Contrôle de trafic
&
Qualité de services

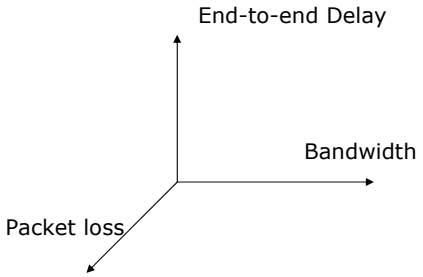
Sommaire Général

- Packet-level traffic control mechanisms
  - Applications requierement
  - The different types of guarantees
  - Bandwidth guarantees
  - Delay guarantees
- Flow-level traffic control mechanisms
- Network-level traffic control mechanisms
- Stantardized Services

Packet-level traffic control mechanisms

- Applications requierement
- The different types of guarantees
  - Bandwidth, delay
- Supporting the best-effort service
- Bandwidth guarantees
  - Maximum bandwidth
  - Minimum bandwidth
- Delay guarantees

Application requirements

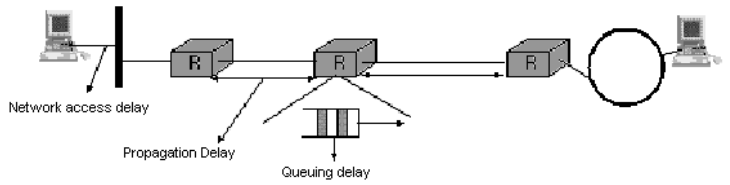


- Ideal world
  - 0 end-to-end delay
  - 0% packet loss
  - infinite bandwidth
- Real world
  - optimize for delay, bandwidth or loss, but not all !

Application requirements

- Interactive reliable applications
  - telnet, tn3270, database access, short www transactions, ftp-control, Xwindow, ...
    - a human is waiting for information from server
    - end-to-end delay should be minimized
    - required bandwidth is low
    - packet losses will be recovered by TCP
      - but high losses will increase application-level end-to-end delay
- Interactive multimedia applications
  - voice or video over IP
    - two humans are discussing through the network
    - end-to-end delay should be minimized
      - a late packet is equivalent to a lost packet
    - required bandwidth can be high
    - can survive with low packet loss ratio

End-to-end packet delay



Components of the end-to-end delay

- network access delay [variable]
  - e.g.CSMA/CD, Token Ring, ...
- propagation delay [fixed]
  - electrical signal needs roughly 5 µsec to travel 1 kilometer
- transmission delay [fixed]
  - a packet of P bytes needs (P*8)/B seconds to be sent on a B bits per second link
- queuing delay inside each intermediate router [variable]
  - varies with buffer occupancy of each router

Application requirements

- Network control
  - OSPF, RIP, SNMP, BGP, ...
    - bandwidth requirement is usually low
    - delay should be low
    - packet loss should be avoided
      - packet loss may increase convergence time for routing
      - protocols or may affect SNMP traps
- Request-response applications
  - client server, NFS, ...
    - bandwidth requirement can be high
    - delay should be low for short transactions especially when human/computer is waiting for the answer
    - packet loss can be recovered by higher layer protocols

Application requirements

- Batch applications
  - ftp, remote backup, long http transactions,...
    - a human wants the job to be done but is not waiting
    - end-to-end packet delay can be high
    - bandwidth requirements can be high
    - packet loss ratio does not need to be low
      - packet losses will be recovered by TCP
- Non-interactive multimedia applications
  - distance learning, audio/video broadcasts, ...
    - a human is receiving a continuous flow of information
    - end-to-end delay does not need to be low
    - bandwidth requirements can be high
    - packet losses can usually be tolerated
      - but too lost packets may completely kill the application

Packet-level traffic control mechanisms

- Les besoins des applications
- The different types of guarantees
  - Bandwidth, delay
- Supporting the best-effort service
- Bandwidth guarantees
  - Maximum bandwidth
  - Minimum bandwidth
- Delay guarantees

What kind of guarantees ?

- Three types of bandwidth guarantees
  - Best effort (no guaranteed bandwidth))
    - suitable for classical, non-critical, elastic applications
  - Maximum guaranteed bandwidth
    - some amount of bandwidth is reserved for the flow
    - the flow cannot send faster than its maximum bandwidth
    - suitable for non-adaptive streaming applications
  - Minimum guaranteed bandwidth
    - at any time the flow will always be allowed to use at least its minimum guaranteed bandwidth
    - flow may use more bandwidth if network is not congested
    - suitable for critical elastic applications and adaptive streaming applications

What kind of guarantees ?

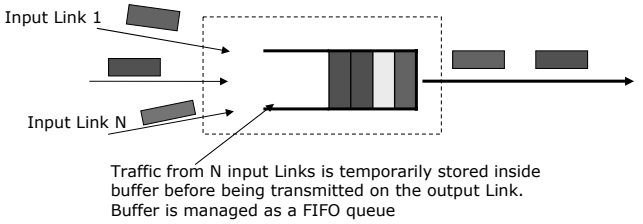
- Delay and delay jitter guarantees
  - Best effort flow
    - no delay or delay jitter guarantee
  - Maximum guaranteed bandwidth flow
    - maximum delay guarantee
      - e.g. For interactive voice
    - delay jitter guarantee
      - e.g. When playback jitter cannot be used by receiver
  - Minimum guaranteed bandwidth flow
    - maximum delay guarantee
      - e.g. For adaptive streaming applications
    - delay jitter does not really make sense

Packet-level traffic control mechanisms

- The different types of guarantees
  - Bandwidth, delay
- Supporting the best-effort service
- Bandwidth guarantees
  - Maximum bandwidth
  - Minimum bandwidth
- Delay guarantees

Best effort service : simple router

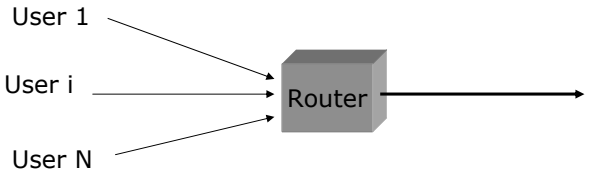
- Simplest possible router
  - N input links
  - 1 output link



- Simplest solution for best-effort service

Characterizing best-effort service

- What should be the goal of a best-effort service ?
  - The network should provide a fair service too all its best-effort users
    - Fairness definition for a single bottleneck link



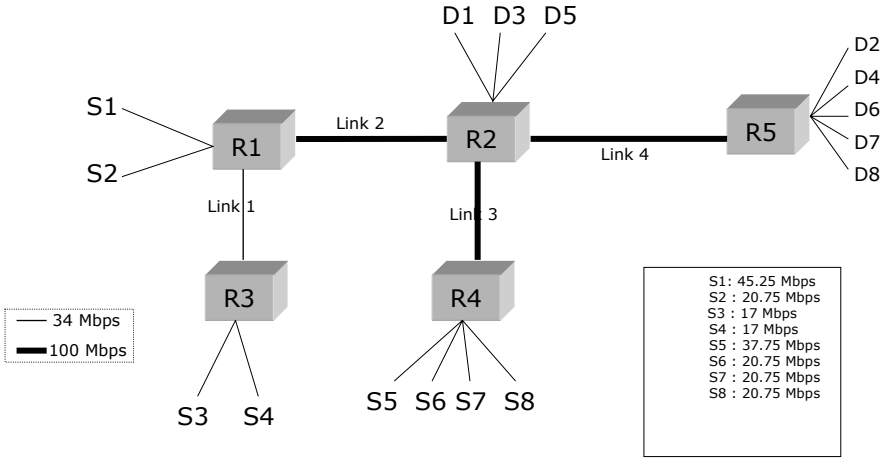
    - Fairness definition for a complete network
      - Maximize bandwidth received by users ?
      - Maximize utilization of network resources ?

