

QoS is ... Quite often Stupid!
James Roberts

The success of the Internet lies in its simplicity; however, this comes at a cost of only best effort non-differentiated service. For years, institutions such as the IETF have been trying to introduce a QoS architecture to the current IP network. Unfortunately, the proposed QoS models, i.e., IntServ [1] and DiffServ [2,3], are not suitable for the Internet as a whole. To provide a service at a reasonable level, under the terms of congestion, some priorities and discriminations must be imposed. The aforementioned architectures propose the use of a reservation protocol and a packet marking scheme, respectively; however, these solutions require proper inter-domain agreements, complex router implementations, and, most of all, end user compliance. Besides IntServ and DiffServ, many other QoS architectures have been proposed for IP networks.

An efficient and robust QoS architecture for IP networks requires that the user-network interface remains the same as today, no signaling protocol or packet marking is introduced, and no new user-operator or operator-operator agreements are signed. These constraints are very strict, yet they have been met. This chapter introduces a novel approach to achieving QoS guarantees in the Internet—Flow-Aware Networking, or FAN for short.

The description of FAN starts with Sect. 2.1 which shows why the new QoS architecture is needed. Section 2.2 describes the basic concepts of FAN. Sections 2.3 and 2.4 introduce the flow-aware approach and a Cross-Protect (XP) router, respectively. Section 2.5 describes one of the FAN-specific mechanisms i.e., measurement-based admission control, while fair queuing algorithms are presented and compared in Sect. 2.6. Section 2.7 briefly surveys other mechanisms and architectures which have been proposed for Flow-Aware Networks.

2.1 The Need for a New QoS Architecture

IETF introduced two ideas on how to assure QoS. Chronologically, the first was Integrated Services. IntServ has many advantages, such as real (as opposed to statistical) assurances, easy control in nodes, use of a reservation protocol, and the option to create various traffic profiles. However, there are certain disadvantages, which make IntServ unsuitable for larger networks. These include maintaining information about all flows in every node and demanding that end users explicitly define required transmission parameters. These pros and cons make IntServ a good solution for dealing with a small network, where all the end users are known, traffic is mostly defined, and every router in the network can be easily configured by a single network operator.

To overcome the scalability issue, IETF introduced a new idea—the Differentiated Services. At the cost of certain constraints, DiffServ avoids the problems that eventually halted the development of its predecessor. That is why in DiffServ the assurances are statistical and the admission control blocks are only placed at the borders of each DiffServ domain. Moreover, the inner nodes do not keep the flow information, which suits it better to larger networks; however, the scalability issue is not completely overcome. Still, all routers in a domain must be pre-configured so that the per-hop behavior matches the actual classes of service which are provided inside the domain. Although DiffServ is more flexible and scalable than its predecessor, it still has features which make it unsuited for extra-large networks like the Internet.

It can be said that IntServ and DiffServ represent a trade-off between fine service granularity and scalability. Over the years, many attempts to alleviate this relationship have been proposed, including combined use of IntServ and DiffServ, or new and better congestion control mechanisms cooperating with the service isolation provided by DiffServ. However, no solution has attracted enough attention to be widely implemented and used.

2.2 Basic Concepts of FAN

Flow-Aware Networking is a new direction for QoS assurance in IP networks. The original idea was initially introduced by J. Roberts et al. in [4,5] and then presented as a complete system in 2004 [6,7]. Their intention was to design a novel QoS architecture that would be possible to use in networks of all sizes, including the global IP network—the Internet. In [8], the belief that adequate performance can be assured much more simply than in classical QoS architectures and more reliably than in over-provisioned best effort networks is expressed.

The goal of FAN is to enhance the current IP network by improving its performance under heavy congestion. To achieve this, certain traffic management mechanisms to control link sharing are proposed, namely measurement-based admission control [8] and fair scheduling with priorities [6,9]. The former is used to keep the flow rates sufficiently high to provide a minimal level of performance for each flow in case of overload. The latter realizes the fair sharing of link bandwidth while ensuring

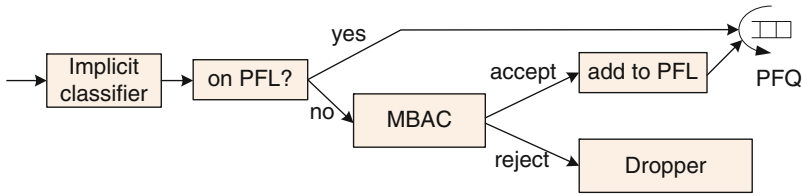


Fig. 2.1 Operation of FAN

negligible packet latency for flows emitting at lower rates. All the new functionalities are performed by a unique router, named the Cross-Protect router. This device alone is responsible for providing admission control and fair queuing.

Figure 2.1 illustrates the operation of FAN. All incoming packets are first classified into flows. The flow identification process is implicit and its goal is not to divide flows into different classes, but only to create an instance on which the service differentiation will be performed. Then, all the flows that are currently in progress are present on the Protected Flow List (PFL) and are forwarded unconditionally, whereas all new flows are subject to admission control. The admission control in FAN is measurement based (MBAC) which implies that the accept/reject decisions are based only on the current link congestion status. If a new flow is accepted, it is put onto the PFL list and then all successive packets from this flow are forwarded without checking the status of the outgoing link by MBAC.

In FAN, admission control and service differentiation are implicit. In classic explicit service differentiation architectures, user requirements are signaled to the network and the nodes perform differentiation actions based on the information received. For example, the provision of better quality is a consequence of the explicit identification of a certain transmission. On the contrary, implicit service differentiation performs differentiation actions based on traffic characteristics and network measurements.

In FAN, there is no need for a priori traffic specification, as well as no class of service distinction. Both streaming and elastic flows achieve the necessary QoS without any mutual detrimental effect. Nevertheless, streaming and elastic flows are implicitly identified inside the FAN network. This classification, however, is based solely on the current flow peak rate. All flows emitting at lower rates than the current fair rate are referred to as streaming flows, and the packets from those flows are prioritized. The remaining flows are referred to as elastic flows.

FAN is intended to be suited even to the whole Internet. This is due to a number of constraints that have been imposed by the designers. First of all, nodes do not need to exchange any information among themselves; they do not even need any explicit information about the flows. All information is gathered through local measurements. Flow-Aware Networking is based on the XP mechanism, which enhances the current IP router functionality.

One of the most important aspects of FAN is that it is only an enhancement of the currently existing IP network. Both networks can easily coexist. Moreover, the

advantages of FAN can be seen even if not all nodes are FAN based. This means that it is possible (and advisable) to improve the network gradually by replacing nodes, starting with those that are attached to the most heavily congested links.

2.3 Flow-Aware Approach

FAN is flow oriented. This means that traffic management is based on user-defined flows. The definition of a flow in Flow-Aware Networking comes from [8]: “By flow we mean a flight of datagrams, localized in time and space and having the same unique identifier.” The datagrams are localized in space as they are observed at a certain interface (e.g., on a router) and in time, as they must be spaced by no more than a certain interval, which is usually a few seconds. The space localization means that a typical flow has many instances, one at every interface on its path.

The identifier is obtained from certain IP header fields, including IP addresses and some other fields, e.g., the IPv6 *flow label*. One idea is to allow users to freely define flows that correspond to a particular instance of some application. The intention is to allow users as much flexibility as possible in defining what the network should consider as a single flow. Such an approach is surely beneficial for the user; however, it always introduces the possibility of malicious behavior. A flow label may also be deduced from IPv4 header fields. Typically, it could be a standard 5-tuple, though this approach limits the flexibility, allowing users no control in defining their flows.

All flows in FAN are divided into either streaming or elastic, and hence two classes of service. This distinction is implicit, which means that the system categorizes the flows based on their current behavior. There is no need for a priori traffic specification as the classification is based solely on the current flow peak rate. All flows emitting at lower rates than the current fair rate¹ are referred to as streaming flows, and packets from those flows are prioritized. The remaining flows are referred to as elastic flows. The association of a flow with a certain class is not permanent. If a flow, initially classified as streaming, surpasses the current fair rate value, it is degraded to the elastic flow category. Analogously, a flow is promoted to streaming if at any time it emits at a lower rate than the current fair rate. Note that both factors, i.e., flow bitrate and current fair rate, can change.

The assumption of FAN was to provide two classes of service. For low-rate flows which are typically associated with streaming transmissions, the streaming type is considered. All flows classified as streaming receive prioritized treatment—packets of those flows are sent through priority queues, hence, small delays, and delay variations. For the rest of the flows, proper fair queuing algorithms provide fair bandwidth sharing which cannot be assured with standard FIFO-based queuing disciplines. Finally, the distinctive advantage of FAN is that both streaming and elastic flows achieve sufficiently good QoS without any mutual detrimental effect.

¹Fair rate is one of the measured indicators of the link condition and is defined in Sect. 2.6.

2.4 Cross-Protect Mechanism

To install FAN in a network, an upgrade of current IP routers is required. Figure 2.2 shows the concept diagram for an XP router, the standard interconnecting device in FAN. FAN adds only two blocks to the standard IP router, namely the admission control and scheduling blocks. The former is placed in the incoming line cards of the router, whereas the latter is situated in the outgoing line cards.

Admission control is responsible for accepting or rejecting the incoming packets based on the current congestion status. The purpose of scheduling is twofold: it provides prioritized forwarding of streaming flows and assures fair sharing of the residual bandwidth by all elastic flows. If a packet is allowed, the flow associated with it may be added to the PFL, and then all forthcoming packets from this flow will be accepted. The admission control block realizes the measurement-based admission control functionality which is described in Sect. 2.5. The scheduler is responsible for queue management. It is a very important block as it has to ascertain that all flows are treated equally. All flows that currently have at least one packet in a queue are added to the Active Flow List (AFL). Detailed information on the scheduling algorithms is provided in Sect. 2.6.

