AGH University of Science and Technology
Faculty of Electrical Engineering, Automatics, Computer Science and Electronics

Ph.D. Thesis

Jerzy Domżał

Congestion Control in Flow-Aware Networks

Supervisor:
Prof. dr hab. inż. Andrzej Jajszczyk

AGH University of Science and Technology
Faculty of Electrical Engineering, Automatics, Computer Science and Electronics
Department of Telecommunications

Al. Mickiewicza 30, 30-059 Kraków, Poland
tel. +48 12 634 55 82
fax. +48 12 634 23 72

www.agh.edu.pl
www.eaiie.agh.edu.pl
www.kt.agh.edu.pl

Reviewers:

prof. dr hab. inż. Wojciech Kabaciński[1]
prof. dr hab. inż. Andrzej. R. Pach[2]

[1] Poznań University of Technology
[2] AGH University of Science and Technology

ISBN 978-83-88309-57-1

Cover and layout design by Rafał Stankiewicz

Printed in Poland
by *Drukarnia Cyfrowa EIKON PLUS, ul. Wybickiego 46, Kraków*

To Ania, my wife and my love

Acknowledgements

Many people have helped me in my work on this dissertation for the last four years. I would like to thank all of them. There are a few people I want to thank especially. First of all, I would like to express my gratitude towards my supervisor, Professor Andrzej Jajszczyk for his understanding, valuable comments, advice and constant support. I am sure that without Professor Jajszczyk's broad vision and patience, this PhD dissertation would not be made.

I would like to express my sincere gratitude to Krzysztof Wajda for support and help in many fields of my work. His experience of an older colleague were and still is a very important element of my work and life.

I have been fortunate to work with Robert Wójcik, my friend and workmate. His remarks concerning the Flow-Aware Networking concept and many other issues contributed significantly to the improvement of my results. It is a pleasure to work with him.

I would like to thank James Roberts from France Telecom. Our joint work in the project on the Flow-Aware Networking concept was a milestone in my understanding the idea and problems of this network architecture.

My work on this dissertation would not have been possible without the patience, support and love of my Family. I would like to thank my wife Ania, my baby son Adaś and my parents. They deserve my deepest appreciation.

Abstract

The congestion control mechanisms for Flow-Aware Networks are presented in the dissertation. The research was performed for four proposals, the EFM (*Enhanced Flushing Mechanism*), the RAEF (*Remove Active Elastic Flows*), the RBAEF (*Remove and Block Active Elastic Flows*) and the RPAEF (*Remove and Prioritize in access Active Elastic Flows*) mechanisms in two different cross-protect router architectures, with the PFQ (*Priority Fair Queuing*) and with the PDRR (*Priority Deficit Round Robin*) scheduling algorithms. The idea of all presented congestion control mechanisms is similar and relies on total or partial cleaning of the PFL (*Protected Flow List*) content in the MBAC (*Measurement Based Admission Control*) block in congestion.

The main goal of the proposed solutions is to minimize the acceptance time of new streaming flows in the admission control block. The streaming transmission in FAN is proposed for low traffic consuming applications with demands on low packet delays and loss. In the basic version of the MBAC algorithm the new flows cannot be accepted in the congestion state, which sometimes may be kept for a long period of time. The long acceptance time of new streaming flows may not be acceptable by some applications, e.g., VoIP calls. It is important that the new mechanisms cannot considerably deteriorate the traffic performance of the elastic flows, which are subject to the best effort transmission.

The congestion control mechanisms for FAN, proposed in the dissertation are described in details and analyzed by simulation experiments run on the ns-2 simulator. The obtained results show that the new solutions allow for significant decreasing of the acceptance time of a new streaming flow in the MBAC without changing the transmission time of elastic flows. Moreover, it is possible to ensure sufficiently low acceptance times of new streaming flows, which are acceptable by streams realizing the voice transmission, e.g., VoIP calls.

The second part of the dissertation presents the new proposal of realizing the FAN concept. In this solution, the algorithm for random dropping of packets from the queue in congestion is used. It is implemented based on the AFD (*Approximate Fair Dropping*) mechanism. The new proposal is less complex than two well known FAN versions and allows for obtaining similar results when analyzing traffic performance in the network.

Keywords: Flow-Aware Networks, FAN, congestion control, admission control, quality of service, QoS, packet scheduling, DiffServ, internet telephony, VoIP

Streszczenie

W rozprawie zaprezentowano mechanizmy sterowania przeciążeniami w sieciach zorientowanych na przepływy (*Flow-Aware Networks*). W szczególności przedstawione zostały mechanizmy EFM (*Enhanced Flushing Mechanism*), RAEF (*Remove and Block Active Elastic Flows*), RBAEF (*Remove and Block Active Elastic Flows*) oraz RPAEF (*Remove and Prioritize in access Active Elastic Flows*) polegające na okresowym, całkowitym bądź częściowym czyszczeniu listy przepływów chronionych w bloku sterowania dostępem.

Głównym celem zastosowania mechanizmów sterowania przeciążeniami w sieciach FAN jest zapewnienie możliwie szybkiej akceptacji przepływów strumieniowych w bloku sterowania dostępem. Transmisja strumieniowa w sieciach FAN jest przewidziana do obsługi ruchu o niskiej przepływności z zapewnieniem odpowiednio niskich opóźnień i strat pakietów, czyli np. przesyłanie ruchu aplikacji głosowych lub wideo. Istotne jest, by mechanizmy zmniejszające czas rozpoczęcia transmisji dla przepływów strumieniowych nie powodowały znacznego pogorszenia transmisji pozostałego ruchu w sieci.

Zaproponowane mechanizmy sterowania przeciążeniami w sieciach FAN zostały szczegółowo opisane i przeanalizowane przy użyciu symulacji przeprowadzonych w symulatorze ns-2. Uzyskane wyniki pozwalają wnioskować, że nowe rozwiązania umożliwiają znaczące zmniejszenie czasów akceptacji dla nowych przepływów strumieniowych przy jednoczesnym braku zmian czasu transmisji przepływów elastycznych. Co więcej, możliwe jest zapewnienie krótkich czasów akceptacji dla nowych przepływów strumieniowych, spełniających wymagania dla strumieni realizujących transmisje głosowe (w szczególności rozmowy typu VoIP).

Drugą część rozprawy stanowi nowa propozycja realizacji koncepcji Flow-Aware Networking. W rozwiązaniu tym zastosowano algorytm losowego usuwania pakietu z kolejki w sytuacji wystąpienia natłoku, zaimplementowany z użyciem mechanizmu AFD (*Approximate Fair Dropping*). Nowa propozycja jest prostsza

w implementacji i pozwala na uzyskiwanie wyników porównywalnych z innymi rozwiązaniami sieci FAN.

Słowa kluczowe: sieci zorientowane na przepływy, FAN, sterowanie przeciążeniami, blok sterowania dostępem, jakość obsługi, QoS, szeregowanie pakietów, DiffServ, telefonia internetowa, VoIP

Contents

List of Figures

List of Tables

Abbreviations

ABS	Approximate Buffer Size
AC	Admission Control
AF	Assured Forwarding
AFL	Active Flow List
AFAN	Approximate Flow-Aware Networking
AFD	Approximate Fair Dropping
AFRED	Adaptive Flow Random Early Drop
AP	Application Provider
BFL	Blocked Flow List
BFM	Basic Flushing Mechanism
DES	discrete event simulation
DiffServ	Differentiated Services
DPFA	Dynamic Priority Based Flow Aggregation
DPS	Dynamic Priority Scheduling
DRR	Deficit Round Robin
DSCP	Differentiated Services Code Point
EF	Expedited Forwarding
EFM	Enhanced Flushing Mechanism
EIAC	Endpoint Implicit Admission Control

ER	Edge Router
ETSI	European Telecommunications Standards Institute
FAbS	flow-aggregate-based services
FAN	Flow-Aware Networking
FFQ	Frame-based Fair Queuing
FIFO	First-In, First-Out
FSA	Flow State Aware
HDTV	High Definition TV
ID	Identifier
IDFA	Inter-Domain Flow Aggregation
IETF	Internet Engineering Task Force
IntServ	Integrated Services
IP	Internet Protocol
IRM	independent replication method
ISP	Internet Service Provider
ITU-T	International Telecommunication Union
IU	Internet User
LBFA	Link-BAsed Fair Aggregation
MBAC	Measurement Based Admission Control
MPLS	Multi-Protocol Label Switching
MSF	Multiservice Switching Forum
MTU	Maximum Transfer Unit
ns-2	network simulator ver. 2
PAFL	Priority Access Flow List
PDRR	Priority Deficit Round Robin
PFL	Protected Flow List
PFQ	Priority Fair Queuing
PHB	Per Hop Behavior
PIFO	Push-In, First-Out

PQ	Priority Queue
QoS	Quality of Service
RAEF	Remove Active Elastic Flows
RBAEF	Remove and Block Active Elastic Flows
RCFQ	Real-time-Clock Fair Queuing
RED	Random Early Detection
RNG	random number generator
RPAEF	Remove and Prioritize in access Active Elastic Flows
RTT	Round Trip Time
SCFQ	Self-Clocked Fair Queuing
SFQ	Start-time Fair Queuing
SLA	Service Level Agreement
SMAML	Sharing with a Minimum Allocation and Maximum Limit
SNMP	Simple Network Management Protocol
TC	Traffic Class
TCL	Tool Command Language
TCP	Transmission Control Protocol
TFRC	TCP-Friendly Rate Control
ToS	Type of Service
UDP	User Datagram Protocol
VoIP	Voice over Internet Protocol
VoD	Video on Demand
WF2Q	Worst-case Fair Weighted Fair Queuing
WDM	Wavelength Division Multiplexing
WFQ	Weighted Fair Queuing

Part I

Introduction and background

1 Introduction

They say a year in the Internet business is like a dog year.. equivalent to seven years in a regular person's life. In other words, it's evolving fast and faster.

— Vint Cerf

The Internet has changed significantly since it was proposed. Vint Cerf, one of its founding fathers, said that it still evolves very fast. The Internet traffic grows rapidly and new services and applications appear one by one. They need a high bandwidth and good performance to work properly. The problem of high throughput requirements of many applications is usually solved by using high capacity links, but sometimes it may not guarantee the proper transmission parameters in the network. It is not the best solution to increase the amount of resources without providing traffic engineering mechanisms. In many cases, the Internet services and applications in packet networks need QoS (*Quality of Service*) assurances to operate properly. The network performance may be characterized by some basic parameters like availability of connection or traffic delay and loss. These parameters are crucial for games, voice and video applications, cable television services or even file-sharing. The assurance of a high quality for the bandwidth demanding services and applications is a real problem for Internet Service Providers (ISPs) as well as large carriers.

The QoS requirements may be realized by providing specific traffic policies or by the well known mechanisms like the Integrated Services model (IntServ) [14] or the Differentiated Services model (DiffServ) [9]. The former one is hardly ever used, because of its scalability problem. It is very difficult or even impossible to

use it in large networks. The latter one is more popular, but has some drawbacks. DiffServ is complicated and it is easy to evade its mechanisms. Many researchers say that this architecture does not work as expected and are still looking for new effective QoS solutions. A comparison of both mechanisms may be found in [91].

There is a need to provide a new solution for ensuring proper transmission of traffic of relatively new applications (like VoIP) not only on overprovisioned links. Flow-Aware Networking (FAN) [70, 81], proposed in 2004 by S. Oueslati and J. Roberts from France Telecom, is a new concept for packet switched networks with QoS guarantees. The main goal of FAN is to provide maximum possible benefits using the minimal knowledge on the network and the minimal presence of control mechanisms. Two traffic types are defined in this relatively new technique: elastic (for best effort transmission) and streaming (usually used by real-time applications). The packets are classified implicitly to the proper traffic type and served with priority (streaming) or not (elastic). The cross-protect (also known as XP) router is proposed for FAN [58]. The admission control block and the scheduler are two basic parts of it. The former decides on accepting or dropping the packets, while the latter schedules them in the queues. We do not need any signalling or any additional packet marking for transmission in FAN.

The FAN concept is simple and work properly in soft state (congestion-less). In overload we need some new mechanisms to control the access to a FAN link. Four congestion control mechanisms are proposed in this dissertation: EFM (*Enhanced Flushing Mechanism*)[24], RAEF (*Remove Active Elastic Flows*) [23], RBAEF (*Remove and Block Active Elastic Flows*) and RPAEF (*Remove and Prioritize in access Active Elastic Flows*). The main goal of them is to ensure a quick acceptance of new streaming flows in the admission control block. In congestion it is impossible to begin the transmission by a new flow. It means that a user may have to wait for a long time before her or his application, e.g., VoIP call, begins to send the packets. The congestion control mechanisms work based on partial or total cleaning of the content of the PFL (*Protected Flow List*) in the admission control block. The simulation experiments, carried out in the ns-2 simulator [68], show the advantages and drawbacks of the proposed solutions.

The reliable and stable transmission in congestion is more effective if the network architecture is simple. The new architecture of FAN, based on the AFD (*Approximate Fair Dropping*) [74] algorithm and called AFAN (*Approximate Flow-Aware Networking*), is proposed in the second part of this dissertation. In this solution the scheduling and selecting of a packet for sending is simpler than in the basic FAN architectures. Simulation results show that it works as expected and the obtained results for traffic analysis are comparable with those presented for basic FAN. The new architecture may be used in future Internet.

1.1 Scope and thesis

This dissertation proposes congestion control mechanisms for Flow-Aware Networks. The new solutions are described in details and implemented in the ns-2 simulator. The simulation analysis shows their usefulness as well as advantages and drawbacks. The proposed mechanisms ensure short acceptance time of new streaming flows in FAN. Moreover, it is possible to decrease the time, which has to expire before the transmission of a new streaming flow begins without affecting the transmission of other traffic in the link. The new architecture of FAN is also proposed in the dissertation. This solution, simpler than original, is presented with all algorithms needed for its implementation. It is also implemented and analyzed in the ns-2 simulator.

The following thesis of this dissertation has been proposed and proved:

> ***It is possible to define efficient and simple congestion control mechanisms in Flow-Aware Networks.***

The proposed congestion control mechanisms and the new architecture of FAN were intended to be very simple and not to require a high computational complexity. This aim was achieved for all the proposed solutions. The algorithms for realizing them in routers are presented as well as the ranges of values of the necessary parameters are provided.

1.2 Publications

Some of the achievements presented in the dissertation were published in four conference papers. The list of relevant publications is as follows:

[24] J. Domzal and A. Jajszczyk. The Flushing Mechanism for MBAC in Flow-Aware Networks. In *Proceedings of 4th EURO-NGI Conference on Next Generation Internet Networks, NGI 2008*, pages 77–83, Krakow, Poland, April 2008.

[23] J. Domzal and A. Jajszczyk. New Congestion Control Mechanisms for Flow-Aware Networks. In *Proceedings of International Conference on Communications, ICC 2008*, Beijing, China, May 2008.

[25] J. Domzal and A. Jajszczyk. The Impact of Congestion Control Mechanisms for Flow-Aware Networks on Traffic Assignment in Two Router Architectures. In *Proceedings of International Conference on the Latest Advances in Networks, ICLAN 2008*, Toulouse, France, December 2008.

[26] J. Domzal, R. Wojcik, and A. Jajszczyk. The Impact of Congestion Control Mechanisms on Network Performance after Failure in Flow-Aware Networks. In *Proceedings of International Workshop on Traffic Management and Traffic Engineering for the Future Internet, FITraMEn 2008*, Porto, Portugal, December 2008.

The flushing mechanism is presented in [24]. Two versions of this solution are described and analyzed. The EFM (*Enhanced Flushing Mechanism*) is an improved version of the BFM (*Basic Flushing Mechanism*). In the EFM, only the identifiers (IDs) of elastic flows are removed from the PFL, while in the BFM the whole content of the PFL is cleaned in congestion. Once accepted streaming flows are never dropped from the admission control block in the former solution. It allows for more reliable transmission of streaming connections.

Three congestion control mechanisms, EFM, RAEF and RBAEF, are analyzed in [23]. They are implemented in the FAN module with the PFQ (*Priority Fair Queuing*) scheduling algorithm for the ns-2 simulator. The results of the analysis show that the mechanisms are an interesting solution to be used in FAN. They allow for decreasing the acceptance time of new streaming flows without decreasing traffic performance of other flows in a link. The proposed mechanisms are compared. Network operators have a possibility to choose the algorithm that best suits their needs.

The same congestion control algorithms are described and analyzed in [25], but in FAN with the PDRR (*Priority Deficit Round Robin*) scheduling algorithm. The results show the similarities and differences in effects of using the proposed congestion control mechanisms, independently of the FAN architecture. The acceptance times of new streaming flows as well as some other transmission parameters of both traffic types are analyzed in the paper.

The paper [26] shows that the congestion control mechanisms proposed for FAN play a good role in a situation when a link or node fails and the traffic has to be redirected via other links. If we do not have a backup link, the traffic has to be sent through the link used for normal transmission. If the FAN link on the new route is overloaded the active flows from the failed link are not accepted in the admission control block of the new router and their transmission is stopped. If we use the congestion control mechanisms, the transmission of redirected flows may be continued immediately.

1.3 Structure of the dissertation

The dissertation is organized into three parts. The introduction, thesis and the theoretical background for the research is presented in the first part (Chapters 1 – 3). Chapter 1 shows the general information on the issues presented in the

dissertation. In Chapter 2, the characteristics of Diffserv and net neutrality concepts are given. The FAN architecture is described in details. Chapter 3 provides an overview of the literature related to the research on FAN and the congestion control issue. The most important papers and books are briefly described to show that the task analyzed in the dissertation is very important.

The results of theoretical and simulation analysis provided by the PhD candidate are presented in the second part of the dissertation (Chapters 4 – 5). The new methods for estimating the congestion control indicators in FAN are given in Chapter 4. The crucial parameters, which need to be set in the FAN routers, are also presented. The accepted ranges of their values are provided and explained. Section **??** presents the congestion control mechanisms proposed by the PhD candidate for FAN. The EFM, RAEF, RBAEF and RPAEF algorithms are presented in details as well as their advantages and drawbacks. The simulation analysis of congestion control mechanisms for both basic versions of FAN (with PFQ and PDRR scheduling algorithms) is provided in Chapter 5. Chapter 6 presents a new version of FAN based on the AFD algorithm and called by the author AFAN (*Approximate Flow-Aware Networking*). The methods for realizing the new concept and the algorithm for serving the packets are given in details. The results of simulation analysis of AFAN are presented in Section 6.3. The simulation experiments were run to show that AFAN works similarly to other FAN versions. The congestion control mechanisms proposed for FAN may be successfully implemented in AFAN and allow for ensuring short acceptance times for new streaming flows.

In the third part of the dissertation only one chapter is placed. Chapter 7 summarizes the research presented in the dissertation and gives some hints for network operators.

An example of the TCL file is presented in the appendix. It was used by the author as a basic script for the simulation experiments.

2 Area of research

The advance of technology is based on making it fit in so that you don't really even notice it, so it's part of everyday life.

— Bill Gates

The goal of this chapter is to present the motivation for developing the FAN concept. The brief description of DiffServ is presented in the first section. The possibilities of realizing QoS guaranties in packet networks are given. It is shown that DiffServ is complex and in may cases does not work as expected. In the second section, the net neutrality concept is described. Current network architectures should be neutral and ensure fair access to the resources independently of used applications or services. The main body of this chapter is the detailed description of Flow-Aware Networks as a neutral network proposal, which resolves the DiffServ inconveniences.

2.1 QoS and Differentiated Services model

2.1.1 Quality of Service

Quality of Service (QoS) in packet networks may be identified and understood in many ways. From the engineer point of view, it should be seen as a possibility to ensure low packet loss and delays, high bandwidth utilization, fair access to

the resources for the same traffic types and priorities between other traffic types. The end user, on the other hand, should be satisfied with the perceived quality of all his or her applications and services.

There are two main approaches to the QoS concept. The definition *"a set of service requirements to be met by the network while transporting a flow"* given by IETF (*Internet Engineering Task Force*) in [20] meets the requirements desired by the scientists without analyzing the perceived quality of services. This engineering approach is considered in most science and technical analyzes. The definition of QoS: *"the collective effect of service performance which determine the degree of satisfaction of a user of the service"*, given by ITU-T (*International Telecommunication Union*) and presented in [42] represents the second approach. The detailed description of QoS mechanisms presented by ITU-T may also be found in [43]. Almost the same opinion on QoS is given by ETSI (*European Telecommunications Standards Institute*) in [28] and in [44].

The analysis of QoS, presented by the author in this dissertation, is based on a mixed approach. The obtained results of the simulation analysis allow for improving the transmission parameters of streaming flows in FAN. The positive effects of implementing the congestion control mechanisms as well as the new version of FAN may also be perceived by end users.

2.1.2 Differentiated Services

The Differentiated Services model [9, 17] is a proposal, which allows for ensuring the different levels of service quality in IP networks. The detailed methods for implementing DiffServ in IP networks lie in network operators' hands. They may configure network elements to classify the packets to different number of classes. The functionality of the model is usually implemented in the edge routers (ER) of a DiffServ domain. The functionality of ERs is presented in Fig. 2.1. The incoming packets are assigned to the proper service class in the classifier block. Packets are selected based on the information from IP headers. They have to meet the requirements of the negotiated SLA (*Service Level Agreement*). The classified packet is sent to the marker block, which sets the DSCP (*Differentiated Services Code Point*) value of the packet written to the packet header: the Type of Service (ToS) byte in IPv4 [73] or the Traffic Class (TC) byte in IPv6 [22]. Marker makes a decision based on the information from the classifier and meter. The meter block collects some information from the network as a result of measurements provided during transmission and sends them to the other blocks. It is responsible for checking the compliance of the transmitted traffic to the profile described in the contract between a customer and a network operator. The main function of a shaper is to ensure that the traffic meets the requirements of a negotiated traffic profile. It may delay or even drop packets in order to shape the stream. This

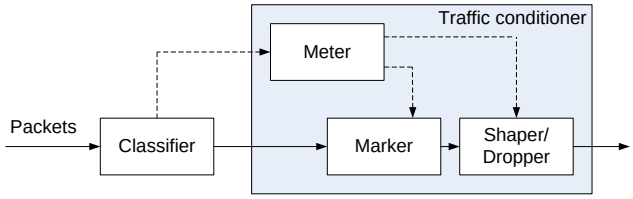

Figure 2.1: Functionality of edge router in DiffServ domain

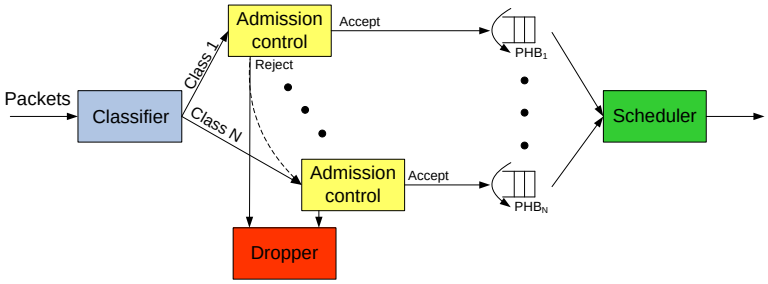

Figure 2.2: Logical structure of edge router

block is used for smoothing the traffic and eliminating some packets of flows causing congestion.

The logical structure of an edge router is presented in Fig. 2.2. Firstly, a flow must be identified by the classifier block and assigned to a certain class. The classification process is explicit because nodes are pre-informed on how to recognize and treat a particular transmission. The classified packets are sent to the block realizing the admission control function (marker and meter). The accepted packets are put to the proper queue with specified rules for treating them. The rules are called PHB (*Per Hop Behavior*). The packets form the queues are scheduled in the scheduler block, which functions as a shaper.

Two main traffic classes are defined in DiffServ: Assured Forwarding (AF) [39] and Expedited Forwarding (EF) [21]. Packets not classified to any group are sent as a best effort service. The AF class is proposed for serving the traffic with guaranteed minimum throughput in one or in multiple domains [62]. It is usually used by flows, which do not allow any loss during transmission. The delay and jitter are not crucial for them. When a network is not overloaded the available bandwidth is fairly shared by the competing flows, while in congestion the flows of the AF class transmit their traffic with the minimum guaranteed rate. The AF class is usually used for traffic using TCP connections. The RED (*Random Early Detection*) mechanism, described in [30], is usually used for scheduling

the packets in the buffer. The Expedited Forwarding class is used by flows, which need low loss, latency and jitter, e.g., for VoIP (*Voice over IP*) or VoD (*Video on Demand*) services [60]. These flows accept low packet loss but have to be served quickly. A separate queue is usually used for the AF traffic class in real implementations. There are many mechanisms which realize the function of shaper, e.g., WFQ (*Weighted Fair Queuing*) [53], SCFQ (*Self-Clocked Fair Queuing*) [35], FFQ (*Frame-based Fair Queuing*) [95], WF2Q (*Worst-case Fair Weighted Fair Queuing*) and WF2Q+ [8], SFQ (*Start-time Fair Queuing*) [36] or DRR (*Deficit Round Robin*) [92].

The traffic engineering aspects for Quality of Service are presented in [77]. The author shows the need for QoS in current networks. Two types of traffic are described in details: elastic and streaming. The packets of both flow types should be served with the admission control mechanism implemented in the network. This simple service model is used in Flow-Aware Networks as an alternative to the DiffServ model.

2.2 Net Neutrality

The quality assurance is a very important issue, which may help to guarantee the proper traffic characteristics for all traffic types. On the other hand, the QoS mechanisms may be used by Internet Service Providers (ISPs) to prioritize the traffic of selected applications. The effect of such a behavior is that the packets of the similar applications may be treated differently depending not on the type but, e.g., on the content providers or charging aspects. The idea of net neutrality is that an application traffic is not discriminated at all in relation to a traffic generated by other network applications or services. The legal conditions of the net neutrality are still discussed all over the world, including the United States Congress. It is a very hot topic in the consideration of the future Internet. In the most rigorous concept of the net neutrality, all the Internet traffic is sent as a best effort service and the ISPs are not allowed to introduce any kind of traffic discrimination. It means that there are no possibilities to give a priority to any type of traffic, application or service and the mechanisms like IntServ or DiffServ are prohibited.

The definition of net neutrality is still under consideration. Many researchers as well as politicians work on the most suitable term of it. In [37] four nightmare scenarios for the net neutrality issue are presented. The first one, called "inequity nightmare", assumes that big companies may offer a special tier, the access to which will be more expensive for the users. The net neutrality followers claimed that such an investment may lead to an advanced Internet that will be available to only a small part of users. It is contrary to the idea of the Internet. A similar approach is presented in the second scenario: "corporate bureaucracy

nightmare". According to it, large corporate broadband firms may require a special charge for the access to the upper tier, e.g., for a new web site that operates better on this tier or requires technical assistance from the access providers. This is the straight way for the telecommunication companies or ISPs to extract more money from the users who use the expensive services. The third scenario, named "bad incentive nightmare" shows the situation where network operators have their own services (like VoIP or VoD) and may not allow for effective work of similar applications provided by other companies. It discriminates the competition and is very unfavorable for developing new Internet services. The small firms, like software producers and developers, have almost no chance to promote their solutions if this scenario is possible. The last presented scenario, called "less innovative content nightmare" is related to the previous one. It involves worries that the firms may produce new applications and services and protect their interests in those applications giving no chance for using and developing them by other providers. It also causes that the development of Internet is not as fast as it could be. It is possible that service providers may be charged twice, firstly to its own network operators and secondly to the ISP of every single user who wants access to that services' content. This scenario is presented in [99]. It may begin to break the unique many-to-many nature of the Internet. The net neutrality problem is complex. Its definition represent a conflict of interests between application providers (APs), Internet users (IUs), and ISPs.

Some APs and IUs groups think that all traffic in the Internet should be served in the best effort manner. They agree with the authors of the most rigid version of net neutrality and believe that the access to the Internet services should be fair and cheap or even free. They argue that current network links have capacities high enough to carry all traffic with the proper guarantees. In most cases it is really true because network resources are often over-dimensioned, but we have to be aware of an enormous progress in the telecommunications, especially in the area of access networks. The engineers have to be aware of the fact that new applications and services grow rapidly and the number of Internet users rises significantly as well. Probably, in future the network link capacities will not be sufficient to carry all traffic with a proper QoS. It is one of the key arguments raised by ISPs to prohibit the implementation of the net neutrality concept in the Internet. In their opinion, service differentiation and traffic prioritization have to be allowed for proper transmission in current and future networks. They claim that the data transmission in the networks without QoS mechanisms will be unacceptable from the user point of view in the future. ISPs are also concerned that providing the net neutrality may discourage development of new services and applications because it is necessary to give the priorities to their traffic. As an effect of the lack of application and service development, the network investment may be slowed down. Currently, as a result of a possible consensus,

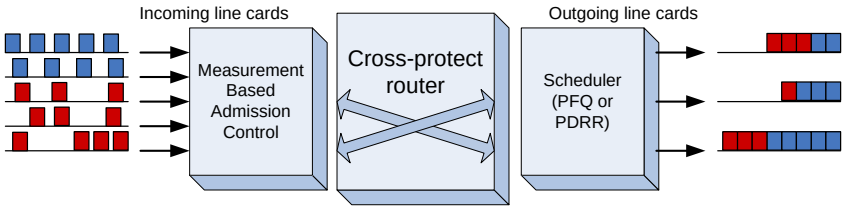

Figure 2.3: Architecture of cross-protect router

the net neutrality followers may agree with differentiation of services, but without additional charging for it. They argue that some applications may be blocked or a traffic of the selected services might be poorly treated if appropriate fees are not paid to the ISPs.

There are also some other proposals of how to cope with the net neutrality problem. The question how to guarantee the proper quality of the transmitted traffic has been a hot topic for the last 15 years. The network architecture that will allow for implicit traffic differentiation and prioritization of a selected traffic without user or ISP intervention may be a desired proposal. Flow-Aware Networking architecture is a good solution to ensure the net neutrality with the awareness of QoS.

2.3 Flow-Aware Networking

The FAN concept was proposed by S. Oueslati and J. Roberts from France Telecom as a new method for realizing the data transfer with guaranteed quality (QoS) [70]. The traffic is implicitly classified and sent as flows. Two types of flows are considered in FAN:

- *elastic* — usually used for data transmission as a best effort service;
- *streaming* — used for low bandwidth consuming services, e.g., real-time applications like VoIP calls, and treated as priority traffic.

The packets are classified to the flows based on their identifiers (ID) calculated from the content of packet headers (the source and destination ports and addresses) and type of transport protocol [80].

The main elements of the FAN architecture are: the MBAC (*Measurement Based Admission Control*) block, sometimes referred to as AC (*Admission Control*) block and the scheduler block, which make up the cross-protect router (also called XP router) [58] The architecture of the FAN router is presented in Fig. 2.3.

Both mechanisms realizing the functionality of XP router blocks depend on each other. The values measured in the scheduler block are the input information for the AC block. On the other hand, in the first mechanism, the decisions of accepting or dropping packets of a flow are taken.

2.3.1 Admission Control operation

The PFL (*Protected Flow List*) is the list of flows being served at the moment. If a packet arriving to the admission control block belongs to the flow, the identifier of which is in the PFL, it is accepted and sent to the scheduler block for queuing. The time of such an event is updated in the PFL. However, if the identifier of the flow is not in the PFL, the decision of accepting or dropping the packet must be taken based on two parameters from the scheduler block. In that part of the cross-protect router, the values of the *priority_load* and the *fair_rate* are estimated. They are described as follows:

- *fair_rate* — the maximum rate that is or might be realized by a flow,
- *priority_load* — the quotient, which represents the rate of incoming priority packets with reference to the link capacity.