Max-min fairness

- Fairness definition for networks
  - a max-min allocation of bandwidth is an allocation of bandwidth which maximizes the allocation of bandwidth to the sources receiving the smallest allocation
  - Property
    - a max-min fair allocation is such that in order to increase the bandwidth allocated to one source, it is necessary to decrease the bandwidth allocated to another source which already receives a lower allocation

Max-min fairness : example

- Max-min fair bandwidth allocation

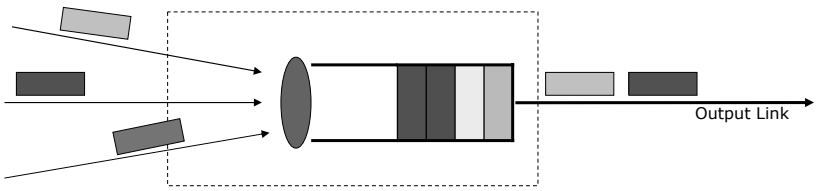


How to provide max-min fairness ?

- Max-min fairness in TCP/IP network
  - an ideal goal, difficult to attain in practice
- Fairness will depend on 2 factors
  - Packet treatment inside routers
    - Which packets are dropped when congestion occurs
  - Congestion control mechanism implemented inside endsystems
    - TCP congestion control for applications based on TCP
    - what about UDP-based applications

Simple router v1

- Packet treatment inside router



Buffer Acceptance Algorithm

When a packet arrives from an input link, the buffer acceptance decides whether this packet can be accepted inside the router's buffer.

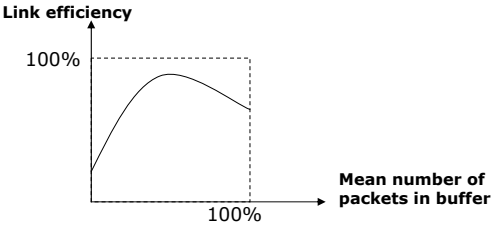
Buffer acceptance algorithms

- Two fundamental questions
  - When do we drop a packet ?
    - when the buffer is full
      - example : tail drop
    - when the buffer occupancy increases too much
      - example : Random Early Detection
  - Which packet should be dropped
    - The arriving packet (the packet at the tail of the queue)
      - but is this packet responsible for congestion ?
    - Another packet from the same flow as the arriving packet
      - this might help congestion control algorithms
    - A packet from some flow
      - not necessarily from the same flow as the arriving packet
    - The packet at the head of the queue
      - could improve the performance with TCP

Buffer acceptance algorithms

Objectives

- control the amount of packets in the buffer to
  - efficiently support best-effort traffic
    - should provide a fair utilization of the routers buffers
  - provide protection among different flows
    - one flow should not prohibit other flows from having packets inside the router's buffers
  - achieve a good utilization of output link



Tail drop

- Simplest buffer acceptance algorithms
- Principle
  - when a packet arrives at a full buffer, the arriving packet is discarded
  - Advantages
    - easy to implement
    - can limit the number of packet losses for large buffer
  - Disadvantages
    - no distinction between the various flows
    - not the best solution for TCP traffic

Random Early Detection

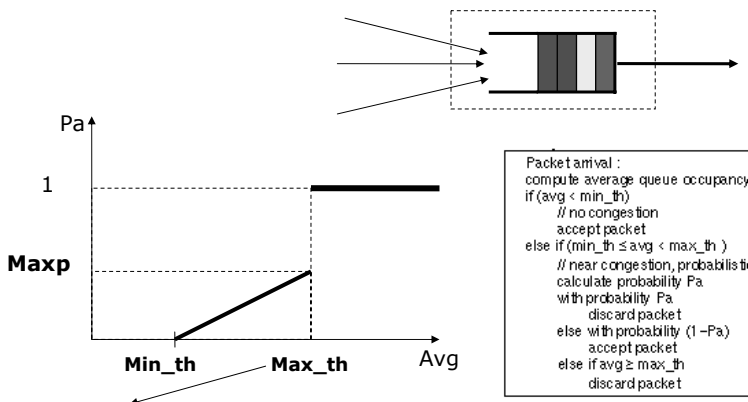
- Goals
  - should be easily implemented in simple routers with a single logical queue
  - achieve a low, but non-zero, average buffer occupancy
    - low average occupancy provides low delay for interactive applications and ensure fast TCP response
    - non-zero average occupancy ensures an efficient utilization of the output link
  - approximate a fair discard of packets among the active flows without identifying them
  - discard packets in a TCP friendly way
    - we should avoid discarding bursts of packets since TCP reacts severely to burst losses

Random Early Detection

- Principle
  - How can we detect congestion ?
    - measure average buffer occupancy by using a low-pass filter
    - buffer is considered congested when its average occupancy is above a configured threshold
      - threshold value usually around 10%- 20% of buffer size
  - What do we do in case of congestion ?
    - Probabilistic drop for incoming packet
      - drop will force TCP to slow down
      - drop probability should increase with congestion level
    - Why probabilistic drop ?
      - Avoid dropping burst of packets from single flow
      - Try to drop packets for each flow in proportion of network usage
      - Avoid synchronization effects

Random Early Detection

- Implementation
  - suitable for routers with a single queue




```
Packet arrival :
compute average queue occupancy, avg
if (avg < min_th)
      // no congestion
      accept packet
else if (min_th ≤ avg < max_th )
      // near congestion, probabilistic drop
      calculate probability Pa
      with probability Pa
            discard packet
      else with probability (1 –Pa)
            accept packet
      else if avg ≥ max_th
            discard packet
```

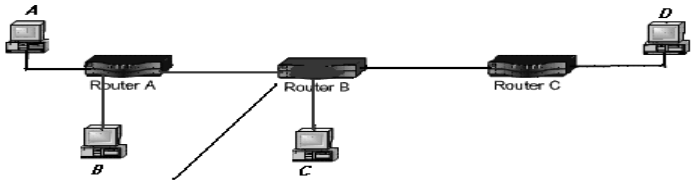
Steady state: average occupancy buffer between Min_th and Max_th.

Issues with RED

- Shall you deploy RED in your network ?
- Difficult to provide a clear answer today
  - Some argue that RED provides
    - a better network utilization
    - a lower queuing delay
  - Others complain on the complexity of tuning RED
    - How do we set minth, maxth, maxp and wq in an operational network ?
      - Do the settings depend on link speed, type of traffic, ... ?
    - A bad choice of the RED parameters may provide a worse performance than plain old tail-drop

TCP Explicit Congestion Notification

- Congestion control in TCP/IP networks
  - Basic assumption
    - packet loss is *the (only) indication of congestion*
    - *Router behavior*
      - *Discard packets when congestion occurs*
    - *Endsystems behavior*
      - *TCP congestion control (slow-start, congestion avoid....)*
    - *Problem caused by this assumption*



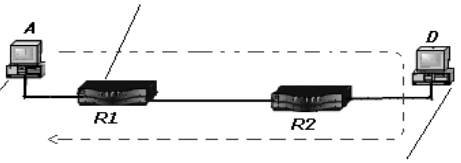
TCP Explicit Congestion Notification

- How can we improve TCP/IP congestion control?
- Modify router behavior
  - if lightly congested, a router could inform the sources of the congestion
  - if heavily congested, router will drop packets
- How can a router inform sources ?
  - Send a special message directly to the source
    - ICMP source quench, part of ICMP standard, but rarely used because sending new packets in case of congestion is not always the best solution...
  - Add some notification to "congested" packets
    - this notification will be received by the destination and returned to the source that will react...

TCP Explicit Congestion Notification

- Basic solution

Congestion Notification:
Mark the IP packet that caused congestion by setting one bit flag (CE: Congestion Experienced)



TCP Source Behavior:
Upon reception of a ECN-echo=1 TCP ack, behave as if the corresponding TPDU was lost (perform congestion avoidance).

