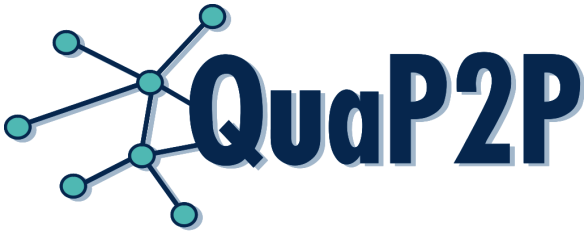

Technische Universität Darmstadt

Department of Electrical Engineering and Information Technology
Department of Computer Science (Adjunct Professor)
Multimedia Communications Lab
Prof. Dr.-Ing. Ralf Steinmetz

Taxonomy of Message Scheduling Strategies in Context of Peer-to-Peer Scenarios

Technical Report KOM-TR-2007-02

By

Kálmán Graffi, Nicolas Liebau, Ralf Steinmetz

Contact: [graffi;liebau]@kom.tu-darmstadt.de
http://www.kom.tu-darmstadt.de

First published: 01. December 2006
Last revision: 03. April 2007

For the most recent version of this report see
ftp://ftp.kom.tu-darmstadt.de/pub/TR/KOM-TR-2007-02.pdf

DFG Research Group 733: Improving the Quality of Peer-to-Peer Systems

Contents

List of Tables

Chapter 1

Message Scheduling in P2P: Motivation and Classification

Peer-to-peer (P2P) principles have recently attracted a lot of attention in the research community. They are of particular interest for large-scale systems, such as the Internet. P2P systems have no single point of failure because they build on the principle of self-organization. However self-organization of the peers necessitates the exchange of various maintenance messages, which cause a significant overhead. It may occur that maintenance of the network requires a great fraction of available resources; typically over-provisioning solves the problem of limited resources.

Efficiency is relevant when over-provisioning cannot be done. In case of a catastrophe scenario, for example, bandwidth is scarce and the participating devices are highly heterogeneous. In addition to the limitations on the technical level, crucial services have to be available and be performed in an acceptable quality. These circumstances require efficient utilization of the scarce bandwidth in each peer and of the scarce resources in the overlay.

1.1 Message Scheduling in Peer-to-Peer

In today's networks bandwidth is the most scarce resource. The trend to mobile devices increases the lack of sufficient bandwidth. As bandwidth cannot be increased easily, the efficient utilization of available bandwidth is crucial. However we assume that various services of differing relevance are provided by a peer-to-peer system. Some users require high throughput, other users may have critical demands for low delay. Some real-time requests need to be processed fast, while replication-mechanisms for P2P do not have to be processed with strict deadlines.

Time-critical and not-time-critical services need to be distinguished from each other. They have differing relevance and differing requirements in terms of delay, fairness, and demands on bandwidth. These considerations can be found in literature for the network layer as well. It is worth looking at the solutions that have been developed on the network layer. Fair scheduling and active queue management provide principles to enable the fair processing of delay-critical and loss-critical traffic.

Applying fair scheduling mechanisms results in rearranging the order in which incoming messages are processed in each peer. This step requires the introduction of message priorities with respect to the demands of the corresponding flows in terms of delay, loss-tolerance, bandwidth, and other criteria. The second step in applying fair scheduling mechanisms consists of enhancing the peers with queue modification capabilities. Peers should be able to use the priorities of the incoming messages for deciding in which order to process the packets. With this, the scarce outgoing bandwidth can be utilized smarter and delay-critical services can be provided. However, if delay-critical flows are prioritized, other flows may receive a too small (unfair) share of service. To cope with this problem, in this paper we survey solutions suggested for the network layer considering fair bandwidth allocation of time-critical and non-time-critical flows.

In order to provide guarantees for time-critical services in P2P networks, solutions from the network layer have to be investigated and adopted if possible. Quality of service models discussed in literature can be classified to follow the over-provisioning principle, using integrated services (IntServ) or follow the differentiated services (DiffServ) principle. Over-provisioning is not always feasible. In the catastrophe scenario [BSBKM07] the available bandwidth in the network is constrained by damages of the network infrastructure and over-provisioning is not applicable du to high costs for extending device capabilities. The integrated services principle requires to establish a flow with guarantees using signaling protocols. This approach is expected to result in high overhead, as in a P2P network two nodes may communicate only sporadically and the number of contacts is very high. A network deploying differentiated services treats packets according to their type. Each packet is labeled according to parameters that have effects on the service the packet receives. This characteristic is most convenient, as not all peers are required to participate. Moreover, there is no additional overhead for flow reservation and the principle of per-packet-processing fits to the idea of P2P.

1.2 Classification Classes Related to Scheduling Mechanisms

In this section we give an overview on design and implementation aspects that can be used to classify scheduling algorithms. We present them and and give a short description.

We present in Chapter 2 a survey on scheduling mechanisms and a classification of each algorithm according to this overview on classification aspects. We use flat classification groups, as hierarchical groups (see [CK88]) require that the surveyed scheduling mechanisms differ essentially. Additionally a strict hierarchical classification is not possible if two independent classes exist. Thus, we propose a flat classification to describe the properties of the surveyed scheduling mechanisms.

1.2.1 Approach of the solution

Under this classification we subsume the family of solutions to which the observed mechanism belongs. There exist various solutions extending previously proposed mechanisms.

1.2.2 Optimality vs. approximation vs. heuristic

This classification gives detail on a special characteristic of the proposed model: whether it is optimal in some point. In some cases new metrics are introduced, which are considered relevant by the authors. In relation to these metrics, optimal solutions are presented. However, some solutions only approximate an optimal solution due to several constraints. The third group is the class of heuristic approaches.

1.2.3 Adaptability

We define a system being adaptable, if it allows users of the system (sources of passing flows) to increase or decrease their packet rate. Some scheduling mechanism do not consider changing demands. When a flow changes its behavior, e.g. by increasing the amount of traffic, some scheduling models react with adapting the service rates provided for this flow. However some models stick to static service rates. Hereby the creator of the flow effects the changes of the flow.

1.2.4 Fairness

A scheduling solution provides fairness if greedy, malicious flows cannot allocate additional service share at the cost of flows that already receive less service (*Min-Max-Fairness*). When the share of a flow is bounded, we call the scheduling algorithm fair.

1.2.5 Cooperative vs. individual

Most scheduling algorithms are applied on each peer individually: peers do not have to communicate directly or indirectly with each other to provide the scheduling functionality.

However, some mechanisms require interaction and cooperation of the peers to provide good results.

1.2.6 Role-based tasks vs. homogeneous model

This classification considers the models which require all peers to perform the same strategy. In contrast to this homogeneous peer strategy, some scheduling mechanisms introduce various roles for the peers. In this case the algorithms and strategies how to cope with packets is not equal in each peer.

1.2.7 Flow types: hierarchical vs. flat

Systems with flat flow types have a single set of flow types. With this each flow can only be of one type (although this single type may be multidimensional, considering delay-criticality, loss-tolerance, etc.). Systems with hierarchical flow types introduce a tree classification for the flow types. Except the leaf types each inner type is split in sub-types, having different demands at the network.

1.2.8 Sorted priority list vs. frame-based scheduling

Scheduling mechanisms based on sorted priority lists maintain a virtual time counter per outgoing queue. Each arriving packet is labeled with a time-stamp and inserted into the sorted queue. Packets are transmitted by their order in the sorted queue. In contrast to this, frame-based scheduling mechanisms split the timescale in frames and determine which flows may be active during each frame. In a single frame all flows have the same priority and receive the same fraction of the whole service the system offers.

1.2.9 Timer-based vs. self-clocking solutions

In self-clocking systems the peer estimates the order of the packets strictly based on the current packets in the system. Only virtual clocks are considered, which are measured relative to the arrival of the packets. In timer-based solutions, each peer has an independent clock, which is used to determine service times for the packets.

1.2.10 Dynamic service rates vs. static service rates

A scheduler typically tries to provide the same amount of service constantly to a flow. However, some schedulers vary the service provided to a flow, independently to changes in the flow requirements defined by the flow's user. Whether the system varies the service rate for a flow or not, is expressed by this classification.

1.2.11 Dynamic packet priorities vs. static packet priorities

This classification is similar to the previous one. But here the scheduler observes a changed behavior and reacts with relabeling the packets. Schedulers with static packet priorities do not check the correctness of the packet labels, once a priority is determined, it is not adapted.

1.2.12 Start-time-based vs. finish-time-based

In scheduling mechanisms an order of processing has to be determined. Start-time based models derive the order mainly from the arrival time of the packet. Finish-time based models order the packets according to their finishing time in a ideal fluid model. Figures 1.1 and 1.2 show the principles of start-time and finish-time based models. They show a set of flows, each flow has a current message pending. The solid line boundary of the message show which time point is considered for scheduling. The dotted boundary of the message is hereby ignored.

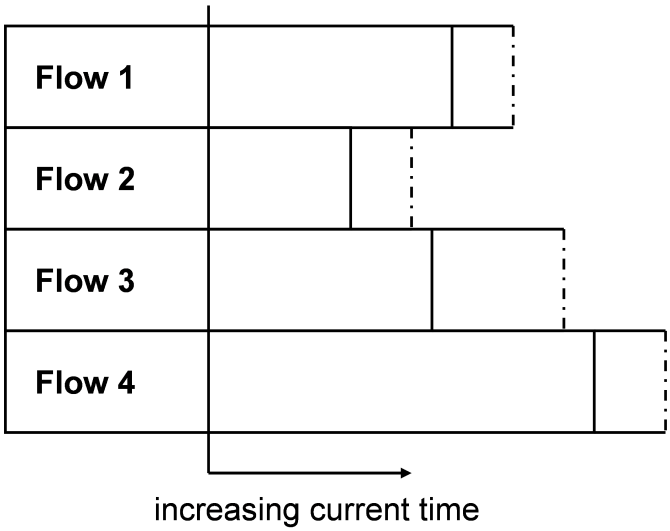

Figure 1.1: This figure shows the principle of start-time based scheduling. The arrival rate is used to determine the scheduling order.

1.2.13 Independent flows vs. inter-dependent flows

Two kinds of models exist. The first one allows flows to have effect on the service received by other flows in a negative way. Malicious flows may try to demand as much bandwidth

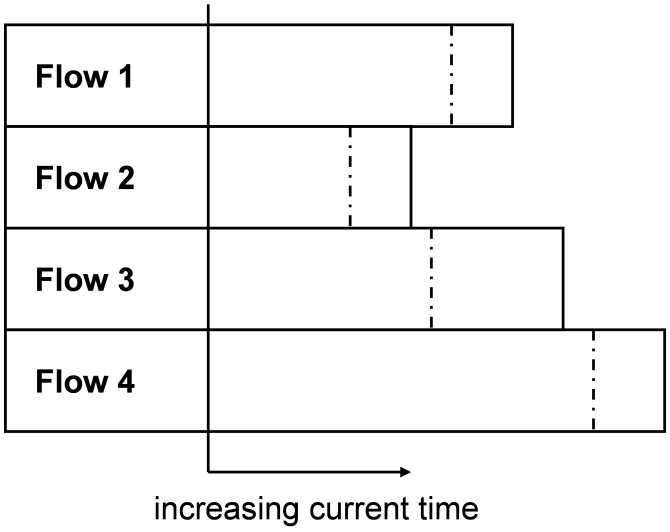

Figure 1.2: This figure shows the principle of finish-time based scheduling. The finish time is calculated using a fluid model

as possible at the expense of other flows. The other kind of scheduling mechanisms introduce some *firewall* concepts, which protect well-behaving flows from malicious flows.

1.2.14 Limited service saving vs. unlimited service saving

Scheduling algorithms need to take into account the service that has been provided to the individual flows. The time window which is considered for the service provisioning is limited in some scheduling mechanisms. If the time window is unlimited, the provided service is accumulated from start of the flow on.

1.2.15 Single-dimensional service demands vs. multi-dimensional service demands

Flow demands might be constraint with respect to various parameters. Delay, bandwidth, loss, jitter, etc. are different Quality of Services parameters that have different relevance for each flow. Most of the scheduling mechanisms consider only one priority per packet, which represents all service demands related to this flow. However, some scheduling mechanisms take the diversity of service demands into account and provide strategies how to incorporate the demands for each of these QoS aspects.

Chapter 2

Survey on Scheduling Mechanisms

In this section we present scheduling mechanisms discussed in literature that potentially could be adopted to our P2P scenario. For each scheduling mechanism we present the algorithm and additionally the ideas and concepts behind it. Some of the algorithms were designed for process scheduling but can also be easily adapted to packet scheduling. Throughout in this survey we use the terminology for packet scheduling in order to be consistent. Furthermore, we provide details how the specific scheduling mechanism fits into the classification described in Section 1.2.

2.1 FIFO: First-In-First-Out

The First-In-First-Out principle processes the incoming packets ordered by there arrival time. This principle describes the First-Come-First-Serve behavior. The mechanism is quite simple and the oldest known. Queuing theory extends this simple version with a mathematical framework to be able to calculate mean waiting times, average queue lengths and other points of interest. A classification of FIFO according to Chapter 1.2 is presented in Table 2.1.

2.2 RM: Rate-Monotonic Priority Assignment

Rate-Monotonic Priority Assignment was introduced in 1973 by Liu and Layland in [LL73]. The authors state following assumptions for the system:

- Packets arrive periodically in constant rates in their flows
- For all packets of a flow the flow-specific packet-length is fix

Scheduling classification	FIFO
Approach	Deterministic
Optimality vs. approximation vs. heuristic	Stateless, minimal overhead: $O(1)$
Adaptability	Yes, changes are directly adopted.
Fairness	No
Cooperative vs. individual	Individual solution
Role-based tasks vs. homogeneous model	Homogeneous, all peers have follow the same strategy.
Flow types: hierarchical vs. flat	Flat. (All flows have same priority)
Priority list vs. frame-based scheduling	Priority list: all flows have same priority.
Timer-based vs. self-clocking	No timer needed.
Service rates: dynamic vs. static	Dynamic (best effort)
Packet priorities: dynamic vs. static	Static (all equal)
Start-time-based vs. finish-time-based	Arriving time determines processing order.
Independent flows vs. inter-dependent flows	Flows have direct effect on each other
Service saving: limited vs. unlimited	No service history is maintained
Service demands: single-dim. vs. multi-dim.	Zero-dimensional, no deadlines are considered.

Table 2.1: FIFO in Context of Scheduling Classification

- The deadline for every packet is defined by the next packet arriving on this flow
- The flows are independent from each other, they do not influence each other on a higher level in the ISO/OSI layer
- For scheduling non-periodically appearing packets other mechanisms exist in the system

The RM algorithm assigns fixed priorities to the flows. A high arrival rate of the packets in each flow results in a high priority for scheduling. The scheduler chooses among all packets in the system (there is only one per flow) the packet with the highest priority, i.e. the flow with the highest rate. In Figure 2.1 we present the main principle of RM using a small example.

We name the i^{th} flow (out of m) in the system f_i. It is characterized by the arrival period T_i and the constant packet size l_i. The utilization factor U_S of the system S describes the fraction of the time in which the peer is transmitting in relation to the total time. For static priority based schedulers one can derive upper bounds for the utilization factor. By increasing the utilization further than this upper bound one cannot guarantee to meet the deadlines of each packet. For all utilization factors below this upper bound RM provides a fixed priority assignment which fulfills the deadlines. The upper bound in RM is $U_S = \sum_{i=1}^{m} \{l_i/T_i\}$.

