# Evaluating the Number of Active Flows in a Scheduler Realizing Fair Statistical Bandwidth Sharing

A. Kortebi     L. Muscariello[*]     S. Oueslati     J. Roberts

France Télécom R&D[†]   Politecnico di Torino[‡]   France Télécom R&D   France Télécom R&D

## ABSTRACT

Despite its well-known advantages, per-flow fair queueing has not been deployed in the Internet mainly because of the common belief that such scheduling is not scalable. The objective of the present paper is to demonstrate using trace simulations and analytical evaluations that this belief is misguided. We show that although the number of flows *in progress* increases with link speed, the number that needs scheduling at any moment is largely independent of this rate. The number of such *active* flows is a random process typically measured in hundreds even though there may be tens of thousands of flows in progress. The simulations are performed using traces from commercial and research networks with quite different traffic characteristics. Analysis is based on models for balanced fair statistical bandwidth sharing and applies properties of queue busy periods to explain the observed behaviour.

## Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Packet-switching networks*

## General Terms

Performance

---

[*]Work done while visiting France Télécom R&D.

[†]A. Kortebi, S. Oueslati, J. Roberts, France Télécom R&D, CORE/CPN, 38 rue du Général Leclerc, 92794 Issy-Les-Moulineaux, France
{abdesselem.kortebi, sara.oueslati,
james.roberts}@francetelecom.com

[‡]L.Muscariello, Politecnico di Torino, Dipartimento di Elettronica, Corso Duca degli Abruzzi 24, 10129 Torino, Italy
luca.muscariello@polito.it

## Keywords

Fair queueing, statistical bandwidth sharing, trace simulations, analytical traffic model

## 1. INTRODUCTION

The number of flows traversing a network link is a random process that varies as a large population of potential users independently initiate and complete their applications. Most flows are established to transfer some kind of digital document and can adjust their rate to make the best use of available bandwidth. The way these elastic flows share bandwidth is currently determined by end-to-end transport protocols like TCP that hopefully implement standardized congestion control algorithms. In this paper we consider an alternative approach where network mechanisms are employed to govern bandwidth sharing, without having to rely on end user cooperation.

The assumed sharing objective is max-min fairness between user flows [1]. This can be realized by performing per-flow fair queueing on all network links [2]. Users must still perform end-to-end congestion control but only to ensure they fully use the allocated rate.

The proposal to implement fair queueing is clearly not new and it has already been demonstrated that this is possible on high rate links [3]. Nevertheless, it is widely considered that generalized implementation is not scalable: complexity increases with the number of flows in progress and this number increases with link rate; required operating speed therefore increases too fast. The objective of the present paper is to demonstrate that this belief is false and that fair queueing is both scalable and feasible.

The desirability of max-min fairness as a bandwidth sharing objective has been called into question [4]. In particular, if users gain different utilities from a given bandwidth allocation, there are sound economic reasons to realize shares that reflect these differences. This argument is less compelling, however, when one takes account of the fact that the population of flows in progress varies. It turns out that, under a realistic stochastic traffic model, the differences in the performance of policies like max-min fairness and weighted proportional fairness are hardly significant and not necessarily to the advantage of those providing greater discrimination [5].

We study the scalability of fair queueing by means of trace driven simulations and analytical modelling. The traces are recorded on networks with quite different traffic characteristics. In all cases, the simulations demonstrate that the

number of flows that the scheduler needs to be aware of at any instant is measured in hundreds, even when there are tens of thousands of flows in progress. The analytical model, based on a quasi-stationary separation of flow level and packet level processes, accurately predicts the simulation results. The model shows how different traffic characteristics impact performance, highlighting the significant role of the flows' exogenous rate limits.

Fair queueing scheduling has been widely studied since the early proposal of Nagle [6]. The main focus of research has been on the development of algorithms with reduced complexity, precise fairness and provable performance bounds. For present purposes, a rough degree of fairness is adequate and we are not interested in delay bounds since there are no constraints on incoming traffic. Our objective is to evaluate the complexity of fair queueing in dynamic, random traffic. To this end we consider the specific case of Start-time Fair Queueing [7]. However, a parallel analysis could be performed to show the equivalence of alternative schedulers, like Deficit Round Robin, for example [8].

In view of the imagined complexity of schedulers, a number of active queue management algorithms have been proposed that realize approximate fair sharing [9, 10, 11]. Our analysis would also apply roughly to an evaluation of the number of flows that need to be handled by these schemes.

The characteristics of individual IP flows have been catalogued according to a variety of criteria including size, rate, duration and burstiness [12, 13, 14]. 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.* [14] shows that the rate varies widely from flow to flow. Our own analysis shows that the distribution of rates can vary significantly depending on the considered network.

We have previously claimed that fair queueing is scalable in [15, 16] and provided preliminary evidence to back up this claim. The present work goes much further in both experimental validation using trace driven simulation and in developing a comprehensive analytical model.

We first discuss relevant characteristics of IP traffic at flow level and introduce the three network traces used in the simulations. In Section 3, we describe the considered fair queueing algorithm and present simulation results illustrating its performance under the traffic of the three traces. The analytical model is developed in Section 4 and evaluated using data representative of the traces. The impact on performance of particular traffic characteristics at flow and packet level is discussed in Section 5. We also highlight a number of issues raised by the previous analysis. Section 6 concludes the paper and identifies a number of outstanding issues.

# 2. FLOW LEVEL CHARACTERISTICS OF IP TRAFFIC

We elaborate on the representation of traffic as a stochastic process at flow and session levels and introduce the traffic traces used in our evaluation.

## 2.1 Traffic as a stochastic process

IP traffic on a network link can be considered as a superposition of independent sessions, each session relating to some piece of user activity and being manifested by the transmission of a collection of flows. Assuming sessions are mutually independent and are generated by a large population of users leads naturally to a Poisson session arrival process. This characteristic has been confirmed empirically and recognized as one of the rare invariants of IP traffic [17].

Sessions and flows are defined locally at a considered network element. Flows can generally be identified by common values in packet header fields (e.g., the 5-tuple of IP addresses, port numbers and transport protocol) and the fact that the interval between such packets is less than some time out value (20s, say). It is not usually possible to identify sessions just from data in packets and this notion cannot therefore be used for resource allocation.

A useful traffic model is to suppose flows in a session are emitted one after the other, separated by "think times". The number of flows in a session, the sizes of successive flows and think times and their correlation can vary widely depending on the underlying applications (mail, Web, P2P,...). We refer to a Poisson process of sessions of alternating flows and think times with otherwise general characteristics as the *Poisson session model*. It forms the basis of the analytical model introduced later.

An obvious discrepancy with reality arises if we must identify flows using the 5-tuple of header fields. For instance, the transfer of a Web document would be considered as a single flow in the session model but is frequently realized using several distinct TCP connections in parallel. This discrepancy does not have a significant impact on the present evaluation of the complexity of per-flow fair queueing, however, as discussed later in Section 5.

A more significant flow characteristic is the exogenous peak rate at which a flow can be emitted. This is the highest rate the flow would attain if the link were of unlimited capacity. This limit may be due to the user access line capacity, the maximum TCP receive window or the current available bandwidth on other links of the path, for instance.

The complexity of fair queueing depends somewhat on packet length. IP packets are well known to have a size distribution concentrated around a few particular values such as 40, 570 and 1500 bytes [18]. This length distribution is generally correlated with flow characteristics such as the size or peak rate.

