Topic 8. Active Queue Management: An overview

Parts were taken from Richelle Adams, IEEE COMMUNICATIONS SURVEYS & TUTORIALS
Congestion

• A list of (not exhaustive) questions
  – What is congestion?
  – Where does it occur?
  – Can it be quantified?
  – How is it sensed and how is it measured?
  – What ‘measurable’ evidence of congestion exists?
  – How and where is it managed/controlled? Ideally, how is it avoided?
  – Are we doing well enough with TCP congestion control?
  – Are we doing the best we can with TCP congestion control?
Congestion

• No definition of congestion, but generally speaking, congestion will occur when
  – the total demand for a resource, e.g. link bandwidth, exceeds the capacity of the resource, OR
  – when the arrival rate at a link interface exceeds the departure rate out of the link interface

• Results of congestion include:
  – high delay in packet delivery;
  – packet losses
  – low throughput, and in extreme cases even congestion collapse (where the throughput diminishes, as the load demand increases)
Congestion avoidance and congestion control

- **congestion control** aims to bring the network out of an already congested state,

- **congestion avoidance** aims to keep the network at an optimal state (low delay, high utilization) and out of congestion in the first place

- Even though there may be congestion avoidance in a network, congestion control is still necessary for network recovery in the event congestion avoidance fails.
Congestion control approaches

• **End-to-end (TCP)**
  
  – **Simple** to implement algorithm, yet it provides ‘robustness’ in the control
  
  – **Intelligence pushed on the ends** of the network, thus simpler network core (fundamental design philosophy of the (early) internet)

  – commonly acknowledged that there is a **limit to how much control** can be provided in the end-to-end approach

  – Classical TCP implementation ‘creates’ **bursty and synchronised traffic behaviour**-- in order to sustain high throughput requires very large buffers in the network, with as a consequence build-up of very large delays. There is a trade-off between acceptable delays and throughput, which is not easy (if at all possible) to control.

  – **TCP end-to-end congestion control acts after congestion has already occurred** (as sensed by timeouts due to undelivered packets/ACK inferred as congestion state)

  – ‘**large’ time delay** (propagation delay+queueing delay) between the packet drop event at the router and the source **detecting this loss**---during this delay source keeps sending at transmission rates which the network cannot support, leading to a high number of packet drops
Congestion control

- **Network assisted**
  - More responsive, as feedback delay is less than end-to-end and congestion state can be assessed better (e.g. examining the queue state)
  - can provide more sophisticated (not necessarily complex) control approaches

  - Generally more complex, as well as pushing more complexity within the network, rather than the edge
  - So ‘benefits’ over end-to-end must be there to displace TPC congestion control… Even so, not that easy. Why?
Congestion control in the Internet

- **Source algorithm** (e.g., TCP)
  - Sending rates
  - Congestion feedback signal (e.g., packet loss or ECN packet mark)

- **Network algorithm** (e.g., scheduling, queue management)
  - Scheduling (e.g., Fair Queuing)
  - Queue management (e.g., drop tail, active queue management)
    - Congestion indicator (e.g., queue length)
    - Control function (e.g., piece-wise linear)
    - Packet marking algorithm (e.g., WURM)
Active Queue Management (AQM)

- formal introduction to IP networks in 1993 (RED) as a viable complementary approach for CC (but can also stand alone)
- steady stream of research output since 1993
- ‘crude’ taxonomy based on the general attributes of AQM schemes, and the design approaches taken such as e.g. heuristic, control-theoretic and deterministic optimization as well as congestion feedback evaluation and indication (e.g. inferred via drops, ECN bit set, Multivalued feedback, Explicit rate calculation, etc…)
- of additional interest is the role of AQM in QoS provisioning particularly in the DiffServ context, as well as the role of AQM in the wireless domain
Active Queue Management (AQM)

- **Aims to control congestion** (congestion prevention/avoidance)
- Can **complement the end-to-end** congestion control
- Can be a **proactive** congestion control scheme by which the network sends information (implicitly or explicitly) to the sources when it detects incipient congestion
- **More responsive**: As a complement to end-to-end TCP CC, as the congestion level increases the AQM scheme intensifies its feedback to the TCP endpoints, i.e., by marking or dropping more packets. The sources, in response to the congestion notifications, reduce their transmission rates so as to prevent queue overflow and limit the losses that can result
- AQM algorithm **acts within a router** (the place where the congestion is actually taking place) it will obtain more accurate congestion information faster than the traffic source at the edge can
Active Queue Management (AQM)

• with the packet dropping policy, since AQM acts on all packets in the queue, it also helps with the case of uncontrolled sources (e.g. video streaming over UDP), which despite being unresponsive to AQM congestion notifications, some degree of congestion control at the router is provided, as packets are dropped as incipient congestion is sensed.

• In case of the AQM random dropping (or marking) of packets before buffer-overflow occurs, it eases the global synchronization problem experienced when droptail queues are instead used.

• random dropping may also improve fairness among flows

• AQM can detect link congestion using any one or combination of: queue length, input rate, and events of buffer overflow and emptiness

• AQM has been recommended by the Internet Engineering Task Force (IETF) RFC2309
Active Queue Management (AQM)

• besides major objective of providing congestion control, AQM schemes can be designed with the aim to provide:
  – predictable queueing delay and
  – to maximize link utilization

• Given above goals are conflicting, more sophisticated schemes are required to control this tradeoff.
  – AQM allows one to design such schemes.
  – E.g. by keeping the queue length around a reference (target) value that is small, low and predictable queueing latency can be realised, independent of traffic load (i.e., the number of connections), whilst also ensuring high link utilization since unnecessary packet drops will not occur, and the queue will not ‘run’ empty (in the case the demand is there)
queue management schemes comprise:
(1) the congestion indicator,
(2) the congestion control function
(3) the feedback mechanism
AQM realisation

• congestion indicator (probe)
  – used by the queue management to decide/infer when there is congestion.
    • e.g. the congestion indicator could be the instantaneous queue length occupancy with
      the congestion level calculated as a % of queue occupancy

• feedback mechanism
  – congestion signal feedback to alert the source to alter its transmission rates
    • Could be binary (e.g. ECN bit) or full (multivalued) feedback of the congestion indicator
      state, or even explicit rate, as calculated by the control function at the queue level

• congestion control function
  – decides what must be done when congestion is detected (as calculated by
    the congestion indicator)
    • e.g. to drop all incoming packets with probability of one (1) when the congestion
      indicator signals that there is high congestion
AQM realization: Congestion indicator

- Different realisations exist. Some examples:
  - rate-based AQM – aim to keep packet arrival rate at the queue at a target value, say, at some percentage of the link capacity. It indirectly controls queue length
  - queue-based AQM – aim is to keep the queue length at a reference (target) target value. E.g. it uses instantaneous or average length of the queue, and its rate of change so as the affected control maintains the error between queue length and its reference close to zero
  - a combination of both arrival rate and queue length to measure congestion at the queue, can also be adopted for AQM. Some researchers say it can provide a tradeoff between stability and responsiveness
  - Yet, other measures can be adopted, e.g. round-trip time, link capacity (loading) and number of sources
AQM realization: Feedback mechanism

• There are many possibilities to provide feedback. E.g explicit or implicit, single bit or multivalued.