There is no possibility to accept a new flow and serve its packets in congestion. This state is noticed when the *fair_rate* value is less or equal to the *min_fair_rate* (minimum allowed value of the *fair_rate*) or the *priority_load* value is greater than or equal to the *max_priority_load* (maximum allowed value of the *priority_load*). The first discarded packet should play a role of a congestion indicator for the source node. It is expected that applications would be designed to react to this information appropriately by slowing the rate of its transmission. New flows may be accepted in the AC block only in the congestion-less state. A packet of a new flow is always accepted in the AC block and sent for queuing if congestion is not observed. The ID of the flow it belongs to is added to PFL with probability *P_add*. If *P_add* is small (e.g., 0.1) the IDs of most very small flows are never included in the PFL. On the other hand, the identifiers of big flows are added to the list in a short time. The goal of this mechanism is not to allow for overloading the PFL. If there is no update in the PFL for a flow during the specified time period (*flow_time_out*), the ID of such a flow is removed from the PFL. The idea of FAN is that flows transmitting the traffic with a rate lower than the *min_fair_rate* are treated as streaming flows. Their packets should be queued in the priority queues. Flows, which want to send their traffic with a rate higher than the *min_fair_rate* are ought to be classified as elastic. The described algorithm is possible to realize thanks to implementation of the scheduler module. A queuing algorithm, implemented in the scheduler block is the most important mechanism while considering the congestion control issue. It decides (by measurements) if

new flows may begin their transmission. Over time, numerous queuing algorithms have been proposed. Two of them may be implemented in the cross-protect router: PFQ (*Priority Fair Queuing*), proposed in [58], and PDRR (*Priority Deficit Round Robin*) presented in [57, 59]. The description of both proposals is given in details in the following two sections.

2.3.2 Priority Fair Queuing

The PFQ (*Priority Fair Queuing*) algorithm inherits the advantages from the SFQ (*Start-time Fair Queuing*), described in [36], by providing the prioritizing possibilities in the scheduler module. PFQ implicitly gives priority to the packets of flows whose peak rate is less than the *fair_rate*. In the scheduler with the PFQ algorithm, the new objects have to be implemented. The PIFO (*Push-In, First-Out*) queue is defined for storing the packets in the decreasing order with respect to their time stamp. The pointer P identifies the last of the priority packets at the head of the queue. The object AFL (*Active Flow List*) contains the identifiers of all active flows. The value of the *virtual_time* counter is equal to the start tag of the last packet that has begun transmission. The complexity of PFQ is similar to that of SFQ and is logarithmic with respect to the number of active flows. The algorithm ensures scalability because the number of active flows is limited by the admission control block.

The pseudo code for enqueuing the packets in FAN with PFQ is presented in Tab. 2.1.

The incoming packet, accepted in MBAC, may be queued in the PIFO queue only if it is in the congestion-less state. In the other case, the flow with the longest backlog (the number of bytes in the buffer) is selected and its last enqueued packet is removed from the buffer (lines 1-3). If the identifier of flow F, the incoming packet p belongs to, is in the AFL, the packet p may is enqueued and the backlog of its flow is increased by the packet length in bytes (lines 4-6). The packet p may be pushed to the priority part of the buffer or to another part, where packets realizing the best effort transmission are enqueued (also known as elastic part). Packets are classified as priority while the cumulative volume of transmitted bytes of their flow is less than the maximum packet size, MTU (line 7). It enables to realize implicit differentiation between packets. Packets of elastic flows are queued with the proper value of *flow_time_stamp* while the packets of streaming flows are pushed to the queue with the stamp equal to the *virtual_time* value (lines 8-10). The pointer P is used for distinguishing the packets of the backlogged and non-backlogged flows having the same time stamp and necessary to realize priority queuing. Its value is updated after any enqueuing operation (lines 11 and 18). The value of the cumulative volume of transmitted bytes is increased by packet's p length if it is enqueued in the priority part of the buffer

Table 2.1: Pseudo code of enqueuing module for FAN with PFQ

```
Enqueuing module:
1.  on arrival of packet p
2.  If PIFO is congested then
3.     reject packet at a head of longest backlog
4.  If F ∈ AFL then
5.  begin
6.     backlog(F) = backlog(F) + size(p)
7.    If bytes(F) ≥ MTU then
8.       push {packet, flow_time_stamp} to PIFO
9.    Else begin
10.       push {packet, virtual_time} to PIFO
11.      behind P; update P
12.      bytes(F) = bytes(F) + size(p)
13.    end
14.      flow_time_stamp(F) = flow_time_stamp(F) + size(p)
15.  end
16.  Else begin
17.    push {packet, virtual_time} to PIFO
18.    behind P; update P
19.    If AFL is not saturated then
20.    begin
21.       add flow F to AFL
22.       flow_time_stamp(F) = virtual_time + size(p)
23.       backlog(F) = size(p)
24.       bytes(F) = size(p)
25.    end
26.  end
```

(line 12). In the other case, when packet p is pushed to the elastic part of the buffer the *flow_time_stamp* of its flow is increased by packet's p length (line 14).

If packet p represents the flow, which ID is not written to the AFL, it is treated as streaming and enqueued in the priority part of the buffer (lines 16-17). Moreover, if the AFL is not saturated, the ID of flow F is added to the AFL and the *flow_time_stamp* is set to the value of *virtual_time* increased by the packet length. The backlog of the added flow and the cumulative volume of transmitted bytes are set to the initial values of the packet size.

The pseudo code for dequeuing the packets in FAN with PFQ is presented in Tab. 2.2.

If the buffer is empty the content of the AFL has to be cleaned (lines 1-2).

Table 2.2: Pseudo code of dequeuing module for FAN with PFQ

```
Dequeuing module:
1.  If PIFO is empty then
2.     clean AFL content
3.  Else begin
4.     backlog(F) = backlog(F) − size(p)
5.     serve packet at head of line
6.     next_time_stamp = time_stamp(p)
7.     If next_time_stamp ≠ virtual_time then
8.     begin
9.        virtual_time = next_time_stamp
10.       for all flows f ∈ AFL
11.       begin
12.          If next_time_stamp(f) ≤ virtual_time then
13.             remove f from AFL
14.       end
15.    end
16. end
```

In the other case, the backlog of flow F, represented by the packet p selected for dequeuing, is decreased by the packet length (line 4). After sending the packet from the head of the line the *next_time_stamp* parameter is set to the value of *time_stamp* of packet p (lines 5-6). Then, if the *next_time_stamp* parameter is less or equal to *virtual_time*, the *virtual_time* is set to the value of *next_time_stamp* parameter (lines 7-9). At the same time, all flows from the AFL, the *next_time_stamp*(f) of which is less or equal to *virtual_time*, are removed from the AFL.

In PFQ, the *fair_rate* is computed from the following formula:

$$fair\_rate = \frac{max\{S \times C, (vt(t_2) - vt(t_1)) \times 8\}}{t_2 - t_1} \tag{2.1}$$

where $vt(t)$ is the *virtual_time* in time t and represents the start tag of the last packet of the fictitious permanently backlogged flow, which sends 1 byte long packets between packets of real flows in proper order (compatible with the algorithm assumptions), (t_1, t_2) is the time period measured in seconds, S is the total length of inactivity in the transmission during the (t_1, t_2) period, C is the link bit rate.

We may conclude from Formula (2.1), that the first value from the curly bracket is chosen when the link is lightly loaded and allows for using the high priority for almost all flows. The value of *fair_rate* is usually high and new

flows may begin their transmission. The second value is equal to the throughput allocated to every active elastic flow and is chosen when the link is heavily loaded. The flows with rates lower than the *fair_rate* have the high priority. In that way, streaming flows with peak rates less than the *fair_rate* are subject to the bufferless multiplexing and, therefore, perceive low delays and losses.

The *priority_load* is estimated from the following formula:

$$priority\_load = \frac{(pb(t_2) - pb(t_1)) \times 8}{C(t_2 - t_1)} \tag{2.2}$$

where $pb(t)$ is the value of the counter increased by the packet's length in bytes, when this packet arrives at the router, (t_1, t_2) is the time period, measured in seconds, in which the measurement is performed, C is the link bit rate.

2.3.3 Priority Deficit Round Robin

The PDRR (*Priority Deficit Round Robin*) is a fair queuing algorithm based on the DRR (*Deficit Round Robin*) scheduling mechanism [92]. PDRR inherits the advantages from DRR (e.g., $O(1)$ complexity and fairness) and improves packet latency by using the priority queue for low rate flows (streaming flows). The PDRR algorithm allows for discriminating flows on the basis of the transmission rate. The bottlenecked flows have the guaranteed current max-min fair rate. Packets of flows with the transmission rate less than the current *fair_rate* receive priority and are transmitted through the priority queue. It allows for distinction between streaming and elastic flows (the streaming flows usually are transmitted with a low peak rate). In the scheduler with the PDRR algorithm, the new objects have to be implemented. The FIFO (*First-In, First-Out*) queues are defined for storing the packets in the buffer. The packets of streaming flows are transmitted through the priority queue, while each elastic flow has its own queue. AFL (*Active Flow List*) is a data structure that stores information (flow identity, current flow deficit counter DC, flow quantum Q and pointers realizing a FIFO linked list of queued packets) for the flows that have or have recently had packets in the queue. The $ByteCount(i)$ is used to determine whether the incoming packets of flow i should or should not be sent via the priority queue. If $ByteCount(i)$ is less than $Q(i)$ packets are treated as prioritized. The entries of the AFL are visited in a certain order in each scheduling cycle. The complexity of PDRR is higher than the $O(1)$ DDR complexity because some empty queues may be visited before sending a packet in the dequeue operation. It may be corrected by maintaining the list of only non-empty queues. As well as the PFQ, the PDRR algorithm ensures scalability. The number of active flows is limited by the admission control block [55].

The pseudo code for enqueuing the packets in FAN with PDRR is presented in Tab. 2.3.

Table 2.3: Pseudo code of enqueuing module for FAN with PDRR

Enqueuing module:	
1.	on arrival of packet p
2.	**If** no place in the buffer **then**
3.	reject packet from the longest queue
4.	**If** $F \notin AFL$ **then**
5.	begin
6.	add flow F to AFL
7.	$DC(F) = 0$
8.	$ByteCountF = size(p)$
9.	$Enqueue(PQ, p)$
10.	end
11.	**Else** begin
12.	$ByteCountF = ByteCountF + size(p)$
13.	**If** $ByteCountF \leq Q(F)$ **then**
14.	$Enqueue(PQ, p)$
15.	**Else**
16.	$Enqueue(Queue(F), p)$
17.	end

If there is no space in the buffer for incoming packet p accepted in MBAC, the last packet of the flow with the highest number of enqueued packets is removed from the buffer (lines 1-3). If the AFL is not saturated, and if the flow F represented by packet p is not in the AFL, it is added to the list. The deficit count $DC(F)$ is set to its initial value equal to zero and $ByteCount(F)$ is set to its initial value of packet's p size. After these operations, packet p is pushed to the priority queue (lines 4-10). On the other hand, if the ID of F is in the AFL, the $ByteCount(F)$ is increased by the packet size. At the same time, if the increased recently $ByteCount(F)$ is less or equal to the flow quantum, packet p is enqueued in the priority queue. In the other case, if $ByteCount(F)$ is higher than the flow quantum, packet p is pushed to its own queue for elastic packets (lines 11-17).

The pseudo code for dequeuing packets in FAN with PDRR is presented in Tab. 2.4.

The dequeuing operation begins from checking if the buffer is empty, and if so, the content of the AFL has to be cleaned (lines 1-2). In the other case, the packets from the priority queue are sent (lines 4-7). The deficit counter of flow F represented by the packet p selected for dequeuing is decreased by this

Table 2.4: Pseudo code of dequeuing module for FAN with PDRR

```
Dequeuing module:
1.  If buffer is empty then
2.     clean AFL content
3.  Else begin
4.     While PQ is not empty do
5.     begin
6.        p = Dequeue(PQ)
7.        Send(p)
8.        DC(F) = DC(F) − size(p)
9.     end
10.    If AFL is not empty then
11.    begin
12.       get flow f from the head of AFL
13.       DC(f) = DC(f) + Q(f)
14.       While DC(f) ≥ 0 and Queue(f) is not empty do
15.       begin
16.          PacketSize = Size(Head(Queue(f)))
17.          If PacketSize ≤ DC(f) then
18.          begin
19.             Send(Dequeue(Queue(f)))
20.             DC(f) = DC(f) − PacketSize
21.          end
22.          Else break; (*skip while loop*)
23.       end
24.       remove flow f from AFL
25.       If Queue(f) is not empty then
26.          insert flow f in AFL
27.    end
```

packet's size (line 8). If the priority queue is empty, the packets of elastic flows may be sent (lines 10-27). The deficit counter of the first flow f from the AFL is increased by the flow's quantum (line 13). While this deficit counter is greater than or equal to zero and the queue of flow f is not empty, the packets of flow f are selected for sending (lines 14-23). If the size of the packet selected for sending is less or equal to the deficit counter of the corresponding flow, the packet is sent and the deficit counter is decreased by the packet size (lines 17-21). In the other case, the dequeuing operation is broken. The flow f is removed from the AFL and added to it again (lines 22-27). It allows for changing the place of flow f in the AFL from first to last.

In PDRR, the *priority_load* is estimated as in the PFQ from Formula (2.2). The *fair_rate* is computed from the following formula:

$$fair\_rate = \frac{max\{S \times C, fair\_bytes \times 8\}}{t_2 - t_1} \tag{2.3}$$

where *fair_bytes* is a 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, C is the link bit rate.

We can conclude from Formula (2.3), that the first value from the curly bracket is chosen when the link is lightly loaded and allows for using the high priority for almost all flows. The second value is equal to the throughput allocated to every active flow and is chosen when the link is heavily loaded. The flows with rates less than *fair_rate* are assigned a high priority.

The smoothing parameter α is applied in both versions of FAN, such that:

$$\begin{aligned} fair\_rate(n) &= \alpha \times fair\_rate(n-1) \\ &+(1-\alpha) \times measured\_fair\_rate(n) \end{aligned} \tag{2.4}$$

where $fair\_rate(n)$ is the value of *fair_rate* in the n-th iteration and the $measured\_fair\_rate$ is the value calculated from the Formula (2.1) or (2.3) in the n-th iteration.

The smoothing parameter β is applied when computing the *priority_load* values in both FAN versions:

$$\begin{aligned} priority\_load(n) &= \beta \times priority\_load(n-1) \\ &+(1-\beta) \times measured\_priority\_load(n) \end{aligned} \tag{2.5}$$

where $priority\_load(n)$ is the value of *priority_load* in the n-th iteration and the $measured\_priority\_load$ is the value calculated from the Formula (2.2) in the n-th iteration.

The logical structure of the FAN router is presented in Fig. 2.4. The header of an incoming packet is analyzed in the flow analyzer block to calculate the flow ID. If flow's ID is in the PFL, the packet is queued and served in the scheduler block. On the other hand, based on the decision made in the admission control block, it may be discarded or accepted. In the latter case, the flow ID is added to the PFL and its packet is scheduled. This simple structure is consistent with the net neutrality concept.

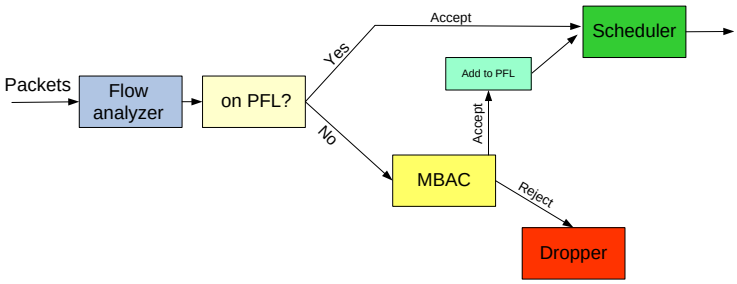

Figure 2.4: Logical structure of edge router

2.4 Congestion Control vs. Congestion Avoidance

The congestion control and congestion avoidance are different ideas, but have the same goal: to ensure effective and efficient operation of the network. In the first case, the resources should be properly (fairly) allocated when a failure occurs in the network. The goal of the second idea is to allocate resources in such a way that the congestion should not take place at any time. The general patterns of the throughput and response time as the network load increases are presented in Fig. 2.5. If the network load is low, the throughput increases linearly along with the increasing demands from the sources. In point (a) the traffic load is high and it causes that the throughput increases slowly and the link is almost congested. If the traffic load still increases, the congestion occurs in point (b) and some packets are dropped. A similar situation takes place if we consider the second characteristic (the response time in function of the traffic load). The congestion avoidance mechanism has to ensure the operation of the network in the vicinity of point (a), while the congestion control mechanism has to ensure the proper work of network on the left side of point (b). The well planned congestion avoidance mechanism allows for increasing traffic load if the response time is low and forces the decrease of the traffic load when the response time quickly increases. The algorithm realizing the congestion control mechanism decides which packets and how many of them need to be dropped in case of congestion. The basic architecture of the congestion control system is presented in Fig. 2.6. The most popular method for sending information that indicates the congestion is based on using one bit in the confirmation packets (1 shows that the network is in the congestion state, while 0 indicates no congestion). Following this information, the algorithm decides if the traffic should be increased or decreased.

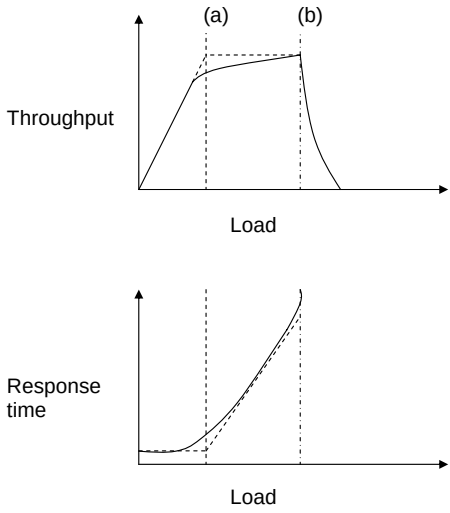

Figure 2.5: Network parameters in function of the load; dashed lines show the theoretical (deterministic) shapes of the curves [18]

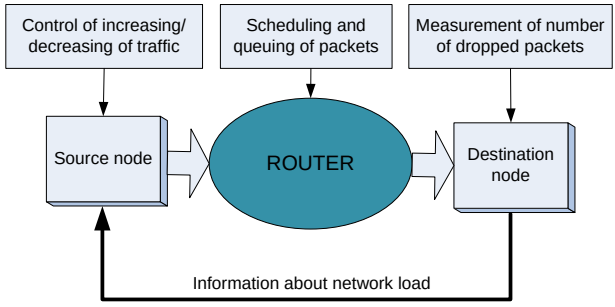

Figure 2.6: The basic architecture of congestion control system [66]

The most effective method for controlling the traffic load is presented in the following formula:

$$x_i(t+1) = \begin{cases} a_1 + x_i(t) & \text{when } y(t) = 0 \quad \Longrightarrow \text{increase} \\ b_D x_i(t) & \text{when } y(t) = 1 \quad \Longrightarrow \text{decrease} \end{cases} \tag{2.6}$$

where $x(t)$ is the source and $y(t)$ is the response information. Increasing traffic load by adding a certain information and decreasing it by multiplying by the factor lower than one, allows to obtain the least oscillations near the (a) point of Fig. 2.5.

The congestion in FAN has to be analyzed in two ways. The congestion in the admission control block is indicated when the values of *fair_rate* or *priority_load* are beyond the fixed thresholds. No new flows can begin their transmission in this state, but packets of all flows accepted before have to be served. The congestion of the buffer for the queued packets is noticed if there is no space for new packets in it. If the buffer is overloaded even the packets of the protected flows may be dropped. There is a simple congestion control algorithm and no congestion avoidance mechanisms for buffers in FAN. If an arriving packet has to be queued to the overloaded buffer, then the packet at the head of the longest backlog is removed from the buffer and dropped. The free space in the buffer is then available for a new packet. This mechanism, proposed by Suter in [96], does not play the role of the congestion avoidance tool because it causes that the buffer may be overloaded for a long time. The congestion control mechanisms for FAN presented in this dissertation allow for accepting new flows in congestion noticed in the admission control block and control their number and the type of them. Moreover, the buffer is never overloaded in the new version of FAN (AFAN) proposed by the author. The implemented AFD (*Approximate Fair Dropping*) algorithm plays the role of a congestion avoidance mechanism for the buffer in AFAN.

3 Related work

The difficulty lies, not in the new ideas, but in escaping the old ones, which ramify, for those brought up as most of us have been, into every corner of our minds.

— John Maynard Keynes

The survey of the literature related to research on FAN, congestion and admission control is given in this chapter. The basic description of FAN, the architecture proposals and results of some simulation experiments are presented in papers discussed in Section 3.1. The new concepts of mechanisms, which improve the operation of FAN are shown in papers listed in Section 3.2. The selected papers on statistical bandwidth sharing and Internet traffic theory are presented in Section 3.3. The admission control concept is presented in papers discussed in Section 3.4. The idea of this part is to show some solutions for realizing admission control mechanisms and the motivation and possibility for implementing MBAC in FAN. The research results on congestion control and congestion avoidance mechanisms are provided in Section 3.5.

3.1 Flow-Aware Networking concept

The first paper presenting the Flow-Aware Networking concept was published in 2000 by James Roberts and Sara Oueslati from France Telecom [82]. The

possibility of ensuring the QoS in the Internet by using FAN as well as the nature of the Internet traffic are presented in this paper. It is shown that the problem of service differentiation may be solved by using Flow-Aware Networks. Various aspects of such a solution, e.g. flow identification, measurement-based admission control, flow aware routing and pricing are presented.

The characteristics of the Internet traffic are described in [10, 80]. The QoS guaranties and pricing requirements are highlighted in the first paper. The main blocks of FAN as well as the basic router architecture are also presented. The second paper gives some claims on QoS, provisioning for transparency and controlling accessibility, along with short explanations.

The FAN router architecture (cross-protect router) is presented in [58]. The scheduling algorithm (PFQ) is described with pseudocode for implementing it in the device. The methods for estimating the *fair_rate* and *priority_load* in the scheduler block are discussed. Some simulation results on FAN are also presented in the paper.

The overall proposal of FAN is described in [70]. The functionality of the admission control and scheduler blocks is presented as well as a performance analysis and some aspects of the Internet design philosophy. The rationale of the new architecture with some proposals on how to implement FAN in current networks are given.

The new version of FAN is presented in [59]. The PDRR algorithm, to be used in the scheduler block, is proposed. The pseudocode of algorithm realizing the PDRR functionality is given and explained. The performance analysis of the new solution is also provided.

Very interesting hints on how to implement FAN in real networks are given in [57]. Problems of scalability as well as the detection of flows and addressing the memory are discussed. The issues of protection and sizing the tables are also presented by the authors.

The detailed information on FAN and the implementation aspects are presented in several US patents filed by researchers from France Telecom. In [69], the method and the scheduler scheme for implicit differentiation of quality of service in a network are presented. The scheduling algorithm is based on SFQ with possibilities to prioritize the real-time application traffic. The second version of the scheduler, which may be used in FAN, is based on DRR and described in [81]. As in the previous solution, it it possible to send the traffic of streaming flows through the priority queue.

The FAN architecture is described and analyzed by authors of [52]. They present the overview of FAN and propose a new method for classifying the flows in the admission control block. It performs traffic control on the flow level and improves traffic performance in overload. The simulation analysis in the ns-2 environment shows that the proposed solution works efficiently.

The paper [93] reviews the QoS control architectures defined by the standard bodies such as Cable-Lab, DSL Forum, MSF (*Multiservice Switching Forum*), ETSI and ITU-T. The Flow-Aware Networking concept is presented as an example of flow level control architectures. Another flow-aware technology referred to as FSA (*Flow State Aware*), described in [1], divides the Internet services into several types with the requirements and control procedures.

A survey of established QoS architectures, including FAN, is presented in [51]. The authors also propose a new solution. The QoS management architecture: flow-aggregate-based services (FAbS) has two new blocks: inter-domain flow aggregation and endpoint implicit admission control. The possibility of aggregation of flows across the network domain is provided by implementing the IDFA (*Inter-Domain Flow Aggregation*) mechanism. The EIAC (*Endpoint Implicit Admission Control*) scheme eliminates inefficiency that results from discarding packets in the middle of the path of a flow by congestion notification to the edge nodes.

3.2 Flow-Aware Networking — new proposals

The FAN concept is very stimulating for many researchers. This, still a relatively new proposal for the future Internet is analyzed all over the world. New solutions have been proposed as extensions to FAN in recent years.

The flow aware traffic engineering approach for carrier class Ethernet networks is presented in [29]. The authors' proposal applies specifically to network architectures with traffic tunneling possibilities, e.g., Ethernet over MPLS (*Multi-Protocol Label Switching*). The flow blocking probability is analyzed and the explicit formula is given. A simple load balancing scheme for connections established through several paths is derived. The analysis presented in the paper is validated by simulation experiments.

The paper [57] presents the routing possibilities in the Internet. The flow-oblivious routing, used in current networks allows for realizing the end-to-end connections on multiple paths. It is assumed that in FAN each flow may be routed over one of a set of paths designated by the network. The adaptive multipath routing may be realized by Flow-Aware Networks. The flow is divided into subflows, all with different flow labels. Randomization is used to discover the different paths. Only the short paths should be chosen, so the authors propose to implement the trunk reservation to block subflows on the long routes. The proposed solution is presented as a pragmatic alternative to the commonly used solutions.

The analytic framework and simulation experiments for dimensioning the link capacity of broadband access networks are presented in [64]. The authors introduce a method to easy estimate the network bandwidth by considering the measures like the flow-level delay and packet loss probability. The numerical ex-

periments, provided by the authors, give a certain perspective on the design of the link capacity for access networks. The Flow-Aware Networks are considered as an example of the flow-awareness concept, which may be carefully adopted in access networks.

The authors of paper [48] propose the Static Router Configuration approach as a simple, adequate and feasible solution to the problem of insufficient resources for the emergency VoIP connections. In this concept, the classes of flows are manually defined by network administrators. The proposed solution should be used only locally, always to the nearest emergency center and allows for introducing differentiated blocking to FAN.

The analysis of FAN over WDM (*Wavelength Division Multiplexing*) environment is presented in [65]. The authors suggest that the backbone networks are migrating to the IP/WDM architectures. It their opinion, it is necessary to combine the resources of both layers to ensure efficient QoS of the traffic. The new architecture of the FAN node is proposed in the paper. The authors assume that in this overlayed networks, both optical and electronic resources are available to the FAN node. The policies to decide on using the available resources by the source traffic are described and discussed in terms of the goodput and queuing delay.

The possibilities of using FAN in a Grid environment are presented in [16]. The authors compare the performance of FAN and DiffServ architectures under the Grid environment by simulation experiments. The average transit delay and the average goodput of a Grid session in an IP access router are analyzed and compared. The results of the analysis, provided by the authors, show that FAN enables lower average access delays of GridFTP sessions than DiffServ. The strong degradation of the average access delays of the analyzed sessions were observed for FAN. On the other hand, the achievable goodput per Grid session is lower for FAN than for DiffServ. The analysis shows that FAN is a good proposal to use in the Grid environment.

The authors of [54] propose the QoS mechanisms for flow-based routers, which have to serve millions of flows. The presented solutions may be used not only in FAN, but almost in any flow-based architecture. The method, described in the paper, allows for dynamical prioritizing of traffic according to the characteristics of applications. It may be a good alternative to deep packet inspection and allows for conformity with the net neutrality concept. A calendar queue scheduler is proposed to use in FAN. It is a component of an efficient traffic management mechanism. The solutions proposed by the authors: RCFQ (*Real-time-Clock Fair Queuing*) and AFRED (*Adaptive Flow Random Early Drop*) for the guaranteed and non-guaranteed traffic, respectively, support QoS for flow-based routers.

The end-to-end transport architecture and a scalable control mechanism to support the various flow-level QoS requests from applications are presented in

[94]. The review of QoS architectures is given in the paper as well as the review of current flow-based control architectures. The FAN and FSA architectures are presented as examples of flow-based solutions. The authors propose to separate the transport and service functions of the networks. Moreover, for the scalability and functionality, the control mechanisms of the core and access networks should be designed independently. The proposed solution ensures end-to-end QoS assurance for flows with good scalability in the routers.

The author of [85] proposes new methods for routing the traffic in a network. He proposes to use micro-flows in contrast to packet switching. This work is related to FAN and has the same assumption that operating on flows rather than on packets enables fair bandwidth allocation with guaranties for the streaming traffic. L. G. Roberts proposes a new router architecture, which allows for flow-based transmission in a network. His solutions are presented in several US patents [90, 89, 88, 87, 86, 84].

3.3 Bandwidth sharing

The concept of statistical bandwidth sharing in IP networks is presented in [31, 79]. The simulation analysis and a mathematical model are presented to illustrate the essential properties of bandwidth sharing possibilities. The author of the second paper concludes his analysis with the statement that for fair bandwidth sharing with an acceptable performance, it is desirable to use admission control, like in FAN.

The analysis on evaluating the number of active flows in a scheduler realizing fair queuing is presented in [55, 56]. The authors of both papers show, by simulation experiments and an analytical analysis, that the number of active flows is usually very small, independently of the link capacity. Using the fair queuing algorithm in the scheduler, the complexity does not increase with the link capacity. The authors discuss the fair queuing algorithms, used also in FAN, and show that they are scalable and efficient.

The considerations on the buffer sizing issue are presented in [2, 3]. The authors highlight a very important problem: how to choose the proper size of the buffers. A widely used rule of thumb states that each link needs a buffer of size $B = \overline{RTT} \times C$, where RTT is the round trip time and C is link capacity. It means that for high capacity links we need large buffers. The authors suggest, due to the fact that the number of active flows does not increase with the capacity, that the buffer size should not be too large. They propose to divide the value obtained from the equation quoted above by the square root of the number of flows in progress. The method used in FAN to size the buffers of cross-protect routers is based on results of simulation experiments presented in the second paper.

3.4 Admission control

A nonintrusive TCP connection admission control method for bandwidth management of an Internet access link is presented in [61]. The proposed solution is based on link utilization policies. The SNMP (*Simple Network Management Protocol*) protocol is used for collecting useful network performance measures. The second functionality block, monitor machine, generates control commands to accept or block new flows. The authors present several measurement results that show the efficiency of their approach.

The measurement-based call admission control scheme proposed by M. Grossglauser and D. Tse in [38] performs as well as the optimal scheme, which has a full knowledge of network statistics. The idea of the given solution is similar to MBAC used in FAN. The admission control block should ensure the number of flows sufficiently low to maintain the overload probability below the desired threshold and the maximum bandwidth utilization. The proposed mechanism works based on the observed past history of recently served calls and active flows. The algorithm is quite complex. The authors conclude that there is a need to find a scheme which has a better performance under a high offered load.

A framework of MBAC with the detailed mathematical analysis is presented by the same authors in [97]. They show the impact of measurement errors on the QoS performance. Their research allows for identification of a crucial time-scale for which the effect of admission decisions persists. In the last part of the paper, it is shown how the memory time-scale of the estimators affects performance and how to choose it to achieve robust network performance.

The analysis of possible methods used for bandwidth sharing is presented in [83]. The authors claimed that the fair sharing, broadly used in current networks, may lead to performance degradation in the network. There is a need for a mechanism which limits the number of flows transmitting at the same time and maintains goodput in case of traffic overload. The main assumptions of the admission control mechanism used for Flow-Aware Networks are given in this paper.

The arguments for providing the admission control mechanism in current and future networks are presented in [11, 67]. The authors show that the retransmitted traffic under the TCP control may lead to performance degradation in some cases. Moreover, the incomplete transmissions, e.g., broken by applications, may also not necessarily decrease the throughput in network links. The admission control, proposed for using in flow aware architectures ensures complete transmissions of all accepted flows with an acceptable rate. The traffic, which need to be retransmitted is significantly lower when using the admission control mechanism.

The traffic theory for elastic and streaming flows is presented in [78]. The

author claimed that the admission control mechanism is essential for ensuring QoS in current and future networks. Integration of both traffic types with priority service to streaming packets looks to be an advantage of the proposed mechanism.