## 2.2 Statistics of trace data

Traffic characteristics used in the present study are derived from trace data for three network links:

*ADSL* : an OC3 link concentrating the traffic outgoing to several thousand ADSL users; the trace represents 5 minutes of data recorded on August 25 2003;
*Ab-I* : an OC48 link on the Abilene research network between Indianapolis and Kansas City; the trace represents 5 minutes of data recorded on August 14 2002 (10:30 to 10:35 am);
*Ab-III* : an OC192 link on Abilene III between Indianapolis and Chicago; the trace represents 2 minutes of trace data recorded on June 1 2004 (7:31 to 7:33 pm).
The Abilene traces are publicly available on the NLANR website [1]. The ADSL data is from a commercial network.

Summary statistics for these three traces are shown in

---

[1]See http://pma.nlanr.net/Traces/Traces/long/ipls/1/ and http://pma.nlanr.net/Special/ipls3.html.

**Table 1: Packet trace statistics summary**

|  | ADSL | Ab-I | Ab-III |
|---|---|---|---|
| bandwidth | 155Mbps | 2.5Gbps | 10Gbps |
| total packets | 2.6M | 19M | 156M |
| total flows | 850K | 2.3M | 683K |
| utilization | 28% | 13% | 19% |
| flows in progress | 24000 | 37000 | 62000 |

Table 1. The table refers to 5-tuple flows defined with reference to a 20s time out. These statistics demonstrate significant differences between the traces. In particular, flows are much bigger on average on Ab-III than on the other two links. This is due to a small number of large flows with a very high rate. The utilization of all three links is very low. The absence of congestion means measured flow rates can reasonably be assimilated to the exogenous rates discussed in Section 2.1. The last line of the table gives the average number of flows in progress over the trace lifetime. This number is derived on considering flows to be still "in progress" during the 20s time out. Some 95% of bytes in all traces are in TCP connections.

Figure 1 presents the empirical complementary distribution of average flow rates calculated, as in [14], for all flows lasting longer than 100ms. Flow rates in the ADSL trace are limited by the offered line rates (all less than 1Mbps). Rates in the Abilene traces can be much higher though most flows still have an average rate less than 1Mbps. The Ab-III trace is somewhat atypical in that it includes some flows with the exceptionally high average rate of around 300Mbps. These turn out to be TCP flows using 9000 byte jumbo packets.
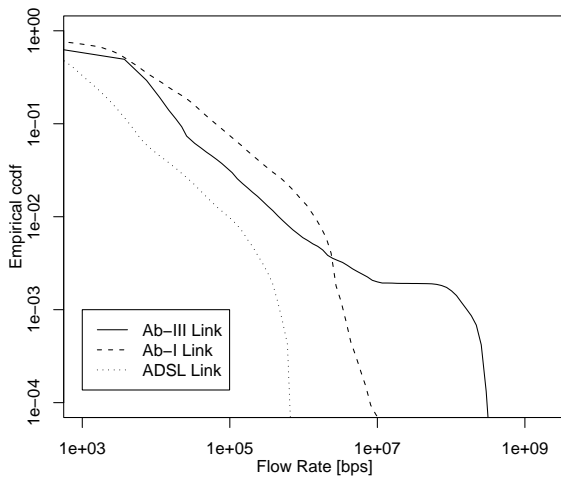


**Figure 1: Empirical complementary distribution of average flow rates.**

As an alternative measure of the exogenous rate we have measured the peak rate attained in successive 10ms intervals over the flow lifetime (the length of the interval is not critical since similar statistics were obtained using 100 ms). Results presented in Figure 2 accentuate the differences between the three traces revealing notably that flows in Ab-III can attain more than 1Gbps while rates of ADSL flows are all less than
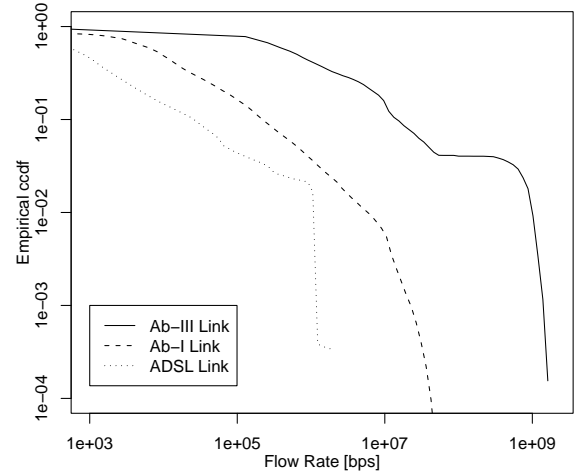


**Figure 2: Empirical complementary distribution of flow peak rates.**

1Mbps.

Packet lengths are correlated with rate for the Abilene traces. On Ab-I, all packets are less than 1500 bytes. Most packets in flows whose peak rate is greater than 10Mbps are of this length. Lower rate flows, that include flows of TCP ACKs, have a much higher proportion of 40 byte packets. Some 67% of bytes in the Ab-III trace are in flows of peak rate greater than 500Mbps. Almost all packets in these flows are of 9000 bytes. Lower rate flows in this trace have more typical packet sizes ranging from 40 to 1500 bytes. The distribution of packet sizes in the ADSL trace is clustered around 40, 570 and 1500 bytes and is broadly independent of flow rate.

# 3. COMPLEXITY OF PER-FLOW FAIR QUEUEING

We choose to exemplify the class of fair queueing schedulers by the self-clocked Start-time Fair Queueing (SFQ) algorithm of Goyal et al. [7]. We discuss how complexity of SFQ in a dynamic traffic setting depends on the number of so-called *active* flows. We then evaluate the distribution of this number using trace driven simulations.

## 3.1 Start-time Fair Queueing

Pseudocode for SFQ adapted to the present context of a dynamically varying flow population is given in Figure 3. Fairness properties of the algorithm are discussed in [7]. It is shown, for example, that the amount of data transmitted by any continuously backlogged flow is always within one maximum sized packet the ideal fair share.

The complexity of the algorithm is usually evaluated in terms of the number of operations necessary per packet under the assumption that the flow population is fixed. This complexity is determined by the sort operation implicit in line 9 of the pseudocode and thus typically increases with the logarithm of the number of flows. In the present dynamic version, the sort must be performed on flows that have a

1.     on arrival of $l$-byte packet $p$ of flow $f$:
2.     if $f \in$ 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 $f \in$ ActiveList
13.          if (FinishTag.$f \le$ VirtualTime) remove $f$

14.     if no packets in scheduler VirtualTime = 0

**Figure 3: Pseudocode of SFQ with a dynamic list of active flows.**

packet with a time stamp greater than the current value of VirtualTime. This is the number of *bottlenecked* flows since this condition only occurs when the flow incoming rate exceeds the current fair rate.

The number of bottlenecked flows is usually much smaller than the number of *active* flows. Active flows are flows with an entry in ActiveList. This includes any bottlenecked flow but also any flow that has recently emitted a packet, as manifested by the fact that its FinishTag is still greater than VirtualTime.

Pseudocode operations 2 and 12-13 depend on the number of active flows. Operations 12-13 are trivial if the list is sorted in FinishTag order, implying log complexity for a sort operation whenever a finish tag is updated. The test in operation 2 can have O(1) complexity if the size of ActiveList is small enough to be realized using content addressable memory. The overall complexity of the algorithm thus depends mainly on the number of active flows and secondarily on the number of these that are bottlenecked.

Note that, in any realization, the size of ActiveList is finite and it is necessary to limit the probability of overflow. It is thus important to understand how the distribution of the number of active flows depends on the intensity and characteristics of offered traffic. It is also necessary to specify what happens when the list is full. We assume packets of new flows that cannot be recorded are scheduled with time stamp equal to VirtualTime. List saturation thus has no impact on scheduling non-bottlenecked flows. Bottlenecked flows will not be slowed to the current fair rate until a subsequent packet arrival when the list is no longer saturated.

