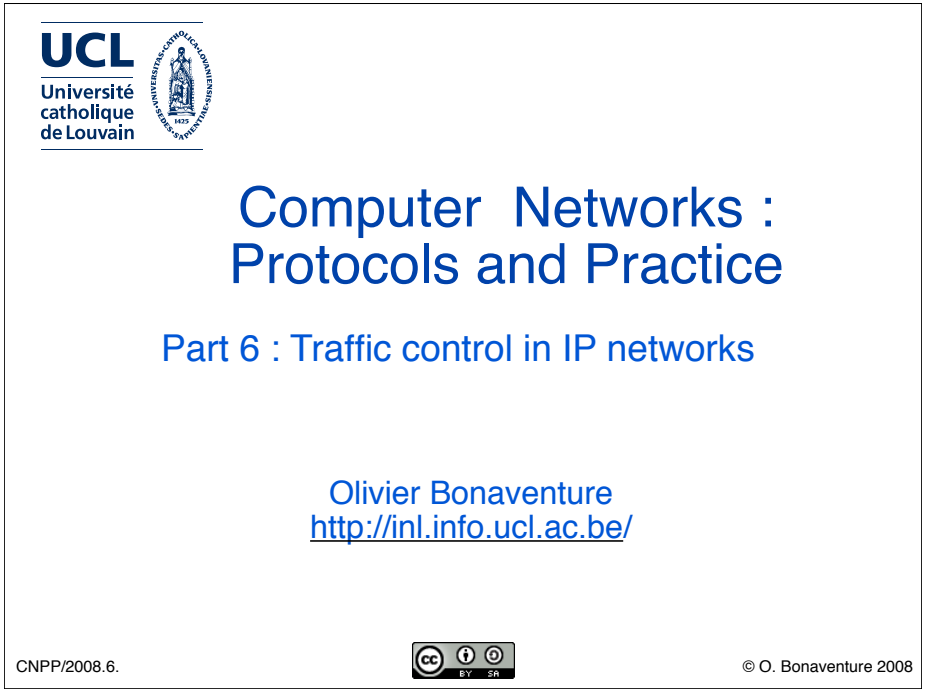


Course outline

- Objective
  - Understand the protocols and mechanisms that are required to support current and emerging applications in IP-based networks
- Topics covered
  - Traffic control and in IP networks
  - IPv6
  - IP Multicast
  - MultiProtocol Label Switching (MPLS)
  - Virtual Private Networks
  -

Traffic control in IP Networks
Outline

- Applications
- Packet-level traffic control mechanisms
  - Flow identification
  - Packet Marking
  - Buffer acceptance
  - Scheduling

Taxonomy of applications

- From traffic control point of view, applications can be divided in two main classes
  - Elastic or opportunistic applications
    - if resources are available, elastic applications will try to consume them
    - if resources are not temporarily available, elastic applications will wait without being severely affected
    - example : www, email, ftp, news, ...
  - Streaming applications
    - a minimum amount of resources is required for one streaming application
    - minimum amount is available, application works well
    - minimum amount not available, application doesn't work

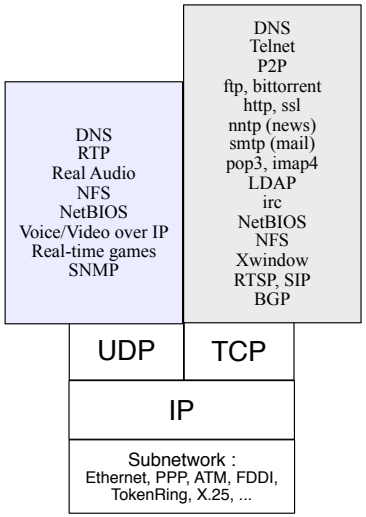
Elastic applications : a few examples

- Interactive reliable applications
  - telnet, tn3270, database access, short www transactions, ftp-control, Xwindow, ...
- Request-response applications
  - client server, NFS, Remote Procedure Calls, distributed computing, ...
  - P2P search for file
- Batch applications
  - ftp, remote backup, long http transactions, news transfers, ...
  - P2P transfers

Streaming applications : a few examples

- Conversational multimedia applications
  - voice or video over IP
- Interactive multimedia applications
  - distributed simulations, network games
- Non-interactive multimedia applications
  - distance learning, audio/video broadcasts, news on demand, ...
    - continuous flow of multimedia information

Applications and protocol stack



- Which transport protocol for which application ?
  - Rule of thumb :
    - UDP is suitable for
      - request-response applications (mainly LAN)
      - interactive multimedia applications (LAN and WAN)
      - interactive gaming
      - multicast capable
    - TCP used for reliability
      - file transfers
      - terminal emulation's
      - request-response in WAN/LAN
      - some multimedia applications
      - unicast only

UDP

- Objectives
  - provide an unreliable connectionless packet service over the unreliable IP service
- Mechanisms
  - simple packet format with optional checksum
  - communicating applications are identified by
    - source IP address
    - destination IP destination
    - source port number
    - destination port number
  - does not include mechanisms to ensure
    - reliable delivery
      - bit errors can be detected but not corrected
    - in-sequence delivery

CNPP/2008.6.

UDP was initially defined in [RFC768] in 1980 and was not modified since then. It relies on an eight bytes header that contains the following information :

- 16 bits source port number
- 16 bits destination port number
- 16 bits UDP length
- 16 bits UDP checksum

The maximum length of a UDP packet corresponds to the maximum length of an IP packet, namely 64 KBytes. Even if UDP supports large packets, most applications avoid to send UDP packets larger than the IP Maximum Transmission Unit .

TCP

- Objectives
  - provide a reliable connection-oriented byte stream service over the unreliable packet-based IP service
- Mechanisms
  - single packet format protected with checksum
  - connection establishment and release
    - a TCP connection is identified by the four-tuple
      - source IP address
      - destination IP address
      - source port number
      - destination port number
  - reliable data transfer
    - acknowledgements and retransmissions of lost packets
  - flow and congestion control

There is a large literature on TCP. Some interesting references include :

J.Postel. Transmission control protocol, protocol specification. Internet RFC 793, September 1981.
V.Jacobson. Congestion avoidance and control. In Proc. ACM SIGCOMM88, pages 314--329, August 1988.
W.Stevens. TCP slow start, congestion avoidance, fast retransmit and fast recovery algorithms. Internet RFC 2001, January 1997.
M.Allmanm, V.Paxson, and W.Stevens. TCP congestion control. Internet RFC 2581, April 1999.
V.Jacobson, B.Braden, and D.Borman. TCP extensions for high-performance. Internet RFC 1323, May 1992.
W.Stevens. TCP/IP Illustrated, volume 1 : The protocols. Addison-Wesley, 1994.
G.Wright and R.Stevens. TCP/IP Illustrated Vol. 2, The Implementation. Addison-Wesley, 1995.

S.Floyd. A report on recent developments in TCP congestion control. IEEE Communications Magazine, 39(4):84--90, April 2001.

TCP : summary

- Important points concerning TCP
  - TCP always tries to send large packets
    - bimodal packet size distribution
  - Maximum throughput is limited by window size
    - max throughput = ~ window / rtt
  - TCP's retransmission mechanisms
    - expiration of retransmission timer
      - always works, but may decrease performance
      - forces TCP to perform slow-start
    - fast retransmit
      - can recover from isolated segment losses
      - difficult to recover from losses of groups of segments
      - forces TCP to perform congestion avoidance

TCP : summary (2)

- TCP's congestion control...
  - Window-based flow control implies self-clocking
  - slow-start and congestion avoidance
  - TCP's congestion control does not always provide a fair bandwidth allocation
    - sources with a large rtt are “penalised”
    - sources with a smaller mss are “penalised”
  - Congestion is only detected when it has occurred
    - TCP must cause congestion to detect it
  - TCP's throughput is more affected by burst losses than by individual packet losses
    - expiration of retransmission timer is costly
    - network should avoid loosing bursts of packets

Internet traffic characteristics

- Which traffic fills Internet backbones ?
  - Study on MCI's backbone in April 1998
  - Traffic mix : transport protocols
    - TCP : 90% -95%, UDP : 5-10% of the traffic
  - Top TCP applications
    - www : 75% of the TCP traffic
      - mainly short lived TCP connections
      - average www TCP connection : 10-20 TCP packets
    - news : 7 % of the TCP traffic
      - mainly long-lived TCP connections (100-2000 packets)
    - ftp : 4% of the TCP traffic
      - mainly long-lived TCP connections (100-1000 TCP packets)
    - email : 3% of the TCP traffic
      - mainly short-lived TCP connections (10-20 TCP packets)
  - Top UDP application : DNS
    - audio/video is not yet a major application

For more information on this topic, see also

K.Claffy, G.Miller, and K.Thompson. the nature of the beast : recent traffic measurements from an internet backbone. In INET98, 1998. available from http://www.caida.org/Papers.
R.Koga and S.McCreary. Traffic workload overview. available from http://www.caida.org/Learn/Flow/tcpudp.html, June 1999.G.Miller, K.Thomposon, and R.Wilder. Performance measurement
K.Thompson, G.Miller, and R.Wilder. Wide-area internet traffic patterns and characteristics. IEEE Network Magazine, 11(6), November/December 1997. also available from http://www.vbns.net/presentations/papers.
National Laboratory for Applied Networking Research. Tutorial: Insight into current internet traffic workloads. available from http://www.nlanr.net/NA/tutorial.html, 1997.
A.Mena and J.Heidemann. An empirical study of real audio traffic. In INFOCOM2000, March 2000.
S.McCreary and K.Claffy. Trends in wide area IP traffic paterns : a view from Ames Internet Exchange. available from http://www.caida.org/outreach/papers/AIX0005/, 2000.

For enterprise traffic, see
A First Look at Modern Enterprise Traffic (30 min.)
Ruoming Pang, Princeton University; Mark Allman, International Computer Science Institute; Mike Bennett and Jason Lee, Lawrence Berkeley National Laboratory; Vern Paxson, International Computer Science Institute and Lawrence Berkeley National Laboratory; Brian Tierney, Lawrence Berkeley National Laboratory

Internet traffic characteristics (2)

- 1999-2000 study on Fix-West in California
  - OC-3 link carrying part of interswitch traffic
- Transport protocol mix
  - TCP : 91.5% of bytes, UDP : 5%, tunnels: 3%
- TCP Application mix
  - HTTP : 64 % of bytes
  - unidentified applications : 10% of bytes
  - News : 12%, FTP : 4.5%
  - Email : 4 %, Napster : 1%
- UDP application mix
  - Application unknown : 28 %
  - Realaudio : 24 %, DNS : 22 %
  - Games [Half life, Quake, Starcraft,...] : 20%

Internet traffic characteristics (3)

- Current Internet traffic
  - A 2.5 Gbps link on the Sprint backbone in 11/02
  - TCP/UDP mix
    - 94%/5% of packets carry TCP/UDP
    - 98%/1.5% of bytes carried by TCP/UDP
  - Application mix

	Packets (‘	Bytes (%)	Flows (%)
Web	62,45	80,89	39,26
File Sharing	5,07	4,61	10,86
FTP	0,37	0,54	0,56
Email	5,56	3,03	3,61
Streaming	0,87	0,69	0,95
DNS	0,57	0,14	3,29
Games	0,28	0,07	0,12
Other TCP	20,02	8,44	33,97
Other UDP	4	1,2	4,59
Not TCP/UDP	0,81	0,38	2,79

Source :

http://ipmon.sprintlabs.com/packstat/packetoverview.php

For a good study of file sharing (P2P) versus web, see :

Stefan Saroiu, Krishna P. Gummadi, Richard J. Dunn, Steven D. Gribble, Henry M. Levy: An Analysis of Internet Content Delivery Systems. Proceedings of 5th Symposium on Operating Systems Design and Implementation (OSDI) 2002, Boston, MA, USA, December 2002.

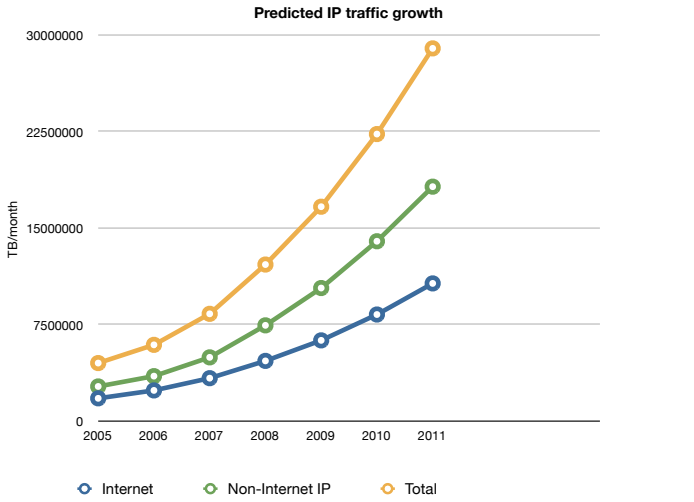
Internet traffic characteristics (4)

- Another 2.5 Gbps link on the Sprint backbone in 11/02
- TCP/UDP mix
  - 87.7%/11% of packets carry TCP/UDP
  - 96%/3.2% of bytes carried by TCP/UDP
- Application mix

	Packets (‘	Bytes (%)	Flows (%)
Web	32,32	32,83	23,4
File Sharing	22,7	23,34	21,3
FTP	1,11	1,64	0,51
Email	1,95	1,2	2,65
Streaming	3,22	2,78	2,12
DNS	1,18	0,27	5,51
Games	1,08	0,23	0,26
Other TCP	28,12	35,18	33,62
Other UDP	7,09	1,77	7,79
Not TCP/UDP	1,24	0,76	2,76

Source : http://ipmon.sprintlabs.com/packstat/viewresult.php?16:appsbreakdown:nyc-27.0-021121:

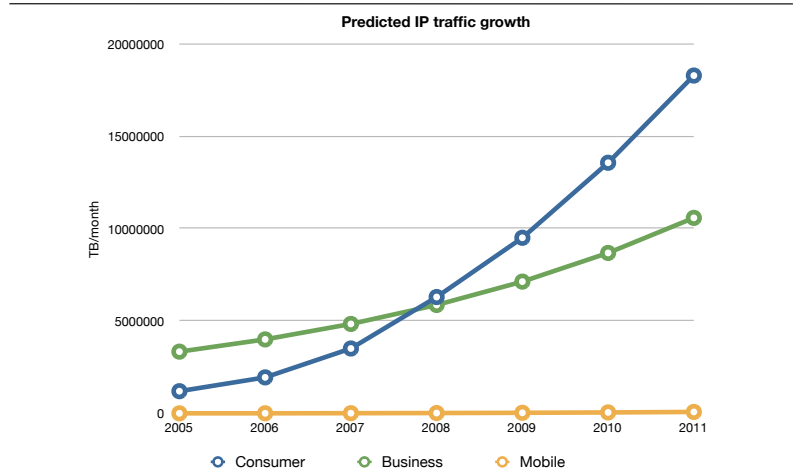
Predicted IP traffic growth



CNPP/2008.6.