The analysis of an integrated admission control mechanism for two traffic types is presented in [5, 32]. The authors show the motivation for providing the admission control concept and present some implementation aspects. The mathematical and simulation analysis allows for estimating the proper values of the admission threshold. The results show that it should be on the level of a few percent of link capacity. The authors conclude that the choice of the threshold, while very important, is not crucial and the estimation of the available bandwidth does not need to be highly accurate.

The QoS aspects and the idea of admission control in the Internet are presented in [6]. Two types of flows, streaming and elastic, are described. It is shown that routers need admission control to ensure adequate throughput for both flow types and to avoid the retransmission of discarded packets. The integrated admission control scheme and its analytical model are described in details. The simulation analysis of the proposed solution is performed. The results show that it works satisfactorily.

The simulation and emulation experiments on two selective service protection mechanisms: DiffServ and implicit admission control are presented in the paper [7]. The results of the presented analysis show that both proposals ensure proper QoS assurance to the premium class traffic. The admission control mechanism allows for achieving greater network efficiency and a higher quality for the best effort traffic.

In the paper [49], the new concept of Measurement-Based Admission Control is presented. The new solution aims to provide possibly different statistical service guarantees to individual flows. The DPS (*Dynamic Priority Scheduling*) algorithm is proposed. It ensures that a newly admitted flow is always given a lower priority than all other flows. Moreover, its priority level is improved each time an existing flow leaves the system. The main part of the paper is the stochastic QoS analysis. A simulation experiment is also provided in the paper. The presented work needs further studies. It is expected to limit the maximum buffer size to bound the maximum packet delay. The set of some parameters needs to be evaluated to ensure high bandwidth utilization.

The proposal of using the admission control for Grid services in IP networks is presented in [15]. The efficiency of two architectures: Grid over DiffServ and Grid over FAN is compared by computer simulations. The results show that the average delay for a Grid session is lower in FAN networks. On the other hand, the DiffServ architecture provides better goodput.

A new scheme of admission control for flow-aware architectures is proposed in [75]. It works based on the fuzzy logic algorithm. The motivation to draw up a

new mechanism was to ensure the end-to-end guarantees for new services in future networks. The problems of admission control based on single link information, like in FAN, are discussed. The decisions of the admission control block may not be accurate and cannot transfer the network to a new state rapidly. The new solution, proposed by authors, makes a fast decision on the fuzzy link state and improves the network performance.

An implicit admission control scheme is also adopted at the routers realizing the QoS assurance for DiffServ networks. In paper [50], a mechanism of implicit admission control for Differentiated Services networks is proposed. Two versions of the authors' solution are presented: LBFA (*Link-Based Fair Aggregation*) for the deterministic EF (*Expedited Forwarding*) service and DPFA (*Dynamic Priority Based Flow Aggregation*) for the AF (*Assured Forwarding*) service. The simulation results obtained by the authors show that the new proposals do not affect the QoS guarantees of existing flows. As an effect of it, the authors conclude that the proposed admission control scheme should be used only for the new flows.

3.5 Congestion control and avoidance

The first papers on congestion control and congestion avoidance were presented in the eighties of the last century. The issue of ensuring a good performance in overload is very important and interesting for may researchers and practitioners. In this section a few selected papers, which present the aspects of congestion control and avoidance are described.

The main features of the congestion control concept are presented in [101]. The authors show the most important goals and advantages of using the congestion control mechanisms in packet networks. They also propose a new taxonomy, based on the control theory, for algorithms controlling the traffic in overload. It provides a coherent framework for a comparative study of the existing solutions.

The detailed description of the congestion control issue is presented in [46]. The author shows the main problems of the congestion control mechanisms, their classification and solutions. He also presents three own proposals: timeout-based congestion control, the DECbit scheme for congestion avoidance and the delay-based scheme for congestion avoidance.

In [45] the weaknesses of a few congestion control and avoidance algorithms used in high-speed networks are identified. The analysis and comparison of control ideas are provided. The author concludes that a single congestion scheme is not sufficient. It is a need for a combination of several schemes to ensure the complete congestion management in a network.

One of the proposals on how to implement congestion control in real networks is described in [33]. The authors present the TCP-friendly congestion

control algorithm in a system designed to deliver HDTV (*High Definition TV*) content over IP. It is important because it shows the problems of congestion control caused by the reordered packets of high-rate applications. The presented solution: the TFRC (*TCP-Friendly Rate Control*) protocol allows for regulating the throughput, but has some limitations, which lower its usefulness.

A very interesting proposal of congestion control mechanisms is presented in [66]. The sender functionality, packet loss control algorithm and receiver functionality are the parts of the proposed solution. The sender steers the rate of incoming traffic based on RTT (*Round Trip Time*) and retransmit timeout values. On the other hand, the receiver provides feedback to allow the sender to estimate RTT. The packet loss control mechanism decides which packets should be removed from the buffer in overload. The idea of this solution is similar to that used in the cross-protect router in FAN.

The deep analysis of congestion at the flow level is presented in [12]. The authors claim that the currently used congestion control mechanisms have scalability problems. There is a problem to guarantee a proper throughput for best effort traffic in overload. The impact of user behavior is very important in a congestion state. The authors conclude that the application of admission control is a more effective overload control than simply relying on user impatience in continuing the transmission.

An overview of optimization flow control for unicast and multicast transmission is presented in [13]. It presents the theory of selected solutions as well as the implementation issues. The paper shows the description and analytical analysis of a few concepts.

The book [100] presents all important aspects of congestion control. The principles of existing solutions as well as the experimental enhancements are deeply analyzed. The ISPs' perspective on the congestion problem is described. The future ideas of traffic control algorithms are mentioned. The simulation tools of overload and mechanisms to deal with it are also presented to show the problem in practice.

The congestion avoidance mechanisms allow for limiting the number of congestion events in computer networks. Their main goal is to control the traffic in a way that congestion does not occur.

The simulation results for four congestion avoidance schemes are presented in [40]. The mechanisms work based on the algorithms limiting the number of permits of flows. The best one, called SMAML (*Sharing with a Minimum Allocation and Maximum Limit*) also limits the number of permits a network node can use at a time.

The authors of [47, 76] present the congestion control and congestion avoidance issues. They also propose their own solution to work as a congestion avoidance mechanism. The concept of using one bit in the packet header for feedback

from the network to the users is provided. The deep analysis of the proposed solution is shown. The authors conclude that their system has been found to operate optimally.

The analysis of congestion avoidance mechanisms is presented in [19]. By analytical and simulation experiments, the authors claim that a simple additive increase and multiplicative decrease algorithm satisfies the sufficient conditions for convergence to an efficient and fair network.

The congestion avoidance mechanism RED (*Random Early Detection*) is presented in [30]. It detects when the congestion begins by estimating the average queue size. The congestion notification is made by dropping packets or by setting a bit in packets' headers. RED gateways keep the average queue size low enough which allows for queuing bursts of traffic occasionally arriving at the router. The proposed solution is effective, scalable and used in a new version of FAN presented in this dissertation.

Part II

Congestion control mechanisms for FAN

4 The algorithms of congestion control mechanisms for FAN

It is folly to use as one's guide in the selection of fundamental science the criterion of utility. Not because (scientists)... despise utility. But because. .. useful outcomes are best identified after the making of discoveries, rather than before.

— John C. Polanyi

The congestion control mechanisms for FAN, presented in [23] and [24], were proposed to decrease the access time to the congested link for new streaming flows that may represent, for example, VoIP connections. From the user point of view it is important to begin redialling immediately. It means that a user expects a quick acceptance time of his or her flow. The acceptable time for international calls should not be greater than 11 seconds for the 95% of the calls, while for the local calls it should not exceed 6 seconds [41]. In the basic version of the admission control algorithm, the new flows cannot be accepted in the XP routers under the state of congestion. It may cause that, in some cases, the whole bandwidth may be occupied by a finite number of flows giving no chance for new flows to begin their transmission for a long time. The new mechanisms based on the whole or partial cleaning of the PFL content in the congestion state allow for decreasing the acceptance time of new streaming flows. It is possible to choose such values of the parameters characteristic for the proposed solutions that allow for achieving a short acceptance time of new streaming flows

without significantly increasing the value of the mean transmission time of elastic flows. Four versions of congestion control mechanisms; EFM, RAEF, RBAEF and RPAEF are proposed and described below in details.

4.1 The Enhanced Flushing Mechanism

The EFM (*Enhanced Flushing Mechanism*), presented in [23] and [24], is a proposition to solve the problem of too long acceptance times of new streaming flows in the AC (*Admission Control*) block. The EFM is based on BFM (*Basic Flushing Mechanism*) [24]. In this original solution, the content of the PFL is deleted if congestion occurs. It is necessary to provide a time variable denoted by the authors of this work as *pfl_flushing_timer*. The value of this variable represents the minimum period of time that has to expire between any flushing actions. It ensures a stable operation of the algorithm. The BFM works properly, but has one very important drawback. The PFL is completely cleaned in overload, including the IDs of streaming flows. It may cause that transmission of such flows is broken. In most cases they are accepted in the AC block immediately, but sometimes it takes several seconds. This situation is unacceptable for VoIP calls. Thus, in the EFM the identifiers of only elastic flows are removed from the PFL if a packet of a new flow comes to the AC block in the congestion state (see Fig. 4.1). The elastic flows, which identifiers were removed from the PFL, are not blocked in the AC block and can resume the transmission promptly. After removing the identifiers of the elastic flows from the PFL, the flows have to compete with each other for acceptance in the AC block and it may take some time before they will be accepted again.

The pseudo code for realizing the EFM functionality in FAN is presented in Tab. 4.1.

Table 4.1: Pseudo code for realizing the EFM functionality in FAN

```
1.  on a new flow packet arrival in congestion
2.      current_time = Scheduler :: instance().clock()
3.      If current_time − flushing_start ≥ pfl_flushing_timer then
4.      begin
5.          For (i = 1; i ≤ pfl_size; i++) do
6.          begin
7.              If flow_bytes(i) ≥ MTU then
8.                  remove ID(i) from PFL
9.          end
10.         flushing_start = Scheduler :: instance().clock()
11.     end
```

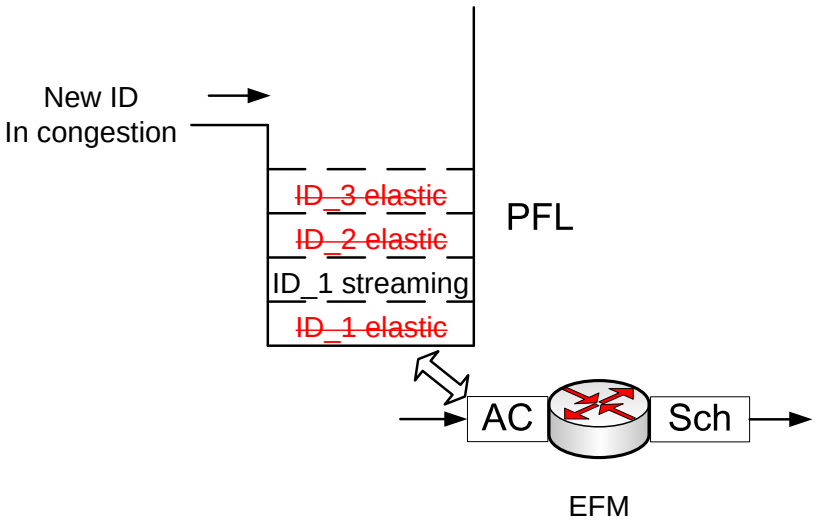

Figure 4.1: The operation principle of EFM

If a packet of a new flow arrives at the admission control block in congestion and the time period from the last flushing action is greater or equal to the value of *pfl_flushing_timer*, the Enhanced Flushing Mechanism is activated (lines 1-3). The *For* loop is executed for each flow, which identifier is written to the PFL (line 5). The ID of each elastic flow is then removed from the PFL (lines 6-9). A flow is selected as elastic if the number of bytes of its traffic is greater than or equal to the MTU. At the end of each flushing action, the value of the pointer, which indicates time of the last flushing action is set to the current time.

The implementation of the EFM in the cross-protect router is quite simple and does not increase the complexity and processing resources significantly.

4.2 The RAEF Mechanism

RAEF (*Remove Active Elastic Flows*) is the second mechanism proposed to solve the problem of the too long acceptance time of new streaming flows in the AC block. In this algorithm, the identifiers of all elastic flows being active for at least a specified time period (*active_time*) are removed from the PFL each time the congestion is noticed by a packet of a new flow (see Fig. 4.2). The flows which identifiers were removed from the PFL are not blocked in the AC block and can resume the transmission promptly, as in the EFM. The disadvantages of this algorithm are the same as in the EFM. It is possible that the identifiers of such

flows will not be added to the PFL again immediately or even in a short time and the transmission of their traffic may lengthen. The flows, which identifiers were removed from the PFL, have to compete with other flows for acceptance in the AC block and it may take some time before such flows will be accepted again.

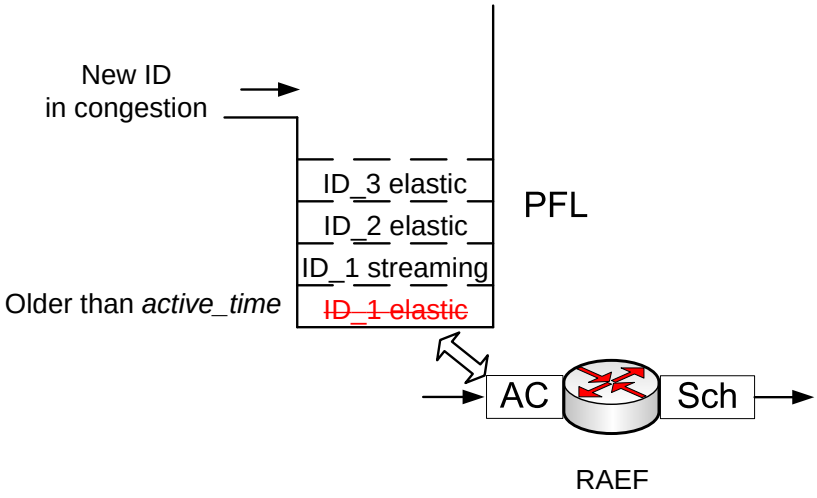

Figure 4.2: The operation principle of RAEF

The pseudo code for realizing the RAEF functionality in FAN is presented in Tab. 4.2.

Table 4.2: Pseudo code for realizing the RAEF functionality in FAN

```
1.  on a new flow packet arrival in congestion
2.     current_time = Scheduler :: instance().clock()
3.     For (i = 1; i ≤ pfl_size; i++) do
4.     begin
5.        active_time(i) = current_time − first_operation_time(i)
6.        If flow_bytes(i) ≥ MTU && active_time(i) ≥ active_time then
7.           remove ID(i) from PFL
8.     end
```

If a packet of new flow arrives at the admission control block the procedure of the RAEF mechanism is activated. The *For* loop is executed for each flow, which identifier is written to the PFL (line 3). If elastic flow i has transmitted its traffic for at least *active_time*, its flow ID is deleted from the PFL (lines 5-7).

A flow is selected as elastic if the number of bytes of its traffic is greater or equal to the MTU.

If a next packet of flow i arrives to the admission control block it may be accepted in a congestion-less state. Then its $first\_operation\_time$ is set to the *current_time* value. On the other hand, in congestion, it is dropped.

The implementation of the RAEF mechanism in the cross-protect router is simple and does not increase the complexity and power processing significantly.

4.3 The RBAEF Mechanism

The next mechanism, proposed in this dissertation to decrease the time interval between beginning of sending the packets by a new streaming flow and its acceptance in the AC block is called RBAEF (*Remove and Block Active Elastic Flows*). In this algorithm, the identifiers of the elastic flows being active for a specified period of time (*active_time*) are removed from the PFL every time a packet of a new flow arrives to the AC block in congestion. The identifiers of such flows are then written to the BFL (*Blocked Flow List*) for a short, fixed period of time (see Fig. 4.3). If a packet arriving to the admission control block belongs to the flow, the identifier of which is in the BFL, the packet is always dropped. Therefore, the flows removed from the PFL can continue transmission only after their tag has been removed from the BFL. The flows which identifiers were removed from the BFL, can continue transmission, but again, they have to compete with other flows for link resources and it may take some time before such flows will be accepted again.

The pseudo code for realizing the RBAEF functionality in FAN is presented in Tab. 4.3.

If a packet of a new flow arrives at the admission control block the procedure of the RBAEF mechanism starts. The *For* loop is executed for each flow, which identifier is written to the PFL (line 3). If elastic flow i is active for at least time value written to the *active_time* parameter, its flow ID is removed from the PFL and added to the BFL (lines 6-9). A flow is selected as elastic if the number of bytes of its traffic is greater than or equal to the MTU. At the end of each RBAEF action, the value of *blocked_start* of flow i is set to the current time (line 10).

If packet p of flow i arrives at the admission control block within the time less than the *blocked_period* it must be dropped (line 21). On the other hand, its ID is removed from the BFL and packet p is further processed (lines 16-19).

The implementation of the RBAEF mechanism in the cross-protect router is more complicated than in the previous case, but does not increase the complexity and processing resources significantly, either.

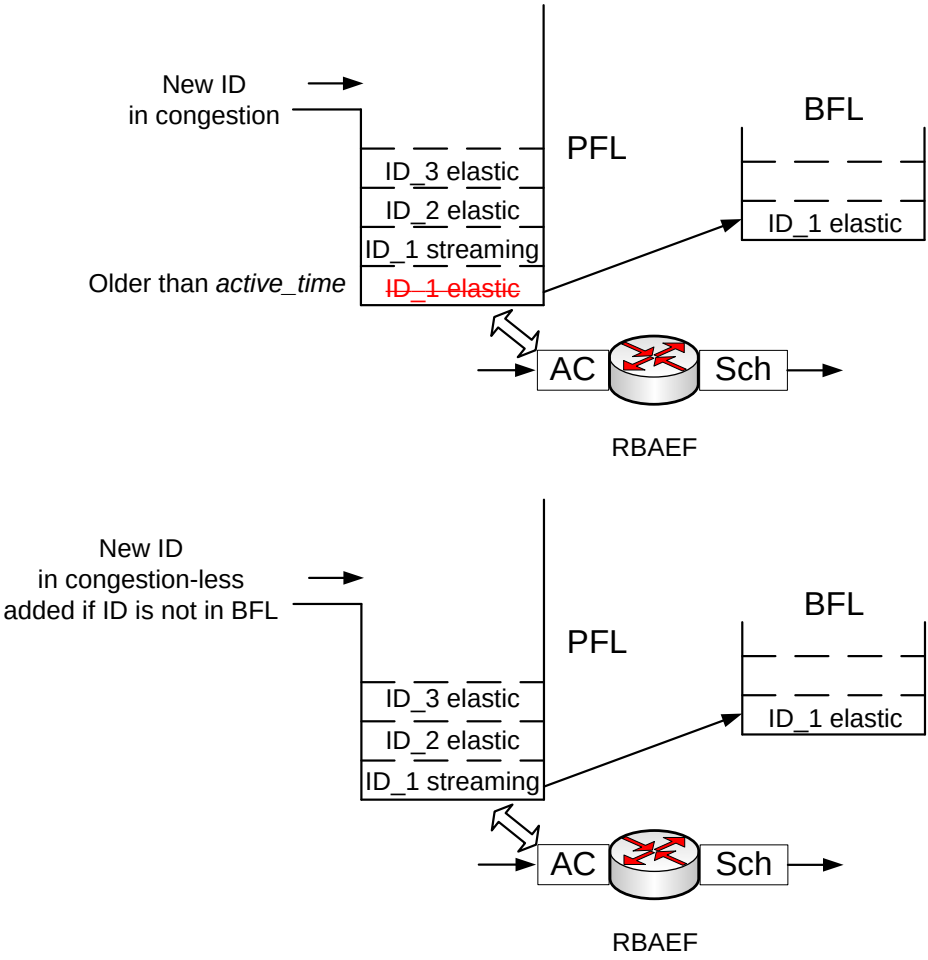

Figure 4.3: The operation principle of RBAEF

4.4 The RPAEF Mechanism

The last mechanism proposed in this work to decrease the acceptance time of a new streaming flow in the AC block is RPAEF (*Remove and Prioritize in access Active Elastic Flows*). In this solution, the identifiers of the elastic flows being active for a *active_time* period are removed from the PFL every time when a new flow wants to begin transmission in congestion. The identifiers of removed flows are then written to the PAFL (*Priority Access Flow List*) for

Table 4.3: Pseudo code for realizing the RBAEF functionality in FAN

```
1.  on a new flow packet arrival in congestion
2.     current_time = Scheduler :: instance().clock()
3.     For (i = 1; i ≤ pfl_size; i++) do
4.     begin
5.        active_time(i) = current_time − first_operation_time(i)
6.        If flow_bytes(i) ≥ MTU && active_time(i) ≥ active_time then
7.        begin
8.           remove ID(i) from PFL
9.           add ID(i) to BFL
10.          blocked_start(i) = current_time
11.       end
12.    end
**********************************************************************
13.    ********************admission decision**********************
14.    on arriving packet p of flow i
15.       current_time = Scheduler :: instance().clock()
16.       If current_time − blocked_start(i) > blocked_period then
17.       begin
18.          remove ID(i) from BFL
19.          proceed with packet p
20.       end
21.       Else drop p
22.    ************************end*****************************
**********************************************************************
```

a short *priority_access* time period (see Fig. 4.4). If a packet arriving at the admission control block in a congestion-less state belongs to the flow, the identifier of which is in the PAFL, the packet is always accepted. On the other hand, the packets of flows which identifiers are not in the PAFL are accepted with low probability P_{RPAEF}. The P_{RPAEF} probability is set to 1 if there is no ID in the PAFL. The idea of this solution is to ensure a quick acceptance time of new streaming flows and a short inactivity time of elastic flows which identifiers are removed from the PFL. Moreover, the proposed mechanism allows for decreasing the total number of all flows accepted after a cleaning action of the PFL content. This is a very important advantage of this algorithm. It ensures that the once accepted elastic flows have an opportunity to transmit their traffic with very short, harmless breaks and with an acceptable rate.

The pseudo code for realizing the RPAEF functionality in FAN is presented in Tab. 4.4.

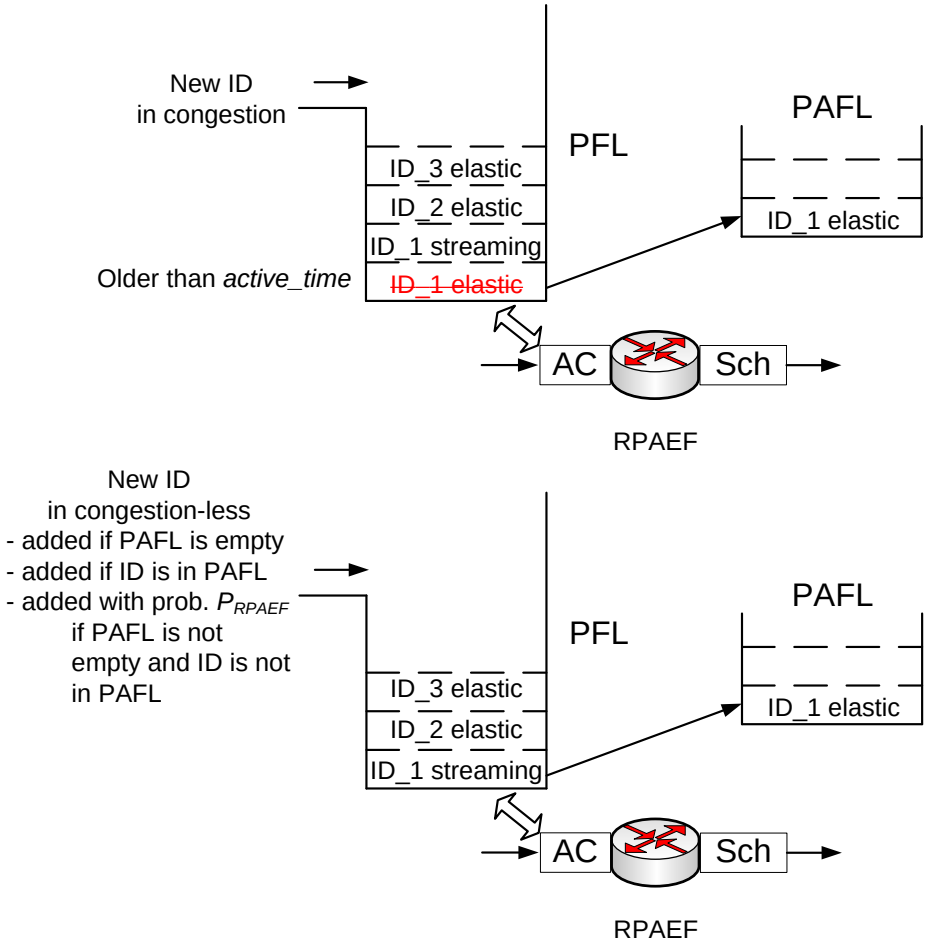

Figure 4.4: The operation principle of RPAEF

If a packet of a new flow arrives at the admission control block in congestion, the procedure of the RPAEF mechanism starts. The *For* loop is executed for each flow, which identifier is written to the PFL (line 3). If elastic flow i is active for at least the time value written to the *active_time* parameter, its flow ID is removed from the PFL and added to the PAFL (lines 5-9). A flow is selected as elastic if the number of bytes of its traffic is greater than or equal to the MTU. At the end of each RPAEF action, the value of $last\_RPAEF\_action$ is set to the current time (line 10).

Table 4.4: Pseudo code for realizing the RPAEF functionality in FAN

```
1.  on a new flow packet arrival in congestion
2.     current_time = Scheduler :: instance().clock()
3.     For (i = 1; i ≤ pfl_size; i++) do
4.     begin
5.        active_time(i) = current_time − first_operation_time(i)
6.        If flow_bytes(i) ≥ MTU && active_time(i) ≥ active_time then
7.        begin
8.           remove ID(i) from PFL
9.           add ID(i) to PAFL
10.          last_RPAEF_action = Scheduler :: instance().clock()
11.       end
12.    end
**********************************************************************
13.    ********************admission decision**********************
14.    on arriving packet p of flow i
15.       current_time = Scheduler :: instance().clock()
16.       If current_time − last_RPAEF_action > priority_access then
17.          clean PAFL content
18.       If PAFL is not empty then
19.       begin
20.          If i is in the PAFLthen
21.             proceed with packet p
22.          Else proceed with p with acceptance probability P_RPAEF
23.       end
24.       Else proceed with p
25.    *************************end*****************************
**********************************************************************
```

If packet p of flow i arrives at the admission control block after a time given by the *priority_access* parameter from the last cleaning action on the PFL, the content of the PAFL is cleaned and packet p may be accepted in a congestion-less state (lines 15-17 and line 24). If the PAFL is not empty, packet p is accepted if its flow ID is in the PAFL (lines 20-21). On the other hand, when the PAFL is not empty and the flow ID is not in the PAFL, the packet p may be accepted with a low P_{RPAEF} probability (line 22).

The implementation of the RPAEF mechanism in the cross-protect router is more complex than EFM or RAEF and the same as RBAEF, but does not increase the complexity and processing resources significantly, either.

5 Verification of the models

Chance favors the prepared mind.

— Louis Pasteur

Many simulation experiments were performed to experimentally validate the usefulness of congestion control mechanisms described in Chapter 4. Selected results are presented in this section. They are discussed and analyzed. As a background, the description of the simulation tool with new modules and the statistical credibility are presented.

5.1 Background

The simulation analysis is widely used in evaluation of mechanisms and systems used in telecommunications. The well prepared simulation experiment gives credible results, which may facilitate real implementations. It is very important to choose a good simulation tool and to set the proper values of needed parameters. Researchers have to select the type of simulation and technique for collecting data. The statistical analysis process also needs to be provided to authenticate the outcomes. The network model and topology used in the simulation runs and some additional assumptions are presented in the following sections.

5.1.1 Simulation Tool

The simulations were performed with the ns-2 simulator [68], version 2.33. This open-source software is a very popular environment for simulating the traffic per-

formance in packet networks. A combination of TCL scripts with modules written in C++ gives the possibility to implement almost every packet-switched architecture. The module of FAN for the ns-2 simulator, used by the author of this dissertation, was written by Łukasz Drzewiecki and Monika Antoniak-Lewandowska from TP R&D, Warsaw, Poland. They also created the Flow Simulator software realizing the FAN functionality, written in the C++ language [27]. The code of FAN functionality for the ns-2 simulator had a few errors which were corrected by Marcin Nossowski from the same team as its authors. In this improved version, the functionality of admission control was added and the method for estimating values of *fair_rate* was corrected. The stable version of FAN was developed only for the PFQ scheduling algorithm. The first part of work on this dissertation was to correct errors in the code for FAN with the PDRR algorithm. The methods for estimating the *fair_rate* and *priority_load* as well as the schemes for enqueuing and dequeuing the packets were changed. The functionality of admission control was also added to this version of FAN. All the simulation runs were provided with two stable and fully implemented versions of FAN.

The obtained results in each simulation run were adapted in the *Perl* language [72] to the Microsoft Excel needs. 95% confidence intervals were calculated by using the Student's t-distribution. The statistical estimation of the expected values and confidence intervals for examined variables were made by using Microsoft Excel. The plots were drawn by using the *gnuplot* software [34].

5.1.2 Simulation type and technique for data collection

The discrete event simulation (DES) type simulation was used [4]. The events are managed in time in such simulation experiments. The queue of events is maintained by the simulator and the events are sorted by the simulated time they should occur. The simulator executes the events one by one. There is no need to execute the simulation in real time. It is important to be able to access the data produced by the simulation and analyze them.

The data are collected as results of each simulation run, which is repeated ten times. It allows for estimating the statistical values of measured parameters. The technique for collecting simulation data used in the research is *the independent replication method* [98].

5.1.3 Simulation parameters

It is an important problem how to choose the proper size of the buffers. In a most popular method, each link needs buffer of size $B = \overline{RTT} \times C$, where B is the buffer size, RTT is the round trip time, C is link capacity. This solution may be used for relatively slow links. Due to the fact that the number of active flows does not increase with the capacity, there is no need to set the buffer size to too

large value. In high capacity links the buffer size should be estimated from the following formula:

$$B = \overline{RTT} \times C/\sqrt{n} \tag{5.1}$$

where n is the number of flows in progress.

Based on simulation parameters presented in [3], the buffer size for a 50 Mbit/s link should be set to the value of 100 packets. The buffer is sized to absorb approximately one delay bandwidth product. For a 100 Mbit/s link (used in simulation runs of this dissertation) the reasonable value is 1000 packets. It allows for low packet delays and jitter.

Two types of traffic are considered during the simulation experiments. A single FAN link with many source nodes and many destination nodes was considered. For calculating the volume of TCP traffic to be sent by each of the elastic flows in the FAN link with capacity equal to 100 Mbit/s and 1 ms delay, the traffic pattern with Pareto distribution was provided:

$$F(s) = Pr[size > s] = (k/s)^{\gamma} \tag{5.2}$$

where k is a minimum size and $1 < \gamma \leq 2$. The characteristic of the Pareto distribution is that most flows are very small (mice flows) while most of the traffic is contained in large flows (elephant flows). At the beginning, the link is empty and fed by the exponentially distributed arrivals of elastic TCP flows with the average value of size $E(s)$=150 Mbytes and $\gamma = 1.5$. The elastic flows are expected to occupy the bandwidth for a long time giving small chances for new flows for being accepted in the admission control block. The exponential distribution was also used to generate the time intervals between beginnings of the transmissions by the streaming flows, which arrive with the mean intensity equal to 0.1 (ten per second). The packet size (100 bytes) and the transmission rate (80 kbit/s) used for the streaming flows are typical values of the VoIP stream transmission, e.g., in Skype. The simulation experiments were provided in various conditions changing the number of elastic and streaming flows.