TCP destination Behavior:
Upon reception of CE=1 IP packet indicate the congestion to the source by setting a speciel flag (ECN-echo) in the returning TCP ack .

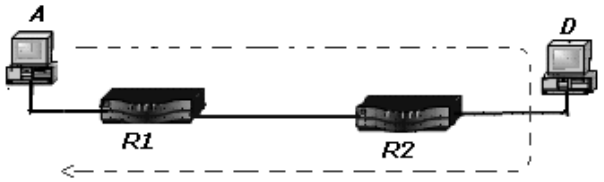
- Potential problems
  - What happens if the returning ECN-echo ack is lost ?
  - How can we deploy such a solution when 99.99% of the TCP sources/destinations are not ECN capable ?

TCP ECN and lost acks

Problem

- If an ECN-Echo ack is lost, the sender will not be aware of the congestion
- The receiver could send N ECN-Echo acks, but the sender should only decrease once its cwnd

- Solution
  - allow sender to confirm notification to receiver



TCP sender behavior
Upon reception of a ECN-Echo=1 TCP ack
Perform congestion avoidance and set CWR
Flag in next TCP PDU

TCP receiver behavior
Upon reception of a CE=1 IP packet indicate the
Congestion to the sourceby by setting a special
Flag (ECN_Echo) in all returning TCP asks until
A TCP TPDU with CWR setis received

Deployment of ECN

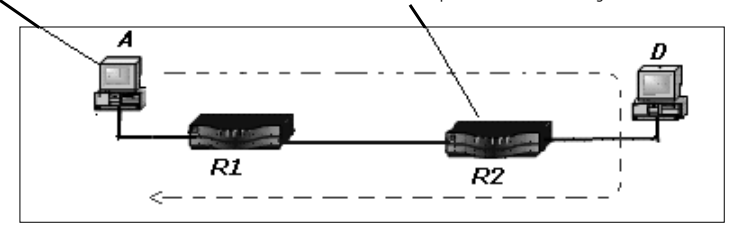
- How can we support ECN in endsystems ?
  - TCP
    - TCP must be modified to support the new flags
    - at connection establishment time, the utilization of ECN will be negotiated during the three way handshake
      - if both endsystems support ECN, it will be used
      - if one of the endsystems does not support ECN, fallback to normal TCP congestion control
  - Other transport protocols
    - work is required to adapt the congestion control mechanism used by these protocols to support ECN

Deployment of ECN

- How can we deploy ECN-aware routers ?
  - to work properly, a router should be able to distinguish between :
    - IP packets from ECN-capable flows
      - these flows should be notified when congestion occurs
    - IP packets from non-ECN capable flows
      - packets from these flows should be discarded during congestion
  - ECT bit in IP header

ECN-capable souce
If destination is also ECN capable
Set ECT bit in all IP packets toward destination
Otherwise Reset ECT bit

In case of congestion
If ECT bit is set
Mark the IP packet that caused congestion
By setting on bit flag (CE: Congestion Experienced)
If ECT bit is not set
Discard the IP packet that caused congestion



Packet-level traffic control mechanisms

- The different types of services
  - Best effort, Maximum and Minimum bandwidth
- Supporting the best-effort service
- Supporting the Maximum bandwidth service
  - How to identify flows
  - Mechanisms to provide a maximum bandwidth to identified flows
- Supporting the Minimum bandwidth service

Packet-level traffic control mechanisms

- The different types of guarantees
  - Bandwidth, delay
- Supporting the best-effort service
- Bandwidth guarantees
  - Maximum bandwidth
  - Minimum bandwidth
- Delay guarantees

Best effort versus guaranteed service

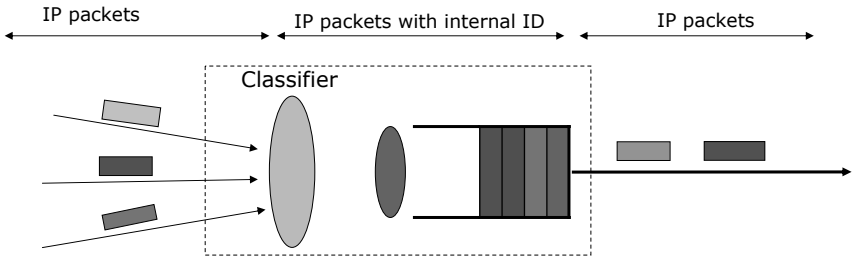
- Provision of best effort service
  - Can be done by assuming that all IP packets want to receive exactly the same service
- Provision of guaranteed bandwidth service
  - All IP packets are not equal anymore
  - At some places inside the network, some devices must know what kind of guarantee has been associated with a particular IP packet
    - bandwidth is a characteristic of a flow of packets
  - Problems to solve
    - Associate IP packets to flows
    - Provide guarantees for specific flows

What is a flow ?

- Definition
  - a flow is a sequence of packets with one common "characteristic"
    - characteristic can be based on any field of the packets
    - a flow usually exist for some period of time

  - a layer-N flow is a sequence of packets with one common layer-N characteristic
    - layer two flow
      - e.g. ATM or frame relay circuits
    - layer three flow [IP related]
    - layer four flow [TCP or UDP related]
    - layer seven flow [application level flow]

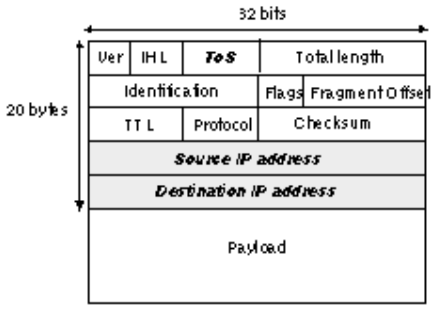
Simple router v2



- Roles of the classifier
  - identify the flow to which an arriving packet belongs
    - identification can require complex operations
  - store this information internally so that other parts of the router will easily determine the flow of a packet
    - classification should be done at most once in each router

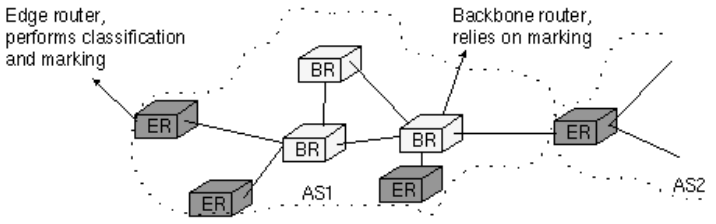
Layer-three flow

- Identification of layer-three flows
  - source and destination IP addresses with or without associated netmasks
    - e.g. all traffic from 138.48.0.0/16
  - all IP traffic with same route or BGP next hop
    - requires a route table lookup by the classifier



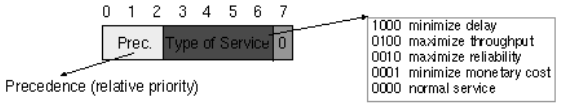
Layer-three flow

- Layer-3 classification on each intermediate router can be expensive
- Alternative solution
  - perform classification at the ingress of the network
  - explicitly mark the classified packets
    - downstream (backbone) routers will rely on the marking without needing to classify each packet



IP packet marking

- How can we mark an IP packet ?
  - Steal one field of the IP header
    - ToS : Type of Service Octet
    - defines the relative importance of the IP packet and the type of service required for this packet



    - current status
      - definition of ToS Octet changed several times
      - Precedence is used in some networks
      - ToS field is rarely used
    - Using the ToS Octet for marking
      - advantage : easy to implement
      - disadvantage : limited number of marked flows

