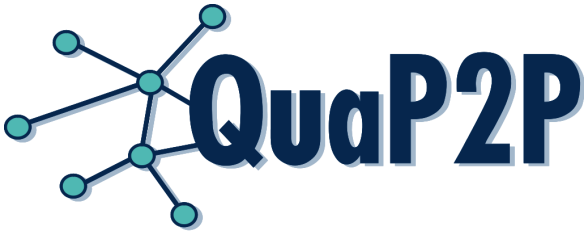

Technische Universität Darmstadt

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

Taxonomy of Active Queue Management Strategies in Context of Peer-to-Peer Scenarios

Technical Report KOM-TR-2007-01

Submitted by

Kálmán Graffi, Konstantin Pussep, Nicolas Liebau, Ralf Steinmetz

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

First published: 01. December 2006
Last revision: 29. March 2007

For the most recent version of this report see
ftp://ftp.kom.tu-darmstadt.de/pub/TR/KOM-TR-2007-01.pdf

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

Contents

List of Tables

Nomenclature

ADSL Asymmetric Digital Subscriber Lines

AIMD Additive Increase Multiplicative Decrease

ARED Adaptive Random Early Detection

ATM Asynchronous Transfer Mode

AVQ Adaptive Virtual Queue

BLUE BLUE Active Queue Management Algorithm

CHOKe Choose and Keep Packets from Responsive Flows

CHOKe Choose and Kill Packets from Unresponsive Flows

E-RED Exponential Random Early Detection

ECN Explicit Congestion Notification

EPD Early Packet Detection

ERD Early Random Drop

FBA Fair Buffer Allocation

FRED Fair Random Early Detection

PI Proportional Integral (Controller)

PIP Proportional Integral Controller with Position Feedback Compenstation

PPD Partial Packet Discard

RED Random Early Detection

RED-PD Random Early Detection with Preferential Dropping

REM Random Exponential Marking

RTT Round Trip Time

SFB Stochastic Fair BLUE

SPD Selective Packet Dropping

SRED Stabilized Random Early Detection

Chapter 1

Active Queue Management in P2P: Motivation and Classification

Peer-to-peer (P2P) principles are evolving in the Internet, due to their self-organization capabilities. P2P systems have no single point of failure, which would compromise their scalability. However avoiding single points of failure in the system comes with the need of a system-wide self-organization of the peers. However self-organization of the peers need 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 of resources solves this problem.

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 boundaries on the technical level, crucial services have to be available and perform in an acceptable quality. These circumstances require efficient utilization of scarce bandwidth in each peer and scarce resources in the overlay.

1.1 Message Scheduling in Peer-to-Peer

Nowadays bandwidth is the most scarce resource in networks. In P2P networks available bandwidth for services is even lower, as the maintenance of P2P requires a part of the bandwidth and in addition to this, asymmetric bandwidth capabilities are common. Asymmetric Digital Subscriber Lines (ADSL) connections are dominant in the Internet[1]. This causes that asymmetric bandwidth availability, differing down-link and up-link capabilities, have to be assumed as normal. Typically the out link provides less bandwidth,

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

than the incoming link. However this results that it may occur, that not all incoming data can be processed and transmitted, as the out bandwidth is less than the in bandwidth.

Congestion is a problem that may occur, when no suitable mechanisms are used.

Common reliable protocols on the transport layer (like TCP) support mechanisms to adapt the sending rate to characteristics of the network. The sending rate is decreased using the congestion window principle, when packet loss is detected. Routers in the network can drop (or mark) packets intentionally in order to signal to the flow sources to choke / decrease their sending rate. Additionally a fair utilization of the bandwidth can be enforced by degrading the service for greedy flows. Active Queue Management (AQM) aims to detect congestion in the network before it becomes severe by overfilling the router queue. It means that the router tries to reduce the sending rate of the traffic sources by dropping or marking packets. In Figure 1.1 we show the principle task of AQM. There exist two approaches to indicate congestion: Packets can be dropped and packets can be marked. First strategy requires cooperation of the endpoints and latter generates additional overhead through re-sending. Endpoints have to react on marked packets as they have been dropped and decrease their throughput. With this the same improvement of bandwidth utilization can be achieved, but without additional overhead costs. Additionally some AQM mechanisms aim to reduce the bandwidth of greedy flows by dropping their packets at higher rates.

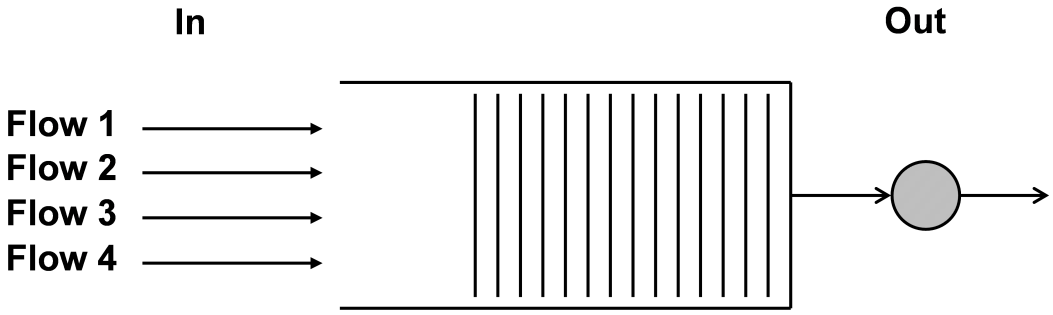

Figure 1.1: This figure shows the main problem solved by AQM: Which packets should be dropped if the queue is at risk to become congested.

1.2 Classification Classes related to AQM Mechanisms

There are more than 50 proposals for Active Queue Management schemes [BRHG]. In order to classify even a subset of those proposals we identified the most important properties which are described in the following subsections. With the classification points described in this section, we analyze the Active Queue Management mechanisms in Chapter 2.

1.2.1 Main goal

Each scheme was designed to improve some particular properties of the system. As there are often trade-offs between some goals an AQM approach cannot satisfy all requirements at the same time. So some schemes focus on small loss while other concentrate on high stability or fast responsiveness.

1.2.2 Approach of solution

The proposed AQM mechanism use various underlying models to make decisions. Some rely on heuristics, others work deterministically and some even use ideas and theories from related research topics. This classification points out the theory behind the proposed AQM mechanism.

1.2.3 Congestion measure

In general AQM schemes aim to prevent congestion by detecting them and notifying the traffic sources, i.e some of the observed parameters are monitored to adjust the routers behavior. Typical congestion measures for AQM are:

- current queue size (i.e. the current utilization of the buffer queue, not to confuse with the queue capacity),
- average queue size, which is computed or estimated from the current and previous queue sizes
- number of active flows, which can be observed by comparing and storing source/target pairs from packet headers
- packet arriving rate, i.e. how many packets per time unit are arriving

1.2.4 Fairness

In case of congestion an unbalanced marking strategy will lead to a unfair allocation of the bandwidth to different flows. For instance if a router starts to mark all incoming packets with the same probability the flows with small bandwidth consumption will be discriminated compared to flows which use more than their fair share. Even if an AQM scheme does not provide an explicit handling for high-bandwidth flows it can still reduce their bandwidth consumption indirectly.

1.2.5 Awareness of malicious or unresponsive flows

Some flows may ignore the congestion notified by a router, either because of their unresponsive nature like for UDP video or audio flows or intentionally to receive larger than the share of the bandwidth. An AQM algorithm can handle such schemes in different ways. The algorithm can ignore such flows or detect and punish them.

1.2.6 Target quality: static vs. dynamic

A target quality is used in some AQM mechanisms to converge the performance of the system to some distinct level. For those schemes which have a target value, like a target queue size, it can be either fixed or adjusted according to the system load. So we further distinguish between static and dynamic quality targets.

1.2.7 State to maintain

Typically an AQM algorithm needs to keep some information like the observed traffic density or unresponsive flows. Some schemes require only very small number of values to be maintained. Other requires more state to be maintained.

1.2.8 Number of required parameters

Different approaches require different number of parameters which must be set to ensure proper function in a specific scenario or topology. These parameters must usually be either hard-coded or can be set by an administrator. Often it is not obvious how to set them appropriately.

1.2.9 Special characteristic

This point is not meant to classify the approaches, it is meant to point out remarkable characteristics of the system, that are special. Under this point we list properties of the solution rarely found in other solutions.

Chapter 2

Survey on Active Queue Management Mechanisms

Active Queue Management (AQM) aims to detect congestion in the network before it becomes severe by overfilling the router queue. It means that the router tries to reduce the sending rate of the traffic sources by dropping or marking packets. There exist two approaches to indicate congestion: Packets can be dropped and packets can be marked. First strategy requires cooperation of the endpoints and latter generates additional overhead through re-sending. Endpoints have to react on marked packets as they have been dropped and decrease their throughput. With this the same improvement of bandwidth utilization can be achieved, but without additional overhead costs. Additionally some AQM mechanisms aim to reduce the bandwidth of greedy flows by dropping their packets at higher rates. In this chapter we give a survey on Active Queue Management algorithms, that are suitable for Peer-to-Peer networks. In addition to the survey we present a detailed flat taxonomy.