Naming FAN devices as “Cross-Protect routers” is a consequence of the mutual cooperation and protection which exist between both extra blocks. The admission control block limits the number of active flows in a router, which essentially improves the queuing algorithm functionality and reduces its performance requirements. It is vital that queuing mechanisms operate quickly, as for extremely high-speed links the available processing time is strictly limited. On the other hand, the scheduling block

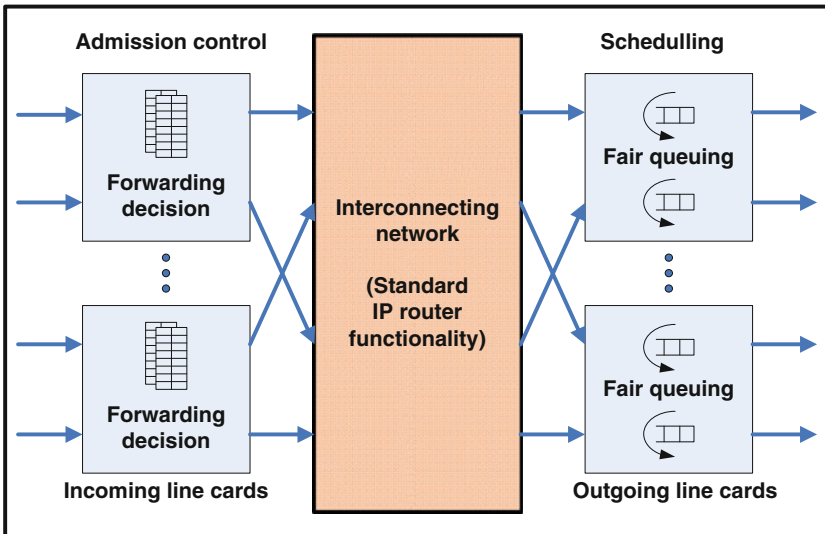


Fig. 2.2 Concept diagram of a Cross-Protect router

provides admission control with information on the congestion status on the outgoing interfaces. This information is derived based on, for example, current occupancy of the queues. This mutual protection contributes to a smaller protected flow list and active flow list sizes, which significantly improves FAN's scalability.

The advantage of XP routers is that they may be introduced progressively, starting from the most heavily loaded links. In this scenario, the overall network efficiency will gradually improve. However, obviously, for the best performance, all nodes in a network should be FAN-aware. Incremental replacement is possible because each XP router operates independently and transparently to other standard IP routers. There is no need for a signaling protocol, end user compliance, or any inter-network agreements. Moreover, in [8], the belief that in FAN there is "virtually no requirement for standardization" with an exception only for an "agreed convention for defining the flow identifier" is expressed. The lack of standardization is possible because of the local nature of the XP router functionality. As long as nodes perform well and maintain their functions, the exact method of their operation is insignificant. Lastly, once developed and implemented, the proposed mechanisms are thought to be particularly inexpensive.

2.5 Measurement-Based Admission Control

Admission control is a mechanism which allows blocking of some portion of traffic where congestion should occur. This ensures that the quality of currently realized transmissions will not deteriorate below a certain threshold. The benefits of using admission control in IP networks were presented in [10–12]. In FAN, admission control is used to keep the maximum flow rates at a reasonable level, while ensuring negligible latency for low-rate flows.

In FAN, the admission control block implements the measurement-based admission control (MBAC) functionality [13], and is designed to protect both streaming [14] and elastic flows [15]. Measurement based means that the admission decision relies solely on the measurements of outgoing link congestion. Therefore, MBAC is local to a particular network link. Since no signaling is used in FAN networks, MBAC must be performed with minimal knowledge about ongoing traffic. All of the above renders FAN admission control implicit, as it does not rely on any explicit user-network signaling. In [16], it is shown that MBAC in FAN is able to protect both streaming and elastic flows.

MBAC is not class based or user based: each new flow obtains the same treatment, and in case of congestion all new flows are blocked. Such an approach may be considered "unfair" service differentiation, since in congestion some flows are admitted and some are blocked. However, MBAC treats all flows equally, i.e., (a) the decision to accept or reject the traffic affects all new incoming flows, not just some of them, and (b) admission decisions are implicit, based only on internal measurements.

MBAC performs actions based on information derived from the scheduling algorithms. Two parameters are constantly measured, i.e., fair rate (FR) and priority load (PL). Following [6], “fair rate is an estimation of the rate currently realized by backlogged flows,” and represents the amount of link bandwidth guaranteed to be available for a single flow, should it be necessary. Similarly, “priority load is the sum of the lengths of priority packets incoming to the router in a certain interval divided by the duration of that interval,” and shows the amount of data that is prioritized. The manner of calculating both indicators is a feature of the scheduling algorithm, and is presented in Sect. 2.6 where fair queuing algorithms, suitable for FAN, are described.

Figure 2.3 illustrates the admission decision in MBAC. Each router has two pre-defined threshold values for FR and PL which are to be maintained, namely the minimum value of FR (minFR) and the maximum value of PL (maxPL). If the current FR is lower than minFR or if the current PL is greater than maxPL , the incoming flow is blocked. Otherwise, when in the admission region, the new flow is admitted.

The user-defined flows discussed earlier appear as the most appropriate entity on which the admission control should be performed. Admitted flows and those currently in progress are registered in PFL. If the flow identity of a newly arriving packet is already on the PFL, the packet is forwarded. If not, the flow is subject to admission control. If the outgoing link is congested, the packet is simply discarded. In the absence of congestion, the packet is forwarded, and its flow may be added to the PFL. This decision on whether to include the flow on the PFL or not is probabilistic. The flow is added with the probability p , e.g., $p = \frac{1}{10}$. This procedure aims to decrease the size of the PFL, as with high probabilities very short flows (a few packets) will not be added to the PFL. Flows with tens of packets and more will be added to the PFL eventually.

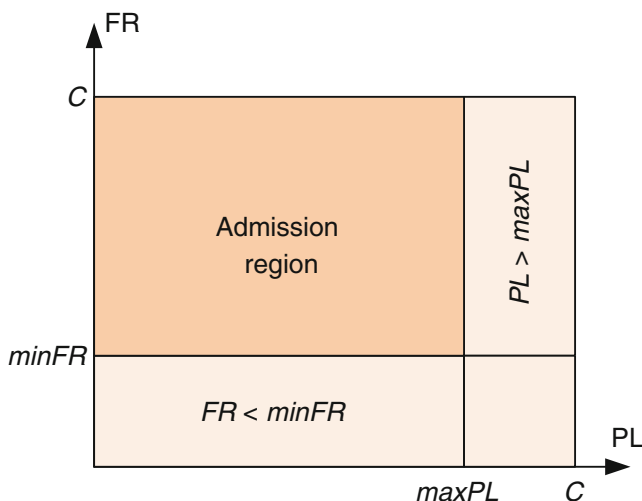


Fig. 2.3 Admission region in FAN

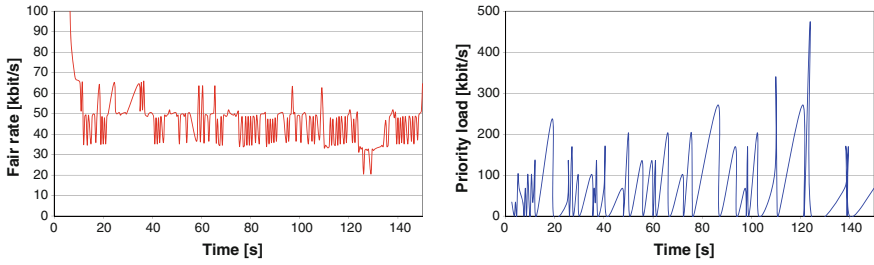


Fig. 2.4 FR and PL measurements; no exponential smoothing

In the simplest example, the admission criteria could rely only on PL and FR measurements. For instance, FR and PL thresholds could be set to 0.1 and 0.7, respectively. This means that packets of flows not registered in the PFL are discarded if the current FR is lower than 10 % of the link capacity or if the amount of priority traffic exceeds 70 % of the link capacity. However, the MBAC algorithm may also be based on some additional factors. In [14], such an algorithm is proposed. This algorithm also takes into account the measured aggregate load which should be less than a predefined threshold for new flows to be admitted. The ns-2 [17] simulations have shown that high link utilization can be achieved while maintaining a low packet delay and loss rate.