Identifying applications

- The problem
  - Not all applications use well-known port numbers
- Examples
  - FTP
    - server and client may negotiate other non-default port numbers than 20/21 for some file transfers
  - Netshow/RealAudio
    - RTSP protocol (port 554) is used to negotiate the content to be streamed and the UDP/TCP port numbers
  - SunRPC
    - RPC services utilize any port number, portmap/rpcbind (port 111) is used to locate the service
  - Napster/Gnutella and other new applications
    - rarely utilize well-known port numbers

Identifying applications

- How to classify ftp/realaudio/RPC packets ?
- Principle of the solution
  - Examine contents of control packets for these applications
    - ftp-control, RTSP, portmap, ...
    - based on the specific port information captured, perform layer-4 classification
  - Drawbacks
    - can be CPU intensive
    - a specific protocol handler needs to be written for each control protocol
    - won't scale to a large number of such applications

Identifying applications Limitations

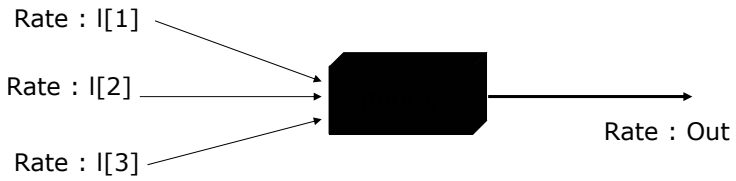
- Not all applications are known and can be easily identified
  - see Internet traffic characteristics
- Deployment of encrypted tunnels (IPSEC, L2TP, PPTP) will hide TCP and UDP headers to intermediate routers
  - if special treatment is required inside network, packets should be marked at layer 3 before being encrypted
  - deployment of encryption together with traffic control should be carefully done

Classification today

- At which layer should we classify ?
  - layer-7 classification is very expensive
    - requires examination of packet headers and contents
    - will probably only be used by special equipments
      - will not work at high-speed
      - will not be deployed in backbones
  - layer-3 versus layer-4 classification
    - no real consensus today
    - some believe that layer-4 classification can be performed by each router, even in backbones
      - ASICs required to perform layer-4 classification at high speed
    - many others believe that backbone routers cannot perform complex layer-3 and layer-4 classifications
      - classification should be done by edge routers
      - backbone routers should only look at special markings

Providing bandwidth guarantees

- How can we provide bandwidth guarantees in a packet-based network ?



- If traffic is a fluid flow, bandwidth can be guaranteed provided that
  - $\sum_{i=1}^{i=N} I_{[i]} \leq Out$

Providing bandwidth guarantees

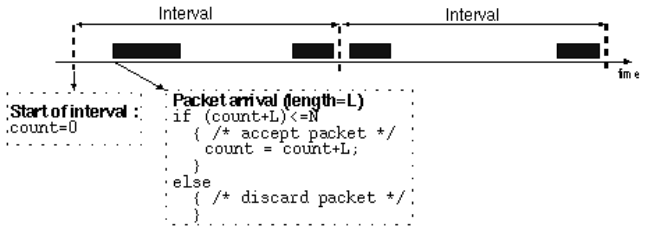
- Problem
  - In packet based networks, traffic is a flow of variable length packets and not a fluid flow
- How to provide bandwidth guarantees ?
  - Ensure that the output link will not be a bottleneck
    - $\sum_{i=1}^{i=N} I_{[i]} \leq Out$
    - We must limit the rate of flows on the input links
  - Ensure that the buffers of the router will not overflow
    - We must limit the amount of buffer consumed by the flows on the input links

Limiting rate of incoming flows

- How can we limit the rate of a flow of variable length packets on an input link ?
  - Define flow rate for traffic contract
    - Which rate unit ?
    - Number of packets per unit of time
      - one 40 bytes packet per second versus one 1500 bytes packet per second
      - amount of information inside each packet must be considered
    - Number of bytes (bits) per unit of time
      - sounds better, but what appropriate unit of time ? one microsecond, one millisecond one second, one hour, one day...
  - On packet arrival
    - If current rate is within contract, accept packet
    - Otherwise discard packet

Measuring the rate of a flow

- Simple solution
  - time divided in fixed length intervals
  - Traffic contract defined as N bytes per interval

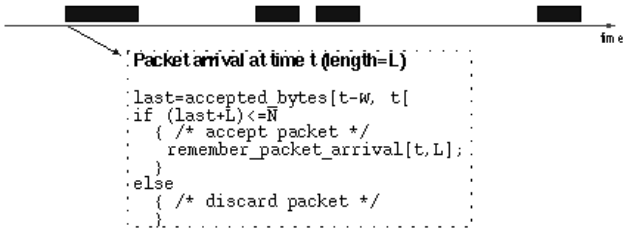


  - Drawback
    - starting time of first interval may have an influence over which packets are accepted

Measuring the rate of a flow

Improved solution

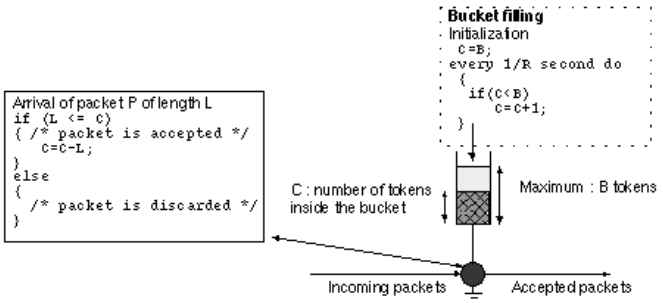
- when a packet arrives, the rate is equal to the number of bytes received during the last *W* sec divided by *W* (W: time window)



  - Drawback
    - not really implementable in practice

Token Bucket

- Token bucket
  - R : average rate in bytes/sec
  - B : size of the token bucket



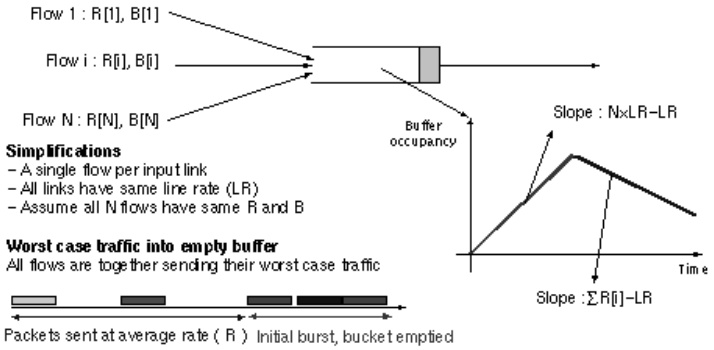
  - During a period of *T* seconds, the token bucket accepts at most (B + *T* R) bytes of traffic
    - worst case traffic at output of token bucket

Token Bucket

- Advantages
  - can be used by a network provider to enforce a traffic contract since it provides a precise algorithmic definition for
    - conforming packets
    - non-conforming packets
  - provides a bound on the average rate
  - provides a bound on the maximum amount of traffic during any period of time
    - important to fix the size of buffers inside routers

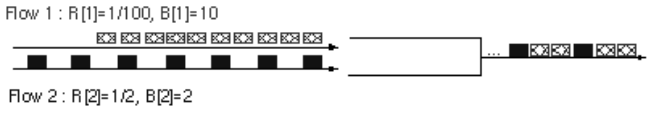
Bound on buffer occupancy in routers

- How can we ensure that no packets will be discarded by router due to buffer overflow ?
  - Worst case analysis

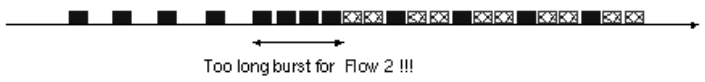


Impact of multiplexing

- What happens when a flow is multiplexed together with another flow inside a router ?



  - Traffic contract of second flow on output link ?