## 3.2   Trace driven simulations

The link utilization for the three traces presented in Section 2 is too low to produce any interesting scheduling behaviour. For example, the maximum size of ActiveList observed on the OC192 link was 11 flows. To evaluate the performance of SFQ we therefore apply the trace data to a simulated scheduler operating at a rate considerably less than the rate of the link on which the trace was measured. The different simulated link capacities are given in Table 2.

**Table 2: Simulated link capacities in bps, all traces**

| Load | ADSL | Ab-I | Ab-III |
|------|------|------|--------|
| 0.6  | 72M  | 541M | 3.16G  |
| 0.9  | 48M  | 361M | 2.11G  |

For each trace in Table 1, link capacities were calculated to produce a load of 0.6 and 0.9, where the load is defined as flow arrival rate × mean flow size, divided by the link capacity. This produces realistic results under the assumption that the transport protocol controlling each flow is capable of using the current fair rate, whenever this is less than the exogenous peak rate. In the simulation, this condition is satisfied since sufficient buffering is provided to avoid any packet loss. Figure 4 shows the complementary distributions of the observed ActiveList size. The most significant observation is that the number of active flows in all three cases, even at 90% utilization, is very much smaller than the average number of flows in progress (see Table 1). Secondly, the number does not increase with link rate. Indeed, the worst case is that of the ADSL link. This is due to the fact that most flows are not bottlenecked on this link because of the limited peak rate. The distinction between the impact of bottlenecked and non-bottlenecked flows is clarified in the development of the analytical model in the next section.

These results are significant in that they suggest fair queueing is scalable and feasible. It is scalable since complexity does not increase with link speed; it is feasible since the required ActiveList size is relatively small.

## 4.   ANALYTICAL MODEL

In this section we develop an analytical model to explain why fair queueing is scalable and to understand the impact of the various traffic characteristics. An earlier model outlined in [16] successfully reproduced simulation results for another trace but relied for this on a traffic classification derived from the simulation. The present model relies only on intrinsic traffic characteristics.

### 4.1   Traffic model

We assume traffic is generated according to the Poisson session model introduced in Section 2.1. We further suppose flows have a *constant* peak rate drawn from a set $\{c_1, c_2, \ldots, c_M\}$ with $c_1 > c_2 > \cdots > c_M$. Flows with peak rate $c_i$ constitute class $i$. The number of class $i$ flows in progress at time $t$ is denoted $X_i(t)$.

The analysis is based on a quasi-stationary timescale separation. We first suppose the $X_i$ are fixed and evaluate the distribution of the number of active flows for each state $x = (x_1, \ldots, x_M)$. We then introduce assumptions allowing us to estimate the stationary distribution of $X$ in order to derive the required unconditional distribution.
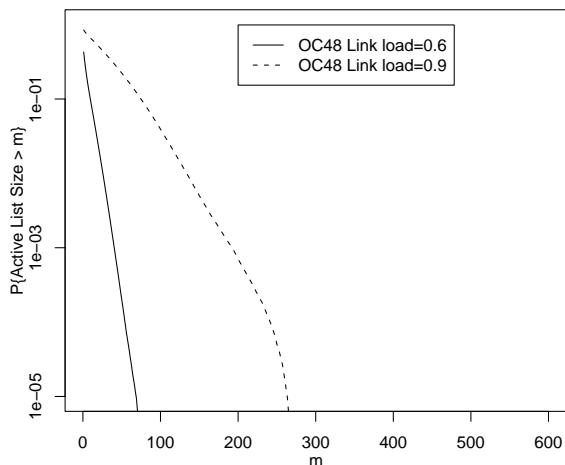
### 4.2   Bottlenecked and active flows

For notational convenience, define $c_0 = C$ and $c_{M+1} = 0$. If $\sum_{i=1}^{M} x_i c_i > C$, fair queueing realizes max-min fair sharing with fair rate $\theta$ given by
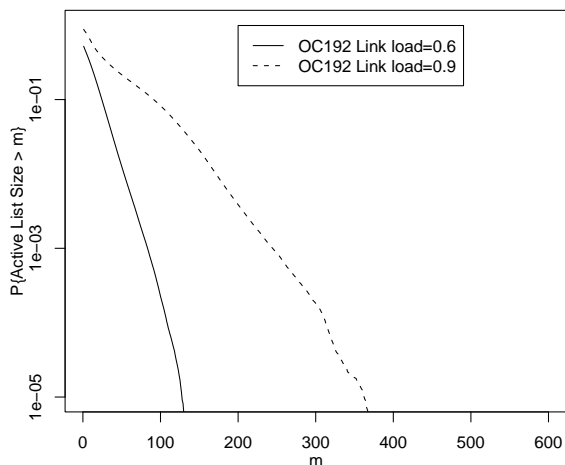
$$\theta = \frac{C - \sum_{i>J} x_i c_i}{\sum_{i \le J} x_i} \qquad (1)$$
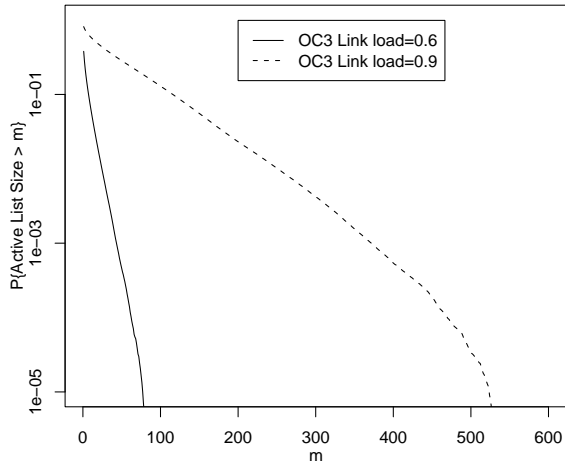
(a) ADSL



(b) Ab-I



(c) Ab-III

**Figure 4: Complementary distribution of ActiveList size from trace driven simulations at utilizations of 60% and 90%**

where $J = J(x)$ is the unique integer, $1 \leq J \leq M$, such that $c_J \geq \theta > c_{J+1}$.

Flows of peak rate greater than or equal to $c_J$ are bottlenecked and realize the fair rate $\theta$ while the others preserve their peak rate through the scheduler. If $\sum_{i=1}^{M} x_i c_i \leq C$, all flows preserve their peak rate. Let $J(x) = 0$ in this case. Note that $J$ defines a partition of the state space.

Consider now the operation of the SFQ algorithm. All bottlenecked flows are included in ActiveList. In addition, any non-bottlenecked flow having emitted a packet in the recent past, such that its finish tag is less than VirtualTime, is also included. To evaluate the stationary distribution of the total number we must make some assumptions about packet lengths and the arrival process of packets from non-bottlenecked flows.

The most convenient assumption is to suppose all packets of bottlenecked flows have maximum size MTU. This is reasonable given the packet size statistics of high peak rate flows reported above and is arguably a worst case assumption for the distribution of ActiveList size[2]. Given this assumption, and taking MTU as the unit of virtual time, it is easy to see that VirtualTime takes only integer values.