• implicit
  – Dropped packets (e.g. TCP will infer congestion through triple-duplicate acknowledgements or retransmission timeouts)

• explicit (single bit or multivalued)
  – ECN-marked (e.g. TCP reacts by inferring congestion as a loss event).
  – Explicit feedback, where the rate to be adopted by the source is indicated in the feedback signal
AQM realization: congestion control function

- Can take many forms, as e.g.
  - a **marking/dropping probability function** which maps the current level of congestion (as given by the congestion indicator) to the probability of marking or dropping the incoming packets. At the source the control function reacts to the event (dropped/marked packet), according to its own control function (e.g. TCP Reno)
  - an **explicit rate** calculated at the queue level using e.g. a control theoretic or optimisation based or ad-hoc based approach. At the source level, source ‘obeys’ the fed-back explicit rate.
Control implementation design

• control design follows heuristically or analytic approaches.
  – **Heuristic design**: depends heavily on intuition. Quite a number of AQM schemes have been built on heuristics and have performed exceptionally well, especially when compared to droptail queues,
  – **Drawbacks**
    • Not able to guarantee behaviour, not even reliability or stability, especially if operating conditions change from initial experimentation and parameter tuning
    • many of their parameters are ‘manually’ tuned, most often using discrete even simulation, according to differing network conditions (not all operating conditions can be covered, even if simulation is adopted).
    • effects of these parameters not fully understood; in some cases can quickly drive system into instability (and hence low utilization) or can cause quite sluggish behaviour (i.e., system slowly converges to a new operating level when traffic environment changes)
    • The simulation environment usually abstracts real behaviour; the important/real dynamics may be in the hidden/abstracted detail
Control implementation design

- **Control theoretic design:**
  - Aim is to provide a **more predictable response**, or at least know the bounds of operation before it fails. Essential behavior is guaranteed over a known operating region (preferably globally), plus other desirable features such as faster responding system can be assessed.
  
  - By modeling one can gain much deeper **insight** into the system’s behaviour (for a wider range of possible scenarios) and can prove essential controlled system behavior, as well as desirable features, such as speed or response.
  
  - However, an ‘**accurate**’ enough model of the behaviour of the system end-to-end, including all the interacting components is necessary.
  
  - Often, even if the **mathematical model** is derived and it is accurate, it turns out to be overly complex for the control design tools to handle.
  
  - Many design approaches: some adopt non-linear designs (not easy on the maths), some linearise, but important system behaviour may not be ‘captured’, yet others rely on other types of models, e.g. fuzzy* to implement the control function, or even adopt optimisation techniques.

*we will briefly look at fuzzy later
Control theoretic approaches

- AQM design using classical control theory (more later)
Control theoretic approaches

- Fuzzy-logic controller structure (more later)
AQM design

• Optimisation based designs
  – The original optimization approach was developed by Kelly et al in 1998, extended further by Low et al, and many more similar approaches since then
  – formulates congestion control problem as resource utilisation, i.e. maximization of an aggregate source utility via an optimisation algorithm (e.g. approximate gradient) subject to network capacity constraints. The optimal controller (i.e., the AQM scheme) from this approach typically optimises source rates or the congestion measure
  – it is a global optimization process with the computation carried out by sources and links in the network (i.e. distributed computation).
  – We will briefly outline Adaptive Congestion Protocol-ACP by Lestas et al
Optimisation based techniques
### Comparisons of Different Approaches to Designing AQM Schemes

<table>
<thead>
<tr>
<th></th>
<th>No AQM (i.e., DropTail)</th>
<th>Heuristic</th>
<th>Control-Theoretic</th>
<th>Deterministic</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Advantages</strong></td>
<td>• Simplicity in implementation</td>
<td>• Simplicity in design and implementation to a certain extent</td>
<td>• Justification available for choice of parameter values, i.e., stability, responsiveness, transient analysis</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Lock-out</td>
<td>• Arbitrary and difficult parameter tuning</td>
<td>• Most are TCP-centric</td>
<td>• No TCP model assumed</td>
</tr>
<tr>
<td></td>
<td>• Global synchronization</td>
<td></td>
<td>• Limited TCP model used</td>
<td>• Can work different round-trip times among users</td>
</tr>
<tr>
<td></td>
<td>• Network instability</td>
<td></td>
<td>• Linearization of non-linear system</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• No service differentiation</td>
<td></td>
<td>• Congestion avoidance phase – long-lived flows</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• No explicitly treatment for fairness</td>
<td></td>
</tr>
<tr>
<td><strong>Disadvantages</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Assumptions</strong></td>
<td></td>
<td></td>
<td>Most analyses consider</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• A single congested link</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• No retransmission timeouts</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Synchronized sources</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Same RTT</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Number did not change with time</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Same window progression</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Same window sizes</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Same average packet size</td>
<td></td>
</tr>
</tbody>
</table>
AQM performance

• Besides achieving main goals of congestion avoidance, predictable queueing delay (that is low), and high link utilization, an AQM scheme should promote high network stability, robustness, responsiveness and scalability.
  – In terms of **robustness**, it is necessary that AQM algorithm performs consistently well under extreme and unfavourable network conditions, i.e., it should not be sensitive to variations in network parameters
  – robustness is enhanced when parameters in AQM scheme can be dynamically tuned according to changing traffic load
  – robustness can increase when AQM takes into account the long-range dependence property of the traffic since this can provide sufficient information to predict traffic intensity
AQM performance

- With regards to **stability**, the performance of AQM algorithm should not vary dramatically due to a sudden change in network conditions.
  
  • One test of stability is the extent to which the queue length varies when, for example, the number of connections flowing through the queue suddenly increases. It is imperative to avoid oscillations between queue overflows and underflows since this surely leads to overall link under-utilization and variation in queuing delay.