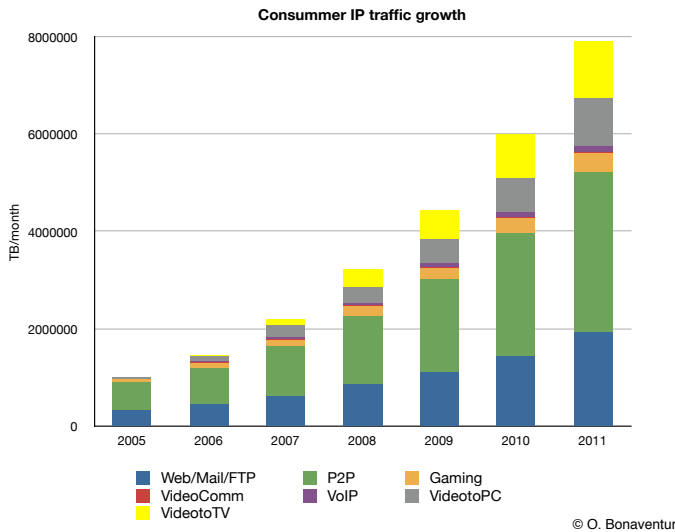
Source for this data Global IP traffic forecast and methodology, 2006-2011, Cisco 2007 white paper, [online version]
http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/net_implementation_white_paper0900aecd806a81aa.pdf

Predicted IP traffic growth



CNPP/2008.6.

Predicted IP traffic growth (3)



CNPP/2008.6.

Peer-to-peer file sharing

- Evolution of file sharing on the Internet
- Servers using `ftp` protocol
  - A single server that serves all files on disk
  - A set of mirror serves serving the same content
- The innovation introduced by Napster
  - How to distribute many files from many nodes ?
    - Keep the files on their source nodes
    - Central Napster server stores description and URL of each shared file
    - Users willing to obtain a file consult central server to obtain file URL and then download file from their respective source nodes
      - server remains simple and can index large number of files
      - server does not directly participate in file transfer

Peer-to-peer file sharing (2)

- Limitations of the Napster approach
  - a single server indexes all files
  - if a source node fails, then the ongoing file transfers must be restarted
    - completely or partially depending on the file transfer protocol begin used for the transfer
  - performance of file transfer is function of performance of the corresponding source node
    - if source node is connected via ADSL, performance will be severely limited
- How to improve ?
  - Divide the file in blocks
  - Each block can be served by multiple nodes
    - provides redundancy
  - Download from several nodes at the same time
    - one TCP connection may be slow and others faster

CNPP/2008.6.

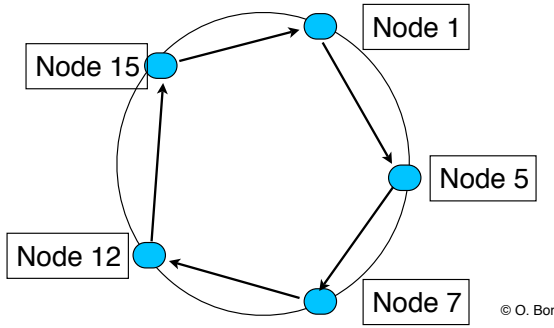
Today, Napster does not work anymore as explained due to copyright violations reasons.

One of the most efficient file transfer protocol used today is Bittorrent. Bittorrent also divides files in blocks and allows files to be downloaded from several nodes at the same time. This provides good redundancy in case of node/link failures, but also allows an efficient utilisation of the available link bandwidth by using uncongested paths (the node with the highest bandwidth will automatically serve blocks faster than a congested node). A Bittorrent node will not necessarily receive blocks in sequence. Furthermore, to ensure that all Bittorrent users contribute to the system, Bittorrent implementations apply the tit-for-tat principle which implies that once a node has received a block, it must serve this block to other nodes before being allowed to download new blocks.

Additional information about the Bittorrent protocol may be found in

Distributed Hash Table based P2P

- How to scale file sharing to a very large number *n* of nodes ?
- Principle of the solution
  - Use a hash function such as SHA-1
  - Each node has one identifier, id=hash(IPaddress) and a pointer to its successor on the Chord ring



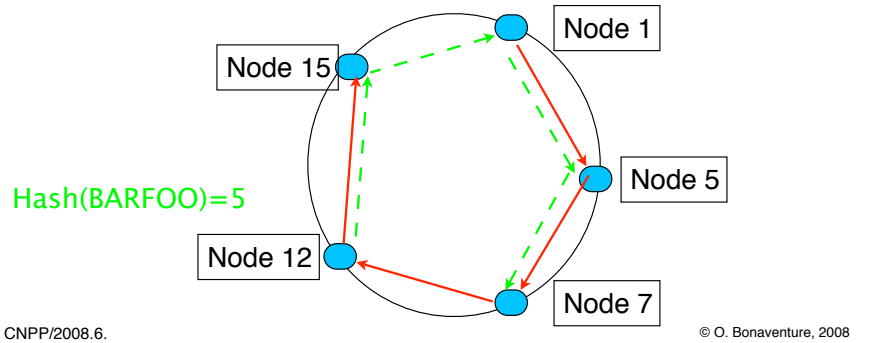
CNPP/2008.6.

Several Distributed Hash Tables have been proposed in the literature. One of the first and most influential ones is Chord

Ion Stoica, Rober t Morris, David Karger, M. Frans Kaashoek, Hari Balakrishnan , " Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications" ACM SIGCOMM 2001

How to store files ?

- Principle
  - File FOOBAR is stored on the node whose id is the successor of hash("FOOBAR") on the ring
  - A node on the ring uses its successors to find the responsible node for a given file
    - Examples

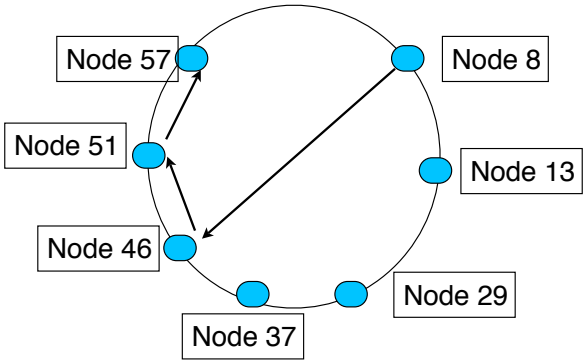


How to find files faster ?

- Performance of file lookup
  - O(n)
    - worst case is to follow linked list of n nodes
- How to improve ?
  - Allow nodes to know addition pointers to other nodes on the Chord ring to speedup lookup
    - m = number of bits in the key/node identifiers
    - Each node maintains routing table of m entries
    - The ith entry in the table at node n contains the identity of the first node, s, that succeeds n by at least 2^i–1 on the identifier circle, i.e., s = successor(n + 2^i–1)
      - arithmetic modulo 2^m is used
    - A finger table entry includes both the Chord identifier and the IP address (and port number) of relevant node.

Scalable lookup with Chord

- Example
  - m=8



N8+1	N13
N8+2	N13
N8+4	N13
N8+8	N29
N8+16	N29
N8+32	N46

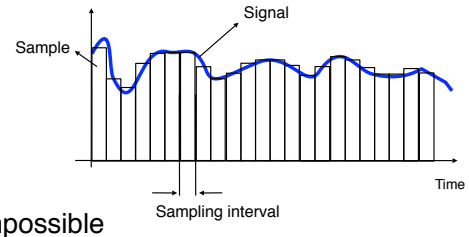
- How to find key 53 from node 8 ?

Audio : a closer look

- Physical characteristics
  - sound is composed of acoustical waves travelling through a physical medium (air)
  - An acoustical wave is characterised by its
    - frequency
      - expressed in Hertz [Hz]
    - amplitude or power conveyed by the wave
      - expressed in decibels
- Limitation
  - human ear is only sensitive between
    - 20 Hz and 22000 Hz
  - If we are only interested by humans, we can neglect frequencies outside this range
    - but multimedia applications for dogs would have different requirements ...

Audio : from sound to bits

- How can we digitise sound
  - Nyquist theorem
    - any signal can be digitised by sampling provided that exact samples are taken at least at twice the highest frequency of the signal



  - Quantification
    - exact samples are impossible in practice and cannot be transmitted
    - we accept a small degradation of the signal quality by using N bits to represent each sample

Audio : from sound to bits (2)

- Telephone
  - human voice comprises sounds in the frequency range 0-4000 Hz
  - historically, phone lines have been artificially limited by low-pass filters in this frequency range to improve transmission of signals on long links
    - although the telephone invented by Bell was initially sold as a way to remotely listen to Opera...
  - Example : G.711 (PCM)
    - one 8 bits sample every 125 microsec -> 64 kbps

Audio : from sound to bits (3)

- Music
  - Good quality music should convey the complete signal (0 - 22000 Hz) for human ear
    - lower qualities (e.g. AM radio) are possible with lower frequency ranges
  - Samples should be accurately quantified
  - Stereo means two different channels

- Example : CD
  - Sampling frequency : 44 kHz
  - 16 bits per sample
  - two separate channels for CD-quality stereo music
  - Required bandwidth : 44000*16*2=1.4 Mbps

Audio : reducing bandwidth requirements

- Nyquist based digitisation provides an almost perfect encoding of the signal
  - drawback is the relatively high required bandwidth
- Better solution would be to encode signal into an equivalent signal by taking into account the characteristics and imperfections of the human ear
  - audio compression
    - reduce the number of transmitted samples
    - reduce the number of bits per sample
    - map audio signal onto voice/music models and transmit model parameters instead of samples

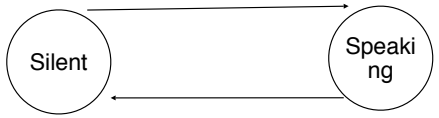
A few common audio compression schemes

Name	Sampling freq. (kHz)	Bit rate
MPEG L3	22.05	44.1 48..128 kbps
G.711	8.0	64 kbps
G.721	8.0	32 kbps
G.722	16.0 (7 kHz spectrum)	64 kbps
G.723.1	8.0	5.3 and 6.3 kb/s
G.726	8.0	16, 24, 32, 40 kb/s
G.728	8.0	16 kb/s
G.729	8.0	8 kb/s

Source : RTP FAQ, maintained by Henning Schulzrinne (http://www.cs.columbia.edu/~hgs)

Audio : silence suppression

- A closer look at the behaviour of phone users
- Average phone conversation
  - two persons are speaking
    - a single person is speaking during 64 to 73% of time
    - both persons are speaking together during 3 to 7% of time
    - nobody is speaking during 33 to 20% of time
  - Simple model of someone on the phone



Mean time: 1.8 sec Mean time: 1.2 sec

  - Avoiding to transmit samples during silences can reduce the required bandwidth down to 60%

Audio : lossy compression

- Human ear
  - a high amplitude signal at frequency *f* will completely mask lower amplitude signals whose frequency is close to f during some time
    - masked signals can be neglected without any loss of audio quality
- Audio signal
  - amplitude does not change quickly from one sample to the next one
    - fewer bits can be used to encode each sample
- Define models for human voice or music
  - fit audio signal to model and select appropriate parameters and/or model
  - only transmit parameters and model id

Issues for multimedia transmission

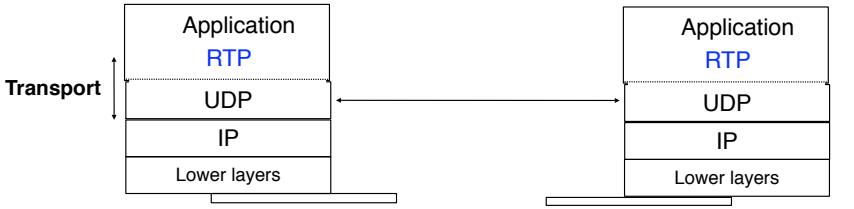
- How to packetise multimedia information?
  - samples are a few bits with audio
  - a single image can be very large for video
- How does a destination distinguish different medias and encoding ?
- How to recover from the limitations of IP
  - packet loss
  - packet reordering
  - packet duplication
- How can a source know that her information is received correctly
- How to support multiparty communications ?