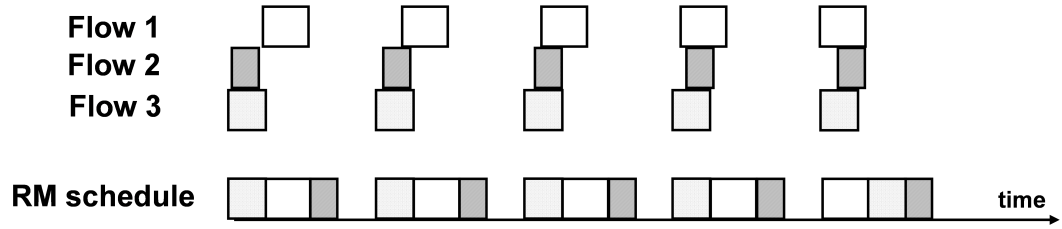

Figure 2.1: This figure shows the main principle of RM. The packets are scheduled according to their arrival rate. The example for EDF in Figure 2.2 uses the same setting, but results in another schedule.

The authors show that Rate-Monotonic Priority Assignment is an optimal static-priority scheduling algorithm. If a flow-set can be scheduled to meet the deadlines using static priorities, then RM provides a feasible schedule for this flow-set. RM finds a feasible schedule for a flow-set consisting of m flows, in case of $U_S \leq m \cdot \{ \sqrt[m]{2} - 1\}$. For large m, U_S converges to a fix value: $lim_{m\to\infty} U_S \approx ln2 \approx 70\%$. So for any utilization factor smaller than approximately 70% RM provides a feasible solution. A classification of RM according to Chapter 1.2 is presented in Table 2.2.

Scheduling classification	Rate-Monotonic Priority Assignment
Approach	Deterministic
Optimality vs. approximation vs. heuristic	Optimal for fixed priorities
Adaptability	No changes in the packet arrival rate considered.
Fairness	No. Service share is proportional to arrival rate.
Cooperative vs. individual	Cooperation of peers needed to keep static arrival rates.
Role-based tasks vs. homogeneous model	Homogeneous model.
Flow types: hierarchical vs. flat	Flat.
Priority list vs. frame-based scheduling	Priority lists, priorities according arrival rates.
Timer-based vs. self-clocking	Self-clocking, but arrival rates has to be known.
Service rates: dynamic vs. static	Static. Arrival rates are considered to be static.
Packet priorities: dynamic vs. static	Dynamic, other peers may have higher prioritized flows.
Start-time-based vs. finish-time-based	The arrival/start time of a packet defines its order.
Independent flows vs. inter-dependent flows	Inter-dependent flows, one may consume all bandwidth.
Service saving: limited vs. unlimited	No service history.
Service demands: single-dim. vs. multi-dim.	Single-dimensional: Arrival rate.

Table 2.2: RM in Context of Scheduling Classification

2.3 EDF: Earliest Deadline First

The authors of RM introduce in the same paper [LL73] a scheduling principle called Earliest Deadline First. In contrast to RM the EDF algorithm assigns dynamically priorities to the incoming packets. As according to the assumptions the rate of the packets is known, deadlines can be calculated. EDF processes the packets according to the deadlines in increasing order. This deadline driven scheduling algorithm provides optimal solutions with respect to feasibility. If the packets can be scheduled in a way that all of them meet there deadlines, than EDF provides a valid schedule. EDF is feasible if $U_S = \sum_{i=1}^{m} \{l_i/T_i\} <= 1$. This means that up to a utilization factor of 100% EDF provides a valid schedule. In Figure 2.2 we present the main principle of EDF using a small example.

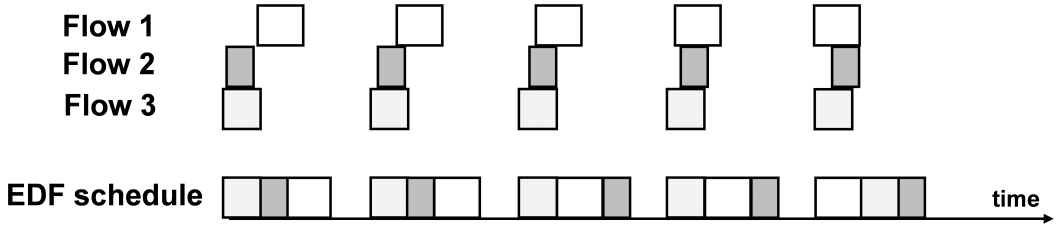

Figure 2.2: This figure shows the main principle of EDF. The packets are scheduled according to their deadlines. The example for RM in Figure 2.1 uses the same setting, but results in another schedule.

Liu and Layland propose in [LL73] additionally a mixed scheduling algorithm. The first k flows with shortest inter-packet periods are scheduled with RM. The other flows are scheduled using EDF. The authors motivate mixed scheduling algorithms with decreased scheduling costs in comparison to EDF and increased utilization tolerance in compare to RM. A classification of EDF according to Chapter 1.2 is presented in Table 2.3.

2.4 RCSP: Rate-Controlled Static Priority

Zhang and Ferrari discuss in [ZF94] problems that arise when only per-hop delay and bandwidth bounds are guaranteed. To provide end-to-end deterministic and statistical performance guarantees it is necessary to limit the effects of accumulation of per-hop delay and per-hop traffic bursts.

The authors assume that every flow has a priori reserved resources according to expected traffic characteristics and performance requirements. An architecture consisting of

Scheduling classification	EDF
Approach	EDF-family, static packet arrival rates.
Optimality vs. approximation vs. heuristic	Optimal: Provides a schedule if one exist.
Adaptability	Yes. Service rate adapts to arrival rate.
Fairness	Yes. All deadlines are met if a valid schedule exists.
Cooperative vs. individual	Individual. No cooperation of peers required.
Role-based tasks vs. homogeneous model	Homogeneous model, all peers behave equally.
Flow types: hierarchical vs. flat	Flat.
Priority list vs. frame-based scheduling	Priority list, time progresses continuously.
Timer-based vs. self-clocking	External timer needed to meet the packet deadlines.
Service rates: dynamic vs. static	Dynamic, service rates depend on arrival rates.
Packet priorities: dynamic vs. static	Dynamic, rates of other flows have effect.
Start-time-based vs. finish-time-based	Finish-time based. EDF-principle.
Independent flows vs. inter-dependent flows	All flow deadlines are met independently, if possible.
Service saving: limited vs. unlimited	No service history is maintained.
Service demands: single-dim. vs. multi-dim.	Single-dimensional demand: deadline.

Table 2.3: EDF in Context of Scheduling Classification

a rate-controller and a packet scheduler is proposed. The rate-controller allocates bandwidth and controls traffic distortion. The traffic is forced to obey the desired traffic patterns, i.e. jitter is eliminated. The scheduler orders the processing and transmission of real-time critical packets in order to control delay bounds. The authors focus on the rate-controller and leave the decision open which scheduler to choose.

In order to decrease traffic distortions, traffic on each connection is monitored. The measured traffic characteristics are compared to the negotiated flow rules. This is done by assigning to each packet an eligibility time. Packets that are ahead of the schedule are hold, until their eligibility time starts. By this the traffic patterns are reconstructed and shaped so that negative effects (delay, bursts) do not accumulate.

The regulator which is used to calculate the eligibility time for a packet determines how the traffic is shaped according to the desired traffic pattern.

Two regulators are introduced: the rate-jitter controlling regulator (RJ) and the delay-jitter controlling regulator (DJ). Using the RJ regulator the eligibility time of the k^{th} packet in flow i is: $ET_i^1 = AT_i^1$ for the first packet arriving on the flow and $ET_i^k = max\{ET_i^k + T_{min}, ET_i^{l-\frac{T_{mes}}{T_{avg}}l+1} + T_{mes}, AT_i^k\}$. Hereby we name AT_i^k the arrival time of the k^{th} packet in flow i. T_{min} and T_{avg} describes the minimum and average inter-arrival times between two packet of a flow. T_{mes} is the time interval during which traffic has been monitored to determine T_{avg}.

In the DJ regulator the eligibility time is calculated for the first packet as $ET_i^1 = AT_i^1$

and for the following packets as $ET_i^k = ET_i^{k-1} + d_{i-1} + m_i$. Here we name d_{i-1} the local delay bound for this flow in the previous peer and m_i the maximum delay from peer i-1 to peer i.

Zhang and Ferrari show that end-to-end delays are bounded when using rate-controlling and a scheduling algorithm, which itself provides an upper bound on delay. A classification according to Chapter 1.2 of RCSP is presented in Table 2.4.

Scheduling classification	RCSP
Approach	EDF-family: rate controlled EDF
Optimality vs. approximation vs. heuristic	Approximation
Adaptability	No. Service for flows is bound to reservations.
Fairness	No, has to be done during reservation process.
Cooperative vs. individual	Cooperative. End-to-end perf. guarantees aimed.
Role-based tasks vs. homogeneous model	Homogeneous behavior of all peers.
Flow types: hierarchical vs. flat	Flat.
Priority list vs. frame-based scheduling	Priority list according eligibility times.
Timer-based vs. self-clocking	Timers needed for monitoring.
Service rates: dynamic vs. static	Static, bounded to resource reservations.
Packet priorities: dynamic vs. static	Static, reservations are flow-related.
Start-time-based vs. finish-time-based	Start-time based. Order by lowest eligible start time.
Independent flows vs. inter-dependent flows	Independent flows bound to their reservations.
Service saving: limited vs. unlimited	Limited service history is maintained to eliminate jitter.
Service demands: single-dim. vs. multi-dim.	Single-dimensional demand defined by deadlines.

Table 2.4: RCSP in Context of Scheduling Classification

2.5 FQ: Fair Queuing (Round Robin)

Nagle proposed 1987 in [J. 87] a simple scheduling mechanism in which each flow is assigned to a queue of its own. Packets are transmitted using the round robin principle to choose the next queue to be serviced. This approach can be implemented very efficiently, as no further computation is needed. However the mechanism does not take packet lengths into account, so that the bandwidth allocated to the flows may differ extremely. In Figure 2.3 we show the main principle of round robin.

A classification of FQ according to Chapter 1.2 is presented in Table 2.5.

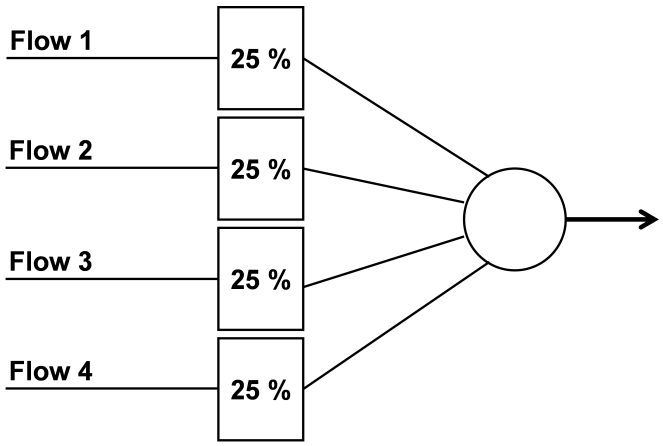

Figure 2.3: In Round Robin each flow has the same probability to be chosen.

Scheduling classification	Fair Queuing (Round Robin)
Approach	Round robin
Optimality vs. approximation vs. heuristic	Approximation
Adaptability	No, each flow receives a static share of bandwidth.
Fairness	Fair. Per flow same amount of service.
Cooperative vs. individual	Individual: Schedules of peers are independent.
Role-based tasks vs. homogeneous model	Homogeneous. All peers behave equally.
Flow types: hierarchical vs. flat	Flat. All flows have equal priority.
Priority list vs. frame-based scheduling	Frame-based: Service all flows in one round.
Timer-based vs. self-clocking	Self-clocking: service order is time-independent.
Service rates: dynamic vs. static	Static. Share of bandwidth per flow is fixed.
Packet priorities: dynamic vs. static	Static. All packets have the same priority.
Start-time-based vs. finish-time-based	Start-time-based order in the flow queue.
Independent flows vs. inter-dependent flows	Independent flows. Flow queue length has no effect.
Service saving: limited vs. unlimited	No history maintained.
Service demands: single-dim. vs. multi-dim.	Zero-dimensional: Packets have no deadline

Table 2.5: FQ in Context of Scheduling Classification

2.6 WRR: Weighted Round Robin

Classical round robin provides an equal share of service to all flows in the system. Weighted Round Robin, presented in [Kat91] by Katevenis et al., introduces for each flow i weights w_i, which define the amount of share they receive. The round robin probability

for each flow i is $\frac{w_i}{\sum_{j \in F} w_j}$, this is also the fraction of the total service provided for flow i. In Figure 2.4 we show the main principle of Weighted Round Robin. A classification of

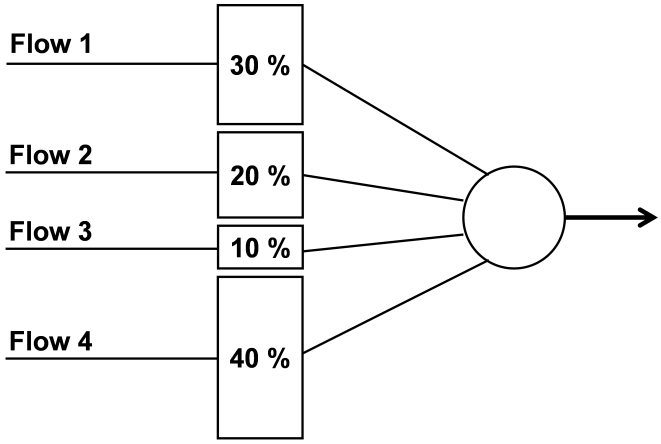

Figure 2.4: In Weighted Round Robin each flow has the probability to be chosen according to its weight.

WRR according to Chapter 1.2 is presented in Table 2.6.

Scheduling classification	Weighted Round Robin
Approach	Round robin
Optimality vs. approximation vs. heuristic	Heuristic
Adaptability	No. Static share per flow.
Fairness	Yes, each flow is guaranteed a weighted share.
Cooperative vs. individual	Round robin approaches do not rely on cooperation.
Role-based tasks vs. homogeneous model	All peers are modeled homogeneously
Flow types: hierarchical vs. flat	Flat.
Priority list vs. frame-based scheduling	Frame-based: Each flow shall receive service in a frame.
Timer-based vs. self-clocking	Self-clocking. Time is not relevant in WRR.
Service rates: dynamic vs. static	The service rate is bound to the static weight of a flow.
Packet priorities: dynamic vs. static	Depends on changes of the weight of a flow from p2p.
Start-time-based vs. finish-time-based	Start-time-based. WRR uses per-flow FIFO.
Independent flows vs. inter-dependent flows	Independent flows. Greedy flows have no effect.
Service saving: limited vs. unlimited	No service history is maintained.
Service demands: single-dim. vs. multi-dim.	Single-dimensional demand: weight of a flow.

Table 2.6: WRR in Context of Scheduling Classification

2.7 DRR: Deficit Round Robin

Deficit round robin was proposed by Shreedhar and Varghese in [SV95] to improve the fairness of the classical round robin principle when applied to networks (as in Section 2.5). In order to provide the same amount of service to every flow packet lengths has to be considered. The authors apply the leaky bucket principle to round robin by introducing a time discretion (rounds) and providing periodically a ticket to every flow. These tickets represent an amount of bytes that may be sent in the current round. The deficit counter that is introduced in this algorithm counts the remaining amount of bytes that are allowed to be sent. Using the round robin principle each flow may send on its turn when its deficit counter is smaller than the current packet size. In the case that a flow has enough deficit, the packet is transmitted and the deficit counter decreased by the length of the transmitted packet. However when the deficit of the flow is not large enough, it must not transmit any packet but has to wait for a new ticket in the next round, which increases the deficit counter of the flow.