- **Scalability** essential, since one would want the AQM to continue to operate soundly even when the speed and the number of routers and links have dramatically increased in a network.
AQM performance

• The **ideal AQM**:  
  
  – should **achieve** stability, robustness, responsiveness, high throughput, low latency and fairness all at the same time, whilst being scalable and very simple to implement  
  
  – deliver congestion notifications at a rate so as to **keep the aggregate TCP sending rates into the queue just below the output rate of the queue**  
  
  – indicator and control function **should be adaptive** to changes in amount and nature of traffic
AQM performance

• The ideal AQM:
  – ensure that the following be accomplished:
    • does not unnecessarily penalize bursty flows and also minimizes global synchronization, leading to lower packet losses and therefore higher throughputs.
    • deals with bursts so that packets are not dropped unnecessarily.
    • minimize jitter which is due to large oscillations in queue length. Also, these oscillations are such that the queue becomes empty quite often. This leads to low link utilization
    • work in large delay networks
    • does not introduce additional bias to flows with smaller round-trip times (TCP does this)
    • work in an environment with multiple bottlenecks and should take into account the flows that are already being controlled by other nodes.
AQM performance

• Factors that impact AQM performance:
  – **Buffer size and link capacity.** These factors tend to be time-invariant factors which are pre-determined.
  – **Traffic load and round-trip time:** The traffic load (which is the number of flows) and the RTT change dynamically with time. The AQM must be robust enough to deal with such behavior, as an increase in the round-trip time degrades system stability.
  – rate-variability within and the **phase relation** among flows impact AQM performance. If window progression of an individual TCP flow is a smooth process so will be its arrival rate into the queue. If all the flows entering a queue are uniformly out-of-phase then AQM performance will not be significantly impacted, however if they are completely in-synch with each other then the **burstiness** will be more intense and the AQM performance may deteriorate.
AQM performance

• **Factors** that impact AQM performance:
  
  – there is a **connection** between routing and congestion. Using a term called the routing matrix gain to mathematically represent topologies (see Kelly et al.), it was found that the network robustness was inversely proportional to this routing matrix gain. If it were possible to know a priori the topology of the network in which the AQM would exist, then the AQM could be tuned to deal with this potential instability. But, in reality, this could not be done, therefore there was the question of whether or not some range of routing matrix gains exists which can instead be used. Routing affects congestion (and hence AQM performance) because it determines the path of queues through which packets must flow. If there are a number of congested queues along the path, the round-trip time (and hence the TCP sender’s ability to respond to congestion) is affected due to varying queue lengths along the path.
  
  – The presence of **short-lived flows** and **unresponsive flows** among the responsive TCP long-lived flows, and even a mix of TCP variants.
  
  – **Reverse-path asymmetry**. With congestion on the reverse path, ACK packets can be lost or bunched together (ACK compression). This in turn increases the burstiness of TCP
AQM performance

• Quite a number of performance metrics used and presented in the literature for evaluation and comparison of AQM schemes.

• The lack of consistency in evaluating AQM may have been the reason that there had been very little deployment in real networks. To this end a benchmarking framework for AQM was proposed.

• But it must be understood that even though a standard set of performance criteria were established, the task of comparing AQM algorithms will still be a daunting one, especially for the heuristic schemes. Nevertheless, whatever performance comparison is attempted, it is customary practice to use the simple droptail queue as the baseline.
AQM performance – other factors

• Other important factors/attributes of an AQM scheme is **fairness**, coping with unresponsive flows, and QoS guarantees

• To deal with issue of fairness and unresponsive flows, some AQM algorithms for their own operation use some degree of per-flow information as do their scheduling counterpart. Although it is preferrable not to use any state information whatsoever, AQM schemes that use no per-flow information, though least complex, give the worst performance in terms of fairness. Three types of AQM with respect to fair bandwidth allocation:
  – (1) AQM with no per-flow information,
  – (2) AQM with per-flow information and
  – (3) AQM with per-flow scheduling.
  – latter, of course will achieve the greatest level of fairness but will be the most complex. The second option is a compromise.
AQM performance – other factors: Fairness

1) **Fairness**: One of the earlier objectives of AQM was to provide fairness in the context of bursty traffic versus non-bursty traffic. Droptail queues unnecessarily penalized bursty traffic flows, therefore the very first AQM at the outset attempted to remove this bias by introducing queue averaging into the congestion indicator. Many debates as to the efficacy of this approach.

   - a) **UDP flows**: some applications, including realtime multimedia, streaming media and Voice-over-IP (VoIP), do not use TCP congestion control:
     - generate large amounts of traffic.
     - have stringent delay and jitter requirements which TCP could not provide
     - due to its retransmission mechanism and its windowing operation they contribute to traffic burstiness.
     - These applications might be unresponsive to any congestion indication (e.g., dropped packets) from the AQM, therefore they would not reduce their rates when there is congestion.
     - On the other hand, competing TCP flows, upon detecting congestion, would reduce their rates. This means that unresponsive flows would obtain a greater share of the available bandwidth and TCP flows would be penalized.
AQM performance – other factors: Fairness

– b) “TCP-friendly” flows:
  • In order to encourage new applications to adopt congestion control a number of source algorithms were proposed which are compatible with TCP, yet more conducive to the demands of multimedia and other streaming and real-time traffic, were developed.
  • If these were also ‘friendly’ to TTCP, they were termed ‘TCP-friendly’
  • Examples include Generalized AIMD (GAIMD), Binomial and TCP Friendly Rate Control (TFRC)
  • Even though in the long-term a flow using these protocols is expected to produce, on average, the same throughput as a flow using TCP, there were times when these protocols were more aggressive or more gentle than TCP due to the manner in which they reacted to congestion indications.

– c) Difference in round-trip time (RTT):
  • Even when flows use the very same endpoint congestion control implementation, there can be unfairness due to differences in their round-trip times.
  • For example, TCP flows with shorter round-trip times usually obtain more bandwidth than those with longer round-trip times. This is because they get their acknowledgments faster and can therefore ramp up their congestion window faster.
AQM performance – other factors: Fairness

- **d) Web traffic:**
  - Traditionally, AQM research focused primarily on long-lived TCP flows (e.g., FTP), however, the protection of short-lived flows such as Web transfers also important due to increasing web-traffic volumes and increasing sophistication of web-page content.
  - A typical web-page may initiate a separate connection for each embedded object, and each of these connections can be considered to be a short-lived flow which spends most of its time in slow-start. In slow-start, the transmission rates are sub-optimal due to small window sizes. Also, due to these small window sizes the susceptibility to retransmission timeouts is higher when there are losses. This is because there may not be enough packets in a window to successfully do a triple-duplicate-acknowledgment loss indication (which is faster).

- **e) TCP burstiness:**
  - This may be caused by the TCP congestion control algorithm behavior, as well as the cumulative ACK sent back to the source
f) The role of AQM with respect to fairness:

- An AQM that does not maintain any state information will have difficulty in maintaining fairness since there is no way for it to differentiate between responsive and unresponsive flows, conforming or greedy flows.
- Even so, some researchers suggest that an AQM with per-flow information using a single FIFO queue still cannot guarantee fair bandwidth allocation.
- Even though the AQM may drop packets from individual flows within the buffer to keep the buffer occupancy among the flows equal, the actual buffer output rate (i.e., bandwidth) for each flow may be unequal.
- Simple AQM designs may be difficult to find and also provide fairness; see however Lestas (will discuss later)
AQM performance – other factors: QoS

2) **QoS:** The Internet, by design, does not, and cannot, make any QoS commitments to users but serves them as best it can (‘best-effort service’). But increasingly applications (e.g. real time) demand QoS guarantees.