Transporting multimedia information

- Can we rely on TCP ?
  - TCP provides a reliable transfer, but its usage of retransmissions and timeouts may cause large delays when losses occur
    - audio and video can easily survive with limited losses
  - TCP does not support multicast
- Initially, TCP was rarely used by multimedia applications
  - except when the application transfers "files"
- Today, some multimedia applications rely on TCP,
  - e.g. Skype, youtube, distribution of Video on Deman movies
  - main advantage of TCP : NAT and firewall traversal

Transporting multimedia information (2)

- UDP-based multimedia distribution
  - UDP supports multicast but lacks mechanisms to
    - reorder information received out of sequence
    - detect losses
    - recover from delay variations
    - identify multimedia information inside UDP segments
- Solution chosen by IETF
  - Real-Time Transport Protocol (RTP) above UDP

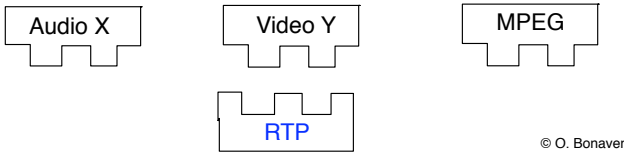


CNPP/2008.6.

RTP is used by most conversational multimedia applications such as interactive Voice or Video over IP

RTP

- RTP can be considered as a framework over which various applications can be built
  - RTP provides the basic mechanisms needed by most multimedia applications
    - two sub protocols
      - RTP which deals with the flow of data packets
      - RTCP which "controls" the flow of data packets
      - RTP uses even UDP port, RTCP uses next UDP port number
    - another protocol is used to establish session
  - applications may add additional mechanisms above RTP if needed



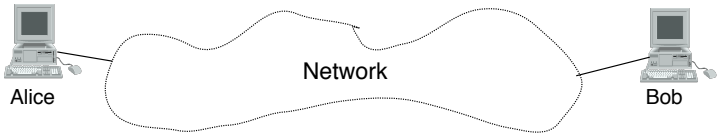
Source for this figure : W. Stallings, High-Speed networks : TCP/IP and ATM design principles, Prentice Hall, 1998

RTP is defined in
RFC1889 RTP: A Transport Protocol for Real-Time Applications. Audio-Video Transport Working Group, H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson. January 1996.
RFC1890 RTP Profile for Audio and Video Conferences with Minimal Control. Audio-Video Transport Working Group, H. Schulzrinne. January 1996.

For more information about RTP, a good webpage is
http://www.cs.columbia.edu/~hgs/rtp/

A simple streaming application

- Voice over IP
  - Alice wants to call Bob on his PC-phone



- Three different problems
  - Establish the voice conversation between Alice and Bob
    - find the IP address suitable to reach Bob
    - negotiate audio compression scheme...

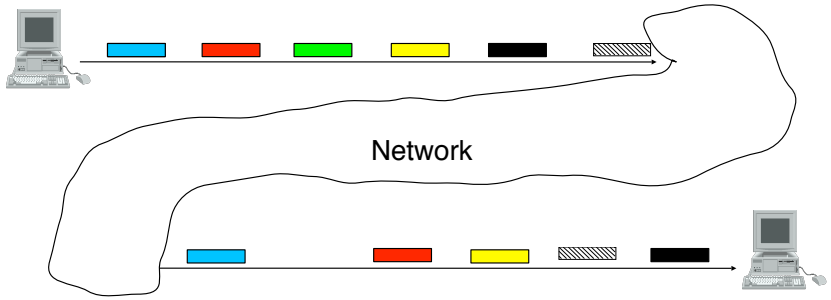
    Signalling and session establishment Protocols
  - Once session is established, transmit audio packets

    RTP
  - For time to time, provide feedback about quality of reception

    RTCP

Dealing with losses and reordering

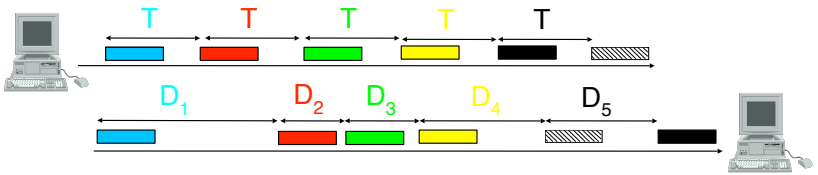
- Problem
  - RTP segments can be lost or reordered on their way towards the destination



- Classical solution
  - insert sequence number inside RTP header since UDP does not contain any sequence number

Dealing with delay variations

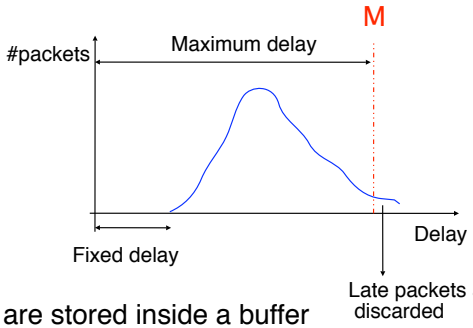
- Problem
  - Synchronous signal : one packet every T seconds
  - Network may introduce variable delays even when no packets are lost
    - for example some packets may have to wait in the buffers of an intermediate router



  - How to recover synchronous signal at destination?

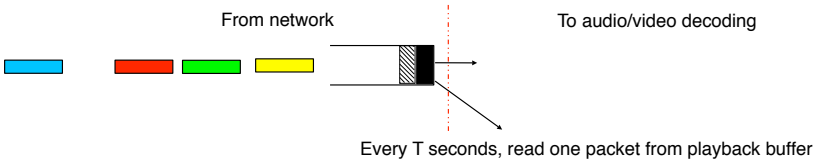
Dealing with delay variations (2)

- Solution
  - introduce additional delay at destination



  - playback buffer
    - incoming packets are stored inside a buffer
    - one packet is read by the application every T seconds
    - the size of the buffer is chosen to emulate a fixed transmission delay of M seconds
      - packets whose transmission delay is larger than M seconds are discarded inside the playback buffer
      - M is compromise between low delay and low losses

Dealing with delay variations (3)

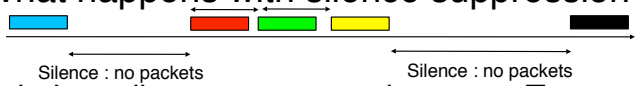


- How to deal with reordered packets ?
  - Sequence number is sufficient to reorder them inside buffer if they are not too late

- How to deal with lost packets ?
  - Sequence number is sufficient to detect losses and lost packets can be replaced by dummy packets if needed

Dealing with delay variations (4)

- What happens with silence suppression ?



  - during talk spurts, one packet every T seconds
  - during silences, no packet

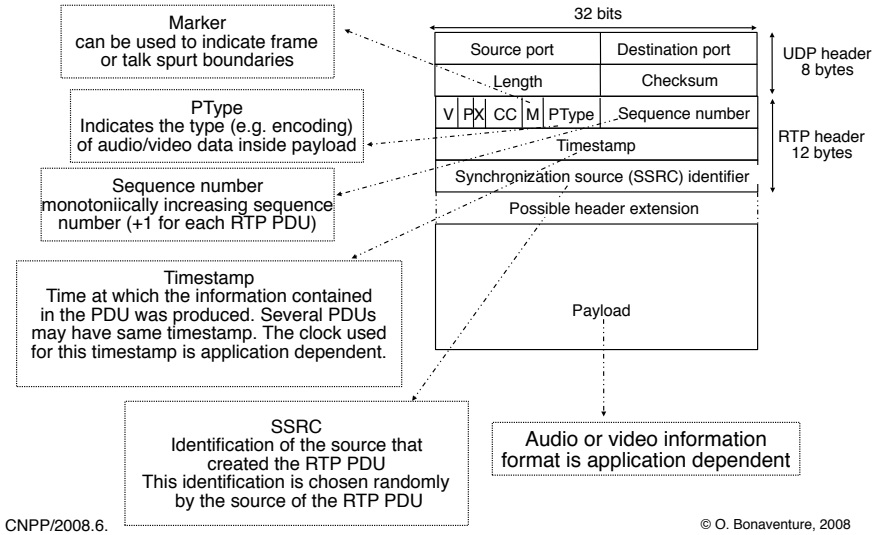
- Problem
  - an expected packet can be late because either
    - the packet was delayed inside the network
    - no packet was generated due to silence suppression
  - when does the destination restart playback after a period of silence ?
- Solution
  - insert timestamp inside each packet

Packetising audio

- Unit of information : audio sample
  - First solution
    - each sample is placed inside one RTP PDU
    - overhead would be prohibitive with one byte per sample and 20 bytes of IP header, 8 bytes of UDP header and 12 bytes of RTP header
  - Better solution
    - N samples are placed inside one RTP PDU
      - we need `N*sampling_interval` seconds to prepare a complete RT PDU
      - Packetisation may cause a large delay inside source
        - 64 Kbps voice with 10 samples per packet
        - packetisation delay : 1.25 msec
        - overhead RTP/UDP/IP : 80% (40 bytes out of 50)
      - the chosen value for N is always compromise between overhead and packetisation delay
        - common choice is to put 20 msec of audio in one packet

A common packet size for audio is to pack inside one packet 20 milliseconds worth of audio, but larger values are sometimes used as well, depending on the amount of overhead that can be tolerated.

RTP/UDP PDU format



CNPP/2008.6.

Registered payload types are defined in RFC1890 and the updated list is available at http://www.isi.edu/in-notes/iana/assignments/rtp-parameters

RTCP

- Three main objectives
  - Quality of service and congestion control
    - RTCP segments can be sent by a receiver as a kind of low frequency ACK segments that are used to indicate the quality of the reception
    - based on the receiver reports, the sender may adapt its encoding (e.g. lower bandwidth during congestion)
  - Identification
    - provide more information about the application/user that are sending RTP segments
  - Estimate the number of participants in multicast sessions
    - RTCP feedback should only consume a fraction of the bandwidth consumed by RTP
    - for multicast session, an estimate of the number of receivers is needed to limit the RTCP bandwidth

RTCP (2)

- RTP entities regularly send reports to provide feedback about quality of transmission
- Sender report
  - used by a sender of RTP packets to summarize the amount of information it has sent recently
  - each report contains
    - absolute timestamp of the report
      - indicates the absolute transmission time of the report
      - 64 bits NTP timestamp
    - timestamp relative to the flow of RTP packets
      - indicates the position of the report in the RTP flow
    - amount of data sent since beginning of session
      - total number of RTP segments sent
      - total number of octets sent

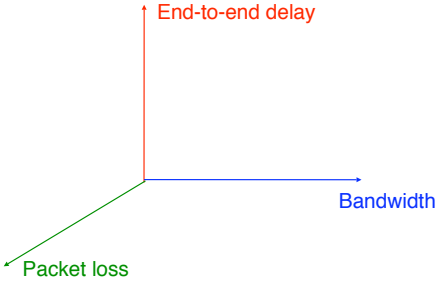
RTCP (3)

- Receiver reports
  - used to inform senders about the quality of reception
  - one report for each source (SSRC) heard
  - Each receiver report contains
    - indication of source (SSRC)
    - timestamp of last sender report received from this source
    - delay (seconds) since last sender report from this source
    - highest RTP sequence number received from this source
    - number of lost RTP packets for this source
    - fraction of lost RTP packets for this source
    - estimation of delay jitter for RTP packets from this source

RTCP (4)

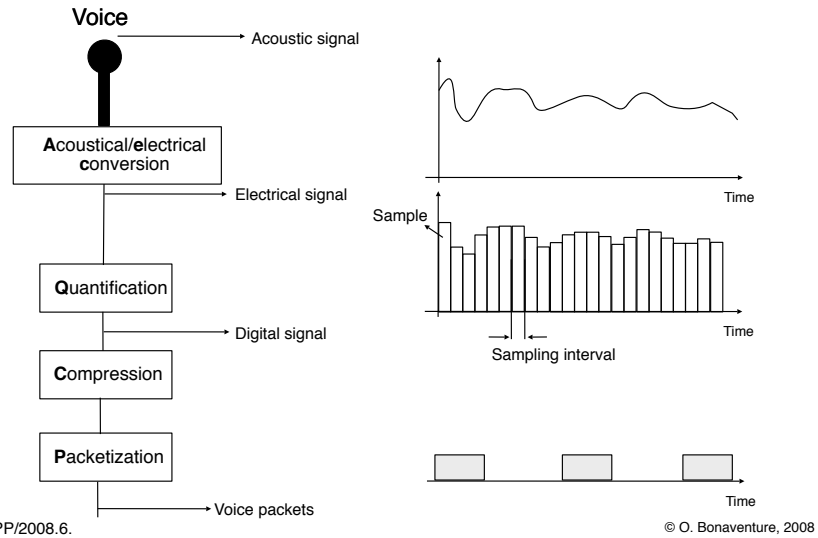
- Additional features
  - Information about applications/user
  - Source description (SDES) RTCP segments
    - can be used by senders or receivers in a session
    - CNAME : identification of user
      - typically user@host.domain
    - NAME
      - real name of the person using the application
    - EMAIL
    - PHONE
    - LOC
      - geographic location of user
    - TOOL
      - application sending RTP segments
    - NOTE
      - any comment

Application requirements



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

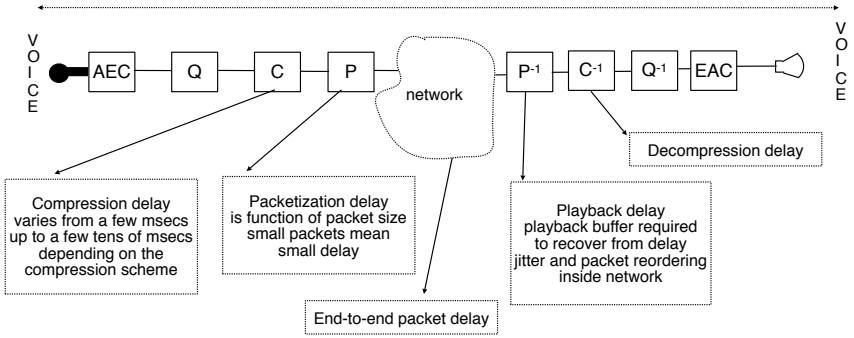
Example streaming application packetized voice



CNPP/2008.6.