2.1 ECN: Explicit Congestion Notification (passive solution)

The Network Working Group proposed in a RFC [RFB01] the addition of Explicit Congestion Notification to IP. Nowadays in networks congestion is detected on transport layer (e.g. by TCP) upon the loss of packets. The authors argue that congestion should be detected before buffers overflow and packets have to be dropped. They propose to use two reserved bits in the IP header for signaling congestion purposes. The following two issues has to be solved by the solution.

- Non-ECN aware routers in the system shall be migrated to ECN aware routers, both

type of routers shall cooperate.

- Existing mechanisms like packet dropping or scheduling mechanisms shall remain applicable without interference.

The authors propose to use two reserved bits (6 and 7) in the IP header to indicate the congestion status of the flow. Following assignment is presented:

- 00 - The packet is not using ECN.
- 10 or 01 - The packet is using ECN, but no congestion is monitored.
- 11 - The packet is using ECN and is indicating congestion.

Upon reception of a packet marked the the ECN-code 11, the receiving endpoint reacts in the same way, as if the packet would have been dropped on its way: it halves the congestion windows in TCP. In comparison to dropping, using this mechanism prevents the loss of data as sending nodes are notified in an early stage before congestion can occur. A classification according to Chapter 1.2 of ECN is presented in Table 2.1.

AQM classification	Explicit Congestion Notification
Goal of algorithm	Report congestion
Solution approach	Not discussed
Congestion detection	Not discussed
Fair bandwidth alloc.	No, all flows are treated equal.
Malicious-aware	No, as not fair.
Target quality interval	$Q_{current} \leq Q_{max}$, where Q_{max} is static.
Required state	Non, just report on detection.
# of predef. params.	1
What has effect on P_{drop}	$Q_{current}$, Q_{max}
Special characteristic	Dropping is avoided if possible.

Table 2.1: ECN in context of AQM classification

2.2 DT: Drop Tail (active solution)

The simplest way to handle a router congestion is usually called Drop Tail. It means that new packets are enqueued as long as there is place for them in the queue. If the queue is full because the send rate of the output link is smaller than the arrival rate at the input

link, all new packets are dropped. Figure 2.1 shows how DT is used in an AQM regulated system. A classification according to Chapter 1.2 of DT is presented in Table 2.2.

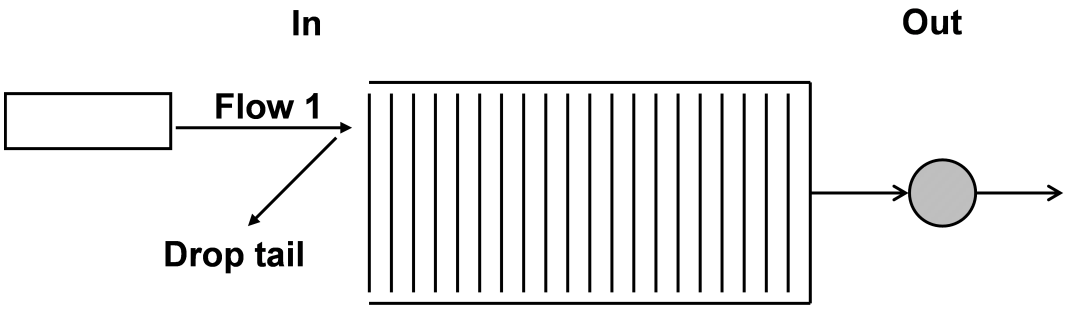

Figure 2.1: This figure shows the main principle of Drop Tail: Upon arrival of a new packet in a system with a full queue, the packet is dropped.

AQM classification	Drop Tail
Goal of algorithm	Solve congestion problem
Solution approach	Deterministic
Congestion detection	$Q_{current} \geq Q_{max}$
Fair bandwidth alloc.	No, all flows are treated equal.
Malicious-aware	No, as not fair.
Target quality interval	$Q_{current} \leq Q_{max}$, where Q_{max} is static.
Required state	Non, just drop new packets on congestion.
# of predef. params.	1
What has effect on P_{drop}	$Q_{current}, Q_{max}$
Special characteristic	Most intuitive strategy to cope with congestion.

Table 2.2: DT in context of AQM classification

2.3 RED: Random Early Detection

Sally Floyd and Van Jacobson present in [FJ93] a mechanism called Random Early Detection (RED) that aims congestion avoidance. Their work is motivated by the goal to keep average queue sizes in routers small. This is done by dropping or marking some packets that have a position in the queue exceeding a certain threshold. For marking packets ECN can be used, indicating congestion on the route.

System-wide parameters Q_{min} and Q_{max} define the threshold boundaries of the queue size. Upon arrival of a new packet in the system the average queue size Q_{avg} of the flow is calculated and compared to Q_{min} and Q_{max}. Q_{avg} is updated each time a packet arrives with following formula: $Q_{avg} = (1 - W_q) \cdot Q_{avg} + W_q \cdot Q_{curr}$, where W_q is the weight of the former average queue size and Q_{curr} the current queue size.

When Q_{avg} exceeds Q_{max} the packet is marked. In the case that Q_{avg} is within the boundaries of Q_{min} and Q_{max}, the marking probability P_m is calculated using following equation: $P_m = \frac{P_{avg}}{1-count \cdot P_{avg}}$, where $count$ is the number of packets since the last marked packet and P_{avg} is defined as followed: $P_{avg} = P_m^{max} \frac{Q_{avg}-Q_{min}}{Q_{max}-Q_{min}}$. The packet is marked with a probability of P_m, in this case $count$ is reset. If the packet is not marked, $count$ is incremented.

With this mechanism the average queue size can be controlled and congestion can be avoided. With the parameter W_q additionally the burst-awareness of the mechanism can be modeled. Q_{min} and Q_{max} define the expected range of queue length, Q_{min} defines the minimum queue length at which no packets are dropped, as Q_{avg} exceeds Q_{min} the dropping probability increases with increasing Q_{avg} and $count$, up to the maximum dropping probability P_{avg}^{max}. These parameters can be configured to suit to different environments. Fairness is provided based on the assumption that the number of dropped packets correlates with the utilization of the bandwidth by the specific flow. A classification according to Chapter 1.2 of RED is presented in Table 2.3.

AQM classification	Random Early Detection
Goal of algorithm	Provide performance guarantees (delay, throughput)
Solution approach	Heuristic
Congestion detection	Q_{avg}, weighted average with burst-awareness
Fair bandwidth alloc.	No, all flows have same dropping probability.
Malicious-aware	No, as not fair.
Target quality interval	$Q_{min} \leq Q_{current} \leq Q_{max}$, where Q_{min} and Q_{max} are static.
Required state	$O(1)$, maintaining Q_{avg}, but no history.
# of predef. params.	4
What has effect on P_{drop}	$Q_{current}$, Q_{min}, Q_{max}, P_m^{max}, W_q.
Special characteristic	Reference AQM algorithm. Burst-awareness adjustable.

Table 2.3: RED in context of AQM classification

2.4 ATM-RED: Random Early Detection for ATM

Random Early Detection for Asynchronous Transfer Mode (ATM) networks was proposed by Rosolen, Bonaventure and Leduc in [RBL99]. They claim that for efficiency reasons in ATM networks which support TCP, not only cells should be dropped, but whole packets, as single cells (48 bytes) are not retransmitted. The whole packet (approximately 1500 bytes) is anyway detected as corrupted. In Figure 2.2 we show the problem solved by ATM-RED. The authors present several strategies how to take the characteristics of ATM

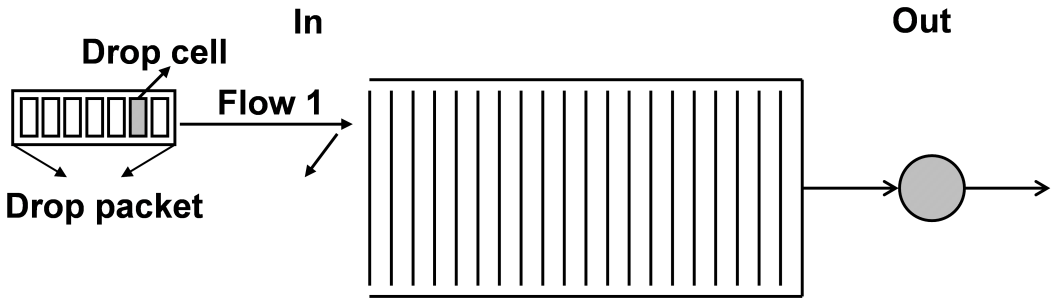

Figure 2.2: This figure shows shows the main problem solved by ATM-RED: In ATM networks a dropped cell causes the dropping of the whole packet, so from beginning on whole packets should be dropped.

networks (small cells) into account when using RED. All these strategies aim to increase the network throughput of TCP traffic, utilize the bandwidth efficiently and be fair to all TCP flows.