- Different applications using the internet have differing QoS requirements:. E.g. FTP and email cannot tolerate losses, can have relaxed delay requirements, but may require high throughputs. VoIP, video and other real-time and streaming applications have very tight delay and jitter requirements, yet they tend to be more packet loss resilient.

- It is required for the Internet to simultaneously support these different applications in an acceptable way, which it tolerably does at the moment. However, if strict guarantees are required, the the Internet may need to be re-engineered to meet their different QoS demands.

- Different approaches have been adopted, the most prominent being DiffServ (IntServ, which offered per-flow QoS, did not fly), in addition to some explicit rate based approaches (some from the ATM era).
AQM performance – other factors: QOS

• **a) DiffServ:** (already covered earlier in the course)
  
  – At the network edge, flows are tagged according to Service Level Agreements (SLAs) which define expected performance in terms of delay, jitter, loss and throughput.
  
  – In the core (or DS Domain), the tag on the packet is used to selectively forward, or even drop it (without any regard to what flow it belongs)
  
  – So instead of the more complex per-flow QoS, there is aggregate QoS (but still no guarantee, as it offers, at best, selectively ‘better’ behavior (in the limit, with the help of traffic without any priority it may attain the minimum required performance for each class of traffic). Specifically,
    
    • Each tag type is mapped unto a specific Per-Hop Behaviour (PHB) implemented by the router.
    
    • The Assured Forwarding (AF) is a standardized PHB which consists of four classes and three dropping probabilities within each class. If the actual sending rate of the user is below the minimum assured rate, packets are marked green; whereas if above the minimum assured rate, either yellow or red. A scheduler allocates bandwidth among the classes, whereas AQM enforces priorities within the class, so that during times of long-term congestion, green packets get the lowest drop rates, and red the highest. It is said that intra-class fairness depends on the AQM and inter-class fairness depends on the scheduler.
    
    • More later when we present Fuzzy Explicit Marking (FEM). See also RFC 3246, RFC 3260 of the Internet Engineering Task Force (IETF).
Diff-serv Architecture

**Edge router:**
- per-flow traffic management
- marks packets as in-profile and out-profile

**Core router:**
- per class traffic management
- buffering and scheduling based on marking at edge
- preference given to in-profile packets
AQM taxonomy (according to Richelle Adams, IEEE COMMUNICATIONS SURVEYS & TUTORIALS, to appear)

• different AQM schemes can be classified and compared in terms of their mechanisms of operation, including:
  – type of congestion indicator used,
  – manner in which their parameters are tuned
  – methods by which they perform flow differentiation if any
  – their control function
  – nature of their feedback signal to the source algorithms.

• They can also be classified according to their
  – context of use, whether it be in best-effort networks or networks with differentiated services, wired networks or wireless networks
  – performance of the AQM according to specified criteria (e.g., fairness, packet loss, throughput, delay, delay variation).
A congestion control taxonomy
Classification by AQM mechanisms

- AQM proposals have been
  - queue-based
  - rate-based
  - load-based
  - packet-loss-based
  - or a combination of these

- Mechanisms based on:
  - congestion indicator
    - instantaneous samples
    - the average (usually exponential weighted moving average (EWMA))
      some proposals updated the EWMA on each packet arrival while others performed the update at constant, predetermined intervals.
  - dynamic parameter configuration
    - static
    - time-varying network conditions such as changing network load (i.e., number of flows)
Classification by AQM mechanisms

- control function derived
  - heuristically
  - control theory
    - linear and nonlinear control
    - optimal control
    - robust control,
    - fuzzy logic, etc..
  - optimization
    - mostly deterministic
    - computational intelligence (e.g. neural, nature inspired, …)
    - none, or very few, use stochastic optimisation

- feedback signal used
  - explicit, e.g. ECN marking
  - implicit, e.g. packet dropping
    - decision to mark or drop a packet could be random or deterministic
AQM classification by mechanisms
AQM classified by mechanisms

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Congestion Indicator</td>
<td>Queue-based</td>
<td>Enqueue-events only</td>
</tr>
<tr>
<td></td>
<td>Rate-based</td>
<td>Arrival rate</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Congestion window size</td>
</tr>
<tr>
<td></td>
<td>Load-based</td>
<td>Flow count</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Traffic composition</td>
</tr>
<tr>
<td></td>
<td>Packet Loss</td>
<td>Loss volume</td>
</tr>
<tr>
<td></td>
<td></td>
<td>No. of buffer over/under flow events</td>
</tr>
<tr>
<td>Mixture</td>
<td>Load/Delay Controllers, Yellow</td>
<td></td>
</tr>
<tr>
<td>Parameter Tuning</td>
<td>Static</td>
<td>RED, GRED</td>
</tr>
<tr>
<td></td>
<td>Dynamic</td>
<td>Network load</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bandwidth-Distance Product (BDP)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Round-Trip Time (RTT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Link Capacity</td>
</tr>
<tr>
<td>Mixture</td>
<td>ARED, A-RIO, PSAND</td>
<td></td>
</tr>
<tr>
<td>Flow Differentiation</td>
<td>None</td>
<td>RED, GRED</td>
</tr>
<tr>
<td>Flow aggregates</td>
<td>Two-class system: Unresponsive versus TCP</td>
<td>Stochastic Fair Blue, RIO-C</td>
</tr>
<tr>
<td></td>
<td>Two-class system: Web vs FTP</td>
<td>SHREd</td>
</tr>
<tr>
<td></td>
<td>Multiple-class system</td>
<td>RIO-DC, WRED, RB-RIO, D-CBT, SFED</td>
</tr>
<tr>
<td>Individual flows</td>
<td>RED, BRED, BLACK</td>
<td></td>
</tr>
<tr>
<td>Control Function</td>
<td>Heuristic</td>
<td>Equation-based</td>
</tr>
<tr>
<td></td>
<td>Non-Equation based</td>
<td>MRED [Koo et. al. 2001]</td>
</tr>
<tr>
<td>Control Theoretic</td>
<td>Classical and Modern Control</td>
<td>PI, PD, Pd, Predictive PID, Adaptive GPC, PI-PD, PD-PD</td>
</tr>
<tr>
<td></td>
<td>Fuzzy Control</td>
<td>Fuzzy Control RED, Adaptive Fuzzy RED, FUZREM, Fuzzy Green, Deep Blue, PLCd, Fuzzy PID, Fuzzy CHOke</td>
</tr>
<tr>
<td>Optimization</td>
<td>Deterministic</td>
<td>REM, AVQ, SVB</td>
</tr>
<tr>
<td>Stochastic</td>
<td>Neural Networks</td>
<td>Neural-network-based P/I controller</td>
</tr>
<tr>
<td>Feedback Signal</td>
<td>Packet dropping</td>
<td>Random</td>
</tr>
<tr>
<td></td>
<td>Deterministic</td>
<td>AVQ</td>
</tr>
<tr>
<td>ECN Marking</td>
<td>Random</td>
<td>RED, REM, G</td>
</tr>
<tr>
<td>Deterministic</td>
<td>AVQ</td>
<td></td>
</tr>
</tbody>
</table>
Another AQM classification: context and performance
Congestion control in the Internet
Aside: AQM – history

• Traditional role of Active Queue Management (AQM) in IP networks was to complement the work of end-system protocols such as the Transmission Control Protocol (TCP) in congestion control so as to increase network utilization, and limit packet loss and delay.

