L2: Congestion Control

- When one part of the subnet (e.g. one or more routers in an area) becomes overloaded, congestion results.
- Because routers are receiving packets faster than they can forward them, one of two things must happen:
  - The subnet must prevent additional packets from entering the congested region until those already present can be processed.
  - The congested routers can discard queued packets to make room for those that are arriving.

- http://www.cisco.com/en/US/products/ps6558/products_ios_technology_home.html
- http://www.cisco.com/en/US/netsol/ns577/networking_solutions_white_paper09186a00801eb831.shtml
- http://www.cisco.com/univercd/cc/td/doc/cisintwk/ito_doc/qos.htm

Traffic conditioner

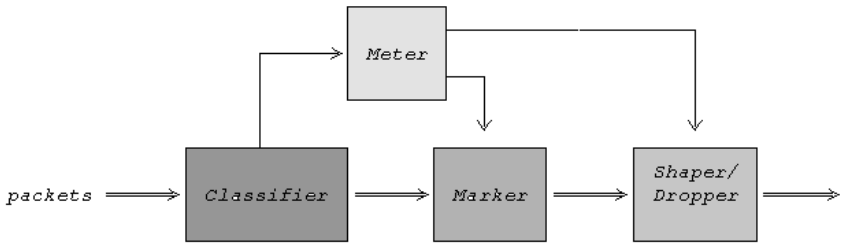


Figure 1.4.1

- A traffic stream is selected by a classifier, which steers the packets to a logical instance of a traffic conditioner.
- A meter is used (where appropriate) to measure the traffic stream against a traffic profile.
- The state of the meter with respect to a particular packet (e.g., whether it is in- or out-of-profile) may be used to affect a marking, dropping, or shaping action.

Factors that Cause Congestion

- Packet arrival rate exceeds the outgoing link capacity.
- Insufficient memory to store arriving packets
- Bursty traffic
- Slow processor

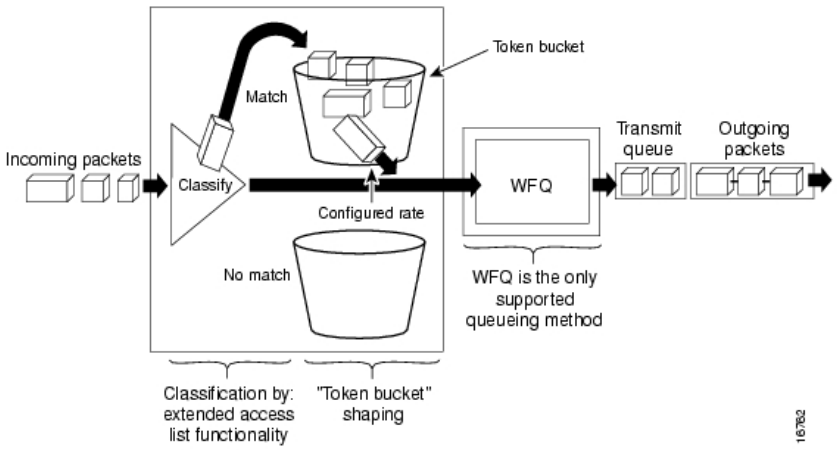
Congestion Control vs Flow Control

- Congestion control is a global issue – involves every router and host within the subnet
- Flow control – scope is point-to-point; involves just sender and receiver.

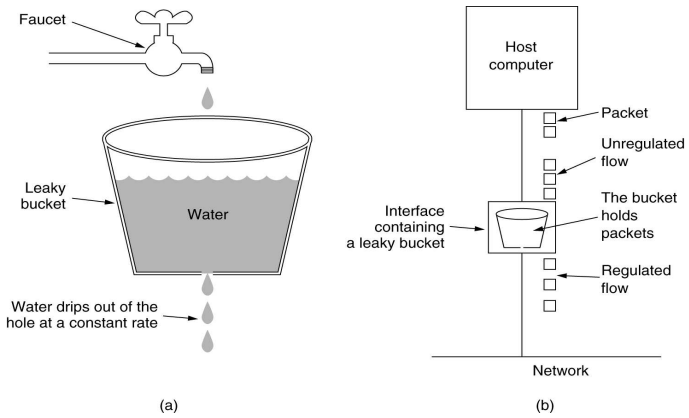
Traffic Shaping

- Another method of congestion control is to “shape” the traffic before it enters the network.
- Traffic shaping controls the *rate* at which packets are sent (not just how many). Used in ATM and Integrated Services networks.
- At connection set-up time, the sender and carrier negotiate a traffic pattern (shape).
- Two traffic shaping algorithms are:
  - Leaky Bucket
  - Token Bucket