Following strategies are presented:

1. Partial Packet Discard (PPD): This is the simple strategy where a the cell at the tail of the queue is dropped not considering to which packet it belongs. However, the packet to which the cell belongs is then incomplete and will be discarded and retransmitted.

2. Early Packet Detection (EPD): This strategy monitors the length of the queue. When the length exceeds a specific threshold, whole packets are dropped instead of cells only. This increases the performance as less packets have to be retransmitted in total. Implementation requires that corresponding cells can be identified.

3. Selective Packet Dropping (SPD): SPD is an extension to EPD that aims to improve fairness. Cells corresponding to a packet are still dropped in total, but in contrast to

EPD in SPD it is necessary that the flow corresponding to the packet to be dropped occupies a large part of the queue.

4. Fair Buffer Allocation (FBA): FBA uses instead of thresholds as criteria for dropping packets a rejection function. For all incoming cells the normalized share occupied by the corresponding flow is calculated. This normalized flow share is compared to limit function of the current buffer occupancy. Only when the normalized share exceeds the limit function, packets can be discarded.

5. RED adapted to ATM (ATM-RED): ATM-RED takes into account that packet sizes are not known in advance, so cell dropping probabilities (P_C) are used. The probability of dropping a packet P_P is then $P_P = 1 - (1 - P_C)^n \approx n \cdot P_C \ \ if P_C \ll 1$, where n is the number of cells in a packet. With increasing size of the packet, the probability to be dropped increases as well.

The quality of the strategies presented in this list improve in the order they are listed. EPD leads to less packet drops in compare to PPD. SPD is more fair than EPD. FBA is more flexible than SPD and finally ATM-RED keeps the average buffer occupancy at a lower state. A classification according to Chapter 1.2 of ATM-RED is presented in Table 2.4.

AQM classification	Random Early Detection for ATM
Goal of algorithm	RED optimized for cell-based architecture in ATM networks.
Solution approach	Heuristic
Congestion detection	Q_{avg}, weighted average with burst-awareness
Fair bandwidth alloc.	Depends on strategy, ranging from unfair to fair.
Malicious-aware	No, P_{drop} increases linearly with bandwidth utilization.
Target quality interval	Depends on strategy, from no target to dynamic target.
Required state	Ranges from $O(1)$ to $O(number\ of\ flows)$.
# of predef. params.	4
What has effect on P_{drop}	$Q_{current}$, Q_{min}, Q_{max}, P_m^{max}, W_q.
Special characteristic	Takes the characteristics of ATM networks into account.

Table 2.4: ATM-RED in context of AQM classification

2.5 ARED: Adaptive Random Early Detection

Floyd, Gummadi and Shenker present in [FGS01] a slight modification of RED, called Adaptive RED, that provides in face of congestion average pre-defined delay to the flows.

The strategy Adaptive RED applies is adapting the parameters of RED to the current situation. The parameters are Q_{min} and Q_{max} being lower and upper threshold of expected queue length, W_q is the weight of the current queue length for the calculation of the average queue length. Finally P_m^{max} is the maximum probability for marking packets that a system can achieve.

The authors argue that high link utilization, requiring large buffers, and low transfer delays, requiring small buffers, are concurrent aims and a trade-off has to be found. RED randomly drops packets with probability related to the current average queue size. Only congestion and the parameter setting have effect on the balance between link utilization and low delays.

ARED adapts the parameters to reduce the packet loss rate and the variance of the queue size. The main goal is to improve the average queuing delay (anti-proportional to Q_{avg}) by adapting P_m^{max} periodically. According the Additive Increase Multiplicative Decrease (AIMD) principle P_m^{max} is increased by an amount α_i if Q_{avg} is greater than a target queue interval Q_{target} and P_m^{max} is less or equal 0.5. In the case that Q_{avg} is smaller than the lower bound of Q_{target} and P_m^{max} is greater or equal 0.01, P_m^{max} is set to a fraction β_i of its old value.

The authors suggest to perform this update every 0.5 second, with a target queue size of about $Q_{min} + \frac{Q_{max}-Q_{min}}{2}$, α_i is suggested to be $\alpha_i = min(0.01, \frac{P_m^{max}}{4}$ and $\beta_i = 0.9$. A classification according to Chapter 1.2 of ARED is presented in Table 2.5.

AQM classification	Adaptive Random Early Detection
Goal of algorithm	Adaptive trade-off between link utilization and delay
Solution approach	Heuristic
Congestion detection	Q_{avg}, weighted average with burst-awareness
Fair bandwidth alloc.	No
Malicious-aware	No, as not fair.
Target quality interval	Static target interval Q_{target}
Required state	$O(1)$
# of predef. params.	3
What has effect on P_{drop}	$Q_{current}$, Q_{min}, Q_{max}, α_i, β_i, Q_{target}, P_m^{max}
Special characteristic	Adapt Q_{target} to meet delay and throughput requirements.

Table 2.5: ARED in context of AQM classification

2.6 SRED: Stabilized Random Early Detection

In contrast to normal RED which focus on estimating the average queue size, in SRED, introduced by Ott et al. in [OLW99], the most important value is the estimation of the number of active flows. This is done by keeping a list of so called *zombies* - a list of size M of flows that were recently active. Each *zombie* carries the information about which flow it is corresponding to currently and a counter, which is initialized to 0 at the beginning. The *zombie* list may contain more than one entry per flow. The counter values of different entries are independent, even if the *zombies* are representing the same flow.

On packet arrival the new packet is compared with a randomly chosen entry from the list. If they both belong to the same flow, the counter of the entry is increased. If the new packet and the *zombie* do not correspond, with a probability of P_{over} the flow identifier of the *zombie* is overwritten with the flow identifier of the new packet. In this case the counter is reset as well. In addition depending on the current occupancy of the queue and the normalized hit rate for that flow, the new packet is dropped. Outlining the essence, the drop probability of a new arriving packet is stated by the following two formula. In the formula B is the total queue size, P_m^{max} the maximum dropping/marking probability, Q_C is the current queue length, $P_{est}(t)$ is an factor estimating the number of active flows, and $Hit(t)$ is either 1 or 0, depending on whether the current packet had a match in the *zombie* list or not.

$$P_{SRED}(Q_C) = \begin{cases} 0 & if \quad 0 \leq Q_C < \frac{B}{6} \\ \frac{1}{4} \cdot P_m^{max} & if \quad \frac{B}{6} \leq Q_C < \frac{B}{3} \\ P_m^{max} & if \quad \frac{B}{3} \leq Q_C < B \end{cases}$$

The probability of dropping/marking is then:

$$P_m^{max}(Q_C) = P_{SRED}(Q_C) \cdot min(1; \frac{1}{(256 \cdot P_{est}(t))^2}) \cdot (1 + \frac{Hit(t)}{P_{est}(t)}$$

A classification according to Chapter 1.2 of SRED is presented in Table 2.6.

2.7 FRED: Fair Random Early Detection

Lin and Morris state in [LM97] that RED allows unfair bandwidth sharing when different types of traffic share one link. The reason for this is that RED does not take the bandwidth utilization of the flows into account, when dropping packets. Packets of flows using a great amount of the bandwidth are dropped with the same probability as packets of choked flows. The authors propose Fair RED (FRED) as solution. FRED measures the utilization

AQM classification	Stabilized Random Early Detection
Goal of algorithm	Like RED: provide performance guarantees (delay, throughput)
Solution approach	Heuristic
Congestion detection	$Q_{current}$ and diversity of entries in the *zombie* list.
Fair bandwidth alloc.	Yes, dropping probability is proportional to bandwidth share per flow.
Malicious-aware	No, as not fair.
Target quality interval	High entropy at incoming flows.
Required state	$O(M)$, where M is the static number of entries in the *zombie* list.
# of predef. params.	3
What has effect on P_{drop}	$Q_{current}$, P_m^{max}, M, Number of hits in the *zombie* list.
Special characteristic	Considers the bandwidth share of the flows.

Table 2.6: SRED in context of AQM classification

of bandwidth per flow in order to impose on each flow a loss rate that is related to its bandwidth utilization.

The main goal of FRED is to provide different dropping strategies to different kind of flows. Misbehaving flows, that take too much bandwidth, shall be isolated. Bursty and low-speed flows should be protected and spared from dropping. The authors introduce various parameters to model the queue of the system. Q^{min} and Q^{max} represent the number of packets a flow i is allowed to buffer. Q_{avg}^{min} and Q_{avg}^{max} are the minimum and maximum average buffer sizes. Q_i is the number of packet currently buffered for flow i and Q_{avg} the average buffer size in the system.