The periods when at least one flow is bottlenecked and VirtualTime is constant constitute "busy cycles". Any packet from a non-bottlenecked flow arriving in a busy cycle acquires the time stamp VirtualTime and further perpetuates the cycle for the duration of its own transmission. This flow remains in ActiveList until the end of the cycle. It is then removed since its finish tag cannot be greater than VirtualTime $+ 1$ (it has size $l \leq 1$ in MTU units). Figure 5 illustrates the notion of busy cycle.
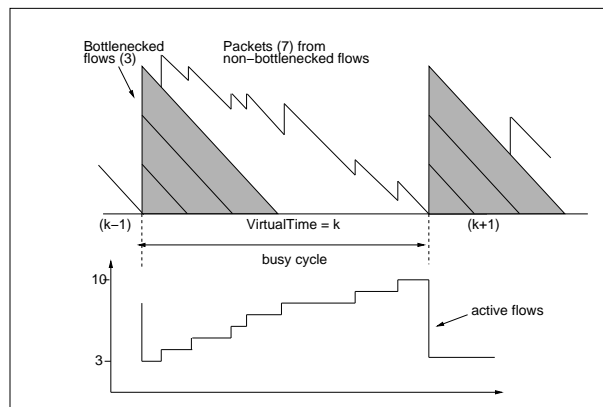


**Figure 5: Illustration of the notion of busy cycle.**

Busy cycles also occur when there are no bottlenecked flows in progress. In this case a busy cycle is initiated by a packet arriving to an empty system. VirtualTime remains equal to zero in such cycles.

The size of ActiveList is largest just before the end of a busy cycle. To evaluate the stationary distribution of this number *MaxList*, we assume each non-bottlenecked flow emits at most one packet in a cycle[3].

To model the arrival process we assume each flow inde-

---

[2]This is because VirtualTime changes more rarely so that small non-backlogged packets remain longer in the list.
[3]A flow cannot emit more than MTU bytes since this would contradict its non-bottlenecked status.

pendently emits its packet between time $t$ and $t + dt$ after the start of the busy cycle with probability $\alpha dt + o(dt)$. This finite source Poisson process facilitates analysis while accurately accounting for the number of contributing flows $\sum_{i>J} x_i$.

## 4.3 Conditional distribution of ActiveList size

From the above description it may be recognized that an estimate for the number of non-bottlenecked flows contributing to *MaxList* can be derived from an evaluation of the busy period of a queue with exceptional first service. The latter corresponds to the transmission of one MTU sized packet from each of the $\sum_{i \le J} x_i$ bottlenecked flows [4].

The arrival process results from $N = \sum_{i>J} x_i$ flows independently emitting at most one packet according to the process described at the end of the last section. The service time distribution derives from that of the packet length $F(s)$. We assume packet lengths are i.i.d. for all non-bottlenecked flows[5]. Let packet length be measured in units of MTU and denote the mean by $\sigma$.

The assumed per-flow arrival rate $\alpha$ is the average rate for all non-bottlenecked flows: $\alpha = R/(N\sigma)$ where $R = \sum_{i>J} x_i c_i$ is their overall bit rate. The probability $k$ packets arrive in an interval of length $u$ from the start of the busy cycle is then:

$$\Lambda_N(k, u) = \binom{N}{k}(1 - e^{-\alpha u})^k e^{-(N-k)\alpha u} \qquad (2)$$

The conditional distribution of *MaxList* is given by the following proposition.

PROPOSITION 1. *Let $g(m; b, n, r)$ be the conditional distribution of* MaxList *given $b$ bottlenecked flows and $n$ non-bottlenecked flows contributing overall rate $r$. Assuming packets arrive according to process with distribution $\Lambda$ and have independent length drawn from distribution $F$, we have:* $g(m; b, n, r) =$

$$\begin{cases} \frac{1}{m} \int \Lambda_n(m-1, u) dF^{[m]}(u), & b = 0, m \ge 1, \\ \Lambda_n(0, b), & b > 0, m = b, \\ b \int \Lambda_n(m-b-1, u)/u dF^{[m-b]}(u-b), & b > 0, m > b, \end{cases}$$
$$(3)$$

*where $F^{[m]}$ is the m-fold convolution of $F$.*

PROOF. We adapt classical results for the M/G/1 queue by substituting $\Lambda_n$ for the corresponding Poisson distribution. This is possible because the considered arrival process has the same indiscernible properties as the Poisson process allowing us to apply the combinatorial lemma of Takàcs [19, page 231] and its generalization by Niu and Cooper [20]. The first case, when no flows are bottlenecked, derives from the classical result for the number of customers served in an M/G/1 busy period [19, page 63]. The second case just expresses the probability that no non-bottlenecked flow emits a packet in the busy cycle. The third case is derived from results in [20] for the number of customers served in a busy period starting with an exceptional first service corresponding to $b$ packets from the bottlenecked flows. ☐

[4] The order of service of packets with the same time stamp has no impact on *MaxList*; it is convenient to suppose the bottlenecked packets are emitted first.

[5] We therefore ignore the fact that this distribution depends on the number of flows in each class and its specific distribution.

From the distribution of *MaxList* we can derive the distribution of the active list size at the moment when a flow should be added. Let this random variable be *AddToList*. Under the quasi-stationary assumption, the flow population $x$ is fixed and only non-bottlenecked flows are ever *added* to the list. The following proposition gives the conditional distribution of *AddToList*.

PROPOSITION 2. *Let $h(m; b, n, r)$ be the conditional distribution of* AddToList *given that there are $b$ bottlenecked flows and $n$ non-bottlenecked flows contributing overall rate $r$. We have, for $m \ge b \ge 0, n > 0$:*

$$h(m; b, n, r) = \frac{\sum_{k>m} g(k; b, n, r)}{\sum_{l>b}(l-b)g(l; b, n, r)} \qquad (4)$$

PROOF. The distribution $g(k; b, n, r)$ gives the relative frequency of busy cycles starting with ActiveList size $b$ and terminating with size $k$ due to $k - b$ packet arrivals. The probability that that an arbitrary arrival occurs in such a busy cycle is thus $(k-b)g(k; b, n, r)/\sum_{l>b}(l-b)g(l; b, n, r)$. The probability that arrival encounters $m$ flows in ActiveList is then $1/(k-b)$ for $b \le m < k$. The expression for $h$ is derived on multiplying the two probabilities and summing over all possible values of $k$. ☐

## 4.4 Flows in progress

Consider now the stationary distribution of $X(t)$ assuming a fluid model where flow rates adjust instantly as flows start and end. The fair queueing scheduler realizes max-min fair sharing with fair rate $\theta$ as defined in equation (1). Unfortunately, analysing max-min sharing is intractable under any realistic traffic assumptions. We therefore consider the alternative sharing objective of *balanced fairness*. This leads to a relatively simple expression for the distribution of $X$ [21]. The assumption, that we partially justify later, is that the distribution of $X$ is approximately the same under max-min and balanced fairness.

### 4.4.1 Balanced fair allocation

Under balanced fairness it is possible to evaluate exactly the stationary distribution of $X$. Moreover, this distribution is insensitive to all characteristics of the Poisson session traffic model defined in Section 2 other than the respective class demands $a_i$ (= class $i$ flow arrival rate × average class $i$ flow size). The load is equal to $a/C$ where $a = \sum_i^M a_i$.

Following Bonald and Virtamo [22] we adopt the shorthand $c^x = \prod_{i=1}^M c_i^{x_i}$, $a^x = \prod_{i=1}^M a_i^{x_i}$ and $x! = \prod_{i=1}^M x_i!$ and denote by $e_i$ the $M$ vector with 1 in position $i$ and 0 elsewhere. The stationary distribution $\pi_{BF}$ is then given by:

$$\pi_{BF}(x) = \Phi(x)a^x/G \qquad (5)$$

where $G$ is a normalizing constant ([22] provides a recursive formula to evaluate $G$) and the balance function $\Phi$ is determined recursively as follows:

$$\Phi(x) = \begin{cases} 1/(x!c^x) & \text{if } xc^T \le C, \\ \sum_{i=1}^M \Phi(x - e_i)/C & \text{if } xc^T > C. \end{cases} \qquad (6)$$

### 4.4.2 Recurrence relations

Let $B_j = \sum_{i \le j} X_i$, $N_j = \sum_{i>j} X_i$ and $R_j = \sum_{i>j} X_i c_i$ for $0 \le j \le M$. We need to evaluate, from $\pi_{BF}(x)$, the joint stationary distribution of $B_J$, $N_J$ and $R_J$ where $J$ is the index defined in Section 4.2. Note that as $J$ defines a

partition of states $x$, it also partitions allowable values of $(B_J, N_J, R_J)$ (we have, for example, $\theta = (C - R)/B$ and $c_J \leq \theta < c_{J+1}$).