End-to-end packetized voice delay

- Requirements for interactive voice
  - end-to-end (mouth-to-ear) delay should not be much larger than around 100 msec
    - end-to-end delay is composed of many delays



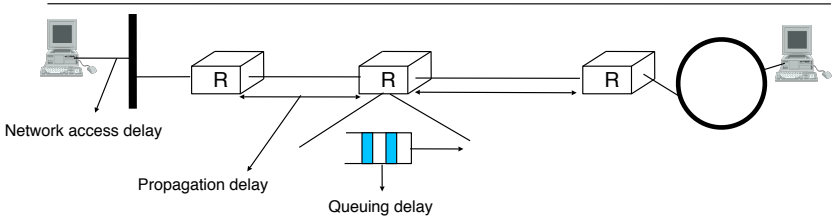
CNPP/2008.6.

Sample voice compression standards

G. 711 : 64 Kbps, compression delay : 0.75 msec
G.726 : 32 kbps, compression delay : 1 msec
G. 728 : 16 kbps, compression delay : 3-5 msec
G. 729 : 8 kbps, compression delay : 10 msec
G. 723.1 : 6.3 kbps, compression delay : 30 msec

Source : F. Tobagi, Multimedia Networking, IBM Chair, VUB, December 1998

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 microsec to travel 1 kilometer
  - transmission delay [fixed]
    - a packet of P bytes needs $\frac{P * 8}{B}$ 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

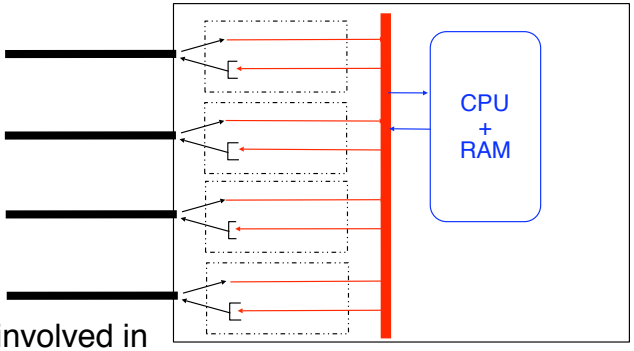
Applications : summary

- Two classes of applications
  - Elastic applications
    - rely on TCP
    - dominant applications in today's Internet
  - Streaming applications
    - rely on RTP/UDP
    - negligible in today's Internet backbones
- Application requirements
  - delay
  - (and delay jitter)
  - bandwidth
  - loss ratio

Traffic control in IP Networks
Outline

- Applications
- Packet-level traffic control mechanisms
  - Flow identification
  - Packet Marking
  - Buffer acceptance
  - Scheduling

First generation router



- CPU involved in
  - Packet reception (Interface -> RAM)
  - Packet queuing (in RAM)
  - Packet forwarding
  - Routing protocols to maintain forwarding table
  - Packet transmission (RAM -> interface)

Examples of first generation routers include

Cisco 2500 series

http://www.cisco.com/en/US/products/hw/routers/ps233/ps235/index.html

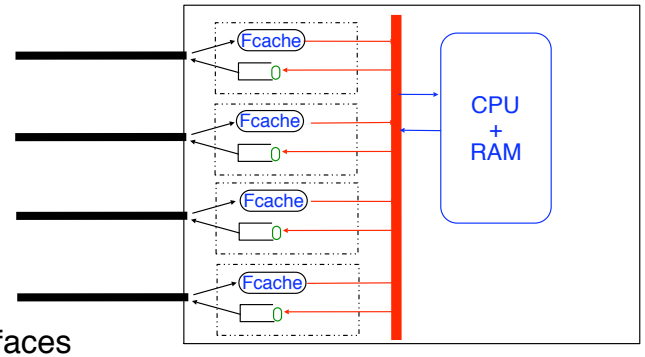
Cisco IGS series

http://www.cisco.com/en/US/products/hw/routers/ps986/index.html

Wellfleet' Link Node

Current Linux/FreeBSD based routers behave like that as well

Second generation routers

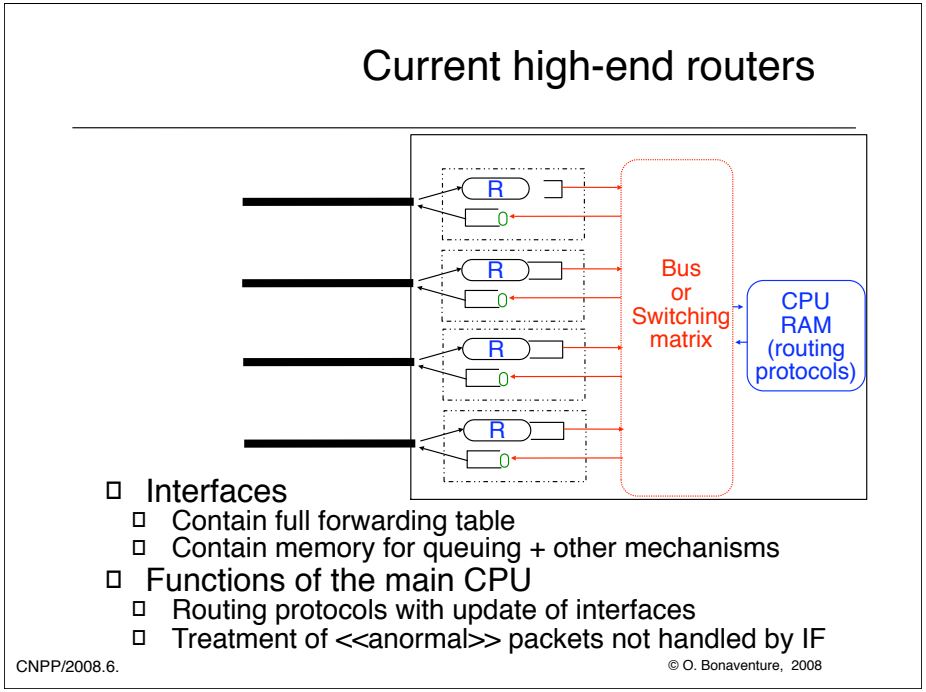


- Interfaces
  - Packets can be directly sent from input IF to output IF
  - Forwarding cache allows to bypass CPU
- Functions of the main CPU
  - Routing protocols with update of Forwarding cache
  - Treatment of packets not handled by Forwarding cache

CNPP/2008.6.

Examples :

Cisco 7200
http://www.cisco.com/en/US/products/hw/routers/ps341/index.html



In this tutorial, we place, for pedagogical reasons, the traffic control functions on the router's output ports. It should however be noted that one some router architectures, some of these functions may be placed elsewhere.

Example of high-end routers

Alcatel R7770

http://www.alcatel.com/products/productsummary.jhtml?_DARGS=/common/opg/products/include/productbrief.jhtml_A&_DAV=/x/opgproduct/a7770obx.jhtml

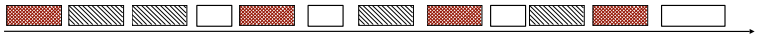
Juniper M160
http://www.juniper.net/products/ip_infrastructure/m_series/index.html

Cisco 12000

http://www.cisco.com/en/US/products/hw/routers/ps167/index.html

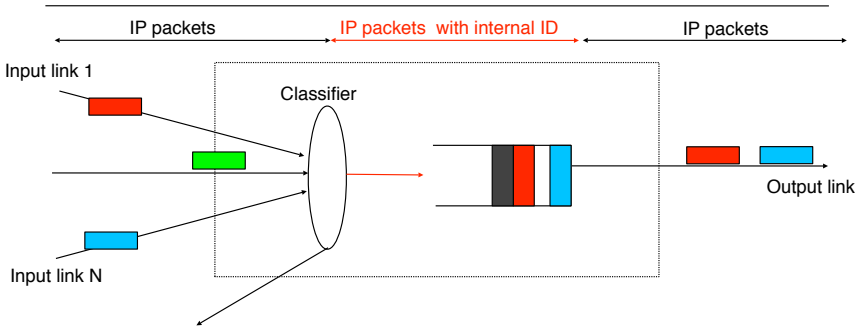
What is a flow ?

- 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 exists 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 v1



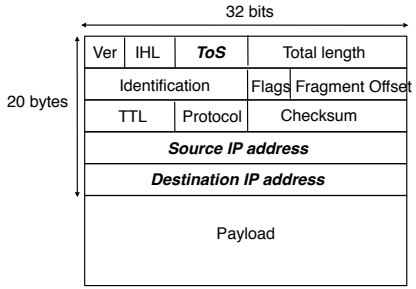
- Roles of the classifier
  - identify the flow to which an arriving packet belongs
    - identification can require complex operations
  - store this information internally so that other parts of the router will easily determine the flow of a packet
    - classification should be done at most once in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

Layer-two flow

- Some layer 2 technologies provide specific flow identification mechanisms
  - routers may carry flow identification but using several layer-2 flows to a given destination
  - ATM
    - VCI/VPI
  - Frame-relay
    - DLCI
  - 802.x LANs
    - VLANs, 802.1q

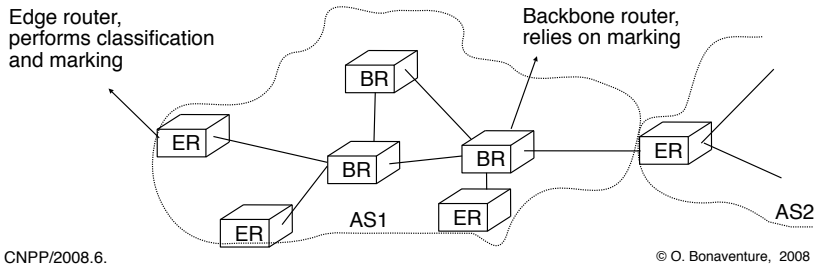
Layer-three flow

- 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



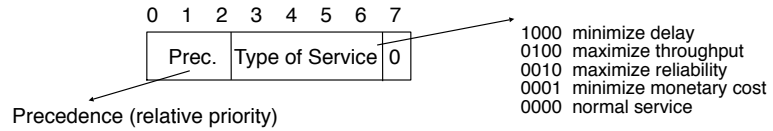
Layer-three flow (2)

- 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

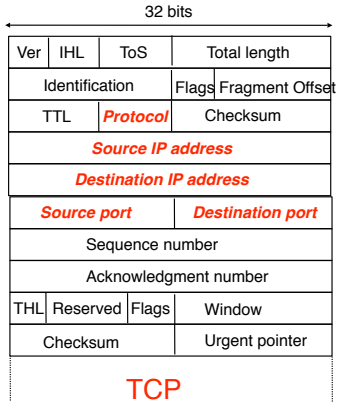
- How can we mark an IP packet ?
  - Steal one field of the IP header
    - ToS : Type of Service Octet
    - defines the relative importance of the IP packet and the type of service required for this packet

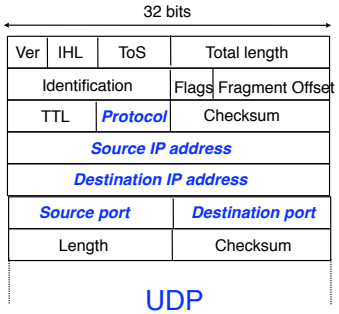


  - current status
    - definition of ToS Octet changed several times
    - Precedence is used in some networks
    - ToS field is rarely used
  - Using the ToS Octet for marking
    - advantage : easy to implement
    - disadvantage : limited number of marked flows

Layer-four flow

- Layer four flow identified by quintuple
  - source IP address, destination IP address, Protocol, source port, destination port





Identifying applications

- Simple solution
  - Look at TCP/UDP port numbers

Application	Transport	Port number
DHCP/Bootp	UDP	67, 68
DNS	TCP/UDP	53
HTTP	TCP	80
HTTP	TCP	443
IMAP	TCP/UDP	143, 220
LDAP	TCP/UDP	389
MS-SQL	TCP	1433
NetBIOS	TCP	137, 139
NFS	TCP/UDP	2049
NNTP	TCP/UDP	119
Lotus Notes	TCP/UDP	352
POP	TCP/UDP	109/110
SMTP	TCP	25
SNMP	TCP/UDP	161, 162
SSH	TCP	22
Syslog	UDP	14
Telnet	TCP	23
X Windows	TCP	6000-6003

IANA maintains a list of well know ports, see http://www.iana.org
On most Unix machines, such a list is also present in the /etc/services file

Identifying applications (2)

- The problems
  - Not all applications use well-known port numbers
    - FTP
      - server and client may negotiate other non-default port numbers than 20/21 for some file transfers
    - P2P and other new applications
      - rarely utilise well-known port numbers
  - Deployment of encrypted tunnels (IPSEC, L2TP, PPTP) hide TCP/UDP headers to 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
- Accurately identifying apps is difficult and often requires specialised hardware that looks at packet content
  - but encryption is more and more used by P2P apps