To protect flow that use less than their fair share of bandwidth, all incoming packets satisfying following condition are accepted:

$$(Q_i \leq Q_{avg}^{min}) \quad AND \quad (Q_{avg} < Q_{avg}^{max})$$

The second type of flows, called heterogeneous robust flows, are characterized by the condition

$$(Q_{avg}^{min} < Q_{avg} \leq Q_{avg}^{max}) \quad AND \quad (Q_i > Q_i^{min}) \quad AND \quad (Q_i > Q^{avg})$$

In this normal case packets are dropped according to the same probability function as used in RED. The dropping probability is $P_m = \frac{P_{avg}}{1 - count \cdot P_{avg}}$, where $count$ is the number of packets since the last dropped packet and $P_{avg} = P_m^{max} \frac{Q_{avg} - Q_{min}}{Q_{max} - Q_{min}}$.

Non-adapting flows are detected by their excess usage of bandwidth:

$$(Q_i \geq Q_i^{max}) \quad OR \quad ((Q_i \geq Q^{avg}) \quad AND \quad (flag_i))$$

In this last equation $flag_i$ is set to *true*, once $(Q^i \geq Q_i^{max})$ is valid. After this $flag_i$ remains true until flow i is showing well behavior for a period of time. Flows with long queues are penalized. Different to RED the counter for the average queue length Q^{avg} is updated not only each time a packet arrives but also each time a packet leaves the system. A classification according to Chapter 1.2 of FRED is presented in Table 2.7.

AQM classification	Fair Random Early Detection
Goal of algorithm	Make RED fair.
Solution approach	Heuristic
Congestion detection	$Q_{avg} \leq Q_{avg}^{max}$.
Fair bandwidth alloc.	Yes, P_m is proportional to bandwidth utilization per flow.
Malicious-aware	No.
Target quality interval	Static target interval $[Q_{avg}^{min}, Q_{avg}^{max}]$ for Q_{avg}.
Required state	$O(\|F\|)$, where F is the set of flows.
# of predef. params.	5
What has effect on P_{drop}	Q_i, Q_{avg}, Q_{avg}^{min}, Q_{avg}^{max}, P_m^{max}.
Special characteristic	RED combined with per-flow state.

Table 2.7: FRED in context of AQM classification

2.8 RED-PD: Random Early Detection with Preferential Dropping

In [MFW01] Mahajan, Floyd and Whetherall introduce a Preferential Dropping Mechanism for RED. They suggest to maintain a dropping history and identify by this flows that utilize bandwidth in large amount. Packets corresponding to flows identified by the dropping history are preferred at dropping. For the other flows normal RED should be applied.

RED-PD is only active if there is not enough bandwidth to provide sufficient service to all flows. In the case of congestion flows that use more of the bandwidth than their fair share should be cut back in service to a *target* bandwidth by packet dropping. The authors assume that the behavior of flows can be predicted by looking at their prior behavior.

Therefore monitoring flows can detect high-bandwidth consumption and it is fair to cut corresponding flows back.

Upon arrival of a packet it is checked whether its flow is already detected as exceeding its fair share. If so, the packet is dropped with a flow-specific probability. If the flow is not suspected of being consuming too much service, it is dropped with a probability according to normal RED. In the case that the packet is really dropped, the flow is checked by inspecting the drop history, whether it should be monitored and degraded or not. It has to be noted that the drop history only contains packets dropped with normal RED and not those packets that are preferred for dropping, due to being monitored.

RED-PD uses several lists containing the drop history of consecutive intervals of time. Let D_T be the target delay of a reference flow, P the steady-state packet drop rate, H_l is the number of history lists and K the number of hits necessary to identify a flow as high-bandwidth utilizing. Each list has a length of $\frac{K}{H_l} \cdot \frac{D_T}{\sqrt{1.5P}}$. A flow is identified as using too much bandwidth, when it has losses in at least K out of H_l lists. Flows are monitored until they decrease their bandwidth consumption below a threshold of $f(D_T, P) \approx \frac{\sqrt{1.5}}{D_T \cdot \sqrt{P}}$. Iteratively the dropping probabilities are adapted to support the convergence of utilization to this threshold. The probability of dropping increases for flows that use too much bandwidth and are not cooperate. However, dropping probability never reaches 1 as false positives may exist. Concludingly the probability $P_{unmon,j}$ for dropping of an unmonitored flow j is halved each round and released after falling below a certain threshold P_{min}. The dropping probability $P_{mon,i}$ of a monitored flow i is increased by $P_{\triangle} = \frac{dropcount_i}{dropcount_{avg}} \cdot P$.

The solution presented by the authors provides relative fairness among monitored flows while avoiding starvation of monitored flows. Unmonitored flows are protected from the excessive usage of bandwidth by the monitored flows. A classification according to Chapter 1.2 of RED-PD is presented in Table 2.8.

2.9 CHOKe: Choose and Keep Packets from Responsive Flows Choose and Kill Packets from Unresponsive Flows

CHOKe has been proposed in the year 2000 by Pan, Prabhakar and Psounis [PPP00]. The authors aim two main goals for the design of CHOKe:

- CHOKe shall be applicable on high-speed routers, therefore it has to be memoryless to avoid additional state.
- Min-max fairness should be provided, by dropping packets of flows, that utilize the bandwidth much more, at a higher rate.

AQM classification	Random Early Detection with Preferential Dropping
Goal of algorithm	Fair RED and malicious flow detection: cut back to target bandwidth.
Solution approach	Heuristic
Congestion detection	Based on Q_{avg}.
Fair bandwidth alloc.	Yes, P_m is proportional to bandwidth utilization per flow.
Malicious-aware	Yes.
Target quality interval	Adapt per-flow buffer to Q_{avg} and to D_T, a target delay.
Required state	$O(H_l)$, where H_l is the drop history flows.
# of predef. params.	5
What has effect on P_{drop}	Q_i, H_l, average and per-flow drop count.
Special characteristic	Like SRED but history contains dropped packets.

Table 2.8: RED-PD in context of AQM classification

Like the previous strategies to, CHOKe is an extension to RED and uses the same parameters like RED to describe its algorithm: Q_{min} and Q_{max} define the threshold boundaries of the queue size. The average queue size is Q_{avg}, W_q is the queue weight and Q_{curr} the current queue size. CHOKe is only active if there is congestion ($Q_{avg} > Q_{min}$). The main idea of CHOKe is to compare an arriving packet with m random packets in the queue. All packets having the same flow identifier like the arriving packet are dropped. If they do not belong to the same flow, CHOKe checks whether the maximum utilization is reached ($Q_{avg} > Q_{max}$). If the maximum utilization is reached, the new packet is dropped. Otherwise the packet is enqueued with a probability of P, that is related to Q_{avg} similar to the drop rate in RED.

The parameter m, deciding the number of packets with which the incoming packet is compare, has effect on the detection quality. The authors suggest to partition the queue space between Q_{min} and Q_{max} in smaller intervals R_i and increase m proportional to the average queue length: $m = 2 \cdot i \;\; for\, Q_{avg} \in R_i$. In Figure 2.3 we show the main principle of CHOKe.

CHOKe is fairer than RED, as it drops packets from flows, that use the bandwidth more intensive, at a higher rate. However, CHOKe does not provide min-max fairness, that means, it may be, that a flow is receives more service on the cost of another flow, that has already less service. A classification according to Chapter 1.2 of CHOKe is presented in Table 2.9.

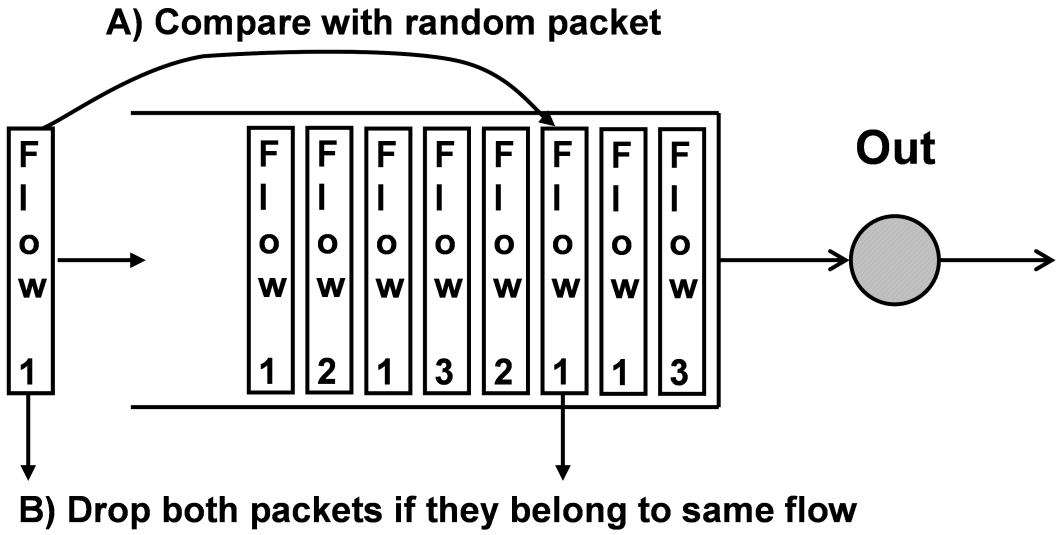

Figure 2.3: This figure shows the main principle of CHOKe: Upon arrival of a new packet, it is compared with one packet randomly picked from the queue. If both flow-affiliations match, the two packets are dropped.