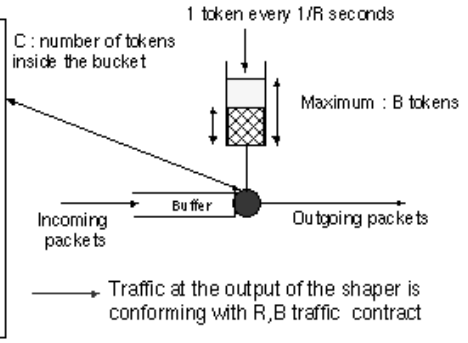


    - The multiplexing with flow1 has increased the burstiness of flow 2
  - In networks, burstiness of one flow may increase at *each* intermediate router !

Token Bucket in shaping mode

- Problem
  - How can we ensure that one particular flow (e.g. after multiplexing) is conforming to a contract ?
    - utilize modified token bucket in shaping mode (shaper)

```
Arrival of packet of size L
if (L <= C)
{ /* packet arrived on time */
   C=C-L;
   transmit_packet();
}
else
{/* packet arrived too early
  * delay packet inside buffer
  * until it becomes conforming
  */
 while (C<L)
  { /* wait */ }
 /* now C=L and packet is
conforming */
 C=C-L;
 transmit_packet();
}
```



Packet-level traffic control mechanisms

The different types of guarantees
- Bandwidth, delay

- Supporting the best-effort service

- Bandwidth guarantees
  - Maximum bandwidth
  - Minimum bandwidth

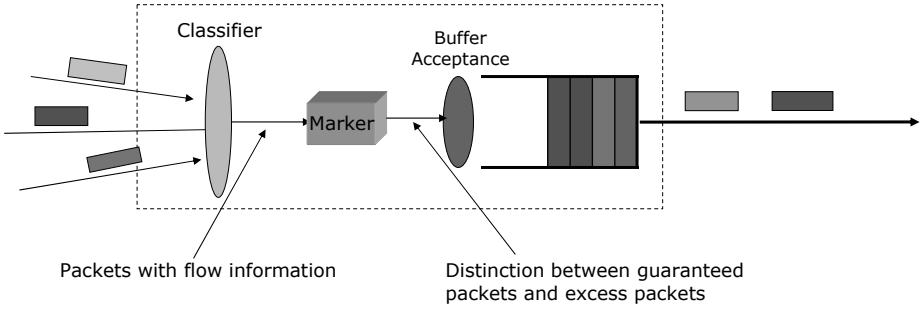
- Delay guarantees

How to provide minimum guaranteed bandwidth ?

- Problem
  - within each flow, we now have 2 types of packets
    - IP packets that are part of the minimum guaranteed bandwidth for the flow
      - these packets cannot be discarded inside the router
    - IP packets that are in excess of the minimum guaranteed bandwidth
      - these packets should be treated as best-effort packets and can be discarded if necessary to preserve the guarantees
- Principle
  - identify the two types of packets
  - discard preferably the non-guaranteed packets when congestion occurs inside router

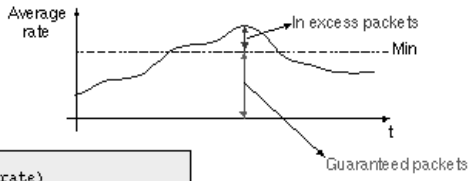
Identification of the guaranteed packets

- Principle
  - Measure the rate of the incoming flow
  - Identify the packets within the minimum bandwidth
  - Identify the packets in excess of the min. bandwidth
  - Packets may be explicitly or internally marked



Probabilistic marking

- Principle
  - Measure average rate of incoming flow
  - Mark packets in proportion of excess amount



Marking algorithm

```
if (average_rate <= Min_rate)
{ /* packet is guaranteed */ }
else
{
  P = (average_rate - Min) / average_rate
  with probability P
    { /* packet is in excess */   }
  else
    { /* packet is guaranteed */  }
}
```

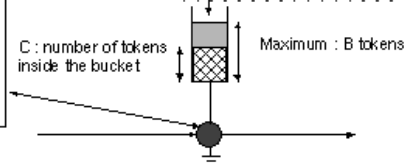

Deterministic marking

- Principle
  - Modify token bucket to mark non-conforming packets instead of discarding them
    - must specify bucket size in addition to minimum bandwidth

Bucket filling

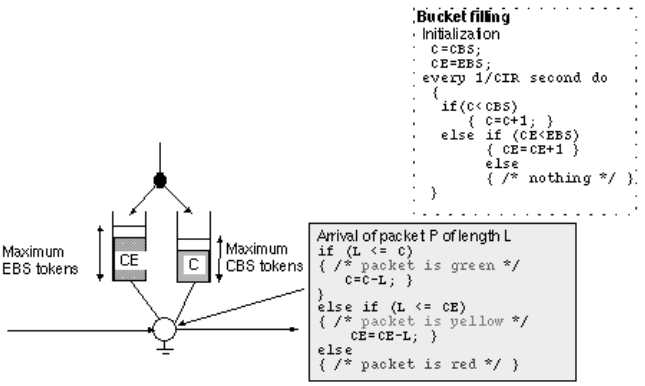
```
Initialization
 C=B;
every 1/R second do
 {
  if(C<B)
     C=C+1;
 }
```

```
Arrival of packet P of length L
if (L <= C)
{ /* packet is guaranteed */
   C=C-L;
}
else
{
  /* packet is in excess */
}
```



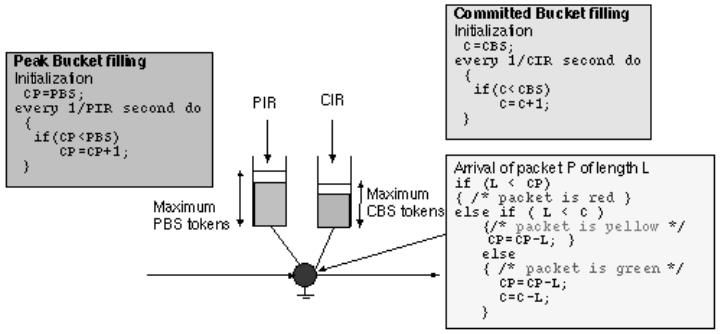
Extensions to token bucket

- Single rate three color marker
  - Committed Information Rate (CIR)
  - Committed Burst Size (CBS)
  - Excess Burst Size (EBS)



Extensions to token bucket

- Two rate three color marker
  - Committed Information Rate (CIR)
  - Committed Burst Size (CBS)
  - Peak Information Rate (PIR)
  - Peak Burst Size (PBS)



Probabilistic versus Deterministic marking

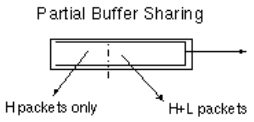
- Probabilistic marking
  - approximately mark packets in function of rate
  - adapted to TCP behavior
  - no mathematical theory to support it and prove bounds
- Deterministic marking
  - supported by strong mathematical theory
  - easy to implement
    - similar mechanisms are used in other technologies
  - may not be the best solution for TCP
    - TCP more bursty than what is accepted by token bucket

Packet discard preferences

Problem

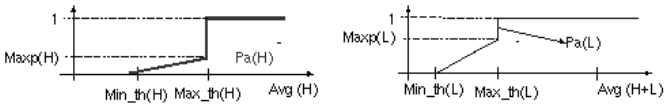
- two types of packets
  - high and low priority packets
- carry preferably high priority packets

- Solution
  - Discard less-important packets earlier than others



Packet discard preferences

- Weighted RED
  - extension of RED to support several packet discard preferences
  - example with two discard preferences
    - Two RED algorithms run in parallel
      - the first one decides the acceptance of high priority packets that should only be discarded in case of severe congestion
      - the second one decides the acceptance of low priority packets that should be discarded earlier than high priority packets