General Traffic Shaper



The Leaky Bucket Algorithm



- The **Leaky Bucket Algorithm** used to control rate in a network. It is implemented as a single-server queue with constant service time.
- If the bucket (buffer) overflows then packets are discarded.

http://www.cisco.com/en/US/products/hw/switches/ps1893/products_feature_guide_chapter09186a008007e39e.html

Leaky Bucket Algorithm

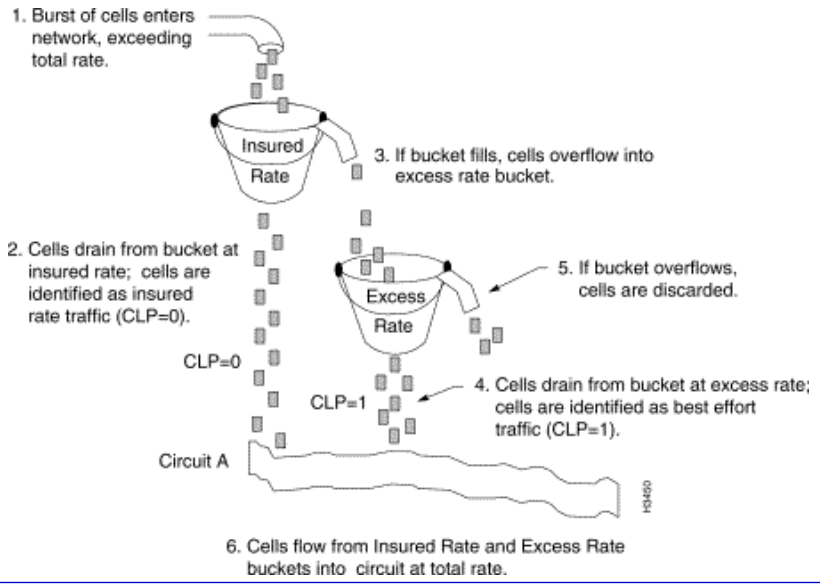
- The leaky bucket algorithm uses two parameters to control traffic flow:
  - **Average rate**: The average number of cells per second that "leak" from the hole in the bottom of the bucket and enter the network.
  - **Burst rate**: The rate at which cells are allowed to accumulate in the bucket, expressed in cells per second. For example, if the average burst rate is 10 cells per second, a burst of 10 seconds allows 100 cells to accumulate in the bucket.
- The leaky bucket algorithm also uses two state variables:
  - **Current time**: The current wall clock time.
  - **Virtual time**: A measure of how much data has accumulated in the bucket, expressed in seconds.
    - For example, if the average rate is 10 cells per second and 100 cells have accumulated in the bucket, then the virtual time is 10 seconds ahead of the current time.

Leaky Bucket Algorithm

- If, for example, the average rate is 10 cells per second, and the burst is 50 cells, the virtual time and current time remain the same as long as the input rate remains at or below 10 cells per second.
- If an instantaneous burst of 25 cells is received, the virtual time moves ahead of the current time by 2.5 seconds. If this is followed immediately by a second burst of 30 cells, the virtual time moves ahead of the current time by 5 cells, and the last 5 of the 30 cells are dropped.
- For packet traffic, the unit of incoming data is larger than a single ATM cell. For packet interfaces, the leaky bucket algorithm takes the packet size into account, as shown in the following formula:

```
virtual time = max (virtual time, current time)
        if (virtual time + (packet size / average rate) > current time + burst)
        drop the incoming packet
else
        segment the packet into cells
        put the cells in the bucket
        virtual time = virtual time + (packet size/average rate)
```