• During the earlier days of IP networks, the network traffic consisted mainly of bulk data transfers. The volume of web traffic was gradually increasing.

• The first formal and full proposal of an AQM scheme was Random Early Detection (RED), introduced in 1993.

• What followed was a plethora of AQM schemes proposed in the research literature, many of which sought to improve upon the RED algorithm itself in one aspect or another. Additionally, there was also work that consisted primarily of a rigorous analysis of RED and which consequently highlighted its drawbacks.

• There were, however, AQM schemes (earlier and later than RED) that were completely new, and some very different in philosophy.
Aside: AQM – history: CC and QoS support

• With the increasingly rapid march to convergence, i.e., data, voice, video and mobility, supported by a common IP platform that is shared by a growing heterogeneous set of communication technologies there was a lot of development and argumentation as to a network architecture that could simultaneously and efficiently service the diverse requirements of the different types of traffic flows. The theme of quality-of-service (QoS) provisioning became very important, and that also affected proposed Congestion Control techniques.

• The role of AQM was to support QoS in the different architectures that were proposed, with IntServ and DiffServ being prominent at the time.

• DiffServ, which provides service differentiation survived, whereas IntServ, which could provide QoS to individual services (flows) remained shelved.
Aside: AQM – history: DiffServ

• DiffServ architecture works in conjunction with other QoS mechanisms such as traffic conditioning and packet scheduling so that their combined effect would be to, in an average sense, distinguish one network service from another in terms of overall end-to-end delay, delay variation or jitter, packet loss and bandwidth according to mutually agreed upon ‘service level agreements’ (SLAs)

• (DiffServ architecture already covered - see DiffServ slides)
Aside: AQM – history: DiffServ and CC approaches

- Based on the specifications for DiffServ, the early candidate AQM scheme for DiffServ was based on RED (specifically RIO (RED In/Out) having a different set of parameter values for each drop precedence.
- However, it is beneficial to capitalize on the vast AQM research that already exists, exploring those feasible alternative schemes and approaches that can be used in the DiffServ context so as to improve network performance and QoS.
- Fuzzy RED and FEM are two proposals we will briefly review later.
RED AND SOME OF ITS VARIANTS
Related Work for Best Effort

• Random early detection (RED)
  • first popular AQM mechanism proposed
  • a heuristic-based AQM technique
    – sets a marking probability between a min and max marking thresholds in router queues

If average queue between min and max
RED randomly marks packets based on a probability depending on average queue

If average queue exceeds max threshold, every packet is marked

Diagram:
- Current probability of packet mark
- Current average queue
- Marking probability range
- Thresholds for min and max queue
RED design

• The design of RED and many of its variants, though intuitive, has been, for the most part, heuristic.

• As a result, parameter-tuning has been one of their main limitations.

• Some researchers discovered that by applying more formal and rigorous techniques as found in control theory (whether it be classical control, modern control, optimal control or nonlinear control), this limitation may be alleviated if not eliminated.

• Other researchers have also invented AQM schemes based upon control and optimization techniques in the context of congestion control.
• RED Sally & Jacobson
• RED presentation by Konicki
• Fuzzy RED Chrysostomou, Pitsillides et al plus presentation
• Fuzzy Explicit Marking Chrysostomou, Pitsillides, et al plus presentation (see later)
EXPLICIT RATE APPROACHES: ACP, AN EXAMPLE
ACP: A congestion control protocol with learning capability

Model queue as a simple integrator

\[ \dot{q} = y - C \]

and set desired rate, \( p \)

\[ \dot{p} = \frac{1}{N} \left[ \frac{k_i}{d} (0.99 \ast C - y) - \frac{k_q}{d^2} q \right], \quad p(0) = p_0 \]

Divide by \( N \) required to ensure stability because of delays, as flows increase stability bounds decrease

\[ \hat{N}(k+1) = Pr[\hat{N}(k) + \gamma \frac{[y(k) - \hat{N}(k)p(k)]p(k)}{1 + p^2(k)}], \quad \hat{N}(0) = 10 \]

In discrete form

\[ p(k+1) = Pr[p(k) + \frac{1}{\hat{N}(k)} [k_i(0.99 \ast C - y(k)) - \frac{1}{d(k)} k_q q(k)]] \]

Effective number of users

each link updates desired sending rate every control period equal to average RTT

\( K_i, k_q \) design constants

Integrate excess capacity

Proven global stability for max-min, (but ignoring delays and queue)

To avoid queue build-up and allow for statistical fluctuations

queue length factor in order to ensure that at equilibrium queue converges to zero.

to maintain stability in the presence of delays decrease feedback gain as delays increase

- divide excess capacity term with propagation delay
- divide queue size term with square of propagation delay
ACP: A congestion control protocol with learning capability

Stability results:
For the given design constants $K_i, K_q$, all trajectories for time delays in the range 100-600msec converge to origin.

Comparative evaluation

Competing with difficult traffic

goodput of the long and short flows for ACP, XCP and TCP. At all traffic loads, ACP achieves higher goodput than the others.
Estimating effective number of users at a bottlenecked link

• Many CC protocols recently proposed for high speed networks
  – require estimates of number of users utilizing each link in network to maintain stability in presence of delays

• propose novel estimation algorithm based on online parameter identification techniques
  – shown through analysis and simulations to converge to effective number of users utilizing each link
  – does not require maintenance of per flow states within network or per packet processing
  – outperforms previous proposals based on point wise division in time
  – estimation scheme designed independently from control functions of protocols and is thus universal in sense that it operates effectively in a number of CC protocols

• proposed estimation scheme used to design 3 representative Internet CC protocols: ACP, DMM and QLB
  – demonstrate using simulations satisfy key design requirements.
  • stable equilibrium characterized by high network utilization, small queue sizes and max-min fairness
  • scalable with respect to changing bandwidths, delays and number of users
  • generate smooth responses which converge fast to the desired equilibrium
Theorem 1: At each link $j = \{1, 2, \ldots, m\}$, if $N_j^*(k) = N_j^*$, \[ \forall k, \lim_{k \to \infty} \hat{N}_j(k) = N_j^* \text{ and } N_j^* \epsilon D_j, \] where: \[ D_j = \{ \hat{N}_j \epsilon R | N_j \leq \hat{N}_j \leq n_j \} \]

number of users changes from 30 to 10, at 30 seconds and from 10 to 50 at 45 seconds.

