DELAY DIFFERENTIATION WITHOUT RATE TINKERING

MARTIN KARSTEN, CHERITON SCHOOL OF COMPUTER SCIENCE, UNIVERSITY OF WATERLOO,
EMAIL: MKARSTEN@UWATERLOO.CA
JENS SCHMITT, DEPARTMENT OF COMPUTER SCIENCE, UNIVERSITY OF KAISERSLAUTERN,
EMAIL: JSCHMITT@INFORMATIK.UNI-KL.DE
MICHAEL BECK, DEPARTMENT OF COMPUTER SCIENCE, UNIVERSITY OF KAISERSLAUTERN,
EMAIL: BECK@INFORMATIK.UNI-KL.DE

University of Waterloo Technical Report CS-2012-13

ABSTRACT. Link buffering is a key element in packet-switched networks to absorb temporary traffic bursts without excessive dropping. The resulting queuing delay is a critical performance factor. Research on buffer management typically studies differentiated buffer allocation to control forwarding rates or buffer size management to control the overall queuing delay. On the other hand, delay differentiation is typically accomplished by packet scheduling and thus implies rate differentiation. In this paper a radically simpler approach to delay differentiation through buffer management is studied. Instead of actively managing throughput, *Delay Differentiated FIFO* (DDF) uses a simple drop-based mechanism to offer multiple delay classes, but closely tracks the per-class throughput of the corresponding single-class FIFO queuing system. End systems choose which delay class best balances their loss and delay requirements. Architecturally, DDF does not interfere with management and policy decisions at end and edge systems and does not add another control loop to the existing mix of traffic management techniques. The forwarding characteristics of DDF are analyzed using stochastic models. Refinements to the basic algorithm are presented and it is shown how DDF can be implemented with very little execution overhead compared to FIFO. Packet-level simulation results are presented to complement the analytical findings.

1. INTRODUCTION

The management of link buffers is a key challenge in packet-switched networks. If the effective buffer size is too large, traffic suffers from unnecessary queuing delay. This problem has very recently garnered a lot of renewed attention due the so-called *bufferbloat* phenomenon [1]. However, insufficient buffers can also cause problems, because traffic bursts might then result in high packet loss. Furthermore, feedback-controlled sources, such as TCP senders, face an inherent feedback delay in proportion to the propagation delay and a corresponding lag of responsiveness. It is expected that intermediate buffers at the bottleneck keep utilizing the link during the lag period of TCP. While there is considerable research on using significantly smaller buffers when a large number of TCP flows is multiplexed [2], the exact nature of favorable conditions and the full impact of such a change are not yet understood. Ideally, a network router should support low-latency applications as well as traffic flows that benefit from a larger amount of buffering. In the past, research proposals have tried to address this requirement by introducing service differentiation, however, with the side effect that service differentiation treats some traffic as more important than other and thus requires some form of admission policy.

In this paper we investigate a low-complexity, but most importantly future-proof, approach to link buffering. Single-class FIFO queuing is extremely simple and treats all incoming traffic identically. Almost any existing alternative to FIFO queuing implements

some form of hard-coded policy that is imposed on other networking components, such as explicit mechanisms (signaling) or implicit requirements (edge admission control), or adds a local control loop to an already complicated stack of traffic control functionality (active queue management). In contrast, *Delay Differentiated FIFO* (DDF) adds delay differentiation to FIFO queuing, but with little side effects. It is flexible and future-proof, because it does not inherently tie delay differentiation to any other policy. The basic idea for DDF has been proposed in previous work [3] and studied with simple simulation experiments. In this paper, we present analytical models along with further in-depth simulations to investigate two critical conjectures for DDF:

- A single-class FIFO queue with finite buffers forwards multi-class traffic roughly in proportion to class arrival rates, i.e., FIFO is essentially *rate-neutral*, and
- DDF closely tracks the throughput behavior of the corresponding single-class FIFO queue.

The remainder of the paper is organized as follows. Section 2 contains a discussion of related work. In Section 3, novel analytical and simulation results are presented that illustrate the forwarding behavior of a regular single-class FIFO queue with finite buffers. The basic design of DDF is presented in Section 4 and analyzed in Section 5. Based on those results, a refinement of the DDF algorithm is given in Section 6, which is used for investigating additional specific scenarios using packet-level simulations. The paper is wrapped up with a brief summary and conclusions in Section 7.

2. Related Work

The original proposal [3] contains a fairly detailed discussion of high-level related work, such as differentiated services [4], edge-based load control [5], and small router buffers [2]. To avoid duplication, we limit the discussion here to very recent or very closely related literature and such work that relates to our FIFO analysis. To the best of our knowledge, there is no directly comparable system to the delay differentiation approach that we investigate. Both the ABE [6] as well as the RD Service [7] proposal for delay differentiation through buffer management are computationally complex when applied more than two service classes. Furthermore, they entrench a specific forwarding policy in core routers, which would contribute to the ossification of the Internet architecture. For example, Mathis [8] has questioned the traditional and narrow focus on "TCP friendly" congestion control.

The so-called *bufferbloat* phenomenon [1] has received a lot of attention recently, which has resulted in a novel active queue management proposal termed CoDel (Controlled Delay) [9]. The CoDel proposal represents another attempt at finding a one-size-fits-all solution for buffer management. While CoDel is very promising, it is unclear whether such a uniform approach to buffer management adequately considers the diverse service requirements from a multitude of different applications. However, our multi-class queuing scheme is orthogonal to and can be combined with any non-preemptive active queue management scheme.

Somewhat surprisingly, while FIFO's rate-neutrality appears intuitive (and has been validated in measurements for CBR traffic [10]), there is only little analytical knowledge about it. Ghiassi and Liebeherr [11] show FIFO's rate-neutrality for CBR traffic, and Ciucu et al. [12] extend this to a wide range of traffic models using the framework of the stochastic network calculus. However, both of these results critically rely on the assumption of an infinite buffer. Another recent paper [13] derives a probabilistic bound on the instantaneous loss rate under FIFO with a finite buffer, but this cannot be directly applied to determine

the long-term loss rate. In Section 3, we present a novel result on FIFO's rate-neutrality for a finite buffer and stochastic arrivals with i.i.d. inter-arrival times.

3. FIFO'S RATE NEUTRALITY

We investigate the intuitive, but to our knowledge unproven conjecture that a *finite* FIFO buffer is rate-neutral, i.e., it outputs flows proportionally to their arrival rates. To that end, we first establish an analytical result about FIFO's rate neutrality under fairly general assumptions on traffic characteristics, yet neglecting traffic correlations, which is why we next investigate through simulation how that result may extend to traffic sources with complex correlation structures.

3.1. **Analytical Result.**

Without loss of generality, we assume there are two traffic flows accessing the FIFO buffer. Let $A_1(t), A_2(t)$ be the cumulative arrivals up to time t from flow 1 and 2, respectively; further, we denote with $A_i^+(t)$ and $A_i^-(t)$ the cumulative arrivals of flow i up to time t that were and were not able to enter the buffer, with $A_i(t) = A_i^+(t) + A_i^-(t)$. We define the following (long-term) rates

$$\lambda_i^\bullet = \lim_{t\to\infty} \frac{A_i^\bullet(t)}{t} \qquad \bullet \in \{+,-,\sqcup\}, i \in \{1,2\}\,.$$

The following generic proposition provides a criterion for the rate-neutrality of a finite FIFO buffer:

Proposition 1. *Let λ_1 and λ_2 be given. If $K \in [0,1)$ exists for which $\mathbb{P}\left(\lambda_i^- = K\lambda_i\right) = 1$ for $i = 1,2$, then it applies that*

$$\mathbb{P}\left(\frac{\lambda_1^+}{\lambda_2^+} = \frac{\lambda_1}{\lambda_2}\right) = 1\,.$$

In other words, the FIFO buffer is rate-neutral almost surely.

Proof. At first, let us assume that $\exists K \in [0,1) :\ \lambda_i^- = K\lambda_i$, $i \in \{1,2\}$, then it follows that

$$\frac{\lambda_1^+}{\lambda_2^+} = \frac{\lim_{t\to\infty} \frac{A_1^+(t)}{t}}{\lim_{t\to\infty} \frac{A_2^+(t)}{t}} = \frac{\lim_{t\to\infty} \frac{A_1(t)-A_1^-(t)}{t}}{\lim_{t\to\infty} \frac{A_1(t)-A_1^-(t)}{t}} = \frac{\lambda_1 - K\lambda_1}{\lambda_2 - K\lambda_2} = \frac{\lambda_1}{\lambda_2}\,.$$

Hence it applies that

$$\begin{aligned}
\mathbb{P}\left(\frac{\lambda_1^+}{\lambda_2^+} = \frac{\lambda_1}{\lambda_2}\right) &\geq \mathbb{P}\left(\left\{\lambda_1^- = K\lambda_1\right\} \cap \left\{\mathbb{P}\left(\lambda_2^- = K\lambda_2\right)\right\}\right) \\
&\geq 1 - \mathbb{P}\left(\lambda_1^- \neq K\lambda_1\right) - \mathbb{P}\left(\lambda_2^- \neq K\lambda_2\right) \\
&= 1\,,
\end{aligned}$$

according to Boole's inequality and the given condition. □

The proposition essentially just provides a sufficient condition for traffic flows to be treated rate-neutrally at a finite FIFO buffer: that the loss of packets is proportionally equal to their respective input rates. That means the mathematical argument can be made via their loss processes, which essentially decouples it from the FIFO scheduling order. Next, after some preparatory lemmata we provide a result for sources with i.i.d. inter-arrival times that satisfy the condition of Proposition 1.