AQM classification	Choose and Keep Packets from Responsive Flows
Goal of algorithm	Stateless, fair, RED-based algorithm
Solution approach	Heuristic
Congestion detection	Based on $Q_{current}$.
Fair bandwidth alloc.	Yes, P_m is proportional to bandwidth utilization per flow.
Malicious-aware	No.
Target quality interval	High packet entropy, fair share for all flows.
Required state	$O(1)$, algorithm works on current queue.
# of predef. params.	5
What has effect on P_{drop}	Q_{avg}, packets in the queue.
Special characteristic	Drops packets at the end and in the middle of the queue.

Table 2.9: CHOKE in context of AQM classification

2.10 E-RED: Exponential Random Early Detection

In order to improve RED's stability Liu, Basar and Srikant proposed in [LBS05] an extension to RED: Exponential RED. The solution implements a primal-dual algorithm, known from optimization theory, in order to compute optimal dropping parameters for RED.

The mathematical model uses similarly to RED the following parameters: Q_{min} and Q_{max} define the threshold boundaries of the queue size. The average queue size is Q_{avg}, C is the link capacity of the system, Q_c the current queue size and P_m^{min} denotes the minimum dropping probability for Q_c being greater than Q_{min}.

The dropping (or marking) probability of arriving packets is

$$P_m = \begin{cases} 0 & if \quad 0 \leq Q_c \leq Q_{min} \\ P_m^{min} \cdot e^{\frac{Q_c(Q_c - Q_{min})}{C}} & if \quad Q_{min} < Q_c < Q_{max} \\ 1 & if \quad Q_{max} \leq Q_c \end{cases}$$

This means, that in contrast to RED the drop probability is increasing exponentially and not straight proportional. As a result the virtual queue length in E-RED oscillates around its equilibrium very slightly. Due to this E-RED is more stable and predictable than RED. A classification according to Chapter 1.2 of E-RED is presented in Table 2.10.

AQM classification	Exponential Random Early Detection
Goal of algorithm	Improved stability (at the cost of support for bursty traffic).
Solution approach	Optimization theory
Congestion detection	Based on $Q_{current}$.
Fair bandwidth alloc.	Yes, P_m is related to bandwidth utilization per flow.
Malicious-aware	Yes, exponential dropping.
Target quality interval	Variance of $Q_{current}$ should converge against 0.
Required state	$O(1)$, the average queue length Q_{avg}.
# of predef. params.	3
What has effect on P_{drop}	Q_C, Q_{avg} and P_m^{min}.
Special characteristic	Exponentially increasing dropping probability.

Table 2.10: E-RED in context of AQM classification

2.11 AVQ: Adaptive Virtual Queue

An Adaptive Virtual Queue Algorithm (AVQ) [KS04] was proposed by Kunniyur and Srikant to achieve the stability of the queue length. AVQ tries to keep the queue length constantly small in order to reduce the end-to-end delay experienced by users. The approach is to maintain a virtual queue with capacity smaller than that of the real queue. On packet arrival the size of the virtual queue is increased and if it is full the packet is dropped. On packet departure the size of the virtual queue is decremented. The maximum size of the virtual queue is adapted as follows: $V_{max} = \alpha \cdot (\gamma \cdot Q_{current} - \lambda)$ where γ is the arrival rate at the link, α the smoothing parameter and λ the desired utilization of the link. Note, that the virtual queue can be implemented as a simple counter storing its actual size together with its current maximum value. In Figure 2.4 we present the main principle of AVQ.

AVQ requires no probability to be computed and regulates the link utilization by providing early feedback to the flow sources. The main motivation is to achieve robustness in the presence of very short flows, to keep the real queue small (and so the end-to-end delay) and to achieve the stability of the queue. Unlike in RED the fluctuations of the current queue length are small. By keeping the delay and packet loss small while the utilization of the link is high AVQ tries to maximize the sum of utility functions of single users. A classification according to Chapter 1.2 of AVQ is presented in Table 2.11.

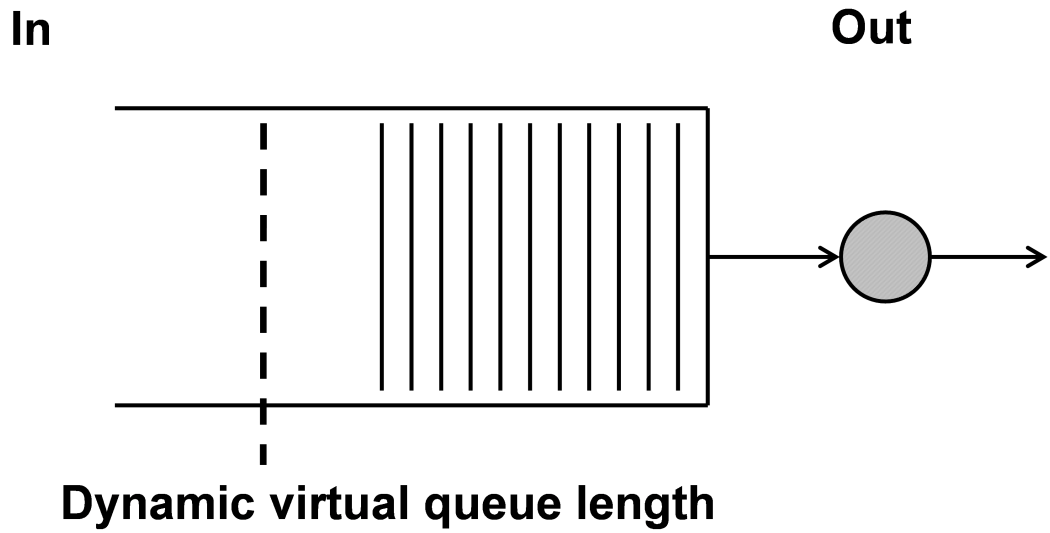

Figure 2.4: This figure shows the main principle of AQM. The size of a virtual queue is dynamically adapted to provide tighter performance bounds.

AQM classification	Adaptive Virtual Queue
Goal of algorithm	Robustness, stability and delay bounds. (Regulate the queue size).
Solution approach	Deterministic
Congestion detection	Congestion if $Q_{current} \geq VQ_{max}$.
Fair bandwidth alloc.	No.
Malicious-aware	No.
Target quality interval	$Q_{current} \leq VQ_{max}$ and desired link utilization λ.
Required state	$O(1)$, the average virtual queue length VQ_{max}.
# of predef. params.	3
What has effect on P_{drop}	Only Q_C and the three static predefined parameters.
Special characteristic	No probabilities are computed, deterministic capacity.

Table 2.11: AVQ in context of AQM classification

2.12 PI: Proportional Integral (Controller)

In [HMTG00] Hollot et. al. apply the control theory to design a controller which can regulate the queue length and keep the queuing delay small. Thereby some limitations

of RED, namely coupling of queue length and loss probability together with the slow response to load changes. The approach uses the current queue size as feedback input for the current link utilization and design an appropriate Proportional Integral (PI) controller.

To simplify the implementation of PI controller the integral expression is linearized and results in marking probability in time-period t being computed as

$$P_t = a \cdot \triangle Q(t) - b \cdot \triangle Q(t-1) + P(t-1)$$

where $\triangle Q(t) = Q_{curr}(t) - Q_{target}$, Q_{target} is the target queue size and a, b are coefficients. In their experiments the authors set $a = 1.822 \cdot 10^{-5}$ and $b = 1.816 \cdot 10^{-5}$. If the queue size is near to the target queue size and the queue size changes marginally between samples then the system in the *steady state*, i.e the queue length is close to the target.

Open issues of this approach are that the loss rate may be unnecessary high for bursty traffic and its fairness. A classification according to Chapter 1.2 of PI is presented in Table 2.12.

AQM classification	Proportional Integral (Controller)
Goal of algorithm	Regulate the queue length to keep the delay small.
Solution approach	Control theory
Congestion detection	Changes in the queue size.
Fair bandwidth alloc.	No.
Malicious-aware	No.
Target quality interval	Q_{target}, static target queue size.
Required state	$O(1)$.
# of predef. params.	2
What has effect on P_{drop}	Q_t, Q_{t-1} P_{t-1}
Special characteristic	Linearized probability calculation.