Let $Q_J^{BNR}(b, n, r)$ denote the joint distribution of $B_J$, $N_J$ and $R_J$. To evaluate this distribution directly is impractical in view of the huge size of the corresponding array. We therefore evaluate the following approximation:

$$
\begin{aligned}
Q_J^{BNR}(b, n, r) &= \mathbb{P}\{B_J = b | N_J = n, R_J = r\} Q_J^{NR}(n, r) \\
&\approx Q_J^{BR}(b, r) Q_J^{NR}(n, r) / Q_J^{R}(r), \quad (7)
\end{aligned}
$$

with the obvious interpretation for the distributions on the right hand side. This is reasonable since $N_J$ and $R_J$ are closely correlated. To evaluate the two-term distributions we apply the recurrence relations stated in the following propositions (proofs are in the annex).

PROPOSITION 3. *The joint distribution of $B_J$ and $R_J$, for $0 \leq J \leq M$, is given by $Q_J^{BR}(b, r) = q_J(b, r)$, where the $q_j(b, r)$, for $0 \leq j \leq M$, satisfy the following recurrence relations:*

$$
\begin{aligned}
q_j(b, r) &= \sum_{i \leq j} \frac{a_i}{C} [q_j(b - 1, r) + \sum_{s=C-r-c_i+1}^{C-r} p_j(b-1, s, r)] \\
&+ \sum_{i > j} \frac{a_i}{C} [q_j(b, r - c_i) + \sum_{s=C-r+1}^{C-r+c_i} p_j(b, s, r - c_i)]
\end{aligned}
$$

(8)

*and*

$$
\begin{aligned}
p_j(b, s, r) &= \sum_{i \leq j} \frac{a_i}{(s + r) \wedge C} p_j(b - 1, s - c_i, r) \\
&+ \sum_{i > j} \frac{a_i}{(s + r) \wedge C} p_j(b, s, r - c_i).
\end{aligned}
$$

(9)

*Values of $q_j$ and $p_j$ are zero if any argument is negative and $q_j(0, 0) = p_j(0, 0, 0) = \pi_{BF}(0)$ is determined by the normalization condition.*

PROPOSITION 4. *The joint distributions of $N_J$ and $R_J$, for $1 \leq J \leq M$, is given by $Q_J^{NR}(n, r) = q_J(n, r)$, where the $q_j(n, r)$, for $1 \leq j \leq M$, satisfy the following recurrence relations:*

$$
(1 - \sum_{i \leq j} \frac{a_i}{C}) q_j(n, r) = \sum_{i \leq j} \frac{a_i}{C} \sum_{s=C-r-c_i+1}^{C-r} p_j(s, n, r)
$$

$$
+ \sum_{i > j} \frac{a_i}{C} [q_j(n - 1, r - c_i) + \sum_{s=C-r+1}^{C-r+c_i} p_j(s, n - 1, r - c_i)]
$$

(10)

*and*

$$
\begin{aligned}
p_j(s, n, r) &= \sum_{i \leq j} \frac{a_i}{(s + r) \wedge C} p_j(s - c_i, n, r) \\
&+ \sum_{i > j} \frac{a_i}{(s + r) \wedge C} p_j(s, n - 1, r - c_i) \quad (11)
\end{aligned}
$$

*Values of $q_j$ and $p_j$ are zero if any argument is negative and $p_j(0, 0, 0) = \pi_{BF}(0)$ is determined by the normalization condition.*
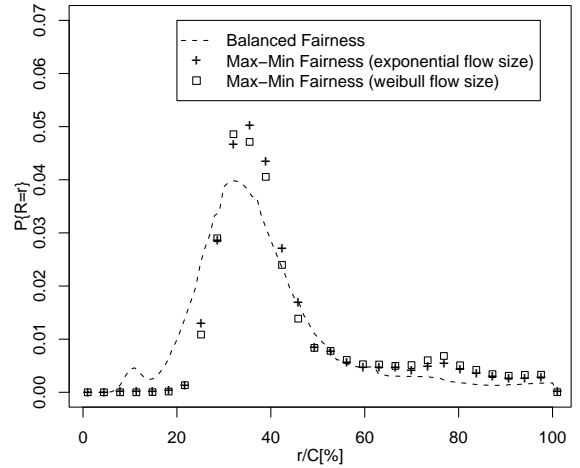
### 4.4.3 Comparison between max-min and balanced fair sharing

Previous work has shown that max-min and balanced fairness result in similar flow level performance [21]. This has been confirmed in the present case by evaluating the behaviour of max-min fair sharing by fluid simulation and comparing the marginal distributions of $B$ and $R$ (over all $J$) with that derived for balanced fairness.

We assume a Poisson arrival process of flows having a Weibull size distribution with shape parameter equal to 0.3 (yielding a long tailed distribution) in the first simulation scenario, and an exponential size distribution with the same mean size equal to 1Mbit in the second one. Arrival rates were chosen to produce the demand proportion associated with each class. Flow peak rates in Mbps are $c = \{1500, 1000, 800, 500, 100, 50, 10, 1\}$ with respective demands in proportion to $a = \{21, 22, 17, 7, 6, 17, 6, 4\}$. These data are representative of the Abilene III OC192 trace.
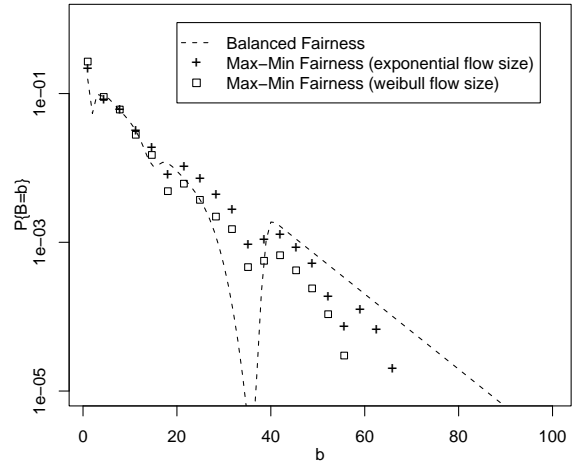


**Figure 6: Comparison between balanced and max-min allocations, load = 0.9, AbileneIII trace.**



**Figure 7: Comparison between balanced and max-min allocations, load = 0.9, AbileneIII trace.**

**Table 3: Flow classes and traffic proportions**

| Class | ADSL c [bps] | ADSL $a_i/a$ [%] | Ab-I c [bps] | Ab-I $a_i/a$ [%] | Ab-III c [bps] | Ab-III $a_i/a$ [%] |
|---|---|---|---|---|---|---|
| $\mathcal{C}_1$ | 1M | 52 | 100M | 24 | 1.5G | 21 |
| $\mathcal{C}_2$ | 512K | 26 | 20M | 10 | 1G | 22 |
| $\mathcal{C}_3$ | 128K | 12 | 15M | 9 | 800M | 17 |
| $\mathcal{C}_4$ | 50K | 10 | 12M | 14 | 500M | 7 |
| $\mathcal{C}_5$ | - | - | 10M | 14 | 100M | 6 |
| $\mathcal{C}_6$ | - | - | 5M | 11 | 50M | 17 |
| $\mathcal{C}_7$ | - | - | 1M | 4 | 10M | 6 |
| $\mathcal{C}_8$ | - | - | 500K | 12 | 1M | 4 |

Figures 6 and 7 compare the marginal distributions of $B$ and $R$, respectively, for link load 0.9. The rough agreement is typical of results for data corresponding to the other traces. Note the somewhat erratic fluctuations in these distributions due to the discrete mixture of peak rates. In this fluid model, all flows belonging to the same class become bottlenecked or non bottlenecked simultaneously which explains the observed jumps in the distribution of $B$. Errors introduced by the approximation do not appear to be too significant. At load 0.6, there is little controlled sharing (i.e., $J = 0$ with high probability) and max-min and balanced fairness are practically the same.