Deficit round robin + (DRR+) is an extension of this algorithm which provides additional mechanisms to cope with delay-critical flows. Two classes of flows are introduced: latency-critical and best-effort. In DRR+ latency critical flows are prioritized and serviced at the beginning of a round at first. After processing the latency-critical flows the remaining best effort flows are processed. In both cases round robin is used. The authors assume for DRR+ that there is enough bandwidth to process all flows, i.e. no starvation occurs. A classification of DRR according to Chapter 1.2 is presented in Table 2.7.

2.8 BR: Bit-by-Bit Round Robin

Demers, Keshav and Shenker analyzed in [DKS89] the deficits of FIFO and FQ and state that bit-by-bit round robin is most fair according to the simple Max-Min-Fairness metric. Bit-by-bit round robin assumes a system with various flows to be serviced in which each flow may send a bit at a round. As this is not realistic in packet-based networks, they propose to emulate bit-by-bit round robin. For each packet the proposed algorithm calculates the time the packet would be transmitted completely when using BR. Packets are inserted according to their finishing time in a sorted queue. The scheduling algorithm sends the packet at the head of the sorted queue. This packet-by-packet sending scheme can be extended to prioritize latency-critical flows and flows that use not their full amount of fair share of the bandwidth.

A classification of BR according to Chapter 1.2 is presented in Table 2.8.

Scheduling classification	DRR
Approach	Round robin
Optimality vs. approximation vs. heuristic	Approximation of optimal fairness.
Adaptability	No. Share of bandwidth per flow does not change.
Fairness	Yes. Taking also packet sizes into account.
Cooperative vs. individual	Individual. Cooperation between peers is not needed.
Role-based tasks vs. homogeneous model	Homogeneous model, all peers behave equally.
Flow types: hierarchical vs. flat	Flat. All flow types are equal.
Priority list vs. frame-based scheduling	DRR uses the frame-based leaky bucket principle.
Timer-based vs. self-clocking	Self-clocking, in DRR only packet sizes matter.
Service rates: dynamic vs. static	Static. All flows receive equal service.
Packet priorities: dynamic vs. static	Static. Priority of a packet is related to its size.
Start-time-based vs. finish-time-based	Finish-time, defined by the size of the packet.
Independent flows vs. inter-dependent flows	Independent flows. Bursty flows have no effect.
Service saving: limited vs. unlimited	Limited service saving, bound by max. packet length.
Service demands: single-dim. vs. multi-dim.	DRR: 1-dim. demand for throughput. DRR+: 2-dim.

Table 2.7: DRR in Context of Scheduling Classification

Scheduling classification	BR
Approach	WFQ-family
Optimality vs. approximation vs. heuristic	Optimal approximation
Adaptability	No. Share of bandwidth for each flow is static.
Fairness	Yes. Greedy flows do not benefit.
Cooperative vs. individual	Individual. Each peer applies BR without cooperation.
Role-based tasks vs. homogeneous model	Homogeneous model for all peers.
Flow types: hierarchical vs. flat	Flat.
Priority list vs. frame-based scheduling	Finish-time ordered priority list.
Timer-based vs. self-clocking	Timer-based. Needed to calculate finish-time.
Service rates: dynamic vs. static	Static service rate.
Packet priorities: dynamic vs. static	Static. Packets have on each peer the same priority.
Start-time-based vs. finish-time-based	Finish-time based. Packet sizes are considered.
Independent flows vs. inter-dependent flows	Independent flows. Fair share per flow does not change.
Service saving: limited vs. unlimited	Limited service saving, bound by max. packet length.
Service demands: single-dim. vs. multi-dim.	Single-dimensional demand for fair throughput.

Table 2.8: BR in Context of Scheduling Classification

2.9 PGPS: Packet-by-Packet General Processor Sharing WFQ: Weighted Fair Queuing

The idea of bit-by-bit round robin is basis for the proposal of PGPS by Parkh and Gallager in [PG93], which is identical to WFQ proposed by Demers, Keshav and Shenker

in [DKS89]. They extend BR with leaky bucket admission control to guarantee end-to-end performance bounds. First a fluid model is introduced. In this each flow i receives a fraction $\frac{w_i}{\sum_j w_j}$ of the total service rate in each time instance. PGPS is a packet-based algorithm emulating the fluid model. When a packet arrives it is labeled with its finishing time according to the fluid model. The server is work-conserving and processes packets in the increasing order of their labels. To each flow a specific weight can be assigned and so throughput guarantees can be provided. Using a leaky bucket principle on the traffic entering the system leads to control on the burstiness of traffic. The burstiness is well-defined so that one can derive the worst-case packet delay. The maximum delay occurs when all sessions start to be greedy and want to use all of their saved tokens for transmission packets. In their work the authors give a detailed mathematical model of PGPS, which they use to derive delay and fairness bounds.

A classification of PGPS is presented according to Chapter 1.2 in Table 2.9.

Scheduling classification	Packet-by-Packet General Processor Sharing
Approach	WFQ-family
Optimality vs. approximation vs. heuristic	Optimal approximation
Adaptability	No. Share of bandwidth for each flow is static.
Fairness	Yes. Greedy flows do not benefit.
Cooperative vs. individual	Individual. Each peer applies PGPS without coop..
Role-based tasks vs. homogeneous model	Homogeneous model for all peers.
Flow types: hierarchical vs. flat	Flat.
Priority list vs. frame-based scheduling	Finish-time ordered priority list.
Timer-based vs. self-clocking	Timer-based. Needed to calculate finish-time.
Service rates: dynamic vs. static	Static service rate.
Packet priorities: dynamic vs. static	Static. Packets have on each peer the same priority.
Start-time-based vs. finish-time-based	Finish-time based. Packet sizes are considered.
Independent flows vs. inter-dependent flows	Independent flows. Fair share per flow does not change.
Service saving: limited vs. unlimited	Limited service saving, bound by max. packet length.
Service demands: single-dim. vs. multi-dim.	Single-dimensional demand for fair throughput.

Table 2.9: PGPS/WFQ in Context of Scheduling Classification

2.10 SCFQ: Self-Clocked Fair Queuing

Davin and Heybey introduce in [DH90] Self-Clocked Fair Queuing which uses Fair Queuing, No Punishment policy. As basis of their scheduling mechanism they apply the idea of WFQ and calculate for each incoming packet its finishing time according to the fair scheduling fluid model (Fair Queuing). The authors state that the size of an incoming

packet cannot be known in advance. They propose to approximate the length of the current packet by the length of the last packet in the flow. In addition to the scheduling algorithm, they suggest two strategies for queue management. They argue that if the queue is full and new packets arrive, these new packets shall be dropped, but the dropped packet shall not be counted as processed. With this only the processed and transmitted packets are taken into account to determine the share of service a flow received (No Punishment).

The authors introduce a new scheduling principle, based on different services classes, called Fair Queuing, Fixed Quota (FQFQ). They motivate that with separation of service classes inter-effects can be minimized. For each service class a buffer of its own is assigned. If the buffer corresponding to a class is full, when a new packet of this class arrives, the packet is dropped, even if the other buffers can provide enough space. With this, malicious flows are hindered to fill the buffer of the system and causing with this the dropping of packets of other flows. However, the maximum utilization of the system is decreased with this principle, as single flows cannot use the whole bandwidth, even if no other flow is active. A classification of SCFQ according to Chapter 1.2 is presented in Table 2.10.

Scheduling classification	Self-Clocked Fair Queuing
Approach	WFQ-family
Optimality vs. approximation vs. heuristic	Approximation
Adaptability	Yes, the system adapts to changing packet sizes.
Fairness	FQNP: No, FQFQ:Yes.
Cooperative vs. individual	FQNP: individual, FQFQ: cooperative.
Role-based tasks vs. homogeneous model	Homogeneous model, all peers are equal.
Flow types: hierarchical vs. flat	Flat. The packet classes of FQFQ have no hierarchy
Priority list vs. frame-based scheduling	Priority list, no need for frames.
Timer-based vs. self-clocking	Self-clocking, measurements based on observations.
Service rates: dynamic vs. static	Static: equal share of bandwidth per-flow.
Packet priorities: dynamic vs. static	Static, as for each packet its size is fixed.
Start-time-based vs. finish-time-based	Finish-time-based. The packet size is important.
Independent flows vs. inter-dependent flows	Inter-dependent. In FQFQ limited to own serv. class.
Service saving: limited vs. unlimited	Service saving limited to maximum packet length.
Service demands: single-dim. vs. multi-dim.	Single-dimensional: Only throughput matters.

Table 2.10: SCFQ in Context of Scheduling Classification

2.11 FFQ: Frame-Based Fair Queuing

Stiliadis and Varma introduce in [SV96] Frame-Based Queuing as an alternative to scheduling mechanisms based on sorted priority queues. They point out that FFQ is designed to isolate flows from each other, provide low end to end delay, be fair, and simple and efficient to implement. The authors introduce a metric called *potential* to measure the share of service each flow received. The potential of a flow it the service received from the system: $P_i = A_i + S_i$ where P_i is the potential of flow i, A_i is the potential of the system at the establishment of flow i and S_i is the amount of service flow i received since its establishment. FFQ maintains the potential of the system, which is the minimum amount of service each flow in the system received: $P_S = min_{i \in F} P_i$ where F is the set of flows. The potential of the system is a monotonic growing function. FFQ basic objective is to equalize the potential of all flows. The principle of FFQ is shown in Figure 2.5.

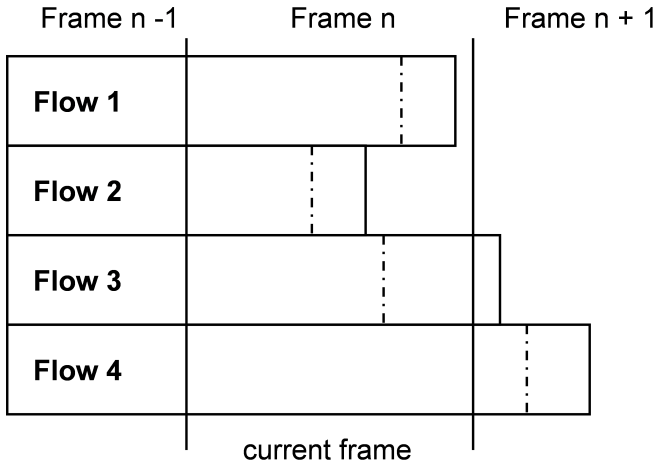

Figure 2.5: First all flows in a frame have to be serviced before moving to the next frame.

Upon packet arrival the current system potential is calculated, taking the packet currently in transmission into account. The new packet k in flow i is labeled with the current system potential and a finishing time E_i^k is calculated, which is the start potential A_i^k and the expected transmission time: $E_i^k = A_i^k + \frac{l_i^k}{r_i}$, where l_i^k is the length of the new packet and r_i is the amount of service flow i received in the current frame. The finishing time is checked whether it passed the frame border, in this case that packet is marked.

Upon transmission of a packet m of flow j, the system potential is updated: $P_{Snew} =$

$P_S + \frac{l_j^m}{l_{frame}}$, where l_{frame} is the frame size. As packet to process next the packet with the smallest potential is chosen among all backlogged flows. If this packet is the last to cross the border between two frames (last marked packet), than the frame counter is increased and the system potential is updated. The chosen packet is then transmitted. A classification of FFQ according to Chapter 1.2 is presented in Table 2.11.

Scheduling classification	Frame-Based Fair Queuing
Approach	WFQ-family
Optimality vs. approximation vs. heuristic	Approximation
Adaptability	No, the share for each flow is static and equal.
Fairness	Yes, greedy flows are bounded to their fair share.
Cooperative vs. individual	Individual. Cooperation of peers is not needed.
Role-based tasks vs. homogeneous model	Homogeneous. All peers are modeled equal.
Flow types: hierarchical vs. flat	Flat.
Priority list vs. frame-based scheduling	Frame-based, frames represent service levels.
Timer-based vs. self-clocking	Timer-based, needed for finish time calculations.
Service rates: dynamic vs. static	Static, all flows get same service per frame.
Packet priorities: dynamic vs. static	Static, packets priorities do not change.
Start-time-based vs. finish-time-based	Service all flows with finish-time in current frame.
Independent flows vs. inter-dependent flows	Independent flows. But new flows are preferred.
Service saving: limited vs. unlimited	Saving service is limited by the frame length.
Start-time-based vs. finish-time-based	Finish-time based.
Service demands: single-dim. vs. multi-dim.	Single-dimensional, only fair throughput.

Table 2.11: FFQ in Context of Scheduling Classification

2.12 SFQ: Start-Time Fair Queuing

Start-Time Fair Queuing is introduced in [GVC96] and motivated by the common goals of fair schedulers: delay bounds, fairness, efficiently implementable, and in this case support for hierarchical link sharing. SFQ requires to calculate for each arriving packet k of flow i its finishing time E_i^k according to a fluid model and its arrival time A_i^k. In contrast to previous solutions, the authors suggest to schedule packets in increasing order of their start times. The start time S_i^k of a packet k of flow i is defined by $S_i^k = max(A_i^k, E_i^{k-1})$. The finishing tag of a packet is calculated as $E_i^k = S_i^k + \frac{l_i^k}{r_i}$, where l_i^k is the length of the kth packet of flow i and r_i the amount of service flow i receives from the system in a time unit. The finishing times are used to define the virtual timer $v(t)$, which is $v(t) = max_{p_i^k \in F(t)}\{E_i^k\}$, where $F(t)$ is the set of packets serviced until time t. The virtual timer $v(t)$ is used to initiate the arrival time of new packets in the system. The authors show that fluctuation of service a flow receives, i.e. the amount of service that

a flow is lacking to a fair share, is bounded by an exponential probability function. A classification of SFQ according to Chapter 1.2 is presented in Table 2.12.

Scheduling classification	Start-Time Fair Queuing
Approach	WFQ-family
Optimality vs. approximation vs. heuristic	Approximation
Adaptability	No. Flows are bound to their fair share.
Fairness	Yes. Greedy flows have no effect on other flows.
Cooperative vs. individual	Individual, each peer maintains the schedule by its own.
Role-based tasks vs. homogeneous model	All peers are modeled homogeneously equal.
Flow types: hierarchical vs. flat	Flat. All flows are equal.
Priority list vs. frame-based scheduling	Priority list, no frames are used.
Timer-based vs. self-clocking	Self-clocked, calculating new times relative to known ones.
Service rates: dynamic vs. static	Static, each flow receives same amount of bandwidth.
Packet priorities: dynamic vs. static	Static, packet priorities do not change.
Start-time-based vs. finish-time-based	Start-time-based.
Independent flows vs. inter-dependent flows	Independent flows, have no effect on each other.
Service saving: limited vs. unlimited	Service saving limited by maximum packet length.
Service demands: single-dim. vs. multi-dim.	Single-dim.: throughput demand defined by arrival rate.