Figure 2.4 shows examples of FR and PL measurements in FAN-based routers, over time, using the Priority Fair Queuing scheduling algorithm. The experiment is performed over a 1 Mbit/s link with the offered load exceeding the link capacity by almost double. Strong variations in the values measured can easily be observed. These appear due to the extremely dynamic nature of packet-based transmission. In order to make the measurements more reliable, the notion of exponential smoothing was proposed in [14, 18]. The smoothing formula is presented in Eq. 2.1, in which α represents the smoothing parameter.

$$\text{new value} = \alpha \times \text{old value} + (1 - \alpha) \times \text{new measurement} \quad (2.1)$$

Figures 2.5 and 2.6 show the same measurements as in Fig. 2.4, but with the additional use of a smoothing parameter of 0.5 and 0.9, respectively. The measurements seem to be stabilized and more reliable, because the amplitude differences between

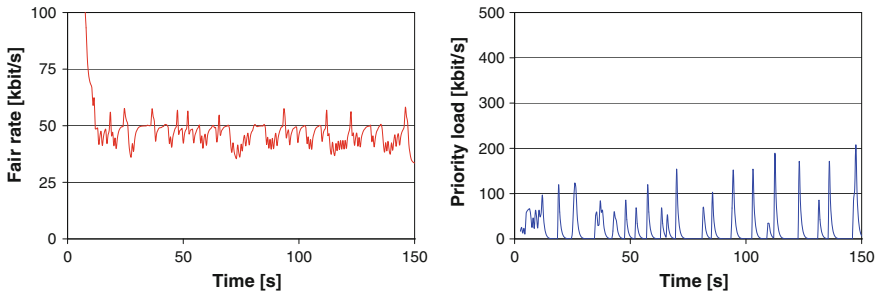


Fig. 2.5 FR and PL measurements; exponential smoothing $\alpha = 0.5$

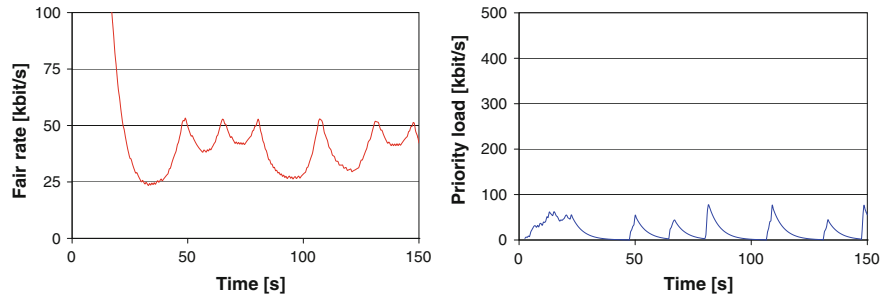


Fig. 2.6 FR and PL measurements; exponential smoothing $\alpha = 0.9$

consecutive measurements are significantly decreased. However, smoothing also has a drawback. The smoothed system reacts much more slowly to changes. This becomes extremely important when the currently measured values of fair rate drop below the threshold and the system should start to block incoming new flows. When the smoothing parameter is set to a high value, the system is not able to respond for quite a time, which leads to fair rate degradations.

2.6 Fair Queuing with Priority

Queue management in FAN is realized in the scheduling block of the XP router. Fair queuing (FQ) ensures that link bandwidth is shared equally, without relying on cooperative behavior by end users. This is a different approach to that currently used in IP routers, where, usually, a FIFO queue is implemented. The FIFO queuing discipline does not ensure fair sharing of the link bandwidth. Instead, all flows are limited to the same percentage of their nominal rates. Figure 2.7 shows the behavior of FIFO and FQ queues when two flows struggle for link resources.

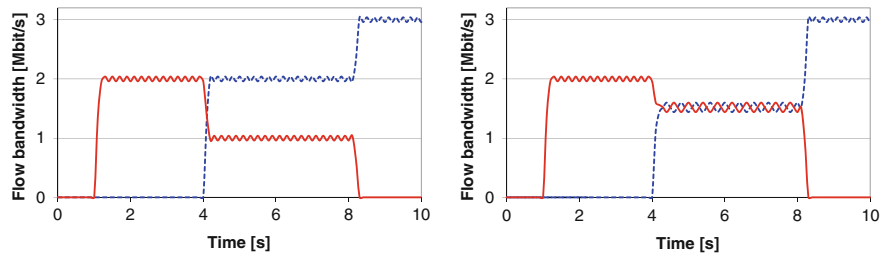


Fig. 2.7 FIFO (*left*) and FQ (*right*) scheduling comparison

The bottleneck link capacity is 3 Mbit/s, and two UDP flows² have nominal rates of 2 and 4 Mbit/s, respectively. Between the 4th and 8th second of the simulation time, both flows are in progress. The FIFO queue limits the rates of both flows to approximately 50 % of their desired rates, allowing the faster flow to utilize almost twice as much bandwidth as its competitor. The FQ discipline, which is represented in this example by Priority Fair Queuing, limits both flows to the same rate: 1.5 Mbit/s, which is exactly half of the total link capacity. It should be noted that the order in which flows appear on the link is irrelevant to the outcome of fair queuing.

There are two per-flow fair queuing algorithms proposed for FAN: Priority Fair Queuing (PFQ) and Priority Deficit Round-Robin (PDRR). Both algorithms have, logically, one priority queue and a secondary queuing system. They are intended to realize fair sharing of link bandwidth to elastic flows and priority service to streaming flows. The latter (PDRR) was primarily suggested to speed up commercial adoption since it improves the algorithm complexity from $O(\log(N))$ to $O(1)$, where N is the number of currently active flows. However, it has been shown that both scheduling algorithms achieve a similar performance [19].

Regardless of the algorithm used, it has been shown in [19,20] that fair queuing in FAN is scalable, since the complexity does not increase with link capacity. Moreover, fair queuing is feasible as long as link loads are not allowed to attain saturation levels, which is asserted by admission control. The latest FAN architecture has been proposed in [21], based on the Approximate Fair Dropping [22] queuing algorithm. This architecture is referred to as Approximate Flow-Aware Networking or AFAN for short, and aims at further simplifying the queuing processes. As shown in [21], the enqueue and dequeue operations in AFAN are realized in a simpler way than in the previous proposals for PFQ and PDRR.

2.6.1 Priority Fair Queuing

A large number of queuing algorithms have been proposed in the literature. The Start-time Fair Queuing (SFQ) [23] is particularly well suited to the FAN architecture. However, in [6], an enhancement of the SFQ algorithm is proposed. This modified queuing discipline is referred to as Priority Fair Queuing. PFQ differs from SFQ by the fact that it gives head of line priority to packets of flows whose rate is lower than the current fair rate. Therefore, PFQ implicitly prioritizes packets of low-rate flows, which are streaming flows in FAN.

The PFQ algorithm is based on the Push-In, First-Out (PIFO) queue. PIFO is the shorthand for a sorting algorithm that allows a packet to join the queue at any position and always serves the packet at the head of the line. The position that a packet is inserted into is determined by a time stamp, according to which packets are sorted within the queue. Therefore, every element in the PIFO queue has the form

²UDP flows were chosen as they are only shaped by the queuing algorithms, and not by the protocol behavior. This allows us to present solely the features of the queuing disciplines.

```

1  if PIFO congested, reject packet at head of longest backlog
2  if  $F \in \text{flow\_list}$ 
3    begin
4       $\text{backlog}(F) += L$ 
5      if  $\text{bytes} \geq \text{MTU}$ 
6        push {packet, flow\_time\_stamp} to PIFO
7      else begin
8        push {packet, virtual\_time} to PIFO behind P; update P
9        (counter of priority bytes  $+= L$ )
10        $\text{bytes}(F) += L$ 
11     end
12      $\text{flow\_time\_stamp}(F) += L$ 
13   end
14 else begin
15   push {packet, virtual\_time} to PIFO behind P; update P
16   (counter of priority bytes  $+= L$ )
17   if flow\_list is not saturated
18     begin
19       add flow F
20        $\text{flow\_time\_stamp}(F) = \text{virtual\_time} + L$ 
21        $\text{backlog}(F) = L$ 
22        $\text{bytes}(F) = L$ 
23     end
24   end

```

Fig. 2.8 PFQ packet arrival operations [6]

packet, *time stamp*, where *packet* represents the data relating to the packet (e.g., a memory location pointer) and *time stamp* is a packet “start tag” determined by the PFQ algorithm.

Figure 2.8 shows the pseudocode that is executed on each packet arrival, as proposed in [6]. First, it is necessary to test whether the queue is congested, and if so, which packet should be dropped. Dropping the packet at the head of the longest backlog is one proposal, although different criteria are possible. If a flow is active (line 2), its *backlog* is increased by the size of the packet (*L*). The packet is given priority when the cumulative volume of transmitted bytes is lower than the maximum transfer unit (*MTU*) (lines 7–11). Then, the packet is enqueued with the *time_stamp* of *virtual_time*, which is essentially the head of the queue. When the transmitted byte count is greater than *MTU*, the packet is placed in a PIFO queue, according to its nominal place. Lines 14–24 represent the situation in which the arriving flow is not active. Then, the packet is given a priority (line 15), and providing that the PFL is not saturated (line 17), the flow is added to the PFL (lines 18–23).

Figure 2.9 shows the operations performed after each packet departure. If the PIFO queue is empty, obviously all flows must be removed from the PFL (lines 1–2). Otherwise, the next packet in the queue is prepared (lines 4–6): the flow backlog is reduced and the *next_time_stamp* is set to this packet’s *time_stamp*. If *virtual_time* is equal to the *next_time_stamp* (line 7), no further operations are

```

1  if PIFO is now empty
2    remove all flows from flow_list
3  else begin
4     $\text{backlog}(F) - = L$ 
5    serve packet at head of line
6    next_time_stamp designates time stamp of this packet
7    if next_time_stamp  $\neq$  virtual_time
8      begin
9        virtual_time = next_time_stamp
10       for all flows  $f \in \text{flow\_list}$ 
11         begin
12           if  $\text{flow\_time\_stamp}(f) \leq \text{virtual\_time}$ 
13             remove  $f$  from flow_list
14         end
15       end
16     end

```

Fig. 2.9 PFQ packet departure operations [6]

required, as *virtual_time* has not changed since the last packet departure. If it has changed (lines 8–15), the *virtual time* is updated, and flows that become inactive (i.e., their *flow_time_stamp* is less than or equal to the new value of the *virtual_time*) are removed from the PFL.