Fig. 4. Parking lot topology used to investigate the ability of the estimation algorithm to calculate the effective number of users utilizing each link in the network.

Link 4 is the bottleneck link, where 40 users share it. In all other links the effective number of users is equal to 27.
DIFFSERV AND CC
The DiffServ architecture
• Fuzzy Explicit Marking Chrysostomou, Pitsillides, et al

• plus presentation
CC IN WIRELESS NETWORKS
Location of AQM in infrastructure-type wireless access networks
PERFORMANCE EVALUATION
## AQM Performance Evaluation Framework

### Performance Criteria

<table>
<thead>
<tr>
<th>Performance Metrics</th>
<th>Standardized Definitions</th>
<th>Standardized Measurement Methodology</th>
<th>Standardized Statistics for Reporting</th>
<th>Standardized Application Dependent Thresholds</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fairness</td>
<td>?</td>
<td></td>
<td>Mean, standard deviation, minimum, maximum percentiles, empirical distributions</td>
<td>?</td>
</tr>
<tr>
<td>Delay</td>
<td>e.g., IETF IP Performance Metrics (IPPM) Framework</td>
<td></td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>Jitter</td>
<td>?</td>
<td></td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>Packet Loss</td>
<td>?</td>
<td></td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>Application throughput/ goodput</td>
<td>?</td>
<td></td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>Link Utilization</td>
<td>?</td>
<td></td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>Queue stability</td>
<td>?</td>
<td></td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>Responsiveness</td>
<td>?</td>
<td></td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>Robustness</td>
<td>?</td>
<td></td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>Complexity</td>
<td>?</td>
<td></td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>

### Network Parameters that Impact AQM Performance

- Traffic load
- Traffic patterns and stochastic nature
- Link capacity
- Round-Trip-Time (RTT)
- Network topology and routing
- Reverse-path asymmetry
- Buffer size
- Phase relationship among flows
- Presence of short-lived flows
- Presence of unresponsive flows

### Choice of Traffic Types and Traffic Mixes with Standardized Models

- Standardized Scenarios for Performance Evaluation
- Standardized Topologies for Wired and Wireless Testing
- Simulator Choice; Testbed (Equipment) Configuration
- Multidimensional Sensitivity Analysis w.r.t. AQM Parameters
- Common, Online Database for AQM Comparison
- Common Format for Algorithm Depiction
Selected own papers


• C. Chrysostomou, A. Pitsillides, G. Hadjipollas, M. Polycarpou, A. Sekercioglu, Fuzzy Logic Control for Active Queue Management in TCP/IP Networks, IEEE 12th Mediterranean Conference on Control and Automation (MED’04), Kusadasi, Aydin, Turkey, 6-9 June 2004 (CD ROM Proceedings), (6 pages).


• Chrysostomos Chrysostomou (Ph.D.), Fuzzy Logic based AQM Congestion Control in TCP/IP networks, Thesis was defended in 2006. (thesis advisor; A. Pitsillides)


Fuzzy slides
Fuzzy Logic based AQM Congestion Control in TCP/IP Networks


‘So far as the laws of mathematics refer to reality, they are not certain. And so far as they are certain, they do not refer to reality.’

Albert Einstein
‘As complexity rises, precise statements lose meaning and meaningful statements lose precision.’

Lotfi Zadeh (‘father’ of Fuzzy Logic)
Precision and significance in real world

A 1500 kg mass is approaching your head at 45.3 m/s

Precision

LOOK OUT!!

Significance
Our FLC-based Research Study

• investigated the complex, but challenging, concepts of
  – ATM QoS aware flow control
  – TCP/AQM (Active Queue Management) CC
    • BE (best-effort)
    • Diff-Serv (differentiated services) environments

• FOCUS OF PRESENTATION IS TCP/AQM CC
Problem Statement

• **Main Aim**: provide effective control for high link utilization with low loss and queuing delay
  – focus on AQM mechanisms with ECN support, thus keeping TCP’s CC mechanisms unchanged
    • Internet standards track protocol RFC3168
    • most routers support ECN
  – furthermore, address **Diff-Serv congestion control** at core for aggregated QoS support
Motivation

• **Current CC solutions based on AQM/ECN are ineffective to meet diverse needs of today’s Internet**
  – they have serious limitations and drawbacks, as identified in literature

• **Extremely difficult for traditional modeling techniques to capture the network’s essential dynamics**
  – even if they do resulting model is overly complex

• **Common approach in classical control theory is to**
  – ignore such complex parameters in mathematical model
  – simplify model, often making overly conservative with restrictive stability bounds

• **Given need for effective control methodology**
  – to capture system behaviour under widely differing operating conditions
  • investigate usefulness of **fuzzy logic control** to meet such objectives
Why Use Fuzzy Logic Control

• Fuzzy Logic Control
  – particularly appealing in nonlinear complex systems where
  • satisfactory analytic models are impractical
  • their behavior is well understood and can be captured by linguistic models

• Has solid theoretical foundation (at times controversial)
  – achieves ‘inherent’ robustness and reduces design complexity

• Part of what is termed Intelligent Control, or Computational Intelligence Control

• Remarkable success demonstrated in research literature and commercial products in many diverse disciplines
Contributions

• Offer significant improvements
  – achieve high link utilization and regulated queues
  – fast system response and robustness to varying system dynamics (differing topologies and traffic conditions)
  – in Dif-Serv, adequate and effective differentiation among different priority classes

• Demonstrate that Fuzzy Logic based AQM control methodology better handles nonlinearities and dynamics, in contrast with existing, well-known, conventional counterparts
Proposed Mechanisms: BE

• Proposed AQM scheme for BE (Best-Effort) environments
  – Fuzzy Explicit Marking (FEM)
    • regulates queues of IP routers at predefined levels
      – by achieving a specified target queue length (TQL)
    • in same spirit as RED
    • Fuzzy inference engine (FIE) operates on router buffer queues
Proposed Mechanisms: BE (cnt’d)

• Feedback system model of FEM

- Based on two network state inputs
- Error on instantaneous queue length for two consecutive sampling intervals

Like RED FEM, dynamically calculates mark probability $p(kT)$
Proposed Mechanisms: BE (cnt’d)

- System model of FEM

**surface structure**

**Table 6.1 FEM Linguistic rules – Rule base**

<table>
<thead>
<tr>
<th>( p(kT) )</th>
<th>( Q_{error}(kT-T) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>NVB</td>
<td>H</td>
</tr>
<tr>
<td>NB</td>
<td>B</td>
</tr>
<tr>
<td>NS</td>
<td>T</td>
</tr>
<tr>
<td>PS</td>
<td>Z</td>
</tr>
<tr>
<td>PB</td>
<td>Z</td>
</tr>
<tr>
<td>PVB</td>
<td>Z</td>
</tr>
</tbody>
</table>