The *fair_rate* and *priority_load* estimates are calculated periodically. The choice of the interval length in both cases is not highly critical. Considering the time scales of the respective congestion phenomena, the measuring interval for the *priority_load* parameter should be set to several or a few tens of miliseconds (the value of 50 ms was used in the simulation experiments). The interval of several hundred miliseconds is sufficient for the *fair_rate* estimation (500 ms was used). The *max_priority_load* (the maximum allowed value of the priority load) and the *min_fair_rate* (the minimum allowed value of the fair rate) parameters were set to 70% and 5% of the link capacity, respectively. This way, a minimum bandwidth for elastic flows is guaranteed, and the load induced by the streaming

flows is controlled. The *pfl_flow_timeout* parameter was set to 20 s, what means that the identifier of a flow being inactive for at least 20 s is removed from the PFL.

The number of enqueued, dequeued and dropped packets in the scheduler block was collected during the simulation runs. The values of *priority_load* and *fair_rate* were estimated periodically and stored for further analysis. The number of active flows written to the PFL was also observed.

5.1.4 Simulation credibility

Several issues have to be considered to achieve valuable results of the experiments. First of all, the credibility of the random number generator (RNG) used in the simulator should be confirmed. The second issue is to assess the length of the simulation warm-up period properly, that is to determine the beginning of the steady state of the simulation.

Random number generator in ns-2

The random generator class (RNG) is used to determine the random values. The implementation of it combines multiple recursive generator MRG32k3a proposed by L'Ecuyer [63]. This generator provides $1.8x10^{19}$ independent streams of random numbers, each of which consists of $2.3x10^{15}$ substreams. Each substream has a period (i.e., the number of random numbers before overlap) of $7.6x10^{22}$. The period of the entire generator is $3.1x10^{57}$. Each random variable used in simulation should use a separate RNG object, which is automatically seeded to the beginning of the next independent stream of random numbers. It means that the implementation allows for a maximum of $1.8x10^{19}$ random variables.

It is important to set the proper values of the seed for each generator instance. To get the non-deterministic behavior, the seed value may be set to 0. It sets the seed based on the current time. This method should not be used in the independent replications method. It does not give any guarantees that the values will not overlap. The substream capability provided by the RNG generator ought to be used in this case. This method was used in all simulation runs, the results of which are presented in this dissertation. The implementation method of this solution in the TCL script is presented in the Appendix A.

Warm-up period

All observations of simulation parameters should be provided in a steady-state. The setting of the appropriate value of the warm-up period is the key element for the credibility of simulation results. There are many methods for estimating the value of this parameter [71]. One of them is based on the assumption that

the values of the observed variable stops moving in one direction and starts oscillating. In another, the steady-state is noticed at point, which value is neither minimum nor maximum of the rest of observations. The *moving average* method, based on the second approach, is used to determine the warm-up time in the simulation experiments presented in this work. The estimator $\hat{\mu}(n,k)$ denotes output sequences and is given by the following formula:

$$\begin{cases} \hat{\mu}(n,k) = (2k+1)^{-1} \sum_{i=-k}^{k} \hat{\mu}_{n+k} & \text{if } n \geq k+1 \\ \hat{\mu}(n,k) = (2n-1)^{-1} \sum_{-n-1}^{n-1} \hat{\mu}_{n+k} & \text{if } n < k+1 \end{cases} \tag{5.3}$$

The k value should be chosen carefully. The value of $k = 10$ seems to be a good choice. The value of the warm-up period was estimated based on the *fair_rate* values estimated in function of time. This is a key parameter for noticing the congestion in FAN. The simulation experiment for estimating the warm-up period was executed 50 times for each FAN architecture. While the *fair_rate* was estimated periodically after each dequeuing operation, its values were written at random time instants. Thus, it was impossible to draw a mean graph of its values. The values of the warm-up period in all simulation runs are summarized and divided by the number of simulation runs (the arithmetic average). The obtained results are presented in Tab. 5.1.

Table 5.1: Values of warm-up period in various FAN architectures

FAN with PFQ [s]	**FAN with PDRR** [s]	**AFAN** [s]
8.27	7.21	5.05

The results show that the warm-up time period is relatively short in each FAN architecture. The finally assumed value of this parameter used in the simulation experiments was set to 20 s. It ensures that the results of each simulation run were analyzed in a steady-state.

Simulation topology

Most of simulation experiments were provided for a single FAN link with many source and destination nodes. The topology is presented in Fig. 5.1. While the congestion control mechanisms were analyzed, there was no need to provide a more complex topology. The acceptance time of new streaming flows and the mean transmission time of elastic flows is a local problem and may be analyzed in such a simple network.

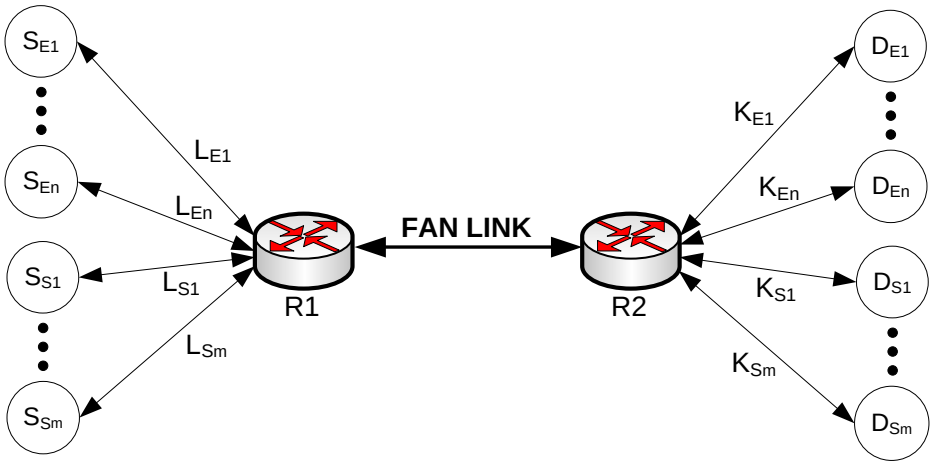

Figure 5.1: The basic simulation topology

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 of the elastic and streaming traffic, respectively.

5.2 New methods for estimating the congestion indicators

The methods for estimating the congestion indicators, *fair_rate* and *priority_load*, are presented in Chapter 2. The implementation of solutions estimating the *fair_rate* values, in both FAN architectures, in real devices may by difficult and complex. Moreover, *priority_load* represents the offered load of the priority traffic, while in fact it should give the information on the bandwidth used by outgoing high priority packets. These drawbacks can be rectified. In this section, new methods for calculating the values of congestion control indicators are proposed and evaluated by simulation experiments.

5.2.1 The *fair_rate* parameter

The *fair_rate* computation is based on adding a constantly backlogged fictitious flow to a system. The maximum transmission rate which is available to that flow estimates the *fair_rate*. This mechanism is used in both well known FAN versions. While this method is correct for estimating such a parameter, it causes

that a packet processing in the cross-protect router is unnecessarily complex. A new method for calculating the approximated value of the *fair_rate*, for both FAN versions, in a simpler way is proposed in this section. The values of the parameter may be calculated from the following formula:

$$fair\_rate = \frac{max\{S \times C, FB \times 8\}}{t_2 - t_1} \tag{5.4}$$

where FB is the number of bytes sent by elastic flows during the time interval (t_1,t_2), measured in seconds, divided by the number of elastic flows, which identifiers are written to the PFL, S is the total length of inactivity in the transmission during measuring time period, C is the link bit rate. The flow is classified as elastic if its enqueued number of bytes is greater than the maximum allowed transfer unit (in bytes). It is easy to count the number of bytes transmitted by elastic flows during the measuring time period and to get the number of elastic flows from the PFL. The *fair_rate* calculated in this way does not represent the real rate realized by elastic flows but estimates it with a precision sufficient to decide on congestion.

The pseudo code for measuring *fair_rate* in FAN with PFQ or PDRR is presented in Tab. 5.2.

If the buffer occupation by flow F is greater than or equal to MTU then this flow is classified as elastic and its packet's size is added to the value of $fair\_bytes$ (lines 2-3). The implementation of the method for computing the inactivity time is presented in the table between lines 4 and 14. The counter of *idle_time* is incremented if there are no packets of elastic flows ready to be sent from the buffer. If, during the dequeuing operation, the *idle_measured* pointer is set to the *true* value, it is changed to $false$ and if the time of dequeuing is greater than the time of beginning of measuring *idle_time*, the *idle_time* is incremented by the time period from the beginning of measuring the *idle_time* value to the current time (lines 4-8). If, after these operations, the buffer is empty the *idle_measured* pointer is set to the *true* value and the *idle_period_start* is set to the value of the current time instant incremented by the packet size in bits divided by the link capacity (lines 10-13).

The procedures of the function for calculating the *fair_rate* values are presented between lines 16 and 32. For simplicity, the declarations of parameters are omitted. The number of elastic flows is written to the variable *length* (line 17). The function *get_nb_elastic*() counts the flows, which have the content in the buffer greater than or equal to the MTU. If the time interval from the beginning of computing *fair_rate* is greater than or equal to the computation interval (500 ms) the value of *fair_rate* is estimated (line 19). The higher value of *numerator* is selected for further analysis. The value of *numerator*1 parameter is equal to the number of bits of dequeued elastic packets, while *numerator*2

Table 5.2: Pseudo code for measuring *fair_rate* in FAN architecture with PFQ or PDRR

```
1.  after dequeuing of packet p of flow F
2.     If enqueued_bytes(F) ≥ MTU then
3.        fair_bytes+=packet_size(p)
4.     If idle_measured == true then
5.     begin
6.        If Scheduler :: instance().clock() > idle_period_start then
7.           idle_time+=Scheduler :: instance().clock() − idle_period_start
8.        idle_measured = false
9 .    end
10.    If buffer_length == 0 && idle_measured == false then
11.    begin
12.       idle_period_start = Scheduler :: instance().clock() + 8 × L/C
13.       idle_measured = true
14.    end
15.    compute_FR()
**************************************************************************
16.    **********compute_FR()**********
17.       length = get_nb_elastic()
18.       current_time = Scheduler :: instance().clock()
19.       If actual_time − start_FR_interval ≥ FR_interval_length then
20.       begin
21.          numerator1 = fair_bytes × 8/length
22.          numerator2 = idle_time × C
23.          If numerator1 ≥ numerator2 then
24.             numerator = numerator1
25.          Else numerator = numerator2
26.          fair_rate = (numerator/(current_time − start_FR_interval))
27.          idle_time = 0
28.          fair_bytes = 0
29.          idle_period_start = Scheduler :: instance().clock()
30.          start_FR_interval = Scheduler :: instance().clock()
31.       end
32.    **********end**********
**************************************************************************
```

represents the number of bits, which might be sent during the inactivity time (lines 21-25). The *fair_rate* value is computed as a quotient of the numerator and measurement interval (line 26). After estimating the value of *fair_rate* the parameters used in the function are set to their initial values (lines 27-30).

5.2.2 The *priority_load* parameter

In both basic FAN versions, the *priority_load* parameter signifies the rate of priority packets incoming to the router with reference to the link capacity. The method for estimating this parameter would be more adequate, if *priority_load* is represented by the outgoing rate of the priority packets with reference to the link capacity. In this way, the bandwidth occupation by streaming flows (with priority) may be estimated, what allows for indicating the congestion if this value is too large. The new method looks to be more elegant and, in practice, there is almost no difference in operation of FAN.

The pseudo code for measuring *priority_load* in FAN with PFQ or PDRR is presented in Tab. 5.3.

Table 5.3: Pseudo code for measuring *priority_load* in FAN architecture with PFQ or PDRR

```
1.  after dequeuing the priority packet p of flow F
2.      priority_bytes+ = packet_size(p)
3.      compute_PL()
*************************************************************************
4.  **********compute_PL()**********
5.      current_time = Scheduler :: instance().clock()
6.      interval_length = actual_time − start_PL_interval
7.      If interval_length ≥ PL_interval_length then
8.      begin
9.        priority_load = (priority_bytes × 8)/(interval_length × C)
10.          start_PL_interval = Scheduler :: instance().clock()
11.          priority_bytes = 0
12.       end
13.   **********end**********
*************************************************************************
```

The size of the dequeued packet classified as priority is added to the value of *priority_bytes* (line 2). The procedures of the function for calculating the *priority_load* values are presented between lines 4 and 13. For simplicity, the declarations of parameters are omitted. The *interval_length* is calculated as difference of the current time and the start time of the interval (line 6). If this value is greater than or equal to the computation interval (50 ms) the value of priority_load is estimated (line 7). The *priority_load* value is computed as a quotient of the number of priority bits and the measurement interval multiplied by the link capacity (line 9). After estimating the value of *priority_load* the parameters used in the function are set to their initial values (lines 10-11).

5.2.3 Simulation analysis

In this section the results of carefully selected simulation experiments run in the ns-2 simulator are presented. They show that the new methods for estimating the *fair_rate* and the *priority_load* work as expected in both FAN versions.

Experiment 1

Firstly, 240 simulation runs (60 for each FAN version with a different method for calculating the *fair_rate*) in various conditions were executed to show the mean deviation from the *min_fair_rate* value in function of the smoothing parameter (see Fig. 5.2). The traffic pattern with Pareto distribution for calculating the volume of traffic to be sent by each of 200 flows was used in the FAN link with the capacity equal to 100 Mbit/s. The exponential distribution for generating the time intervals between beginnings of the transmissions of the flows was provided. The duration of each simulation run was set to 1200 s. The measurement interval for the *priority_load* parameter was set to 50 ms while the *fair_rate* values were estimated every 500 ms. The *max_priority_load* parameter was set to 70%, the *min_fair_rate* parameter was set to 5% and the warm-up period was set to 20 s. 95% confidence intervals were calculated by using the Student's t-distribution.

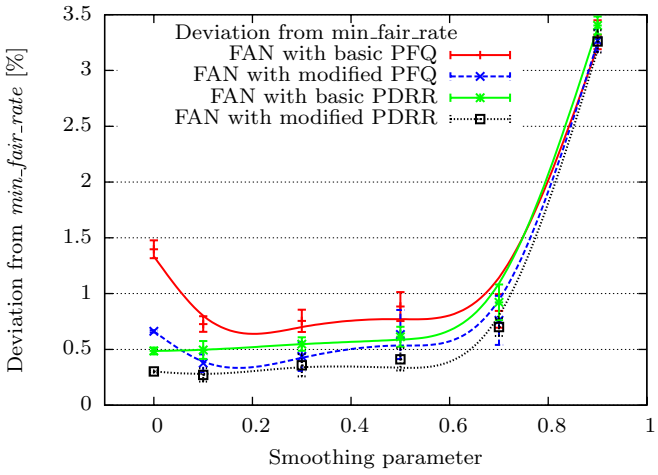

Figure 5.2: The mean deviation from *min_fair_rate*

The mean values of deviation from the *min_fair_rate* are estimated as a part of the link capacity. The values of *fair_rate* change in time and oscillate around *min_fair_rate*. The situation is better if the deviations from *min_fair_rate* are

lower. We can see that for FAN with PDRR the analyzed mean deviation is lower than for FAN with PFQ. The new method for estimating the *fair_rate* parameter allows for decreasing the mean values of deviations in both FAN versions. The mean values of the *fair_rate* may be approximately calculated as the difference between the *min_fair_rate* and the mean deviation of *fair_rate*. Of course, if the mean deviation of *fair_rate* decreases the mean *fair_rate* increases. The new method for estimating the *fair_rate* ensure a lower mean deviation from the *min_fair_rate* and, at the same time, the higher mean *fair_rate* value. It means that the proposed solution ensures more stable transmission in FAN.

The values of the mean deviation from the *min_fair_rate* change in function of the smoothing parameter used for estimating the values of *fair_rate* (see Chapter 2). We can observe that, for FAN with PFQ, the values of the estimated deviation are most suitable in the range from 0.1 to 0.6 of the values of the smoothing parameter. In this range, the value of the deviation from *min_fair_rate* increases insignificantly. For FAN with PDRR the appropriate range of the smoothing parameter is even wider. In this FAN version it is not necessary to use the smoothing parameter. In FAN with PFQ the best results are observed for relatively small values of the smoothing parameter.

Experiment 2 shows the analysis of a FAN link subject to failure. The results using two methods for computing *priority_load* are presented.

Experiment 2

In this simulation experiment the analysis of an impact of a failure in the same FAN link as in the previous case is presented. The traffic load is increased by adding 800 flows sending the UDP traffic to show the weakness of the basic method for estimating the *priority_load* parameter after repairing a significantly overloaded link. The exponential distribution was used to generate the start of the transmission time of these flows. The packet size was set to 200 bytes and the transmission rate of each flow to 500 kbit/s. At 100 s the link was turned off and at 125 s turned on again. The main goal of this simulation experiment was to show the values of *priority_load* in situation when many flows are accepted in the PFL in a short time. We have to notice that, in FAN, the first packet of each new flow accepted in the MBAC is always treated as priority. It means that, after turning on the repaired link, the number of priority packets accepted for enqueuing is very high.

We can see that, if we use the basic method for calculating the *priority_load* in both FAN versions, the maximum observed values of this parameter decrease linearly with increasing values of the smoothing parameter (see Fig. 5.3). The most important point to notice is that if we do not use smoothing (or for low values of the smoothing parameter) the value of *priority_load* is over one what

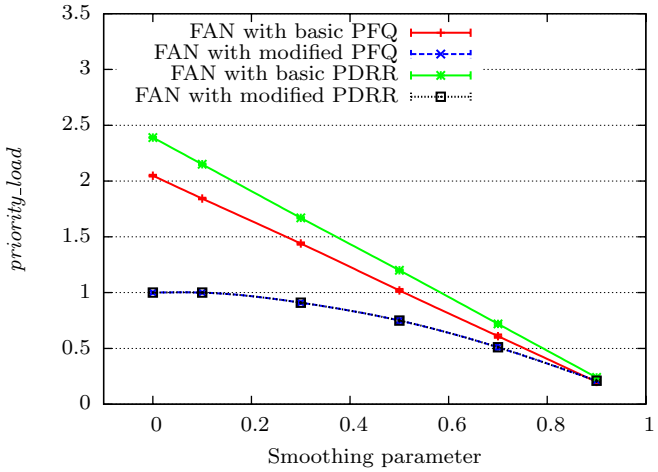

Figure 5.3: The maximum values of *priority_load*

suggests that the priority traffic in the link exceeds its capacity. If we use the new method for estimating *priority_load* its values are never higher than one. When using mild smoothing and without it the priority traffic consumes all available bandwidth. It is more proper if the maximum observed value of *priority_load* in this case is equal to one.

The new method for estimating the values of *priority_load* does not affect the transmission in the network. The number of flows accepted in MBAC was the same and the acceptance time of them did not change during the simulation experiment. In most cases *fair_rate* is the parameter which decides on congestion notification. Only in links with a large number of flows with rates low enough, the *priority_load* is significant in stating the overload status. The values of *priority_load* obtained while using FAN with the basic method for estimating it differ, even though the definition of it is the same in both versions, with PFQ and PDRR (see Fig. 5.3). In FAN with PFQ the congestion is stated earlier than in FAN with PDRR and the number of queued incoming packets is lower. In this case, the values of *priority_load* are lower for FAN with PFQ. If we use the new method for estimating the *priority_load* parameter the values of it are always less than or equal to one. As we see in Fig. 5.3, they do not differ in both FAN versions because the number of priority packets dequeued in the measurement time period is the same independently of the FAN version.

Conclusion for Experiments 1 and 2

The new methods for estimating *priority_load* and *fair_rate* in the scheduling block of the cross-protect router, proposed and described in this section, allow for a more stable transmission in FAN. There is no need to use complex methods for calculating the congestion control parameters. The simulation analysis shows that these simple methods give better results.

The basic method for calculating the *fair_rate* uses the fictitious flow to estimate the traffic, which may be sent. The solution proposed in this section allows for calculating the *fair_rate* based on the amount of traffic sent by elastic flows realizing the best effort transmission. The results of the simulation experiments show that the new proposal allows for reducing the oscillations around the *min_fair_rate* value and for decreasing the mean deviation from it.

The basic method for estimating the *priority_load* is based on analyzing the incoming traffic to the router. In the solution proposed in this section the values of *priority_load* are calculated as a part of total bandwidth occupation by priority traffic. It allows for a more realistic representation of the traffic served in the routers and does not change its characteristics.

5.3 Analysis of new congestion control mechanisms for FAN

Results of simulation experiments which show the usefulness of congestion control mechanisms for FAN are presented in this section. The advantages and drawbacks of the proposed solutions are described. The mechanisms are separately analyzed in two FAN architectures, with PFQ and PDRR. At the end of this section they are also compared and their usefulness in case of link failure is highlighted.

Experiment 3

The first version of flushing mechanism considered by the author is called BFM (*Basic Flushing Mechanism*). In this proposal, the identifiers of all flows are removed from the PFL. This solution has a one major drawback. Completely cleaning of the PFL content causes that both elastic and streaming flows are rejected and their further transmission may be continued only after repeated acceptance in the AC block. It may cause that, e.g., a VoIP call may be interrupted or even rejected. It is a very uncomfortable situation for users and may cause instability in network operation. 100 simulation runs were made for a FAN link with 20 streaming flows and the number of elastic flows ranging from 200 to 600. The first streaming flow began its transmission at 50 s and the rest of streaming

flows began their transmission one by one. The exponential distribution for generating the time intervals between beginnings of the transmissions by those flows was used. The values of the smoothing parameters for estimating the *fair_rate* were set to 0.1 in both FAN architectures. The smoothing was not used when computing the *priority_load* values. The rest of the simulation parameters were set as in the previous cases. The mean values of break in transmission of streaming flows in function of the number of elastic flows active in the background are presented in Fig. 5.4.

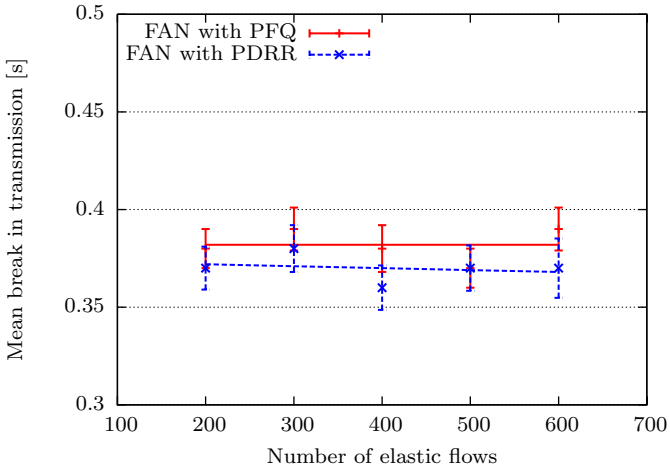

Figure 5.4: The mean break time in transmission of streaming flows in BFM

We can see that the breaks in transmission of streaming flows are independent of the number of elastic flows being active in the background. The breaks may be troublesome for users watching a movie or calling another user. Even if they are relatively short, they are not acceptable by a user. This drawback was a stimulus to look for a more suitable solution. In the EFM (*Enhanced Flushing Mechanism*) only identifiers of elastic flows are removed from the PFL. It means that if a streaming flow begins transmission it will not be broken by the mechanism.

5.3.1 Simulation experiments on EFM

By the following simulation experiments it is shown how the EFM may improve the network performance in the congestion state.

Experiment 4

400 simulation runs were made in various conditions to show the time of acceptance of a new streaming flow (*waiting_time*) in the AC block in a FAN link. The duration of each simulation run was set to 250 s, which allowed for observing the acceptance time of flows. The other simulation parameters were set as in the previous simulation experiment. The mean values of *waiting_time* in function of the number of elastic flows active in background for FAN with PFQ or with PDRR are presented in Fig. 5.5 and Fig. 5.6, respectively.

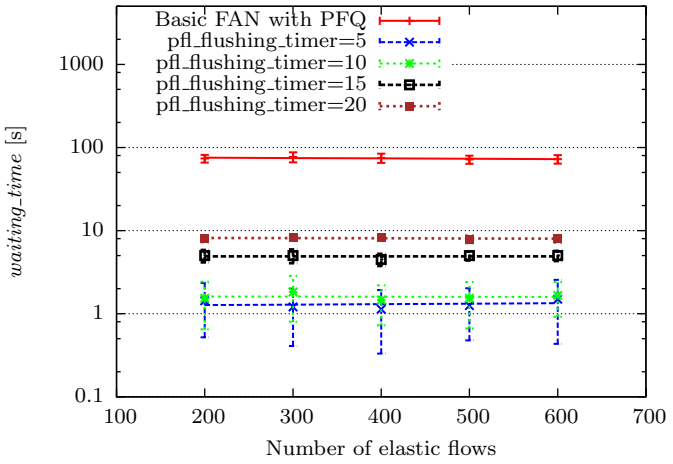

Figure 5.5: The mean *waiting_time* in FAN with PFQ and EFM

We can see that the values of the observed parameter are constant for each *pfl_flushing_timer* value in both FAN architectures. It means that a new streaming flow is accepted in the AC block at the same time for particular values of the *pfl_flushing_timer* parameter, independently of the number of elastic flows that want to send the traffic. Without using the flushing mechanism the *waiting_time* values are much greater than those in the cases when the flushing mechanism (EFM) is implemented. The *waiting_time* values increase with increasing values of the *pfl_flushing_timer* parameter in the examined range. The results for both FAN architectures are similar. As one can see, the best results were obtained when the *pfl_flushing_timer* was set to 5 s. The simulation results show that in this case a new flow was always accepted within 2 s from the time when it began to send the traffic. In the cases when the *pfl_flushing_timer* was set to 10 s or 15 s a new flow was accepted a few seconds later, but the obtained results are still acceptable for local voice calls [41]. In the case when the *pfl_flushing_timer*

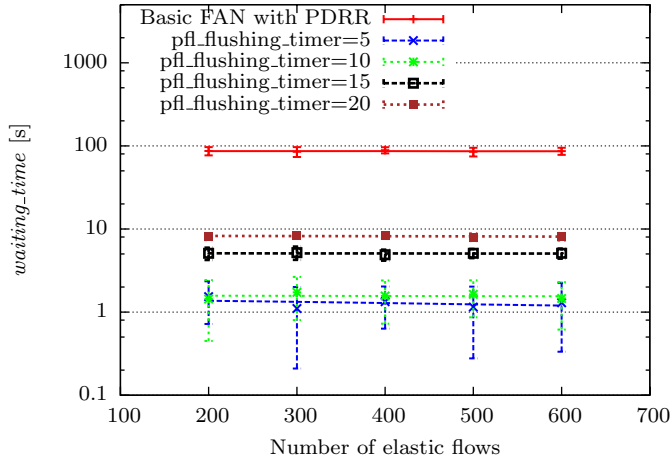

Figure 5.6: The mean *waiting_time* in FAN with PDRR and EFM

was set to 20 s the obtained results show that a new streaming flow was accepted after approximately 8 s, which is not acceptable for local calls, but low enough for international calls [41].

Experiment 5

After almost every cleaning action the number of elastic flows accepted again is greater than in the basic FAN. In this simulation experiment, the mean number of elastic flows in FAN with PFQ or PDRR was estimated. The simulation duration was set to 500 s, which allows for calculating the values of the analyzed parameter in the steady state for a sufficiently long time. The rest of the simulation parameters were set as in the previous case.

The values of the mean number of elastic flows in function of the number of elastic flows active in background for FAN with PFQ or with PDRR are presented in Fig. 5.7 and Fig. 5.8, respectively.

The mean number of elastic flows accepted in the router after any flushing action increases with the increasing number of elastic flows which want to begin or continue transmission in the background. In FAN with PFQ the number of accepted flows after flushing is significantly higher than in FAN with PDRR. Unfortunately, in both cases, this number is much more higher than in the basic FAN, where the mean number of accepted elastic flows is almost independent of the number of all elastic flows which want to transmit their packets. This a

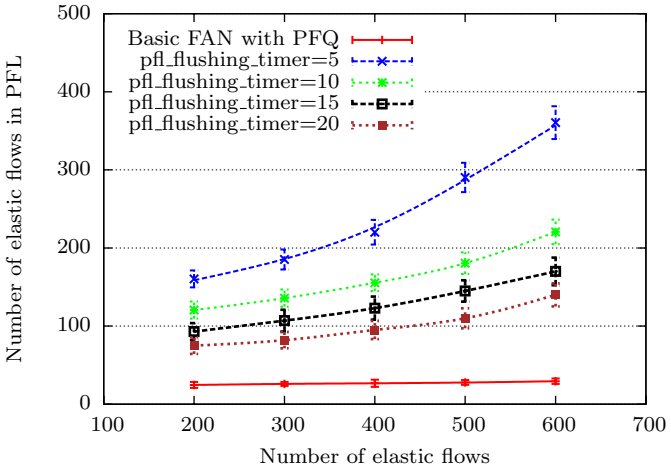

Figure 5.7: The mean number of elastic flows in PFL in FAN with PFQ and EFM

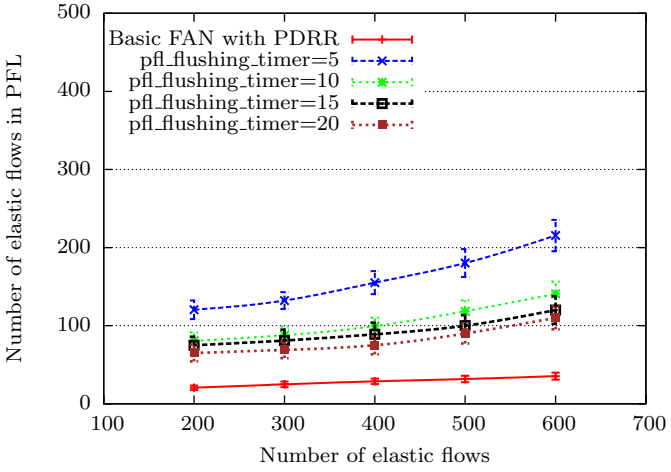

Figure 5.8: The mean number of elastic flows in PFL in FAN with PDRR and EFM

strong drawback of the EFM and shows that this solution may be not scalable, especially for low values of the *pfl_flushing_timer* parameter.

Experiment 6

The main drawback of EFM is that after any flushing action the transmission of elastic flows is broken. The rejected flows may continue to send packets only after renewed acceptance in the AC block what lengthens their transmission time. In this simulation experiment, it was checked how a decrease of the values of the *waiting_time* parameter for new streaming flows affects the mean transmission time of the elastic flows. The duration of each simulation run was set to 1500 s, which allowed for observing the beginning and finishing time instants of each elastic flow. The rest of the simulation parameters was set as in the previous case.

The mean time of transmission of the elastic flows in the FAN link with PFQ is equal to 123.87±16.07 s, which means that the transmission time period of the elastic flows usually lasts from 107.80 s to 139.94 s. The mean value of this parameter in FAN with PDRR is 125.16±3.69 s. We can see that in both FAN architectures the mean transmission time of elastic flows is similar. The values of the transmission time of the elastic flows in function of *pfl_flushing_timer* for the case with 200 elastic flows active in background for both FAN architectures with EFM are presented in Fig. 5.9.