To provide the admission control block with a proper congestion status, *PL* and *FR* indicators are measured periodically. An estimation of the priority load is derived from Eq. 2.2. Variables $pb(t)$ represent the values of a counter, incremented on the arrival of each priority packet by its length in bytes, at time t . (t_1, t_2) is a measured time interval (in seconds), and C is the link bit rate. The priority load, therefore, represents the sum of the lengths of priority packets incoming to the router in a certain time interval, divided by the duration of that interval, and normalized with respect to the link capacity.

$$PL = \frac{(pb(t_2) - pb(t_1)) \times 8}{C(t_2 - t_1)} \quad (2.2)$$

The smoothing parameter α is applied such that

$$PL(n) = \alpha \times PL(n - 1) + (1 - \alpha) \times \text{measured_PL}(n)$$

where $PL(n)$ is the value of PL in the n th iteration and the *measured_PL* is the value calculated from the formula (2.2) in the n th iteration.

Equation 2.3 is used to calculate the fair rate, which is an estimation of the rate currently realized by backlogged flows. To estimate the fair rate, a fictitious flow emitting single-byte packets is considered. In an idle period, the fictitious flow could transmit at the link rate. Otherwise, the number of bytes that could have been transmitted is given directly by the evolution of virtual time. In Eq. 2.3, $vt(t)$ is the value

of virtual time at time t , (t_1, t_2) is the measurement interval, S is the total idle time during the interval, and C is the link bit rate.

$$FR = \frac{\max\{S \times C, (vt(t_2) - vt(t_1)) \times 8\}}{t_2 - t_1} \quad (2.3)$$

The smoothing parameter β for calculating the values of FR is used in a similar way to the α parameter for PL.

When the measured link is lightly loaded, the first term of the $\max\{\dots\}$ formula is significant, as the fictitious flow uses all residual link capacity. When the link is busy, the second term becomes important, as it approximately measures the throughput achieved by any flow that is continuously backlogged in this time interval.

As mentioned, both congestion indicators are calculated periodically. Considering the extremely dynamic variations in priority packet occurrence, the period between two consecutive measurements of the priority load is advised to be several milliseconds, whereas a several-hundred-millisecond period is sufficient for estimating the fair rate. Regarding frequent measurements of the congestion indicators in PFQ, the complexity may be an issue, especially in the core. However, all the mathematical calculations in Eqs. 2.2 and 2.3 are very simple, and not time consuming. Moreover, the complexity of enqueueing and dequeueing operations of PFQ, like SFQ, is logarithmic with respect to the number of active flows, which is essentially limited by the admission control block.

2.6.2 Priority Deficit Round-Robin

PFQ was the first queuing algorithm proposed for FAN. In fact, simulations have shown that PFQ performs well and cooperates with the admission control block correctly [6]. Furthermore, the scalability of PFQ has been demonstrated by means of trace-driven simulations and analytical modeling in [19,20]. However, PFQ can be advantageously replaced by an adaptation of the Deficit Round-Robin (DRR) [24] algorithm. An enhancement to DRR, called Priority Deficit Round-Robin, is presented in [9].

PDRR retains the low complexity of DRR, while at the same time providing low latency for streaming flows. PDRR complexity is constant ($O(1)$); therefore, it does not increase with the growing number of active flows (PFQ complexity was logarithmic with respect to the number of active flows). PDRR enhances the DRR algorithm in which it introduces the priority queue, which is used for low-rate flows. Figure 2.10 shows the operations performed by PDRR on each packet arrival.

Initially, if the buffer for incoming packets is full, a certain packet must be selected for dropping (lines 1–3). PDRR does not specify which dropping mechanism should be used. One policy would be to drop packets at the head of the flow with the longest backlog; however, this approach is not mandatory. If packet P does not belong to an active flow, a new flow is added to PFL, the Deficit Count (DC) and Byte Count counters are properly initiated, and the packet is forwarded to the priority queue (PQ) (lines 5–11).

```

1  on arrival of packet P
2  if no free buffers left then
3      FreeBuffer()
4  i = ExtractFlow(P)
5  if (i ∉ PFL)
6      begin
7          add i to PFL
8          DC_i = 0
9          ByteCount_i = Size(P)
10         Enqueue(PQ, P)
11     end
12 else begin
13     ByteCount_i += Size(P)
14     if (ByteCount_i ≤ Q_i) then
15         Enqueue(PQ, P)
16     else
17         Enqueue(Queue_i, P)
18 end

```

Fig.2.10 PDRR packet arrival operations [9]

If an arriving packet belongs to a flow currently on the flow list, it may be placed at the end of his flow queue (line 17) or in the priority queue, providing that $ByteCount_i \leq Q_i$ (lines 14–15). The variable $ByteCount_i$ holds the number of bytes inserted in the queue for flow i , while Q_i represents flow quantum: the cumulative number of bytes allowed for transmission after every cycle of the algorithm. Although DRR allows for differentiation in resource allocation, by means of assigning different quanta Q_i for different flows, the FAN fairness concept implies that the same quanta should be used for each flow.

In PDRR, the FR is computed from the following formula:

$$FR = \frac{\max\{S \times C, fair_bytes \times 8\}}{t_2 - t_1} \quad (2.4)$$

where $fair_bytes$ is the number of bytes which could be sent by a fictitious permanently backlogged flow during the time interval (t_1, t_2) , S is the total length of inactivity in the transmission during the (t_1, t_2) period, and C is the link bit rate.

The PL is estimated in the same way as in the PFQ. Moreover, the smoothing parameters are used in a similar way to that in the PFQ.

Figure 2.11 shows the dequeue operations in the PDRR algorithm. The priority queue is served, whenever it is not empty (lines 3–9). When a packet is sent through the priority queue, the deficit counter of its flow is decreased by the size of the packet. This operation prevents serving more than one quantum in a single round. When there are no packets in the priority queue, and the active flow list (AFL) contains some flows, the flow at the current head of the AFL cycle is selected for service (line 12). The deficit counter for this flow is incremented by one quantum (line 13), and packets at the head of this flow's queue are prepared for being serviced (lines 14–24). The flow may emit up to DC_i bytes. At the end of the cycle, the AFL is rebuilt, i.e., completely erased (line 25) and re-created (lines 26–27).

```

1  while TRUE do
2  begin
3      while PQ not empty do
4      begin
5          P = Dequeue(PQ)
6          i = ExtractFlow(P)
7          Send(P)
8          DC_i = Size(P)
9      end
10     if AFL is not empty then
11     begin
12         get head of AFL, say flow i
13         DC_i += Q_i
14         while (DC_i ≥ 0) and (Queue_i not empty) do
15         begin
16             PacketSize = Size(Head(Queue_i))
17             if (PacketSize ≤ DC_i) then
18             begin
19                 Send(Dequeue(Queue_i))
20                 DC_i = DC_i - PacketSize
21             end
22             else
23                 break; (*skip while loop*)
24             end
25             RemoveActiveList(i)
26             if Queue_i is not empty then
27                 add i to AFL
28         end
29     end

```

Fig.2.11 PDRR packet departure operations [9]

The congestion indicators are measured differently than in PFQ. To measure the fair rate, we count the number of bytes that a fictitious and permanently backlogged flow could emit in a certain time interval and divide that value by the duration of the interval. This procedure is very easy to implement. The algorithm maintains one fictitious flow and treats it as a normal transmission, except that it does not transmit any packets. Therefore, the value of the deficit counter, which is regularly increased by the quantum, represents the theoretical amount of data that could be emitted by that flow. Since this flow is permanently backlogged, it does not disappear from the AFL, and thus its DC_i value is sustained. The priority load measurements are even simpler and are performed by averaging the emitted bit rate from a priority queue over a suitable time interval.

The introduction of a priority queue in PDRR results in situations in which flows with empty queues, i.e., flows whose packets are forwarded only via the priority queue, may exist. This implies that the dequeue procedure complexity is not strictly $O(1)$. However, according to [9], this can be corrected by modifying the AFL, to be a list for nonempty queues only. This list would be updated, whenever a new flow receives more than its quantum in the initial round.

The enqueueing module complexity depends on the speed of detecting the presence of a flow in the AFL (line 5 in Fig. 2.10). In order to maintain the $O(1)$ complexity, Content-Addressable Memory (CAM) must be used. CAM is a special kind of memory designed to search its entire contents in a single operation, but hardware implementation issues require that the size of AFL be small enough. However, the ns-2 based simulations have shown [9] that the required size of AFL is relatively small and, most importantly, does not increase with the link speed.

2.6.3 PFQ and PDRR Comparison

PDRR and PFQ are similar queuing algorithms. Although their operation is quite different, they realize the same objectives. The only advantage of PDRR over PFQ is simplicity. As mentioned before, the complexity of queuing disciplines in PFQ is logarithmic with respect to the number of active flows, whereas the same complexity is constant, and does not depend on the AFL size in PDRR.

