the link rate (measurements in [16] do not reveal any!), they will be obliged for the sake of efficiency to use one of the newer very high speed variants of TCP and buffer requirements would need to be re-calculated in consequence. The design of a buffer-economical, high-speed transport protocol and its introduction would, of course, be greatly facilitated if routers were equipped with fair queueing.

6. CONCLUSIONS

We believe the above analysis throws new light on the feasibility of implementing per-flow fair queueing in core routers. Fair queueing is scalable since complexity does not increase with link capacity. It is also feasible as long as link load is not allowed to attain saturation levels. This observation results from a performance evaluation taking account of known statistical characteristics of IP traffic at flow level.

The scalability results apply equally for exact schedulers like SFQ and DRR and for approximate AQM-based mechanisms. It largely remains to evaluate the optimal compromise between complexity and fairness. There is scope for simplifying the schedulers, notably by applying randomization to reduce the number of non-bottlenecked flows that are unnecessarily included in the active list.

Our analysis highlights the importance of avoiding overload since there is little any scheduler or AQM can do to alleviate the resulting flow level congestion. Per-flow admission control may be a more effective overload control than over-provisioning and meticulous traffic engineering.

Fair queueing has the side benefit of allowing performance assurances for latency critical applications. An open issue is to determine how precise these assurances can be when fairness is approximate, as with the AQM mechanisms or the randomized scheduler envisaged in Section 5.1.

The traffic models presented above can be improved and more extensive trace-based simulations are clearly desirable. The present analysis illustrates the importance of taking proper account of the stochastic nature of traffic at flow level in this evaluation.

7. REFERENCES

[1] G. Appenzeller, I. Keslassy, N. McKeown, Sizing router buffers, In Proc. of ACM SIGCOMM 2004.

[2] S. Ben Fredj, T. Bonald, A. Proutière, G. Régnié and J.W. Roberts. Statistical bandwidth sharing: A study of congestion at flow level, In Proc. of ACM SIGCOMM 2001.

[3] N. Brownlee, kc claffy, Understanding Internet traffic streams: Dragonflies and tortoises, IEEE Communications Magazine, 2002.

[4] S. Floyd, V. Paxson. Difficulties in simulating the Internet, IEEE/ACM Transactions on Networking, Vol. 9, No. 4, Aug 2001, pp 392-403.

[5] P. Goyal, H. Vin, H. Cheng. Start-time fair queueing: A scheduling algorithm for integrated services packet switching networks. IEEE/ACM Transactions on Networking, Vol 5, No 5, Oct 1997.

[6] A. Kortebi, S. Oueslati, J. Roberts, Cross-protect: implicit service differentiation and admission control, Proceedings of HPSR'04, Phoenix, 2004.

[7] D. Lin, R. Morris, Dynamics of Random Early Detection, In Proc. of ACM SIGCOMM 1997.

[8] R. Mahajan, S. Floyd, D. Weatherall, Controling high-bandwidth flows at the congested router, Proceedings of ICNP 2001.

[9] J. Nagle, On packet switches with infinite storage, RFC 970, IETF, 1985.

[10] S. Niu, R. Cooper, Duality and Other Results for M/G/1 and GI/M/1 Queues, via a New Ballot Theorem. Mathematics of Operations Research, Vol. 14, No. 2, pp. 281-293, 1989.

[11] R. Pan, L. Breslau, B. Prabhakar and S. Shenker, A Flow Table-Based Design to Approximate Fairness, in Hot Interconnects, Palo Alto, California, August 2002.

[12] S. Sarvotham, R. Riedi, R. Baraniuk, Connection-level analysis and modeling of network traffic, Proceedings of Internet measurement workshop 2001.

[13] M. Shreedhar, G. Varghese, Efficient fair queueing using deficit round robin, IEEE/ACM Transactions on Networking, Vol 4, No 3, June 1996.

[14] I. Stoica, S. Shenker, H. Zhang, Core-Stateless Fair Queueing: A Scalable Architecture to Approximate Fair Bandwidth Allocations in High Speed Networks, In Proc. of ACM SIGCOMM'98, 1998.

[15] J. Virtamo, Idle and busy period distributions of an infinite capacity N*D/D/1 queue, Proceedings of ITC 14, Elsevier, 1994.

[16] Y. Zhang, L. Breslau, V. Paxson, S. Shenker, On the characteristics and origins of Internet flow rates, In Proc. of ACM SIGCOMM 2002.