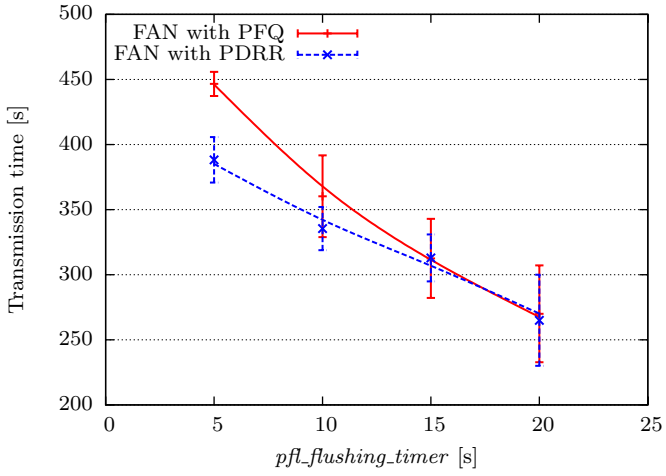

Figure 5.9: The mean transmission time of elastic flows in FAN with EFM

As we can see, the values of the transmission time period of the elastic flows decrease with increasing values of the *pfl_flushing_timer* parameter. Of course, if the value of the *pfl_flushing_timer* parameter is greater the process of removing the identifiers of the elastic flows from the PFL is performed more rarely. For the

case when the *pfl_flushing_timer* is equal to 5 s a new streaming flow is accepted in the AC block within 2 s from the time when it began to send the traffic, but the mean time of transmission of the elastic flows in the same case increases significantly in both FAN architectures. We can notice that in FAN with PDRR the mean transmission time of elastic flows is lower than in FAN with PFQ for the low values of the *pfl_flushin_timer*. Thus it decreases more softly than in FAN with PFQ. The dependence of the transmission time period of the elastic flows on the *pfl_flushing_timer* parameter in FAN with PFQ may be approximated by the following logarithmic function:

$$y = -108.94 \ln x + 611.43 \tag{5.5}$$

If we want to decrease the acceptance time of a new streaming flow in the AC block without increasing the mean transmission time period of the elastic flows we should solve Equation (5.5) for $y = 123.87$ s (for the value of the mean transmission time of the elastic flows in the basic FAN link with PFQ). 10 simulation runs were made for the *pfl_flushing_timer* equal to the value of $x = 89.50$ s calculated from Equation (5.5). The results, presented in Tab. 5.4, show that a new streaming flow is accepted in the AC block after 51.38±0.16 s from the time when it began to send the traffic and the mean time of transmission of the elastic flows is equal to 139.21±7.19 s. The EFM allows for decreasing of the acceptance time of a new streaming flow in the AC block of the FAN router with PFQ without increasing the mean transmission time of the elastic flows. The gain in the *waiting_time* value is on the level of 30%.

The dependence of the transmission time period of the elastic flows on the *pfl_flushing_timer* parameter in FAN with PDRR may be approximated by the following logarithmic function:

$$y = -82.84 \ln x + 521.37 \tag{5.6}$$

If we want to decrease the acceptance time of a new streaming flow in the AC block without increasing the mean transmission time period of the elastic flows we should solve Equation (5.6) for $y = 125.16$ s (for the value of the mean transmission time of the elastic flows in the basic FAN link with the PDRR). 10 simulation runs were made for the *pfl_flushing_timer* value equal to the value of $x = 119.50$ s calculated from Equation (5.6). The obtained results are presented in Tab. 5.4. They show that a new streaming flow is accepted in the AC block after 81.98±2.66 s and the mean transmission time of elastic flows is equal to 125.96±6.39 s. As we can see, the transmission time period of the elastic flows in this case is comparable with the value of this parameter obtained for the basic FAN algorithm. Unfortunately, the value of the acceptance time of a new streaming flow in this case is also almost the same as for the basic FAN algorithm and significantly greater than the value of this parameter obtained for the EFM

with the PFQ algorithm. We may conclude that it is impossible to decrease the *waiting_time* for streaming flows without increasing the transmission time of elastic flows.

Conclusion for Experiments 3 – 6

The EFM has better properties than the BFM. It ensures uninterrupted transmission of accepted streaming flows. Based on the results obtained in Experiments 3-6 we can conclude that, in EFM, it is possible to ensure short acceptance times of new streaming flows in the AC block independently of the number of elastic flows being active in the background. For FAN with PFQ it is also possible to decrease the mean acceptance time of streaming flows without increasing the mean transmission time of elastic flows. The EFM works sufficiently well with the PFQ and the PDRR algorithms, but for low values of *pfl_flushing_timer* it drastically deteriorates the transmission properties of elastic flows.

5.3.2 Simulation experiments on the RAEF mechanism

The RAEF mechanism is proposed as an alternative to the EFM. The identifiers of elastic flows are removed from the PFL each time a packet of a new flow arrives to the router in congestion. Only the IDs of elastic flows being active for the specified time period (*active_time*) are removed in a flushing action. The results of the following simulation experiments show the usefulness of the RAEF mechanism and its possibilities for improving the network performance in the congestion state.

Experiment 7

The *waiting_time* values were analyzed in function of the number of elastic flows being active in background in this simulation experiment. 400 simulation runs were made in various conditions with the number of elastic flows which wanted to transmit their traffic ranging from 200 to 600. The duration of each simulation run was set to 250 s, which allowed for observing the acceptance time of new streaming flows. The other simulation parameters were set as in the previous simulation experiments. The mean values of *waiting_time* in function of the number of elastic flows active in background for FAN with PFQ or with PDRR are presented in Fig. 5.10 and Fig. 5.11, respectively.

The obtained results show that in the examined range the *waiting_time* values are constant independently of the number of the elastic flows that want to send the traffic in both FAN architectures. As we can see, the *waiting_time* values increase with the increasing values of the *active_time* parameter. The results are similar to that obtained in the experiment provided for the EFM and, of course

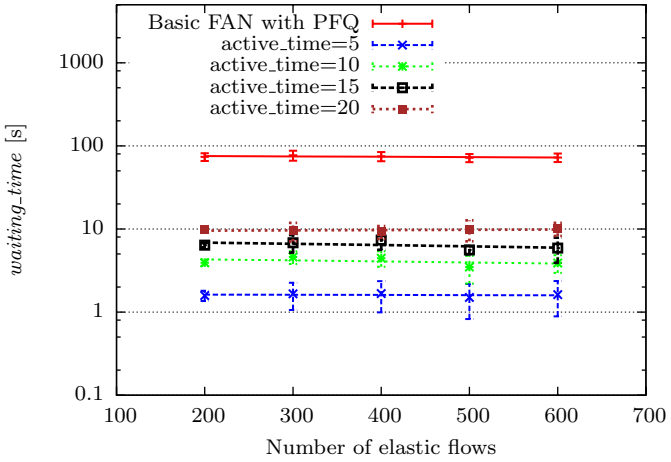

Figure 5.10: The mean *waiting_time* in FAN with PFQ and RAEF

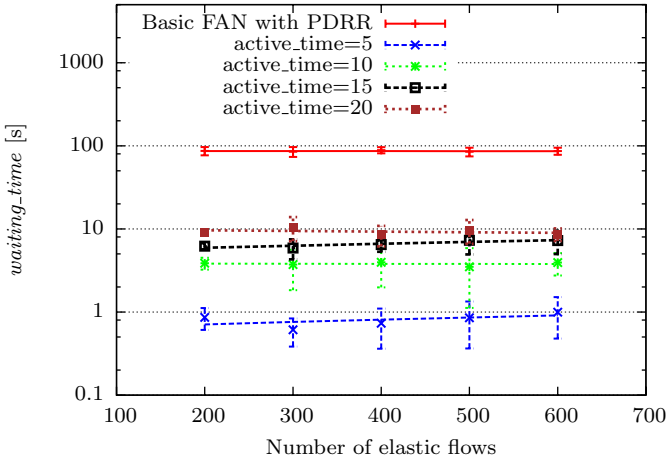

Figure 5.11: The mean *waiting_time* in FAN with PDRR and RAEF

significantly better than those presented for the basic FAN without any flushing mechanism. The results for both FAN architectures are similar and the best were obtained when the *active_time* was set to 5 s. In this case, a new flow was always accepted within 2 s from the time when it began to send the traffic, as in the EFM. In FAN with PDRR almost all streaming flows were accepted in time even

less than 1 s when the *active_time* was set to 5 s. The values of the *active_time* parameter equal to 10 s or 15 s are acceptable for local voice calls and the value of 20 s allow for ensuring a sufficiently low acceptance time of new streaming flows for international calls.

Experiment 8

The reason of the long transmission time of elastic flows for low values of *active_time* is the same as for FAN with EFM. After almost each cleaning action the number of elastic flows accepted again is greater than in the basic FAN. In this simulation experiment, the mean number of elastic flows in FAN with PFQ or PDRR was analyzed. The simulation duration was set to 500 s, which allows for estimating the values of the estimated parameter in the steady state for a sufficiently long time. The rest of the simulation parameters were set as in the previous case.

The values of the mean number of elastic flows in function of the number of elastic flows active in background for FAN with PFQ or with PDRR are presented in Fig. 5.12 and Fig. 5.13, respectively.

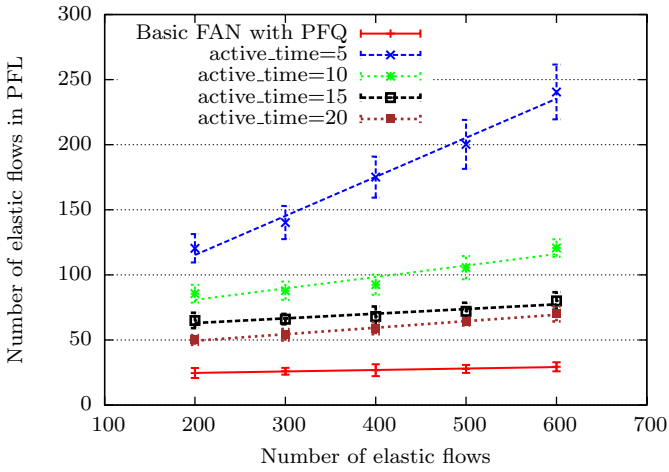

Figure 5.12: The mean number of elastic flows in PFL in FAN with PFQ and RAEF

The results are better than for FAN with the EFM. While the mean number of elastic flows accepted in the router after any flushing action increases with the increasing number of elastic flows which want to begin or continue transmission in the background, their values are significantly lower than in FAN with EFM. In FAN with RAEF and PFQ, the number of accepted flows after flushing is

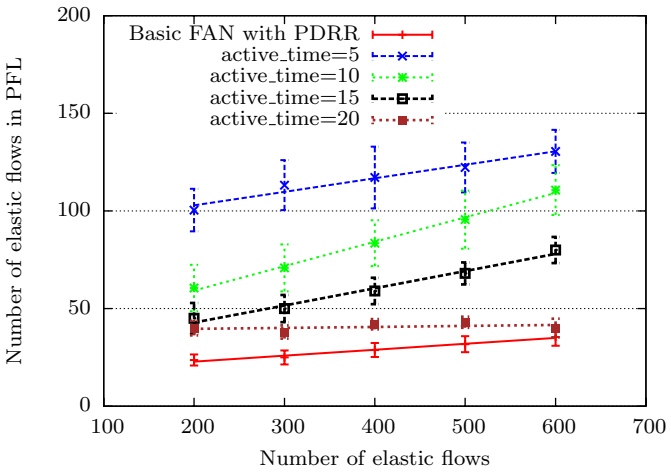

Figure 5.13: The mean number of elastic flows in PFL in FAN with PDRR and RAEF

significantly greater than in FAN with PDRR only in case when the *active_time* parameter was set to 5 s. For the other values of this parameter, the results are similar for both FAN architectures. Unfortunately, in both cases, the obtained results are still much worse than in the basic FAN, where the mean number of accepted elastic flows is almost independent of the number of all elastic flows which want to transmit their packets. This a disadvantage of the RAEF mechanism and shows that this solution may be not scalable, especially for low values of the *active_time* parameter. On the other hand, the mean number of elastic flows after any cleaning action of the PFL is lower than in FAN with EFM, what makes this solution more appropriate to use.

Experiment 9

The RAEF mechanism has the same drawbacks as the EFM. One of them is that after any flushing action transmission of elastic flows is broken. The rejected flows may continue transmission only if they are accepted again in the router. In this simulation experiment, it was analyzed how a decrease of the values of the *waiting_time* parameter for new streaming flows affects the mean transmission time of the elastic flows. The mean transmission time of the elastic flows for the case with 200 elastic flows was estimated by simulation. The duration of each simulation run was set to 1500 s, at allowed for observing the beginning and finishing time instants of each elastic flow. The rest of the simulation parameters was set as in the previous case.

The values of the transmission time of the elastic flows in function of *active_time* for the case with 200 elastic flows active in background for both FAN architectures with RAEF are presented in Fig. 5.14.

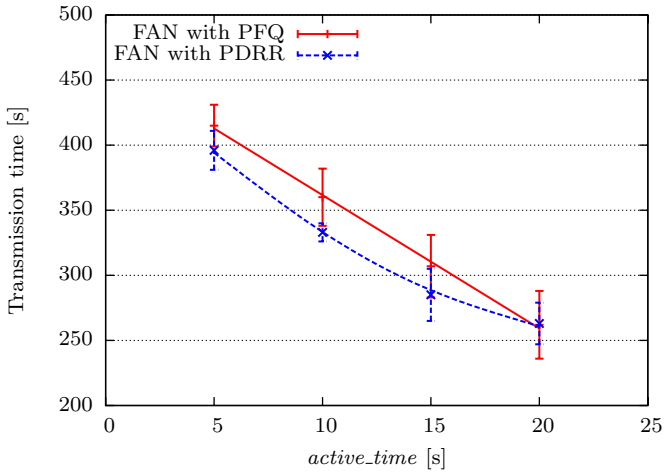

Figure 5.14: The mean transmission time of elastic flows in FAN with RAEF

The results show that the transmission time period of the elastic flows decrease with increasing values of the *active_time* parameter. Of course, if the *active_time* parameter value is greater we select a smaller number of elastic flows whose identifiers are removed from the PFL any time congestion is noticed. As we can see, the PDRR works better than the PFQ for the low values of the *active_time*. The dependence of the transmission time period of the elastic flows on the *active_time* parameter for FAN with PFQ may be approximated by the following logarithmic function:

$$y = -10.25 \ln x + 464.69 \tag{5.7}$$

If we want to decrease the acceptance time of a new streaming flow in the AC block without increasing the mean transmission time period of the elastic flows we should solve Equation (5.7) for $y = 125.16$ s (for the value of the mean transmission time of the elastic flows in the basic FAN link with the PFQ). 10 simulation runs for the *active_time* value equal to the value of $x = 33.25$ s calculated from Equation (5.7). The results obtained in this simulation scenario are presented in Tab. 5.4. The transmission time period of the elastic flows in this case is comparable with the value of this parameter obtained for the basic FAN algorithm. What is more important, the acceptance time of a new streaming

flow is over seven times less than the same parameter obtained for the basic FAN algorithm. This is a very strong advantage of the RAEF mechanism.

The dependence of the transmission time period of the elastic flows on the *active_time* parameter in FAN with PDRR may be approximated by the following logarithmic function:

$$y = -91.64 \ln x + 537.54 \tag{5.8}$$

If we want to decrease the acceptance time of new streaming flows in the AC block without increasing the mean transmission time period of the elastic flows we should solve Equation (5.8) for $y = 125.16$ s (for the value of the mean transmission time of the elastic flows in the basic FAN link with the PDRR). As in the previous case, 10 simulation runs were made for the *active_time* value equal to the value of $x = 91.45$ s calculated from Equation (5.8). The obtained results are presented in Tab. 5.4. As we can see, the transmission time period of the elastic flows in this case is comparable with the value of this parameter obtained for the basic FAN algorithm. The mean acceptance time of new streaming flows is smaller than the same parameter obtained for the basic FAN algorithm, but significantly greater than the value of this parameter presented for the RAEF mechanism with the PFQ algorithm.

Conclusion for Experiments 7 – 9

The results of carefully selected simulation experiments for analyzing the RAEF mechanism show that the algorithm ensures quick acceptance times of new streaming flows in the AC block independently of the number of elastic flows being active in the background and the number of streaming flows which want to begin the transmission. Similarly to EFM, the RAEF mechanism works satisfactorily with both analyzed versions of the scheduling algorithm, the PFQ and the PDRR. The mechanism works better than the EFM and is more suitable for using in FAN.

5.3.3 Simulation experiments on the RBAEF mechanism

The following three simulation experiments present the possibilities of the RBAEF mechanism to improve the network performance in congestion. The analyzed mechanism is proposed as an answer to the drawback of the EFM and RAEF proposals. The rejected flows after a flushing action are temporarily blocked in the BFL and cannot compete with other flows for acceptance in the AC block immediately. This method allows for reducing the number of flows accepted in the router after a flushing action, as shown in the following simulation experiments.

Experiment 10

The mean *waiting_time* of streaming flows was estimated in various conditions of 400 simulation runs. The *blocked_period* parameter was set to 1 s. This value was chosen experimentally. For greater values of this parameter, there were no advantages in the acceptance time of new streaming flows and the mean transmission time of elastic flows was longer. On the other hand, the smaller values of *blocked_period* are not appropriate because the removed flows from the PFL are accepted again too quickly significantly increasing the number of accepted elastic flows after a flushing action. The other simulation parameters were set as in the similar simulation experiment made for the EFM. The mean values of *waiting_time* in function of the number of elastic flows active in background for FAN with PFQ or with PDRR are presented in Fig. 5.15 and Fig. 5.16, respectively.

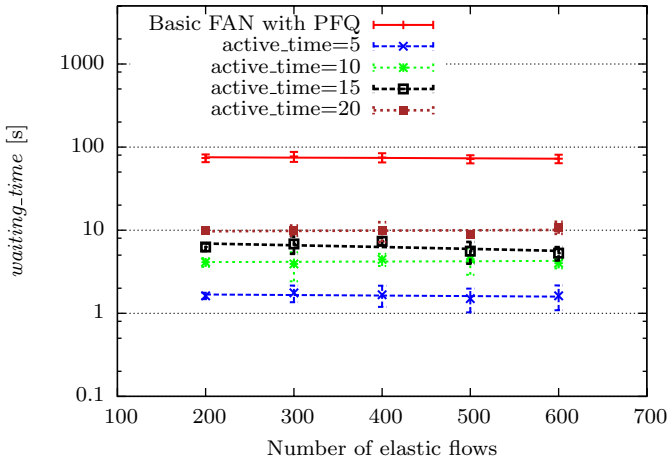

Figure 5.15: The mean *waiting_time* in FAN with PFQ and RBAEF

The obtained results show that in the examined range the *waiting_time* values are constant independently of the number of elastic flows that want to send the traffic in both FAN architectures. The *waiting_time* values increase with increasing values of the *active_time* parameter in the examined range. The obtained results for FAN with PDRR are significantly better than the results presented for the basic FAN link and slightly better when comparing to the values presented for the link with the PFQ (especially for the lowest values of the *active_time* parameter). The results for FAN with RBAEF are similar to those obtained for FAN with EFM. The best results are observed for *active_time* set to 5 s. The simulation results show that in this case a new flow was always accepted within

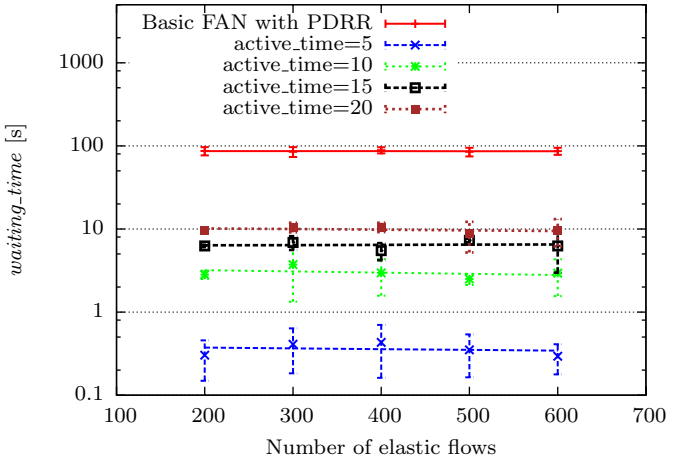

Figure 5.16: The mean *waiting_time* in FAN with PDRR and RBAEF

2 s, as in FAN with EFM. Similarly, in the cases when the *active_time* was set to 10 s or 15 s new flows were accepted a few seconds later, but still in the acceptable range for local voice calls. When the *active_time* was set to 20 s, new streaming flows were accepted after approximately 9 s, which is acceptable for international calls.

Experiment 11

The long transmission time of elastic flows for small values of the *active_time* parameter is caused among other things by a too high number of elastic flows accepted after flushing. In this simulation experiment, the mean number of elastic flows in FAN with PFQ or PDRR was analyzed. The simulation duration was set to 500 s and the rest of the simulation parameters were set as in the previous case.

The values of the mean number of elastic flows in function of the number of elastic flows active in background for FAN with PFQ or with PDRR are presented in Fig. 5.17 and Fig. 5.18, respectively.

The mean number of elastic flows accepted in the router after any flushing action increases with increasing number of elastic flows which want to begin or continue transmission in the background. In FAN with PFQ the best results are observed for *active_time* equal to 15 s or 20 s. For the smaller values of this parameter, the mean transmission time of elastic flows increases faster. As we can see, in FAN with PDRR, the values of the analyzed parameter are similar

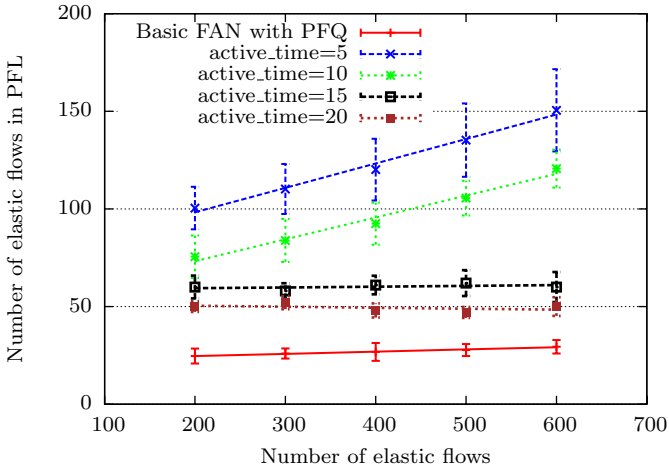

Figure 5.17: The mean number of elastic flows in PFL in FAN with PFQ and RBAEF

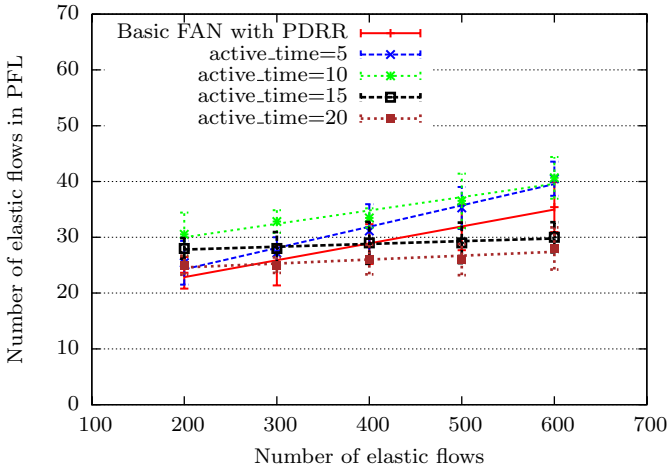

Figure 5.18: The mean number of elastic flows in PFL in FAN with PDRR and RBAEF

for each *active_time* value. Moreover, they are comparable with values observed for the basic FAN link and increases relatively slowly. This an advantage of the RBAEF mechanism and shows that this solution may be scalable for all analyzed values of the *active_time* parameter.

Experiment 12

The main motivation for the RBAEF mechanism was to eliminate the drawback of the EFM and RAEF mechanisms, i.e., long transmission time of elastic flows for low values of *pfl_flushing_timer* or *active_time*. In this simulation experiment, the mean transmission time of 200 elastic flows active in background was analyzed. As in the previous cases, the transmission time was set to 1500 s. The rest of the simulation parameters were set as in the previous case. The values of the mean transmission time of the elastic flows in function of the *active_time* parameter for both FAN architectures are presented in Fig. 5.19.

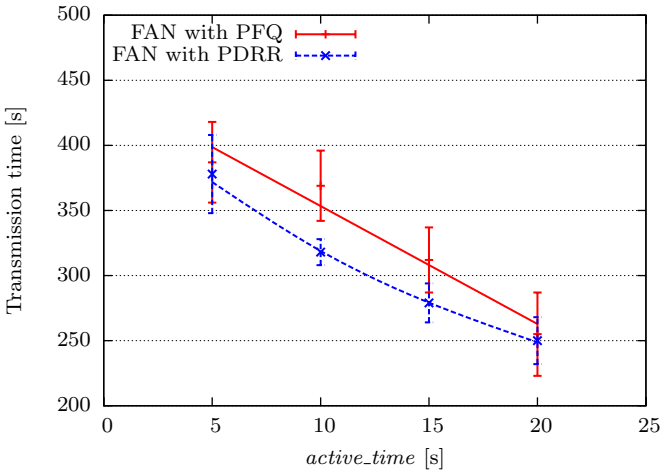

Figure 5.19: The mean transmission time of elastic flows in FAN with RBAEF

As we can see, the values of the transmission time period of the elastic flows decrease with increasing values of the *active_time* parameter. Of course, if the *active_time* parameter value is greater we select a smaller number of the elastic flows which identifiers are removed from the PFL any time when congestion is noticed what causes an increase of the acceptance time of a new streaming flow. Unfortunately, the mean transmission time is still too great. The mean number of accepted elastic flows after flushing is significantly lower than in two mechanisms presented before. It means that the main reason of the long transmission time of elastic flows is that the flows are blocked after flushing and cannot continue the transmission immediately. It lengthens the transmission of each elastic flow.

The dependence of the transmission time period of the elastic flows on the

active_time parameter for the *blocked_time* parameter equal to 1 s in FAN with PFQ may be approximated by the following logarithmic function:

$$y = -9.05 \ln x + 444.11 \tag{5.9}$$

If we want to decrease the acceptance time of a new streaming flow in the AC block without increasing the mean transmission time period of the elastic flows we should solve Equation (5.11) for $y = 123.87$ s (for the value of the mean transmission time of the elastic flows in the basic FAN link). The results of 10 simulation runs for the *active_time* value equal to the value of $x = 35.40$ s calculated from Equation (5.11) are presented in Tab. 5.4. The transmission time period of the elastic flows in this case is comparable with the value of this parameter obtained for the basic FAN algorithm, while the acceptance time of a new streaming flow is over six times less than the same parameter obtained for the basic FAN algorithm. This is a very strong advantage of the RBAEF mechanism.

The dependence of the transmission time period of the elastic flows on the *active_time* parameter for the *blocked_time* parameter equal to 1 s for FAN with PDRR may be approximated by the following logarithmic function:

$$y = -93.48 \ln x + 530.50 \tag{5.10}$$

As we can see, the PDRR works better than PFQ for the small values of the *active_time* parameter. If we want to decrease the acceptance time of a new streaming flow in the AC block without increasing the mean transmission time period of the elastic flows we should solve Equation (5.12) for $y = 125.16$ s (for the value of the mean transmission time of the elastic flows in the basic FAN link with the PDRR). The results of 10 simulation runs for the *active_time* value equal to the value of $x = 76.40$ s calculated from Equation (5.12) are presented in Tab. 5.4. As we can see, the transmission time period of the elastic flows in this case is comparable with the value of this parameter obtained for the basic FAN algorithm. The mean acceptance time of new streaming flows is significantly smaller than the value of the same parameter obtained for the basic FAN algorithm, but greater than the value of this parameter obtained for the RBAEF mechanism with the PFQ algorithm.

Conclusion for Experiments 10 – 12

Based on the results obtained in the simulation experiments we can conclude that the RBAEF algorithm ensures quick acceptance of new streaming flows in the AC block independently of the number of elastic flows being active in background and the number of streaming flows which want to begin transmission. As both previously presented mechanisms, the RBAEF mechanism also works

satisfactorily with both analyzed versions of the scheduling algorithm, the PFQ and the PDRR. This mechanism allows for decreasing the mean number of elastic flows accepted after a flushing action, but it is impossible to decrease the mean transmission time of elastic flows for small values of the *active_time* parameter.

5.3.4 Simulation experiments on the RPAEF mechanism

The RPAEF mechanism has the same goals as RBAEF, i.e., to decrease the number of elastic flows accepted after a flushing action and to ensure the mean transmission time of elastic flows at an acceptable level. While the idea of RPAEF and RBAEF is similar, they work completely differently. In RPAEF, the IDs of rejected flows after flushing are written to the PAFL. If in the congestion-less state the ID of an incoming packet is in the PAFL or the PAFL is empty, the flow is accepted with probability equal to one. On the other hand, if the flow's ID is not in the PAFL, the flow is accepted with a low probability. It allows for reducing the number of elastic flows accepted after flushing and almost continuous transmission of each flow accepted in the router. The following three simulation experiments present the possibilities of the RPAEF mechanism to improve the network performance in congestion.

Experiment 13

The *waiting_time* parameter for both FAN versions with RPAEF is analyzed in this simulation experiment. The mean acceptance time of new streaming flows (*waiting_time*) in the AC block in a FAN link was analyzed in this experiment by 400 simulation runs made in various conditions. The duration of each simulation run was set to 250 s, which allowed for observing the acceptance time of new streaming flows. The P_{RPAEF} probability was set to 0.03. This value was chosen experimentally. The other simulation parameters were set as in the previous simulation experiment. The mean values of *waiting_time* in function of the number of elastic flows active in background for FAN with PFQ or with PDRR are presented in Fig. 5.20 and Fig. 5.21, respectively.

The values of the *waiting_time* parameter are constant for each *active_time* value in both FAN architectures in the examined range. It means that, as in all previously described mechanisms, new streaming flows are accepted in the AC block independently of the number of elastic flows, which want to send the traffic. The *waiting_time* values increase with increasing values of the *active_time* parameter in the examined range, but are significantly smaller than in the basic FAN. The results for both FAN architectures are similar. The best results were obtained when the *active_time* was set to 5 s. The *active_time* equal to 10 s is also acceptable for local calls, while the values of 15 s and 20 s ensure sufficiently small the *waiting_time* values for international VoIP connections.

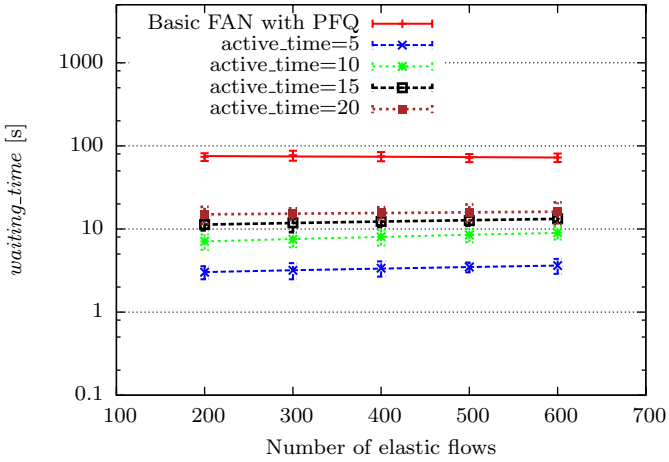

Figure 5.20: The mean *waiting_time* in FAN with PFQ and RPAEF

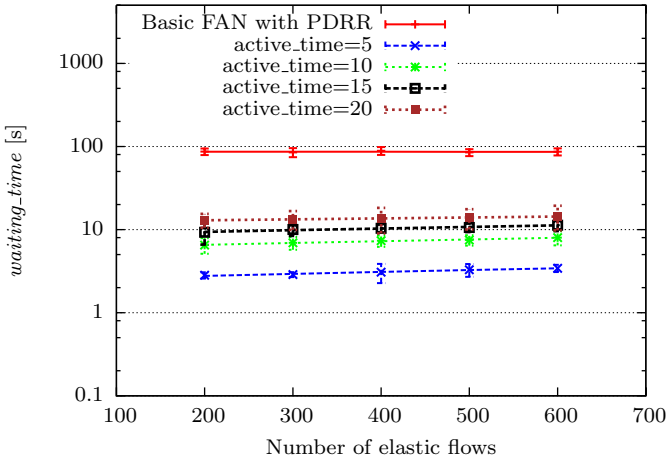

Figure 5.21: The mean *waiting_time* in FAN with PDRR and RPAEF

Experiment 14

In this simulation experiment, the mean number of elastic flows in FAN with PFQ or PDRR accepted after any flushing action was analyzed. The simulation duration was set to 500 s, which allows for estimating the analyzed parameter in the steady state for a sufficiently long time. The rest of the simulation parameters were set as in the previous case.