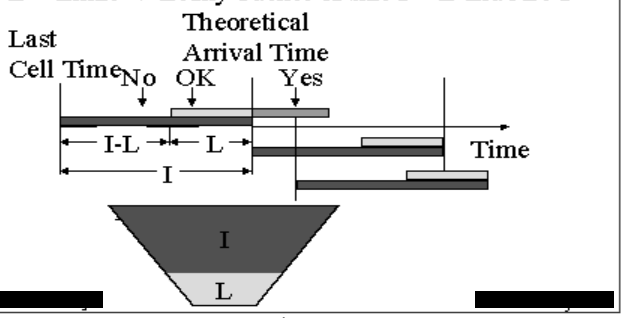
Leaky Bucket Algorithm



GCRA

Generic Cell Rate Algorithm: GCRA(I, L)

- I = Increment = Inter-cell Time = Cell size/PCR
- L = Limit ⇒ Leaky bucket of size I + L and rate 1

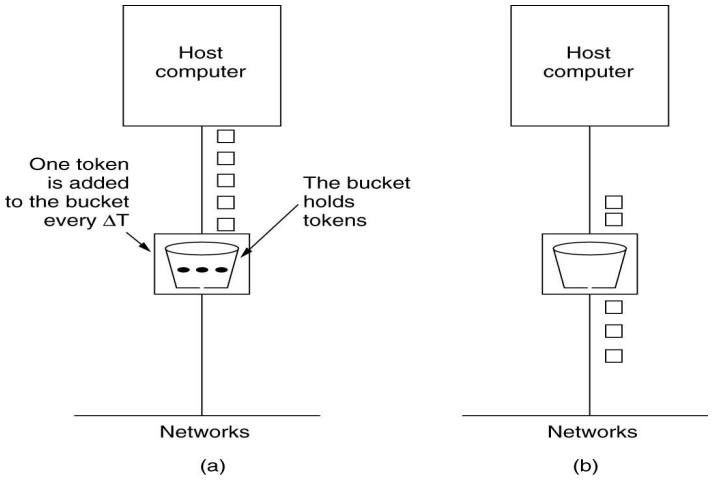


4

Token Bucket Algorithm

- In contrast to the LB, the Token Bucket Algorithm, allows the output rate to vary, depending on the size of the burst.
- In the TB algorithm, the bucket holds tokens. To transmit a packet, the host must capture and destroy one token.
- Tokens are generated by a clock at the rate of one token every Δt sec.
- Idle hosts can capture and save up tokens (up to the max. size of the bucket) in order to send larger bursts later.
- Example from: http://www.opalsoft.net/qos/CDS-22-A1.htm

The Token Bucket Algorithm



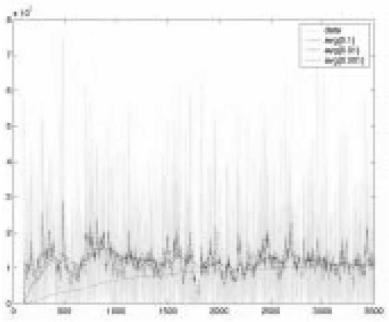
(a) Before. (b) After.

Leaky Bucket vs Token Bucket

- LB discards packets; TB does not. TB discards tokens.
- With TB, a packet can only be transmitted if there are enough tokens to cover its length in bytes.
- LB sends packets at an average rate. TB allows for large bursts to be sent faster by speeding up the output.
- TB allows saving up tokens (permissions) to send large bursts. LB does not allow saving.

Packet by packet EWMA meter

- Measures packet stream by using exponentially weighted moving average filter.
  - **Tunable by parameter**
    - Memory (ϵ)
- Example: for a packet arriving at time t: if (avg_rate(t) > AverageRate) non-conforming else conforming



Initial condition:

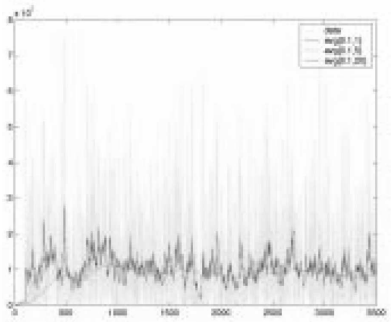
$avg(0) = 0$

After every packet arrival

$$avg(n+1) = (1-\epsilon)\cdot avg(n) + \epsilon \cdot \frac{PacketLength}{t_{n+1} - t_n}$$

Windowed EWMA meter

- Measures packet stream by using exponentially weighted moving average filter with sampling window.
  - **Tunable by parameters**
    - Memory (ϵ)
    - Sampling interval (ΔT)