The following rather general lemma will be of some help:

Lemma 2. *Let $x,y \in \mathbb{R}$ $\{A = x\}$ be an almost sure event, i.e. $\mathbb{P}(A = x) = 1$, as well as $\mathbb{P}(B = A + y) = 1$. Then it applies that*

$$\mathbb{P}(B = x + y) = 1\,.$$

If $\mathbb{P}(C = A \cdot y) = 1$, then it also applies that

$$\mathbb{P}(C = x \cdot y) = 1\,.$$

Proof. It holds

$$\begin{aligned}\mathbb{P}(B = A + y) &= \mathbb{P}(B = A + y \,|\, A = x)\mathbb{P}(A = x) + \mathbb{P}(B = A + y \,|\, A \neq x)\mathbb{P}(A \neq x)\\ &= \mathbb{P}(B = x + y \,|\, A = x) = \frac{\mathbb{P}(\{B = x + y\} \cap \{A = x\})}{\mathbb{P}(A = x)}\\ &= \mathbb{P}(B = x + y)\end{aligned}$$

The second statement follows as an immediate variation. □

Before we can give the desired result, we first need a second lemma which provides a necessary statement about the distribution of loss periods. Here loss periods are defined as the periods in time when the buffer is completely full and arriving packets are lost. For the rest of this section we assume, that the server needs exactly s time units to process a packet and thus loss periods are always of length s.

Lemma 3. *Let A_1 and A_2 be two stochastically independent flows with i.i.d. interarrival times. Further assume the second moments of the interarrival times exist. If we denote by $m(t)$ the number of loss periods up to time t we have for some $k \in \mathbb{R}^+$*

$$\lim_{t\to\infty} \frac{m(t)}{t} = k \qquad a.s.$$

Proof. Since a loss period is of length s, we know already that

$$\frac{1}{s} \geq \frac{m(t)}{t} \geq 0\,.$$

We assume now that $\frac{m(t)}{t}$ does not converge and lead this to a contradiction. Under this assumption it must hold that

$$0 \leq \liminf_{t\to\infty} \frac{m(t)}{t} < \limsup_{t\to\infty} \frac{m(t)}{t} \leq \frac{1}{s}\,.$$

We abbreviate these two quantities by

$$\liminf_{t\to\infty} \frac{m(t)}{t} = \underline{k},$$

$$\limsup_{t\to\infty} \frac{m(t)}{t} = \bar{k}\,.$$

Now we define sequences $(t_{\underline{l}})_{\underline{l}\in\mathbb{N}} \xrightarrow{\underline{l}\to\infty} \infty$, such that $\frac{m(t_{\underline{l}})}{t_{\underline{l}}} \xrightarrow{\underline{l}\to\infty} \underline{k}$, and $(t_{\bar{l}})_{\bar{l}\in\mathbb{N}} \xrightarrow{\bar{l}\to\infty} \infty$, such that $\frac{m(t_{\bar{l}})}{t_{\bar{l}}} \xrightarrow{\bar{l}\to\infty} \bar{k}$. For an arbitrary element $t_{\underline{l}}$ of the first sequence let us define

$$t_{\bar{l}(\underline{l})} := \min\{t_{\bar{l}} : t_{\bar{l}} > t_{\underline{l}}\}\,.$$

Then we have that $t_{\bar{l}(\underline{l})} - t_{\underline{l}} \xrightarrow{\underline{l}\to\infty} \infty$. To see this, assume it would not be the case, then there would exist a constant $C \in \mathbb{R}^+$ such that $t_{\bar{l}(\underline{l})} - t_{\underline{l}} \leq C$. Thus, we could find for each $\underline{l}^* \in \mathbb{N}$ an $\varepsilon(\underline{l}^*) \geq 0$ with

$$\frac{m(t_{\underline{l}})}{t_{\underline{l}}} < \underline{k} + \varepsilon(\underline{l}^*)\,, \tag{3.1}$$

$$\frac{m(t_{\bar{l}})}{t_{\bar{l}}} > \bar{k} - \varepsilon(\bar{l}^*)\,, \tag{3.2}$$

for all $t_{\underline{l}}, t_{\bar{l}} > t_{\underline{l}^*}$ and further $\varepsilon(\underline{l}^*) \to 0$ for increasing $\underline{l}^*$. Hence we could choose $\underline{l}$ large enough to fulfill $t_{\underline{l}} > (\bar{k} - \underline{k} - 2\varepsilon)^{-1} \cdot \frac{C}{s}$. By adding and rearranging 3.1 and 3.2 we would obtain

$$\begin{aligned} m(t_{\bar{l}(\underline{l})}) - m(t_{\underline{l}}) &> t_{\bar{l}(\underline{l})}(\bar{k} - \varepsilon) - t_{\underline{l}}(\underline{k} - \varepsilon) \\ &= t_{\underline{l}}(\bar{k} - \underline{k} - 2\varepsilon) + (t_{\bar{l}(\underline{l})} - t_{\underline{l}})(\bar{k} - \varepsilon) \\ &> t_{\underline{l}}(\bar{k} - \underline{k} - 2\varepsilon) \\ &> \frac{C}{s}\,. \end{aligned}$$

On the other hand, we would also have that

$$\frac{C}{s} \geq \frac{t_{\bar{l}(\underline{l})} - t_{\underline{l}}}{s} \geq m(t_{\bar{l}(\underline{l})}) - m(t_{\underline{l}})\,,$$

and together

$$\frac{C}{s} \geq m(t_{\bar{l}(\underline{l})}) - m(t_{\underline{l}}) > \frac{C}{s}\,,$$

which is, of course, a contradiction. Hence we must have that $t_{\bar{l}(\underline{l})} - t_{\underline{l}} \xrightarrow{\underline{l}\to\infty} \infty$. Similarly, one can show that $t_{\underline{l}(\bar{l})} - t_{\bar{l}} \xrightarrow{\bar{l}\to\infty} \infty$. Denote now by L_m the duration from the beginning of the m-th loss period until the beginning of the $m+1$-th loss period. Since the arrivals are i.i.d., we also have that the L_m are i.i.d. and either $\frac{1}{n}\sum_{m=1}^{n} L_m \xrightarrow{n\to\infty} \mathbb{E}(L_m)$ or $\frac{1}{n}\sum_{m=1}^{n} L_m \xrightarrow{n\to\infty} \infty$ holds almost surely. We only consider the first case, since the second can be handled in the same way by setting "$\mathbb{E}(L_m) = \infty$".

We choose successively increasing intervals $t_{\bar{l}(\underline{l}_j)} - t_{\underline{l}_j}$ (respectively $t_{\underline{l}(\bar{l}_j)} - t_{\bar{l}_j}$). In these intervals an increasing number of loss periods appears. We also know from Equ. 3.1 and 3.2 that

$$\begin{aligned} \frac{m(t_{\bar{l}(\underline{l}_j)}) - m(t_{\underline{l}_j})}{t_{\bar{l}(\underline{l}_j)} - t_{\underline{l}_j}} &> \bar{k} - \varepsilon \\ \frac{m(t_{\underline{l}(\bar{l}_j)}) - m(t_{\bar{l}_j})}{t_{\underline{l}(\bar{l}_j)} - t_{\bar{l}_j}} &< \underline{k} + \varepsilon \end{aligned}$$

for all $\underline{l}$ (respectively $\bar{l}$) large enough. From this we obtain that

$$(\mathbb{E}(L_m))^{-1} = \left(\lim_{n\to\infty} \frac{\sum_{m=1}^{n} L_m}{n}\right)^{-1} = \lim_{j\to\infty} \frac{m(t_{\bar{l}(\underline{l}_j)}) - m(t_{\underline{l}_j})}{t_{\bar{l}(\underline{l}_j)} - t_{\underline{l}_j}} > \bar{k} - \varepsilon\,,$$

$$(\mathbb{E}(L_m))^{-1} = \left(\lim_{n\to\infty} \frac{\sum_{m=1}^{n} L_m}{n}\right)^{-1} = \lim_{j\to\infty} \frac{m(t_{\underline{l}(\bar{l}_j)}) - m(t_{\bar{l}_j})}{t_{\underline{l}(\bar{l}_j)} - t_{\bar{l}_j}} < \underline{k} + \varepsilon$$

and for $\varepsilon < \frac{\bar{k}-k}{2}$ this leads us to a contradiction. Hence

$$\liminf_{t\to\infty} \frac{m(t)}{t} < \limsup_{t\to\infty} \frac{m(t)}{t}$$

cannot hold. Since $\frac{m(t)}{t}$ is bounded from below and above it must hold:

$$\liminf_{t\to\infty} \frac{m(t)}{t} = \limsup_{t\to\infty} \frac{m(t)}{t}$$

hence the limit of $\frac{m(t)}{t}$ exists. □

We are now ready to prove that the conditions of Proposition 1 hold for i.i.d. interarrival times.

Proposition 4. *Let A_1, A_2 be two stochastically independent flows each with i.i.d. interarrival times with finite expectations and variances (i.e., independent renewal processes). For a server that requires s time units to serve a packet it is true that*

$$\exists K(s) \in [0,1) : \mathbb{P}\left(\lambda_i^- = K(s)\lambda_i\right) = 1 \quad i \in \{1,2\},$$

i.e., the condition of Proposition 1 is fulfilled and a finite FIFO buffer is rate-neutral for such sources.

Proof. We only prove that $\mathbb{P}(\lambda_1^- = K(s)\lambda_1) = 1$, since the case $i = 2$ is shown in the same fashion (note that $K(s)$ is the same in both cases, i.e., it does not depend on i). First we develop the following:

$$\lim_{t\to\infty} \frac{1}{t} \sum_{\substack{k \le A_1^-(t) \\ a_k^{(1)} \in \mathscr{L}}} I_k^{(1)} = K \qquad \text{a.s.} \tag{3.3}$$

$$\lim_{t\to\infty} \frac{1}{t} \sum_{\substack{k \le A_2^-(t) \\ a_k^{(2)} \in \mathscr{L}}} I_k^{(2)} = K \qquad \text{a.s.} \tag{3.4}$$

Here, $\mathscr{L}$ denotes the set of all lost packets and $a_k^{(i)}$ the arrival time of the k-th packet of the i-th flow (we also use the latter as identities for the packets). $I_k^{(i)}$ denotes the length of the interval $[a_k^{(i)}, a_{k+1}^{(i)}]$. Since 3.4 can be shown in the same way as 3.3 we only derive the first equation.