Figures 2.12 and 2.13 show a comparison between the measured congestion indicators by PFQ and PDRR algorithms, namely fair rate and priority load, respectively. The ns-2 simulation scenario was identical for both scheduling disciplines. Two UDP

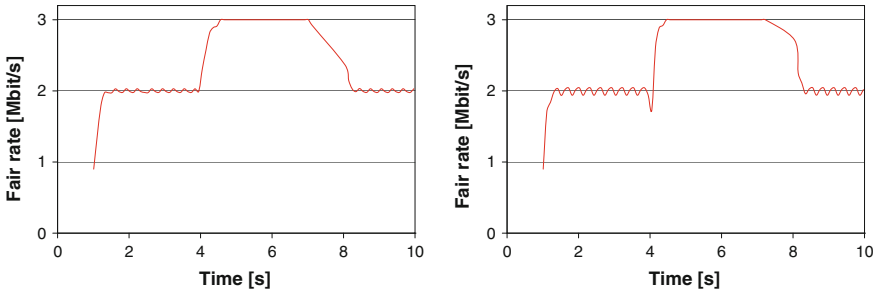


Fig. 2.12 Fair rate measurements; PFQ (on the *left*) and PDRR (on the *right*)

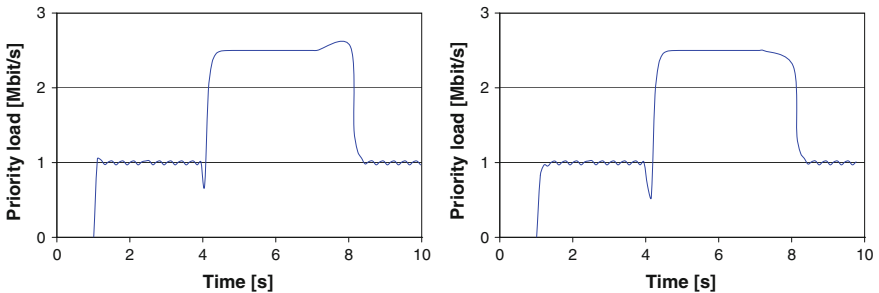


Fig. 2.13 Priority load measurements; PFQ (on the *left*) and PDRR (on the *right*)

flows of 1 and 2.5 Mbit/s nominal bit rates struggled to utilize the 3 Mbit/s bottleneck link. Between the 4th and 7th second, only the latter flow was active.

According to the fair rate definition (see Sect. 2.5), its value should be 100 % of the link capacity when only one flow is active, as that flow could emit at the link maximum bitrate, should it be necessary. When both flows are in progress, a 2 Mbit/s *FR* value is also expected. 1 Mbit/s flow is not backlogged, as it emits through the priority queue constantly (its rate is always below *FR*). Therefore, 2 Mbit/s is left for backlogged flows, but since only one more flow is in progress, 2 Mbit/s is the value of *FR*.

Priority load is the amount of data that is prioritized by the schedulers. When both flows are in progress, as mentioned previously, *FR* is equal to 2 Mbit/s. Therefore, a 1 Mbit/s flow is below *FR*, and its packets constantly go through the priority queue, whereas a 2.5 Mbit/s flow is above *FR* and always utilizes normal queuing. However, when only a 2.5 Mbit/s flow is in service, *FR* is equal to 100 % of link capacity and the flow's packets use the priority queue, as the flow's rate is below the current *FR*. Therefore, the measured *PL* values are 1 Mbit/s in the former case, and 2.5 Mbit/s in the latter.

As expected, no major differences between the measured values by both scheduling disciplines exist. This confirms the previous statement that the difference between PFQ and PDRR lies only in complexity issues. Although minor differences can be observed, they are insignificant to the overall behavior of admission control for which these indicators are used.

Example 2.1 We have 100 Mbit/s FAN link with the PFQ scheduling algorithm. Two 50 Mbit/s access links are connected to it. In each access link, a TCP flow transmits its data. What will the *FR* value be in a steady state?

Solution

The bandwidth in the FAN link will be divided into a fair manner for those TCP flows. Therefore, the *FR* value will be 50 Mbit/s.

Example 2.2 We have 100 Mbit/s FAN link with the PFQ scheduling algorithm. Two 50 Mbit/s access links are connected to it. In each access link, a UDP flow transmits its data with a rate of 10 Mbit/s. What will be the *FR* value in a steady state?

Solution

The flows will consume only 20 Mbit/s. Therefore, the *FR* will be estimated based on the first part of Eq. 2.3. While data will not be transmitted in the FAN link during 80 % of each estimation interval of the *FR*, its value will be equal to 80 Mbit/s.

2.6.4 Approximate Flow-Aware Networking

The queuing and dequeuing operations in two well-known FAN architectures, with PFQ and PDRR scheduling algorithms, are quite complex. It is necessary to analyze the virtual tags of each flow in PFQ. The position of the pointer provided for separating the packets in streaming and elastic flows in the PIFO queue has to be analyzed each time the packet is selected for transmission. In PDRR, it is necessary to implement many FIFO queues, individual for each elastic flow, and one for packets of streaming flows. In both versions, the AFL is used for selecting packets to be sent.

Packet transmission is usually more effective if the network architecture is as simple as possible. In this section, the third FAN architecture, known as Approximate Flow-Aware Networking (AFAN), introduced in [21], is presented. This solution ensures fair transmission of packets which belong to elastic flows through one FIFO queue and prioritizing possibilities using a separate FIFO queue for streaming flows. The pseudocode for enqueueing the packets in AFAN is presented in Fig. 2.14. There is no need to implement AFL or additional objects.

Admission Control Operation in AFAN

Each packet P incoming to the FAN router has to be analyzed in the admission control block. If its flow ID is written to the PFL, the packet is selected for queuing (lines 3–4). On the other hand, if packet P represents a new flow, it must be dropped in overload (lines 5–6). In a congestionless state, it may be accepted and its flow ID is written to the PFL with probability PI (line 7). This probability may be low. As a result, the IDs of short flows will not be added to the PFL. The congestion state is noticed based on the values of the FR and the PL parameters estimated in the scheduler block. The admission control mechanism works exactly the same as in the FAN versions with PFQ and PDRR algorithms. However, different methods for estimating the values of the parameters mentioned above are used.

To estimate the PL in AFAN, a counter incremented on the departure of each priority packet by its length in bytes has to be maintained. Let $pb(t)$ be the value of this counter at time t , (t_1, t_2) is the measurement interval (in seconds), and C is the link bit rate. An estimate of the PL is

$$PL = \frac{(pb(t_2) - pb(t_1)) \times 8}{C(t_2 - t_1)} \quad (2.5)$$

As a result, we estimate the value of the PL based on the transmitted packets. This method is more precise than the one used in PFQ or PDRR (where incoming packets are encountered and which later may be dropped).

The FR is computed by the following formula:

$$FR = \frac{\max\{S \times C, FB \times 8\}}{t_2 - t_1} \quad (2.6)$$

where FB is the number of bytes sent by elastic flows during the time interval (t_1, t_2) divided by the number of elastic flows in the PFL, and S is the total length of inactivity in the transmission during the (t_1, t_2) period, C is the link bit rate.

```

1  ##### Admission control module: #####
2  on arrival of packet P
3  if flow ID  $\in$  PFL then
4      accept P for queuing
5  else if congestion is noticed then
6      drop P
7      else accept P with probability P1
8  ##### Enqueuing module: #####
9  on packet P accepted in MBAC
10 if flow_bytes  $\leq$  Q then
11     begin
12         Enqueue(PQ, P)
13         flow_bytes = flow_bytes + size(P)
14     end
15 else compute ABS of elastic FIFO queue
16 if ABS  $\geq$  max_th then
17     begin
18         drop P
19         count = 0
20     end
21 if min_th  $\leq$  ABS < max_th then
22     begin
23         count = count + 1
24         draw packet D from the elastic FIFO queue
25         if flow ID(P) = ID(D) then
26             begin
27                 drop D
28                 calculate probability P2
29                 drop P with probability P2
30                 if P is dropped then
31                     count = 0
32                 else P is selected for queuing
33             end
34         else P is selected for queuing
35     end
36 if ABS < min_th then
37     begin
38         P is selected for queuing
39         count = -1
40     end
41 if flow_bytes > Q and P is selected for queuing then
42     begin
43         Enqueue(EQ, P)
44         flow_bytes = flow_bytes + size(P)
45     end

```

Fig. 2.14 AFAN packet arrival operations [21]

In this method, we do not need to use any fictitious flow (which is used in PFQ or PDRR). As a result, this method is less complex.

Enqueue Operation

If the accepted packet belongs to a flow, which has less than or equal to Q (flow quantum usually set to the value of MTU) bytes in the buffer, it is enqueued to the priority queue (lines 9–13). In the other case, the value of the *ABS* (*Approximate Buffer Size*) parameter is estimated (line 15) from the following formula:

$$\begin{cases} ABS = (1 - w_q)ABS + w_q q & \text{if the queue is nonempty} \\ ABS = (1 - w_q)^m ABS & \text{if the queue is empty} \end{cases} \quad (2.7)$$

where w_q is queue weight, q is the current buffer size, and m represents the number of packets that could be sent when the line was free [25] and is computed from the following formula:

$$m = (time - q_time)/s \quad (2.8)$$

where $time$ is the current time, q_time is the start time of the buffer idle time, and s is the transmission time of the packet.

If the *ABS* is greater than or equal to the maximum threshold set for the buffer (max_th), the incoming packet must be dropped and the counter *count*, which means the number of incoming packets since the last dropping operation, is set to zero (lines 16–19). If the *ABS* is greater than or equal to the minimum threshold set for the buffer (min_th) and lower than the max_th , the *count* parameter is incremented by one and the flow ID of randomly selected packet D from the elastic FIFO queue is compared with the flow ID of incoming packet P (lines 21–24). In practice, we do not draw a packet but a byte from the buffer and then search the packet it belongs to. This allows for proper transmission of flows with variable-length packets. If both packets represent the same flow, packet D is dropped and packet P is dropped with probability $P2$ (lines 25–29). If P is dropped, *count* is set to zero, if not P is selected for queuing (lines 30–32). Packet P may also be queued if its flow ID is different from the flow ID of packet D , or *ABS* is lower than min_th (lines 36–38). In the latter case, the value of the *count* parameter is set to minus one (the initial value of the algorithm) (line 39). If the number of bytes in the flow the packet P belongs to is greater than Q , the packet is queued in the elastic queue (lines 41–44).