## 4.5 Unconditional ActiveList size distribution

We can now estimate the distribution of ActiveList size by combining the results of Sections 4.3 and 4.4, i.e.

$$\mathbb{P}\{\text{ActiveList size} = m\} =$$
$$\begin{cases} \sum_{(b,n,r):J<M} h(m;b,n,r)Q_J^{BNR}(b,n,r) & J < M \\ Q_M^{BNR}(m,0,0) & J = M \end{cases}$$

We perform the calculations with data representative of the three traces described in Section 2.2 The peak rate classes (in bps) and respective traffic proportions (in %) are given in Table 3.

These values are derived by choosing a set of peak rates roughly covering the domain of the distributions in Figure 2 and attributing to each class the traffic from flows whose peak rate is closest. We set $C$ and adjust the absolute values of the demand vectors $a$ to attain link loads of 0.6 and 0.9, as in the simulations described in Section 3.2 and summarized in Table 2.

For the packet size of non-bottlenecked flows we have used a distribution with just two values as it is difficult to numerically evaluate the convolution in Proposition 1 for more terms. We chose packet sizes of 40 and 1500 bytes with relative proportions derived roughly from the trace statistics (50% for each packet size for the ADSL and Abilene traces, 30% and 70% respectively for the Abilene III trace). The value of MTU was 1500 bytes for the first two traces and 9000 bytes for the third.
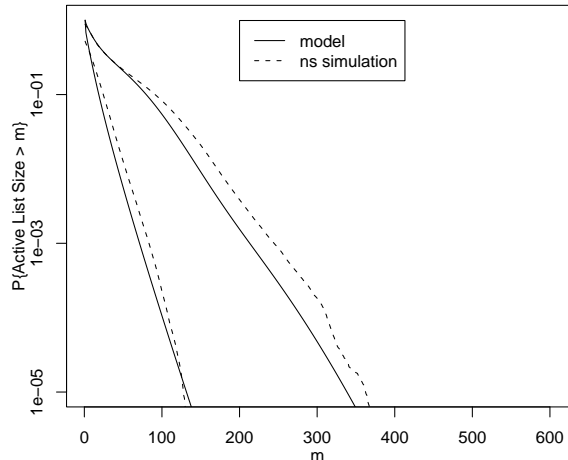
Results are presented in Figure 8. As can be seen, the analytical model predicts the simulated ActiveList size distribution quite accurately. These results are discussed in the next section.
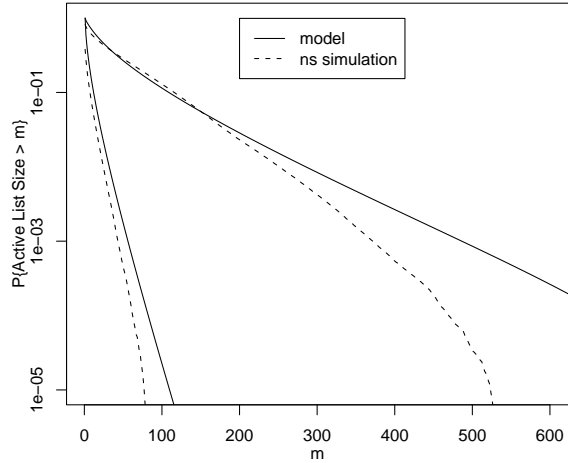


(a) *ADSL*
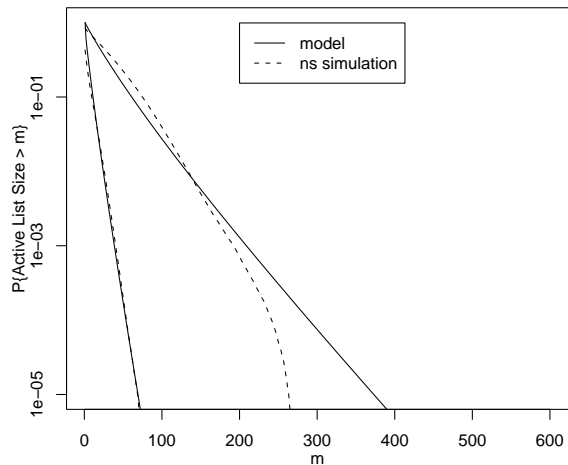


(b) *Ab-I*



(c) Ab-III

**Figure 8: Complementary distribution of ActiveList size from analytical model at utilizations of 60% and 90%**

# 5. DISCUSSION

In this section we discuss the relative importance of traffic characteristics at flow and packet timescales and highlight a number of important issues.

## 5.1 Impact of traffic characteristics

First recall the lack of impact on the population of flows in progress $X$ of the detailed session structure under the assumed Poisson session traffic model. This insensitivity is strictly valid only for balanced fairness but we believe it to be true also for all practical purposes for max-min fairness. The only significant characteristics are the overall load and the distribution of this load over the range of exogenous flow peak rates.

The assumption in the model that the peak rate is one of a finite set of values and is the same throughout the flow lifetime is clearly a simplification. It nevertheless appears to predict performance reasonably accurately. The main impact of the rate distribution is to determine for any load which proportion of traffic is in bottlenecked flows and which is not. These two types of traffic are shown to have qualitatively different impacts on ActiveList size.

Bottlenecked flows are few in number up to high loads of around 90%. The majority of active flows correspond to isolated packets emitted by relatively low rate non-bottlenecked flows. This explains why the ADSL trace gives rise to the largest list: its peak rate limited flows are rarely bottlenecked even at 90% load. The very high rate flows in Ab-III, on the other hand, have a beneficial impact on the statistics of ActiveList size.

The packet length of non-bottlenecked flows has a significant impact. The smaller the packets, the greater the packet arrival rate and, in consequence, the greater the number of flows contributing to a busy cycle and momentarily entering ActiveList.

## 5.2 Benefits of fair queueing

Imposed fairness relieves the network from relying on user cooperation to ensure satisfactory performance. It does remain possible, however, for users to establish multiple parallel flows and thus acquire a greater rate than the fair share. The present models show that such advantage would only accrue to the small proportion of users whose flows are bottlenecked. Such users already realize high throughput. If they nevertheless "cheat" by setting up parallel flows, this should be fairly easy to detect and counter given their small number.

The Poisson session model (with alternating flows and think times) remains appropriate for users whose combined peak rate is less than the fair rate, whatever the number of 5-tuple flows they actually use in parallel. This is because packets of the combination still arrive in isolation (i.e., $\leq 1$ in any busy cycle) and add 1 to the current ActiveList size independently of the particular 5-tuple flow to which they belong.

When fairness is assured by the network, new transport protocols can be tested and introduced without having to ensure they are not overly aggressive to legacy TCP variants. This is an appreciable advantage, notably for the design of protocols adapted to high speed transmission (see Section 5.5 below). It is also possible to introduce known techniques for improving the behaviour of TCP slow-start.