Consider the m-th loss period (m arbitrary) and denote by $\mathscr{L}_m$ the set of packets, which are lost in that period and with I_k the length of the k-th interarrival time (independent of the packets belonging to one or the other flow!)

$$I_k = a_{k+1} - a_k,$$

with a_k being the arrival time of the k-th packet (again it does not matter to which flow it belongs). Further denote by a_{m^*} the last time a packet arrived before the loss period $\mathscr{L}_m$ started (i.e. a_{m^*} is the packet, which filled the last free slot in the queue, marking the beginning of this loss period). Then we have

$$\sum_{\substack{k \\ a_k \in \mathscr{L}_m}} I_k = s + M_m - N_m$$

with

$$M_m := \min\{a_k : a_k > a_{m^*} + b\}$$

$$N_m := \min\{a_k : a_k > a_{m^*}\}$$

Since the $I_k^{(i)}$ are i.i.d., the random variables M_m and N_m are also i.i.d for differing $m \in \mathbb{N}$. In addition, we have by using $M_m, N_m \leq I_k$, that $\mathbb{E}(M_m)$, $\mathbb{E}(N_m)$, $\mathbb{V}(M_m)$, $\mathbb{V}(N_m) < \infty$. Considering all loss periods $\mathscr{L} = \bigcup_{m=1}^{\infty} \mathscr{L}_m$ we have that

$$\begin{aligned}
\lim_{t\to\infty} \frac{1}{t} \sum_{\substack{k \leq A_1^-(t) \\ a_k \in \mathscr{L}}} I_k &= \lim_{t\to\infty} \frac{1}{t} \sum_{m=1}^{m(t)} \sum_{\substack{k \leq A_1^-(t) \\ a_k \in \mathscr{L}_m}} I_k = \lim_{t\to\infty} \frac{1}{t} \sum_{m=1}^{m(t)} s + M_m - N_m \\
&= \lim_{t\to\infty} \frac{m(t)}{t} s + \sum_{m=1}^{m(t)} \frac{M_m}{m(t)} \cdot \frac{m(t)}{t} - \sum_{m=1}^{m(t)} \frac{N_m}{m(t)} \cdot \frac{m(t)}{t} \\
&= \lim_{t\to\infty} \frac{m(t)}{t} \left(s + \sum_{m=1}^{m(t)} \frac{M_m}{m(t)} - \sum_{m=1}^{m(t)} \frac{N_m}{m(t)} \right) \\
&= k \cdot (s + \mathbb{E}(M_m) - \mathbb{E}(N_m)) =: K(s)
\end{aligned}$$

almost surely. Here, we have used the strong law of large numbers in the last line. Hence, we almost reached3.3. Next, we analyze the difference between $\sum_{k : a_k \in \mathscr{L}_m} I_k$ and $\sum_{k : a_k^{(1)} \in \mathscr{L}_m} I_k^{(1)}$:

$$\sum_{\substack{k \\ a_k \in \mathscr{L}_m}} I_k = M'_m - N'_m + \sum_{\substack{k \\ a_k^{(1)} \in \mathscr{L}_m}} I_k^{(1)}$$

with

$$M'_m := [\min\{a_k^{(1)} : a_k^{(1)} > a_{m^*}\} - \min\{a_k^{(2)} : a_k^{(2)} > a_{m^*}\}]^+$$

and

$$N'_m := [\min\{a_k^{(1)} : a_k^{(1)} > a_{m'}\} - \min\{a_k^{(2)} : a_k^{(2)} > a_{m'}\}]^+$$

where $a_{m'}$ is the last lost packet in the m-th loss period. Note that M'_m and N'_m have the same distribution, thus

$$\begin{aligned}
\lim_{t\to\infty} \frac{1}{t} \sum_{\substack{k \leq A_1^-(t) \\ a_k^{(1)} \in \mathscr{L}}} I_k^{(1)} &= \lim_{t\to\infty} \frac{1}{t} \sum_{m=1}^{m(t)} \sum_{\substack{k \leq A_1^-(t) \\ a_k^{(1)} \in \mathscr{L}_m}} I_k^{(1)} \\
&= \lim_{t\to\infty} \frac{1}{t} \sum_{m=1}^{m(t)} N'_m - M'_m + \sum_{\substack{k \leq A_1^-(t) \\ a_k^{(1)} \in \mathscr{L}_m}} I_k \\
&= \lim_{t\to\infty} \frac{1}{t} \sum_{m=1}^{m(t)} \sum_{\substack{k \leq A_1^-(t) \\ a_k^{(1)} \in \mathscr{L}_m}} I_k \\
&= \lim_{t\to\infty} \frac{1}{t} \sum_{\substack{k \leq A_1^-(t) \\ a_k^{(1)} \in \mathscr{L}}} I_k \\
&= K(s),
\end{aligned}$$

if we can show that $\lim_{t\to\infty} \frac{1}{t} \sum_{m=1}^{m(t)} N'_m - M'_m = 0$ almost surely. For this define the random variables $X_m := N'_m - M'_m$, which have expectation zero and finite variance (since N'_m as

well as M'_m are bounded by interarrival times, which have finite variance). Hence we can use the strong law of large numbers and get:

$$\lim_{t\to\infty}\frac{1}{t}\sum_{m=1}^{m(t)}X_m=\lim_{t\to\infty}\frac{m(t)}{t}\sum_{m=1}^{m(t)}\frac{X_m}{m(t)}=0 \qquad \text{a.s.}$$

Next, we consider the following two equations:

$$\mathbb{P}\left(\lim_{t\to\infty}\frac{A_1(t)}{t}=\frac{1}{\mathbb{E}(I_1^{(1)})}\right)=1\,, \tag{3.5}$$

$$\mathbb{P}\left(\lim_{t\to\infty}\frac{A_1^-(t)}{t}=\frac{K(s)}{\mathbb{E}(I_1^{(1)})}\right)=1\,. \tag{3.6}$$

If they are fulfilled we can conclude this proof by (using Lemma 2)

$$1=\mathbb{P}\left(\lim_{t\to\infty}\frac{A_1^-(t)}{t}=\frac{K(s)}{\mathbb{E}(I_1^{(1)})}\right)=\mathbb{P}\left(\lim_{t\to\infty}\frac{\lambda_1^l(t)}{t}=\lambda_1 K(s)\right).$$

Equ. 3.5 is well known from renewal theory.

All left to prove is Equ. 3.6. Again, we start by the strong law of large numbers:

$$\begin{aligned}
1&=\mathbb{P}\left(\mathbb{E}(I_1^{(1)})=\lim_{n\to\infty}\frac{\sum_{k=1}^{n}I_k^{(1)}}{n}\right)=\mathbb{P}\left(\mathbb{E}(I_1^{(1)})=\lim_{A_1^-(t)\to\infty}\frac{\sum_{k=1}^{A_1^-(t)}I_k^{(1)}}{A_1^-(t)}\right)\\
&=\mathbb{P}\left(\mathbb{E}(I_1^{(1)})=\lim_{A_1^-(t)\to\infty}\frac{\sum_{k=1}^{A_1^-(t)}I_k^{(1)}}{t}\cdot\frac{t}{A_1^-(t)}\right)\\
&=\mathbb{P}\left(\mathbb{E}(I_1^{(1)})=\lim_{A_1^-(t)\to\infty}K(s)\cdot\frac{t}{A_1^-(t)}\right)=\mathbb{P}(\mathbb{E}(I_1^{(1)})=K(s)\cdot(\lambda_1^-)^{-1})\\
&=\mathbb{P}\left(\lambda_1^-=\frac{K(s)}{\mathbb{E}(I_1^{(1)})}\right)
\end{aligned}$$

This concludes the proof. □

Note that the i.i.d. assumption in Proposition **??** is per-flow, which means that the two flows may behave very differently (even in distribution); for example, one of them could be a CBR flow while the other could be from a Poisson source, so the proposition is quite flexible. On the other hand, it relies critically on the i.i.d. assumption, so we use simulations to investigate rate-neutrality under correlated sources.

3.2. **Simulation Study.** We study FIFO's rate-neutrality for pairs of competing flows using ns-2 packet-level simulations [14]. Experiments are carried out in a simulated dumbbell topology with a dedicated pair of sender and receiver nodes for each traffic flow. The bottleneck link's transmission rate is set to 50 Mbps. Access links are configured with 100 Mbps. Link propagation delays are set to 10 ms, i.e., the one-way end-to-end propagation delay is 30 ms. Each simulated traffic flow is comprised of a mix of 50, 100, 300, and 500 byte packets. Each experiment runs for 50 seconds of simulated time and is repeated 50 times with different seeds. The average results are reported, along with the 95% confidence interval, if visible in the graph. The source models used are Poisson and Pareto On-Off (with Hurst parameter 0.8). In both cases, the flows are an aggregate of 32

sources. The long-term input rates of both flows are configured to add up to 1. To assess the rate-neutrality, we compute the *throughput skew* between both flows as

$$\frac{\lambda_1^+ \lambda_2}{\lambda_1 \lambda_2^+},$$

where, with some abuse of notation, the rates are the empirically observed long-term rates of each flow. Under perfect rate-neutrality the throughput skew should be 1. If it is greater than 1, then Flow 1 is preferred, otherwise Flow 2. In Fig. 3.1, the simulation results for the rate-neutrality of different flow type combinations for different rate combinations (from the perspective of Flow 1) are displayed.

The following observations can be made: the case Poisson vs. Poisson is close to perfect, which is no surprise as Poisson arrivals are a renewal process with i.i.d. exponential inter-arrival times, thus validating the analytical result on FIFO's rate-neutrality from the previous section. Pareto vs. Pareto is not quite as perfect, although still within 3% of the ideal rate-neutrality. This is evidence that rate neutrality might not hold for a highly correlated, and thus non-renewal process like a Pareto On-Off source. In particular, there is a trend for the higher rate flow to be off a little worse.

While Poisson vs. Pareto is not far away from ideal rate-neutrality, it is interesting to observe that Poisson is consistently preferred over Pareto. The conjecture here is that a better matching between the server process (basically CBR) and the arrival process is beneficial for the FIFO rate allocation of the respective flow. From another perspective, in periods of high load where the buffer is almost full, bursty traffic loses more than smoother traffic which can cope better with the remaining space in the buffer. In a certain sense, under high load, FIFO has a burst-filtering characteristic. This bias for smoother traffic (and the corresponding small deviation from rate-neutrality) could rather be seen as a feature than a bug as it provides incentives for flows to be less bursty.