Table 2.12: PI in context of AQM classification

2.13 PIP: Proportional Integral Controller with Position Feedback Compensation

Heying et al. extended the Proportional Integral Controller with position feedback compensation in [HBW03] in order to improve the robustness and responsiveness. In PIP the marking probability depends not only on the changes in the current queue size and the

target queue size but also on the *trend* of the queue size. Again the control theory is used to derive the formula which results in following marking probability for the k-th packet:

$$P(k) = P(k-1) + \frac{\triangle t}{T}(Q_{curr}(k) - Q_0) + (\frac{\tau}{T} - K_h)(Q_{curr}(k) - Q_{curr}(k-1))$$

The probability is computed from three components: the previous drop probability, the weighted deviation from the reference queue length and the weighted drift trend. The drift trend is computed as the difference of the current and the previous queue size. Appropriate values for the parameters K_h, τ and T are computed based on the upper RTT bound, lower bound for the number of TCP sections and link capacity.

The authors argue that PIP can eliminate the PI's inaccuracy and its sensitivity to the changes in system parameters like load level and propagation delay. So PIP tries to minimize the queue length oscillations under varying network conditions. A classification according to Chapter 1.2 of PIP is presented in Table 2.13.

AQM classification	Proportional Integral Controller with Position Feedback Compensation
Goal of algorithm	Minimize PI's queue length variance.
Solution approach	Control theory.
Congestion detection	Changes in the queue size.
Fair bandwidth alloc.	No.
Malicious-aware	No.
Target quality interval	Q_{target}, static target queue size.
Required state	$O(1)$.
# of predef. params.	4
What has effect on P_{drop}	Q_t, Q_{t-1} P_{t-1}
Special characteristic	Like PI but faster response to load changes.

Table 2.13: PIP in context of AQM classification

2.14 REM: Random Exponential Marking

In [ALLY01] Athuraliya et al. suggested to decouple the congestion measure from the performance measure. The congestion measure should represent the demand for bandwidth and the number of active users, while the performance measure should be adjusted around its target value to keep the packet loss and delay small.

The performance is measured in queue length and loss. The congestion measure,

called *price*, is a weighted sum of the rate mismatch and queue mismatch and its value is updated periodically. So the price depends on the number of active users and the current bandwidth consumption. If the price is low the user will increase their send rate and reduce it if the price is high. The authors state that in equilibrium the mismatches are close to zero, the link utilization is high, while the loss and delay are small. The price is computed as follows (with the rate mismatch approximated from the queue length changes):

$$C_{price}(t) = max\{0, P(t-1) + \gamma(Q_{curr}(t) - (1-\alpha)Q_{curr}(t-1) - \alpha Q_{ref})\}$$

where α and γ are small constants. $Q_{curr}(t) - Q_{ref}$ is the queue mismatch while $Q_{curr}(t) - Q_{curr}(t-1)$ is the approximation of the rate mismatch.

The marking probability for the price P is computed as: $P_m(t) = 1 - \phi^{-C_{price}(t)}$ where ϕ is a constant bigger than 1.

A further feature of REM is that due to the exponential dropping probability the end users can observe the total price of the flow's path. The total marking probability for a path is $P_{total} = 1 - \phi^{-\sum_l^L P_l(t)}$ and for small marking probabilities on each router $P_{total} \backsimeq (log\phi)\Sigma_l^L P_l(t)$

Note, that the price used by REM equals to the PI's marking probability if we set the coefficients appropriately. A classification according to Chapter 1.2 of REM is presented in Table 2.14.

AQM classification	Random Exponential Marking
Goal of algorithm	Decouple congestion measure and performance.
Solution approach	Optimization
Congestion detection	Bandwidth consumption and number of active users.
Fair bandwidth alloc.	Yes.
Malicious-aware	Yes. Dropping probability increases exponentially.
Target quality interval	Static target performance Q_{ref}
Required state	$O(1)$.
# of predef. params.	3
What has effect on P_{drop}	$Q_{current}$ and # of active users
Special characteristic	User can observe the price for whole path

Table 2.14: REM in context of AQM classification

2.15 BLUE: BLUE Active Queue Management Algorithm

The BLUE algorithm [cFSKS02] proposed by Feng et al. uses packet loss and eventually occurring idle times of the output link as indicator of congestion and tries to prevent high loss rates and to reduce the queue length oscillations. The main idea is to increase the marking probability upon each packet's arrival event by δ_1 and to decrease it upon each idle event by δ_2 if the time elapsed between two events is larger than a time-period called *freeze-time*. δ_1 should be bigger than δ_2, e.g. by factor of 10. The *freeze-time* parameter should be randomized to avoid global synchronization.

An extension to BLUE called Stochastic Fair BLUE use Bloom Filters to identify non-responsive flows with small space consumption. The approach uses L levels with N bins on each level. L independent hash functions map each flow identifier to L bins, one bin per layer. Each arriving packet increases the size of its bins on all levels. If a bin overflows the dropping probability assigned to it is increased or decreased if the bin becomes empty. The dropping probability of a packet a minimum of dropping probabilities of all its bin. A flow with marking probability of one is considered to be non-responsive and its transmission rate is limited. As flows share some bins there may be some false-positives which in turn can be alleviated by exchanging the hash functions periodically. Setting proper values for L and N is an open question. A classification according to Chapter 1.2 of BLUE is presented in Table 2.15.

AQM classification	BLUE Active Queue Management Algorithm
Goal of algorithm	Low loss rates and low queue length oscillation.
Solution approach	Heuristic
Congestion detection	Arrival rate of packets.
Fair bandwidth alloc.	Yes.
Malicious-aware	No.
Target quality interval	No, just minimize loss rate and queue size.
Required state	$O(L \cdot N)$.
# of predef. params.	1
What has effect on P_{drop}	Packet loss per flow and packet arrival rate
Special characteristic	Hash-bin based detection of greedy flows

Table 2.15: BLUE in context of AQM classification

2.16 Comparative Performance Results

2.16.1 AQM Schemes on the Internet

In [BRHG] Bitorika et al. compare several AQM schemes in the same evaluation setup. As there are approximately 50 proposals of AQM schemes, the authors selected a subset of them. They selected 8 schemes (ARED, REM, CHOKe, PI, AVQ, DRED, GREEN and LDC) due to the following methodology:

- the schemes can be deployed on the Internet incrementally (i.e. work with the currently deployed routers and end-user systems)
- no per flow state required
- applicable to best-effort IP networks

The analyzed approaches were designed based either on a heuristic (ARED, CHOKe, AVQ, GREEN and LDC), the control theory (PI and DRED) or optimization (REM). Most of them use current queue length as congestion metric. The only exceptions are GREEN which uses traffic load as metric and LDC which uses both traffic load and queue length. The schemes focus on optimization of different characteristics like overall network performance (ARED, AVQ, GREEN, LDC), queue stability (REM, PI, DRED) or fairness (CHOKe). The tested fairness was Jain's fairness index which is computed as $\frac{(\sum x_i)^2}{n \sum x_i^2}$ where x_i it the share of the bandwidth for the flow i and n is the total number of flows. The evaluation scenario comprises two topologies:

Dumpbell single link with one-way congestion

Reversebell reverse path traffic with multiple congested links

The different measurements used a mix of long-lived TCP flows, short-lived TCP flows and unresponsive UDP flows which send a constant bit rate.

The evaluation results can be roughly summarized as shown in Table 2.16. The authors observer that PI, DRED, ARED and REM perform very similar, AVQ is good in bandwidth utilization and keeping queue length small while CHOKe provides better fairness but lower utilization than AVQ. LDC is too aggressive in dropping packets and GREEN performs very similar to Drop Tail.

2.16.2 AQM Schemes and Web Performance

Le et al. analyzed in [LAJS03] the impact of three AQM schemes (PI, REM and ARED) on Web Performance with and without ECN. In their evaluation setup only Web-like TCP

AQM scheme	Queue Length Stability	Jain's Fairness Index	Performance
ARED	+	~ 0.25	++
REM	0	~ 0.25	+
CHOKe	0	~ 0.35	++
PI	+	~ 0.25	+
AVQ	++	~ 0.15	+++
DRED	++	~ 0.25	+
GREEN	- -	~ 0.25	- -
LDC	0	~ 0.25	0

Table 2.16: Comparison of AQM schemes

flows were generated and the links were loaded with 80%, 90%, 98%, and 105%. In this setup and special implementations of these AQM schemes the authors observed:

- At 80% load all three AQM schemes provide no performance gain
- ARED performs similar to Drop Tail
- With ECN enabled PI and REM perform better than Drop Tail at 90% load and more
- At 90% load PI performs better than Drop Tail
- At 98% and 105% load PI and REM are slightly better than Drop Tail and ARED

It has to be mentioned that the result of the evaluation presented in [BRHG] are somehow limited as AQM was designed to alleviate performance during congestion and so AQM schemes should not be expected to improve the performance for a link load of less than 100%. Actually, it should be tested for scenarios where the load is at least temporarily significantly bigger than 100%.

Chapter 3

Classification of Selected Active Queue Management Mechanisms

3.1 Classification according to the goal of the solution

In Table 3.1 we present the goals of the surveyed AQM mechanisms. The motivation for proposing a new AQM solution is considered important, as it shows the main contribution. However, the goals are very diverse so a classification in a few classes is not feasible.