Dequeue Operation

The process of selecting a packet for sending is very simple. The pseudocode for such an operation is presented in Fig. 2.15. If there are any packets in the priority queue, the first of them is selected to be served (lines 2–6). If the PQ is empty, the first packet from the elastic queue is sent (lines 8–12).

Complexity

Admission control complexity is the same as in the FAN implementations with PFQ and PDRR. The enqueue operation is less complex in AFAN. We need to compute the *ABS* for the elastic flows, but in fact this is a fraction of the current buffer size

```

1  ##### Dequeuing module: #####
2  while PQ is not empty do
3  begin
4      P = Dequeue(PQ)
5      Send(P)
6      flow_bytes = flow_bytes - size(P)
7  end
8  if elastic FIFO queue is not empty then
9  begin
10     P = Dequeue(EQ)
11     Send(P)
12     flow_bytes = flow_bytes - size(P)
13  end

```

Fig. 2.15 AFAN packet departure operations [21]

estimated in each FAN version. There is also a need for random selection of packet D from the elastic queue to compare it with the incoming elastic packet P . In the well-known versions of FAN, it is necessary to check the buffer load and whether it is full (in almost every case) to find the longest flow and drop its last packet. The complexity of queuing the elastic packets is the same in all FAN versions. The advantage of AFAN is visible when we compare the enqueueing process for streaming flows. In the versions of FAN with PFQ and PDRR, these are served in the same way as elastic flows. In AFAN, the packets of streaming flows are queued immediately without any additional operation. This is possible because in this solution it is impossible to overload the buffer. The real gain in algorithm complexity may be seen in AFAN while comparing the dequeuing operations for each FAN version. In this proposal, we only have to check whether there are any packets in the priority queue and if so serve them as first. There is no need to maintain an additional structure like the AFL implemented in both known versions of FAN. We do not need to find a flow before sending a packet and update the content of AFL. All the arguments presented in this section confirm that AFAN is less complex than its predecessors. We can assume that it is also easy to implement in routers.

Calculation of the AFAN Parameters

New parameters, which have to be set in routers, are provided in AFAN. In this proposal, a low-pass filter to calculate the ABS is used (see formula (2.7)). This method is taken from the RED (*Random Early Detection*) queuing algorithm [25] and ensures insignificant changes in the ABS in spite of bursty incoming traffic. The weight w_q represents the time constant of the low-pass filter. The proper value of this parameter affects the effectiveness of the whole algorithm. If it is too large, the transient congestion may not be noticed in the router. On the other hand, if w_q is too low the mechanism may react to changes in the queue too slowly. As a result, the router may be unable to detect some symptoms of congestion. Based on the analysis presented in [25], we can assume that the value of the w_q parameter should be set in the range from 0.001 to 0.0042. In the simulation experiments presented in the

following section, we used the value of 0.002, which ensured the achievement of the assumed goals.

min_th and max_th are the key parameters for AFAN. Their optimal values affect buffer occupation and allow for fair and efficient transmission in the links. If the traffic is fairly bursty, the value of min_th should be set to a value large enough to ensure link utilization at an acceptable level. On the other hand, it ought not to be too large because of packet delay demands. Following [25], the difference $max_th - min_th$ should be larger than the typical increase in the estimated ABS in one roundtrip time. In practice, max_th should be set to at least twice min_th . We set the min_th parameter to 40 kB and max_th to 90 kB in the experiments. The buffer size was set to 100 kB.

Two methods for computing $P2$ are proposed in [25]. Method 2 called “uniform random variables” is described as the better one. $P2$ is calculated from the following formula:

$$P2 = P2_{temp} / (1 - count \cdot P2) \quad (2.9)$$

The $P2_{temp}$ is calculated as the function of ABS and estimated from the following:

$$P2_{temp} = max_p(ABS - min_th) / (max_th - min_th) \quad (2.10)$$

where max_p is the maximum allowed value of $P2_{temp}$. It is important to set the proper value for the max_p parameter to estimate $P2$ efficiently.

The expected value for this method is estimated from

$$E[X] = 1 / (2P2) + 1/2 \quad (2.11)$$

This means that if we set max_p to 0.02 (as we did in the simulation experiments), and the ABS is halfway between min_th and max_th , an average of one out of fifty arriving packets is dropped in the router. The value of max_p should be set to values which allow for slow changes in the $P2$ probability. This is needed for minimizing the fluctuations in the ABS .

Simulation Experiments of AFAN

In this section, we present the results of simulation experiments in the ns-2 simulator [17]. They show that AFAN is very simple in implementation and works similar to the FAN versions with PFQ and PDRR. We evaluated AFAN performance by analyzing the traffic parameters during transmission in an overloaded AFAN link.

The experiments were provided for a single FAN link with many source and destination nodes. The topology is presented in Fig. 2.16. It is simple, yet adequate to analyze traffic performance in Flow-Aware Networks. The reason behind this is that all nodes in FAN operate independently and all the decisions are taken without any information from the network.

The nodes $S_{E1}-S_{En}$ are the sources of the elastic traffic, while the nodes $S_{S1}-S_{Sm}$ are the sources of the streaming traffic. The nodes $D_{E1}-D_{En}$ and $D_{S1}-D_{Sm}$ represent the destination nodes for the elastic and streaming traffic, respectively.

To provide credible simulations, it is important to set proper values for the simulation parameters. The capacity of the FAN link was set to 100Mbit/s. Of course, this value is too low when considering core links; however, the obtained results are

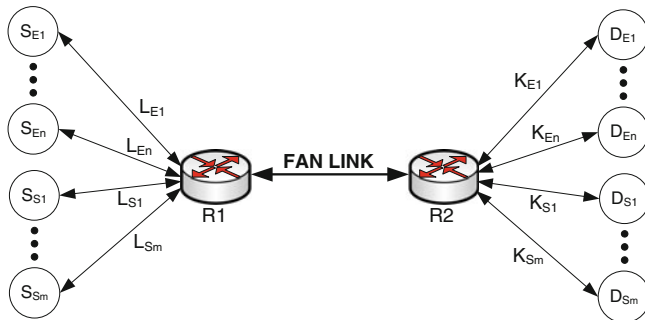


Fig. 2.16 The basic simulation topology

scalable. The only reason to make simulations for a low capacity core link is the time needed to conduct the experiments. In our conditions, one simulation run took about an hour. The capacity of other links was set to 1 Gbit/s.

Based on the simulation parameters presented in [26], the buffer size for a 50 Mbit/s link should be set to a value of 100 packets. The buffer is sized to absorb approximately one delay bandwidth product. For a 100 Mbit/s link (used in the simulation runs for this book), a reasonable value is 1000 packets. This allows for low packet delays and jitter.

All observations of simulation parameters should be provided in a steady state. The setting of an appropriate value for the warm-up period is a key element for the credibility of simulation results. There are many methods for estimating the value of this parameter [27]. One of these is based on the assumption that the values of the observed variable stop moving in one direction and start oscillating. In another, the steady state is noticed at the point at which the value is neither the minimum nor the maximum of the remaining observations. The warm-up for all simulations experiments described in this section, based on several observations of the results obtained, was assumed to be 20 s.

The other simulation parameters were set as follows. The traffic pattern was provided with Pareto distribution for calculating the volume of traffic to be sent by each of the elastic flows in the FAN link ($k = 1.5$, mean size of elastic flow was set to 150 Mbit). We used exponential distribution for generating the time intervals between beginnings of the transmissions of the elastic flows (mean inter-arrival time was set to 0.1 s) and 20 streaming flows (mean inter-arrival time was set to 1 s). We made our simulation runs under various conditions changing the number of elastic flows from 200 to 600. The duration of each simulation run was set to 1200 s. The packet size for the elastic flows was set to 1000 bytes, while that for the streaming flows was set to 100 bytes. The transmission rate of streaming flows was set to 80 kbit/s (as in a typical Skype VoIP connection). The measurement interval for the *PL* parameter was set to 50 ms, while the *FR* values were estimated every 0.5 s. The *maxPL* parameter was set to 70 %, the *minFR* parameter was set to 5 % and the *flow_time_out* parameter was set to 20 s, which means that the ID of an inactive flow

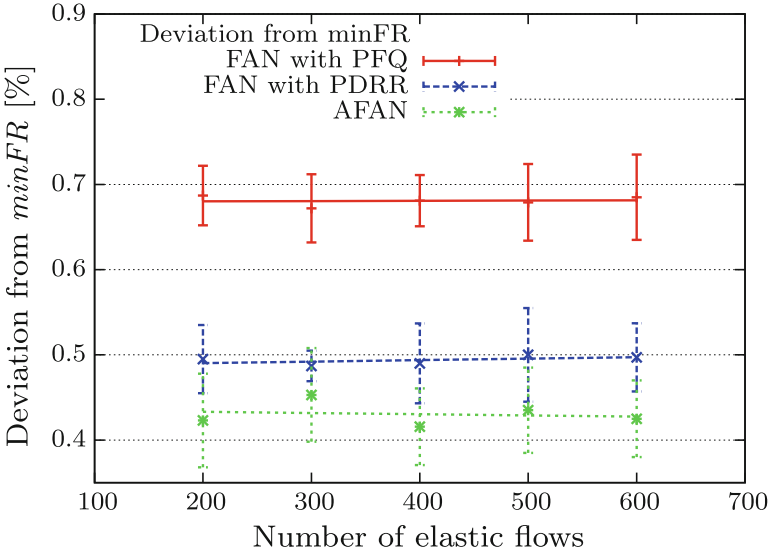


Fig. 2.17 The mean deviation from the $minFR$

was removed from PFL after 20 s. Each experiment was repeated at least 10 times. 95 % confidence intervals were calculated using the Student's t-distribution.