Initial condition:

$avg(0) = 0$

After every ΔT time units

$$avg(t_{n+1}) = (1-\epsilon)\cdot avg(t_n) + \epsilon \cdot bytes\ during [t_{n+1}, t_n]$$

Time Sliding Window Meter

- TSW is memory based, windowed average rate estimator
- **Tunable by parameter**
  - Window length

Initial condition:

$avg(0) = 0$

$Win_{length} = C$

$T_{front} = 0$

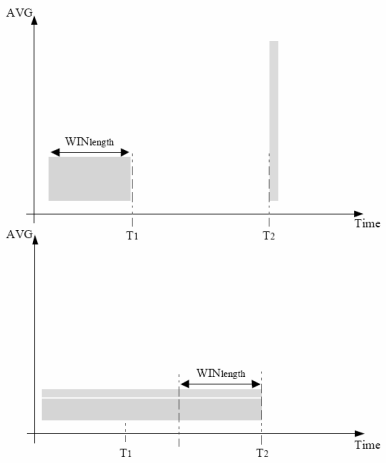
After every packet arrival:

$Bytes_{TSW} = avg(n) \cdot Win_{length}$

$New_{bytes} = Bytes_{TSW} + PacketLength$

$$avg(n+1) = \frac{New_{bytes}}{T_{now} - T_{front} + Win_{leght}}$$

$T_{front} = T_{now}$



Metering

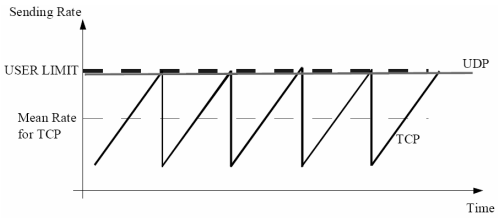
- Based on the measured information a conformance statement is declared
- **Conformance is the observation whether the measured variable is within predefined boundaries.**
  - Customer has contracted rate of ***X*** bps with variation of ***x*** bps
  - Customer has contract of average rate ***X*** bps and peak of ***Y*** bps. He is allowed to send bursts of ***Z*** kB in peak rate.

Conformance Algorithms

- **Strict conformance**
  - Packets exceeding contracted rate are marked immediately as non−conforming
- **TSW conformance**
  - Packets exceeding 1.33 times contracted rate are marked as non−conforming
- **Probability conformance**
  - Packets exceeding contracted rate are marked as non−conforming with increasing probability

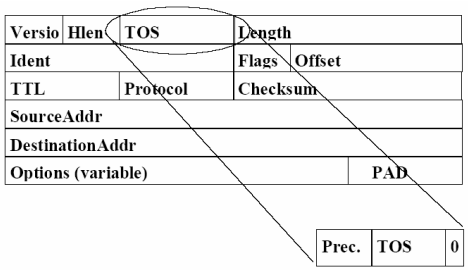
Rate Control Problems

- Two parallel transport protocols with contradicting control:
  - UDP − with no control
  - TCP − with additive increase exponential decrease rate control
- **Problem:** Metering system cannot easily offer fair service to both TCP and UDP clients in the same system.



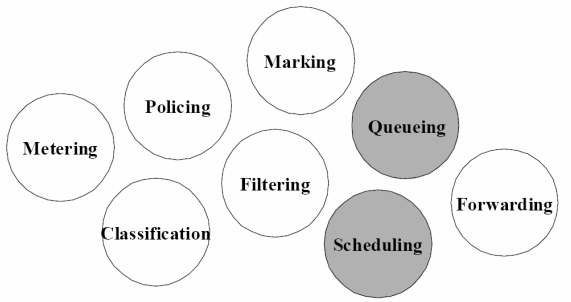
Marking

- Marker is used to attach conformance / class information to every packet.
- Marker uses IPv4 TOS/DSCP field to convey information for other processing elements in the network.
  - TOS
    - Prec: 3 bit priority
    - TOS: user preference for routing
  - DSCP
    - Class and precedence