A particularly interesting benefit of fair queueing is the protection afforded to relatively low rate real time flows. As long as the rate of audio and video flows is less than the current fair rate (and our models demonstrate that this is typically quite high), their packets achieve low latency in traversing the FQ scheduler. This effectively realizes service differentiation without the cumbersome requirement to explicitly mark packets as real time or delay tolerant. Packet latency can be further reduced for flows whose peak rate is less than the current fair rate by giving priority to their packets (i.e., packets that acquire time stamp VirtualTime in line 7 of the SFQ code go to the head of the schedule), as proposed in [15].

## 5.3 Alternative realizations

We have used the SFQ algorithm to exemplify the performance of fair queueing but it is clearly possible to use different algorithms. Similar results would apply for other self-clocking schedulers and to round robin variants like DRR [8].

Some authors have proposed to use active queue management schemes to avoid the pretended scalability issues of scheduling [9, 10, 11]. The present discussion would also apply (with somewhat less precision) to determining the performance of these algorithms in a realistic dynamic traffic setting. These algorithm aim to operate only on the bottlenecked flows but would presumably need to account momentarily for active flows in order to be able to detect new bottlenecked ones.

Core stateless fair queueing [23] distributes the complexity of scheduling to the network edge where the current rate of each flow is estimated. Note that all in progress flows must be tracked and measured with this approach.

Of course, the congestion control of TCP currently realizes approximate fair sharing with simple FIFO queues (albeit with well known biases and inefficiencies). The present models are again useful in understanding how the number of flows actually contending for resources on a given link (the bottlenecked flows) is typically very small.

## 5.4 Avoiding overload

In this paper we have shown that fair queueing is scalable and feasible at normal to heavy loads. When demand exceeds around 90% of link capacity, however, the number of in progress and active flows increases rapidly. In such overload the scheduler (like the above mentioned AQM mechanisms) tends to revert to a FIFO queue. Per-flow throughput gets smaller and smaller as an increasing number of flows compete for the available bandwidth.

It is clearly desirable to avoid such overload, whatever queue management scheme is employed. A possible solution avoiding the current requirement for excessive over-provisioning would be to use per-flow admission control as advocated in [15]. The measure of the current fair rate realized by the scheduler is a convenient criterion for deciding when new flows should be blocked.

## 5.5 Buffer sizing

It has recently been observed that the rule of thumb whereby router buffers should be able to store some 200 ms of data at line rate is excessive and unsustainable as link bandwidth increases [24]. The present discussion on the numbers of in progress, active and bottlenecked flows throws more light on typical buffer requirements.

It is shown in [24] that buffer requirements decrease substantially as the number of flows in progress increases, under the assumption that these flows are all bottlenecked. This trend is confirmed using different models in [25], again assuming traffic is composed mainly of "locally bottlenecked persistent flows". In fact, our results demonstrate that the number of locally bottlenecked flows is typically relatively small, the vast majority of flows having a peak rate considerably less than the current fair rate. If more than a hundred flows *are* actually bottlenecked, this is usually a sign of severe overload that buffering alone cannot control, as discussed in the previous section.

If only a handful of flows are bottlenecked (there is a non-negligible probability that the number is only one or two), they necessarily have a high rate and the models in [24] show that a large buffer is indeed necessary to sustain 100% utilization using TCP. We might therefore argue that the above rule of thumb should be applied, at least to the sizeable line rate fraction used by the bottlenecked flows. However, this is not what we would advocate.

Most of the time, on most links, no flow is bottlenecked and a small buffer (for around 100 packets) is adequate to ensure low loss. Rather than applying the rule of thumb to cater for the small number of exceptional flows that can saturate the residual link capacity, it seems more appropriate to require that these flows employ one of the recently proposed high speed TCP variants (e.g., [26, 27, 28]). A feature of these protocols is that they require much less buffering to attain 100% link utilisation.

## 6. CONCLUSION

We have shown that the advantages of network assured max-min fair sharing can be realized because per-flow fair queueing is *scalable* - since the number of flows that need to be scheduled is independent of link speed - and *feasible* - since this number is relatively small. Performance has been evaluated using trace driven simulations and explained by means of an analytical model.

The model demonstrates that, after the average link load, the most significant traffic characteristics are the exogenous peak rates the flows can attain in the absence of link congestion. These determine the proportion of flows that are bottlenecked and those that are not. The model shows how these flows combine to determine the complexity of the scheduler and explains why this remains low up to utilizations of around 90%.

The number of bottlenecked flows is typically measured in tens. Many more flows are in progress but they are non-bottlenecked and only contribute episodically to the list of active flows to be accounted for by the scheduled. The overall number of flows that need to be scheduled is limited to a few hundreds, independently of the proportions of traffic in bottlenecked and non-bottlenecked flows.

In addition to protecting the quality of service of individual flows against possible user misbehaviour, the use of fair queueing would bring two significant advantages. First, it would be possible to test and introduce new, more efficient transport protocols without requiring that they be TCP-friendly. This is useful notably for very high speed transfers for which the congestion avoidance algorithm of TCP is known to waste bandwidth. Second, fair queueing

ensures low packet latency for flows whose rate is less than the current fair rate, the non-bottlenecked flows. This allows audio and video flows to coexist with high speed data transfers without the need to explicitly identify the former.

The present work suggests several directions for further research. The more approximate fairness realized by AQM mechanisms may be sufficient for elastic data traffic but it remains to evaluate the performance they provide for below fair rate audio and video flows. How could a transport protocol best exploit the assurance of max-min fairness in order to improve the current slow-start and congestion avoidance algorithms? Fair queueing is feasible as envisaged only up to a load of around 90%: it is necessary therefore to implement some means to prevent overload. This may take the form of dynamic load balancing or adaptive routing. A necessary component appears to be some form of flow-level admission control, as envisaged in [15].

This paper presents credible technical arguments, based on analysis, simulations and recent Internet measurements, in favour of the feasibility and desirability of fair queueing. We hope these arguments will encourage router architects to reconsider the opportunity of implementing fair queueing in the design of next generation routers allowing the development of a more efficient and reliable Internet.

## Annex

### Proof of Proposition 3

Define variables $S_j = \sum_{i \leq j} x_i c_i$ and sets $\mathcal{S}_j(b, s, r) = \{x : B_j = b, S_j = s, R_j = r\}$. Consider the probabilities $p_j(b, s, r) = \mathbb{P}\{B_j = b, S_j = s, R_j = r\}$ for $j > 0$:

$$
\begin{aligned}
p_j(b, s, r) &= \sum_{x \in \mathcal{S}_j(b,s,r)} \pi(x) \\
&= \sum_{x \in S_j(b,s,r)} \sum_{i=1}^{M} \frac{a_i}{(s+r) \wedge C} \pi(x - e_i) \\
&= \sum_{i \leq j} \frac{a_i}{(s+r) \wedge C} \sum_{x \in S_j(b-1, s-c_i, r)} \pi(x) + \\
&+ \sum_{i > j} \frac{a_i}{(s+r) \wedge C} \sum_{x \in S_j(b, s, r-c_i)} \pi(x),
\end{aligned}
$$

and this is (9). The second step uses properties (5) and (6), (see [22]). If $b = 0$ then $q(0, r)$, for $r = 0, \ldots, C$, is directly given by the Kaufman-Roberts recursion (see below and [22]). Observe that when $b > 0$, $S_J + R_J > C$. Thus we can define $q_j(b, r) = \sum_{s > C-r} p_j(b, s, r)$ for $j > 0$ so that

$$
\begin{aligned}
q_j(b, r) &= \sum_{s > C-r} p_j(b, s, r) = \\
&= \sum_{i \leq j} \frac{a_i}{C} \sum_{s > C-r} p_j(b-1, s-c_i, r) \\
&+ \sum_{i > j} \frac{a_i}{C} \sum_{s > C-r} p_j(b, s, r-c_i), \\
&= \sum_{i \leq j} \frac{a_i}{C} \sum_{s > C-r-c_i} p_j(b-1, s, r) \\
&+ \sum_{i > j} \frac{a_i}{C} \sum_{s > C-r} p_j(b, s, r-c_i),
\end{aligned}
$$

The last two sums can be split as in (8). $Q_J^{BR}(b,r)$ is given limiting $q_j(b,r)$ in the state space partition induced by $J$.