Table 2.12: SFQ in Context of Scheduling Classification

2.13 WF^{2}Q: Worst-case Fair Weighted Fair Queuing

Worst-case Fair Weighted Fair Queuing or WF^{2}Q has been introduced by Bennet and Zhang in [J.C96] with focus on fairness and bounds on the maximum delay in the system, as well a low computational complexity. They motivate a new fairness metric, by arguing that the simple fairness metric, which compares the service shares received by two flows in the system, does not consider the difference between the service received according to the idealized fluid model (e.g. GPS) and the packet-based solution (e.g. PGPS). They identify the problem of burstiness, where a flow can receive all the service for a long time, while other flows do not receive any service. This comes from the fact that most prior scheduling mechanisms calculate the finish times of incoming packets and use these times to insert the packet in a sorted list. However the finishing time F_i^k of the kth packet of flow i is $F_i^k = A_i^k + \frac{l_i^k}{r_i}$, where A_i^k is the arrival time of the packet, l_i^k its length and r_i the rate the flow receives service from the system. This rate is an average on the service received over time: $r_i = \frac{w_i}{\sum_{j \in F} w_j}$, where w_j is the weight of the flow j in the set of flows F. In reality only one flow is serviced at a time, they do not share service during transmission. By this fact, packets can be faster processed in reality than according to the

idealized fluid model. The authors state packet-based solutions (e.g. WFQ) can be far ahead the idealized fluid model (e.g. GPS) in terms of bits served for a flow. This can cause burstiness.

The authors propose WF^{2}Q as a solution that provides boundaries for the difference of service provided in the fluid model and reality. In WFQ the next packet to be serviced is the one with the smallest finishing time according to the fluid model. In WF^{2}Q the next packet to be served is the one with the smallest finishing time according to the fluid model as well, but an additional requirement is that the packet started to receive service in the fluid model. The eligibility time of the packet, which is defined by the finishing time of the previous packet in the flow, has to start before processing the packet. WF^{2}Q can be ahead GPS only by a fraction of the maximum packet length, whereas WFQ can be ahead by an unbounded amount. The service a flow receives in WF^{2}Q is within a maximum packet length in compare to GPS. This property is called Globally Bounded Timestamp (GBT). In Figure 2.6 we show the main idea behind this smallest eligible finish time first policy.

A big contribution of the authors is the introduction of a new fairness metric, called Worst-case Fair Index (WFI). WFI is defined as the minimal value C_i can have in the following equation: $d_i^k < \frac{|Q_i^k|}{r_i} + C_i$, where d_i^k is the delay the kth packet of flow i has to wait in the queue. $|Q_i^k|$ is the length of the queue including p_i^k on its arrival and r_i is the service share flow i receives from the system. WFI of a flow i is calculated by decreasing C_i to a minimal value for which the equation is still valid. WFI of the system is defined as $WFI_S = max_{i \in F}\{C_i\}$. The WFI metric describes the burst resistance of the metric. According to this metric WF^{2}Q is optimal.

A classification of WF^{2}Q according to Chapter 1.2 is presented in Table 2.13.

2.14 W^{2}F^{2}Q: Wireless Worst-case Fair Weighted Fair Queuing

Yi, Seok and Park analyzed in [YSK$^+$00] the applicability of WF^{2}Q in a wireless network scenario. They discuss that the traffic in wireless networks may be bursty and location-dependent errors may occur, errors which are independent of the behavior of the flows. To be fair, flows that experienced lagging and therefore could not be serviced, should receive more service after the lag to catch up. The authors classify the flows in the system in 3 categories: leading, in-sync and lagging. The main idea of W^{2}F^{2}Q is that leading flows give service time to lagging flows, so that they can compensate for the missed service. A virtual timer is maintained per flow that calculates the amount of service received. If a flow did not receive service due to errors, its virtual time is not increasing. By comparing to the service it should receive according the fluid model GPS, lagging flows can be detected.

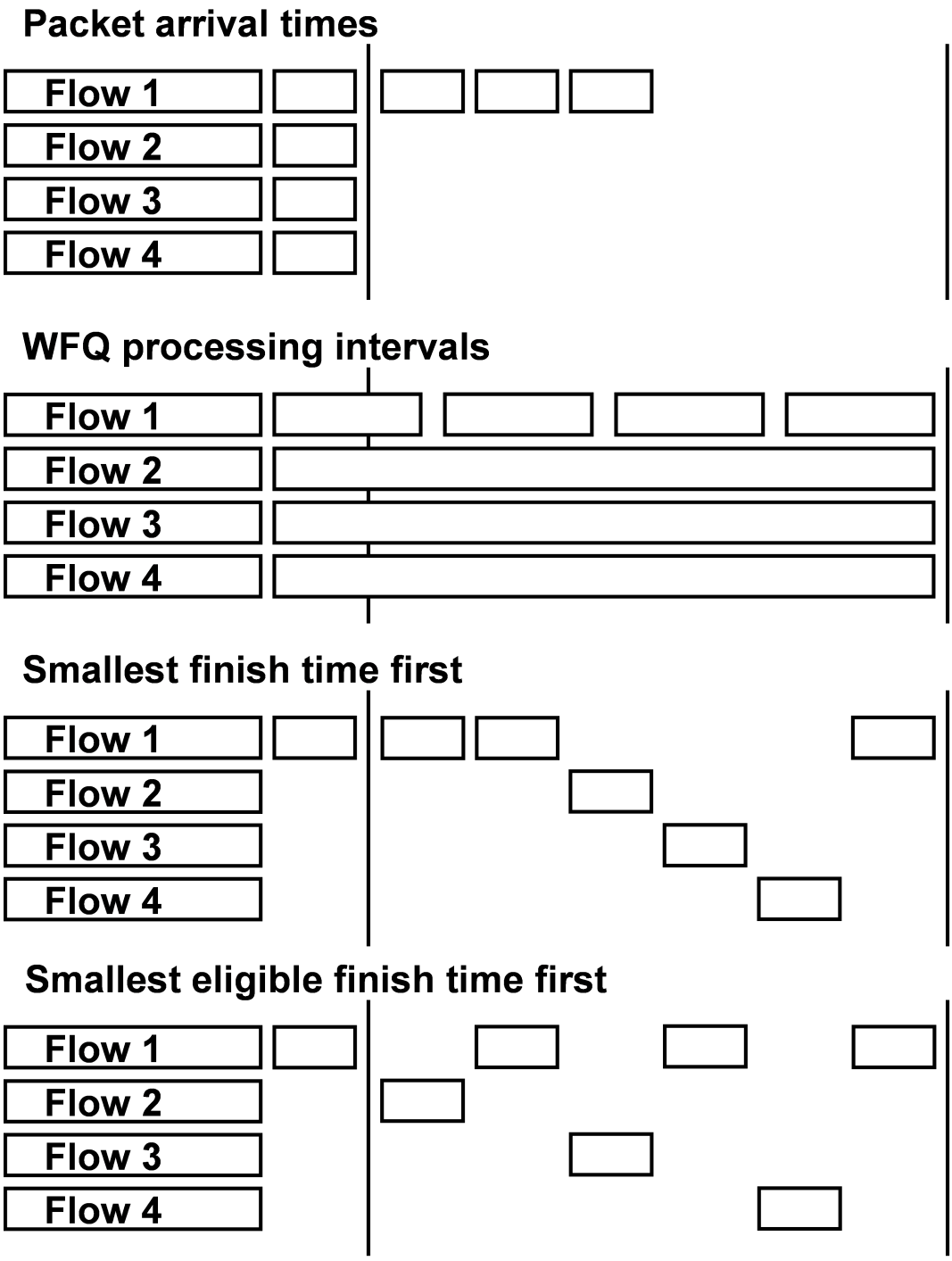

Figure 2.6: In WF^2Q packets are only processed when their eligibility time according to the fluid model has already started. Among these, the packet with the smallest finish time is chosen.

Scheduling classification	Worst-case Fair Weighted Fair Queuing
Approach	WFQ-family
Optimality vs. approximation vs. heuristic	Optimal Worst-case Fair Index
Adaptability	No, higher throughput demand by a flow has no effect.
Fairness	Yes, better WFI than WFQ
Cooperative vs. individual	Cooperation needed to provide tighter bounds on delay.
Role-based tasks vs. homogeneous model	Homogeneous model.
Flow types: hierarchical vs. flat	Flat, no flow aggregation.
Priority list vs. frame-based scheduling	Frame-based: Considers only eligible packets.
Timer-based vs. self-clocking	Timer needed to get arrival times.
Service rates: dynamic vs. static	Static, the weights of the flows are fixed.
Packet priorities: dynamic vs. static	Static, related to arrival rate and packet size.
Start-time-based vs. finish-time-based	Both needed to define the processing time of a packet.
Independent flows vs. inter-dependent flows	Independent flows.
Service saving: limited vs. unlimited	Service saving bound by maximum packet length.
Service demands: single-dim. vs. multi-dim.	Two-dimensional: jitter and throughput.

Table 2.13: WF^2Q in Context of Scheduling Classification

Lagging and leading behavior are defined by thresholds under and above the ideal service share value. Leading flows give a fraction of their service time to lagging flows. They do not pass all of their service, to avoid starvation. They pass at maximum an amount of service to fall under the leading-threshold again. In Figure 2.7 we show the idea that service that has been received to much/less, has to be compensated.

The authors point out that the amount of service that is allowed to be ahead or behind the ideal share is limited. Up from a certain delay packets are dropped. In addition the amount of service to share is restricted. The goal of W^2F^2Q is to achieve fairness among leading and lagging flows. This is done with a rather high complexity but more service provisioning to each flow than in WF^2Q. A classification of W^2F^2Q according to Chapter 1.2 is presented in Table 2.14.

2.15 VC: Virtual Clock

The Virtual Clock algorithm is introduced by Zhang in [Zha91] as a mechanism to control and enforce statistical average transmission rates for reserved flows. Flows in [Zha91] are modeled as consisting of three phases: flow setup, data transmission and flow tear down. The author assumes that for each flow a contract defining the service specification is negotiated during the flow establishment. The Virtual Clock algorithm has three goals: Provide negotiated service, monitor the throughput rate of flows to give feedback to their

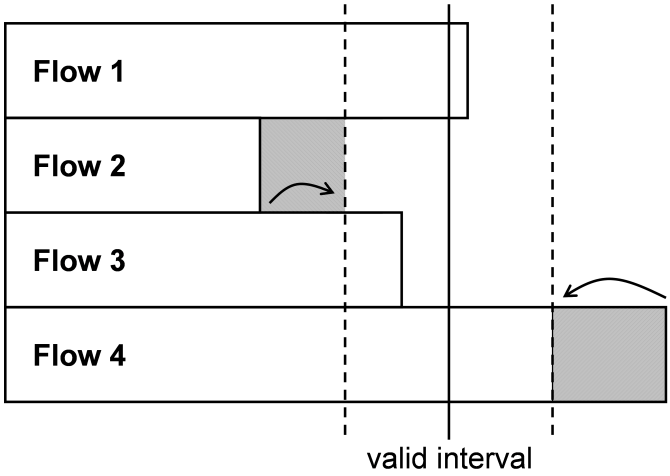

Figure 2.7: In W^2F^2 flows may receive more or less service than expected, due to connectivity problems (lagging). W^2F^2Q proposes to compensate for this service.

Scheduling classification	Wireless Worst-case Fair Weighted Fair Queuing
Approach	WFQ-family
Optimality vs. approximation vs. heuristic	Approximation
Adaptability	No, providing fair share is aimed.
Fairness	Yes, by WFQ-principle and by lagging-compensation.
Cooperative vs. individual	Individual, no cooperation needed.
Role-based tasks vs. homogeneous model	All peers are modeled homogeneously equal.
Flow types: hierarchical vs. flat	3 dynamic flat types of flows: leading, in-sync, lagging.
Priority list vs. frame-based scheduling	Priority list, no time framing.
Timer-based vs. self-clocking	Timer-based, also needed to detect lagging.
Service rates: dynamic vs. static	Dynamic, service rates vary to compensate lagging.
Packet priorities: dynamic vs. static	Static, packet priorities are defined by their length.
Start-time-based vs. finish-time-based	Both, using the same principle (SEFF) as in WF^2Q.
Independent flows vs. inter-dependent flows	Independent flows.
Service saving: limited vs. unlimited	Service saving: bound by maximum packet length and lagging.
Service demands: single-dim. vs. multi-dim.	Two-dimensional, jitter and throughput.

Table 2.14: W^2F^2Q in Context of Scheduling Classification

sources and to isolate individual data flows from each other.

VC assumes constant packet arrival rates for the flows in the system. Two virtual time counters are maintained. The first is used to calculate the finishing times of arriving packets, and the second is used to monitor the traffic behavior of the flows. The finishing

time E_i^k of the kth packet of flow i is defined as $E_i^k = max(E_i^{k-1}, VC_{now}) + \frac{l_i^k}{r_i}$, where VC_{now} is the current Virtual Clock time, l_i^k is the length of p_i^k and r_i the rate of service the flow i receives.

For monitoring VC maintains another virtual time counter to measure the amount of traffic serviced for each flow. The monitoring window is defined by an average interval length AI and an average rate AR in this interval. If the amount of excess traffic a flow produced is higher than a threshold value, the source of the flow is contacted, to report the misbehavior. When the queue contains too many packets, the last message of the largest queue is dropped. The size of AR has some effect on the behavior of VC. Choosing AI small limits the burst-tolerance of the system. Choosing AI large may lead to congestion on occurrence of bursts. Credits for not used service can only be used during the monitoring interval AI.

The idea of resource reservation capabilities of flows has the advantage that traffic behavior is easier to predict, however the consequences are that malicious users may choose to reserve more resources than they need to have a backup. A classification of VC according to Chapter 1.2 is presented in Table 2.15.

Scheduling classification	Virtual Clock
Approach	Virtual Clock family
Optimality vs. approximation vs. heuristic	Heuristic
Adaptability	No, all flows are bound to average service rates
Fairness	No, greedy flows can use eventually all bandwidth.
Cooperative vs. individual	Individual, no cooperation needed.
Role-based tasks vs. homogeneous model	Homogeneous model: all peers are modeled equal.
Flow types: hierarchical vs. flat	Flat, no flow types defined.
Priority list vs. frame-based scheduling	Frames used to monitor resource usage of flows.
Timer-based vs. self-clocking	Self-clocking, VC calculated using in/out rates.
Service rates: dynamic vs. static	Dynamic, service can vary, only average is important.
Packet priorities: dynamic vs. static	Static, packets have same priority in every peer.
Start-time-based vs. finish-time-based	Finish-time-based.
Independent flows vs. inter-dependent flows	Inter-dependent flows.
Service saving: limited vs. unlimited	Service can be saved for a whole monitoring period.
Service demands: single-dim. vs. multi-dim.	Single-dimensional, only average throughput matters.

Table 2.15: VC in Context of Scheduling Classification

2.16 LFVC: Leap Forward Virtual Clock

Leap Forward Virtual Clock is an extension to Virtual Clock introduced by Suri, Varghese and Chandranmenon in [SVC97]. The authors identify three conditions that must be met in order to provide similar delay and throughput bounds like WFQ.

1. Backlog inequality:

$$\sum_{i \in F_t} l_i + \sum_{i \in F_t} r_i \cdot (t - t_i) \leq (t - t_s) \cdot B$$

   In this equation F_t is the set of flows that has packets with deadlines before time t, l_i the length of the next packet in the flow i, r_i the guaranteed rate in bytes per time unit, t a future server clock value, t_s the current server time, and B the output rate of the system. This inequation is valid in the case that all packets currently backlogged and all packets that may enter the system until the time t, can be processed until time t, because the servers output rate is big enough.

2. Delay condition:
   The delay condition is fulfilled when the backlog inequality is valid for any time t. This means that traffic is always processed in its time window, i.e. faster than WFQ would process it.

3. Throughput condition:

$$\forall t_s, f \in F \ : \ t_f \leq t_s + k \cdot \delta_f$$

   This means that every flow receives only a multitude of δ_f more service than it is guaranteed. $\triangle_{max}$ is the time needed to process a packet of maximum length. This condition effects that the WFI of LFVC is near optimal.