Knowledge-base (linguistic rules) generated from IF-THEN control rules, e.g.:

- IF \( e(kT) \) is NVB AND \( e(kT-T) \) is NB, THEN \( p(kT) \) is H
- IF \( e(kT) \) is PVB AND \( e(kT-T) \) is PB, THEN \( p(kT) \) is Z

**deep structure**

Membership functions

Linguistic values

Mark probability responsive due to human reasoning and inbuilt nonlinearity
Proposed Mechanisms: Diff-Serv

- **Diff-Serv Fuzzy Logic Control Design (FIO)**
  - **goal** to achieve *same performance as BE*
    - provide effective congestion for Diff-Serv, plus
    - differentiated treatment of traffic aggregates
  - built on fuzzy controller designed for BE environments
  - **two identical FEM controllers** used
    - one for each differentiated traffic aggregate *(FIO – FEM In-Out)*
    - **high-priority** (low drop precedence / In packets)
    - **low-priority** (high drop precedence / Out packets)
  hence offering (differentiated) QoS in traffic aggregates
Proposed Mechanisms: Diff-Serv (cnt’d)

• Two different TQLs, one for each FEM controller

TQL for low-priority < TQL for high-priority

- objective: regulate queue at TQL$_{low}$, where mark probability for high-priority traffic is closer to zero

- **but**, if high-priority traffic >> low-priority traffic, at least regulate queue at TQL$_{high}$
  - not enough low-priority traffic to maintain TQL$_{low}$
  - in this case mark probability for low-priority traffic is closer to 1

- accomplish both differentiation and bounded delay, by regulating queue between two TQLs, depending on network traffic
Simulative Performance Evaluation

- Use extensive simulative evaluation to demonstrate effectiveness and robustness, in both BE and Diff-Serv environments
- Comparison made with other published results with well-known, AQM schemes
  - A-RED, PI, REM, and AVQ for BE networks
  - RIO and TL-PI for Diff-Serv networks
- Performance of AQM schemes evaluated using most widely used simulator, NS-2, in different topologies and wide varying network conditions
- In all cases, our approach outperforms all others in all scenarios and network conditions

additional results FEM, FIO
FEM Evaluation

Effect of traffic load (increase flows from 100-500) provides some time-varying dynamics and scalability. Single-bottleneck link (TQL = 200 packets ~ 100 msec)

- FEM outperforms other AQM schemes
  - high link utilization, low delay and delay variation
  - exhibits more stable, robust behavior with bounded delay

- lowest drops, over large traffic load

Figure 7.10 Scenario I-5: Loss Rate vs Traffic Load (for 100-500 flows)

Figure 7.11 Scenario I-5: Utilization vs Mean Delay (for 100-500 flows)

Figure 7.12 Scenario I-5: Utilization vs Delay Variation (for 100-500 flows)
FEM Evaluation (cnt’d)

queue length evolution: sudden change in traffic conditions

600 users at t=0
300 leave network t=40
300 re-enter network t=70

multiple bottleneck links
Diff-Serv FIO Evaluation

- Single-bottleneck link
  - Effect of increasing high-priority traffic
    - increases from 2%, 10%, and 90% of total traffic
  - FIO outperforms other schemes with **better differentiation** provided in favor of the high priority traffic

Figure 9.4  Scenarios 1-1-3: Utilization of high-priority traffic vs percentage of high-priority traffic
(high-priority traffic increases from 2%, 10%, and 90% of the total traffic)
Diff-Serv FIO Evaluation (cnt’d)

- Single-bottleneck link
  - Effect of increase of high-priority traffic – (2% of total traffic) on queue behaviour

FIO regulates its queue to $TQL_{\text{low}}$

TL-PI tries to regulate the queue at $TQL_{\text{low}}$ however slow and large delay variation

RIO cannot regulate at all the queue and exhibits very large queue fluctuations

$TQL_{\text{low}} = 100$
$TQL_{\text{high}} = 200$

Figure 9.1 Scenario 1-1 Queue lengths
Diff-Serv FIO Evaluation (cnt’d)

- Single-bottleneck link
  - Effect of increase of high-priority traffic – cnt’d (10% of total traffic)

FIO accomplishes a bounded delay, between the two TQLs

\[ TQL_{low} = 100 \]
\[ TQL_{high} = 200 \]

TL-PI and RIO slowly regulate their queue, after a significant transient period

Figure 9.2 Scenario 1-2 Queue lengths
• Single-bottleneck link
  – Effect of increase of high-priority traffic – cnt’d (90% of total traffic)

FIO regulates its queue at $TQL_{\text{high}}$

RIO exhibits large queue fluctuations

$TQL_{\text{low}} = 100$
$TQL_{\text{high}} = 200$

TL-PI shows a sluggish response and less tight control than FIO

Figure 9.3 Scenario 1-3: Queue lengths
Diff-Serv FIO Evaluation (cnt’d)

- Single-bottleneck link, 100 flows (10% of total traffic is high-priority)
  - Effect of time-varying dynamics (between t=40sec – 70sec, only low-priority traffic is active, i.e 90 flows)

FIO is very robust against the dynamic traffic changes, successfully manages to regulate queue length at TQL for low-priority, between t=40-70sec

RIO fails to regulate queue

TL-PI slow response to regulate queue

FIO

TL-PI

RIO

Figure 9.5 Scenario 1-4: Queue lengths

TQL_{low} = 100
TQL_{high} = 200
Concluding Remarks

• Due to complexity of dynamic network parameters, FLC, a robust intelligent control methodology, is adopted to effectively control the network system under widely varying operating conditions and dynamic changes
  – provide effective CC and QoS support within BE, and furthermore effective differentiation for Diff-Serv environments
  • FIO uses same controller as BE (FEM) for both priority classes
  • no retuning needed
Supplementary Slides for fuzzy
Proposed Mechanisms: BE

• **System model of FEM**
  – Adopted most widely used, simplest MISO FLC (Mamdani-based)
    • avoid exponential increase of rule base and increasing controller complexity
  – Design of rule-base is done based on
    • **Completeness**: all kinds of situations of system behaviour are taken into consideration
    • **Consistency**: rule-base does not contain any contradiction
  – **Philosophy** behind knowledge-base of FEM controller is
    • being **aggressive** when queue length ‘largely’ deviates from TQL
    • smoothly respond when queue length is **around** TQL
    • All other rules represent intermediate situations, thus providing the control mechanism with a highly dynamic action
Related Work for Best Effort

- Random early detection (RED)
  - first popular AQM mechanism proposed
  - a heuristic-based AQM technique
    - sets some min and max marking thresholds in router queues

If average queue between min and max
RED randomly marks packets based on a
probability depending on average queue

If average queue exceeds max
threshold, every packet is marked

![Diagram showing the relationship between current average queue and packet mark probability](image-url)
System model of FEM (cnt’d)

- $e(kT)$ is the error on the controlled variable queue length, $q(kT)$, at each sampling interval $kT$

  $$e(kT) = q_{des} - q(kT)$$

- $e(kT-T)$ is the error on queue length with a delay $T$ (at the previous sampling interval)
System model of FEM (cnt’d)

- $SG_i$ are the input scaling gains, determining the range of values (universe of discourse) for a given controller input.