## Proof of Proposition 4

Define variables $S_j = \sum_{i \leq j} x_i c_i$ and sets $\mathcal{S}_j(s,n,r) = \{x : S_j = s, N_j = n, R_j = r\}$. Consider the probabilities $p_j(s,n,r) = \mathbb{P}\{S_j = s, N_j = n, R_j = r\}$ for $j > 0$:

$$
\begin{aligned}
p_j(s,n,r) &= \sum_{x \in \mathcal{S}_j(s,n,r)} \pi(x) \\
&= \sum_{x \in S_j(s,n,r)} \sum_{i=1}^{M} \frac{a_i}{(s+r) \wedge C} \pi(x - e_i) \\
&= \sum_{i \leq j} \frac{a_i}{(s+r) \wedge C} \sum_{x \in S_j(s-c_i,n,r)} \pi(x) + \\
&\quad + \sum_{i > j} \frac{a_i}{(s+r) \wedge C} \sum_{x \in S_j(s,n-1,r-c_i)} \pi(x),
\end{aligned}
$$

and this is (11). The second step uses properties (5) and (6) (see [22]) as in the previous proof. Now let $q_j(n,r) = \sum_{s>C-r} p_j(s,n,r)$ so that

$$
\begin{aligned}
q_j(n,r) &= \sum_{s>C-r} p_j(s,n,r) = \\
&= \sum_{i \leq j} \frac{a_i}{C} \sum_{s>C-r} p_j(s-c_i,n,r) \\
&\quad + \sum_{i>j} \frac{a_i}{C} \sum_{s>C-r} p_j(s,n-1,r-c_i), \\
&= \sum_{i \leq j} \frac{a_i}{C} \sum_{s>C-r-c_i} p_j(s,n,r) \\
&\quad + \sum_{i>j} \frac{a_i}{C} \sum_{s>C-r} p_j(s,n-1,r-c_i),
\end{aligned}
$$

The last two sums can be split as in (10) where the first term does not depend on $i$. If $J = 0$ we obtain again the Kaufman-Roberts recursion. $Q_J^{NR}(n,r)$ is given considering $q_j(n,r)$ in the state space partition induced by $J$.

## Kaufman-Roberts Recursion

$\mathbb{P}\{X_1 c_1 + \cdots + X_M c_M = n\} = q(n)$ for $n \leq C$ satisfies the following recursion:

$$
q(n) = \sum_{i=1}^{M} \frac{a_i}{n} q(n - c_i)
$$

with $q(n) = 0$ if $n < 0$.

PROOF. The joint distribution defined in (5),(6) in the case $xc^T \leq C$ satisfies the relation $x_i c_i \pi(x) = a_i \pi(x - e_i)$. Thus, summing over $i$, $n\pi(x) = \sum_{i=1}^{M} a_i \pi(x - e_i)$. Finally, summing over all $x$ such that $xc^T = n$ we conclude the proof. $\square$

# 7. REFERENCES

[1] D. Bertsekas and R. Gallager, *Data Networks*, Prentice Hall, Englwood Cliffs, N.J., 1992.

[2] E. L. Hahne, Round-Robin Scheduling for Max-Min Fairness in Data Networks, IEEE Journal on Selected Areas in Communications, Vol. 9, No. 7, Sep. 1991, Pages: 1024-1039.

[3] B. Suter, T.V. Lakshman, D. Stiliadis, A.K. Choudhury; Buffer Management Schemes for Supporting TCP in Gigabit Routers with Per-Flow Queueing, IEEE Journal on Selected Areas in Communications, Aug. 1999.

[4] F. P. Kelly, A.K. Maulloo and D.K.H. Tan, Rate Control in Communication Networks: Shadow Prices, Proportional Fairness and Stability, Journal of the Operational Research Society 49 (1998), Pages: 237-252.

[5] T. Bonald, L. Massoulié, Impact of Fairness on Internet Performance, In Proc. of ACM SIGMETRICS / IFIP Performance, 2001.

[6] J. Nagle, On Packet Switches with Infinite Storage, RFC 970, IETF, 1985.

[7] 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, Pages: 690-704.

[8] M. Shreedhar, G. Varghese, Efficient Fair Queueing Using Deficit Round Robin, IEEE/ACM Transactions on Networking, Vol. 4, No. 3, June 1996.

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

[10] R. Mahajan, S. Floyd, D. Weatherall, Controling High-Bandwidth Flows at the Congested Router, In Proc. of IEEE ICNP 2001.

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

[12] N. Brownlee, KC Claffy, Understanding Internet Traffic Streams: Dragonflies and Tortoises, IEEE Communications Magazine, Vol. 40 , No. 10, Oct. 2002, Pages: 110-117.

[13] S. Sarvotham, R. Riedi, R. Baraniuk, Connection-Level Analysis and Modeling of Network Traffic, In Proc. of ACM Internet Measurement Workshop 2001.

[14] Y. Zhang, L. Breslau, V. Paxson, S. Shenker, On the Characteristics and Origins of Internet Flow Rates, In Proc. of ACM SIGCOMM 2002.

[15] A. Kortebi, S. Oueslati, J. Roberts, Cross-protect: Implicit Service Differentiation and Admission Control, In Proc. of IEEE HPSR 2004.

[16] A. Kortebi, L. Muscariello, S. Oueslati, J. Roberts, On the Scalability of Fair Queueing, In Proc. of ACM HotNets III 2004.

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

[18] C. Fraleigh, S. Moon, B. Lyles, C. Cotton, M. Khan, D. Moll, R. Rockell, T. Seely, C. Diot, Packet-Level Traffic Measurements from the Sprint IP Backbone, IEEE Network, Vol. 17 , No. 6 , Nov.-Dec. 2003, Pages: 6-16.

[19] L. Takàcs, *Introduction to the Theory of Queues*, Oxford University Press, 1962, New York.

[20] 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, 1989, Pages: 281-293.

[21] T. Bonald, A. Proutière, Insensitivity in Processor Sharing Networks, In Proc. of IFIP Performance 2002.

[22] T. Bonald, J. Virtamo, A recursive formula for multirate systems with elastic traffic, To appear in IEEE Communications Letters.

[23] I. Stoica, S. Shenker, H. Zhang, Core-Stateless Fair Queueing: A scalable Architecture to Approximate Fair Bandwidth Allocations in High-Speed Networks.IEEE/ACM Transactions on Networking, Vol. 11, No. 1, Feb. 2003, Pages: 33 - 46.

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

[25] A. Dhamdhere, H. Jiang, C. Dovrolis, Buffer sizing for congested Internet links, In Proc. of IEEE Infocom 2005.

[26] S. Floyd, HighSpeed TCP for Large Congestion Windows, IETF RFC 3649, December 2003.

[27] T. Kelly, Scalable TCP: Improving Performance in Highspeed Wide Area Networks. Computer Communication Review 32(2), April 2003.

[28] C. Jin, D. X. Wei, S. H. Low, FAST TCP: Motivation, Architecture, Algorithms, Performance, In Proc. of IEEE Infocom 2004.