3.2 Classification according to the chosen approach

We identified one main class of approaches and several approaches, that are use rarely. In Table 3.2 we give a classification of the surveyed AQM mechanisms according to the used approach. Heuristic approaches dominate the list, they use different models to determine a dropping/marking probability for incoming packets. Approaches based on control theory or optimization theory are rare, they try to adapt the parameters or to find optimal dropping probabilities.

3.3 Classification according to type of congestion detection

Table 3.3 lists the methods used in the various surveyed AQM mechanisms to detect congestion. Most of the presented solutions monitor the actual queue length to determine whether congestion exist or not. Another large group of solutions use an estimation of the average queue length in order to control the influence of burstiness. Some rare approaches consider only the changes in the queue length or take only the arrival rate of packet into account.

AQM classification	Goal of algorithm
ECN	Report congestion
DT	Solve congestion problem
RED	Provide performance guarantees (delay, throughput)
ATM-RED	RED optimized for cell-based architecture in ATM networks
ARED	Adaptive trade-off between link utilization and delay
SRED	Like RED: provide performance guarantees (delay, throughput)
FRED	Make RED fair
RED-PD	Fair RED and malicious flow detection: cut back to target bandwidth
CHOKe	Stateless, fair, RED-based algorithm
E-RED	Improved stability (at the cost of support for bursty traffic)
AVQ	Robustness, stability and delay bounds. (Regulate the queue size)
PI	Regulate the queue length to keep the delay small
PIP	Minimize PI's queue length variance
REM	Decouple congestion measure and performance
BLUE	Low loss rates and low queue length oscillation

Table 3.1: Comparing the goal of selected AQM algorithms

3.4 Classification according to fairness of the solution

The classification of the surveyed AQM mechanisms with respect to their fairness is presented in Table 3.4. There exist only two cases: Either the solutions provide fairness or not. An exception to this is ATM-RED, which proposes various strategies to drop cells. Depending on the chosen strategy ATM-RED is fair or not.

3.5 Classification according to malicious-awareness

AQM mechanisms that are fair, can be malicious-aware as well. In this case they punish misbehaving flow to compensate for service that has been obtain by fraud. Only few AQM solutions are malicious-aware. An overview on this classification can be found in Table 3.5.

AQM classification	Solution approach
ECN	Not discussed
DT	Deterministic
RED	Heuristic
ATM-RED	Heuristic
ARED	Heuristic
SRED	Heuristic
FRED	Heuristic
RED-PD	Heuristic
CHOKe	Heuristic
E-RED	Optimization theory
AVQ	Deterministic
PI	Control theory
PIP	Control theory
REM	Optimization
BLUE	Heuristic

Table 3.2: Comparing the solution approach of selected AQM algorithms

3.6 Classification according to the target quality of the solution

In Table 3.6 we present the results of our investigation on the target quality in the surveyed AQM mechanisms. Most of the solutions aim to keep the queue length in an target interval, or near a target value. With near-constant queue length, delay and throughput are predictable. Target values define a trade-off between these two performance goals. However some rare mechanism aim at a high diversity of the flow membership of the packets in the queue.

3.7 Classification according to state requirements

Most of the AQM mechanisms aim at a low state complexity of $O(1)$. This is necessary for the quick processing of packets in a router. In most cases the mechanism maintains some data structure to measure the average queue length and to store predefined parameters. However in a peer we have stronger computational components, so that solution needing

AQM classification	Congestion detection
ECN	Not discussed
DT	$Q_{current} \geq Q_{max}$
RED	Q_{avg}, weighted average with burst-awareness
ATM-RED	Q_{avg}, weighted average with burst-awareness
ARED	Q_{avg}, weighted average with burst-awareness
SRED	$Q_{current}$ and diversity of entries in the *zombie* list
FRED	$Q_{avg} \leq Q_{avg}^{max}$
RED-PD	Based on Q_{avg}
CHOKe	Based on $Q_{current}$
E-RED	Based on $Q_{current}$
AVQ	Congestion if $Q_{current} \geq VQ_{max}$
PI	Changes in the queue size
PIP	Changes in the queue size
REM	Bandwidth consumption and number of active users
BLUE	Arrival rate of packets

Table 3.3: Comparing the congestion detection mechanism of selected AQM algorithms

more state are feasible as well. In Table 3.7 we present a comparison of the surveyed AQM mechanisms with respect to their state needed to be maintained.

3.8 Classification according to the number of predefined parameters

The number of predefined parameters give details on the complexity of the solution. Every parameter can be optimized for a specific behavior of the solution. Table 3.8 shows that the number of parameters used vary from 1 to 5.

AQM classification	Fairness
ECN	No, all flows are treated equal.
DT	No, all flows are treated equal.
RED	No, all flows have same dropping probability.
ATM-RED	Depends on strategy, ranging from unfair to fair.
ARED	No
SRED	Yes, dropping probability is proportional to bandwidth share per flow.
FRED	Yes, P_m is proportional to bandwidth utilization per flow.
RED-PD	Yes, P_m is proportional to bandwidth utilization per flow.
CHOKe	Yes, P_m is proportional to bandwidth utilization per flow.
E-RED	Yes, P_m is related to bandwidth utilization per flow.
AVQ	No.
PI	No.
PIP	No.
REM	Yes.
BLUE	Yes.

Table 3.4: Comparing the fairness of selected AQM algorithms

3.9 Classification according to effects on the dropping probability

Table 3.9 show the parameters of the system that effect the dropping probability in each surveyed AQM mechanism. This comparison is useful to estimate the behavior of the system. All AQM solutions take the current queue length (or an average) into account to measure whether congestion exists, in addition to this other parameters have in some cases effect on the decision which packet to drop.

3.10 Classification according to special characteristics

In Table 3.10 we present special characteristics of the surveyed AQM mechanisms. We do not introduce a strict classification on these special properties, as they are unique for the corresponding mechanism.

AQM classification	Malicious-awareness
ECN	No, as not fair.
DT	No, as not fair.
RED	No, as not fair.
ATM-RED	No, P_{drop} increases linearly with bandwidth utilization.
ARED	No, as not fair.
SRED	No, as not fair.
FRED	No.
RED-PD	Yes.
CHOKe	No.
E-RED	Yes, exponential dropping probability.
AVQ	No.
PI	No.
PIP	No.
REM	Yes. Dropping probability increases exponentially.
BLUE	No.

Table 3.5: Comparing the malicious-awareness of selected AQM algorithms

AQM classification	Target quality interval
ECN	$Q_{current} \leq Q_{max}$, where Q_{max} is static.
DT	$Q_{current} \leq Q_{max}$, where Q_{max} is static.
RED	$Q_{min} \leq Q_{current} \leq Q_{max}$, where Q_{min} and Q_{max} are static.
ATM-RED	Depends on strategy, from no target to dynamic target.
ARED	Static target interval Q_{target}
SRED	High entropy at incoming flows.
FRED	Static target interval $[Q_{avg}^{min}, Q_{avg}^{max}]$ for Q_{avg}.
RED-PD	Adapt per-flow buffer to Q_{avg} and to D_T, a target delay.
CHOKe	High packet entropy, fair share for all flows.
E-RED	Variance of $Q_{current}$ should converge against 0.
AVQ	$Q_{current} \leq VQ_{max}$ and desired link utilization λ.
PI	Q_{target}, static target queue size.
PIP	Q_{target}, static target queue size.
REM	Static target performance Q_{ref}
BLUE	No, just minimize loss rate and queue size.

Table 3.6: Comparing the target quality of selected AQM algorithms

AQM classification	Required state
ECN	Non, just report on detection.
DT	Non, just drop new packets on congestion.
RED	$O(1)$, maintaining Q_{avg}, but no history.
ATM-RED	Ranges from $O(1)$ to $O(number\ of\ flows)$.
ARED	$O(1)$
SRED	$O(M)$, where M is the static number of entries in the *zombie* list.
FRED	$O(\|F\|)$, where F is the set of flows.
RED-PD	$O(H_l)$, where H_l is the drop history flows.
CHOKe	$O(1)$, algorithm works on current queue.
E-RED	$O(1)$, the average queue length Q_{avg}.
AVQ	$O(1)$, the average virtual queue length VQ_{max}.
PI	$O(1)$.
PIP	$O(1)$.
REM	$O(1)$.
BLUE	$O(L \cdot N)$.