The values of the mean number of elastic flows in function of the number of elastic flows active in background for two FAN versions with RPAEF are presented in Fig. 5.22 and Fig. 5.23, respectively.

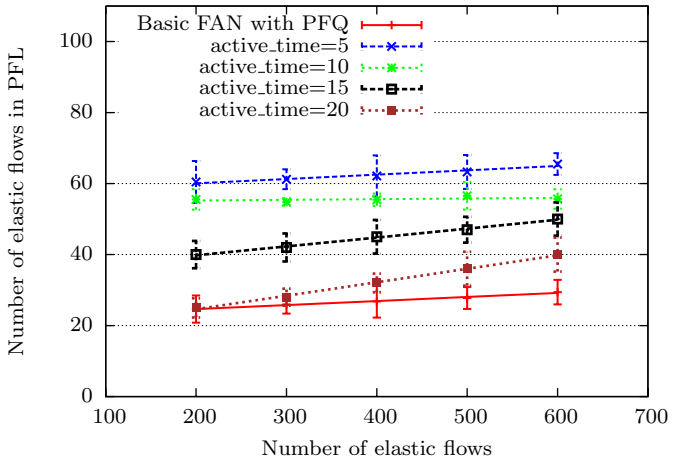

Figure 5.22: The mean number of elastic flows in PFL in FAN with PFQ and RPAEF

As we can see, the number of accepted elastic flows after flushing is smaller than for EFM, RAEF and RBAEF (in FAN with PDRR the results are comparable). Moreover, the transmission of once accepted elastic flows is broken periodically only for a very short time period. The algorithm looks to be stable and scalable. It should be noticed that the mean break in transmission of elastic flows after flushing is 0.41±0.05 s in FAN with PFQ and 0.38±0.04 s in FAN with PDRR. 99.5% of the removed elastic flows are accepted again in time less than the value of the *priority_access* parameter. This advantage of the RPAEF mechanism gives the justified reasons for using it in FAN.

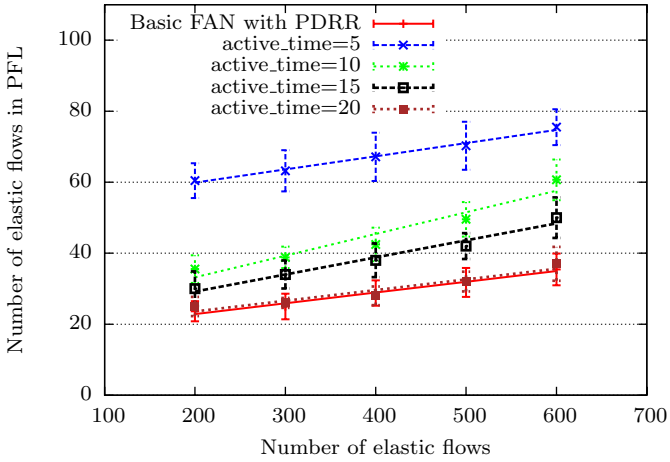

Figure 5.23: The mean number of elastic flows in PFL in FAN with PDRR and RPAEF

Experiment 15

In the last simulation experiment on RPAEF, it was checked how decreasing of the values of the *waiting_time* parameter for new streaming flows affects the mean transmission time of the elastic flows. The duration of each simulation run was set to 1500 s, which allowed for observing the beginning and finishing time instants of each elastic flow. The rest of the simulation parameters were set as in the previous case.

The values of the transmission time of the elastic flows in function of active_time for the case with 200 elastic flows active in background for both FAN architectures with RPAEF are presented in Fig. 5.24.

The values of the transmission time period of the elastic flows decrease with increasing values of the *active_time* parameter. Unfortunately, for small values of *active_time* the mean transmission time of elastic flows is still too high. But it decreases very fast and for *active_time* = 20 s it is comparable with the values obtained for the basic FAN. The dependence of the transmission time period of the elastic flows on the *active_time* parameter in FAN with PFQ may be approximated by the following linear function:

$$y = -10.67x + 343.35 \tag{5.11}$$

If we want to decrease the acceptance time of a new streaming flow in the AC block without increasing the mean transmission time period of the elastic flows we should solve Equation (5.11) for $y = 123.87$ s (for the value of the

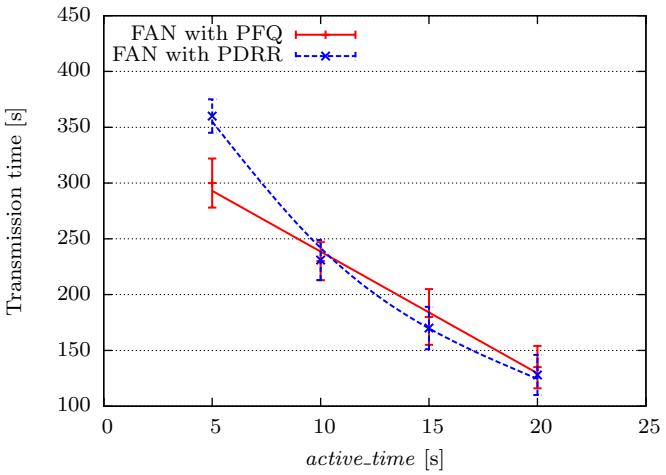

Figure 5.24: The mean transmission time of elastic flows in FAN with RPAEF

mean transmission time of the elastic flows in the basic FAN link with PFQ). The results of 10 simulation runs made for the *active_time* equal to the value of $x = 20.57$ s calculated from Equation (5.11) show that a new streaming flow is accepted in the AC block after 15.52±0.79 s from the time when it began to send the traffic and the mean time of transmission of the elastic flows is equal to 130.12±8.74 s. The results are presented in Tab. 5.4. The RPAEF allows for decreasing the acceptance time of new streaming flows in the FAN router with PFQ without increasing the mean transmission time of the elastic flows. The gain in the *waiting_time* value is on the level of 80%.

The dependence of the transmission time period of the elastic flows on the *active_time* parameter in FAN with PDRR may be approximated by the following logarithmic function:

$$y = -166.91 \ln x + 628.73 \tag{5.12}$$

If we want to decrease the acceptance time of a new streaming flow in the AC block without increasing the mean transmission time period of the elastic flows we should solve Equation (5.12) for $y = 125.16$ s (for the value of the mean transmission time of the elastic flows in the basic FAN link with the PDRR). 10 simulation runs were made for the *active_time* value equal to the value of $x = 20.40$ s calculated from Equation (5.12). The obtained results show that a new streaming flow is accepted in the AC block after 13.46±0.82 s and the mean transmission time of elastic flows is equal to 124.32±2.32 s. The results are also

presented in Tab. 5.4. As we can see, the transmission time period of the elastic flows in this case is comparable with the value of this parameter obtained for the basic FAN algorithm. The mean acceptance time of new streaming flows is over six times less than in the basic FAN link with PDRR.

Conclusion for Experiments 13 – 15

The results of carefully selected simulation experiments for analyzing the RPAEF mechanism show that the algorithm ensures quick acceptance times of new streaming flows in the AC block independently of the number of elastic flows being active in background and the number of streaming flows which want to begin transmission. Similarly to the other congestion control mechanisms proposed for FAN, the RPAEF mechanism works satisfactorily with both analyzed versions of the scheduling algorithm, the PFQ and the PDRR. The analyzed mechanism allows for decreasing the number of elastic flows accepted after flushing, which ensures that the FAN with RPAEF is a scalable solution. Moreover, the transmission of each elastic flow once accepted is broken only for a short time. We have to note that one of the basic FAN assumptions was that the IDs of accepted flows in the router cannot be removed from the PFL before they finish the transmission.

5.3.5 FAN in case of failure

In the basic FAN architecture, the ID of a flow can be removed from the PFL only in case of its long enough inactivity. It means that the transmission of all flows accepted in the AC block cannot be broken in a failure-less state.

In this section we analyze the mean acceptance time of new streaming flows in the topology presented in Fig. 5.25.

For simplicity, there is one source node and two destination nodes in the following experiments. The node S is the source node of streaming and elastic flows, while the nodes $D1$ and $D2$ are destination nodes of traffic sent from the node S. It was assumed that bottleneck links L3 and L5 are FAN links of the 100 Mbit/s capacity. The capacity of the rest of the links, with the FIFO queue, was set to 1 Gbit/s. The shortest path routing was implemented in this network, which means that under normal conditions the traffic to node D1 is sent through nodes R1, R2 and R3 while the traffic to node D2 is sent through nodes R1, R4 and R5. By using such a topology it was assumed that link L5 was treated as a backup for the traffic sent normally through link L3. The effects of failures of link L2 at a chosen time instant were analyzed and presented in the experiments described below. The traffic types and simulation parameters were set as in the previous simulation runs. The acceptance time of each streaming flow in the AC block of node R2 (before failure) and node R4 (after failure) was analyzed.

Table 5.4: Transmission parameters for basic FAN link and with the EFM, RBAEF and RAEF

Mechanism / Parameter	**Value of parameter [s]**	***waiting_time* [s]**	**Transmission time period of elastic flows [s]**
FAN with PFQ			
Basic FAN	-	73.64±7.84	123.87±16.07
EFM / *pfl_flushing_timer*	89.50	51.38±0.16	139.21±7.19
RAEF / *active_time*	33.25	10.32±0.59	145.05±17.35
RBAEF / *active_time*	35.40	12.05±0.56	145.40±19.50
RPAEF / *active_time*	20.57	15.52±0.79	130.12±8.74
FAN with PDRR			
Basic FAN	-	86.29±6.43	125.16±3.69
EFM / *pfl_flushing_timer*	119.50	81.98±2.66	125.96±6.39
RAEF / *active_time*	91.45	66.15±1.19	124.33±12.56
RBAEF / *active_time*	76.40	53.60±0.58	128.79±6.70
RPAEF / *active_time*	20.40	13.46±0.82	124.32±2.32

Experiment 16

In this simulation experiment the mean acceptance time of new streaming flows in FAN links with the number of background elastic flows ranging from 200 to 600 was checked. The duration of each simulation run was set to 500 s. At 250 s, link L2 was turned off and the packets of all flows were sent through link L5. The redirected streaming flows had to compete for access to the L5 link along with all other elastic flows. Basic FAN links have unacceptable values of the *waiting_time* parameter (see Tab. 5.5). It takes tens of seconds (and even hundreds of seconds for the redirected flows) before a new streaming flow is accepted in the router. If we imagine that this flow exemplifies a VoIP call, it is obvious that the break in transmission of such a flow is much too long. The results of simulation runs show that the acceptance time of streaming flows can be decreased by using the congestion control mechanisms presented in Chapter 4

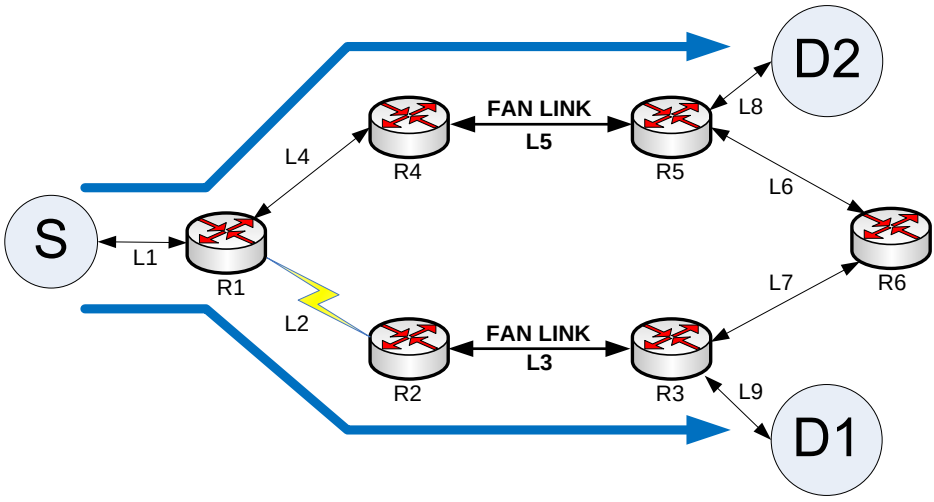

Figure 5.25: The basic simulation topology

on both examined FAN links. The comparison of the *waiting_time* values for link L3 in basic FAN with PFQ and PDRR algorithms and for its modified versions with EFM (*pfl_flushing_timer* = 5 s), RAEF (*active_time* = 5 s), RBAEF (*active_time* = 5 s, blocked time = 1 s) and RPAEF (*active_time* = 5 s, *priority_access* = 1 s and $P_{RPAEF} = 0.03$) are presented in Fig. 5.26 and Fig. 5.28, respectively. Fig. 5.27 and Fig. 5.29 show the similar results for the L5 link.

We can see that the difference between the values for basic FAN and its modified versions are significant. If we use the congestion control mechanisms mentioned above the new streaming flows may be accepted in the routers after less than one second in both the basic and backup links. The presented results show that the acceptance time of new streaming flows does not depend on the number of elastic flows in the background. The simulation results for the case with 200 elastic flows are presented in Tab. 5.5. In the basic FAN, for both queuing algorithms, a new streaming flow is accepted in the AC block of node R2 after tens of seconds while after the L2 link failure this time (observed at node R4) raises to a few hundreds seconds. The *waiting_time* period in the R4 router represents the amount of time in which the streaming flow struggles for acquiring the backup link's resources. The difference between the acceptance time of a new streaming flow in routers R2 and R4 is significant. A new flow may be accepted in router R2 when one or more elastic flows finish their transmission which allows for increasing the *fair_rate* values. After a failure, all redirected flows have to compete for the access to the R4 router which significantly increases the

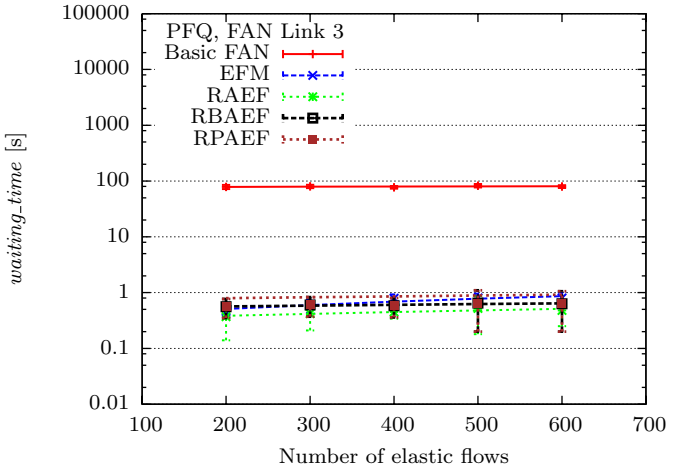

Figure 5.26: The mean *waiting_time* in FAN with PFQ for Link L3

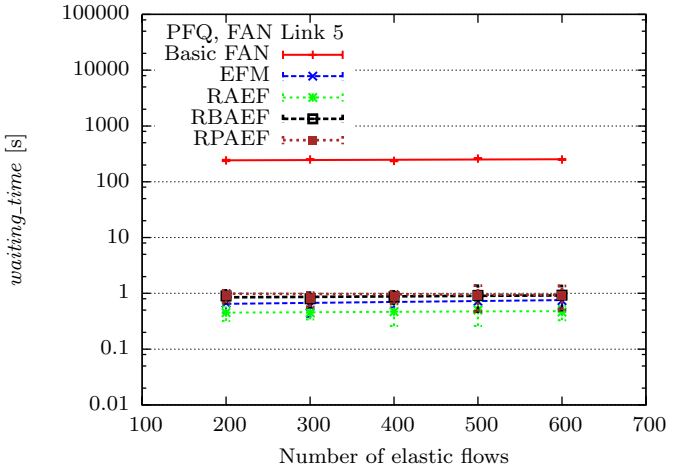

Figure 5.27: The mean *waiting_time* in FAN with PFQ for Link L5

number of competitors in that node. In turn, this decreases their chance for being accepted and, therefore, increases the mean acceptance time.

Based on the obtained results, we may conclude that the congestion control mechanisms proposed for FAN significantly improve the performance of traffic classified as streaming. It is possible to set the values of the *active_time* (for

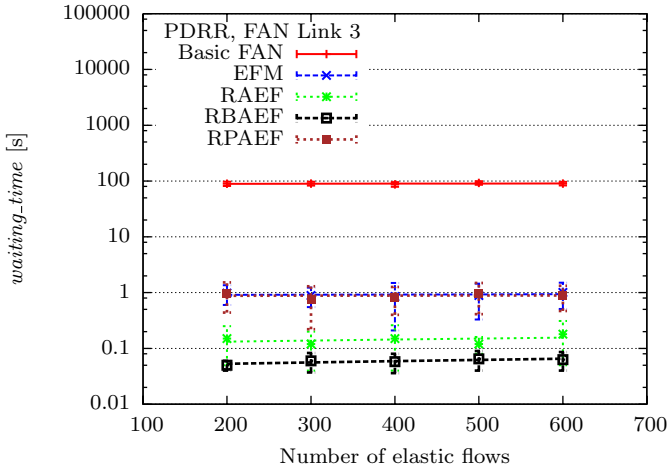

Figure 5.28: The mean *waiting_time* in FAN with PDRR for Link L3

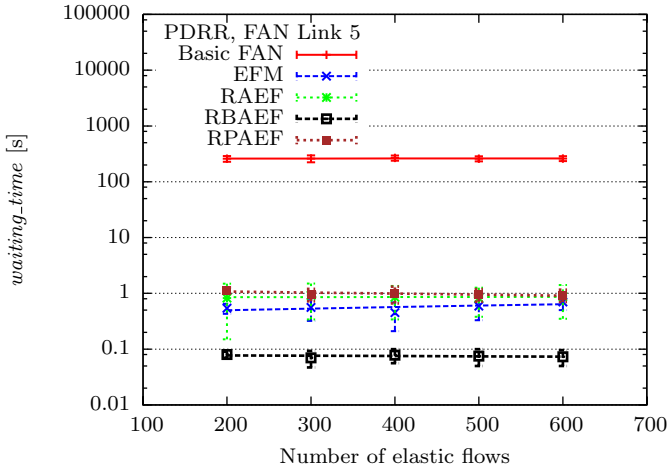

Figure 5.29: The mean *waiting_time* in FAN with PDRR for Link L5

RAEF, RBAEF and RPAEF) and the *pfl_flushing_timer* (for EFM) parameters so that they ensure a very short acceptance time of streaming flows in a backup link after a failure. It is a strong advantage of those congestion control mechanisms.

The problem of many flows competing for access to the overloaded resource

at the same time, presented in this simulation experiment is very important for FAN. This problem is in fact unsolved because we do not know which flows should be accepted and in which way. The mechanisms proposed in this dissertation are the first answer to this problem. They show that it is possible to protect streaming flows not only by ensuring the short acceptance time of them, but also the continuous transmission in normal conditions and short outages in case of failure. Moreover it is possible to ensure the negligible impact of those mechanisms to the transmission of elastic flows and to assure the scalability possibilities on the same level as in the basic FAN. The solutions proposed here are flexible, stable and give the chance for more reliable transmission in Flow-Aware Networks.

Table 5.5: Time mean *waiting_time* of streaming flows in the AC block in an XP router before and after L2 link failure

Mechanism	PFQ		PDRR	
	waiting_time router R2 [s]	*waiting_time* router R4 [s]	*waiting_time* router R2 [s]	*waiting_time* router R4 [s]
Basic FAN	78.43±6.56	240.86±26.64	89.28±7.87	258.34±30.65
EFM	0.46±0.10	0.67±0.22	0.98±0.38	0.54±0.11
RAEF	0.37±0.23	0.45±0.13	0.15±0.10	0.82±0.67
RBAEF	0.56±0.21	0.90±0.23	0.05±0.01	0.08±0.06
RPAEF	0.84±0.34	0.94±0.12	0.98±0.54	1.13±0.14

Part III

Approximate Flow-Aware Networking

6 The new architecture: Approximate Flow-Aware Networking

If I have seen further than others, it is by standing upon the shoulders of giants.

— Isaac Newton

6.1 Motivation for Approximate Flow-Aware Networking

The two well known FAN architectures, with PFQ or PDRR scheduling algorithms, work satisfactorily but have some drawbacks. The main one is complexity of queuing and selecting packets for sending. It is necessary to analyze the virtual tags of each flow in PFQ. The AFL has to be maintained. The position of the pointer provided for separating the packets of streaming and elastic flows in the PIFO queue has to be analyzed each time the packet is selected for transmission. The AFL is used for selecting the proper FIFO queue when sending the packet in PDRR. It is necessary to implement many FIFO queues, individual for each elastic flow and one for packets of the streaming flows.

The packet transmission is usually more effective if the network architecture is as simple as possible. In the following section a new FAN architecture called Approximate Flow-Aware Networking (AFAN) and based on the Approximate Fair Dropping (AFD) algorithm [74] is presented.

6.2 Architecture of AFAN

The AFAN ensures fair transmission of packets which belong to elastic flows and prioritizing possibilities by using a separate FIFO queue for streaming flows. The pseudo code of the admission control mechanism in AFAN is presented in Tab. 6.1.

Table 6.1: Pseudo code of admission control mechanism in AFAN

Admission control module:
1. on arrival of packet p
2. **If** flow ID ϵ PFL **then**
3. accept p for queuing
4. **Else If** congestion is noticed **then**
5. drop p
6. **Else** accept p with probability $P1_{AFAN}$

6.2.1 Admission control operation

At the beginning, 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 2-3). On the other hand, if packet p represents a new flow it must be dropped in overload (lines 4-5). In congestion-less state it may be accepted and its flow ID is written to the PFL with probability $P1_{AFAN}$ (line 6). The congestion state is noticed based on the values of the *fair_rate* and the *priority_load* parameters estimated in the scheduler block. The admission control mechanism works identically as in the FAN versions with PFQ or PDRR algorithms. However, different methods for estimating the values of the parameters mentioned above are used.

To estimate the *priority_load* 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 *priority_load* is:

$$priority\_load = \frac{(pb(t_2) - pb(t_1)) \times 8}{C(t_2 - t_1)} \tag{6.1}$$

The *fair_rate* is computed by the following formula:

$$fair\_rate = \frac{max\{S \times C, FB \times 8\}}{t_2 - t_1} \tag{6.2}$$

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, S is the total length of inactivity in the transmission during the (t_1,t_2) period, C is the link bit rate.

The pseudo code of enqueuing and dequeuing of packets in AFAN is presented in Tab. 6.2. There is no need for implementing AFL or any additional objects.

6.2.2 Enqueue operation

If a packet selected for queuing 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 in the priority queue (lines 1-4). In the other case, the value of the ABS (*Approximate Buffer Size*) parameter is computed (line 5) form the following formula:

$$\begin{cases} ABS = (1 - w_q)ABS + w_q q & \text{if the queue is nonempty} \\ ABS = (1 - w_q)^m ABS & \text{if the queue is empty} \end{cases} \tag{6.3}$$

where w_q is queue weight, q represents the current buffer size and m is the number of packets that might have been transmitted by the router during the time during which the line was free [30] and is estimated from the following formula:

$$m = (time - q\_time)/s \tag{6.4}$$

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

If the ABS is larger 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 6-8). If the ABS is greater than or equal to the minimum threshold set for the buffer (min_th) and less 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 9-11). In practice, we do not draw a packet but a byte from the buffer and then search the packet it belongs to. It 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 $P2_{AFAN}$ (lines 12-15). If p is dropped, *count* is set to zero, if not p is selected for queuing (lines 16-18). Packet p may also be queued if its flow ID is different than flow ID of packet d or ABS is less than min_th (lines 19-21). In the latter case, the value of the *count* parameter is set to minus one (its initial value in the algorithm) (line 22). If the number of bytes of the flow the packet p selected for queuing belongs to is greater than Q, the packet is queued in the elastic queue (lines 23-25).

Table 6.2: Pseudo code of enqueuing and dequeuing of packets in AFAN

Enqueuing module:

1. on packet p accepted in MBAC
2. **If** $flow\_bytes \leq Q$ **then**
3. $Enqueue(PQ, p)$
4. $flow\_bytes = flow\_bytes + size(p)$
5. **Else** compute ABS of elastic FIFO queue
6. **If** $ABS \geq max\_th$ **then**
7. drop p
8. $count = 0$
9. **If** $min\_th \leq ABS < max\_th$ **then**
10. $count = count + 1$
11. draw packet d from the elastic FIFO queue
12. **If** flow ID of p = flow ID of d **then**
13. drop d
14. calculate probability $P2_{AFAN}$
15. drop p with probability $P2_{AFAN}$
16. **If** p is dropped **then**
17. $count = 0$
18. **Else** p selected for queuing
19. **Else** p selected for queuing
20. **If** $ABS < min\_th$ **then**
21. p selected for queuing
22. $count = -1$
23. **If** $flow\_bytes > Q$ and p selected for queuing **then**
24. $Enqueue(EQ, p)$
25. $flow\_bytes = flow\_bytes + size(p)$

Dequeuing module:

26. **While** (PQ not empty) **do**
27. $p = Dequeue(PQ)$
28. $Send(p)$
29. $flow\_bytes = flow\_bytes - size(p)$
30. **If** elastic FIFO queue not empty **then**
31. $p = Dequeue(EQ)$
32. $Send(p)$
33. $flow\_bytes = flow\_bytes - size(p)$

6.2.3 Dequeue operation

The process of selecting a packet for sending is very simple. If there are any packets in the priority queue the first of them is selected to be served (lines 26-29). If the PQ (*Priority Queue*) is empty the first packet from the elastic queue is sent (lines 30-33).

6.2.4 Complexity

The admission control complexity is the same as in the FAN implementations with PFQ or PDRR. The enqueuing 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 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 there is a need to check the buffer load and if it is full (in almost all cases in overload) 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 enqueuing process of streaming flows. In the versions of FAN with PFQ or PDRR they are served in the same way as elastic flows. In AFAN, the packets of streaming flows are queued immediately without any additional operation. It is possible because in this solution it is impossible to overload the buffer. The real gain in the algorithm complexity is noticed in AFAN while comparing the dequeuing operations for each FAN version. In the proposed solution, we have only to check if there are any packets in the priority queue and if so serve them first. There is no need to maintain the 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 arguments presented in this section confirm that AFAN is less complex than its predecessors. It is worth noting that it is also easy to implement in routers.

6.3 Calculation of the AFAN parameters

A few new parameters are provided in AFAN. They have to be set in the routers. In the proposed solution a low-pass filter to calculate the ABS is used (see Formula 6.3). This method is taken from the RED (*Random Early Detection*) queuing algorithm [30] and ensures insignificant changes of the ABS in spite of incoming bursty 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 [30] it may be assumed that the value of the w_q parameter should be set in the range from 0.001 to 0.0042. In the simulation scenarios the value of 0.002 was used, which ensured the achievement of the assumed goals.

min_th and *max_th* are the key parameters of AFAN. The optimal values of them impact 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 the sufficiently large value to ensure the link utilization at an acceptable level. On the other hand, it ought not to be too large because of packet delay demands. Following [30], 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*. In simulation runs the *min_th* parameter was set to 40 kbytes and *max_th* to 90 kbytes. The buffer size was set to 100 kbytes.

Two methods for computing $P2_{AFAN}$ are proposed in [30]. Method 2 called "uniform random variable" is described as better one. $P2_{AFAN}$ is calculated from the following formula:

$$P2_{AFAN} = P2_{AFAN_{temp}}/(1 - count \cdot P2_{AFAN_{temp}}) \tag{6.5}$$

where *count* is the number of incoming packets since last dropping operation of the algorithm. The $P2_{AFAN_{temp}}$ is calculated as the function of ABS and estimated from:

$$P2_{AFAN_{temp}} = max_p(ABS - min\_th)/(max\_th - min\_th) \tag{6.6}$$

where max_p is the maximum allowed value of $P2_{AFAN_{temp}}$.

The expected value for the method of computing the $P2_{AFAN}$ is estimated from:

$$E[X] = 1/(2P2_{AFAN}) + 1/2 \tag{6.7}$$

It means that if we set max_p to 0.02 (as was done in the simulation experiments) and the ABS is halfway between *min_th* and *max_th*, the average one of fifty arriving packets is dropped in the router. The value of max_p should be set to the values which allow for slow changes of the $P2_{AFAN}$ probability. It is needed for minimizing the fluctuations in the ABS.

6.4 Verification of AFAN

6.4.1 Simulation analysis of AFAN with EFM, RAEF and RBAEF mechanisms

In this section the results of carefully selected simulation experiments for AFAN in the ns-2 simulator are presented. They show that AFAN even though is very simple in implementation and works similarly to the FAN versions with PFQ or PDRR. The AFAN performance was evaluated by analyzing the traffic parameters during transmission in an overloaded AFAN link. The mean acceptance time of new streaming flows in the routers and the mean transmission time of the elastic flows were also analyzed. The simulation experiments in the latter case were provided for AFAN with congestion control mechanisms proposed for FAN in [23] and [24], and described in Chapter 4.

Experiment 17

150 simulation runs (50 for each FAN architecture) in various conditions were made in this simulation experiment to show the mean deviation from the *min_fair_rate* value and the mean number of flows accepted in the PFL. The simulation runs were provided in various conditions changing the number of the elastic flows. The duration of each simulation run was set to 1200 s. The rest of simulation parameters were set as in previously presented simulation experiments.

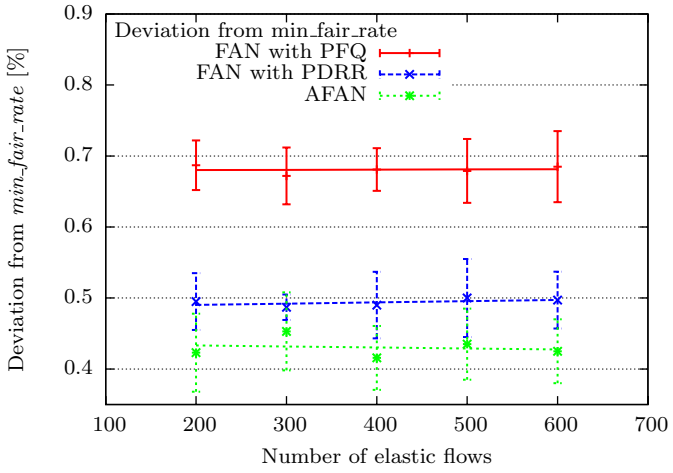

Figure 6.1: The mean deviation from the *min_fair_rate*

The mean values of deviation from the *min_fair_rate* for each FAN architecture are estimated as a fraction of the link capacity and presented in Fig. 6.1. The values of the *fair_rate* change in time and oscillate around the *min_fair_rate*. The situation is better if the deviations from the *min_fair_rate* are lower. We can see that for AFAN and FAN with PDRR the analyzed mean deviation is lower than for FAN with PFQ. It means that the former two versions are more stable than the latter one.

The mean flow count in the PFL was analyzed in the same simulation scenarios. Based on the results shown in Fig. 6.2 we may conclude that the number of flows in the PFL in AFAN is smaller than in other FAN architectures. The results obtained for the AFAN architecture are better than in other cases. We can see that all three versions provide possibility for each flow accepted in the admission control block to transmit its traffic with the rate more or less equal to the minimum acceptable fair rate value. However, in AFAN, due to the lowest number of flows accepted in the PFL, the observed values of *fair_rate* are closer to the minimum acceptable fair rate. It is an advantage of this new architecture.