An analysis of P2P traffic by IPOQUE in 2007, see Internet Study 2007, http://www.ipoque.com, revealed that for edonkey and bittorrent 20% of the traffic was encrypted

Traffic control in IP Networks
Outline

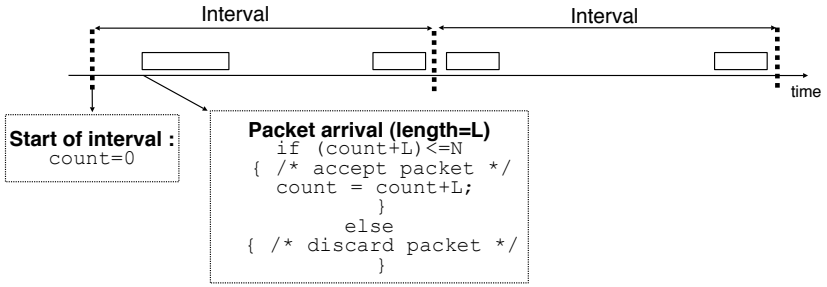
- Applications
- Packet-level traffic control mechanisms
  - Flow identification
  - Packet Marking
  - Buffer acceptance
  - Scheduling

Packet marking algorithms

- Objectives
  - Identify inside the network packets that belong to a class of applications and should receive some kind of service
    - Marking is performed by classifier
  - Enforce a traffic contract for a customer or flow
    - Use different markings for conforming and non-conforming packets
    - Example contracts
      - Maximum bandwidth : 1 Mbps
      - Average bandwidth : 100 kbps

Measuring the rate of a flow

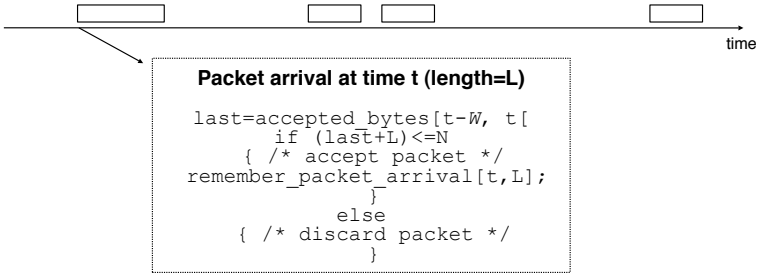
- Simple solution
  - time divided in fixed length intervals
  - Traffic contract defined as N bytes per interval



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

Measuring the rate of a flow (2)

- 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

Time sliding window

- Time sliding window
  - remembering packet arrivals is not possible
    - estimate average rate
    - on packet arrival, assume that flow was fluid and sending at estimated average rate during the last *W* seconds (W : sliding window)

Initialisation

```
last_arrival=0;
Avg_rate=
Contract_Rate;
```

Packet arrival at time t

```
Est_rate=(Avg_rate*W + L) / [(t-last_arrival) +
                            W]
       if (Est_rate <= Contractual_Rate)
              { /* accept packet */
                 Avg_rate=Est_Rate;
                   last_arrival=t
                       }
                   else
             { /* discard packet */  }
```

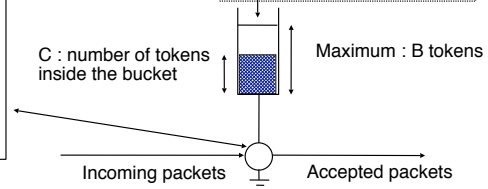
  - drawbacks
    - depends on initialisation time, which value for W ?

Token Bucket

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

Bucket filling
Initialisation
```
 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 accepted */
   C=C-L;
}
else
{
  /* packet is discarded */
}
```



- 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 (2)

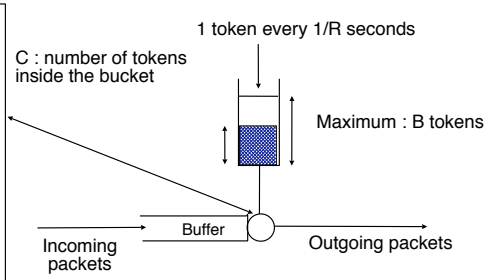
- Advantages
  - can be used by a network provider to enforce a traffic contract since it provides a precise algorithmic definition for
    - conforming packets
    - non-conforming packets
  - provides a bound on the average rate
  - provides a bound on the maximum amount of traffic during any period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

Token Bucket in shaping mode

- Problem
  - How can we ensure that one particular flow is conforming to a contract ?
    - utilise modified token bucket in shaping mode (shaper)

Arrival of packet of size L

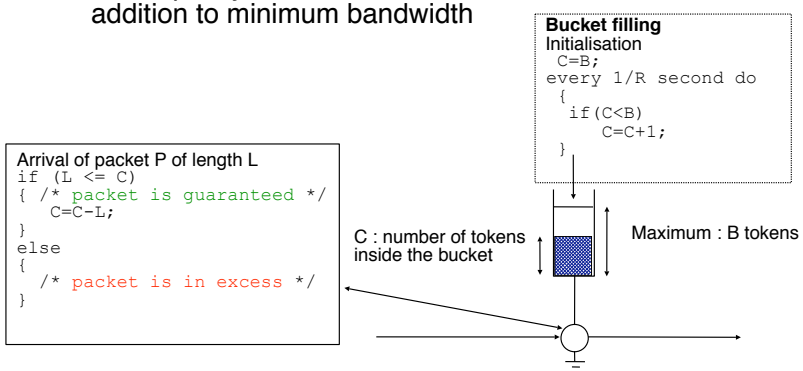
```
if (L <= C)
{ /* packet arrived on time */
   C=C-L;
   transmit_packet();
}
else
{/* packet arrived too early
  * delay packet inside buffer
* until it becomes conforming
*/
 while (C<L)
  { /* wait */  }
 /* now C=L and packet is
conforming */
 C=C-L;
 transmit_packet();
}
```



Traffic at the output of the shaper is conforming with R,B traffic contract

Deterministic marking

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

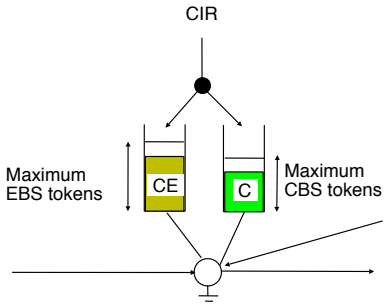


This marker can also be modified to support more than three types of packets.

See J. Heinanen and R. Guerin, A Single Rate Three Color Marker, RFC 2697, Sept. 1999
J. Heinanen and R. Guerin, A Two Rate Three Color Marker, RFC 2698, Sept. 1999

Extensions to token bucket

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



Bucket filling

```
Initialisation
 C=CBS;
 CE=EBS;
every 1/CIR second do
 {
  if(C<CBS)
     { C=C+1; }
  else if (CE<EBS)
        { CE=CE+1 }
       else
       { /* nothing */ }
 }
```

Arrival of packet P of length L

```
if (L <= C)
{ /* packet is green */
   C=C-L; }
}
else if (L <= CE)
{ /* packet is yellow */
      CE=CE-L; }
else
{ /* packet is red */ }
```

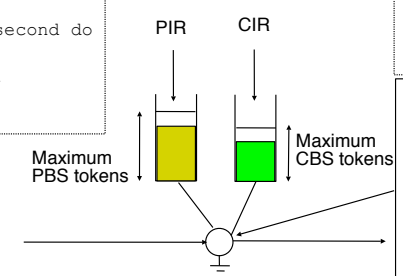

See also
O.Bonaventure and S.De Cnodder. A rate adaptive shaper for differentiated services. Internet RFC2963, October 2000.
for a shaper that can be used to improve the performance of TCP with such markers

Cisco routers have a different way to implement this kind of token bucket with two burst sizes. See
S. Vegesna, IP Quality of Service, Cisco Press, 2001

Extensions to token bucket (2)

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

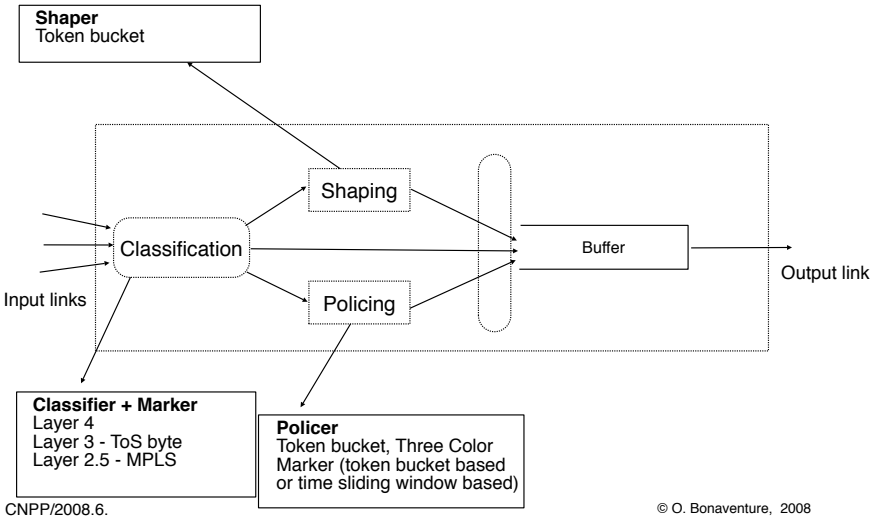
Peak Bucket filling
Initialisation
```
 CP=PBS;
every 1/PIR second do
 {
  if(CP<PBS)
     CP=CP+1;
 }
```



Committed Bucket filling
Initialisation
```
 C=CBS;
every 1/CIR second do
 {
  if(C<CBS)
     C=C+1;
 }
```

Arrival of packet P of length L
```
if (L > CP)
{ /* packet is red }
else if ( L > C )
   {/* packet is yellow */
      CP=CP-L; }
   else
   { /* packet is green */
     CP=CP-L;
     C=C-L;
   }
```

Improved router



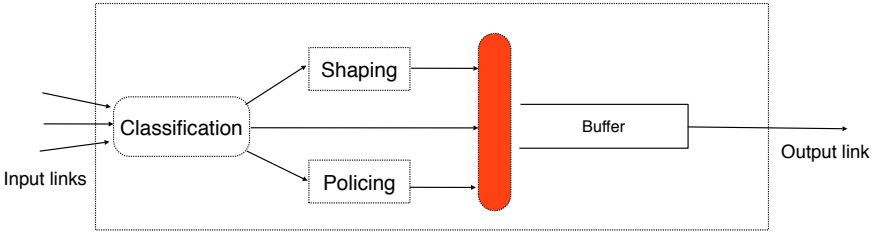
CNPP/2008.6.

Traffic control in IP Networks
Outline

- Applications
- Packet-level traffic control mechanisms
  - Flow identification
  - Packet Marking
  - Buffer acceptance
  - Scheduling

Improved router

- Packet treatment inside router's output port



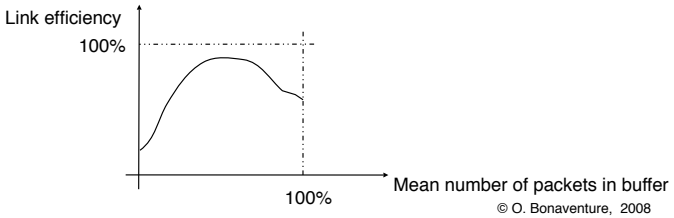
Buffer acceptance algorithm
When a packet arrives from an input link, the buffer acceptance algorithm decides whether this packet can be accepted inside the router's buffer

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 (2)

- Objectives
  - control the amount of packets in the buffer to
    - efficiently support best-effort traffic
      - should provide a fair utilisation 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 utilisation 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 (RED) was proposed in
S. Floyd, V. Jacobson, Random Early Detection gateways for congestion avoidance, IEEE/ACM Trans. Networking, V1, N4, 1993, pp. 397-413

Its utilisation is recommended in
Braden et al., Recommendations on Queue Management and Congestion Avoidance in the Internet, RFC 2309, April 1998

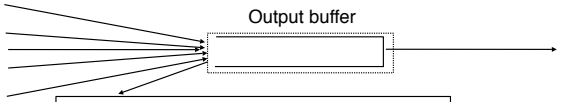
See also : S. Floyd. Red (Random Early detection) queue management. available from http://www.aciri.org/floyd/red.html

Random Early Detection (2)

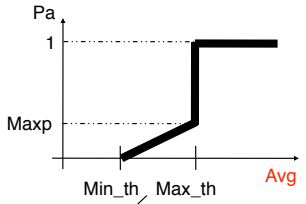
- Principle
  - How can we detect congestion ?
    - measure average buffer occupancy by using a low-pass filter
    - buffer is considered congested when its average occupancy is above a configured threshold
      - threshold value usually around 10%- 20% of buffer size
  - What do we do in case of 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

Random Early Detection (3)

- 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 buffer occupancy between Min_th and Max_th

The computation of the average queue occupancy is based on a low-pass filter . On each packet arrival, the average is computed as :
average= (average * (1-wq)) + (buffer_occupancy * wq)
where wq < 1 and usually of the form 1/2n for implementation reasons

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 min_{th}, max_{th}, max_p and averaging function (w_q) 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

CNPP/2008.6.

Some papers in favor of RED

S. Doran. Red analysis. available from http://adm.ebone.net/~smd/red-1.html , 1998.
B. Reynolds. RED analysis for congested network core and customer egress. In Presented at NANOG, January 1999. available from http://engr.qual.net/papers/reddraft.html.