- For greater flexibility, and generality, the universe of discourse for each input is “normalized” to the interval [-1, +1], by means of constant scaling factors.

\[
SG_i = \begin{cases} 
\frac{1}{q_{des} - BufferSize}, & q_{inst} > q_{des} \\
1, & \text{otherwise}
\end{cases}
\]
System model of FEM (cnt’d)

- $SG_0$, the output scaling gain, is determined so that the range of outputs that is possible is the maximum, but also ensuring that the input to the plant will not saturate around the maximum.
- Following the approach of Floyd, Gummadi, and Shenker (2001) $SG_0$ is dynamically set to a value indicating the maximum mark probability (initially keep $SG_0$ to 10%)
  
  $IF q(kT) > 1.1^{*}TQL \ then\ increase\ SG_0\ by\ 0.01$

  $IF q(kT) < 0.9^{*}TQL \ then\ decrease\ SG_0\ by\ 0.9$

The dynamic selection of $SG_0$ based on formal adaptive control theory is a subject of future research.
Illustrative Example of FEM

For each input each rule visited

minimum membership values of inputs to corresponding linguistic values found

queue-error = -0.3

previous-queue-error = -0.08

For each corresponding output (for each rule)

linguistic value truncated (using the min operator)

using center-of-gravity defuzzification method, compute a numerical value for controller’s output

take aggregated of all four implied fuzzy sets using the max operator

center-of-gravity

mark-probability = 0.67

min

max
Related Work for Best Effort

- Adaptive-RED (A-RED)
  - enhancement of RED
    - Dynamically adjusts maximum \textit{mark} probability to keep average queue length half way between min and max thresholds

- Proportional-Integral (PI) controller
  - A linear control theory based AQM technique
    - uses classical control theory techniques to stabilize router queue length around a target value

- Random exponential marking (REM)
  - An exponentially increasing based probability law
    - uses instantaneous queue size and its difference from a target value to calculate mark probability based on an exponential law

- Adaptive Virtual Queue (AVQ)
  - Virtual queue-based dropping scheme
    - uses a virtual queue to regulate link utilization, rather than queue length
Related Work for Diff-Serv

• Aim to preferentially drop/mark low-priority packets against high-priority packets, during congestion
  – RED In/Out (RIO)
    • most popular implementation based on RED

  – Two-level PI controller (TL-PI)
    • introduces two target queue lengths, which correspond to two levels of drop precedence

  For “In” packet, RIO uses average queue length of “In” packets
  For “Out” packet, RIO uses total average queue length
**Diff-serv Architecture**

**Edge router:**
- per-flow traffic management
- marks packets as in-profile and out-profile

**Core router:**
- per class traffic management
- buffering and scheduling based on marking at edge
- preference given to in-profile packets
Simulative Evaluation (cnt’d)

• examine effects on AQM schemes, of **wide varying network conditions**:
  – dynamic traffic changes – time-varying dynamics
  – traffic load factor
  – heterogeneous propagation delays and different propagation delays at bottleneck links
  – different link capacities
  – introduction of noise-disturbance (background traffic) to the network (e.g. short-lived TCP connections)
  – introduction of reverse-path traffic
  – different types of competing data streams, like TCP/FTP and TCP/Web traffic, as well as unresponsive traffic (UDP-like)
RESOURCE ALLOCATION – FAIRNESS CRITERION
Issues in Resource Allocation

• Evaluation Criteria
  – Effective Resource Allocation
    • A good starting point for evaluating the effectiveness of a resource allocation scheme is to consider the two principal metrics of networking: throughput and delay.
      – Clearly, we want as much throughput and as little delay as possible.
      – Unfortunately, these goals are often somewhat at odds with each other.
      – One sure way for a resource allocation algorithm to increase throughput is to allow as many packets into the network as possible, so as to drive the utilization of all the links up to 100%.
      – We would do this to avoid the possibility of a link becoming idle because an idle link necessarily hurts throughput.
      – The problem with this strategy is that increasing the number of packets in the network also increases the length of the queues at each router. Longer queues, in turn, mean packets are delayed longer in the network.
**Issues in Resource Allocation**

- **Evaluation Criteria**
  - **Effective Resource Allocation**
    - To describe this relationship, some network designers have proposed using the ratio of throughput to delay as a metric for evaluating the effectiveness of a resource allocation scheme.
    - This ratio is sometimes referred to as the *power of the network*.
    - Power = Throughput/Delay
Issues in Resource Allocation

• Evaluation Criteria
  – Effective Resource Allocation

Ratio of throughput to delay as a function of load
Issues in Resource Allocation

• Evaluation Criteria
  – Fair Resource Allocation
    • The effective utilization of network resources is not the only criterion for judging a resource allocation scheme.
    • We must also consider the issue of fairness. However, we quickly get into murky waters when we try to define what exactly constitutes fair resource allocation.
    • For example, a reservation-based resource allocation scheme provides an explicit way to create controlled unfairness.
    • With such a scheme, we might use reservations to enable a video stream to receive 1 Mbps across some link while a file transfer receives only 10 Kbps over the same link.
Issues in Resource Allocation

• Evaluation Criteria
  – Fair Resource Allocation
    • In the absence of explicit information to the contrary, when several flows share a particular link, we would like for each flow to receive an equal share of the bandwidth.
    • This definition presumes that a *fair share of bandwidth means an equal share of bandwidth*.
    • But even in the absence of reservations, equal shares may not equate to fair shares.
    • Should we also consider the length of the paths being compared?
      – One four-hop flow competing with three one-hop flows
• Evaluation Criteria
  – Fair Resource Allocation
    • Assuming that fair implies equal and that all paths are of equal length, networking researcher Raj Jain proposed a metric that can be used to quantify the fairness of a congestion-control mechanism.
      – Jain’s fairness index is defined as follows. Given a set of flow throughputs \((x_1, x_2, \ldots, x_n)\) (measured in consistent units such as bits/second), the following function assigns a fairness index to the flows:

\[
f(x_1, x_2, \ldots, x_n) = \frac{\left(\sum_{i=1}^{n} x_i\right)^2}{n \sum_{i=1}^{n} x_i^2}
\]

– The fairness index always results in a number between 0 and 1, with 1 representing greatest fairness.