Table 3.7: Comparing the required state of selected AQM algorithms

AQM classification	Number of predefined parameters.
ECN	1
DT	1
RED	4
ATM-RED	4
ARED	3
SRED	3
FRED	5
RED-PD	5
CHOKe	5
E-RED	3
AVQ	3
PI	2
PIP	3
REM	2
BLUE	1

Table 3.8: Comparing the number of predefined parameters of selected AQM algorithms

AQM classification	What has effect on P_{drop}
ECN	$Q_{current}$, Q_{max}
DT	$Q_{current}$, Q_{max}
RED	$Q_{current}$, Q_{min}, Q_{max}, P_m^{max}, W_q.
ATM-RED	$Q_{current}$, Q_{min}, Q_{max}, P_m^{max}, W_q.
ARED	$Q_{current}$, Q_{min}, Q_{max}, α_i, β_i, Q_{target}, P_m^{max}
SRED	$Q_{current}$, P_m^{max}, M, Number of hits in the *zombie* list.
FRED	Q_i, Q_{avg}, Q_{avg}^{min}, Q_{avg}^{max}, P_m^{max}.
RED-PD	Q_i, H_l, average and per-flow drop count.
CHOKe	Q_{avg}, packets in the queue.
E-RED	Q_C, Q_{avg} and P_m^{min}.
AVQ	Only Q_C and the three static predefined parameters.
PI	Q_t, Q_{t-1} P_{t-1}
PIP	Q_t, Q_{t-1} P_{t-1}
REM	$Q_{current}$ and # of active users
BLUE	Packet loss per flow and packet arrival rate

Table 3.9: Comparing the effects on P_{drop} of selected AQM algorithms

AQM classification	Special characteristic
ECN	Dropping is avoided if possible.
DT	Most intuitive strategy to cope with congestion.
RED	Reference AQM algorithm. Burst-awareness adjustable.
ATM-RED	Takes the characteristics of ATM networks into account.
ARED	Adapt Q_{target} to meet delay and throughput requirements.
SRED	Considers the bandwidth share of the flows.
FRED	RED combined with per-flow state.
RED-PD	Like SRED but history contains dropped packets.
CHOKe	Drops packets at the end and in the middle of the queue.
E-RED	Exponentially increasing dropping probability.
AVQ	No probabilities are computed, deterministic capacity.
PI	Linearized probability calculation.
PIP	Like PI but faster response to load changes.
REM	User can observe the price for whole path.
BLUE	Hash-bin based detection of greedy flows.

Table 3.10: Comparing the special characteristic of selected AQM algorithms

Chapter 4

Conclusion

Peer-to-peer systems are emerging and popular networks. The scalability of the system is limited by the available bandwidth in the system. The are no restrictions for devices of any type not to participate in peer-to-peer networks. So 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 different up-link and down-link bandwidth. User devices can download faster, than they can upload. A system which relies on tight interaction of the end-user devices, like it is in P2P system, can cause congestion on the end-user device. More packets may be downloaded to process than replies can be transmitted. This may lead to congestion in peers and packets have to be dropped. In the case that the network is to be used for loss-critical flows, dropping of packets is unacceptable.

In order to provide guaranteed service for loss-critical flows even in congested networks, peer-to-peer systems have to adopt Active Queue Management mechanisms. AQM solutions decide in which cases incoming packets have to be dropped/marked to give the source of the corresponding flow feedback on the congestion. Furthermore, AQM mechanisms control the amount of bandwidth share a flow is allowed to have in the system.

In this document we presented a survey on popular Active Queue Management mechanisms discussed in the literature. Furthermore we analyzed them and derived a set of classification points. In the survey and concludingly in Chapter 3 we present a taxonomy on the surveyed AQM mechanisms. From this taxonomy and this overview on AQM solutions we can derive requirements for solutions for P2P systems. First of all, packet priorities have to be introduced in the systems, so that flows can be classified as loss-critical or loss-tolerant. Another point is, that flows in P2P systems need to be identified,

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

which is challenging, as there is no constant traffic flow in the P2P overlay between source and destination peers.

Once this is achieved several fair AQM mechanisms can be applied on P2P systems to provide guarantees on minimal loss. We identified fairness being important for P2P networks, as load balancing in the system is desired. Addtitionally having a target interval for the system performance, measured by the lenght of the queue, enables enhanced planing of the system's behavior. The amount of state a peer has to maintain has minor importance, as usually powerful peers participate in P2P networks. Still for the case that less capable peers are participating, the amount of required memory is to be considered.

We concluded in this document that the oldest strategies Drop Tail and Explicit Congestion Notification are least complex but do not provide any desireable features, they just solve the problem efficiently. The family of Random Early Detection mechanisms introduce fairness with binding the dropping probability of a flow to its bandwidth utilization. The RED based strategies are heurisitics aiming to provide all flows the same share of bandwidth, hindering misbehaving flows to utilize bandwidth on the costs of other flows. Adaptive Virtual Queue aims to reach a tradeoff between the delay time of packets and the throughput of the buffer. The Proportional Integral based solutions introduce a novel approach, they sense on the changes of the queue length and not the queue length itself, aiming to avoid oscillations. Random Exponential Markin introduces a pricing system to motivate senders to urilize cheaper, rarer used flows. BLUE uses bloom filters to combine the dropping probabilities of several flows.

For P2P systems it is challenging to define flows. Once this is done, congestion is easier to detect using characteristics of the queue, as its current or average length. Dynamic congestion detection strategies, as applied in PI, PIP and BLUE are more complex to realize and they may be inefficient. Fairness is essential in a P2P system, as a single peer shall not be able to stress the network on the costs of other peers. Here again most of the RED based approaches, REM and BLUE qualify. The other AQM mechanisms do not bind the dropping probability to the link utilization of a flow. Most of the solutions define a fix value or an interval for the target queue length. This is desirable in context of P2P systems, as one can control with this tool the load on each peer. However, some approaches like ARED and CHOKe aim to increase the diversity of flows in the queue. The effects of this on P2P networks are doubtable, evaluation is needed. In some overlays this strategy may be contraproductive.

However, an optimal AQM mechanisms 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 AQM mechanisms in PeerfactSim.KOM[2][KHK+06]. Further evaluation of the effects of AQM mechanisms on the behavior of P2P systems under high bandwidth utilization will follow.

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

Bibliography

[ALLY01] Sanjeewa Athuraliya, Victor H. Li, Steven H. Low, and Qinghe Yin. REM: Active queue management. *IEEE Network*, 15(3):48 – 53, May/June 2001.

[BRHG] Arkaitz Bitorika, Mathieu Robin, Meriel Huggard, and Ciaran Mc Goldrick. A comparative study of active queue management schemes.

[cFSKS02] Wu chang Feng, Kang G. Shin, Dilip D. Kandlur, and Debanjan Saha. The blue active queue management algorithms. *IEEE/ACM Trans. Netw.*, 10(4):513–528, 2002.

[FGS01] Sally Floyd, Ramakrishna Gummadi, , and Scott Shenker. Adaptive red: An algorithm for increasing the robustness of red's active queue management. Technical report, 2001.

[FJ93] Sally Floyd and Van Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Trans. Netw.*, 1(4):397–413, 1993.

[HBW03] Zhang Heying, Liu Baohong, and Dou Wenhua. Design of a robust active queue management algorithm based on feedback compensation. In *SIGCOMM '03: Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 277–285, New York, NY, USA, 2003. ACM Press.

[HMTG00] C. V. Hollot, V. Misra, D. Towsley, and W. Gong. On designing improved controllers for aqm routers supporting tcp flows. Technical report, Amherst, MA, USA, 2000.

[KHK+06] Aleksandra Kovacevic, Hans Heckel, Sebastian Kaune, 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.

[KS04] Srisankar S. Kunniyur and R. Srikant. An adaptive virtual queue (avq) algorithm for active queue management. *IEEE/ACM Trans. Netw.*, 12(2):286–299, 2004.

[LAJS03] Long Le, Jay Aikat, Kevin Jeffay, and F. Donelson Smith. The effects of active queue management on web performance. In *SIGCOMM '03: Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 265–276, New York, NY, USA, 2003. ACM Press.

[LBS05] Shao Liu, Tamer Basar, and R. Srikant. Exponential-red: a stabilizing aqm scheme for low- and high-speed tcp protocols. *IEEE/ACM Trans. Netw.*, 13(5):1068–1081, 2005.

[LM97] Dong Lin and Robert Morris. Dynamics of random early detection. In *SIGCOMM '97: Proceedings of the ACM SIGCOMM '97 conference on Applications, technologies, architectures, and protocols for computer communication*, pages 127–137, New York, NY, USA, 1997. ACM Press.

[MFW01] R. Mahajan, S. Floyd, and D. Wetherall. Controlling high-bandwidth flows at the congested router, 2001.

[OLW99] Teunis J. Ott, T. V. Lakshman, and Larry H. Wong. SRED: Stabilized RED. In *Proceedings of INFOCOM*, volume 3, pages 1346–1355, 1999.

[PPP00] Rong Pan, B. Prabhakar, and K. Psounis. Choke - a stateless active queue management scheme for approximating fair bandwidth allocation. 2:942–951, 2000.

[RBL99] Vincent Rosolen, Olivier Bonaventure, and Guy Leduc. A red discard strategy for atm networks and its performance evaluation with tcp/ip traffic. *SIGCOMM Comput. Commun. Rev.*, 29(3):23–43, 1999.

[RFB01] K. Ramakrishnan, S. Floyd, and D. Black. RFC 3168, The Addition of Explicit Congestion Notification (ECN) to IP. 2001.