Some papers not really in favor of RED

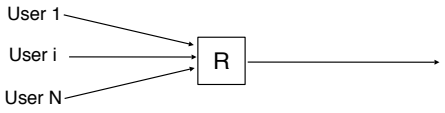
M. May, J. Bolot, C. Diot, and B. Lyles. Reasons not to deploy RED. In IWQoS'99, London, June 1999. G. Iannaccone, M. May, and C. Diot. Aggregate traffic performance with active queue management and drop from tail. ACM Computer Communications Review, 31(3):4--13, July 2001.
M. Christiansen, K. Jeffay, D. Ott, and F. Donelson Smith. Tuning RED for web traffic. In ACM SIGCOMM2000, August 2000.

A description of FRED

D. Lin and R. Morris. Dynamics of random early detection. In SIGCOMM 97, pages 137--145, Cannes, France, September 1997.

Characterising best-effort service

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

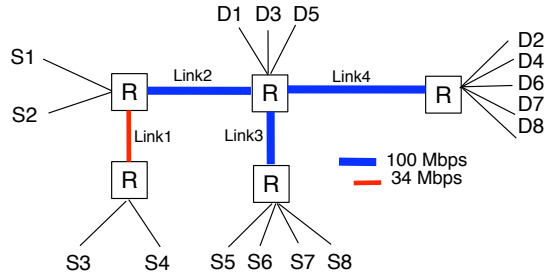


    - Fairness definition for a complete network
      - Maximise bandwidth received by users ?
      - Maximise utilisation of network resources ?

Max-min fairness

- Fairness definition for networks
  - a max-min allocation of bandwidth is an allocation of bandwidth which maximises 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
  - S1: 45.25 Mbps
  - S2 : 20.75 Mbps
  - S3 : 17 Mbps
  - S4 : 17 Mbps
  - S5 : 37.75 Mbps
  - S6 : 20.75 Mbps
  - S7 : 20.75 Mbps
  - S8 : 20.75 Mbps

How to determine a max-min fair bandwidth allocation for a given network ?

Algorithm [Bertsekas & Gallager, Data Networks, 2nd edition, Prentice Hall 1992]

First start with an allocation of 0 Mbps for each source
Then equally increment the allocation to each source until one link becomes saturated. At this point, each source which uses the saturated link receives an allocation equal to the bandwidth of this saturated link divided by the number of sources using this bottleneck link.
Next, the allocation of all the sources which do not use a saturated link is equally incremented until another link becomes saturated.
The algorithm continues from step to step, always incrementing the allocation of the sources which do not use a saturated link, until all sources use at least one of the saturated links.

How to provide max-min fairness ?

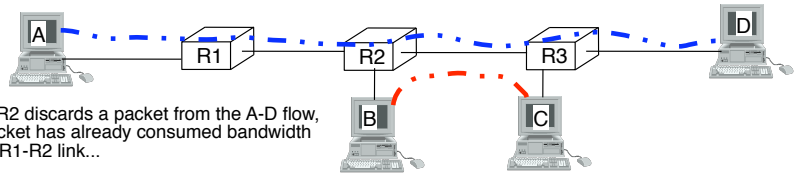
- Max-min fairness in TCP/IP network
  - an ideal goal, difficult to attain in practice

- Fairness will depend on 2 factors

1. Packet treatment inside routers
   - Which packets are dropped when congestion occurs
2. Congestion control mechanism implemented inside endsystems
   - TCP congestion control for applications based on TCP
   - what about UDP-based applications ?

Another way to control congestion

- TCP considers packet losses as the only congestion indication
  - but a dropped packet has already consumed resources on upstream links ...



When R2 discards a packet from the A-D flow, this packet has already consumed bandwidth on the R1-R2 link...

- Can we provide max-min fairness without dropping packets ?

Alternatives to TCP congestion control

- Other methods to detect congestion
  - DECBit
    - proposed at the same conference as TCP's congestion control mechanism
    - routers tag packets when they are congested
    - destination returns congestion indications in acks
  - Frame relay
    - FECN : congestion indicated in forward direction on VC
    - BECN : congestion indicated in backward direction on VC
  - ATM
    - Available Bit Rate
      - Bit-based congestion indication (EFCI bit)
      - Explicit Rate congestion indication (RM cells)

How to provide Explicit Congestion Notification to TCP ?

- Problems to solve
  1. How to detect congestion before it occurs ?
     - Difficult with tail-drop since buffer becomes full
     - easy with RED, since the buffer is never full
  2. How to explicitly inform sources about the network's congestion ?
     - Inform the source with ICMP source quench
       - specific ICMP message defined for this purpose, but almost never used in practice
         - an entire ICMP message (40+ bytes) must be generated
         - the entire ICMP packet must be sent in a potentially congested network
     - Inform the destination by tagging the IP packet
       - the destination must then inform the source with another mechanism

CNPP/2008.6.

The first RFC on ECN

K.Ramakrishnan and S.Floyd. A proposal to add Explicit Congestion Notification (ECN) to IP. Internet RFC 2481, January 1999.

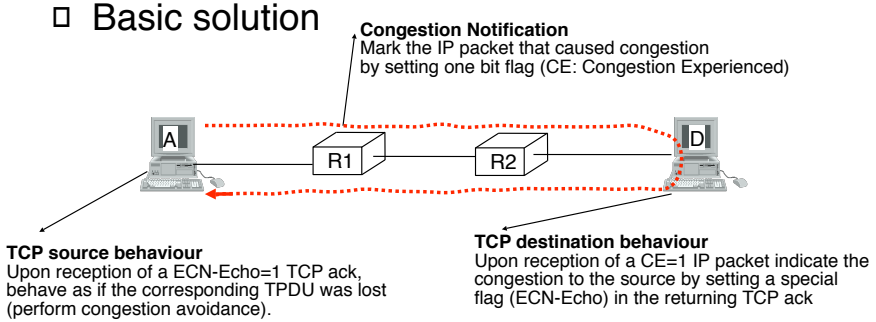
The current version, proposed standard

K.Ramakrishnan, S.Floyd, and D.Black. The addition of explicit congestion notification (ECN) to IP. Internet RFC 3168, September 2001.

A good list of references on ECN :
http://www.aciri.org/floyd/ecn.html

TCP Explicit Congestion Notification (3)

- Basic solution

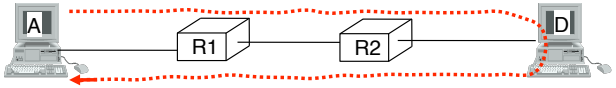


- Potential problems
  - What happens if the returning ECN-echo ack is lost ?
  - How can we deploy such a solution when 99.99% of the TCP sources/destinations are not ECN capable ?

Sources always send IP packets with the CE bit set to 0
A router can change the CE bit from 0 to 1 to indicate congestion, but cannot change it from 1 to 0
The CE bit is part of a previously rarely used space in the IP header
The ECN-Echo flag in an unused flag in the TCP header

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 source by setting a special flag (ECN-Echo) in all returning TCP acks until a TCP TPDU with CWR set is 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 utilisation 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

Ten years ago, upgrading hosts was considered to be a serious deployment problem. Today, with the regular distribution of patches and security fixes from all major OS vendors and the utilisation of automatic upgrades, a very large number of hosts can support a new feature or a new protocol within a few months.

TCP is still the dominant transport protocol in today's Internet. UDP has not been changed to support ECN given its datagram nature. However, some applications running on top of UDP have been enhanced to support ECN. Several researchers have proposed solutions to provide TCP-like or TCP compatible congestion control for UDP based applications.
SCTP is another transport protocol that was developed during the last years. It is rarely used by classical applications, but some operating systems already support it.

Deployment of ECN (2)

- 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

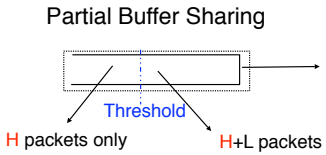
ECN-capable source
If destination is also ECN capable
Set ECT bit in all IP packets towards destination
Otherwise
Reset ECT bit

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

As a side note, it is interesting to note what happened when ECN was added to the Linux kernel. The first tests with this kernel revealed some subtle problems when using ECN on the Internet. Apparently, some firewalls were not ECN aware and dropped ECN packets, considering them as invalid, irrespectively of other firewall rules. Affected vendors have since fixed their firewall software, but this indicates that systems subtle problems may appear during deployment.

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




```
Arrival of packet
if (Packet.Type == H)
{ /* packet is high priority }
  if (Buf.Length < Buf.Size)
       accept_packet();
  else
       discard_packet();
}
else
{/* packet is low priority */
  if (Buf.Length < Buf.Threshold)
       accept_packet();
  else
       discard_packet();
}
```

CNPP/2008.6.

Partial Buffer Sharing can easily be extended to support N different drop priorities.

Packet discard preferences (2)

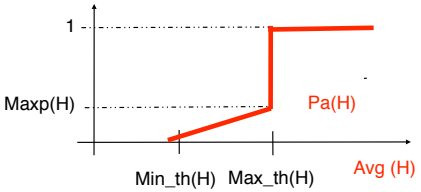
- Weighted RED
  - extension of RED to support N packet discard preferences
  - Principle of the solution
    - N RED algorithms run in parallel
      - the first one decides the acceptance of priority N packets that should only be discarded in case of severe congestion
      - the second one decides the acceptance of priority N-1 packets that should be discarded earlier than high priority packets
      - ...
      - The Nth RED algorithm decides the acceptance of packets without any priority

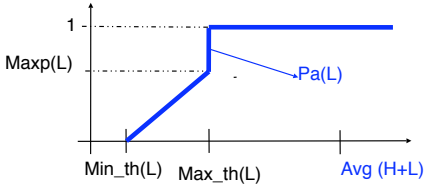
WRED was initially proposed as RIO (RED with In/Out) in
Clark and Fang, Explicit Allocation of Best Effort packet delivery service, IEEE/ACM transactions on networking, August 1998, vol 6, N 4, pp. 362-373

Several variants of RIO have been proposed and implemented since then.

Weighted RED

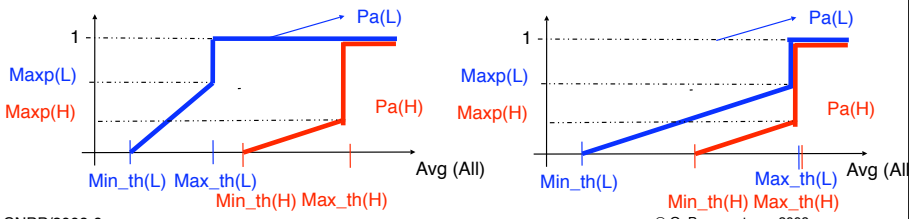
- RED with In/Out
  - Initial proposal to support two priorities
  - Principles
    - compute two averages : Avg(H) and Avg(H+L)
    - Apply conservative RED for H packets with large thresholds
    - Apply aggressive RED for L packets with small thresholds





Weighted RED (2)

- Weighted RED
  - can be used to support N drop priorities
  - Principles
    - compute a single average for all packets in buffer
    - conservative RED algorithm for high priority packets
      - large values for min_th(H) and max_th(H)
    - aggressive RED for low priority packets
      - small values for min_th(L) and min_th(L), and higher drop prob.



CNPP/2008.6.

The configuration guidelines for WRED on cisco routers propose to use the same value for max_th and maxp for all classes and to perform the differentiation only on the basis on min_th. See

http://www.cisco.com/univercd/cc/td/doc/product/software/ios120/12cgcr/qos_c/qcpart3/qcwred.htm

Traffic control in IP Networks
Outline

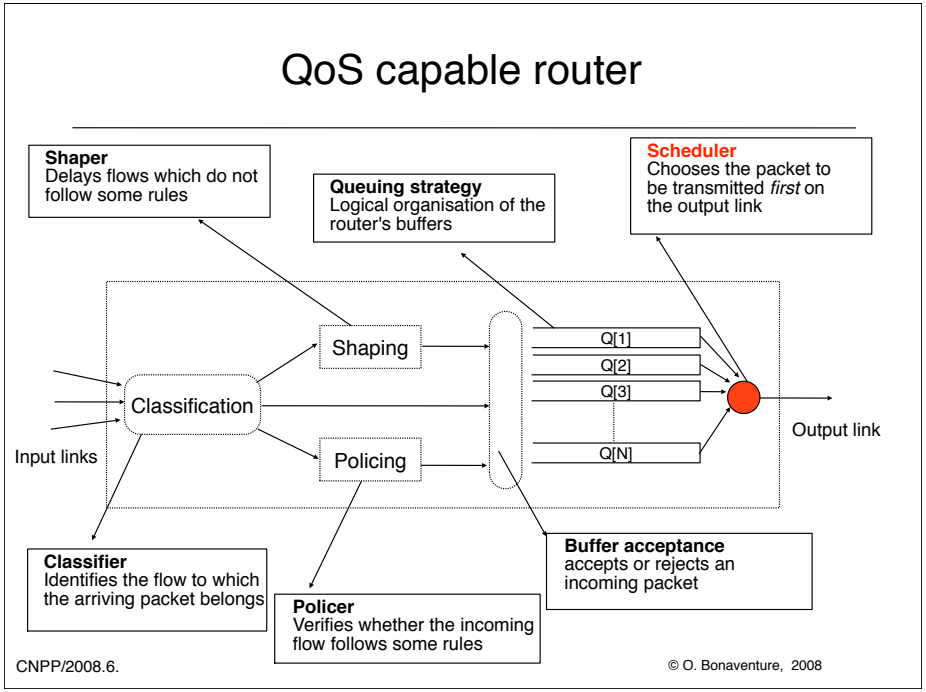
- Applications
- Packet-level traffic control mechanisms
  - Flow identification
  - Packet Marking
  - Buffer acceptance
  - Scheduling

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

For more information on schedulers, see

H.Zhang. Service disciplines for guaranteed performance service in packet-switching networks. Proceedings of the IEEE, 83(10), October 1995.