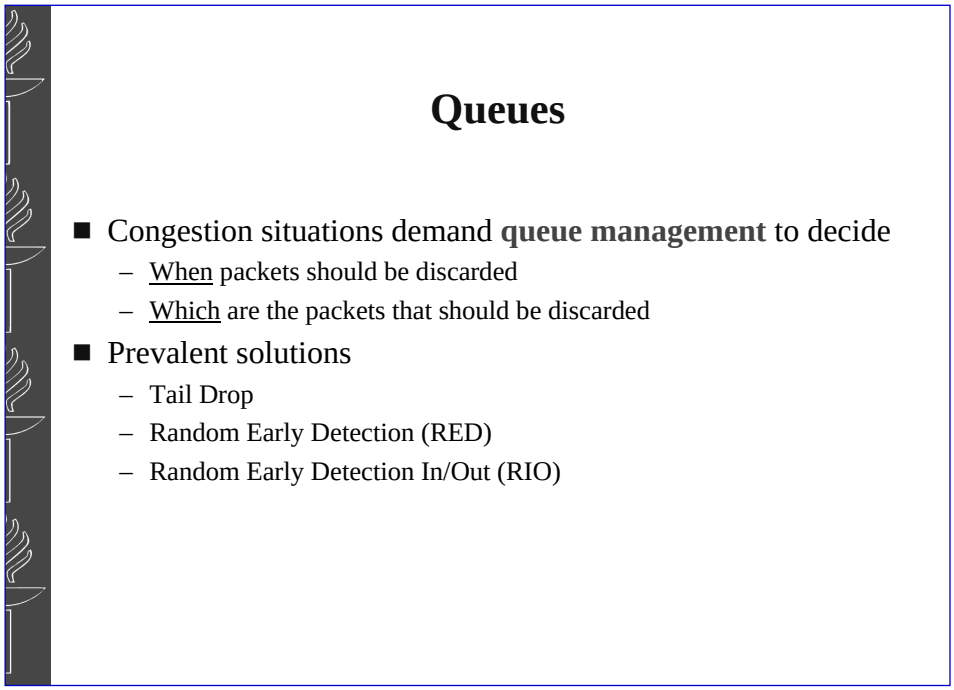
Output Processor

- Output processor of a Internet router consists following elements
  - Queues and their management algorithms
  - Scheduling



Queues

- Queues are used to store **contending** packets
  - Contention is temporary event rising from statistical multiplexing
    - Packets from different input links of a router attempt to the same output link at certain time
    - Packets from a higher speed link arrive temporarily too fast for a slow speed link
- If contention is permanent queues overflow i.e. network is **congested**
- **Difference:**
  - Contention − packets are not lost only delayed
  - Congestion − packets are not only delayed but also lost



Queues

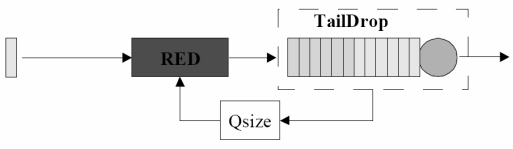
- Congestion situations demand **queue management** to decide
  - When packets should be discarded
  - Which are the packets that should be discarded
- Prevalent solutions
  - Tail Drop
  - Random Early Detection (RED)
  - Random Early Detection In/Out (RIO)

Tail Drop

- Simple algorithm:
  - If arriving packets sees a full queue it is discarded
  - Otherwise it is accepted to the queue
- Problem:
  - Poor fairness in distribution of buffer space
  - Unable to accommodate short transients when queue is almost full
    - Bursty discarding leading global synchronization
- Global synchronization is a process where large number of TCP connections synchronize their window control due to concurrent packet losses.
  - Packet losses are bursty, therefore window decreases to one and halts the communication

Random Early Detection

- RED is an active queue management algorithm (AQM), which aims to
  - Prevent global synchronization
  - Offer better fairness among competing connections
  - Allow transient burst without packet loss
- Algorithm operates on the knowledge of current Qsize
  - Updated on every arrival and departure from the actual queue



RED

- http://www.acm.org/crossroads/columns/connector/july2001.html
- Qsize is used to calculate average length of the queue:

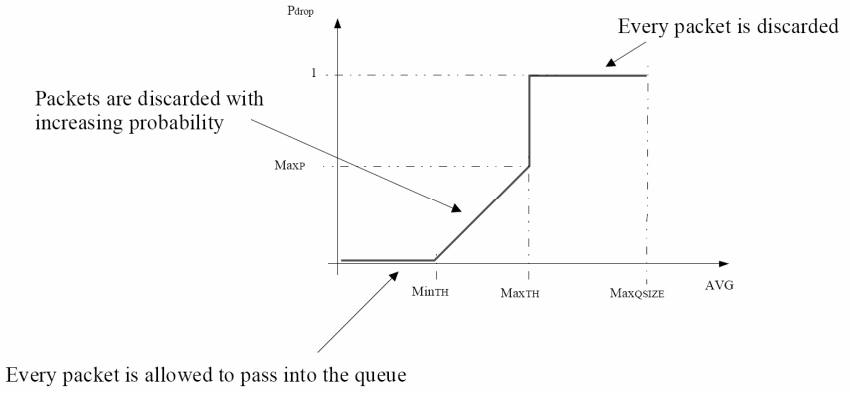
Initial condition:
$avg(0) = 0$
$Count = -1$
When Qsize= 0:
$T_{idle} = T_{now}$
After every packet arrival:
if $Qsize(n) > 0$***:***
$avg(n+1) = (1-\epsilon)\cdot avg(n) + \epsilon \cdot Qsize(n)$
else:
$avg(n+1) = avg(n)\cdot(1-\epsilon)^{f(T_{now}-T_{idle})}$

If queue is empty, averaging is done based on the assumption that N packets have passed the algorithm before actual packet arrival. −> Decay of average during idle times

- Packets are discarded based on the average queue length
- max_p is a user-defined parameter, usually set to 2% or 10%, and *count* is the number of packets since the last packet drop

if $avg(n+1) < min_{th}$***:***
$Count = -1$
else if $min_{th} \le avg(n+1) < max_{th}$***:*** **Stochastic packet discard**
$count = count + 1$
$P_b(n+1) = max_p \cdot \frac{avg(n+1) - min_{th}}{max_{th} - min_{th}}$
$P_a(n+1) = \frac{P_b(n+1)}{1 - count \cdot P_b(n+1)}$
With probability $P_a(n+1)$***:***
Discard packet
$Count = 0$
else if $max_{th} \le avg(n+1)$
Discard packet
$Count = 0$

RED

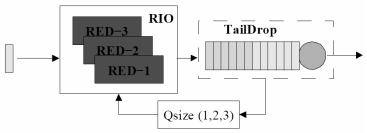

Pdrop
Every packet is discarded
1
Packets are discarded with
increasing probability
MaxP
MinTH
MaxTH
MaxQSIZE
AVG
Every packet is allowed to pass into the queue

Achievements of RED

- **Some packets are discarded even before overflow of the actual buffer**
  - Is it good or bad ?
    - Bad: A part of buffer space is in some occasions wasted
    - Good: A signal is sent to co− operating sources that they should decrease their sending rate or congestion will occure
- **On the average early packet discards will hit connections which use more than their fair share of capacity in contending link**
  - Is it good or bad ?
    - Bad: Makes differentiation impossible
    - Good: Is consistent policy and withing the goal of conventional Best Effort model

RED In/Out – WRED

- When we aim for differentiation of resources we must also allow different shares of resources in contending link or buffer
- One way to do it is to use RED with several parallel algorithms and thresholds
  - RED In/Out −> RIO or WRED
  - Popular implementations use two or three parallel algorithms
- This requires that packets are marked
  - One algorithm is responsible of one or several marks

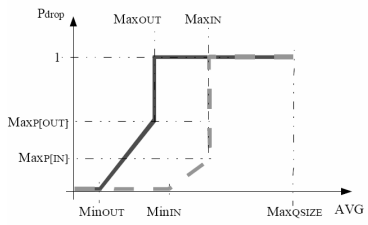


RIO

- Operation is usually based on following idea:
  - Customer has contracted capacity of X bps
  - He sends packets with rate Y bps
  - If Y is greater than X, some packets are marked as out of profile.
    - Out of profile packets usually experience harsh treatment on contending situations
- Calculation of the average queue length is modified to take into account number of packets with different markings:
  - In (green): Only green packets
  - In/Out (yellow): Green and yellow packets
  - Out (red): All packets in the queue
- WRED=configure RED features selectively, eg. based on IP Precedence or IP DSCP

Parameters in WRED

- All parameres are independent for different markings
- More dimensions in creating differentiation



- Some parameters are common for different markings
- Less dimensions but more understandable