Packet-level traffic control mechanisms

The different types of guarantees
- Bandwidth, delay

- Supporting the best-effort service

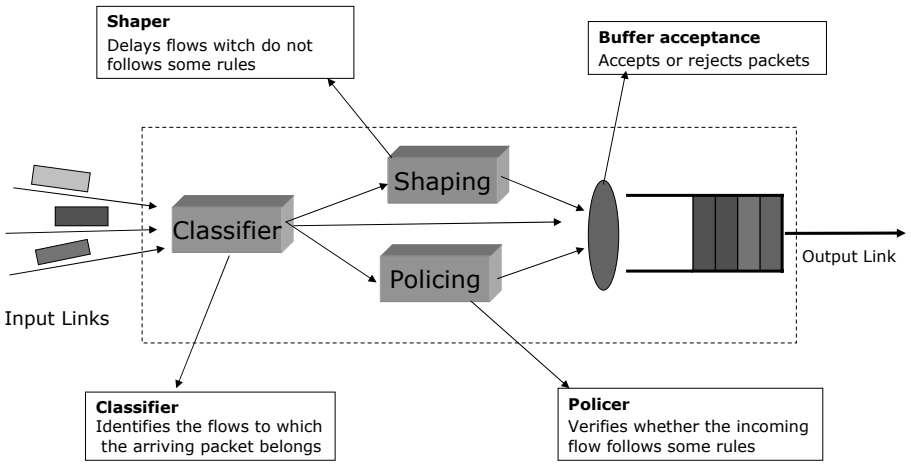
- Bandwidth guarantees
  - Maximum bandwidth
  - Minimum bandwidth

- Delay guarantees

Towards multiservice networks

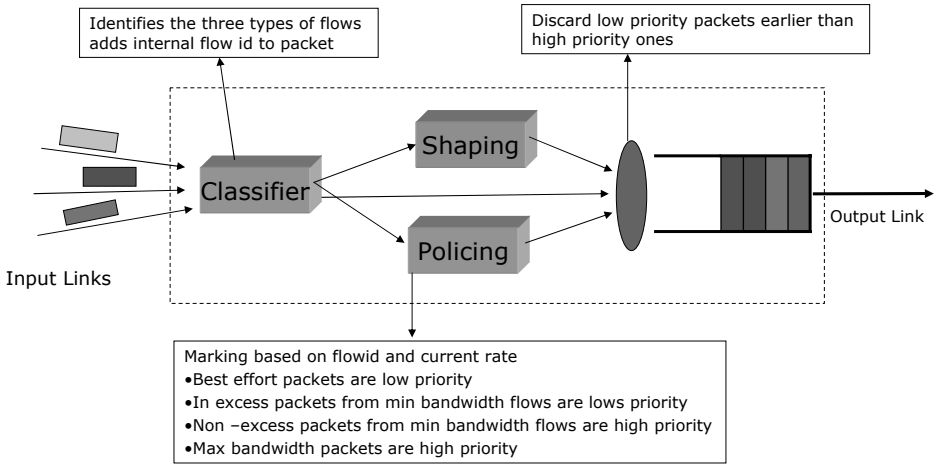
- Problems
  - How can we multiplex on a single link through one router classical best-effort traffic packets and packets from guaranteed flows ?
    - guaranteed packets should not be perturbed by best-effort packets
    - best-effort packets should be able to utilize the output link when there is no guaranteed traffic
  - How can we provide different delay guarantees
- What can we do with our simple router ?

Simple router v3



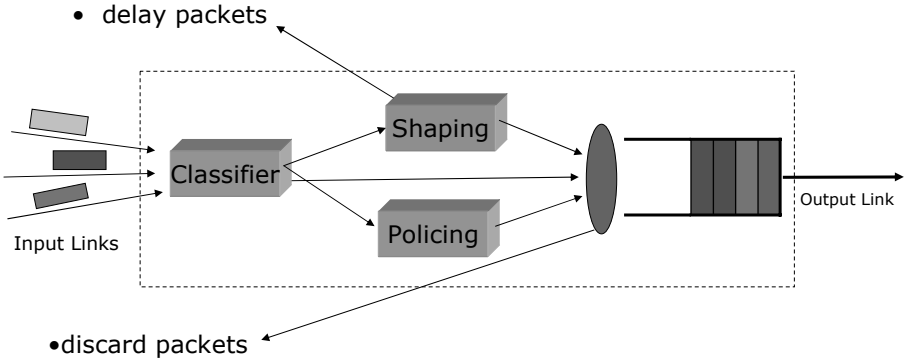
Multiplexing with Simple router v3

- Best-effort, min and max bandwidth flows



What about delay guarantees ?

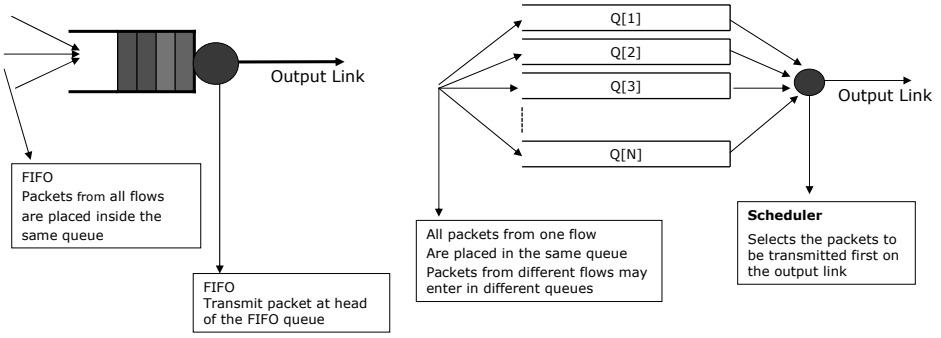
- Mechanisms supported by simple router v3
  - delay packets



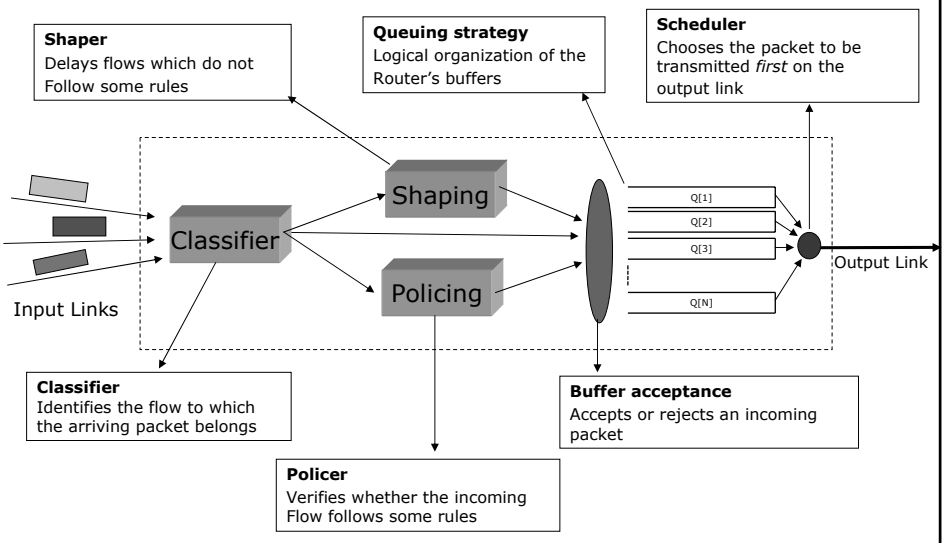
- What can we do to ensure that packets from interactive streaming application will be sent earlier than packets from batch application ?