In practice, the shaper could also be located on the output link, but we don't address this issue here to keep the picture simple and understandable.

Queuing strategies

- Different queuing strategies are possible
  - One queue per layer-4 flow
    - provides good isolation among flows
    - costly to implement on high-speed links given large number of L4 flows
  - One queue per layer-3 or layer-2 flow
    - still provides some isolation
    - less costly to implement
  - One queue per class of service
    - usually a small number of classes (less than ten) are supported by the hardware
    - simplifies hardware
    - but forces network designer to groups packets in classes

Scheduling best-effort flows

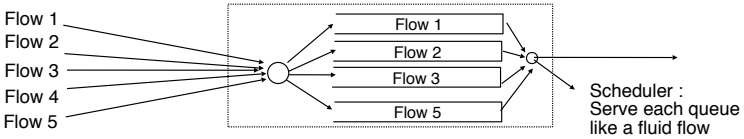
- Design goals
  - Provide a fair distribution of bandwidth between active flows to support max-min fairness at network level
    - fairness should not depend on behaviour of congestion control mechanisms inside endsystems
  - Provide protection between flows
    - a potentially misbehaving flow should 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

An extensive discussion of schedulers may be found in

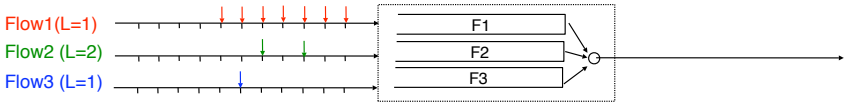
H.Zhang. Service disciplines for guaranteed performance service in packet-switching networks. Proceedings of the IEEE, 83(10), October 1995.

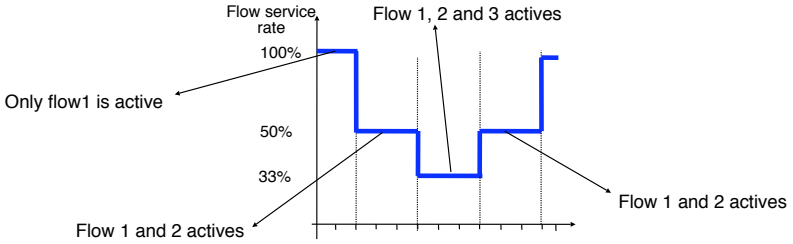
Processor Sharing

- 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
      - $$rate[j](t) = \left(\frac{link_{rate}}{N_{activeflows}(t)}\right)$$
      - a flow is considered active if its queue contains something



Processor Sharing : example





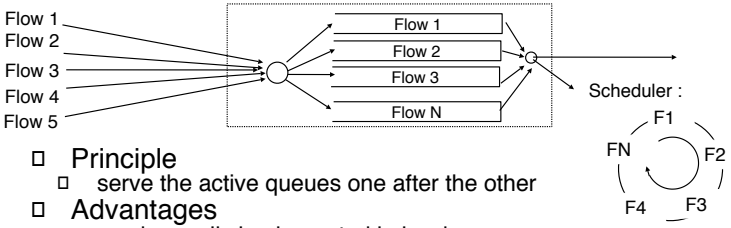
Finish times for the packets from the three flows
F1 : 1,2,4,6.5,9,11,12
F2 : 6.5, 11
F3 : 8

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
      - fairness is achieved for TCP and UDP flows !
    - if packets are discarded, then packet discarding must be performed in a fair manner
- Disadvantage
  - Ideal "mathematical" solution, not implementable in practice
    - approximations to PS are implementable

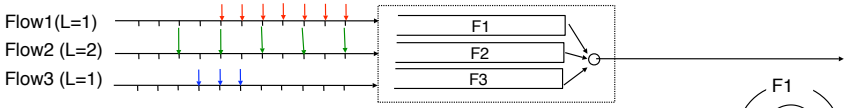
Round Robin

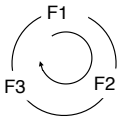
- Round Robin



  - 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

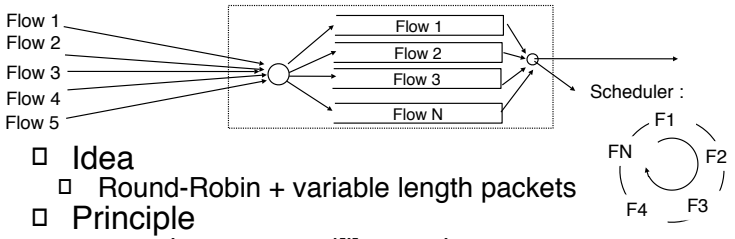




Flow1 and Flow3 send packets of size 1
Flow 2 sends packets of size 2

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

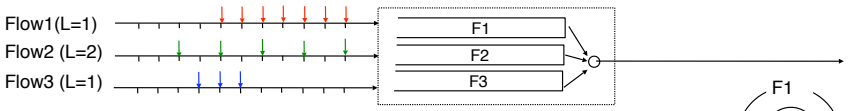


- Idea
  - Round-Robin + variable length packets
- Principle
  - associate counter d[i] to each queue
  - increase d[i] by *quantum* every time queue[i] is visited
    - if first_packet of queue[i] larger than d[i]
      { packet stays in queue[i]; }
    - else
      {
      packet is transmitted on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

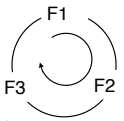
Deficit Round Robin is described in

M.Shreedhar and G.Vargese. Efficient fair queueing using deficit round robin. In Proc. ACM SICOGMM'95, pages 231--242, 1995.

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	-	2:P20(cont)
4	F1	0+1-1	0	0	P14	P22,P24	-	F1:P12
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:P13
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	2: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

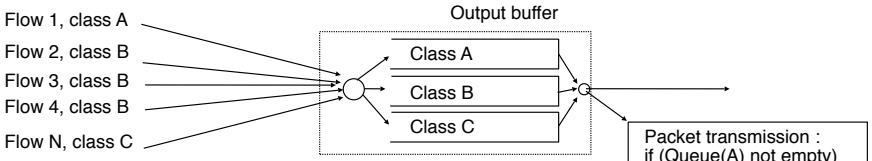
- How can we fairly discard packets when a scheduler is used ?
  - Packet should be discarded from flow causing congestion
    - How can we identify this congesting flow ?
  - If packets from a flow enter a queue faster than they leave, queue will build up
    - Flow with longest queue is responsible 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 jeopardise the guarantees committed to other flows
  - Implementable at high speeds

Priority-based scheduler

- A simple priority scheduler



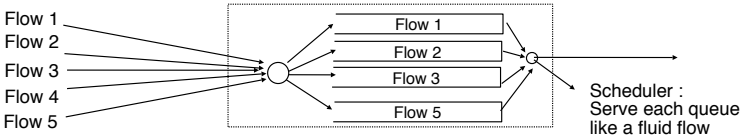

```
Packet transmission :
if (Queue(A) not empty)
 serve Queue(A);
else
  if (Queue(B) not empty)
    serve Queue(B);
  else
    if (Queue(C) not empty)
      serve Queue(C);
    else ....
```

- 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...

Generalised Processor Sharing

- Generalised 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
    - $$rate = link_{rate} \times \left(\frac{W[j]}{\sum_{i=active\ queues} W[i]}\right)$$
    - a queue is active if it contains something



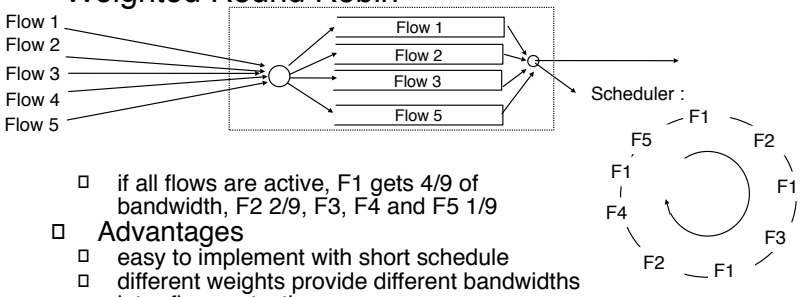
Generalised Processor Sharing (2)

- 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
    - through a network of GPS schedulers $delay\ bound = \frac{B}{R}$
  - provides bound on buffer utilisation $buffer\ bound = B$
  - provides protection among the different flows
    - a flow cannot jeopardise 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

A.Parekh and R.Gallagher. A generalized processor sharing approach to flow control : the single node case. IEEE/ACM Transactions on Networking, 1(3):346--357, 1993.
A.Parekh and R.Gallagher. A generalized processor sharing approach to flow control - the multiple node case. IEEE/ACM Transactions on Networking, 2(2):137--150, 1996.

Virtual Clock

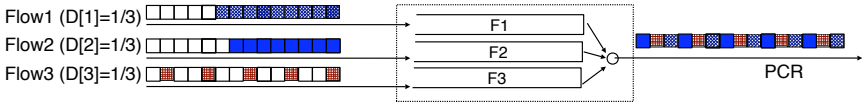
- Approximation of GPS
  - Idea
    - associate one timestamp to each arriving packet
    - scheduler selects among all the queued packets the packet with the smallest timestamp
  - First algorithm
    - D[i] : bandwidth associated with Queue[i]
    - V[i] : state variable associated with Queue[i]
    - 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 was proposed in

L.Zhang. VirtualClock: A new traffic control algorithm for packet switching. ACM Transactions on Computing Systems, 9(2):101--124, May 1991.

Virtual Clock (2)

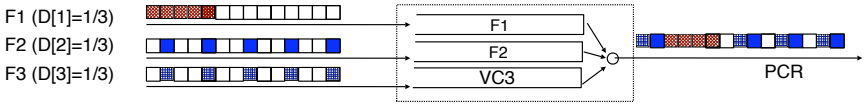
- First example



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 Clock (3)

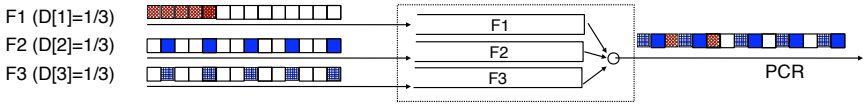
- Second example



T	V(F1)	Q(F1)	V(F2)	Q(F2)	V(F3)	Q(F3)	scheduled
0	0	-	0+3	3	0+3	3	F2
1	0	-	3	-	3	3	F3
2	0	-	3	-	3	-	-
3	0	-	3+3	6	3+3	6	F2
4	0	-	6	-	6	6	F3
5	0	-	6	-	6	-	-
6	0	-	6+3	9	6+3	9	F2
7	0	-	9	-	9	9	F3
8	0	-	9	-	9	-	-
9	0+3	3	9+3	12	9+3	12	F1
10	3+3	6	12	12	12	12	F1
11	6+3	9	12	12	12	12	F1
12	9+3	12	12+3	12,15	12+3	12,15	F1
13	12+3	15	15	12,15	15	12,15	F2

Virtual Clock (4)

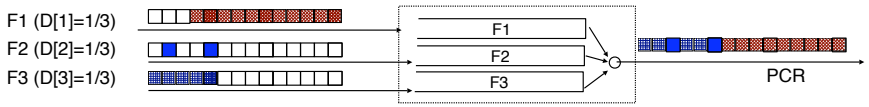
- Second algorithm
  - arrival of a P bytes long packet at time t
    - V[i] = max (V[i] , t)+ (P / D[i])
- Example



T	V(F1)	Q(F1)	V(F2)	Q(F2)	V(F3)	Q(F3)	Scheduled
0	0	-	max(0,0)+3	3	max(0,0)+3	3	F2
1	0	-	3	-	3	3	F3
2	0	-	3	-	3	-	-
3	...						
9	max(0,9)+3	12	max(9,9)+3	12	max(9,9)+3	12	F1
10	max(12,10)+3	15	12	12	12	12	F2
11	max(15,11)+3	15,18	12	-	12	12	F3
12	max(18,12)+3	15,...	max(12,12)+3	15	max(12,12)+3	15	F1
13	max(21,13)+3	18,...	15	15	15	15	F2
...							

Virtual Clock (5)

- Third example



T	V(F1)	Q(F1)	V(F2)	Q(F2)	V(F3)	Q(F3)	cheduled
0	max(0,0)+3	3	0	-	0	-	F1
1	max(3,1)+3	6	0	-	0	-	F1
2	max(6,2)+3	9	0	-	0	-	F1
...							
8	max(24,8)+3	27	0	-.	0	-	F1
9	max(27,9)+3	30	max(0,9)+3	12	max(0,9)+3	12	F2
10	max(30,10)+3	30,33	12	-	max(12,10)+3	12,15	F3
11	33	30,33	12	-	max(15,11)+3	15,18	F3
12	33	30,33	max(12,12)+3	15	max(18,12)+3	18,21	F2
13	33	30,33	15	-	max(21,13)+3	18,21,24	F3
14	33	30,33	15	-	max(24,14)+3	21,24,27	F3
....							

SCFQ / 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

For more information on SCFQ, see

J.Roberts. Virtual spacing for flexible traffic control. International Journal of Communication Systems, 7:307--318, 1994.

J.Roberts, U.Mocci, and J.Virtamo, editors. Weighted Fair Queueing, chapter6, pages 173--187. Number 1155 in Lecture Notes in Computer Science. Springer Verlag, 1996.

S.Golestani. A self-clocked fair queuing scheme for broadband applications. In IEEE INFOCOM94, pages 636--646, 1994.