LFVC picks up the idea of leading and lagging flows and introduces therefore two separate queues: H and L. The queue labeled H contains all flows that are *lagged* or *in-sync*. Queue L contains the *leading* flows that has to be degraded in service. The flows in queue L are sorted by the key $E_i^k - \triangle_{max}$, where E_i^k is the finishing time of the next packet in flow i. At the beginning all flows are in H. The server takes the flow i with the lowest finishing tag from H for processing next. The server time t_s is increased by $\frac{l_i^{next}}{B}$ and after processing and transmission the packet is deleted from the system. If a flow is endangering the throughput condition, i.e. it received more service than its allocated share of bandwidth, it is put in queue L. Flows stay in L until they endanger the delay condition, i.e. some packets may violate their deadlines. A flow is taken out of L and put in H, when

there is only $\triangle_f$ time left to process the packet. If H is empty and only L has entries, the server time has to be advanced (leap forward virtual clock) so that $\triangle_f$ time remains for a flow to process its next packet before violating its deadline. In this case the flow is put in H and the system continues picking flows from H.

In order to decrease the computational complexity, the authors suggest to use rounded tags. With this accuracy is decreased as well. They propose that all tags shall be multitudes of a constant. The computation of tags shall be done with full accuracy but they shall be rounded before insertion in H. With this LFVC has a complexity of O(log log N), where N is the number of active flows. A classification of LFVC according to Chapter 1.2 is presented in Table 2.16.

Scheduling classification	Leap Forward Virtual Clock
Approach	Virtual Clock family
Optimality vs. approximation vs. heuristic	Approximation
Adaptability	No, the per-flow service share is bounded.
Fairness	Yes, all packets are processed until their deadlines.
Cooperative vs. individual	Individual, no cooperation needed.
Role-based tasks vs. homogeneous model	All peers are modeled homogeneously equal.
Flow types: hierarchical vs. flat	2 flat flow types, low and high priority.
Priority list vs. frame-based scheduling	Priority list, framing is not used.
Timer-based vs. self-clocking	Timer is needed to meet deadlines.
Service rates: dynamic vs. static	Dynamic, depends on dyn. ranking of the flow in L/H.
Packet priorities: dynamic vs. static	Static, packets keep their priorities.
Start-time-based vs. finish-time-based	Finish-time-based: deadlines matter.
Independent flows vs. inter-dependent flows	Inter-dependent flows (but only benefiting)
Service saving: limited vs. unlimited	Service saving, bound by avg. service from system.
Service demands: single-dim. vs. multi-dim.	2 dimensional: throughput and delay.

Table 2.16: LFVC in Context of Scheduling Classification

2.17 HLS: Hierarchical Link Sharing

Floyd and Jacobson introduce in [FJ95] the idea of heterogeneous hierarchical flow types. Some flow types are latency-critical, some not. Additionally the flows belong to different agencies. A fair bandwidth allocation has to be provided for all agencies and all of their flows. The authors want to provide a solution that is both supporting latency-critical flows and does not choke other flows to starvation. HLS models the system as a tree with the output resource as root. In the next level the agencies are located, then the traffic classes with differing real-time and bandwidth demands, and finally as leafs the single

applications. Each parent node distributes its share of service received from a level higher, to its child nodes using static or dynamic fractions. The hierarchical link-sharing tree guarantees for each leaf a certain amount of service. A general scheduler is implemented to pick a packet from a leaf node for transmission. The general scheduler takes the shares reserved for each leaf into account. The tree that is built in HLS may be modeled like shown in Figure

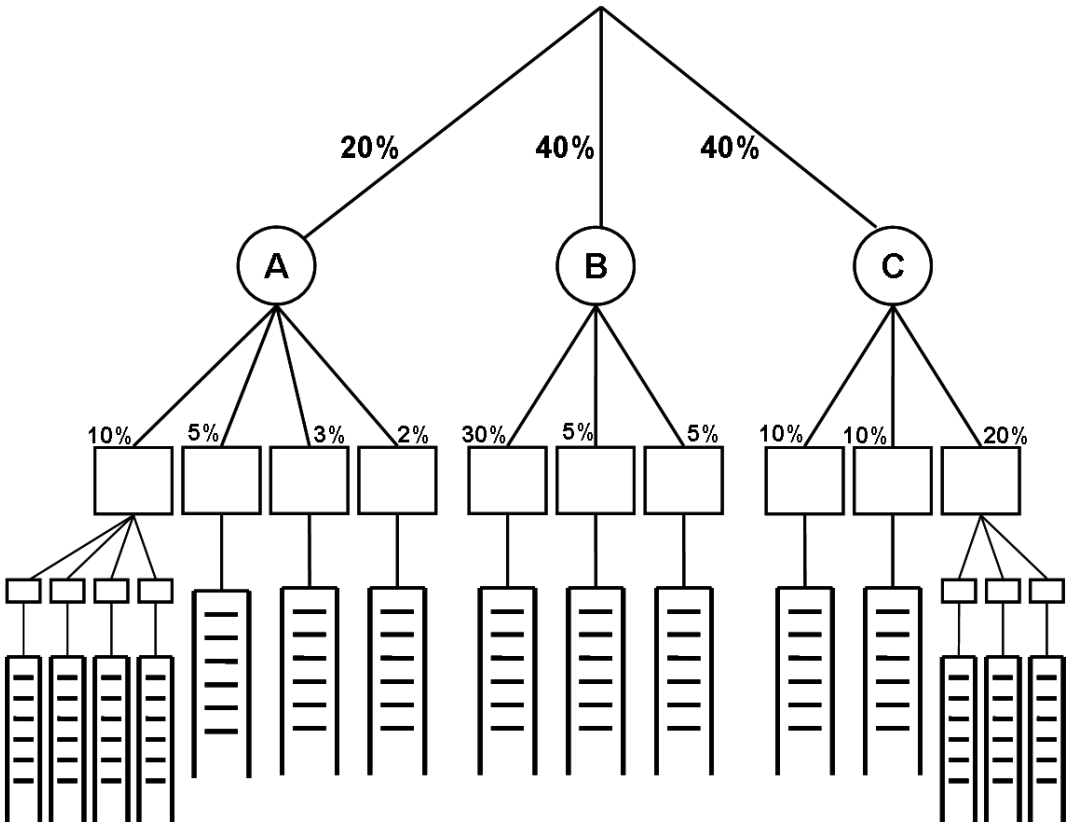

Figure 2.8: This figure shows the hierarchical structure of flows in HLS.

The link-sharing goals are:

- Each class should receive approximately its allocated bandwidth. Upon congestion, all classes are limited to their allocated bandwidth
- Unused. Excess bandwidth should be distributed flowing a set of guidelines

Each class has its own queue, new arriving packets are classified and sorted into the

appropriate classes. If no congestion exists the general scheduler chooses packets to transmit next on the output link. Congestion is detected with an estimation module which checks whether each class gets its fair share. When congestion is detected, the link sharing scheduler is activated, to take share from classes that received more service than they reserved and allocate this amount of share to unsatisfied classes, i.e. classes that received less service than they reserved.

The authors provide three different strategies from which leafs to take bandwidth in order to give to the unsatisfied classes. The following enumeration lists these three conditions that must be fulfilled to keep all the bandwidth allocated to a class A that received more service than it reserved. If the condition is invalid, the share of class A is decreased.

1. Formal link sharing guideline: A has a not-overlimit ancestor at level i and there are no unsatisfied classes at levels below i

2. Ancestor only link sharing guideline: A has an ancestor that received less bandwidth than its allocated share (underlimit).

3. Top-level approach: A has an underlimit ancestor whose level in the tree is at most lvl_{max} high, where lvl_{max} is a system-wide variable.

So if A did not get more service than allocated or one of these strategies count (depending on which strategy one uses), A does not lose allocated share. A classification of HLS according to Chapter 1.2 is presented in Table 2.17. Each lead class uses classical schedulers to utilize its share.

2.18 WF^{2}Q+: Worst-case Fair Weighted Fair Queuing +

Bennett and Zhang introduced in [BZ97] an extension to WF^{2}Q, which provides small delay bounds like WF^{2}Q, a small WFI like WF^{2}Q, but has computational complexity of only O(log N) instead of O(N), where N is the number of flows. WF^{2}Q+ builds a virtual tree for link-sharing and provides separate mechanisms for real-time traffic management and best-effort traffic management. Each inner node maintains a logical queue keeping only a reference to the packet of one of his child nodes which has to be processed next. The node uses hereby the Smallest Eligible Virtual Finish-Time First (SEFF) principle, like in WF^{2}Q, to pick a packet among all available. All physical packets are stored in the physical queue of one leaf node. Leaf nodes receive service only from their father nodes, the service rate is not constant. Any time there exist a path from one leaf node to the root. And all logical queues on this path point to the same physical packet in a leaf's queue

Scheduling classification	Hierarchical Link Sharing
Approach	Hierarchical scheduler
Optimality vs. approximation vs. heuristic	Approximation
Adaptability	Yes, flow can change their demands.
Fairness	Yes, compensation mechanisms exist.
Cooperative vs. individual	Individual, no cooperation needed.
Role-based tasks vs. homogeneous model	Homogeneous model, all peers are equal.
Flow types: hierarchical vs. flat	Hierarchical, organizing various aggregation levels.
Priority list vs. frame-based scheduling	Priority list, no time framing needed.
Timer-based vs. self-clocking	Self-clocking.
Service rates: dynamic vs. static	Dynamic, to compensate too much/less service.
Packet priorities: dynamic vs. static	Not applicable.
Start-time-based vs. finish-time-based	Not applicable.
Independent flows vs. inter-dependent flows	Inter-dependent flows, effect each others share.
Service saving: limited vs. unlimited	Service saving is limited.
Service demands: single-dim. vs. multi-dim.	2-dimensional: throughput and latency

Table 2.17: HLS in Context of Scheduling Classification

that is processed next. The logical queues and the path is updated each time a packet arrives that is inserted in a previously empty queue, and each time after a packet has been processed and transmitted.

In this solution the service rate a leaf node receives is not constant. So that the amount of service received is not proportional to time anymore. A new reference time per node is introduced. The virtual time in node n is defined as: $T_n = \frac{W_n(0,t)}{r_n}$, where $W_n(0,t)$ is the amount of service received by node n in the time interval from 0 to t and r_n is the rate guaranteed to node n. In addition to this a virtual time counter exists. The virtual time counter progresses upon arrival of a packet in an empty queue or upon departure of a packet from the system. The virtual time is used to calculate the start and finish time of the head in a queue. In contrast to WF^2Q this algorithm maintains a start and finish time per queue and not per packet. In Figure 2.7 we show the idea of the WF^{2}Q+ algorithm.

We now describe the algorithm of WF^{2}Q+. Upon arrival of a packet p_i^k as kth packet in flow i. p_i^k is enqueued in the physical queue of flow i. If the queue Q_j of the father node j of flow i is not empty, then nothing more has to be done. Otherwise the packet is inserted in Q_j and the virtual time, the start-time and finish-time of the inserted packet is updated. The start time S_i is updated by following function: $S_i = F_i$ if the queue was empty before, and $S_i = max(F_i, V(A_i))$ otherwise, where F_i is the finish-time of the previous packet in the queue and $V(A_i)$ the virtual time of queue i at time the packet has been enqueued. This means that the eligibility time of the enqueued packet starts directly

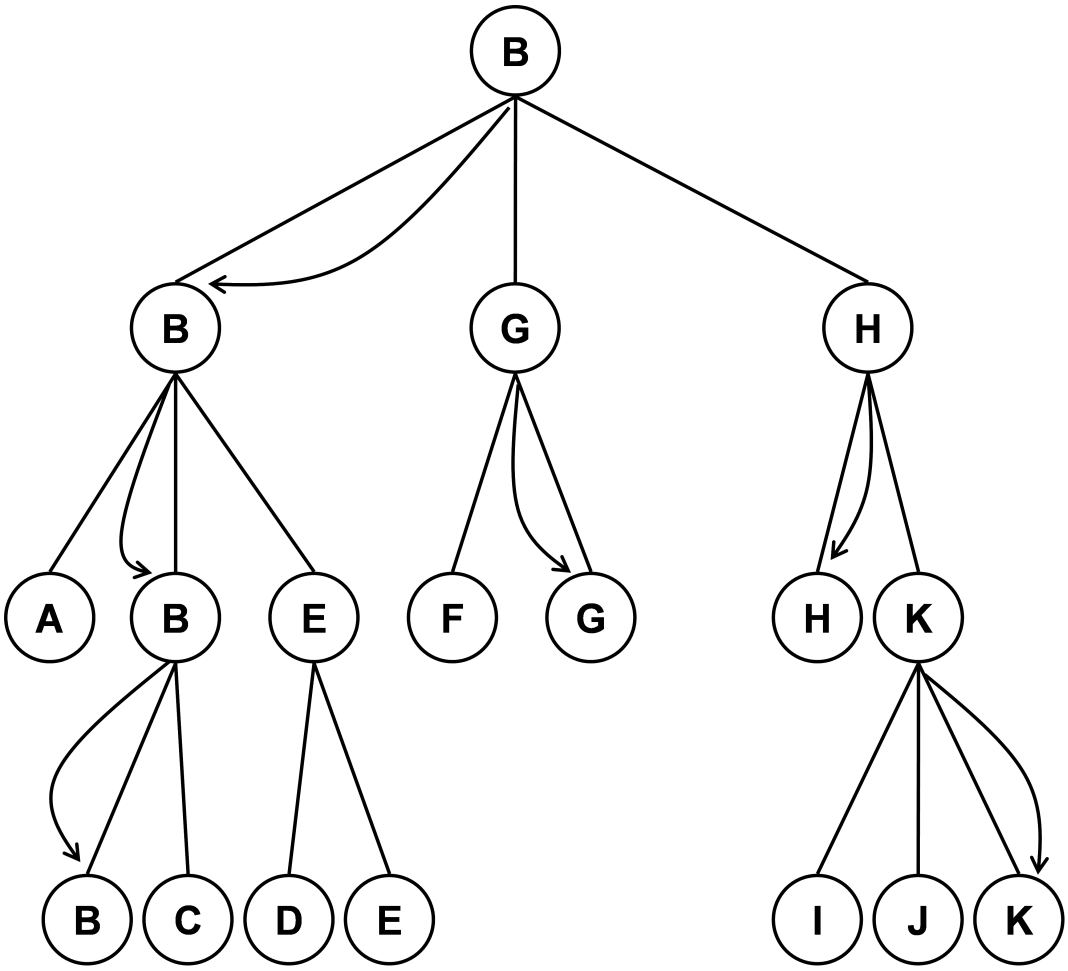

Figure 2.9: In WF^{2}Q+ inner queues are virtual and pointing only at the best choice among their child nodes.

after the finish time of the previous packet or after the virtual time is reached that existed at the time the packet was enqueued. The update function of the virtual time is defined as followed $V_{WF^2Q+}(t + \triangle) = max(V_{WF^2Q+}(t) + \triangle, min_{i \in F}(S_i^1))$, where $\triangle$ is the real-time elapsed since the last update of the virtual time, F is the set of backlogged flows and S_i^1 the start time of the heading packet in the physical queue of flow i. However, after updating these values the node is restarted if it was idle before. Upon restarting the node, the node picks a packet among its child nodes using the SEFF strategy and transmits either the packet (if root) or restarts its parent node. The path is build from bottom to top, finally the root node transmits a packet and resets after transmission the path. By resetting the path an inner or root node, selects an active child and calls the reset-path function of this child. When the reset-path function is called at a leaf node, it gets its next packet, as the previous one has been transmitted, updates the start and finish time and restarts its parent node.