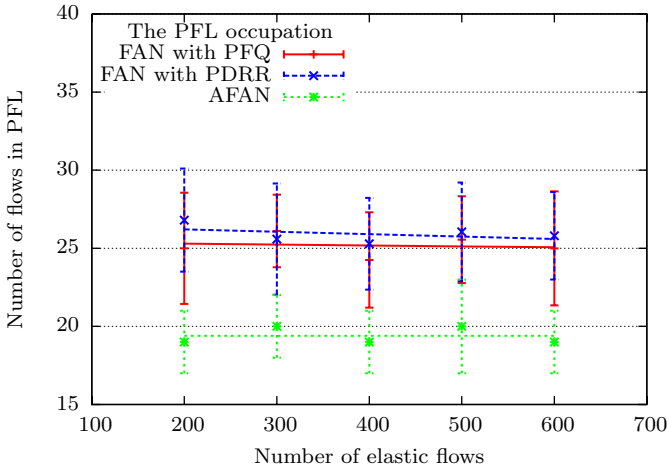

Figure 6.2: The PFL occupation in various FAN architectures

Experiment 18

In this simulation experiment 50 simulation runs were provided in various conditions (with the number of background elastic flows ranging from 200 to 600) to show the mean acceptance time of new streaming flows in the admission control block in AFAN and the mean transmission time of the elastic flows. The

simulation parameters were set as in the previous cases. A new streaming flow is accepted in the AFAN router after 82.51±8.75 s independently of the number of elastic flows in the background. As we can see, this time is not acceptable for the incoming VoIP calls and is comparable with the values obtained for two other FAN architectures. The mean transmission time of an elastic flow is 119.21±5.08 s. This value is also similar to that obtained for FAN with PFQ or PDRR.

Experiment 19

In this part of the simulation investigations presented in this dissertation the mean acceptance time of new streaming flows (*waiting_time*) and the mean transmission time of elastic flows were analyzed in AFAN with three congestion control mechanisms: EFM (*Enhanced Flushing Mechanism*), RAEF (*Remove Active Elastic Flows*) and RBAEF (*Remove and Block Active Elastic Flows*). The simulation parameters were set as in the previous cases. The results (not presented in this work) show that the mean *waiting_time* values are independent of the number of elastic flows active in background. Thus the further analysis is provided for a FAN link with 200 elastic flows active in background.

120 simulation runs (40 for each congestion control mechanism) were made to estimate two parameters mentioned above with the *pfl_flushing_timer* (for EFM) and the *active_time* (for RAEF and RBAEF) ranging from 5 to 20 s. The obtained results are presented in Fig. 6.3 and Fig. 6.4.

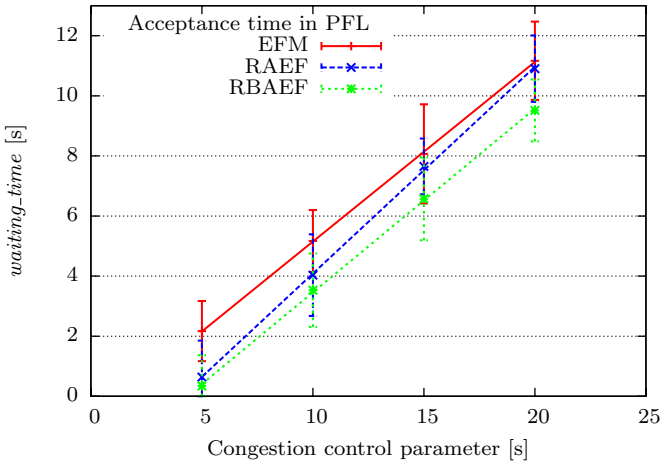

Figure 6.3: The mean *waiting_time* in AFAN with congestion control mechanisms

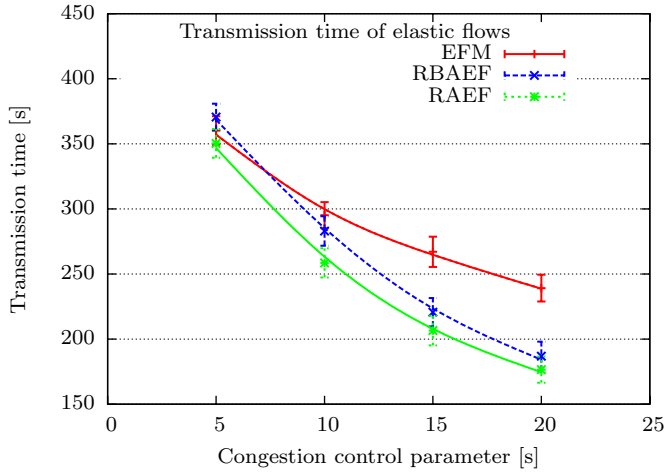

Figure 6.4: The mean transmission time of elastic flows in AFAN with congestion control mechanisms

As we can see the *waiting_time* values increase linearly for each congestion control mechanism in the examined range. The best results are observed for the RBAEF mechanism. However, we have to notice that the differences between the values of the *waiting_time* for each mechanism are small. The acceptable time for international calls should not be greater than 11 seconds for the 95% of the calls, while for the local calls it should not exceed 6 seconds [41]. If we set the congestion control parameter to 5 s or 10 s we can use any solution for the local calls. The requirements for the international calls are met in the whole examined range of the values of *pfl_flushing_timer* or *active_time*. The obtained results are comparable with the results presented for two other FAN architectures in Chapter 5.

The mean transmission time of elastic flows for AFAN with EFM may be approximated by the following function:

$$y = -86.93 \ln x + 499.96 \tag{6.8}$$

If we solve Equation (6.8) for $y = 119.21$ s (for the value of the mean transmission time of the elastic flows in the basic AFAN link) we receive the value of the *pfl_flushing_timer* $= 80.21$ s, which allows for decreasing the mean acceptance time of new streaming flows without increasing the mean transmission time of the elastic flows.

The mean transmission time of elastic flows for AFAN with RAEF may be approximated by the following function:

$$y = -126.11 \ln x + 553.04 \tag{6.9}$$

The solution of Equation (6.9) for $y = 119.21$ s is the value of the *active_time* = 31.11 s.

The mean transmission time of elastic flows for AFAN with RBAEF may be approximated by the following function:

$$y = -132.69 \ln x + 583.63 \tag{6.10}$$

The solution of Equation (6.10) for $y = 119.21$ s is the value of the *active_time* = 33.15 s.

The results of the analysis discussed above are presented in Tab. 6.3 at the end of this chapter. The results received for AFAN are comparable with those for FAN with PFQ presented in Tab. 5.4. It is possible to decrease the *waiting_time* for new streaming flows without increasing the mean transmission time of elastic flows. The results are also compared with those obtained in AFAN with RPAEF (with P_{RPAEF}=0.03).

6.4.2 Simulation analysis of AFAN with RPAEF

The AFAN with RPAEF was analyzed separately to show that this solution is the most promising one proposed and investigated in this dissertation.

Experiment 20

In this simulation experiment the *waiting_time* of streaming flows was analyzed in AFAN with RPAEF. The simulation parameters were set as in the previous simulation experiment. The number elastic flows active in background was changed from 200 to 600.

600 simulation runs (200 for each different value of P_{RPAEF}) were made to estimate the parameter mentioned above with *active_time* ranging from 5 to 20 s. The obtained results are presented in Fig. 6.5, Fig. 6.6 and Fig. 6.7.

As we can see, the *waiting_time* values are independent of the number of elastic flows being active in background, as in other FAN architectures with the RPAEF mechanism. The mean values of *waiting_time* increase with an increasing value of the *active_time* parameter. What is most interesting, the values of *waiting_time* decrease with increasing values of the P_{RPAEF} probability. If $P_{RPAEF} = 0.01$ the results are acceptable for local and international VoIP connections only if *active_time* = 5 s. In two other cases, when $P_{RPAEF} = 0.03$ or $P_{RPAEF} = 0.05$,

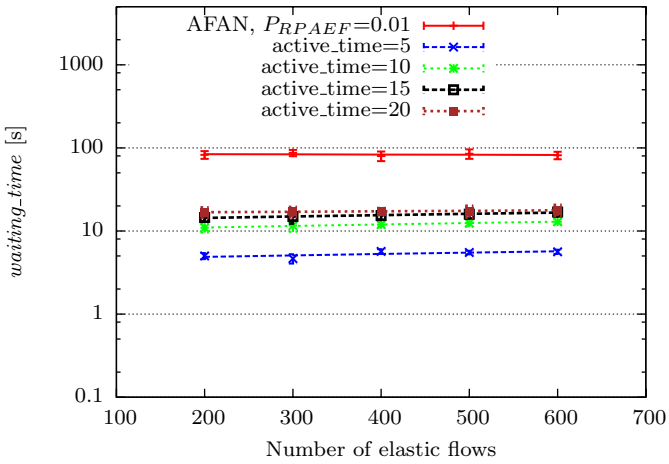

Figure 6.5: The mean *waiting_time* in AFAN with RPAEF and P_{RPAEF}=0.01

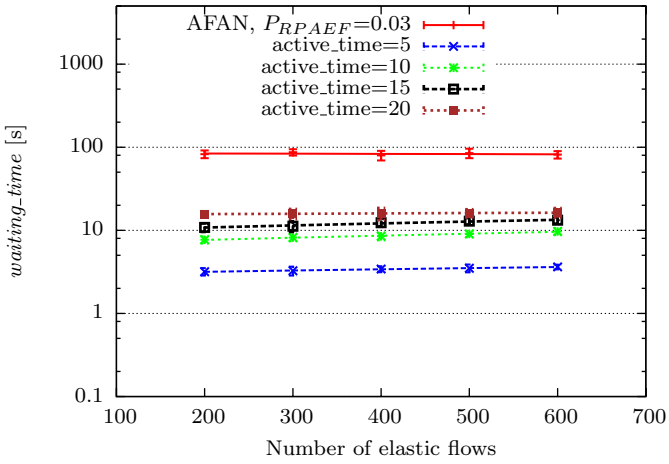

Figure 6.6: The mean *waiting_time* in AFAN with RPAEF and P_{RPAEF}=0.03

the obtained results are acceptable for local calls (if *active_time* = 5 s) and for international calls (if *active_time* = 10 s).

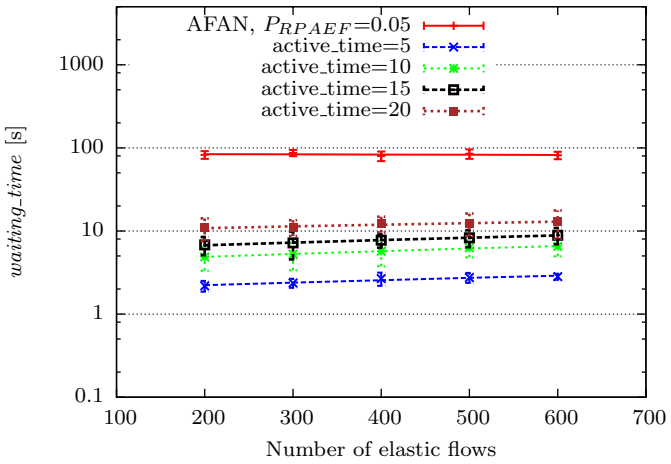

Figure 6.7: The mean *waiting_time* in AFAN with RPAEF and P_{RPAEF}=0.05

Experiment 21

In this experiment, the mean number of elastic flows accepted after any flushing action in AFAN with RPAEF is analyzed. The simulation duration was set to 500 s, which allows for estimating the observed parameter in a steady state for sufficiently long time. The rest of the simulation parameters were set as in the previous case.

The values of the mean number of elastic flows in function of the number of elastic flows active in background for AFAN with RPAEF are presented in Fig. 6.8, Fig. 6.9 and Fig. 6.10.

The number of elastic flows accepted in AFAN routers after a cleaning action of the PFL content is significantly lower than for other FAN architectures and congestion control mechanisms. The results obtained for $P_{RPAEF} = 0.01$ and $P_{RPAEF} = 0.03$ are similar, while the mean values of accepted elastic flows are greater when $P_{RPAEF} = 0.05$. The mean *waiting_time* values are smaller for $P_{RPAEF} = 0.03$ than for $P_{RPAEF} = 0.01$.

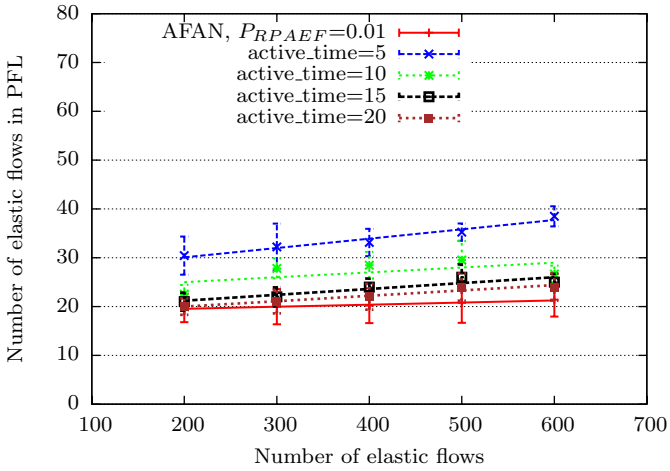

Figure 6.8: The mean number of elastic flows in PFL in AFAN with RPAEF and P_{RPAEF}=0.01

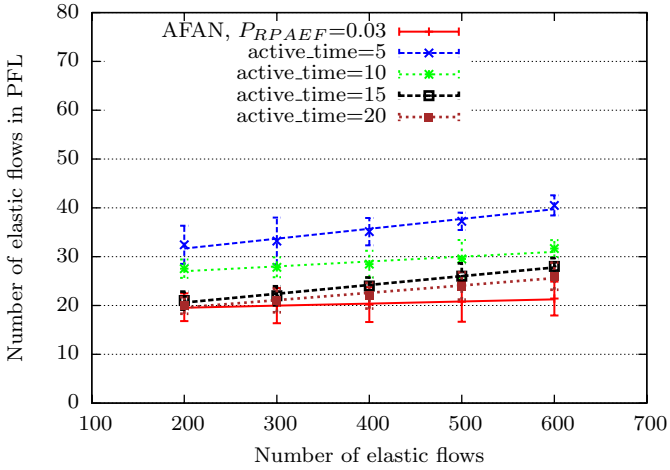

Figure 6.9: The mean number of elastic flows in PFL in AFAN with RPAEF and P_{RPAEF}=0.02

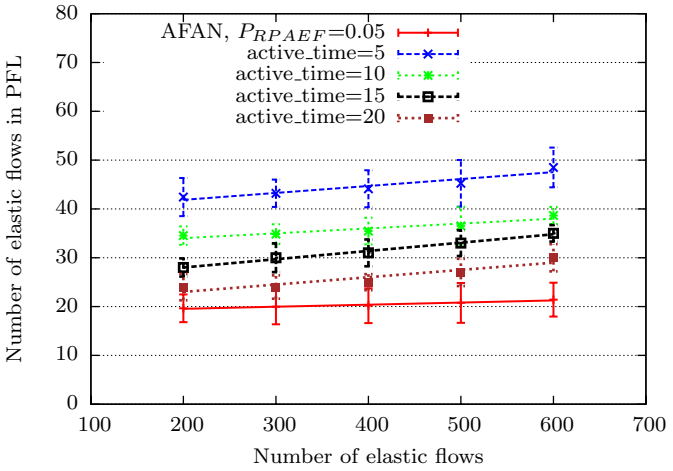

Figure 6.10: The mean number of elastic flows in PFL in AFAN with RPAEF and P_{RPAEF}=0.05

Experiment 22

In this simulation experiment the mean transmission time of elastic flows for AFAN with RPAEF is analyzed. The number of elastic flows active in background was set to 200 and the simulation time was set to 1500 s. The rest of the simulation parameters was set as in the previous case. The results are presented in Fig. 6.11.

The mean transmission time of elastic flows for AFAN with RPAEF and $P_{RPAEF} = 0.01$ may be approximated by the following function:

$$y = -14.39 \ln x + 163.17 \tag{6.11}$$

If we solve Equation (6.11) for $y = 119.21$ s (for the value of the mean transmission time of the elastic flows in the basic AFAN link) we receive the value of the *active_time* $= 21.02$ s, which allows for decreasing the mean acceptance time of new streaming flows without increasing the mean transmission time of the elastic flows.

The mean transmission time of elastic flows for AFAN with RPAEF and $P_{RPAEF} = 0.03$ may be approximated by the following function:

$$y = -17.99 \ln x + 173.96 \tag{6.12}$$

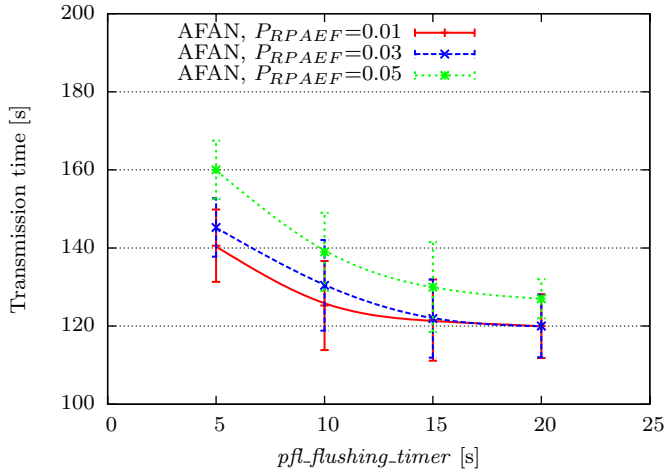

Figure 6.11: The mean transmission time of elastic flows in AFAN with RPAEF

The solution of Equation (6.12) for $y = 119.21$ s is the value of the *active_time* = 20.77 s.

The mean transmission time of elastic flows for AFAN with RPAEF and $P_{RPAEF} = 0.05$ may be approximated by the following function:

$$y = -23.74 \ln x + 198.22 \tag{6.13}$$

The solution of Equation (6.13) for $y = 119.21$ s is the value of the *active_time* = 27.60 s.

The results of the analysis discussed above are presented in Tab. 6.3. The results received for AFAN are comparable with those for FAN with PFQ or with PDRR presented in Tab. 5.4. It is possible to decrease the *waiting_time* for new streaming flows without increasing the mean transmission time of elastic flows.

Experiment 23

The results of simulation experiments presented in this dissertation show that it is possible to significantly decrease the mean value of the *waiting_time* parameter without increasing the mean transmission time of elastic flows. In many cases the results are acceptable for international VoIP connections. Unfortunately, it is impossible to ensure low enough acceptance times for local calls without deteriorating the performance of elastic flows.

If we analyze the results presented in Fig. 6.11 we will see that the mean transmission time of elastic flows for *active_time* = 10 s is comparable with the

Table 6.3: Transmission parameters for basic AFAN link and with the EFM, RAEF, RBAEF and RPAEF

Mechanism / Parameter	**Value of parameter [s]**	***waiting_time* [s]**	**Transmission time period of elastic flows [s]**
AFAN			
Basic FAN	-	82.51±8.75	119.21±5.08
EFM / *pfl_flushing_timer*	80.21	27.48±2.85	121.45±7.17
RAEF / *active_time*	31.11	13.65±2.86	127.51±11.84
RBAEF / *active_time*	33.15	16.35±2.87	127.55±9.24
RPAEF / *active_time*	20.77	16.01±1.95	119.88±10.22

value obtained for the basic AFAN. In such a case the mean *waiting_time* value is equal to 7.70±0.75 s. This value is still too large for local voice calls.

In this last simulation experiment the *waiting_time* parameter and mean transmission time of elastic flows were analyzed in AFAN with RPAEF and with *active_time* set to 8 s and $P_{RPAEF} = 0.03$. The number of elastic flows active in background was set to 200. The duration of simulation runs was set to 200 s.

The results of 10 simulation runs show that the mean *waiting_time* is equal to 5.62±0.32 s what is an acceptable value for local and international VoIP connections. The mean transmission time is equal to 135.86±12.82 s. This value is comparable with those obtained for basic AFAN.

We may conclude that in AFAN with RPAEF the value of $P_{RPAEF} = 0.03$ allows for ensuring a satisfactory traffic performance for the analyzed traffic types.

Part IV

Finale

7 Conclusions

The first goal of the research presented in the dissertation was to propose congestion control mechanisms for Flow-Aware Networks (FAN). The four mechanisms are presented and deeply analyzed. The theoretical and simulation analysis is provided. The simulation analysis was made by using the ns-2 simulator.

The mechanisms: EFM (*Enhanced Flushing Mechanism*), RAEF (*Remove Active Elastic Flows*), RBAEF (*Remove and Block Active Elastic Flows*) and RPAEF (*Remove and Prioritize in access Active Elastic Flows*) work based on periodical partial or total cleaning the PFL (*Protected Flow List*) content. In FAN it is impossible to accept any new flow in the router in congestion. It may cause that a FAN link may be occupied by a finite number of big (called also elephant) flows giving no chance for new flows to begin their transmission for a long time. The acceptance time in the AC (admission control) block of the FAN router of the VoIP calls was analyzed in particular. Such connections have to be quickly accepted in the routers and their transmission cannot be broken or delayed.

In the EFM, the IDs of all elastic flows are periodically removed from the PFL in congestion. The minimum time period between two flushing actions is provided by the *pfl_flushing_timer* parameter. The mechanism ensures low *waiting_time* values for streaming flows but significantly deteriorates the performance of elastic flows. In FAN with PDRR it is even impossible to decrease the acceptance time of new streaming flows without increasing the mean transmission time of elastic flows.

The RAEF mechanism is proposed as an alternative to the EFM. In this proposal, the transmission of elastic flows active for at least *active_time* is broken and their IDs are removed from the PFL each time the congestion is noticed.

The mechanism gives better results than the EFM. The main problem of the EFM and the RAEF proposals is that the number of elastic flows accepted in the router after a flushing action is too large, which means that for low values of the *pfl_flushing_timer* or the *active_time* the mechanisms are not scalable.

In the RBAEF mechanism the number of elastic flows accepted in the AC block after flushing was decreased in comparison to the EFM and RAEF. The values of the *waiting_time* and the mean transmission time of elastic flows are similar to those obtained for the RAEF mechanism. The main drawback of this solution is that the transmission of elastic flows is broken for a fixed time, which does not allow for decreasing the mean transmission time of elastic flows to the acceptable level for low values of the *active_time* parameter.

The RPAEF mechanism looks to be the best of all proposed solutions in this dissertation. The IDs of elastic flows active for at least *active_time* are removed from the PFL after any flushing action. They are not blocked but receive priority in access to the router. For a short time after flushing (*priority_access*) they are always accepted in the congestion-less state, while the rest of flows are accepted with low priority (R_{RPAEF}) during the same time period. It allows for decreasing the mean transmission time of elastic flows even for low values of the *active_time* parameter. Moreover, the transmission of elastic flows is broken for a short time and the number of elastic flows accepted in the router after flushing is lower than in the other mechanisms. This guaranties better scalability properties and looks to be a strong advantage of this solution.

The new methods for estimating the *fair_rate* and the *priority_load* parameters were also proposed during the analysis of FAN. They allow for more stable transmission in FAN and describe the traffic transmission in a more realistic way. It was shown that the smoothing parameters for estimating the *fair_rate* and the *priority_load* values should be set to the low values or may be even not used.

The second goal of this dissertation was to propose a new architecture of FAN, called AFAN (*Approximate Flow-Aware Networking*). The AFAN concept is based on the Approximate Fair Dropping mechanism, which uses the RED algorithm to control buffer occupation. The AFAN is simple and ensures implicit service differentiation, fairness for elastic flows and high priority for the streaming ones. Two main blocks of the FAN architecture, the admission control and the scheduler, are also used in AFAN what allows for proper transmission of flows with a minimal knowledge acquired from the network.

The comparison of AFAN with two other FAN architectures (with PFQ and PDRR scheduling algorithms) shows that the enqueue and dequeue operations are realized in a simpler way in this new solution. The most important issue to note is that there is no need for implementing the AFL in AFAN. The traffic analysis in the new architecture shows that AFAN works similarly to the other FAN versions.

Moreover, the oscillations around the *min_fair_rate* value are slightly lower than in the other cases what may indicate that transmission in AFAN is more stable.

The congestion control mechanisms proposed for FAN (EFM, RAEF, RBAEF and RPAEF) may also be used for AFAN and may play an important role in the traffic control process. The simulation results show that it is possible to decrease the mean acceptance time of the streaming flows to the level acceptable for local VoIP calls. Moreover, it is possible to significantly decrease the *waiting_time* values without increasing the mean transmission time of elastic flows.

It was shown that in AFAN with the RPAEF mechanism it is possible to ensure a low acceptance time for streaming flows (acceptable for local and international VoIP connections) without increasing the mean transmission time of elastic flows. It was impossible in any other architecture.

The AFAN architecture looks to be a good answer to the disadvantages of the currently used QoS architectures, which are complicated and, in many cases, do not work as expected.

Achievements and contributions

The achievements and contributions of the dissertation can be summarized as follows:

1. An in-depth analysis of Flow-Aware Networks is provided in the dissertation. The advantages and drawbacks of FAN are described.

2. New methods for estimating the *fair_rate* and *priority_load* parameters are proposed. They allow for more stable transmission in FAN and are more appropriate when analyzing the network performance.

3. The *Enhanced Flushing Mechanism* is proposed to improve transmission of streaming flows in congestion. Its architecture is described in details. The simulation analysis is provided for three FAN architectures, with PFQ or PDRR, and in AFAN.

4. The *Remove Active Elastic Flows* mechanism was presented in details as an alternative to the EFM. It allows for ensuring better transmission parameters than the EFM. The simulation results show its usefulness in three FAN architectures.

5. The *Remove and Block Active Elastic Flows* mechanism was proposed to improve the scalability problems of EFM and RAEF mechanisms. While the goal was achieved, the transmission parameters of two mechanisms mentioned earlier were similar. The elastic flows, rejected after flushing, have to

be blocked while accessing the router, for a short time. As a consequence, it is not possible to achieve the desirable transmission parameters.

6. The *Remove and Prioritize in access Active Elastic Flows* mechanism ensures quick acceptance times of streaming flows, simultaneously achieving the transmission performance of elastic flows on the acceptable level. It also ensures good scalability even for low values of the *active_time* parameter. The outages in transmission of elastic flows are short. The rejected flows have high priority in access to the router after flushing, while the rest of flows is accepted with small probability at that time. It ensures that the number of flows accepted after a flushing action is limited.

7. The analysis of congestion control mechanisms proposed for FAN is presented in case of link failure. It was shown that the solutions described in this work allow for a significant improvement of the transmission performance in the network after failure.

8. The new version of FAN, called *Approximate Flow-Aware Networking* was proposed in this work. This architecture, simpler than its two predecessors, realizes the Flow-Aware Networking concept. The queuing method is based on the *Approximate Fair Dropping* algorithm. It is shown that this new solution is less complex than FAN with PFQ or PDRR.

9. The analysis of EFM, RAEF, RBAEF and RPAEF mechanisms in AFAN architecture was provided. The results show that they work with a satisfactory results, as in the other FAN architectures.

10. The AFAN architecture with RPAEF allows for ensuring the acceptance times for local and international VoIP calls without deteriorating the transmission parameters of elastic flows.

In the light of the presented achievements it can be stated that the thesis: ***It is possible to define efficient and simple congestion control mechanisms in Flow-Aware Networks***, has been proved.

Appendix

A Simulation scenario in TCL code

The appendix presents a sample code of the simulation scenario. This is an example of a TCL script. The comments are provided to explain the meaning of parameters.