A final issue is the detailed convergence behavior of throughput rates, which determines the short-term vs. long-term rate neutrality. FIFO is instrumental in transforming the long-term rate-neutrality into a short-term equivalent, because it guarantees an optimal worst-case delay for all packets [15]. We have calculated the throughput rates for each flow over different interval sizes (from 1s to 50s). For Poisson traffic, even for 1s interval lengths the standard deviation from rate-neutrality is less than 3.5%, exhibiting a high convergence speed towards the asymptotic result. On the other hand, for Pareto we have observed a deviation of about 50% only in the extreme case of 1s intervals and very unbalanced input rate combinations.

4. DELAY DIFFERENTIATED FIFO

We give a high-level description of DDF. For maximum clarity, this description omits a few details, which are subsequently discussed in Section 6. A discussion of additional aspects that are not in the focus of this paper can be found in the original proposal [3]. DDF is a multi-class queuing system. Each service class is assigned a fixed queuing delay target and it is assumed that packets carry a service class identifier in their packet header.

The key concept of DDF is to decouple buffer admission control from a per-class delay admission control. When a packet arrives and the buffer has sufficient space, a service slot is inserted into a service queue and the buffer counter is increased. This is equivalent to the admission procedure for a typical finite FIFO queue, except that the service queue stores service slots rather than packets. The service slot is tagged with the class number and the time at which the packet would receive service in the corresponding FIFO queue.

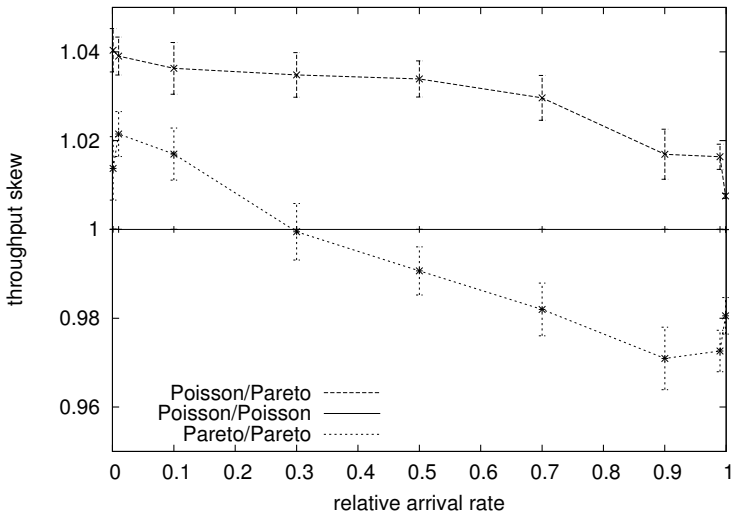

FIGURE 3.1. FIFO's rate-neutrality for different traffic combinations (simulated).

The service time can be trivially computed from the current time and the current buffer occupancy. The second admission step considers the service time of the oldest unused service slot for the packet's service class. If the service time compared to the current time is soon enough to meet the delay target of the service class, the slot is marked used for the current packet and the packet is admitted into the per-class packet queue.

When the link becomes idle, the next class to receive service is chosen according to the first slot in the service queue and the first packet from the class packet queue is transmitted. Both the service slot and the packet are removed from their respective queues. Because the buffer admission test is independent of the packet delay admission test, it is possible that a service slot in the service queue remains unused and that the corresponding class packet queue is empty at service time. We denote this event as *expired slot*.

Because DDF creates service slots using the same pattern as a regular FIFO queue for accepting packets, the slot queue tracks the corresponding FIFO queue closely. In fact, while a packet might be reordered with respect to packets from other classes, DDF maintains the exact FIFO service order at the level of service classes – a key property that is different from other schedulers that differentiate delay. However, DDF service occasionally deviates from FIFO service by not sending a packet corresponding to an expired slot.

It has been shown previously [3] that delay differentiation with DDF works very well, so the following analysis is focused on the attainable throughput rates compared to regular FIFO service. To assess the efficiency of DDF, it is necessary to fully understand the throughput loss caused by expired slots compared to FIFO service. To keep the problem tractable, the following analysis is based on the assumption that DDF service is not work-conserving, i.e., an expired slot leads to a short period during which the link is not utilized.

5. ANALYTICAL EVALUATION

The goal of the DDF analysis is to determine the probability of slot expiration. Furthermore, we gain valuable analytical insight that is fed back into the design of a prototype

implementation. To keep the system analytically tractable we make a number of simplifying assumptions: a discrete time model with one time slot being the service time of a unit packet is used; sources are fully described by their arrival rate (i.e, no correlations are assumed); no single source has more than one packet arriving in a single time slot, which essentially means geometrically distributed packet inter-arrival times (the discrete-time analogue of Poisson arrivals); arrivals in a time slot are assumed to happen before service takes place. Without loss of generality, we consider only two different classes of traffic, 1 and 2. We focus the analysis on Class 1 for which we assume a delay target of N_1 time slots. The total buffer capacity is $N \geq N_1$ slots. The arrival rates for Class 1 and 2 are denoted as r_1 and r_2, respectively. Note that the delay target of Class 2 is irrelevant for the analysis.

In the following, we first discuss the problems with modeling the overall system in a single (global) Markov chain, before we present an approach to find a closed-form solution using a decoupling between different system aspects to arrive at analytically tractable Markov chain models. Because some system aspects are not captured precisely by the decoupling approach, the system model is validated against simulations.

5.1. **Global Markov Chain for DDF.** Under the above assumptions, a global Markov chain could be created, which closely tracks the dynamics of the overall system. The state of the system would be captured by a vector whose entries could take three values: (1) slot filled (by a packet of Class 1 or reserved for Class 2), (2) slot reserved for Class 1, but not yet filled, (3) slot empty. We have experimented with this Markov chain model, but conclude that it is intractable for analysis and infeasible for a numerical solution. The analytical intractability is due to the complex structure of the Markov chain as well as a dramatic state space explosion; the latter also prohibits a numerical solution. For brevity, we omit a detailed discussion of the structure of the global Markov chain, but give a determination of the size of the state space. In particular, taking into account that empty slots always have to be trailing in the state vector and that the number of slots reserved for Class 1 is bounded by $N - N_1$, the size of the state space of the Markov chain is given by

$$\sum_{i=0}^{N} \sum_{j=0}^{\min\{N-N_1, N-i\}} \binom{N-i}{j}.$$

Example: $N = 1000$, $N_1 = 100$ yields $\sim 2.143 \times 10^{302}$ states.

5.2. **Decoupling Approach.** Since the global Markov chain is not a good choice for the analysis of DDF, we use an approximative approach that assumes a decoupling of the arrival and slot generation processes. With this decoupling, the system can be modeled as two Markov chains that interact with each other in a simple way. In particular, we use one Markov chain to model the slot generation process by looking at the total buffer dynamics to derive the drop rate due to a full slot queue. In addition, this Markov chain is used to determine *overload*, i.e., the probability that the number of service slots in the slot queue exceeds N_1, the delay target of Class 1. Following the argument made in Section 3 about the rate-proportionality of drops, the overall drop rate computed from this Markov chain can be used to drive another Markov chain, which keeps track of the position of the first empty slot in the relevant buffer spaces ranging from $1, \ldots, N_1$. This Markov chain is conditioned on the system being in overload, which provides another link between the two Markov chains. More details about the two chains and their interaction are given in Sections 5.2.1 and 5.2.2.

Based on the two Markov chains, we can calculate the steady-state probability for an expired slot of Class 1 as

$$\begin{aligned} p_{E_1} &= \lim_{t\to\infty} \mathbb{P}(E_1(t)) \\ &= \lim_{t\to\infty} [\mathbb{P}(E_1(t) \mid O_1(t-N_1+1)) \cdot \mathbb{P}(O_1(t-N_1-1))] \\ &= p_{E_1|O_1} \cdot p_{O_1}, \end{aligned}$$

where $E_1(t)$ denotes the event that a service slot of Class 1 expires at time t, and $O_1(t-N_1+1)$ denotes the event that the system was in overload at time $t-N_1-1$. Note that we apply the law of total probability correctly (in the second line) as $O_1(t-N_1+1)$ is necessary for $E_1(t)$, i.e., $E_1(t) \subset O_1(t-N_1+1)$. The steady-state probabilities $p_{E_1|O_1} = lim_{t\to\infty}\mathbb{P}(E_1(t) \mid O_1(t-N_1+1))$ and $p_{O_1} = \lim_{t\to\infty}\mathbb{P}(O_1(t-N_1-1))$ are calculated from the Markov chain for the empty slot position and from the Markov chain for the total buffer dynamics, respectively.

5.2.1. *Markov Chain for Total Buffer Dynamics.* We begin with the Markov chain that models the total buffer dynamics oblivious to traffic classes. Under the given assumptions, this is very similar to a Geo/D/1/N queue [16] with the slight difference that the arrivals are a superposition of two different flows with independent geometric inter-arrival times. Therefore, the overall arrivals are no longer a simple Bernoulli process (for example there can be more than one arrival per time slot). Still, the Markov chain is time-homogeneous, irreducible, finite, and aperiodic ($\rightarrow$steady-state probability distribution exists) and has a simple structure. The state variable i counts the number of slots in the queue and the transition probabilities are:

for State $i = 1,\ldots,N$: $p_{ii} = r_1(1-r_2) + r_2(1-r_1)$,

$$p = r_1 r_2,\ p_{i(i-1)} = (1-r_1)(1-r_2);$$

for State 0: $p_{00} = (1-r_1)(1-r_2) + r_1(1-r_2) + r_2(1-r_1)$,

$$p_{01} = r_1 r_2;$$

for State N: $p_{NN} = r_1 r_2 + r_1(1-r_2) + r_2(1-r_1)$,

$$p_{N(N-1)} = (1-r_1)(1-r_2).$$

Without going through the details, the regular structure of the Markov chain enables the calculation of its steady-state probability distribution by using the so-called detailed balance equations [17] between neighboring states:

$$q \neq 1:\ \pi_0 = \frac{1-q}{1-q^{N+1}}\ \forall i : 1 \le i \le N : \pi_i = q^i \pi_0,$$

$$q = 1 : \pi_0 = \frac{1}{N+1}\ \forall i : 1 \le i \le N : \pi_i = \pi_0,$$

where $q = \frac{r_1 r_2}{(1-r_1)(1-r_2)}$.