The focus of the algorithm ascends up the tree fixing the time counters and time stamps and arranging the most suitable packets to be chosen a level higher. On each level the logical queue is filled with the most suitable packet among all queue heads of child nodes, after this procedure the tree is traversed one level higher. The root node then transmits a packet and calls a function to descend the tree again and substitute the currently sent packet by a new one. Upon reaching a leaf node the algorithm ascends the tree again.

The authors introduce a new fairness metric: Bit Worst-case Fair Index (BWFI) to cope with the new virtual time. A server s has a BWFI of $\alpha_{i,s}$ for session i, if during any time backlogged interval [t_1,t_2] of queue Q_i the following holds: $W_i(t_1, t_2) \geq \frac{r_i}{r_s} W_s(t_1, t_2) - \alpha_{i,s}$. This means that the work flow i received in this time period is not smaller than $\alpha_{i,s}$ in compare to its fair share of total work provided by the server. A Service Burstiness Index (SBI) of $\alpha_{i,s}$ is guaranteed by the server s for the flow i, if for any given time t_2 exists t_1 so that the equation above holds.

The authors provide with WF^{2}Q+ a solution that is as good as WF^{2}Q in terms of delay and WFI, but has only a complexity of O(log N), where N is the number of flows. A classification of WF^{2}Q+ according to Chapter 1.2 is presented in Table 2.18.

2.19 H-FSC: Hierarchical Fair Service Curve

Stoica, Zhang and Ng propose in [SZN97] an hierarchical fair queuing solution based on service curves. A service curve models delay and bandwidth requirements of a flow. For a better understanding we give a definition of service curve. A service curve is a monotonically growing function that maps time to an amount of service. In many cases

Scheduling classification	Worst-case Fair Weighted Fair Queuing +
Approach	Hierarchical
Optimality vs. approximation vs. heuristic	Approximation
Adaptability	No, fair share for all flows is aimed.
Fairness	Yes, SEFF policy (like in WF^{2}Q) guarantees fairness.
Cooperative vs. individual	Individual, no cooperation needed.
Role-based tasks vs. homogeneous model	All peers are modeled homogeneously equal.
Flow types: hierarchical vs. flat	Flat types organized in hierarchical queues.
Priority list vs. frame-based scheduling	Priority list, no time framing needed.
Timer-based vs. self-clocking	Self-clocking, using only relative times.
Service rates: dynamic vs. static	Static, due to tight fairness bounds.
Packet priorities: dynamic vs. static	Static, packet priorities result from their size.
Start-time-based vs. finish-time-based	Both, using the SEFF policy
Independent flows vs. inter-dependent flows	Independent flows.
Service saving: limited vs. unlimited	Service saving is bounded by maximum packet length.
Service demands: single-dim. vs. multi-dim.	2-dimensional: delay and throughput

Table 2.18: WF^{2}Q+ in Context of Scheduling Classification

it is a piecewise linear function. A flow i is guaranteed a service curve $S_i(t)$, if for all time instances t_2 a time t_1 exist, where t_1 is the start time of the backlog period, with $S_i(t_2 - t_1) \leq w_i(t_1, t_2)$, where $w_i(t_1, t_2)$ is the amount of service flow i received between t_1 and t_2. This means that the service curve defines the minimum amount of service a flow receives in a specific period of time.

The authors aim at three goals in the design of H-FSC.

1. Each flow shall receive at least an amount of service corresponding to its guaranteed service curve.

2. To meet real-time constraints, some non real-time packets have to be delayed. This delay should be kept minimal.

3. Excess bandwidth that is not needed to meet real-time deadlines, shall be allocated in a fair way.

The main task of H-FSC is to meet real-time constraints of packets and besides that share the remaining bandwidth in a fair fashion. In the first step of the algorithm all latency-critical flows received at least so much service that even in the worst case when all inactive flows get active, there is enough bandwidth available so that all of the service curves can be guaranteed. The rest of the bandwidth shall be distributed using link sharing criteria.

To meet the first goal, the Service Curve Earliest Deadline First (SCED) principle is used. For all packets the deadlines are calculated using the flows deadline curve. The

function $D_i(t) = min_{t_1}(S_i(t - t_1) + w_i(t_1))$ defines the minimum amount of service provided to flow i in a period of time of length t, starting at t_1. The deadline of a packet of length l_i^k at the head of flow i is defined as $d_i = D_i^{-1}(w_i(t) + l_i^k)$., which gives the time t at which at least l_i^k service more will be provided to flow i. And as p_i^k is the next packet to be processed, p_i^k is the packet that has to be processed until time t to guarantee that the service curve is hold.

To decouple the requirements for bandwidth and real-time delay bounds the authors suggest to use non-linear service curves. Furthermore they show that the SCED principle can guarantee all service curves, as long as the sum of all service curves, i.e. the sum of the minimal amount of service to receive by the flows, is smaller than the server capacity.

To be sure that all latency-critical packets are processed before their deadlines, each time-critical flow shall receive $E(t)$ service which is the minimum amount of service time that all active flows should receive by time t. An amount of $E(t)$ service is allocated to all active flows using SCED, the remaining bandwidth is shared according link-sharing rules. The system is modeled as a virtual tree, each flow is modeled as a leaf, maintaining the start and finish time $S_i(\cdot)$ and $E_i(\cdot)$ of its first packet and a counter measuring the service received by flow i. An inner node is choosing its child to service according the Smallest Start Time First (SSTF) principle. The virtual system time V_i^s of an inner node i is defined by $V_i^s = \frac{V_{i,min}+V_{i,max}}{2}$, the average of the minimum and maximum start times of its child nodes. The virtual time counters are updated after a packet has been processed by the root node or a class becomes active, i.e. a new packet arrive at a leaf, which has previously been empty. The virtual timers are updated recursively from the leaf to the root node. By setting the average start time in each inner node, and using the SSTF principle, the root node knows which packet to process. A classification of H-FSC according to Chapter 1.2 is presented in Table 2.19.

2.20 CSFQ: Core-Stateless Fair Queuing

Core-Stateless Fair Queuing (CSFQ) is introduced in [SSZ98] by Stoica, Shenker and Zhang. Their main goal is an efficient fair queuing algorithm with strong complexity reduction. This is achieved by introducing two types of devices: edge and core routers. Core routers are surrounded by edge routers so that all traffic coming from the rest of the network has to pass an edge router before coming to a core router. Edge routers estimate the traffic at the edge of this network island and label packets with the rate of their flows. Core routers use these labels to calculate a minimum service rate for all flows. Upon congestion packets that exceed a specific threshold above the minimum service rate are dropped. In Figure 2.10 we present the topology required by CSFQ..

Scheduling classification	Hierarchical Fair Service Curve
Approach	Hierarchical
Optimality vs. approximation vs. heuristic	Approximation
Adaptability	Yes, flows are serviced according their service curves.
Fairness	Yes, in terms of both latency and bandwidth.
Cooperative vs. individual	Individual, no cooperation needed.
Role-based tasks vs. homogeneous model	Homogeneous, all peers are equal.
Flow types: hierarchical vs. flat	Flat, but all flows have own service curves
Priority list vs. frame-based scheduling	Priority list, no framing needed.
Timer-based vs. self-clocking	Timers are needed for the service curve calculations.
Service rates: dynamic vs. static	Static, service rate is bound to service curve.
Packet priorities: dynamic vs. static	Static, flows keep their service curves.
Start-time-based vs. finish-time-based	Uses start time to determine processing order.
Independent flows vs. inter-dependent flows	Inter-dependent flows, may receive excess bandwidth.
Service saving: limited vs. unlimited	No service saving, as excess bandwidth is shared fair.
Service demands: single-dim. vs. multi-dim.	Multi-dimensional, depends on service curve model.

Table 2.19: H-FSC in Context of Scheduling Classification

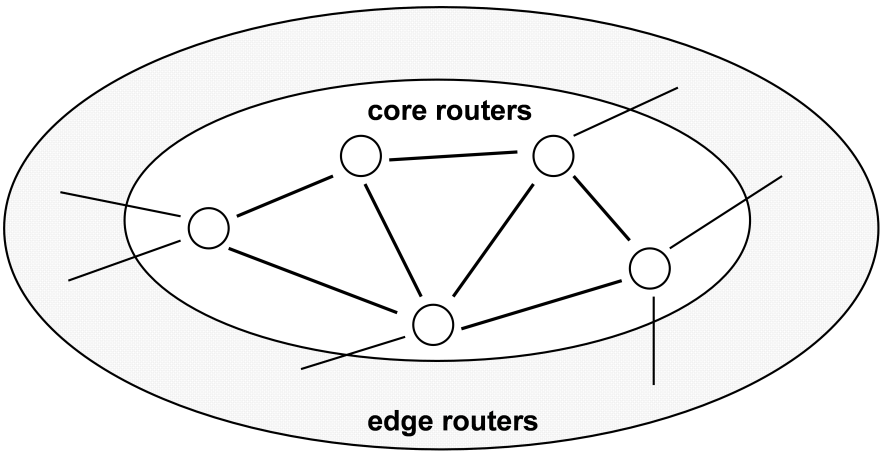

Figure 2.10: This figure shows the topology assumed by Core-Stateless Fair Queuing. Edge routers are expected to surround all core routers and separate them from legacy routers in the Internet.

The fluid model assumes fix arrival rates of the flows. A maximum service rate $\alpha(t)$ is introduced, $\alpha(t)$ is defined as the solution to $B = \sum_{i \in F} min(r_i(t), \alpha(t))$, where B is the output rate of the server, F the set of flows and $r_i(t)$ the service rate of the flow. In the packet-based model the arrival rates are not known, they have to be estimated at the edge

routers. When the sum of the arriving packets exceeds the output capacity i.e. congestion exists, then $\alpha(t)$ is calculated as maximum value for that the equation holds. $\alpha(t)$ is the maximum service rate a flow is allowed to receive, i.e. flow i receives $min(r_i(t), \alpha(t))$ amount of service per time unit.

As the rate estimators may be imprecise, congestion can still occur. In that case $\alpha(t)$ is decreased by 1% each time, at maximum by 25%. Packets that traverse the core network are relabeled according their service rate: $L_{new} = min(L_{old}, \alpha(t))$.

The authors introduce Weighted CSFQ as an extension to CSFQ. Each flow i is assigned a weight w_i that has impact on the share the flow receives. Upon congestion the maximum service rate $\alpha(t)$ is defined by the solution of $B = \sum_{i \in F} w_i \cdot min(\frac{r_i(t)}{w_i}, \alpha(t))$. Packets are dropped using as dropping probability $max(0, 1 - \alpha \cdot \frac{w_i}{r_i})$. The higher the weight of a flow, the smaller the probability that packets of this flow are dropped. Flow i with weight w_i receives in the time interval $[t_1, t_2]$ not more than $w_i \cdot \alpha \cdot (t_2 - t_1)$.

The authors present the concept of Max-Min-Fairness: You cannot add to a flow i more service without taking service from a flow j that has already less service than i. A classification of CSFQ according to Chapter 1.2 is presented in Table 2.20.

Scheduling classification	Core-Stateless Fair Queuing
Approach	Dynamic Packet State
Optimality vs. approximation vs. heuristic	Optimal fair share
Adaptability	Yes, optimal share is calculated dynamically.
Fairness	Yes, greedy flows bound to their fair share.
Cooperative vs. individual	Needs cooperation, one failing node breaks the system.
Role-based tasks vs. homogeneous model	2 roles: edge and core routers.
Flow types: hierarchical vs. flat	Flat, flow demands are expressed in packet labels.
Priority list vs. frame-based scheduling	Priority list, no frames needed.
Timer-based vs. self-clocking	Self-clocking, timing is less important.
Service rates: dynamic vs. static	Dynamically changed to find optimal utilization.
Packet priorities: dynamic vs. static	Dynamic, packet prio. are matched to flow behavior.
Start-time-based vs. finish-time-based	Start-time-based order of packets their flow's queue.
Independent flows vs. inter-dependent flows	Inter-dependency of flows for calculating opt. share.
Service saving: limited vs. unlimited	No service history is maintained.
Service demands: single-dim. vs. multi-dim.	Single-dimensional: packet priorities/flow demands.

Table 2.20: CSFQ in Context of Scheduling Classification

2.21 SV-CSFQ: Self-Verifying Core-Stateless Fair Queuing

Stoica, Zhang and Shenker propose in [INF02] Self-Verifying Core-Stateless Fair Queuing (SV-CSFQ) as an extension to CSFQ. They argue that the concept of having edge and core routers is not applicable, because it is infeasible to isolate an island of core routers by surrounding them with edge routers. In addition any malicious core router could cause severe harm, by relabeling the packets. The authors suggest in [INF02] to use only one kind of routers that periodically check the validity of packet labels. In case of inappropriate labels, packets are relabeled and the service rate is adapted.

The algorithm assumes that end hosts label their packets according to their sending rate. The core system is in charge to verify these labels and to correct them in case of too strong deviation. The verification is done in the router by randomly selecting a flow that is not already in the verification process. The chosen flow is then monitored within a period of time. In this time its arrival rate is computed and compared to its label. When the relative error exceeds a specific threshold and the label is lower than the real flow rate, it is assumed that the flow is misbehaving and hiding its usage of additional bandwidth. In this case the flow has to be contained and the additional service received before has to be compensated. To do this, the packet is relabeled with a value exceeding the real flow rate. The relabeling stops the false allocation of bandwidth. After a period of time the packets corresponding to the contained flow are labeled correctly again, containment is stopped. The punishment time must be greater than the verification interval, so that malicious false labeling is disadvantageous for end hosts. A classification of SV-CSFQ according to Chapter 1.2 is presented in Table 2.21.

Scheduling classification	Self-Verifying Core-Stateless Fair Queuing
Approach	Dynamic Packet State
Optimality vs. approximation vs. heuristic	Heuristic
Adaptability	Yes, optimal share is calculated dynamically.
Fairness	(Yes), probabilistic verification of misbehaving flows.
Cooperative vs. individual	Individual, but expects correct flow labeling of end hosts.
Role-based tasks vs. homogeneous model	2 roles: end hosts and core routers.
Flow types: hierarchical vs. flat	Flat, flow demands are expressed in packet labels.
Priority list vs. frame-based scheduling	Priority list, no frames needed.
Timer-based vs. self-clocking	Self-clocking, timing is less important.
Service rates: dynamic vs. static	Dynamically changed to find optimal utilization.
Packet priorities: dynamic vs. static	Dynamic, packet priorities are matched to flow behavior.
Start-time-based vs. finish-time-based	Start-time-based order of packets in their flow's queue.
Independent flows vs. inter-dependent flows	Inter-dependency of flows for calculating optimal share.
Service saving: limited vs. unlimited	No service history is maintained.
Service demands: single-dim. vs. multi-dim.	Single-dimensional: packet priorities/flow demands.

Table 2.21: SV-CSFQ in Context of Scheduling Classification

Chapter 3

Classification of Selected Scheduling Mechanisms

In this chapter we present a taxonomy of the scheduling mechanisms presented in Chapter 2 using the classifications introduced in Section 1.2.

3.1 Classification According to the Approach of the Solution

Table 3.1 shows a comparison of the approaches the surveyed scheduling mechanisms use. We classified in following algorithm-families: deterministic, EDF-related, round robin, WFQ-related, VC-related, hierarchical and dynamic packet state based schedulers. The deterministic and EDF-related schedulers are the easiest, round robin schedulers introduced fairness. The WFQ-family is even more fair, as they approximate the ideal fluid model better. VC-related solutions introduce two differentiated flow classes. Hierarchical solutions introduce flow classes that are serviced according they class-related service demands. And finally solutions using dynamic packet states label packets according to their flow demands in order to save complexity and state.