For AFAN, min_th was set to 4000 packets, max_th to 9000 packets, and the w_q parameter was set to 0.02.

We made 150 simulation runs (50 for each FAN architecture) under various conditions to show the mean deviation from the $minFR$ value and the mean number of flows accepted in the PFL.

The mean values of deviation from the $minFR$ for each FAN architecture are estimated as part of the link capacity and presented in Fig. 2.17. The values of the FR change in time and oscillate around the $minFR$. The situation is better if the deviations from the $minFR$ are lower. We can see that for AFAN and FAN with PDRR, the analyzed mean deviation is lower than for FAN with PFQ. This means that the former two versions are more stable than the latter one.

We analyzed the mean flow count in the PFL in the same simulation scenarios. Based on the results shown in Fig. 2.18, we may conclude that the number of active flows in AFAN is slightly lower than in the other FAN architectures. The difference is not great. We can see that all three versions work similarly, giving an equal possibility for each flow accepted in the admission control block to transmit its traffic with a rate more or less equal to the FR . However, we should remember that the AFAN architecture is less complex than FAN with PFQ or PDRR.

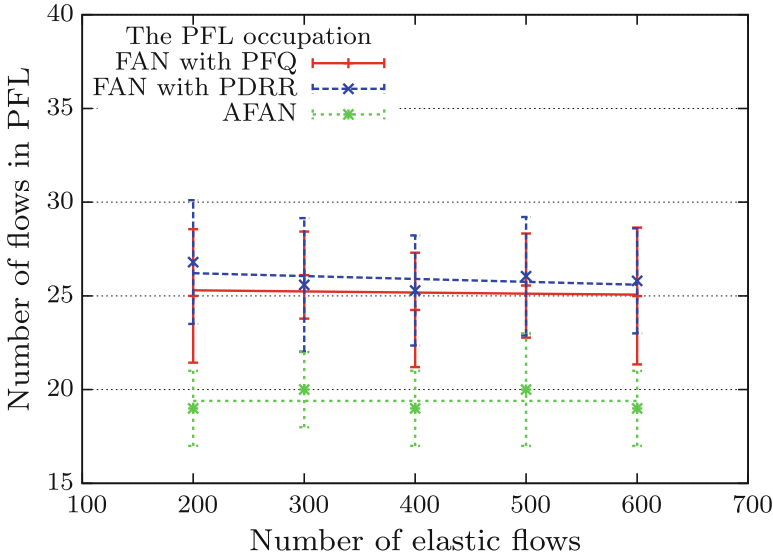


Fig. 2.18 The PFL occupation in various FAN architectures

2.7 Additional FAN Architectures and Mechanisms

The FAN architecture has attracted some worldwide attention which has resulted in many studies. Over the years, numerous additional mechanisms have been proposed for FAN. These were in response to certain technical problems with the implementation or performance. New mechanisms evaluate the possibilities of using FAN in certain scenarios or simply improve its functioning.

When the link is congested, XP routers block new connections which increases their acceptance delay. There are two approaches solving this problem. One is to use the scheme of Static Router Configuration to help with the transmission of emergency calls. This method was presented in [28] and is described in Chap. 7. The second approach is based on periodic partial or total clearing of the PFL list in an XP router. Various modifications of this method were presented in [29–34]. The mechanisms are described in detail in Chap. 4.

The notion of Multilayer Flow-Aware Networking (MFAN) was introduced in [35], and later presented as a complete approach in the PhD dissertation of V. López in 2010 [36]. In this study, it is shown that FAN can be extended by including an optical layer to be considered by the system. The idea is that a FAN router can request additional optical resources once the standard IP link is congested. Under normal circumstances, upon the congestion of the outgoing link, FAN starts to block new incoming connections. In MFAN, the router is able to utilize additional resources and redirect flows to that resource, creating space for new flows to be admitted. There are three admission control policies deciding which flows are to be redirected to the

optical link, i.e., Newest Flow Policy, Oldest Flow Policy, and Most Active Flow Policy. In [35], those policies are compared and their performance is evaluated.

In [37], the authors compare admission control policies proposed for MFAN. As a result of the comparison, a new admission control strategy is proposed. The solution inherits the advantages of the already established admission control proposals while ensuring fast acceptance times for new streaming flows. It is also possible to combine the advantages of MFAN with those of flushing mechanisms. This work was continued under the BONE EU project, where the authors showed the differences between the admission control strategies proposed for IP-level FAN and MFAN.

In [38], a multilayer recovery strategy for the MFAN nodes is presented. The authors propose using the Enhanced Hold-Off Timer (EHOT) algorithm [39], known from RPR networks, to control network operation after link or node failure. Network performance after failures is also presented in [40] where the authors measure the impact of proposed congestion control mechanisms in the case of network overload. The results show that the acceptance times for streaming flows are acceptable even with the presence of network failures, provided that proper congestion control mechanisms are used. Both papers essentially show that FAN networks have great resilience.

FAN was also tested in the Grid environment. In [41], the authors show the impact of DiffServ and FAN on the Grid traffic, and compare the efficiency of those architectures in providing QoS assurances. Further, in [42–44], the performance of FAN in the Grid environment is evaluated. It is shown that FAN outperforms DiffServ in the average GridFTP session delay and the average GridFTP session goodput under increasing offered load.

FAN does not interfere with IP protocol functionality, including routing procedures. However, it is possible to introduce a new routing scheme—one which would cooperate with FAN. In [45], such a scheme is proposed. Adaptive routing clearly improves network performance especially in overload and failure conditions.

2.8 Check Your Knowledge

1. We have a 100 Mbit/s FAN link with the PFQ scheduling algorithm. The $\min FR$ is equal to 5 Mbit/s. The measurement interval for the FR parameter was set to 0.5 s. In 2.5 s FR was equal to 4.93 Mbit/s, while in 3.0 s it was equal to 4.98 Mbit/s. At 3.35 s the video flow wants to begin transmission with rate 1 Mbit/s. Will this flow be accepted immediately?
2. Are PFQ and PDRR the only scheduling algorithm which may be used in FAN?
3. Are packets in FAN marked?
4. In which block of an XP router are the values of FR and PL estimated?
5. The TCP flow wanted to begin transmission in 5 s, but the link was congested. The congestion was eliminated in 10 s. Was the TCP flow accepted immediately?
6. On a 3 Mbit/s FAN link, there are 2 UDP flows: 1 and 2 Mbit/s. What are the values of Fair Rate and Priority Load?

7. On a 5 Mbit/s FAN link, there are 2 UDP flows: 1 and 2 Mbit/s. What are the values of Fair Rate and Priority Load?
8. On a 3 Mbit/s FAN link, there are 3 UDP flows: two 1 Mbit/s and one 2 Mbit/s. What are the values of Fair Rate and Priority Load? What is the rate of each flow?
9. When are new flows admitted on a link?
10. What does it mean when a flow is on a PFL list?
11. What are the differences between the queuing algorithms designed for FAN?
12. How are flows identified in FAN?
13. What are the names of classes of flows in FAN?
14. How do FAN nodes know which class a flow belongs to?
15. Can flows belong to two classes at the same time?
16. Can flows change their class over time?
17. What is the difference in treating flows of different classes?
18. Can a UDP flow be an elastic flow?
19. Can a TCP flow be a streaming flow?
20. Can current FR drop below the *minFR*?
21. Can current PL exceed the *maxPL*?
22. When are flows removed from the PFL list?

References

1. R. Braden, D. Clark, and S. Shenker, "Integrated Services in the Internet Architecture an Overview," IETF RFC 1633, June 1994.
2. S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An Architecture for Differentiated Services," IETF RFC 2475, December 1998.
3. K. Nichols, S. Blake, F. Baker, and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers," IETF RFC 2474, December 1998.
4. T. Bonald, S. Oueslati-Boulahia, and J. Roberts, "IP traffic and QoS control," in *Proc. World Telecommunications Congress, WTC 2002*, Paris, France, September 2002.
5. J. Roberts and S. Oueslati, "Quality of Service by Flow Aware Networking," *Philosophical Transactions of The Royal Society of London*, vol. 358, pp. 2197–2207, 2000.
6. A. Kortebe, S. Oueslati, and J. W. Roberts, "Cross-protect: implicit service differentiation and admission control," in *Proc. High Performance Switching and Routing, HPSR 2004*, Phoenix, AZ, USA, 2004, pp. 56–60.
7. J. Roberts, "Internet Traffic, QoS and Pricing," in *Proc. the IEEE*, vol. 92, September 2004, pp. 1389–1399.
8. S. Oueslati and J. Roberts, "A new direction for quality of service: Flow-aware networking," in *Proc. 1st Conference on Next Generation Internet Networks - Traffic Engineering, NGI 2005*, Rome, Italy, 2005.
9. A. Kortebe, S. Oueslati, and J. Roberts, "Implicit Service Differentiation using Deficit Round Robin," in *Proc. 19th International Teletraffic Congress, ITC 2005*, Beijing, China, August/September 2005.