The steady-state probability of a packet drop due to a full buffer is denoted as $D(t)$ if it happens at time t and be can calculated as

$$p_D = \lim_{t\to\infty} \mathbb{P}(D(t)) = \pi_N r_1 r_2. \tag{5.1}$$

Also, the steady-state probability of being in overload can be calculated as

$$q \neq 1:\ p_{O_1} = \sum_{i=N_1+1}^{N} \pi_i = q^{N_1+1} \frac{1-q^{N-N_1}}{1-q^{N+1}},$$

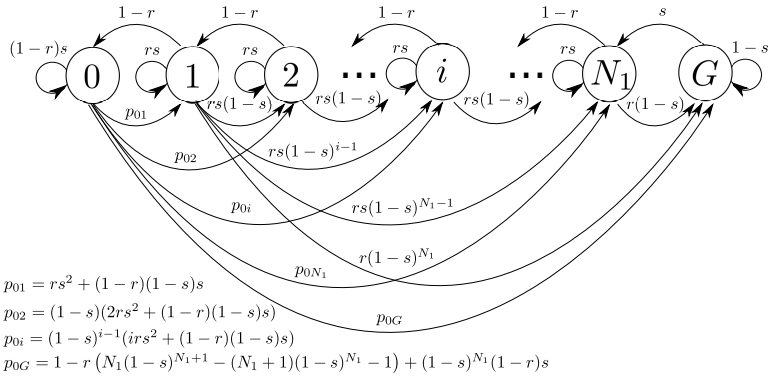

FIGURE 5.1. Markov chain keeping track of first empty slot in the slots $1,\ldots,N_1$.

$$q = 1:\ p_{O_1} = \sum_{i=N_1+1}^{N} \pi_i = \frac{N-N_1}{N+1}\,.$$

5.2.2. *Markov Chain for Empty Slot Dynamics.* The second Markov chain models the position of the first empty slot of Class 1 in the buffer spaces numbered 1 to N_1. If an empty slot reaches the head of the buffer and no packet arrives, this slot expires; this is accounted for by State 0 and thus we are interested in the the steady-state probability of State 0, denoted as π_o. Another peculiarity of this Markov chain is the State G, which accounts for the situation that no empty slot is in the buffer spaces from 1 to N_1. Remember that the Markov chain is conditioned on the system being in overload. Therefore, from State G new empty slots are generated with rate s. This is where the two Markov chains interact and the slot generation rate is set to $s = r(1-p_D)$, with p_D calculated from the Markov chain for the total buffer dynamics (see Eq. 5.1). For ease of presentation, we use s to denote the slot generation rate and r to denote the packet arrival rate of Class 1.

The Markov chain state diagram is shown in Fig. 5.1. It has a considerably more complex structure than the first Markov chain. The transition probabilities are as follows:

for State $i = 1,\ldots,N_1$: $p_{i(i-1)} = 1-r$,

$$\forall j : i \le j \le N_1 : p_{ij} = rs(1-s)^{j-i},\ p_{iG} = r(1-s)^{N_1-i+1};$$

for State 0:

$$p_{00} = (1-r)s,\ p_{0i} = (1-s)^{i-1}\left(irs^2 + (1-r)(1-s)s\right),$$

$$\begin{aligned} p_{0G} \quad &= \quad 1 - r\left(N_1(1-s)^{N_1+1} - (N_1+1)(1-s)^{N_1} - 1\right) \\ &\quad\ + (1-s)^{N_1}(1-r)s; \end{aligned}$$

for State G: $p_{GG} = 1-s,\ p_{GN_1} = s$.

While this Markov chain is also time-homogeneous, finite, aperiodic and irreducible ($\rightarrow$steady-state probability distribution exists), it is no longer reversible and thus not amenable to detailed balance equations. Instead, the global balance equations have to be solved. The following proposition provides the steady-state probability distribution of the empty slot Markov chain by solving the global balance equations in generality.

Proposition 5. *The steady-state probability distribution for the empty slot Markov chain is given as*

$$\pi_0 = \frac{1}{1+\sum_{i=1}^{N_1}\frac{\pi_i}{\pi_0}+\frac{\pi_G}{\pi_0}} = \frac{1}{\frac{r\left(\frac{1-s}{1-r}\right)^{N_1}}{s(r-s)}-\frac{1}{r-s}+r(1-s)^{N_1}},$$

$$\pi_i = \frac{(1-s)^{i-1}}{(1-r)^i}\left(1-s(1-r)^i\right)\pi_o \ \ , i=1,\ldots,N_1,$$

$$\pi_G = \frac{1}{s}\left(\frac{1-s}{1-r}\right)^{N_1}\left(1-s(1-r)^{N_1+1}\right)\pi_o\,.$$

Proof. The global balance equations are used for an induction over state $i=1,\ldots,N_1$. The induction hypothesis is the statement of the proposition regarding the $\pi_i, i=1,\ldots,N_1$:

$$\pi_i = \frac{(1-s)^{i-1}}{(1-r)^i}\left(1-s(1-r)^i\right)\pi_o \ \ , i=1,\ldots,N_1.$$

Basis: We start with the global balance equation for state 0

$$1-(1-r)s\pi_0 = (1-r)\pi_1 \Rightarrow \pi_1 = \frac{1-s(1-r)}{1-r}\pi_0 = \frac{(1-s)^0}{(1-r)^1}\left(1-s(1-r)^1\right)$$

which provides the induction hypothesis for $i=1$.

Inductive step: Now we use the global balance equation for state $i>1$, in particular:

(5.2)

$$(1-r)\pi_{i+1}+\sum_{j=1}^{i-1}rs(1-s)^{i-j}\pi_j+\left(irs^2(1-s)^{i-1}+(1-r)s(1-s)^i\right)\pi_0 = (1-rs)\pi_i$$

Using the induction hypothesis for all π_j with $j \leq i$ and some rearranging we obtain

$$\begin{aligned}
\pi_{i+1} = &\frac{\pi_0}{1-r}\left[(1-rs)\frac{(1-s)^{i-1}}{(1-r)^i}\left(1-s(1-r)^i\right) - \sum_{j=1}^{i-1} rs(1-s)^{i-j}\frac{(1-s)^{j-1}}{(1-r)^j}\left(1-s(1-r)^j\right)\right.\\
&\left. - \left(irs^2(1-s)^{i-1} + (1-r)s(1-s)^i\right)\right]\\
=&\frac{(1-s)^{i-1}}{(1-r)^{i+1}}\pi_0\left[(1-rs)\left(1-s(1-r)^i\right) - \sum_{j=1}^{i-1} rs(1-r)^{i-j}\left(1-s(1-r)^j\right)\right.\\
&\left. - (1-r)^i irs^2 - (1-r)^{i+1}s(1-s)\right]\\
=&\frac{(1-s)^{i-1}}{(1-r)^{i+1}}\pi_0\left[1-s(1-r)^i - \sum_{j=1}^{i} rs(1-r)^{i-j}\left(1-s(1-r)^j\right)\right.\\
&\left. - (1-r)^i irs^2 - (1-r)^{i+1}s(1-s)\right]\\
=&\frac{(1-s)^{i-1}}{(1-r)^{i+1}}\left[\pi_0 1 - s(1-r)^i - \sum_{j=1}^{i} rs(1-r)^{i-j} - \sum_{j=1}^{i} rs^2(1-r)^{i-j}\right.\\
&\left. - (1-r)^i irs^2 - (1-r)^{i+1}s(1-s)\right]\\
=&\frac{(1-s)^{i-1}}{(1-r)^{i+1}}\pi_0\left[1-s(1-r)^i - \sum_{j=1}^{i} rs(1-r)^{i-j} - (1-r)^{i+1}s(1-s)\right]\\
=&\frac{(1-s)^{i-1}}{(1-r)^{i+1}}\pi_0\left[1-s(1-r)^i - s\left(1-(1-r)^i\right) - (1-r)^{i+1}s(1-s)\right]\\
=&\frac{(1-s)^i}{(1-r)^{i+1}}\left(1-s(1-r)^{i+1}\right)\pi_0,
\end{aligned}$$

thereby showing that the rule in the proposition indeed holds for π_{i+1}.

π_G has to be solved separately using the global balance equation for state N_1, in particular

$$s\pi_g + \sum_{j=1}^{N_1-1} rs(1-s)^{N_1-j}\pi_j + \left(N_1 rs^2(1-s)^{N_1-1} + (1-r)s(1-s)^{N_1}\right)\pi_0 = (1-rs)\pi_{N_1}.$$

This has the same form as Equ. 5.2, apart from the factor s in the first term (instead of $1-r$). Thus, essentially the same derivation as in the inductive step above can be performed to obtain

$$\pi_G = \frac{(1-s)^{N_1}}{s(1-r)^{N_1}}\left(1-s(1-r)^{N_1+1}\right)\pi_0 = \frac{1}{s}\left(\frac{1-s}{1-r}\right)^{N_1}\left(1-s(1-r)^{N_1+1}\right)\pi_0.$$

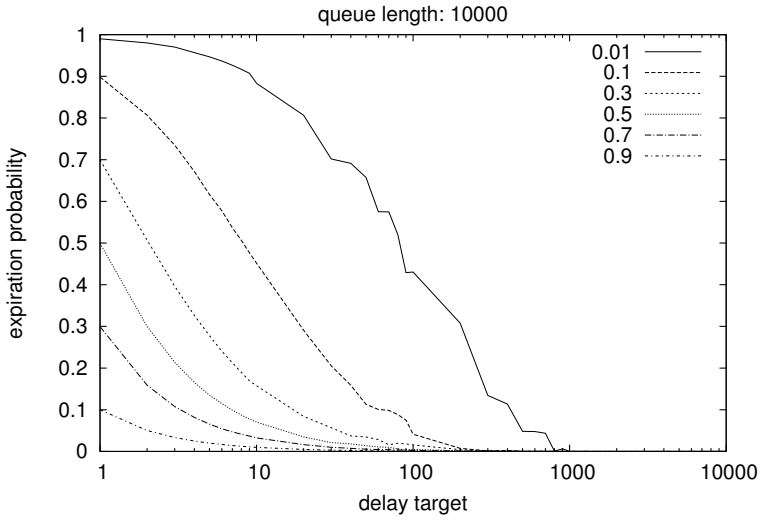

FIGURE 5.2. Slot-level simulation results of DDF.