3.2 Classification According to Optimality

In Table 3.2 we show the optimality of the surveyed solutions. Some mechanisms are optimal according to some metric, others try to approximate optimal states, the third group we present is based on heuristics. However only a few mechanisms introduce new metrics, to which the authors present optimal solutions.

	Scheduling classification regarding the solution approach
FIFO	Deterministic
RM	Deterministic
EDF	EDF-family, static packet arrival rates.
RCSP	EDF-family: rate controlled EDF
FQ	Round robin
WRR	Round robin
DRR	Round robin
BR	WFQ-family
WFQ	WFQ-family
SCFQ	WFQ-family
FFQ	WFQ-family
SFQ	WFQ-family
WF^2Q	WFQ-family
W^2F^2Q	WFQ-family
VC	Virtual Clock family
LFVC	Virtual Clock family
HLS	Hierarchical
WF^2Q+	Hierarchical
H-FSC	Hierarchical
CSFQ	Dynamic Packet State
SV-CSFQ	Dynamic Packet State

Table 3.1: Scheduling Classification Regarding the Approach of the Solution

3.3 Classification According to the Adaptability

This classification provides only two types: Either the proposed solution enables changes in the service provision or not. Many solutions bound the service for a flow to a specific fair share. These algorithms do not allow individual flows to receive more service, than their fixed share. Still some mechanisms provide mechanisms to adapt the demands of a flow, and to adapt the service the flow receives. In Table 3.3 we present a comparison of the surveyed schedulers in context of adaptability.

	Scheduling classification regarding the optimality
FIFO	Optimal stateless, minimal overhead: $O(1)$
RM	Optimal for fixed priorities
EDF	Optimal: Provides a schedule if one exist.
RCSP	Approximation
FQ	Approximation
WRR	Heuristic
DRR	Approximation of optimal fairness.
BR	Optimal approximation
WFQ	Optimal approximation
SCFQ	Approximation
FFQ	Approximation
SFQ	Approximation
WF^2Q	Optimal Worst-case Fair Index
W^2F^2Q	Approximation
VC	Heuristic
LFVC	Approximation
HLS	Approximation
WF^2Q+	Approximation
H-FSC	Approximation
CSFQ	Optimal fair share
SV-CSFQ	Heuristic

Table 3.2: Scheduling Classification Regarding the Optimality

3.4 Classification According to Fairness

Fairness can either be guaranteed or not. There is no third option. Most of the newer scheduling mechanisms stick the share of bandwidth a flow can receive to a specific amount. In this case greedy malicious flows cannot benefit on the costs of other peers. However several fairness metrics are defined. Whereas the Min-Max-Fairness is a condition that can be fulfilled or not, the fairness metric defined by Golestani [gol94] (comparing the maximum and minimum amount of service received by flows in the system) is expressed in service bounds. Finally the Worst-Case Fair Index is introduced with WF^2Q and considers the maximum derivation to the optimal fluid model as well. In Table 3.4 we

	Scheduling classification regarding adaptability
FIFO	Yes, changes are directly adopted.
RM	No changes in the packet arrival rate considered.
EDF	Yes. Service rate adapts to arrival rate.
RCSP	No. Service for flows is bound to reservations.
FQ	No, each flow receives a static share of bandwidth.
WRR	No. Static share per flow.
DRR	No. Share of bandwidth per flow does not change.
BR	No. Share of bandwidth for each flow is static.
WFQ	No. Share of bandwidth for each flow is static.
SCFQ	Yes, the system takes changing packet sizes into account.
FFQ	No, the share for each flow is static and equal.
SFQ	No. Flows are bound to their fair share.
WF^2Q	No, higher throughput demand by a flow has no effect.
W^2F^2Q	No, providing fair share is aimed.
VC	No, all flows are bound to average service rates
LFVC	No, the per-flow service share is bounded.
HLS	Yes, flow can change their demands.
WF^2Q+	No, fair share for all flows is aimed.
H-FSC	Yes, flows are serviced according their service curves
CSFQ	Yes, optimal share is calculated dynamically.
SV-CSFQ	Yes, optimal share is calculated dynamically.

Table 3.3: Scheduling Classification Regarding Adaptability

present a comparison of the surveyed schedulers in context of fairness.

3.5 Classification According to Cooperativeness

Cooperation among peers using a specific scheduling solution is often not needed. For many scheduling approaches it is sufficient that each peer maintains the scheduler on its own. However, some mechanisms require the cooperation and collaboration of numerous peers to provide correct results. This is often considered as drawback, as malfunctioning or malicious peers can sabotage the functionality of the scheduler. In Table 3.5 we present

	Scheduling classification regarding fairness
FIFO	No
RM	No. Service share is proportional to arrival rate.
EDF	Yes. All deadlines are met if a valid schedule exists.
RCSP	No, has to be done during reservation process.
FQ	Fair. Per flow same amount of service.
WRR	Yes, each flow is guaranteed a weighted share.
DRR	Yes. Taking also packet sizes into account.
BR	Yes. Greedy flows do not benefit.
WFQ	Yes. Greedy flows do not benefit.
SCFQ	FQNP: No, FQFQ:Yes.
FFQ	Yes, greedy flows are bounded to their fair share.
SFQ	Yes. Greedy flows have no effect on other flows.
WF^2Q	Yes, better WFI than WFQ
W^2F^2Q	Yes, by WFQ-principle and by lagging-compensation.
VC	No, greedy flows can use all bandwidth for periods of time.
LFVC	Yes, all packets are processed until their deadlines.
HLS	Yes, compensation mechanisms exist.
WF^2Q+	Yes, SEFF policy (like in WF^2Q) guarantees fairness.
H-FSC	Yes, in terms of both latency and bandwidth.
CSFQ	Yes, greedy flows bound to their fair share.
SV-CSFQ	(Yes), probabilistic verification of misbehaving flows.

Table 3.4: Scheduling Classification Regarding Fairness

a comparison of the surveyed schedulers with respect to their need for cooperating peers.

3.6 Classification According to the Model's Heterogeneity

Scheduling solution that do not need cooperation among peers use often homogeneous models of peers. All peers are designed to provide the same task. Cooperative peers can cooperate on several ways: providing each other additional information or share tasks. When peers in the network share the load of tasks, specialization can be used, which requires various roles for peers. However such scheduling solutions are rare. A complete

	Scheduling classification regarding cooperativeness
FIFO	Individual solution
RM	Cooperation of peers needed to keep static arrival rates.
EDF	Individual. No cooperation of peers required.
RCSP	Cooperative. End-to-end perf. guarantees aimed.
FQ	Individual: Schedules of peers are independent.
WRR	Round robin approaches do not rely on cooperation.
DRR	Individual. Cooperation between peers is not needed.
BR	Individual. Each peer applies BR without cooperation.
WFQ	Individual. Each peer applies PGPS without cooperation.
SCFQ	FQNP: individual, FQFQ: cooperative for global service classes.
FFQ	Individual. Cooperation of peers is not needed.
SFQ	Individual, each peer maintains the schedule by its own.
WF^2Q	Individual, no cooperation needed.
W^2F^2Q	Individual, no cooperation needed.
VC	Individual, no cooperation needed.
LFVC	Individual, no cooperation needed.
HLS	Individual, no cooperation needed.
WF^2Q+	Individual, no cooperation needed.
H-FSC	Individual, no cooperation needed.
CSFQ	Needs cooperation, one failing node breaks the system.
SV-CSFQ	Individual, but expects correct flow labeling of end hosts.

Table 3.5: Scheduling Classification Regarding Cooperativeness

comparison of the surveyed schedulers considering their model's heterogeneity, can be found in Table 3.6.

3.7 Classification According to the Flow Types

Additional to heterogeneous flow demands, flows can be categorized in various types themselves. For many scheduling mechanism this differentiation is sufficient, still some introduce additional flow categories. Having dynamically lower or higher prioritized flows is sometimes used to decrease the maintenance costs, as only a subset of flows

	Scheduling classification regarding the model's heterogeneity
FIFO	Homogeneous, all peers have follow the same strategy.
RM	Homogeneous model.
EDF	Homogeneous model, all peers behave equally.
RCSP	Homogeneous behavior of all peers.
FQ	Homogeneous. All peers behave equally.
WRR	All peers are modeled homogeneously
DRR	Homogeneous model, all peers behave equally.
BR	Homogeneous model for all peers.
WFQ	Homogeneous model for all peers.
SCFQ	Homogeneous model, all peers are equal.
FFQ	Homogeneous. All peers are modeled equal.
SFQ	All peers are modeled homogeneously equal.
WF^2Q	Homogeneous model.
W^2F^2Q	All peers are modeled homogeneously equal
VC	Homogeneous model: all peers are modeled equal.
LFVC	All peers are modeled homogeneously equal
HLS	Homogeneous model, all peers are equal.
WF^2Q+	All peers are modeled homogeneously equal.
H-FSC	Homogeneous, all peers are equal.
CSFQ	2 roles: edge and core routers.
SV-CSFQ	2 roles: end hosts and core routers.

Table 3.6: Scheduling Classification Regarding the Model's Heterogeneity

has to be considered for actual transmission schedules. Another way of flow types results from flow aggregation. These approaches are considered valuable, however models with only one flow category dominate in research. Our comparison of the surveyed schedulers, regarding the variety of flow types they provide, can be found in Table 3.7.

3.8 Classification According to Use of Framing

Two concepts can be found in literature modeling the usage of time. Classical solutions stick to exact times, which describe when a packet of a flow has to be serviced. This

	Scheduling classification regarding flow types
FIFO	Flat. (All flows have same priority)
RM	Flat.
EDF	Flat.
RCSP	Flat.
FQ	Flat. All flows have equal priority.
WRR	Flat.
DRR	Flat. All flow types are equal.
BR	Flat.
WFQ	Flat.
SCFQ	Flat. The packet classes of FQFQ have no hierarchy
FFQ	Flat.
SFQ	Flat. All flows are equal.
WF^2Q	Flat, no flow aggregation.
W^2F^2Q	Three dynamic flat types of flows: leading, in-sync, lagging.
VC	Flat, no flow types defined.
LFVC	Two flat flow types, low and high priority.
HLS	Hierarchical, organizing various aggregation levels.
WF^2Q+	Flat types organized in hierarchical queues.
H-FSC	Flat, but all flows have own service curves
CSFQ	Flat, flow demands are expressed in packet labels.
SV-CSFQ	Flat, flow demands are expressed in packet labels.

Table 3.7: Scheduling Classification Regarding Flow Types

requires per-flow timing. In contrast to this, some approaches rely on frame-based solutions. Defining frames decreases the complexity, as it has only to be considered whether each flow has been serviced in a time-frame, before moving to the next frame.

The majority of the scheduling mechanisms discussed in this document used exact-time based priority lists, as they provide more fine grained control on the schedule. We present the results of our classification applied to the surveyed schedulers in Table 3.8.

	Scheduling classification regarding use of framing
FIFO	Priority list: all flows have same priority.
RM	Priority list: priorities according arrival rates.
EDF	Priority list: time progresses continuously.
RCSP	Priority list according eligibility times.
FQ	Frame-based: Service all flows in one round.
WRR	Frame-based: Each flow shall receive service in a frame.
DRR	DRR uses the frame-based leaky bucket principle.
BR	Finish-time ordered priority list.
WFQ	Finish-time ordered priority list.
SCFQ	Priority list, no need for frames.
FFQ	Frame-based, frames represent service levels.
SFQ	Priority list, no frames are used.
WF^2Q	Frame-based: Considers only eligible packets.
W^2F^2Q	Priority list, no time framing.
VC	Frames are used to monitor resource consumption of flows.
LFVC	Priority list, framing is not used.
HLS	Priority list, no time framing needed.
WF^2Q+	Priority list, no time framing needed.
H-FSC	Priority list, no framing needed.
CSFQ	Priority list, no frames needed.
SV-CSFQ	Priority list, no frames needed.

Table 3.8: Scheduling Classification Regarding Use of Framing

3.9 Classification According to the Time Modeling

We distinguish between two time modeling approaches: using an external timer and not using any timer. An external timer is often used to determine the arrival times of packets. Some mechanisms compare the real time with the virtual time to get feedback on the quality of their service. This feedback is then taken into account to adapt the service rates. However the second time model refuses to use external timers. These scheduling mechanisms are self-clocked. They calculate all relevant times relative to prior known times. The detailed classification of the surveyed schedulers into this two models can be found in Table 3.9.

	Scheduling classification regarding time modeling
FIFO	No timer needed. Self-clocking.
RM	Self-clocking, but arrival rates has to be known.
EDF	External timer needed to meet the packet deadlines.
RCSP	Timers needed for monitoring.
FQ	Self-clocking: Order of service is independent of time.
WRR	Self-clocking. Time is not relevant in WRR.
DRR	Self-clocking, in DRR only packet sizes matter.
BR	Timer-based. Needed to calculate finish-time.
WFQ	Timer-based. Needed to calculate finish-time.
SCFQ	Self-clocking, measurements based on observations.
FFQ	Timer-based, needed for finish time calculations.
SFQ	Self-clocked, all times are calculated relative to known times.
WF^2Q	Timer needed to get arrival times.
W^2F^2Q	Timer-based, also needed to detect lagging.
VC	Self-clocking, VC calculated using arrival/departure rates.
LFVC	Timer is needed to meet deadlines.
HLS	Self-clocking.
WF^2Q+	Self-clocking, using only relative times.
H-FSC	Timers are needed for the service curve calculations.
CSFQ	Self-clocking, timing is less important.
SV-CSFQ	Self-clocking, timing is less important.

Table 3.9: Scheduling Classification Regarding Time Modeling

3.10 Classification According to Service Rate Changes

This classification provides only two possible states: static or dynamic. In contrast to the adaptability of the system, we focus in this case only on changes in the service rate of a flow that did not change its service demands. Whether the service rate varies under static service demand or not, is expressed by this classification. The majority of scheduling algorithms bound the service rate to a value near its guaranteed rate. Still some scheduling mechanisms exist that vary the service rate provided to a flow. The exact mapping of schedulers to these two states can be found in Table 3.10.

	Scheduling classification regarding service rate changes
FIFO	Dynamic (best effort).
RM	Static. Arrival rates are considered to be static.
EDF	Changes in the arrival rate lead to changes in the service rate.
RCSP	Static, bounded to resource reservations.
FQ	Static. Share of bandwidth per flow is fixed.
WRR	The service rate is bound to the static weight of a flow.
DRR	Static. All flows receive equal service.
BR	Static service rate.
WFQ	Static service rate.
SCFQ	Static: WFQ provides fair equal share of bandwidth per-flow.
FFQ	Static, all flows get same service per frame.
SFQ	Static, each flow receives same amount of bandwidth.
WF^2Q	Static, the weights of the flows are fixed.
W^2F^2Q	Yes, service rates vary to compensate lagging.
VC	Dynamic, service can vary, only average is important.
LFVC	Dynamic, depends on dyn. ranking of the flow: low or high.
HLS	Dynamic, to compensate too much/less service.
WF^2Q+	Static, due to tight fairness bounds.
H-FSC	Static, flows keep their service curves.
CSFQ	Dynamically changed to find optimal utilization.
SV-CSFQ	Dynamically changed to find optimal utilization.

Table 3.10: Scheduling Classification Regarding Service Rate Changes

3.11 Classification According to Changing Packet Priorities

Almost any scheduling mechanism provides for the same packet, in every peer it passes, the same priority. Packet priorities are often bound to flows. If a flow is consuming to muss or to less share, than upcoming packets receive a compensation amount of service. However, some scheduling mechanisms enforce the compensation not only in upcoming packets, but also in current packets passing them. This approach often requires cooperation of other peers as well. In conclusion we present our classification results related to packet priorities in Table 3.11.