What about delay guarantees

- Solution
  - Add delay differentiation to loss differentiation
    - some packets should be sent earlier than others
    - Replace FIFO buffer by set of queues and scheduler



QoS capable router



Scheduler

- Function
  - among all the logical queues containing at least one packet, select the packet that will be transmitted on the output link
- A scheduler should ...
  - be easy to implement in hardware
  - support best-effort and guaranteed services
  - provide fairness for best-effort traffic
    - max-min fairness is the desired goal
  - provide protection
    - one flow should not be able to steal bandwidth from other existing flows
  - provide statistical or deterministic guarantees
    - bandwidth, delay

Scheduling algorithms

- Two types
- Work-conserving scheduler
  - a work-conserving scheduler will always transmit one packet provided that there is at least one packet inside the router buffers
- Non-work-conserving scheduler
  - a non-work-conserving scheduler may defer the transmission of packets on the output link even if some packets are waiting inside the router buffers
  - can provide guarantees on delay jitter
  - nice in theory, but not often implemented

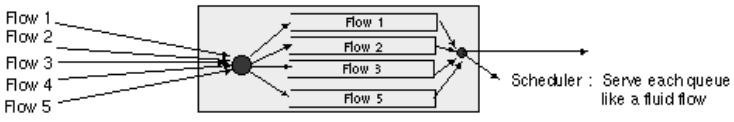
Scheduling best-effort flows

Design goals

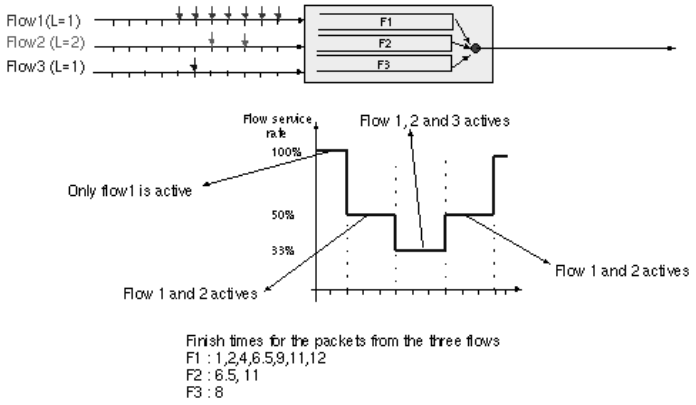
- Provide a fair distribution of bandwidth between active flows to support max-min fairness at network level
  - fairness should not depend on behavior of congestion control mechanisms inside endsystems
- Provide protection between flows
  - a potentially misbehaving flow should not be able to consume most of the available bandwidth
    - scheduler ensures distribution of output link bandwidth
    - packet discard mechanism should ensure that one flow cannot consume all the available buffer space
- Implementable at high speeds

Processor Sharing

- Processor Sharing (PS)
  - ideal work-conserving scheduler
    - each queue is served by the scheduler as if it contained a fluid flow
    - at time t, Queue[j] is served at rate
      - Rate[j]= $link_{res}$ / n activeflows
      - a flow is considered active if its queue contains something



Processor Sharing : example



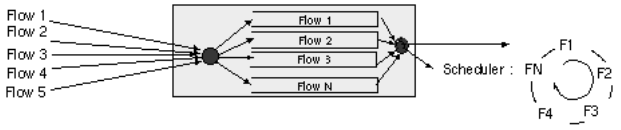
Finish times for the packets from the three flows
F1 : 1,2,4,6.5,9,11,12
F2 : 6.5, 11
F3 : 8

Processor Sharing

- Advantage
  - if no packets are discarded, a network of PS schedulers will provide a max-min fair service
    - the fairness does not depend on any congestion control mechanism
    - if packets are discarded, than packet discarding must be performed in a fair manner
- Disadvantage
  - Ideal "mathematical" solution, not implementable in practice
    - approximations to PS are implementable

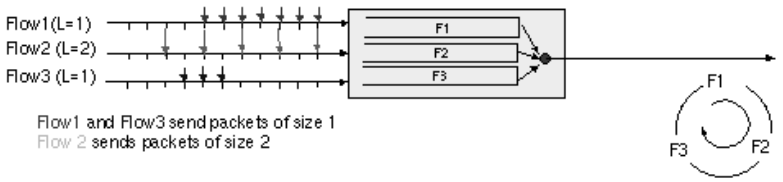
Round Robin

- Round Robin



  - Principle
    - serve the active queues one after the other
  - Advantages
    - can be easily implemented in hardware
    - provides protection for best-effort traffic
    - provides fair distribution of bandwidth with fixed-size packets
      - but fairness is only provided at timescales larger than schedule
  - Disadvantages
    - unfairness with variable length packets

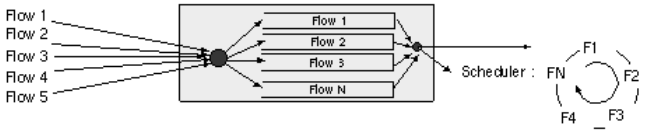
Round Robin : example



Flow1 and Flow3 send packets of size 1
Flow 2 sends packets of size 2

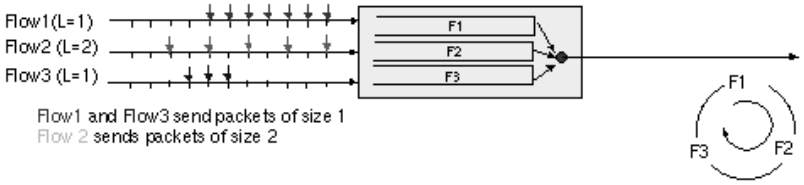
T	In F1	In F2	In F3	Q(F1)	Q(F2)	Q(F3)	Scheduled
0	P10[1]	P20[2]	–	P10	P20	–	F1:P10
1	P11[1]	–	–	P11	P20	–	F2:P20
2	P12[1]	P22[2]	–	P11,P12	P22	–	F2:P20 (cont)
3	P13[1]	–	–	P11,P12,P13	P22	–	F1:P11
4	P14[1]	P24[2]	–	P12,P13,P14	P22,P24	–	F2:P22
5	P15[1]	–	P35[1]	P12,P13,P14,P15	P24	P35	F2:P22 (cont)
6	P16[1]	P26[2]	P36[1]	P12,P13,P14,P15,P16	P24,P26	P35,P36	F3:P35
7	–	–	P37[1]	P12,P13,P14,P15,P16	P24,P26	P36,P37	F1:P12
8	–	P28[2]	–	P13,P14,P15,P16	P24,P26	P36,P37	F2:P24
9	–	–	–	P13,P14,P15,P16	P26,P28	P36,P37	F2:P24
10	–	P2A[2]	–	P13,P14,P15,P16	P26,P28	P36,P37	F3:P36

Deficit Round Robin



- Idea
  - Round-Robin + variable length packets
- Principle
  - associate counter d[i] to each queue
  - increase d[i] by *quantum* every time queue[i] is visited
    - if first_packet of queue[i] larger than d[i] { packet stays in queue[i] }
    - else { packet is transmitted on output link; d[i]=d[i]- packet length; if queue[i] is empty { d[i]=0; } }

Deficit Round Robin : example



Flow1 and Flow3 send packets of size 1
Flow 2 sends packets of size 2

T	Inc.	D[1]	D[2]	D[3]	Q(F1)	Q(F2)	Q(F3)	Scheduled
0	F1	0+1−1	0	0	P10	P20	–	F1:P10
1	F2,F1	0+1−1	0+1	0	P11	P20	–	F1:P11
2	F2	0	1+1−2	0	P12	P22	–	F2:P20
3	–	0	0	0	P13	P22	–	F2:P20(cont)
4	F1	0+1−1	0	0	P14	P22,P24	–	F1:P13
5	F2,F3	0	0+1	0+1−1	P14,P15	P22,P24	P35	F3:P35
6	F1	0+1−1	0	0	P14,P15,P16	P22,P24,P26	P36	F1:P14
7	F2	0	1+1−2		P15,P16	P22,P24,P26	P36,P37	F2:P22
8	–	0	0	0	P15,P16	P24,P26	P36,P37	F2:P22(cont)
9	F3	0	0	0+1−1	P15,P16	P24,P26	P36,P37	F3:P36
10	F1	0+1−1	0	0	P15,P16	P24,P26	P37	F1:P15
11	F2,F3	0	0+1	0+1−1	P15,P16	P24,P26	P37	F3:P37
...								