π_0 is obtained from the normalization condition as

$$\pi_0 = \frac{1}{1+\sum_{i=1}^{N_1}\frac{\pi_i}{\pi_0}+\frac{\pi_G}{\pi_0}} = \frac{1}{1+\sum_{i=1}^{N_1}\frac{(1-s)^{i-1}}{(1-r)^i}\left(1-s(1-r)^i\right)+\frac{1}{s}\left(\frac{1-s}{1-r}\right)^{N_1}\left(1-s(1-r)^{N_1+1}\right)}$$

$$= \frac{1}{1+\sum_{i=1}^{N_1}\frac{(1-s)^{i-1}}{(1-r)^i}\left(1-s(1-r)^i\right)+\frac{1}{s}\left(\frac{1-s}{1-r}\right)^{N_1}-(1-r)(1-s)^{N_1}}$$

$$= \frac{1}{1+\frac{\left(\frac{1-s}{1-r}\right)^{N_1}-1}{r-s}-1+(1-s)^{N_1}+\frac{1}{s}\left(\frac{1-s}{1-r}\right)^{N_1}-(1-r)(1-s)^{N_1}}$$

$$= \frac{1}{\frac{\left(\frac{1-s}{1-r}\right)^{N_1}-1}{r-s}+r(1-s)^{N_1}+\frac{1}{s}\left(\frac{1-s}{1-r}\right)^{N_1}}$$

$$= \frac{1}{\frac{r\left(\frac{1-s}{1-r}\right)^{N_1}}{s(r-s)}-\frac{1}{r-s}+r(1-s)^{N_1}}.$$

Note that in the third line we assumed $s \neq r$, which however is always true for a finite queue (in fact, we have $s < r$). This concludes the proof. □

This provides us with the steady-state probability for an expired slot of Class 1 (under the condition of overload):

$$p_{E_1|O_1} = \pi_0 .$$

5.2.3. *Bringing It All Together.* Now, the probability of an expired slot for Class 1 can be calculated and we can assess its throughput loss under DDF. We (re)set $r_1 = r$ and $s = (1-p_D)r_1$, as well as $q = \frac{r_1 r_2}{(1-r_1)(1-r_2)}$, and calculate the (steady-state) fraction of

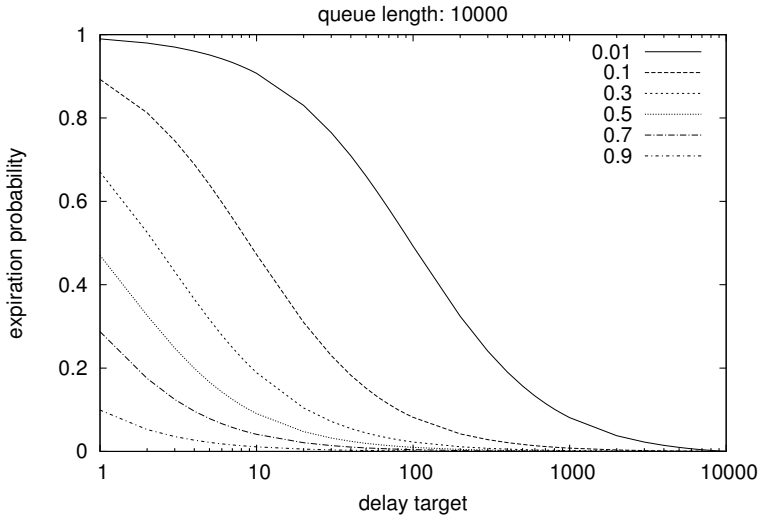

FIGURE 5.3. Markov chain model results for full load ($\rho = 1.0$).

expired slots for Class 1 as

$$q \neq 1: \ \frac{p_{E_1|O_1} \cdot p_{O_1}}{r_1} = \frac{1}{\frac{r_1^2\left(\frac{1-s}{1-r_1}\right)^{N_1}}{s(r_1-s)} - \frac{r_1}{r_1-s} + r_1^2(1-s)^{N_1}} \cdot q^{N_1+1} \frac{1-q^{N-N_1}}{1-q^{N+1}}; \tag{5.3}$$

$$q = 1: \ \frac{p_{E_1|O_1} \cdot p_{O_1}}{r_1} = \frac{1}{\frac{r_1^2\left(\frac{1-s}{1-r_1}\right)^{N_1}}{s(r_1-s)} - \frac{r_1}{r_1-s} + r_1^2(1-s)^{N_1}} \cdot \frac{N-N_1}{N+1}.$$

5.3. **Model Validation and Insights.** The decoupling approach provides us with a closed-form solution for the probability of expiration. However, it is neither completely clear how good the approximation of the decoupling is nor what the formulas tell us exactly. In this section we describe a validation of the model against simulations and attempt to extract insights from the model.

5.3.1. *Validation.* The model is first compared to an abstract slot-level simulation of DDF as described in Section 4, using the traffic assumptions stated for the Markov chain model (i.e., geometric inter-arrival times). The simulation is configured with a total queue length of $N = 10000$, which, for example, corresponds to a 200ms buffer of 125 byte packets at a 50 Mbps link. The delay target N_1 ranges from 1 to N. Fig. 5.2 shows the resulting expiration probabilities for different rates r_1 of the traffic flow studied. The rate of background traffic r_2 is set to $1 - r_1$, so the system is fully loaded. The simulation result is compared to the expiration probabilities computed from the Markov model using the same configuration parameters, shown in Fig. 5.3. Note the logarithmic scale on the x-axis to highlight the low delay targets better as for higher delays the expiration probability is negligible. The two graphs exhibit a strong similarity, which confirms the validity of the decoupling approximation and provides a sanity-check of its closed form.

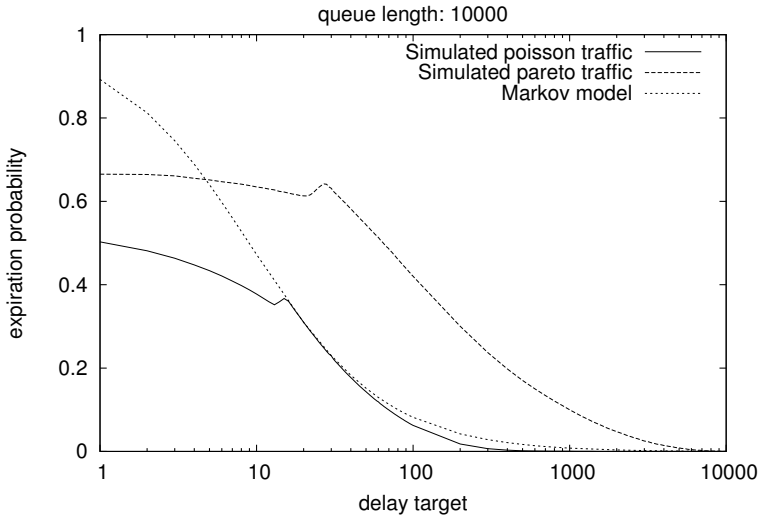

FIGURE 5.4. Packet-level simulation result of DDF.

In a second step, the model is compared to an ns-2 packet-level simulation of DDF, which is set up to correspond to an average arrival rate of $r_1 = 0.1$. The results in Fig. 5.4 show one run with Poisson and one with Pareto arrivals and a noticeable buckle and deviation for small delay targets. We have not yet been able track down the exact cause for this behavior, but strongly assume that it is an artifact of ns-2. For larger delay targets, it can be seen that the Markov model only slightly over-estimates the expiration probability for Poisson traffic, but clearly underestimates the expirations for Pareto arrivals as it does not capture its correlations. However, it is important to note that the shape of the curve is well captured for both traffic types. Thus, we conclude that the model is a useful representation of the real system.

5.3.2. *Insights.* To gain some insights from the Markov chain model, we show a number of numerical experiments. We evaluate the influence of different total loads $\rho = r_1 + r_2$: high load ($\rho = 0.95$) and full load ($\rho = 1.0$). Note that lower loads present no challenge to the system, since expiration probabilities become negligible. The results of these experiments are shown in Fig. 5.3 (again) and 5.5. The number of additionally dropped packets (from expired slots) due to DDF are small in most cases and quickly converge to zero as the delay target approaches the buffer capacity. In particular, for high load the probability of slot expiration is negligible for realistic delay targets, regardless of the rate combination. Only under full load and if the class of interest has a very stringent delay target the expiration probability becomes non-negligible. Intuitively, the class of interest is often not able to fill empty slots, because its rate is too low for a an arriving packet to find a slot from a previous packet within the time frame of the delay target. Asymptotically, this can also be seen directly in the analytical model, if we let $r_1 \to 0$, and for ease of exposition assume that we have an infinite queue ($s = r_1$), then we obtain for the fraction of expired class 1 slots under full load ($\Rightarrow p_{O_1} = 1$)

$$\frac{p_{E_1|O_1} \cdot p_{O_1}}{r_1} = \frac{1}{N_1 \frac{r_1}{1-r_1} + r_1^2 (1-r_1)^{N_1} + 1} \xrightarrow{r_1 \to 0} 1 .$$

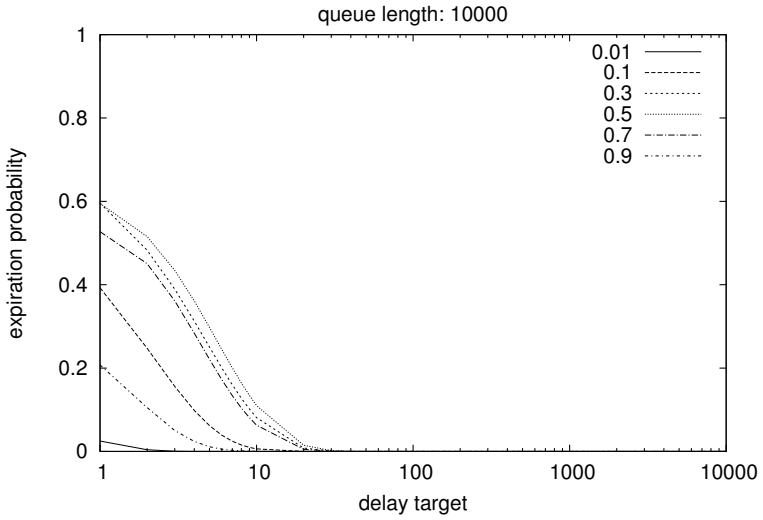

FIGURE 5.5. Markov chain model results for high load ($\rho = 0.95$).