	Scheduling classification regarding changing packet priorities
FIFO	Static (all equal)
RM	Dynamic, other peers may have higher prioritized flows.
EDF	Dynamic, packet priorities depend on the rates of other flows.
RCSP	Static, reservations are flow-related.
FQ	Static. All packets have the same priority.
WRR	Depends on changes of the weight of a flow from peer to peer.
DRR	Static. Priority of a packet is related to its size.
BR	Static. Packets have on each peer the same priority.
WFQ	Static. Packets have on each peer the same priority.
SCFQ	Static, as for each packet its size is fixed.
FFQ	Static, packets priorities do not change.
SFQ	Static, packet priorities do not change.
WF^2Q	Static, related to arrival rate and packet size.
W^2F^2Q	Static, packet priorities are defined by their length.
VC	Static, packets have same priority in every peer.
LFVC	Static, packets keep their priorities.
HLS	Not applicable.
WF^2Q+	Static, packet priorities result from their size.
H-FSC	Static, flows keep their service curves.
CSFQ	Dynamic, packet prio. are matched to flow behavior.
SV-CSFQ	Dynamic, packet priorities are matched to flow behavior.

Table 3.11: Scheduling Classification Regarding Changing Packet Priorities

3.12 Classification according to Start and Finish-Time Relevance

We identified two main concepts that are used to determine the processing order of a packet in a flow. The first approach uses the start-time (which is often the arrival time) of a packet to determine its place in the schedule. The other approach is inspired by the optimal fluid model, which is also used to calculate a virtual finish-time. Some scheduling mechanisms use this virtual finish-time to define at which time a packet has to be processed. The exact classification of the survey schedulers in terms of whether they use

the start-time or the finish-time can be found in Table 3.12.

	Scheduling classification regarding start and finish-time relevance
FIFO	Arriving time determines processing order.
RM	The arrival/start time of a packet defines its order.
EDF	Finish-time based. EDF-principle.
RCSP	Start-time based. Order by lowest eligible start time.
FQ	Start-time-based order in the flow queue.
WRR	Start-time-based. WRR uses per-flow FIFO.
DRR	Finish-time, defined by the size of the packet.
BR	Finish-time based. Packet sizes are considered.
WFQ	Finish-time based. Packet sizes are considered.
SCFQ	Finish-time-based. The packet size is important.
FFQ	Service all flows with finish-time in current frame.
SFQ	Start-time-based.
WF^2Q	Both needed to define the processing time of a packet.
W^2F^2Q	Both, using the same principle (SEFF) as in WF^2Q.
VC	Finish-time-based.
LFVC	Finish-time-based: deadlines matter
HLS	Not applicable.
WF^2Q+	Both, using the SEFF policy
H-FSC	Uses start time to determine processing order.
CSFQ	Start-time-based order of packets their flow's queue.
SV-CSFQ	Start-time-based order of packets in their flow's queue.

Table 3.12: Scheduling Classification Regarding Start and Finish-Time Relevance

3.13 Classification According to the Independency of Flows

We present in Table 3.13 a classification with two types of solutions, considering the influence of flows on each other. The first class enables flows to have effect on the service provision for other flows, the second class permits this influence. While some older mechanisms did not provide firewalls between flows, this resulted in the ability of flows to consume more service at the cost of other flows. However in scheduling mechanisms

discussed lately, inter-dependency is not considered harmful, even more it is used as a tool to compensate and punishment for too much service.

	Scheduling classification regarding the independency of flows
FIFO	Flows have direct effect on each other
RM	Inter-dependent flows, one may consume all bandwidth.
EDF	All flow deadlines are met independently, if valid schedule exist.
RCSP	Independent flows bound to their reservations.
FQ	Independent flows. Flow queue length has no effect.
WRR	Independent flows. Greedy flows have no effect.
DRR	Independent flows. Bursty flows have no effect.
BR	Independent flows. Fair share per flow does not change.
WFQ	Independent flows. Fair share per flow does not change.
SCFQ	Inter-dependent. Effects in FQFQ limited to own service class.
FFQ	Independent flows. But new flows are preferred.
SFQ	Independent flows, have no effect on each other.
WF^2Q	Independent flows.
W^2F^2Q	Independent flows.
VC	Inter-dependent flows.
LFVC	Inter-dependent flows (but only benefiting)
HLS	Inter-dependent flows, effect each others share.
WF^2Q+	Independent flows.
H-FSC	Inter-dependent flows, flows may receive excess bandwidth.
CSFQ	Inter-dependency of flows for calculating optimal share.
SV-CSFQ	Inter-dependency of flows for calculating optimal share.

Table 3.13: Scheduling Classification Regarding the Independency of Flows

3.14 Classification According to Service Saving Capabilities

Some scheduling mechanisms enabled the flows to save service time that they can use later to get more service at a time. This concept supports burstiness. All mechanisms proposed show limited service saving capabilities. Still some schedulers do not allow service saving, they only allow to use service immediately. Having no service saving

memory lessens the complexity of the system. We present in Table 3.14 the classification of the surveyed schedulers according to their service saving capabilities.

	Scheduling classification regarding service saving capabilities
FIFO	No service history is maintained
RM	No service history.
EDF	No service history is maintained, no service can be saved.
RCSP	Limited service history is maintained to eliminate jitter.
FQ	No history maintained.
WRR	No service history is maintained
DRR	Limited service saving, bound by max. packet length.
BR	Limited service saving, bound by maximum packet length.
WFQ	Limited service saving, bound by max. packet length.
SCFQ	Service saving limited to maximum packet length.
FFQ	Saving service is limited by the frame length.
SFQ	Service saving limited by maximum packet length.
WF^2Q	Service saving bound by maximum packet length.
W^2F^2Q	Service saving: bound by maximum packet length and lagging.
VC	Service can be saved for a whole monitoring period.
LFVC	Service saving, bound by avg. service provided by system.
HLS	Service saving is limited.
WF^2Q+	Service saving is bounded by maximum packet length.
H-FSC	No service saving, as excess bandwidth is shared fair.
CSFQ	No service history is maintained.
SV-CSFQ	No service history is maintained.

Table 3.14: Scheduling Classification Regarding Service Saving Capabilities

3.15 Classification According to Service Demands

We identified various approaches to model the service demands of a flow, the results are presented in Table 3.15. While some schedulers model only throughput requirements, some introduce additional delay requirements. Some rare schedulers even provide models for multi-dimensional demands, like decoupled delay and bandwidth needs. However

with increasing complexity of demands, the system complexity grows as well, as these demands have to be met in different ways.

	Scheduling classification regarding service demands
FIFO	Zero-dimensional, no deadlines are considered.
RM	Single-dimensional: Arrival rate.
EDF	Single-dimensional demand: deadline.
RCSP	Single-dimensional demand defined by deadlines.
FQ	Zero-dimensional: Packets have no deadline
WRR	Single-dimensional demand: weight of a flow.
DRR	DRR: 1-dim. demand for throughput. DRR+: 2-dim.
BR	Single-dimensional demand for fair throughput.
WFQ	Single-dimensional demand for fair throughput.
SCFQ	Single-dimensional: Only throughput matters.
FFQ	Single-dimensional, only fair throughput.
SFQ	Single-dim.: Throughput demand defined by arrival rate.
WF^2Q	Two-dimensional: jitter and throughput.
W^2F^2Q	Two-dimensional: jitter and throughput.
VC	Single-dimensional, only average throughput matters.
LFVC	Two-dimensional: throughput and delay.
HLS	Two-dimensional: throughput and latency
WF^2Q+	Two-dimensional: delay and throughput
H-FSC	Multi-dimensional, depends on service curve model.
CSFQ	Single-dimensional: packet priorities/flow demands.
SV-CSFQ	Single-dimensional: packet priorities/flow demands.

Table 3.15: Scheduling Classification Regarding Service Demands

Chapter 4

Conclusion

Peer-to-peer systems aim to overcome the constraints and drawbacks of client/server based systems. End-user devices participate in the network and build an overlay structure to provide the functionality of a client/server based system. The scalability of the system is limited by the available bandwidth in the system. There are no restrictions for devices of any type not to participate in peer-to-peer networks. As a result the diversity in the availability of resources in the network is large. However, we focused in this document on the limited bandwidth in the system. Nowadays ADSL connections dominate the connection of end users to the Internet[1]. ADSL connections provide asymmetric up-link and down-link bandwidth. User devices can download faster than they can upload. With these circumstances one can assume that the processing queue of a peer with low bandwidth capabilities is often filled. Delay-critical request are enqueued in each peer on their route and delayed although they should be processed as fast as possible.

Scheduling mechanisms provide a tool to reorder the processing schedule of packets in the queue. Schedulers pick the next element out of the queue that has to be processed, in this case delay-critical packets should be preferred.

In this document we presented a survey on popular scheduling mechanisms discussed for the network layer. We derived a set of classification aspects that we used to build a taxonomy on the surveyed scheduling mechanisms. The taxonomy can be used to derive requirements for message schedulers in P2P systems in different scenarios.

The taxonomy further gives us details on mechanisms that are needed to build a fair and adaptable scheduler. We can learn how role-based scheduling can be used to distribute the scheduling load in the system and to decrease the complexity. However, a homogeneous modeling of the schedulers has advantages as well, because misbehavior-critical cooperation can be avoided. Introducing flow types can be used to map the var-

[1] See http://en.wikipedia.org/wiki/DSL_around_the_world

ious services in a P2P system. A scheduler for P2P systems should be self-clocked as exact timing is not available on some devices that participate in the overlay. The service rates that are provided for a flow should be elastic, so that more important flows can be preferred when the system considers this as necessary. Whether the system is start-time-based or finish-time-based is not that important; important are the consequences resulting from this. Start-time based approaches tend to propagate delays, finish-time based take packet sizes into account. Another point that has effect on the fairness of the system is the independency of flows. The system should control that greedy flows do not benefit at the cost of other flows. However, it may be necessary to prioritize a flow and have an inter-flow influence in emergency situations. Finally the survey and classification shows us that multi-dimensional service demands can be met at the cost of higher complexity. It has to be evaluated whether peers in the system are sufficiently powerful to fulfill these requirements.

The optimal scheduler for several P2P scenarios has still to be found and evaluated. This is part of our future work. In the future the author of this document will implement scheduling mechanisms in PeerfactSim.KOM[2][KKH+06]. With limited bandwidth in the system we expect functionality problems of common overlays. The authors aim to show the benefits of the usage of scheduling mechanisms and active queue management in P2P systems.

[2]http://www.peerfactsim.com

Bibliography

[BSBKM07] Dirk Bradler, Julian Schröder-Bernhardi, Jussi Kangasharju, and Max Mühlhäuser. Peer-to-Peer Communications for Disaster Management. *IEEE GLOBECOM 07: Global Communications Conference, Exhibition & Industry Forum 2007*, 26 November 2007. under submission.

[BZ97] Jon C. R. Bennett and Hui Zhang. Hierarchical packet fair queueing algorithms. *IEEE/ACM Transactions on Networking*, 5(5):675–689, 1997.

[CK88] T. L. Casavant and J. G. Kuhl. A taxonomy of scheduling in general-purpose distributed computing systems. *IEEE Trans. Softw. Eng.*, 14(2):141–154, 1988.

[DH90] James R. Davin and Andrew T. Heybey. A simulation study of fair queueing and policy enforcement. *SIGCOMM Comput. Commun. Rev.*, 20(5):23–29, 1990.

[DKS89] A. Demers, S. Keshav, and S. Shenker. Analysis and simulation of a fair queueing algorithm. In *SIGCOMM '89: Symposium proceedings on Communications architectures & protocols*, pages 1–12, New York, NY, USA, 1989. ACM Press.

[FJ95] Sally Floyd and Van Jacobson. Link-sharing and resource management models for packet networks. *IEEE/ACM Trans. Netw.*, 3(4):365–386, 1995.

[gol94] *A self-clocked fair queueing scheme for broadband applications*, volume 2, 12 June 1994.

[GVC96] Pawan Goyal, Harrick M. Vin, and Haichen Chen. Start-time fair queueing: a scheduling algorithm for integrated services packet switching networks. *SIGCOMM Comput. Commun. Rev.*, 26(4):157–168, 1996.

[INF02] INFOCOM'02, editor. *Self-verifying CSFQ*, New York, May 2002.

[J. 87] J. Nagle. On Packet Switches with Infinite Storage. *IEEE Trans. Comm vol. 35, No.4*, April 1987.

[J.C96] J.C.R. Bennett and H. Zhang. WF2Q: Worst-case fair weighted fair queueing. pages 120–128, San Francisco, CA, March 1996. IEEE.

[Kat91] Katevenis M. and Sidiropoulos C. and Courcoubetis C. Weighted round-robin cell multiplexing in a general-purpose ATM switch chip. *IEEE Journal on Selected Areas in Communications 1991*, 9(8):1265–1279, 1991.

[KKH+06] Aleksandra Kovacevic, Sebastian Kaune, Hans Heckel, André Mink, Kalman Graffi, Oliver Heckmann, and Ralf Steinmetz. PeerfactSim.KOM - A Simulator for Large-Scale Peer-to-Peer Networks. Technical Report Tr-2006-06, Technische Universität Darmstadt, Germany, 2006.

[LL73] C. L. Liu and James W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the ACM*, 20(1):46–61, 1973.

[PG93] Abhay K. Parekh and Robert G. Gallager. A generalized processor sharing approach to flow control in integrated services networks: the single-node case. *IEEE/ACM Trans. Netw.*, 1(3):344–357, 1993.

[SSZ98] Ion Stoica, Scott Shenker, and Hui Zhang. Core-stateless fair queueing: achieving approximately fair bandwidth allocations in high speed networks. volume 28, pages 118–130, New York, NY, USA, 1998. ACM Press.

[SV95] M. Shreedhar and George Varghese. Efficient fair queueing using deficit round robin. In *SIGCOMM*, pages 231–242, 1995.

[SV96] Dimitrios Stiliadis and Anujan Varma. Design and analysis of frame-based fair queueing: a new traffic scheduling algorithm for packet-switched networks. In *SIGMETRICS '96: Proceedings of the 1996 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 104–115, New York, NY, USA, 1996. ACM Press.

[SVC97] Subhash Suri, George Varghese, and Girish Chandranmenon. Leap forward virtual clock: a new fair queuing scheme with guaranteed delays and throughput fairness. In *PODC '97: Proceedings of the sixteenth annual ACM symposium on Principles of distributed computing*, page 281, New York, NY, USA, 1997. ACM Press.

[SZN97] Ion Stoica, Hui Zhang, and T. S. Eugene Ng. A hierarchical fair service curve algorithm for link-sharing, real-time and priority services. In *SIGCOMM '97: Proceedings of the ACM SIGCOMM '97 conference on Applications, technologies, architectures, and protocols for computer communication*, pages 249–262, New York, NY, USA, 1997. ACM Press.

[YSK+00] Yung Yi, Yongho Seok, Taekyoung Kwon, Yanghee Choi, and Junseok Park. W2f2q: packet fair queuing in wireless packet networks. In *WOWMOM '00: Proceedings of the 3rd ACM international workshop on Wireless mobile multimedia*, pages 2–10, New York, NY, USA, 2000. ACM Press.

[ZF94] H. Zhang and D. Ferrari. Rate-controlled service disciplines, 1994.

[Zha91] Lixia Zhang. Virtualclock: a new traffic control algorithm for packet-switched networks. *ACM Trans. Comput. Syst.*, 9(2):101–124, 1991.