Frame discard mechanisms

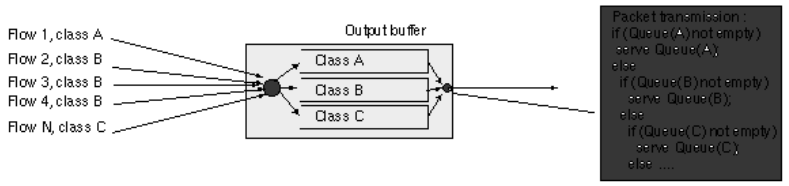
- How can we fairly discard packets ?
  - Packet should be discarded from flow causing congestion
    - How can we identify this congesting flow ?
  - If packets from a flow enter a queue faster than they leave, queue will build up
    - Flow with longest queue is responsible for congestion
    - Discard packet (s) from longest queue
      - at tail of the queue
      - at head of the queue
      - complete queue

Scheduling guaranteed flows

- Design goals
  - Efficiently support flows with minimum and maximum guaranteed bandwidth
    - provide bandwidth guarantees
    - provide delay guarantees
  - Provide protection between flows
    - a potentially misbehaving flow should not be able to jeopardize the guarantees committed to other flows
  - Implementable at high speeds

Priority-based scheduler

- A simple priority scheduler

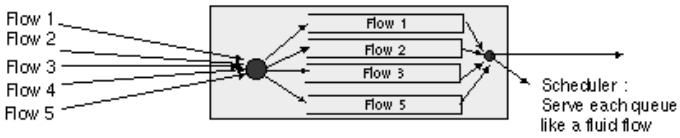


  - Advantages
    - easy to implement
    - packets in high priority class see low delay
  - Disadvantages
    - no protection
    - a high priority flow can "kill" low priority flows
      - this might be desirable in some cases, but not always

→ Priority should be used with care…

Generalized Processor Sharing

- Generalized Processor Sharing (GPS)
  - ideal work-conserving scheduler
    - weight W[i] associated with Queue[i]
    - each queue is served by the scheduler as if it contained a fluid flow
    - at time t, Queue[j] is served at rate
      - Rate= $link_{res} \times (W[j] / \sum_{i} W[i])$
      - a queue is active if it contains something

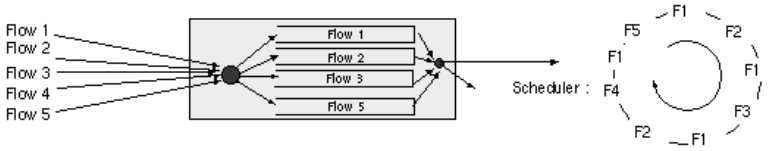


Generalized Processor Sharing

- Advantages
  - provides per-flow bandwidth guarantee
    - through one GPS scheduler
    - through a network of GPS schedulers
  - provides per-flow delay guarantee for token-bucket (R,B) constrained flows
    - through one GPS scheduler (delay bound = B/R)
    - through a network of GPS schedulers
  - provides bound on buffer utilization (buffer bound = B)
  - provides protection among the different flows
    - a flow cannot jeopardize the guarantees for another flow
  - trivial guarantee on delay jitter ([0,D_{max}])
- Disadvantage
  - ideal scheduler not implementable

Weighted Round Robin

Weighted Round Robin



  - if all flows are active, F1 gets 4/9 of bandwidth, F2 2/9, F3, F4 and F5 1/9
- Advantages
  - easy to implement with short schedule
  - different weights provide different bandwidths
  - inter-flow protection
  - Deficit Round-Robin can be extended to support weights
- Disadvantages
  - a long schedule is required to support many flows with small bandwidth, but a long schedule is complex...

Weighted Fair Queuing

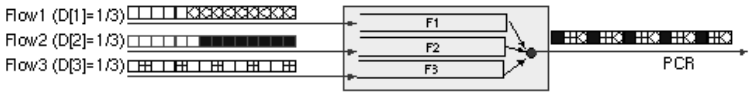
- Objective
  - Define an implementable approximation for GPS
- Idea
  - simulate GPS on a per-packet basis
  - serve the packets in (approximately) the same order as the one they would be served with GPS
- How to do this ?
  - Compute time at which GPS would serve each packet (finish time)
  - Serve packets in order of finish times

Virtual Clock

- Approximation of GPS
  - Idea
    - associate one timestamp to each arriving packet
    - scheduler selects among all the queued packets the packet with the smallest timestamp
  - First algorithm
    - D[i] : bandwidth associated with Queue[i]
    - V[i] : state variable associated with Queue[i]
    - Arrival of a P bytes long packet in Queue[i]
      - V[i] = V[i] + (P / D[i])
      - associate V[i] with the packet
    - Scheduler
      - select the packet with the smallest timestamp for transmission

Virtual Clock

- Example



T	V(F1)	Q(F1)	V(F2)	Q(F2)	V(F3)	Q(F3)	Scheduled
0	0+3	3	0+3	3	0+3	3	F1
1	3+3	6	3+3	3,6	3	3	F3
2	6+3	6,9	6+3	3,6,9	3	-	F2
3	9+3	6,9,12	9+3	6,9,12	3+3	6	F1
4	12+3	9,12,15	12+3	6,9,12,15	6	6	F3
5	15+3	9,12,15,18	15+3	6,9,12,15,18	6	-	F2
6	18+3	9,12,15,18,21	18+3	9,12,15,18,21	6+3	9	F1
7	21+3	12,15,18,...	21+3	9,12,15,18,21	9	9	F3
8	24+3	12,15,18,...	24+3	9,12,15,18,...	9	-	F2
9	....						

Virtual Spacing

- Approximation of GPS
  - Principle
    - associate one timestamp to each arriving packet
    - scheduler selects packet with smallest timestamp
  - Algorithm
    - D[i] : bandwidth associated to Queue[i]
    - V[i] : state variable associated to Queue[i]
    - V : state variable associated to the scheduler
      - at all time, V is equal to the timestamp of the packet being transmitted
    - Arrival of a packet of P bytes in Queue[i]
      - V[i] = max(V[i], V) + (P / D[i])
      - V[i] is associated to the arriving packet
    - Scheduler
      - select the packet with the smallest timestamp for transmission

Guarantees

- Schedulers supporting delay guarantees
  - delay guarantees are only available for token bucket (R,B) limited flows
    - but guarantee does not depend on behavior of other flows
  - Delay through a series of n schedulers (ignoring the fixed delays)
    - GPS $\frac{B}{R}$
    - WFQ / PGPS $\frac{B + n \times P_{max}}{R} + \sum_{j=1}^{n} \frac{P_{max}}{C_j}$
    - Virtual Clock $\frac{B + n \times P_{max}}{R} + \sum_{j=1}^{n} \frac{P_{max}}{C_j}$
    - Virtual Spacing $\frac{B + n \times P_{max}}{R} + \sum_{j=1}^{n} \frac{K_j \times P_{max}}{C_j}$

Traffic Control and QoS in IP Networks

- Applications and transport protocols
- Packet-level traffic control mechanisms
  - Best-effort service
  - Maximum bandwidth service
  - Minimum bandwidth service
  - Delay guarantees
  - TCP specific mechanisms
- Flow-level traffic control mechanisms
- Network-level traffic control mechanisms
- Standardized services

Traffic Control and QoS in IP networks
Packet level traffic control mechanisms