10. N. Benameur, S. B. Fredj, F. Delcoigne, S. Oueslati-Boulahia, and J. Roberts, "Integrated Admission Control for Streaming and Elastic Traffic," in *Proc. Second International Workshop on Quality of Future Internet Services, QoFIS 2001*, Coimbra, Portugal, September 2001.
11. L. Massoulie and J. Roberts, "Arguments in Favour of Admission Control for TCP Flows," in *Proc. 11th International Teletraffic Congress, ITC 1999*, Edinburgh, June 1999.
12. J. W. Roberts and L. Massoulie, "Bandwidth Sharing and Admission Control for Elastic Traffic," in *Proc. ITC Specialist Seminar*, Yokohama, October 1998.
13. Y. Jiang, P. Emstad, A. Nevin, V. Nicola, and M. Fidler, "Measurement-Based Admission Control for a Flow-Aware Network," in *Proc. 1st Conference on Next Generation Internet Networks - Traffic Engineering, NGI 2005*, Rome, Italy, April 2005, pp. 318–325.
14. A. Kortebi, S. Oueslati, and J. Roberts, "MBAC algorithms for streaming flows in Cross-protect," in *Proc. Next Generation Internet Networks EuroNGI Workshop*, Lund, Sweden, June 2004.
15. S. B. Fredj, S. Oueslati-Boulahia, and J. Roberts, "Measurement-based Admission Control for Elastic Traffic," in *Proc. 17th International Teletraffic Congress, ITC 2001*, Salvador, Brasil, December 2001.
16. N. Benameur, S. B. Fredj, S. Oueslati-Boulahia, and J. Roberts, "Quality of Service and flow level admission control in the Internet," *Computer Networks*, vol. 40, pp. 57–71, August 2002.
17. "Network Simulator ns-2," Available at <http://nsnam.isi.edu/nsnam>
18. A. Kortebi, L. Muscariello, S. Oueslati, and J. Roberts, "Minimizing the Overhead in Implementing Flow-aware Networking," in *Proceedings of Symposium on Architectures for Networking and Communications Systems, ANCS 2005*, Princeton, USA, October 2005.
19. A. Kortebi, L. Muscariello, S. Oueslati, and J. Roberts, "Evaluating the number of Active Flows in a Scheduler Realizing Fair Statistical Bandwidth Sharing," in *Proc. International Conference on Measurement and Modeling of Computer Systems, ACM SIGMETRICS 2005*, Banff, Canada, June 2005.
20. A. Kortebi, L. Muscariello, S. Oueslati, and J. Roberts, "On the scalability of fair queueing," in *Proc. Third Workshop on Hot Topics in Networks, ACM HotNets-III 2004*, San Diego, USA, November 2004.
21. J. Domzal and A. Jajszczyk, "Approximate Flow-Aware Networking," in *Proc. IEEE International Conference on Communications ICC 2009*, Dresden, Germany, June 2009.
22. K. Psounis, R. Pan, and B. Prabhakar, "Approximate Fair Dropping for Variable-Length Packets," *Micro, IEEE*, vol. 21, pp. 48–56, January 2001.
23. P. Goyal, H. M. Vin, and H. Cheng, "Start-time Fair Queuing: A Scheduling Algorithm for Integrated Services Packet Switching Networks," *IEEE/ACM Transactions on Networking*, vol. 5, pp. 690–704, October 1997.
24. M. Shreedhar and G. Varghese, "Efficient Fair Queuing Using Deficit Round-Robin," *IEEE/ACM Transactions on Networking*, vol. 4, pp. 375–385, June 1996.
25. S. Floyd and V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance," *IEEE/ACM Transactions on Networking*, vol. 1, pp. 397–413, August 1993.
26. J. Auge and J. Roberts, "Buffer sizing for elastic traffic," in *Proceedings of 2nd Conference on Next Generation Internet Design and Engineering, NGI 2006*, Valencia, Spain, April 2006.
27. K. Pawlikowski, "Steady-state Simulation of Queueing Processes: A Survey of Problems and Solutions," *ACM Computing Surveys*, vol. 22, no. 2, pp. 123–170, June 1990.
28. A. Jajszczyk and R. Wojcik, "Emergency calls in flow-aware networks," *Communications Letters, IEEE*, vol. 11, no. 9, pp. 753–755, September 2007.
29. J. Domzal and A. Jajszczyk, "New Congestion Control Mechanisms for Flow-Aware Networks," in *Proc. IEEE International Conference on Communications ICC 2008*, Beijing, China, May 2008.
30. J. Domzal and A. Jajszczyk, "The Flushing Mechanism for MBAC in Flow-Aware Networks," in *Proc. 4th EURO-NGI Conference on Next Generation Internet Networks, NGI 2008*, Krakow, Poland, April 2008, pp. 77–83.

31. J. Domzal and A. Jajszczyk, "The Impact of Congestion Control Mechanisms for Flow-Aware Networks on Traffic Assignment in Two Router Architectures," in *Proc. International Conference on the Latest Advances in Networks, ICLAN 2008*, Toulouse, France, December 2008.
32. J. Domzal, R. Wojcik, and A. Jajszczyk, "Reliable Transmission in Flow-Aware Networks," in *Proc. IEEE Global Communications Conference GLOBECOM 2009*, Honolulu, USA, December 2009, pp. 1–6.
33. J. Domzał, R. Wojcik, V. López, J. Aracil, and A. Jajszczyk, "EFMP – a new congestion control mechanism for flow-aware networks," *Transactions on Emerging Telecommunications Technologies*, vol. 25, no. 11, pp. 1137–1148, 2014.
34. J. Domzal, N. Ansari, and A. Jajszczyk, "Congestion Control in Wireless Flow-Aware Networks," in *Proc. IEEE International Conference on Communications ICC 2011*, Kyoto, Japan, June 2011.
35. V. Lopez, C. Cardenas, J. A. Hernandez, J. Aracil, and M. Gagnaire, "Extension of the Flow-Aware Networking (FAN) architecture to the IP over WDM environment," in *Proc. 4th International Telecommunication Networking Workshop on QoS in Multiservice IP Networks*, Venice, Italy, February 2008.
36. V. Lopez, "End-to-end quality of service provisioning in multilayer and multidomain environments," Ph.D. dissertation, Universidad Autonoma de Madrid, 2010.
37. J. Domzal, R. Wojcik, A. Jajszczyk, V. Lopez, J. Hernandez, and J. Aracil, "Admission control policies in Flow-Aware Networks," in *Proc. 11th International Conference on Transparent Optical Networks, ICTON 2009*, Azores, Portugal, July 2009, pp. 1–4.
38. J. Domzal, R. Wojcik, K. Wajda, A. Jajszczyk, V. Lopez, J. Hernandez, J. Aracil, C. Cardenas, and M. Gagnaire, "A multi-layer recovery strategy in FAN over WDM architectures," in *Proc. 7th International Workshop on Design of Reliable Communication Networks, DRCN 2009*, Washington, USA, October 2009, pp. 160–167.
39. J. Domzal, K. Wajda, S. Spadaro, J. Sole-Pareta, and D. Careglio, "Recovery, Fairness and Congestion Mechanisms in RPR Networks," in *12th Polish Teletraffic Symposium PSRT*, Poznan, Poland, September 2005. [Online]. Available: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.98.3023>
40. J. Domzal, R. Wojcik, and A. Jajszczyk, "The Impact of Congestion Control Mechanisms on Network Performance after Failure in Flow-Aware Networks," in *Proc. International Workshop on Traffic Management and Traffic Engineering for the Future Internet, FITraME n 2008, Book: Traffic Management and Traffic Engineering for the Future Internet, Lecture Notes on Computer Science 2009*, Porto, Portugal, December 2008.
41. C. Cardenas, M. Gagnaire, V. Lopez, and J. Aracil, "Admission control for Grid services in IP networks," in *Proc. Advanced Networks and Telecommunication Systems, ANTS 2007*, Bombay, India, December 2007.
42. C. Cardenas, M. Gagnaire, V. Lopez, and J. Aracil, "Performance evaluation of the Flow-Aware Networking (FAN) architecture under Grid environment," in *Proc. IEEE Network Operations and Management Symposium, NOMS 2008*, Paris, France, April 2008, pp. 481–487.
43. C. Cárdenas and M. Gagnaire, "Evaluation of Flow-Aware Networking (FAN) architectures under GridFTP traffic," *Future Generation Computer Systems*, vol. 25, pp. 895–903, September 2009. [Online]. Available: <http://portal.acm.org/citation.cfm?id=1550955.1551007>
44. C. Cárdenas, M. Gagnaire, V. Lopez, and J. Aracil, "Admission control in Flow-Aware Networking (FAN) architectures under GridFTP traffic," *Optical Switching and Networking*, vol. 6, pp. 20–28, January 2009.
45. S. Oueslati and J. Roberts, "Comparing Flow-Aware and Flow-Oblivious Adaptive Routing," in *Proc. 41st Annual Conference on Information Sciences and Systems, CISS 2007*, Baltimore, MD, USA, March 2007.

<http://www.springer.com/978-3-319-24973-5>

Guide to Flow-Aware Networking
Quality-of-Service Architectures and Techniques for
Traffic Management

Domżał, J.; Wójcik, R.; Jajszczyk, A.

2015, XVII, 236 p. 124 illus., 26 illus. in color.,

Hardcover

ISBN: 978-3-319-24973-5