Simulation scenario in TCL code

```
#########################################

set pfl_flushing_timer 5     # parameter set in EFM in [s]
set active_time 5            # parameter set in RBAEF, RAEF and RPAEF in [s]
set blocked_period 1         # parameter set in RBAEF in [s]
set priority_access 1        # parameter set in RPAEF in [s]

#########################################
############## AFAN PARAMETERS ##############

set w_q 0.004                # weight used in computing ABS
set min_th 400000            # minimum threshold of the buffer [bytes]
set max_th 900000            # maximum threshold of the buffer [bytes]
set max_p 0.02               # probability used in computing ABS
set b_leng 1000000           # buffer length [bytes]

#########################################
```

```
#########################################

set warm_up_time 20                    # warm-up time in [s]
set QueueLength 1000                   # buffer length for PFQ and PDRR in [packets]
set C 10000000                         # link capacity in [bit/s]

set FlowAvgSize 150000000              # average flow size (Pareto distribution) in [bit/s]
set FlowShape 1.5                      # flow shape (Pareto distribution)

set NSrc 1                             # number of source nodes
set NumFlowPerSrc 600                  # number of flows generated in each source node
set VoIPSrc 20                         # number of VoIP sources

#########################################

set PriorityLoadInterval 0.05          # measurement interval of priority_load in [s]
set FairRateInterval 0.5               # measurement interval of fair_rate in [s]
set alpha 0.1                          # smoothing param. α for computing fair_rate
set beta 0                             # smoothing param. β for computing priority_load
set pfl_probability 1                  # acceptance probability in MBAC
set FlowTimeOut 20                     # timeout of flows in the PFL in [s]
set MinFairRate 5                      # the min_fair_rate in % of link capacity
set MaxPriorityLoad 0.7                # the max_priority_load as a part of link capacity

#########################################
################## PREFACE ##############

set ns [new Simulator]

set nf [open out.nam w]                # for NAM visualization
$ns namtrace-all $nf

#########################################
################## NETWORK ##############

### Nodes
for {set j 1} {$j <= $NSrc} {incr j} {   # create source nodes of elastic traffic
  set S($j) [$ns node]
}

### Nodes for bottleneck link
set n3 [$ns node]                      # first node of FAN link
set n4 [$ns node]                      # second node of FAN link
```

```
### Links
for {set j 1} {$j <= $NSrc} {incr j} {
   $ns duplex-link $S($j) $n3 100Gb 1ms DropTail          # links sources ↔ bottleneck
}

### Bottleneck Link
$ns duplex-link $n3 $n4 $C 1ms FAN/PFQ                    # FAN link

### Store queue class in variable
set queue34_ [[$ns link $n3 $n4] queue]
set MFN [expr $NSrc$NumFlowPerSrc+$VoIPSrc]               # maximum flow number

### Admission Control
set agent34 [new FAN/AdmissionControl/PFQAgent]           # initialization of AC agent
$agent34 set probability $pfl_probability
$agent34 set flowTimeout $FlowTimeOut
$agent34 set Blocked_Timeout $blocked_period
$agent34 set timer_flush_pfl $pfl_flushing_timer
$agent34 set diff $active_time
$agent34 set min_fair_rate $MinFairRate
$agent34 set max_priority_load $MaxPriorityLoad
$agent34 set maxFlowsNumber $MFN
$agent34 createFlowList

### set Admission Control agent
$queue34_ attach-agent $agent34

### queue34_
$queue34_ set max_pifo_length $QueueLength
$queue34_ set priority_load_interval $PriorityLoadInterval
$queue34_ set alpha $alpha
$queue34_ set C $C
$queue34_ set fair_rate_interval $FairRateInterval
$queue34_ set beta $beta
$queue34_ set fair_rate $C                                # initial value of fair_rate
$queue34_ set maxFlowsNumber $MFN
$queue34_ set w_q $w_q                                    # for AFAN
$queue34_ set minth $min_th                               # for AFAN
$queue34_ set maxth $max_th                               # for AFAN
$queue34_ set maxp $max_p                                 # for AFAN
$queue34_ set q_l $b_leng                                 # for AFAN
$queue34_ createFlowList
```

```
set queue43_ [[$ns link $n4 $n3] queue]
set agent43 [new FAN/AdmissionControl/YesAgent]
$queue43_ attach-agent $agent43

### queue43_
$queue43_ set max_pifo_length $QueueLength
$queue43_ set priority_load_interval $PriorityLoadInterval
$queue43_ set alpha $alpha
$queue43_ set C $C
$queue43_ set fair_rate_interval $FairRateInterval
$queue43_ set beta $beta
$queue43_ set fair_rate $C                    # initial value of fair_rate
$queue43_ set maxFlowsNumber $MFN
$queue43_ set w_q $w_q                        # for AFAN
$queue43_ set minth $min_th                   # for AFAN
$queue43_ set maxth $max_th                   # for AFAN
$queue43_ set maxp $max_p                     # for AFAN
$queue43_ set q_l $b_leng                     # for AFAN
$queue43_ createFlowList
#########################################
################## TRACING ###############

### Tracing PFQ queue
set tchan_ [open $trace_file_name w]
$queue34_ trace enqueued
$queue34_ trace dequeued
$queue34_ trace dropped
$queue34_ trace priority_load
$queue34_ trace fair_rate
$queue34_ trace flow_count
$queue34_ trace ACflow_count
$queue34_ attach-trace_file $tchan_

### Monitor the queue for link (n3-n4) for NAM
$ns duplex-link-op $n3 $n4 queuePos 0.5

#########################################
################## TRAFFIC ###############

### Elastic traffic

set k 1
for {set i 1}{$i<=NSrc}{incr i}{
for {set j 1}{$j<=[expr $NumFlowPerSrc]}{incr j}{
```

```
set tcpsrc($i,$j)[new Agent/TCP/Newreno]
$ns attach-agent $S($i) $tcpsrc($i,$j)
$tcpsrc($i,$j) set fid_ $k
$tcpsrc($i,$j) set prio_ 0
$tcpsrc($i,$j) set packetSize_ 1000
$tcpsrc($i,$j) set window_ 500
incr k

set tcpdest($i,$j)[new Agent/TCPSink]
$ns attach-agent $n4 $tcpdest($i,$j)
$ns connect $tcpsrc($i,$j) $tcpdest($i,$j)

set ftp($i,$j)[new Application/FTP]
$ftp($i,$j) attach-agent $tcpsrc($i,$j)
$ftp($i,$j) set type_ FTP
}
}

### VoIP traffic

for {set i [expr $NumFlowPerSrc+1]}{$i<=[expr $NumFlowPerSrc+$VoIPSrc]} {incr i} {
set udp($i)[new Agent/UDP]
$ns attach-agent $S(1) $udp($i)
set Sink($i) [new Agent/LossMonitor]
$ns attach-agent $n4 $Sink($i)
$ns connect $udp($i) $Sink($i)
$udp($i) set fid_ $i

### Setup a FTP over TCP connection
set cbr($i) [new Application/Traffic/CBR
$cbr($i) attach-agent $udp($i)
$cbr($i) set type_ CBR
$cbr($i) set packet_size_ 100
$cbr($i) set rate_ 0.08mb
$cbr($i) set random_ true
}

for {set i [expr $NumFlowPerSrc+1]} {$i<=[expr $NumFlowPerSrc+$VoIPSrc]} {incr i} {
$ns at [expr 50+$i-$NumFlowPerSrc+rand()] "$cbr($i) start"
$ns at [expr 680+5*($i-$NumFlowPerSrc)] "$cbr($i) stop"
}
```

```
#########################################
################ SIMULATION ###############
### Generator for random flow size
set run 1
if {$argc == 1} {
set run [lindex $argv 0]
}
if {$run < 1} {
set run 1
}
### Seed the default RNG
global defaultRNG
$defaultRNG seed 9999

set rng1 [new RNG]
set tng2 [new RNG]
for {set j 1} {$j < $run} {incr j} {
$rng1 next-substream
$rng2 next-substream
}

### Random variable for size
set RV_HT [new RandomVariable/Pareto]
$RV_HT set avg_ $FlowAvgSize
$RV_HT use-rng $rng1

### Random variable for intervals of TCP flows
set RV_IA [new RandomVariable/Exponential]
$RV_IA set avg_ 0.1
$RV_IA use-rng $rng2

for {set i 1} {$i<=$NSrc} {incr i} {
set maxst($i) 0
set t [$ns now]
for {set j 1} {$j<=$NumFlowPerSrc} {incr j} {
$ns attach-agent $n4 $Sink($i)
set t [expr $t + [$RV_IA value]]
set conn_start($i,$j) [expr $t]
set size [expr [$RV_HT value]]
set conn_size($i,$j) $size]
}
}

set minmaxst 1000
```

```
for {set i 1} {$i <= NSrc} {incr i} {
if {$maxst($i) < $minmaxst} {
set minmaxst $maxst($i)
}
}
set minmaxst 200
set maxft [expr $minmaxst]

set startMeasure $warmUpTime
set stopMeasure $minmaxst

$agent34 set meas_start $startMeasure
$agent34 set meas_stop $stopMeasure

for {set i 1} {$i <= $NSrc} {incr i} {
for {set j 1} {$j <= $NumFlowPerSrc} {incr j} {
$ns at $conn_start($i,$j) "$ftp($i,$j) send $conn_size($i,$j)"
}
}
set nbrpkt 1
set t 0.0

########################################
################ PROCEDURES ###############

proc finish {} {

global ns tchan_ NSrc NumFlowPerSrc udpdest trace_file_name queue_name
agent34 queue34_file_thrgh ftimes conn_end fc nf
$ns flush-trace

### Close the NAM trace file
close $nf

### Execute NAM on the trace file
exec nam out.nam &

close throughput file
close $file_thrgh

puts $ftimes "ends"
set l 1
for {set i 1} {$i <= $NSrc} {incr i} {
for {set j 1} {$j <= $NumFlowPerSrc} {incr j} {
```

```
puts $times "$1 $conn_end($i,$j)"
incr l
}
}
close $ftimes

set awkCode {
{
if ($1 == "enqueued" && NF > 2) {
print $2,$3 >> "temp.enq";
set end $2
}
else if($1 == "dequeued" && NF>2)
print $2,$3 >> "temp.deq";
else if($1 == "dropped" && NF>2)
print $2,$3 >> "temp.dro";
else if($1 == "droppedAC" && NF>2)
print $2,$3 >> "temp.droac";
else if($1 == "droppedQueue" && NF>2)
print $2,$3 >> "temp.droqu";
else if($1 == "priority_load" && NF>2)
print $2,$3 >> "temp.pl";
else if($1 == "fair_rate" && NF>2)
print $2,$3 >> "temp.fr";
else if($1 == "flow_count" && NF>2)
print $2,$3 >> "temp.fc";
else if($1 == "ACflow_count" && NF>2)
print $2,$3 >> "temp.acfc";
}
}

set f [open plot.txt w]
puts $f "TitleText: $queue_name traffic"
puts $f "Device: Postscript"

set f1 [open pl.txt w]
puts $f1 "TitleText: $queue_name priority_load"
puts $f1 "Device: Postscript"

set f2 [open fr.txt w]
puts $f2 "TitleText: $queue_name fair_rate"
puts $f2 "Device: Postscript"

set f3 [open fc.txt w]
```

```
puts $f3 “TitleText: $queue_name flow_count”
puts $f3 “Device: Postscript”
set f4 [open acfc.txt w]
puts $f4 “TitleText: $queue_name AC_flow_count”
puts $f4 “Device: Postscript”

if {[info exists tchan_]} {
close $tchan_
}
print $2,$3 >> “temp.enq”;
exec rm -f temp.enq temp.deq temp.dro temp.droac temp.droqu temp.pl temp.fr
temp.fc temp.acfc
exec touch temp.enq temp.deq temp.dro temp.droac temp.droqu temp.pl temp.fr
temp.fc temp.acfc
exec awk $awkCode $trace_file_name
puts $f “enque”
exec cat temp.enq > $f
puts $f “deque”
exec cat temp.deq > $f
puts $f “drop”
exec cat temp.dro > $f
puts $f “drop AC”
exec cat temp.droac > $f
puts $f “drop Queue”
exec cat temp.droqu > $f
puts $f “priority_load”
exec cat temp.pl > $f
puts $f “fair_rate”
exec cat temp.fr > $f
puts $f “flow_count”
exec cat temp.fc > $f
puts $f “AC_flow_count”
exec cat temp.acfc > $f
close $f
close $f1
close $f2
close $f3
close $f4
close $fc

exec xgraph -bb -bg white -tk -x time -y bytes/sec plot.txt &
exec xgraph -bb -bg white -tk -x time -y value pl.txt &
exec xgraph -bb -bg white -tk -x time -y value -ly 0,10000000 fr.txt &
exec xgraph -bb -bg white -tk -x time -y value fc.txt &
```

```
exec xgraph -bb -bg white -tk -x time -y value acfc.txt &
set cr_f [open criteria.txt a+]
set afn [$agent34 set all_flows_number]
set adfn [$agent34 set admitted flows_number]
puts $cr_f "all_flows_number $afn"
puts $cr_f "admitted_flows_number $adfn"
if {$afn > 0} {
puts $cr_f "flows_blocking_probability [expr 1.0-double($adfn)/double($afn)]"
}
puts $cr_f "queue_packet_lost [$queue34_ set queue_packet_lost]"
puts $cr_f "AC_packet_lost [$queue34_ set AC_packet_lost]"
puts $cr_f "FRAverage [$queue34_ set FRAverage]"
puts $cr_f "FRVariance [$queue34_ set FRVariance]"
puts $cr_f "FRstdev [expr sqrt([$queue34_ set FRVariance])]"
puts $cr_f "FRMinimum [$queue34_ set FRMinimum]"
puts $cr_f "FRMaximum [$queue34_ set FRMaximum]"
puts $cr_f "PLAverage [$queue34_ set PLAverage]"
puts $cr_f "PLVariance [$queue34_ set PLVariance]"
puts $cr_f "PLstdev [expr sqrt([$queue34_ set PLVariance])]"
puts $cr_f "PLMinimum [$queue34_ set PLMinimum]"
puts $cr_f "PLMaximum [$queue34_ set PLMaximum]"
close $cr_f

set a [exec date]
puts $a
exit 0
}
### Running simulation
$ns at $maxft "finish"

$ns run
```

Bibliography

[1] J. Adams, A. IJsselmuiden, and L. Roberts. An advanced QoS protocol for real-time content over the internet. In H. de Meer and N. Bhatti, editors, *Proc. 13th International Workshop on Quality of Service (IWQoS)*, volume 3552 of *Lecture Notes in Computer Science*, pages 164–177, Passau, Germany, Jun. 2005. Springer-Verlag. *(Cited on page 29.)*

[2] G. Appenzeller, I. Keslassy, and N. McKeown. Sizing router buffers. In *Proceedings of International Conference on Measurement and modeling of computer systems, ACM SIGMETRICS 2004*, Oregon, USA, August 2004. *(Cited on page 31.)*

[3] 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. *(Cited on pages 31 and 51.)*

[4] J. Banks, J. Carson, B. Nelson, and D. Nicol. *Discrete-event system simulation – fourth edition.* Pearson, 2005. *(Cited on page 50.)*

[5] N. Benameur, S. Ben Fredj, F. Delcoigne, S. Oueslati-Boulahia, and J.W. Roberts. Integrated Admission Control for Streaming and Elastic Traffic. In *Proceedings of Second International Workshop on Quality of Future Internet Services, QofIS 2001*, Coimbra, Portugal, September 2001. *(Cited on page 33.)*

[6] N. Benameur, S. Ben Fredj, S. Oueslati-Boulahia, and J.W. Roberts. Quality of Service and flow level admission control in the Internet. *Computer Networks*, 40:57–71, August 2002. *(Cited on page 33.)*

[7] N. Benameur, A. Kortebi, S. Oueslati, and J. Roberts. Selective protection in overload: differentiated services or per-flow admission control? In *Proceedings of Networks 2004*, Vienna, Austria, June 2004. *(Cited on page 33.)*

[8] J. Bennet and H. Zhang. WF2Q: Worst-case Fair Weighted Fair Queuing. In *Proceedings of The Conference on Computer Communications, IEEE INFOCOM '96*, San Francisco, USA, March 1996. *(Cited on page 12.)*

[9] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An Architecture for Differentiated Services. IETF RFC 2475, December 1998. *(Cited on pages 3 and 10.)*

[10] T. Bonald, S. Oueslati-Boulahia, and J. Roberts. IP traffic and QoS control. In *Proceedings of World Telecommunications Congress, WTC 2002*, September 2002. *(Cited on page 28.)*

[11] T. Bonald and J. Roberts. Performance modeling of elastic traffic in overload. In *Proceedings of International Conference on Measurement and Modeling of Computer Systems, ACM SIGMETRICS 2001*, Cambridge, MA, USA, June 2001. *(Cited on page 32.)*

[12] T. Bonald and J. Roberts. Congestion at flow level and the impact of user behaviour. *Computer Networks*, 42:521–536, 2003. *(Cited on page 35.)*

[13] N. Bonmariage and G. Leduc. A survey of optimal network congestion control for unicast and multicast transmission. *Computer Networks*, 50:448–468, 2006. *(Cited on page 35.)*

[14] R. Braden, D. Clark, and S. Shenker. Integrated Services in the Internet Architecture an Overview. IETF RFC 1633, June 1994. *(Cited on page 3.)*

[15] C. Cardenas, M. Gagnaire, V. Lopez, and J. Aracil. Admission control for Grid services in IP networks. In *Proceedings of Advanced Networks and Telecommunication Systems, ANTS 2007*, Bombay, India, December 2007. *(Cited on page 33.)*

[16] C. Cardenas, M. Gagnaire, V. Lopez, and J. Aracil. Performance evaluation of the Flow-Aware Networking (FAN) architecture under Grid environment. In *Proceedings of IEEE Network Operations and Management Symposium, NOMS 2008*, pages 481–487, Paris, France, April 2008. *(Cited on page 30.)*

[17] B. Carpenter and K. Nichols. Differentiated Services in the Internet. *Proceedings of the IEEE*, 90, Sep. 2002. *(Cited on page 10.)*

[18] D-M. Chiu and R. Jain. Analysis of the Increase and Decrease Algorithms for Congestion Avoidance in Computer Networks. *Journal of Computer Networks and ISDN*, 17:1–14, June 1989. *(Cited on pages xv and 24.)*

[19] D.-M. Chiu and R. Jain. Analysis of the Increase and Decrease Algorithms for Congestion Avoidance in Computer Networks. *Computer Networks and ISDN Systems*, 17:1–14, 1989. *(Cited on page 36.)*

[20] E. Crawley, R. Nair, B. Rajagoplan, and H. Sandick. A Framework for QoS-based Routing in the Internet. *IETF RFC 2386*, Aug. 1998. *(Cited on page 10.)*

[21] B. Davie, A. Charny, J. C. R. Bennett, K. Benson, J. Y. Le Boudec, W. Courtney, S. Davari, V. Firoiu, and D. Stiliadis. An expedited forwarding PHB. *IETF RFC 3246*, Mar. 2002. *(Cited on page 11.)*

[22] S. Deering and R. Hinden. Internet Protocol, version 6 (IPv6). *IETF RFC 2460*, Dec. 1998. *(Cited on page 10.)*

[23] J. Domzal and A. Jajszczyk. New Congestion Control Mechanisms for Flow-Aware Networks. In *Proceedings of International Conference on Communications, ICC 2008*, Beijing, China, May 2008. *(Cited on pages 4, 5, 6, 39, 40 and 99.)*

[24] J. Domzal and A. Jajszczyk. The Flushing Mechanism for MBAC in Flow-Aware Networks. In *Proceedings of 4th EURO-NGI Conference on Next Generation Internet Networks, NGI 2008*, pages 77–83, Krakow, Poland, April 2008. *(Cited on pages 4, 5, 6, 39, 40 and 99.)*

[25] J. Domzal and A. Jajszczyk. The Impact of Congestion Control Mechanisms for Flow-Aware Networks on Traffic Assignment in Two Router Architectures. In *Proceedings of International Conference on the Latest Advances in Networks, ICLAN 2008*, Toulouse, France, December 2008. *(Cited on pages 5 and 6.)*

[26] J. Domzal, R. Wojcik, and A. Jajszczyk. The Impact of Congestion Control Mechanisms on Network Performance after Failure in Flow-Aware Networks. In *Proceedings of International Workshop on Traffic Management and Traffic Engineering for the Future Internet, FITraMEn 2008*, Porto, Portugal, December 2008. *(Cited on page 6.)*

[27] L. Drzewiecki and M. Antoniak-Lewandowska. Flow Simulator - a flow-based network simulator. In *Proceedings of The International Conference on "Computer as a Tool", EUROCON 2007*, Warsaw, Poland, September 2007. *(Cited on page 50.)*

[28] ETSI Technical Report ETR003 second edition. *Network Aspects (NA); General Aspects of Quality of Service (QoS) and Network Performance (NP)*, Oct. 1994. *(Cited on page 10.)*

[29] A. Ferragut, D. Kofman, F. Larroca, and S. Oueslati. Design and Analysis of Flow Aware Load Balancing Mechanisms for Multi-Service Networks. In *Proceedings of 4th EURO-NGI Conference on Next Generation Internet Networks, NGI 2008*, pages 84–91, Krakow, Poland, April 2008. *(Cited on page 29.)*

[30] S. Floyd and V. Jacobson. Random Early Detection Gateways for Congestion Avoidance. *IEEE/ACM Transactions on Networking*, 1:397–413, August 1993. *(Cited on pages 11, 36, 95, 97 and 98.)*

[31] S. Ben Fredj, T. Bonald, A. Proutiere, G. Regnie, and J.W. Roberts. Statistical bandwidth sharing: a study of congestion at flow level. In *Proceedings of Special Interest Group on Data Communication, SIGCOMM 2001*, San Diego, CA, USA, August 2001. *(Cited on page 31.)*

[32] S. Ben Fredj, S. Oueslati-Boulahia, and J.W. Roberts. Measurement-based admission control for elastic traffic. In *Proceedings of 17th International Teletraffic Congress, ITC17*, Salvador, Brasil, December 2001. *(Cited on page 33.)*

[33] L. Gharai and C. Perkins. Implementing congestion control in the real world. In *Proceedings of IEEE International Conference on Multimedia and Expo, ICME '02*, volume 1, pages 397–400, Marina del Rey, CA, USA, 2002. *(Cited on page 34.)*

[34] The Gnuplot Utility. Available at `http://www.gnuplot.info/`. *(Cited on page 50.)*

[35] P. Goyal, S. S. Lam, and H. M. Vin. Efficient Fair Queuing Algorithms for Packet-Switched Networks. *IEEE/ACM Transactions on Networking*, 6:175–185, April 1998. *(Cited on page 12.)*

[36] 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*, 5:690–704, October 1997. *(Cited on pages 12 and 16.)*

[37] S. Greenstein. Four nightmares for net neutrality. *Micro, IEEE*, 26:12–13, November-December 2006. *(Cited on page 12.)*

[38] M. Grossglauser and D. N. C. Tse. A Framework for Robust Measurement-based Admission Control. *IEEE/ACM Transactions on Networking*, 7:293–309, June 1999. *(Cited on page 32.)*

[39] J. Heinanen, F. Baker, W. Weiss, and J. Wroclawski. Assured Forwarding PHB Group. *IETF RFC 2597*, Jun. 1999. *(Cited on page 11.)*

[40] M. Ilyas. Congestion avoidance in computer networks. *Electronics Letters*, 21:1161–1162, 1985. *(Cited on page 35.)*

[41] ITU-T Recommendation E.721. *Network grade of service parameters and target values for circuit-switched services in the evolving ISDN*, May 1999. *(Cited on pages 39, 63, 64 and 102.)*

[42] ITU-T Recommendation E.800. *Terms and Definitions Related to Quality of Service and Network Performance Including Dependability*, Aug. 1993. *(Cited on page 10.)*

[43] ITU-T Recommendation G.1000. *Communications Quality of Service: A Framework and Definitions*, Nov. 2001. *(Cited on page 10.)*

[44] ITU-T Recommendation I.350. *General Aspects of Quality of Service and Network Performance in Digital Networks, Including ISDNs*, Mar. 1993. *(Cited on page 10.)*

[45] R. Jain. Myths about Congestion Management in High Speed Networks. *Internetworking: Research and Experience*, 3:101–113, 1992. *(Cited on page 34.)*

[46] R. Jain. Congestion control in computer networks: issues and trends. *IEEE Network*, May 1999. *(Cited on page 34.)*

[47] R. Jain, K. K. Ramakrishnan, and D. M. Chiu. Congestion avoidance in computer networks with a connectionless network layer. *Techincal Report DEC-TR-506, Digital Equipment Corporation*, 1987. *(Cited on page 35.)*

[48] A. Jajszczyk and R. Wojcik. Emergency Calls in Flow-Aware Networks. *Communications Letters, IEEE*, 11:753–755, September 2007. *(Cited on page 30.)*

[49] Y. Jiang, P.J. Emstad, A. Nevin, V. Nicola, and M. Fidler. Measurement-Based Admission Control for a Flow-Aware Network. In *Proceedings of 1st Conference on Next Generation Internet Networks - Traffic Engineering, NGI 2005*, pages 318–325, Rome, Italy, April 2005. *(Cited on page 33.)*

[50] Y. Jiang, A. Nevin, and P. J. Emstad. Implicit Admission Control for a Differentiated Services Network. In *Proceedings of 2nd Conference on Next Generation Internet Design and Engineering, NGI 2006*, Trondheim, Norway, April 2006. *(Cited on page 34.)*

[51] J. Joung, J. Song, and S. S. Lee. Flow-Based QoS Management Architectures for the Next Generation Network. *ETRI Journal*, 30:238–248, April 2008. *(Cited on page 29.)*

[52] S. Kaczmarek and M. Landowski. Performance of FAN Conception of Traffic Control in IP QoS Networks. In *Proceedings of the 1st International Conference on Information Technology, IT 2008*, Gdansk, Poland, May 2008. *(Cited on page 28.)*

[53] M. Karlsson, Ch. Karamanolis, and J. Chase. Controllable Fair Queuing for Service Utilities. In *HP Laboratories*, February 2005. *(Cited on page 12.)*

[54] N-S. Ko, S-B. Hong, K-H. Lee, H-S. Park, and N. Kim. Quality-of-Service Mechanisms for Flow-Based Routers. *ETRI Journal*, 30:183–193, April 2008. *(Cited on page 30.)*

[55] A. Kortebi, L. Muscariello, S. Oueslati, and J. Roberts. On the scalability of fair queuing. In *Proceedings of Third Workshop on Hot Topics in Networks, ACM HotNets-III 2004*, San Diego, USA, November 2004. *(Cited on pages 19 and 31.)*

[56] A. Kortebi, L. Muscariello, S. Oueslati, and J. Roberts. Evaluating the number of active flows in a scheduler realizing fair statistical bandwidth sharing. In *Proceedings of International Conference on Measurement and modeling of computer systems, ACM SIGMETRICS 2005*, Banff, Canada, June 2005. *(Cited on page 31.)*

[57] 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 '05*, Princeton, USA, October 2005. *(Cited on pages 16, 28 and 29.)*

[58] A. Kortebi, S. Oueslati, and J. Roberts. Cross-protect: implicit service differentiation and admission control. In *Proceeding of High Performance Switching and Routing, HPSR 2004*, Phoenix, USA, April 2004. *(Cited on pages 4, 14, 16 and 28.)*

[59] A. Kortebi, S. Oueslati, and J. Roberts. Implicit Service Differentiation using Deficit Round Robin. In *Proceedings of 19th International Teletraffic Congress, ITC19*, Beijing, China, August/September 2005. *(Cited on pages 16 and 28.)*

[60] Y. Koucheryavy, D. Moltchanov, and J. Harju. An Analytical Evaluation of VoD Traffic Treatment within the EF-enabled DiffServ Ingress and Interior Nodes. In *Proceedings of 10th International Conference on Telecommunications, ICT 2003*, volume 2, pages 1458–1464, March 2003. *(Cited on page 12.)*

[61] A. Kumar, M. Hegde, S. V. R. Anand, B. N. Bindu, D. Thirumurthy, and A. A. Kherani. Nonintrusive TCP Connection Admission Control for Bandwidth Management of an Internet Access Link. *IEEE Communications Magazine*, 38:160–167, May 2000. *(Cited on page 32.)*

[62] K. Kumazoe, Y. Hori, T. Ikenaga, and Y. Oie. Quality of Assured Service through Multiple DiffServ Domains. In *Proceedings of IEEE Pacific Rim Conference on Communications, PACRIM 2001*, volume 1, pages 83–86, August 2001. *(Cited on page 11.)*

[63] P. L'Ecuyer. Good Parameters and Implementations for Combined Multiple Recursive Random Number Generators. *Operations Research*, 47(1):159–164, Jan.-Feb. 1999. *(Cited on page 52.)*

[64] H. Lee and K. Sohraby. Flow-Aware Link Dimensioning for Guaranteed-QoS Services in Broadband Convergence Networks. *Journal of Communications and Networks*, 8:410–421, December 2006. *(Cited on page 29.)*

[65] 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 *Proceedings of 4TH Int. Tel. Net. Workshop on QoS in Multiservice IP Networks 2008*, Venice, Italy, February 2008. *(Cited on page 30.)*

[66] W. Luo, C. Lin, and B. Yan. A Congestion Control Mechanism Supporting QoS Requirements. In *Proceedings of International Conference on Computer Networks and Mobile Computing, ICCNMC 2001*, Beijing, China, October 2001. *(Cited on pages xv, 24 and 35.)*

[67] L. Massoulie and J. Roberts. Arguments in Favour of Admission Control for TCP Flows. In *Proceedings of 11th International Teletraffic Congress – Specialist Seminar on Teletraffic Issues related to Multimedia and Nomadic Communications, ITC11*, Yokohama, Japan, October 1998. *(Cited on page 32.)*

[68] The Network Simulator ver. 2. Available at `http://www.isi.edu/nsnam/ns/`. *(Cited on pages 4 and 49.)*

[69] S. Oueslati and J. Roberts. Method and a device for implicit differentiation of quality of service in a network. *United States Patent 2004/0213265 A1*, Oct. 2004. *(Cited on page 28.)*

[70] S. Oueslati and J. Roberts. A new direction for quality of service: Flow-aware networking. In *Proceedings of 1st Conference on Next Generation Internet Networks - Traffic Engineering, NGI 2005*, Rome, Italy, April 2005. *(Cited on pages 4, 14 and 28.)*

[71] K. Pawlikowski. Steady-state simulation of queueing processes: A survey of problems and solutions. *ACM Computing Surveys*, 22(2):123–170, Jun. 1990. *(Cited on page 52.)*

[72] The Perl Language. Available at `http://www.perl.org/`. *(Cited on page 50.)*

[73] J. Postel. Internet Protocol. *IETF RFC 791*, Sep. 1981. *(Cited on page 10.)*

[74] K. Psounis, R. Pan, and B. Prabhakar. Approximate Fair Dropping for Variable-Length Packets. *Micro, IEEE*, 21:48–56, January 2001. *(Cited on pages 4 and 93.)*

[75] G. Qiu, S. Zhang, and S. Liu. Flow-Aware Based Integrated Admission Control Scheme. In *Proceedings of 2007 International Conference on Wireless Communications, Networking and Mobile Computing, WiCOM 2007*, pages 2933–2836, Shanghai, China, September 2007. *(Cited on page 33.)*

[76] K. K. Ramakrishnan and R. Jain. A Binary Feedback Scheme for Congestion Avoidance in Computer Networks. *ACM Transactions on Computer Systems (TOCS) archive*, 8:158–181, 1990. *(Cited on page 35.)*

[77] J. Roberts. Engineering for Quality of Service. *In Self-Similar Network Traffic and Performance Evaluation*, pages 401–420, 2000. *(Cited on page 12.)*

[78] J. Roberts. Traffic theory and the Internet. *IEEE Communications Magazine*, 39:94–99, January 2001. *(Cited on page 32.)*

[79] J. Roberts. A survey on statistical bandwidth sharing. *Computer Networks: The International Journal of Computer and Telecommunications Networking*, 45:319–332, June 2004. *(Cited on page 31.)*

[80] J. Roberts. Internet Traffic, QoS and Pricing. *Proceedings of the IEEE*, 92:1389–1399, September 2004. *(Cited on pages 14 and 28.)*

[81] J. Roberts, S. Oueslati, and A. Kortebi. Method and device for scheduling packets for routing in a network with implicit determination of packets to be treated as priority. *United States Patent 2007/0291644 A1*, Dec. 2007. *(Cited on pages 4 and 28.)*

[82] J. Roberts and S. Oueslati-Boulahia. Quality of Service by Flow Aware Networking. *Philosophical Transactions of Royal Society*, 358(1773):2197–2207, Aug. 2000. *(Cited on page 27.)*

[83] J. W. Roberts and L. Massoulie. Bandwidth Sharing and Admission Control for Elastic Traffic. In *Proceedings of 16th International Teletraffic Congress, ITC16*, Edinbourg, Wales, June 1999. *(Cited on page 32.)*

[84] L. G. Roberts. Mico-flow management. *United States Patent 6,574,195 B2*, Jun. 2003. *(Cited on page 31.)*

[85] L. G. Roberts. The Next Generation of IP – Flow Routing. In *Proceedings of Scuola Superiore G Reiss Romoli International Conference, SSGRR 2003*, L'Aquila, Italy, July 2003. *(Cited on page 31.)*

[86] L. G. Roberts. Mico-flow management. *United States Patent 6,954,431 B2*, Oct. 2005. *(Cited on page 31.)*

[87] L. G. Roberts. System and method for network tunneling utilizing micro-flow state information. *United States Patent 6,977,932 B1*, Dec. 2005. *(Cited on page 31.)*

[88] L. G. Roberts. Mico-flow label switching. *United States Patent 7,012,919 B1*, Mar. 2006. *(Cited on page 31.)*

[89] L. G. Roberts. Mico-flow management. *United States Patent 7,126,918 B2*, Oct. 2006. *(Cited on page 31.)*

[90] L. G. Roberts. Mico-flow management. *United States Patent 2007/0115825 A1*, May 2007. *(Cited on page 31.)*

[91] S. Shioda and K. Mase. Performance comparison between IntServ-based and DiffServ-based networks. In *Proceedings of IEEE Global Telecommunications Conference, GLOBECOM 2005*, volume 1, page 6 pp, Nov.-Dec. 2005. *(Cited on page 4.)*

[92] M. Shreedhar and G. Varghese. Efficient Fair Queuing Using Deficit Round-Robin. *IEEE/ACM Transactions on Networking*, 4:375–385, June 1996. *(Cited on pages 12 and 19.)*

[93] J. Song, M. Y. Chang, and S. S. Lee. Overview of ITU-T NGN QoS Control. *IEEE Communications Magazine*, 45:116–123, September 2007. *(Cited on page 29.)*

[94] J. Song, S. S. Lee, K-C. Kang, N. Park, H. Park, S. Yoon, K. G. Chun, M. Y. Chang, J. Joung, and Y. S. Kim. Scalable Network Architecture for Flow-Based traffic Control. *ETRI Journal*, 30:205–215, April 2008. *(Cited on page 31.)*

[95] D. Stiliadis and D. Varma. Determining end-to-end delay bounds in heterogeneous networks. *Lecture Notes in Computer Science*, 1018:273–284, April 1995. *(Cited on page 12.)*

[96] B. Suter, T.V. Lakshman, D. Stiliadis, and A.K. Choudhury. Buffer Management Schemes for Supporting TCP in Gigabit Routers with Per-Flow Queueing. *IEEE Journal in Selected Areas in Communications*, August 1999. *(Cited on page 25.)*

[97] D. Tse and M. Grossglauser. Measurement-based call admission control: Analysis and simulation. In *Proceedings of Sixteenth Annual Joint Conference of the IEEE Computer and Communications Societies, INFOCOM'97. Proceedings IEEE*, Berkeley, USA, April 1997. *(Cited on page 32.)*

[98] J. Tyszer. *Object-Oriented Computer Simulation of Discrete-Event Systems*. Kluwer Academic Publishers, USA, 1999. *(Cited on page 50.)*

[99] D.J. Weitzner. Twelve Billion Bargaining Chips: The Web Side of the Net Neutrality Debate. *Internet Computing, IEEE*, 11:78–81, January-February 2007. *(Cited on page 13.)*

[100] M. Welzl. *Network Congestion Control*. John Wiley & Sons. Ltd, England, 2005. *(Cited on page 35.)*

[101] C.-Q. Yang and V. S. Reddy. A taxonomy for congestion control algorithms in packet switching networks. *IEEE Network*, 9, July/August 1995. *(Cited on page 34.)*

Index