Name	Explanation
q_S	main slot queue: class, size, time
q_P	main packet queue: time, packet (sorted by time)
b	main queue backlog including *unused* slots
m_c	meta slot queue of class 'c': pointers to elements in q_S
D_c	delay target of class 'c'
x_c	next allowable admission exception time for class 'c'
p_c	number of packets queued for service for class 'c'
NWC	non-work-conserving configuration

TABLE 1. DDF Variables and Routines

However, for a finite queue, if $N_1 \to N$ then $p_{O_1} \to 0$ and thus the fraction of expired slots goes to zero. However, in some cases this happens slower than may be desirable for small r_1.

There are two options to deal with this problem: (1) avoid such a configuration, (2) modify DDF in such situations to "help" low-rate, low-delay classes. In both cases, the analytical model helps to detect such a situation. In particular, for option (1), given a target expiration probability and a target delay we can solve Eq. 5.3 for r_1 and thus compute the rate threshold below which we run into this problem. The second option is discussed in Section 6, where implementation issues of DDF are presented in more detail.

6. IMPLEMENTATION

6.1. **Algorithm Details.** The analysis in Section 5 demonstrates that the DDF queuing algorithm has very promising properties, but there are also three caveats:

- The analysis only applies to uncorrelated traffic sources. Therefore, we complement the analysis by a simulation study that also includes correlated traffic sources.

Algorithm 1 DDF Arrival Routine

```
1:  p ← received packet
2:  l_p ← size of received packet
3:  c ← service class of p
4:  if not NWC then EXPIRY(t + b − B)
5:  if b < B then
6:      tail(q_S) ← {c, l_p), t + b)}
7:      tail(m_c) ← tail(q_S)
8:      acSuccess ← true
9:  if x_c < t − B then x_c ← t − B
10: if size(m_c) ≥ l_p and head(m_c).time ≤ t + D_c then
11:     q_P[t + b] ← p
12:     b ← b + l_p
13:     p_c ← p_c + 1
14:     while l_p > 0 do
15:         if l_p <head(m_c).size then
16:             head(m_c).size ← head(m_c).size - l_p
17:             l_p ← 0
18:         else
19:             l_p ← l_p−head(m_c).size
20:             remove_head(m_c)
21: else if p_c = 0 and t > x_c and acSuccess then
22:     x_c ← x_c + D_c
23:     q_P[t] ← p
24:     b ← b + l_p
25:     p_c ← p_c + 1
26: else drop(p)
```

- The basic algorithm described in Section 4, as well as the analytical model, does not account for variable packet lengths. This is addressed in the implementation.
- The analysis reveals that a low-delay service class might suffer a large number of expirations, if the arrival rate is too low, i.e., if packet inter-arrival times are not sufficiently small compared to the class delay target. This is addressed by a small extension to the basic algorithm.

In previous work [3] an algorithm is given, which provides work-conserving service in spite of expiring slots, by keeping packets in a separate sorted packet queue and managing slot expiration in the arrival routine. Thereby, the original algorithm proposed has constant algorithmic complexity (using cost amortization over the length of one arriving packet), but relies on a priority queue with constant search complexity. While such a priority queue can be implemented under certain assumptions, it is still more complex and costly than a simple FIFO queue, especially in the harsh low-level execution environment of a network router.

We have implemented a non-work-conserving version of DDF to show an alternative implementation strategy that does not rely on a constant-search priority queue. This implementation is conceptually closer to the analytical model and more clearly exposes the differences compared to FIFO queuing. It handles variable packet lengths and contains an extension to address low-rate traffic in low-delay classes. With this type of implementation

Algorithm 2 DDF Service Routine

```
1: if q_P not empty then
2:     if NWC then
3:         while head(q_P) not accessible do idle()
4:     p ← remove_head(q_P)
5:     b ← b − l_p
6:     p_c ← p_c − 1
7:     transmit(p)
8:     if NWC then EXPIRY(t)
```

Algorithm 3 DDF Expiry Routine

```
1: function EXPIRY(t)
2:     while head(q_S).time < t do
3:         d ← head(q_S).class
4:         if head(q_S) = head(m_d) then
5:             b ← b−head(m_d).size
6:             remove_head(m_d)
7:         remove_head(q_S)
```

a small amount of link utilization is sacrificed to keep the implementation complexity low. While the evaluation is focused on the non-work-conserving version, the implementation is general enough to support both work-conserving and non-work-conserving operation. The implementation somewhat deviates from the description in Section 4 to achieve this flexibility. In particular, packets are stored in a central packet queue q_P, which is sorted by service time stamps. For non-work-conserving service, q_P can be a array-based timer wheel that is scanned for packets as real time progresses. For work-conserving service, q_P has to be a constant-search priority queue, as in previous work [3]. Note that the non-work-conserving variant can also be implemented using only FIFO queues as described in Section 4, because there is no inherent requirement for a sorted packet queue. We present the arrival and service routines in Algorithms 1 and 2. A third routine 'expiry' is shown in Algorithm 3. Variables are summarized in Table 1.

In the arrival routine, Lines 6-10 show the basic admission test, which adds a slot to the main slot queue q_S, if there is sufficient space. Line 13 is the delay admission test, which determines whether the oldest unused slot meets the class delay target and whether enough slots are available for the packet (since a slot might represent a previous, smaller packet). If yes, the packet is added to the packet queue in Line 14. Lines 17-25 show the reconciliation of different packet lengths. The loop iterates over as many slots as are need to cover the size of the packet. Used slot are removed from the per-class slot queue (Lines 22, 23). If the last slot that is used for this packet is too big, its size is adjusted (Lines 18-20).

The expiry routine removes all old slots from the main slot queue and checks whether any of these slots has not been used by comparing with the corresponding per-class slot queue. If an unused slot is removed, this results in a reduction of the backlog counter (Line 5), which effectively expires this slot. In the work-conserving variant, the expiry routine is called during arrival and executes in proportion to the size of arriving packets. A side effect is that unused slots are not expired immediately, which might be beneficial for bursty traffic.

The service routine chooses the next packet from packet queue and transmits it (Lines 6-9). In the non-work-conserving variant, the system might idle while scanning the timer wheel for the next packet available for transmission. Slot expiry is performed in proportion to the idle time and next packet. In this variant, slot expiration is aggressive, i.e., unused slots are expired immediately when their service time has been reached.

Lines 11, and 26-30 in the arrival routine show an extension to the basic algorithm, which handles low-delay classes with low-rate traffic. Following the discussion in Section 5.3.2, a service class needs to have at least one packet arrival per delay target interval to have any reasonable chance that a slot traveling through the delay target region of the main slot queue is picked up by an arriving packet. This is equivalent to a per-class rate allocation of B/D_c packets. The given code emulates such a rate allocation by permitting as many packets to bypass the delay admission test, if the class is otherwise empty and the basic admission test has been successful. Those packets are then forwarded immediately. During times of low-rate arrivals, all packets of a class might be forwarded this way. If the arrival rate for a service class increases, slots will eventually enter the delay target region and be picked up by arriving packets, in which case the bypassing mechanism will cease to forward packets according to the first condition in Line 26. While bypassing the delay admission test in this way distorts the delay targets of existing packets, the effect of this distortion is limited, since it is restricted to low-rate classes or short start-up periods and the rate is limited to B/D_c packets by keeping track of the next allowable exception time x_c.

6.2. **Simulation.** The DDF algorithm has been implemented in the ns-2 simulation environment [14], following the pseudo-code given in the previous section. The simulated environment is identical to the one described in Section 3.2. The DDF queue is configured with two service classes for 10ms and 100ms respectively. The behavior of DDF is investigated in various interesting scenarios and compared to a benchmark FIFO queue that is configured with a buffer size equivalent to 100 ms. In all DDF experiments, two traffic flows are created. One *foreground* flow uses the 10ms service class, while the other *background* flow uses the 100ms class. We show the overall loss compared to the benchmark FIFO queue, as well as the throughput skew from the point of view of the foreground flow. The relative arrival rate of the foreground flow is varied and the background rate is adjusted, so that the total average arrival rate is always set to 1.

In the first experiment (Fig. 6.1) both the foreground and the background flow are comprised of Poisson traffic. DDF forwards this traffic almost perfectly with practically no extra loss. The non-work-conserving version slightly disadvantages the foreground traffic at a low-rate, i.e., the low-rate extension to the algorithm is not yet fully sufficient for this version of DDF. Still, the following experiments study the non-work-conserving version, since it ultimately has lower implementation complexity – comparable to FIFO queuing.

In a second experiment (Fig 6.2) the background flow is comprised of bursty Pareto traffic. In this scenario, DDF causes extra losses compared to the benchmark and the low-rate effect is visible. The extra losses caused by DDF are the trade-off for delay differentiation. To put these losses in perspective, it is instructive to study the behavior of a 10ms FIFO buffer, which would be needed guarantee the low delay. Results for a 10ms FIFO buffer are shown in Fig. 7.1, benchmarked against the 100ms FIFO queue. The results show a smaller FIFO queue causes considerably more losses than DDF.

The last experiment reported here illustrates the incentive structure of DDF by assuming that the background Pareto traffic enters the 10ms service class due to misconfiguration or

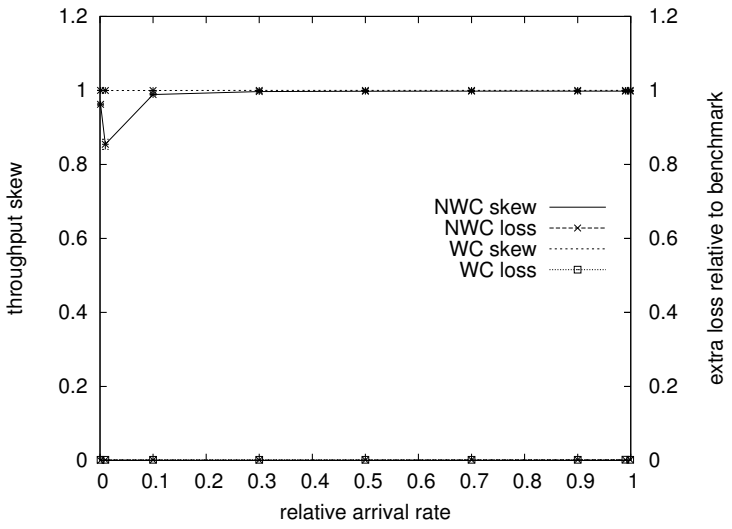

FIGURE 6.1. DDF - Poisson 10 / Poisson 100

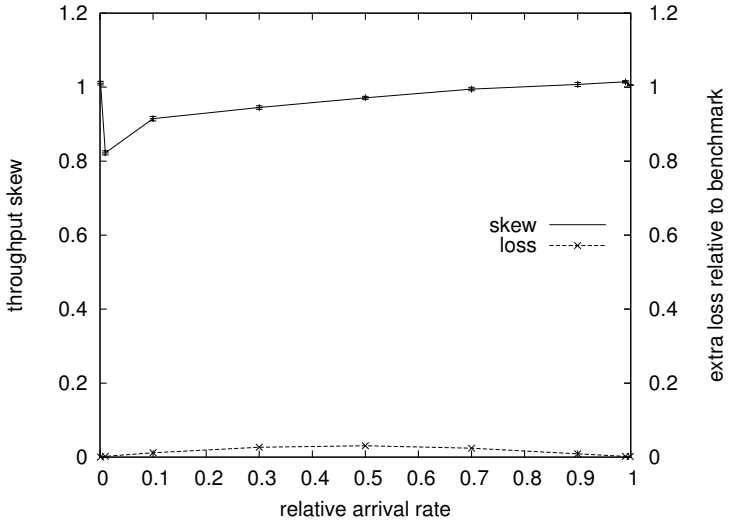

FIGURE 6.2. DDF - Poisson 10 / Pareto 100

ill will. The results are shown in Fig. 7.2 and show that DDF essentially falls back to providing the same service as the 10ms FIFO queue shown previously.

The delay differentiation capabilities of DDF have been illustrated before [3] and are not repeated here. However, all measurements generated during the simulation experiments are verified and the expected queuing delays are met at all times.

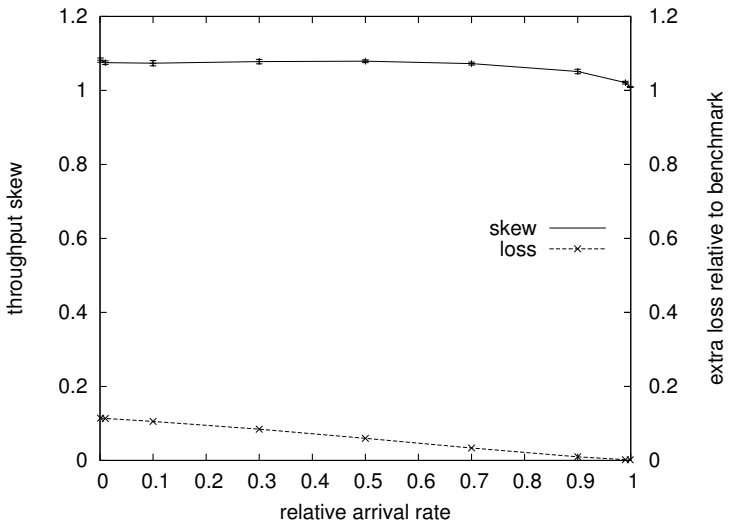

FIGURE 7.1. FIFO 10 - Poisson / Pareto

7. CONCLUSIONS

This paper presents an in-depth study of Delay Differentiated FIFO (DDF) Queuing and investigates its feasibility. The objectives stated in the introduction are largely met. We establish that a shared FIFO buffer is in fact rate-neutral, at least for i.i.d. traffic sources. Additional simulations indicate that FIFO does not deviate much from long-term rate neutrality for non-i.i.d. sources. Markov modeling is used to determine the probability of slot expirations, which is the key performance measure to assess the deviation of DDF from FIFO. The Markov model is validated by simulation experiments and the analytical results are complemented with additional simulations to study traffic sources that are not covered by the model. It turns out that the basic DDF algorithm too aggressively discards packets from low-delay service classes that have low-rate traffic arrivals. This problem is somewhat inherent to DDF, but represents a corner case. We suggest a simple extension of the DDF algorithm, which is promising while not yet perfect, and will be the subject of future work. In fact, any existing proposal to control and differentiate queuing delay in a packet-switched network comes with undesirable side effects – there is no free lunch! DDF is a promising approach to manage queuing delays in a truly decoupled fashion from other aspects of network service.

REFERENCES

[1] J. Gettys and K. Nichols, "Bufferbloat: dark buffers in the internet," *Commun. ACM*, vol. 55, no. 1, pp. 57–65, Jan. 2012.

[2] A. Vishwanath, V. Sivaraman, and M. Thottan, "Perspectives on router buffer sizing: recent results and open problems," *SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 2, pp. 34–39, Mar. 2009.

[3] M. Karsten, "FIFO service with differentiated queueing," in *Proceedings of ACM/IEEE ANCS*, Oct. 2011, pp. 122–133.

[4] S. Blake, D. L. Black, M. A. Carlson, E. Davies, Z. Wang, and W. Weiss, "RFC 2475 - An Architecture for Differentiated Services," Dec. 1998.

[5] P. Eardley, "RFC 5559 - Pre-Congestion Notification (PCN) Architecture," Jun. 2009.

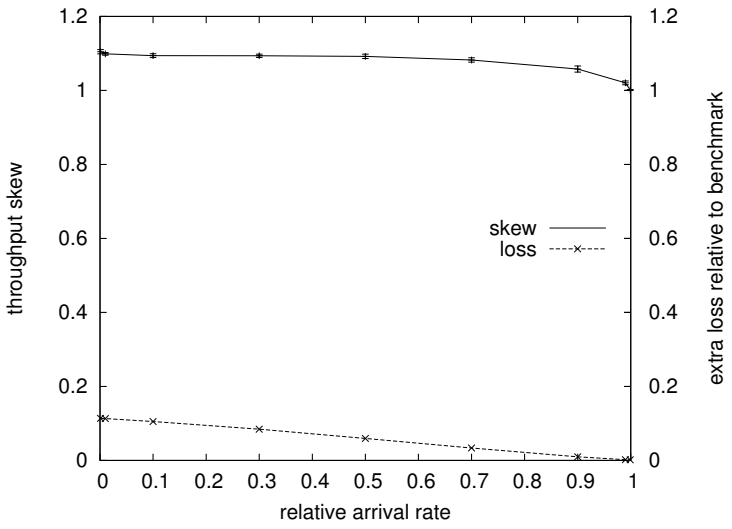

FIGURE 7.2. DDF - Poisson 10 / Pareto 10

[6] P. Hurley, J.-Y. L. Boudec, P. Thiran, and M. Kara, "ABE: Providing a Low-Delay Service within Best-Effort," *IEEE Network*, vol. 15, no. 3, pp. 60–69, May 2001.

[7] M. Podlesny and S. Gorinsky, "Rd network services: differentiation through performance incentives," in *Proceedings of ACM SIGCOMM*, 2008, pp. 255–266.

[8] M. Mathis, "Reflections on the TCP Macroscopic Model," *ACM Computer Communications Review*, vol. 38, no. 1, pp. 47–49, Jan. 2009.

[9] K. Nichols and V. Jacobson, "Controlling queue delay," *ACM Queue*, vol. 10, no. 5, May 2012.

[10] B. Melander, M. Björkman, and P. Gunningberg, "A new end-to-end probing and analysis method for estimating bandwidth bottlenecks," in *Proceedings of IEEE GLOBECOM*, Nov. 2000, pp. 415–420.

[11] Y. Ghiassi-Farrokhfal and J. Liebeherr, "Output characterization of constant bit rate traffic in fifo networks," *IEEE Communications Letters*, vol. 13, no. 8, pp. 618–620, 2009.

[12] F. Ciucu, O. Hohlfeld, and L. Y. Chen, "On the convergence to fairness in overloaded fifo systems," in *Proceedings of IEEE INFOCOM*, Apr. 2011, pp. 1988–1996.

[13] Y. Ghiassi-Farrokhfal and F. Ciucu, "On the impact of finite buffers on per-flow delays in fifo queues," in *Proceedings of ITC 24*, Sep. 2012, forthcoming.

[14] "The network simulator - ns-2," http://nsnam.isi.edu/nsnam/index.php, Accessed Jul 22, 2012.

[15] J. Schmitt, N. Gollan, S. Bondorf, and I. Martinovic, "Pay bursts only once holds for (some) non-FIFO systems," in *Proceedings of IEEE INFOCOM*, Apr. 2011, pp. 1970–1978.

[16] A. Gravey, J.-R. Louvion, and P. Boyer, "On the geo/d/1 and geo/d/1/n queues," *Perform. Eval.*, vol. 11, no. 2, pp. 117–125, 1990.

[17] F. P. Kelly, *Reversibility and Stochastic Networks*. Wiley, 1979.