Guarantees

- Schedulers supporting per-flow bandwidth guarantees and protection between flows
  - GPS
  - WFQ/PGPS
  - SCFQ
  - Deficit-WRR
  - These guarantees are independent of the behaviour of the guaranteed flows and of the behaviour of other flows
    - one flow cannot jeopardise the bandwidth guarantees provided to other flows
    - this implies a good buffer acceptance mechanisms

Guarantees (2)

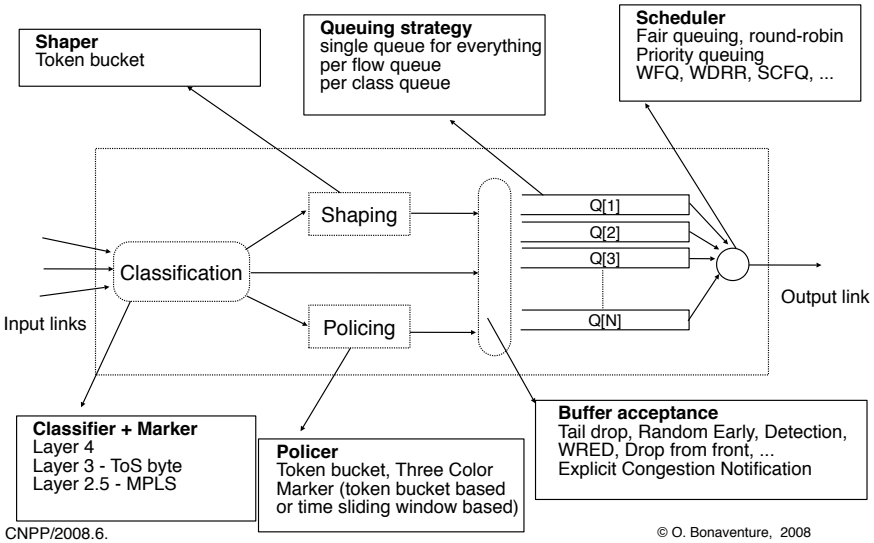
- Schedulers supporting delay guarantees
  - delay guarantees are only available for token bucket (R,B) limited flows
    - but guarantee does not depend on behaviour of other flows
  - Delay through a series of n schedulers (ignoring the fixed delays)
    - GPS $\frac{B}{R}$
    - WFQ / PGPS $\frac{B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

CNPP/2008.6.

Source : H. Zhang, Service disciplines for guaranteed performance service in packet switching networks, Proc. IEEE, Vol 83, No 10, October 1995

A good textbook with a good description of schedulers is
S. Keshav, An engineering approach to computer networking : ATM networks, the Internet and the Telephone network, Addison Wesley, 1997

Traffic Control in IP networks
Packet level traffic control mechanisms



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 ? (2)

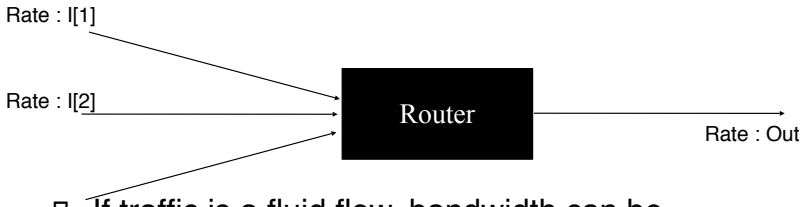
- Delay and delay jitter guarantees
  - Best effort flow
    - no delay or delay jitter guarantee
  - Maximum guaranteed bandwidth flow
    - maximum delay guarantee
      - e.g. For interactive voice
    - delay jitter guarantee
      - e.g. When playback jitter cannot be used by receiver
  - Minimum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

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
    1. Associate IP packets to flows
    2. Provide guarantees for specific flows

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

$$\sum_{j=1}^{j=N} I[j] \leq Out$$

Providing bandwidth guarantees (2)

- Problem
  - In packet based networks, traffic is a flow of variable length packets and not a fluid flow
- How to provide bandwidth guarantees ?
  1. Ensure that the output link will not be a bottleneck

$$Out \geq \sum_{j=1}^{j=N} I[j]$$

     - We must limit the rate of flows on the input links
  2. 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 ?
  1. 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...
  2. On packet arrival
     - If current rate is within contract, accept packet
     - Otherwise discard packet

Traffic control and QoS in IP Networks Outline

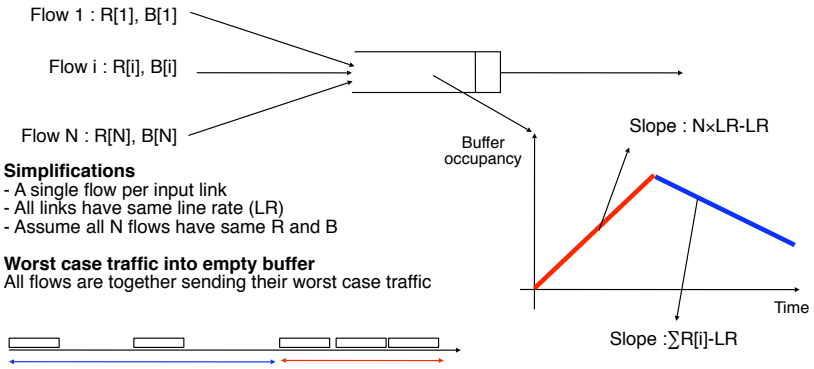
- Applications
- Packet-level traffic control mechanisms
  - Best-effort service
  - Maximum bandwidth service
  - Minimum bandwidth service
  - Delay guarantees
- Standardized Services

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

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



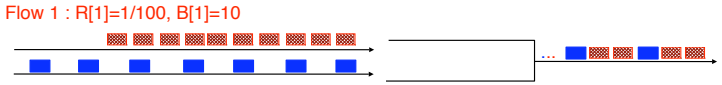
Simplifications
- A single flow per input link
- All links have same line rate (LR)
- Assume all N flows have same R and B

Worst case traffic into empty buffer
All flows are together sending their worst case traffic

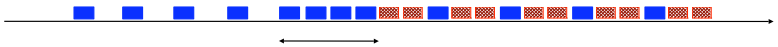
Packets sent at average rate (R) Initial burst, bucket emptied

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 ?



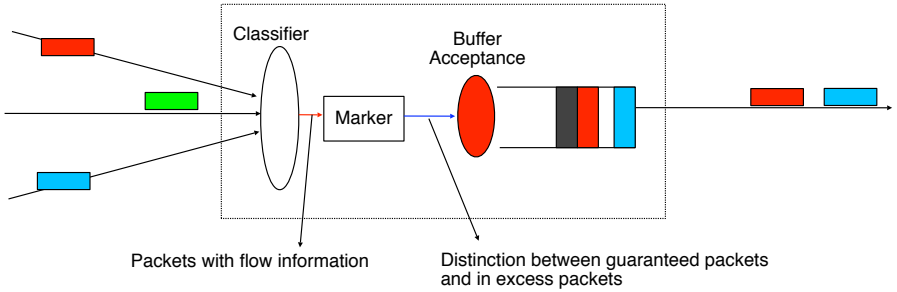
Too long burst for Flow 2 !!!

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

It should be noted that having a bound on the buffer occupancy implies that there will also be a bound on the amount of delay encountered by one flow through a router. However, the bound on delay jitter will only be [0, D_{max}]

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

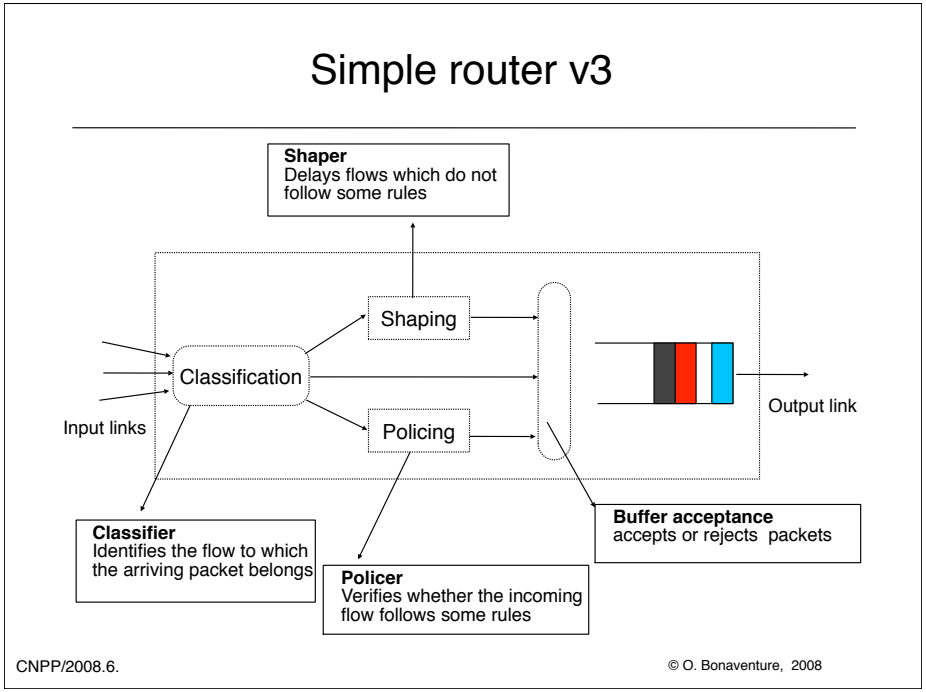


Traffic control and QoS in IP Networks Outline

- Applications
- Packet-level traffic control mechanisms
  - Best-effort service
  - Maximum bandwidth service
  - Minimum bandwidth service
  - Delay guarantees
- Standardized Services

Towards multiservice networks

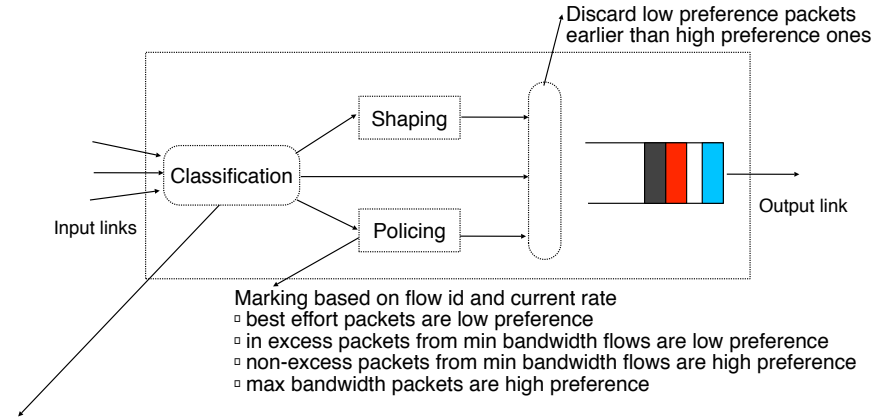
- Problems
  - How can we multiplex on a single link through one router classical best-effort traffic packets and packets from guaranteed flows ?
    - guaranteed packets should not be perturbed by best-effort packets
    - best-effort packets should be able to utilize the output link when there is no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



The shaping mechanism can be placed either at the output of the classifier or directly upstream of the output link depending on whether incoming or outgoing traffic is shaped.

Multiplexing with Simple router v3

- Best-effort, min and max bandwidth flows

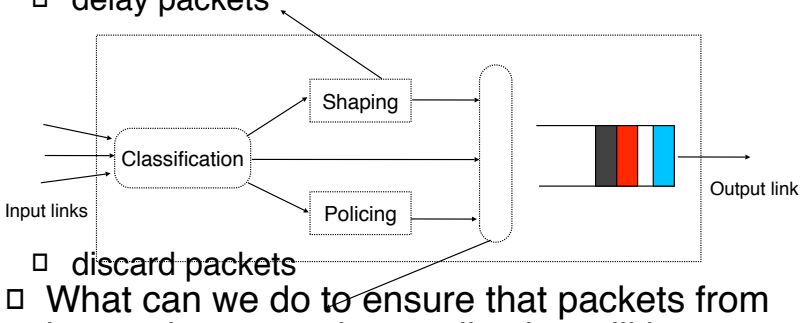


Marking based on flow id and current rate
- best effort packets are low preference
- in excess packets from min bandwidth flows are low preference
- non-excess packets from min bandwidth flows are high preference
- max bandwidth packets are high preference

Identifies the three types of flows
adds internal flow id to packet

What about delay guarantees ?

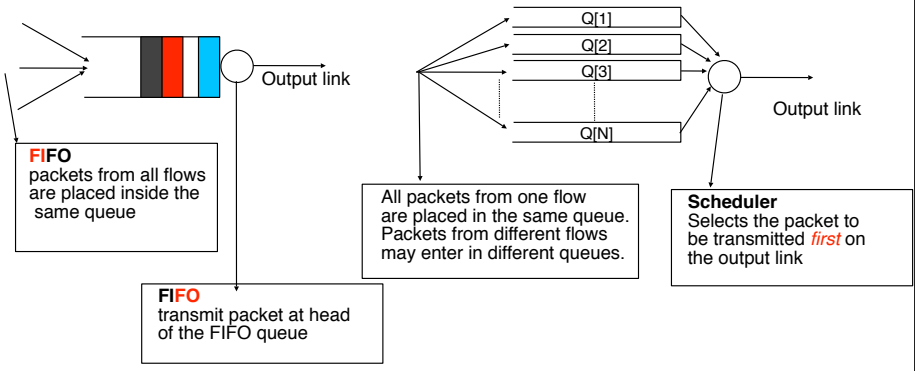
- Mechanisms supported by simple router v3
  - delay packets

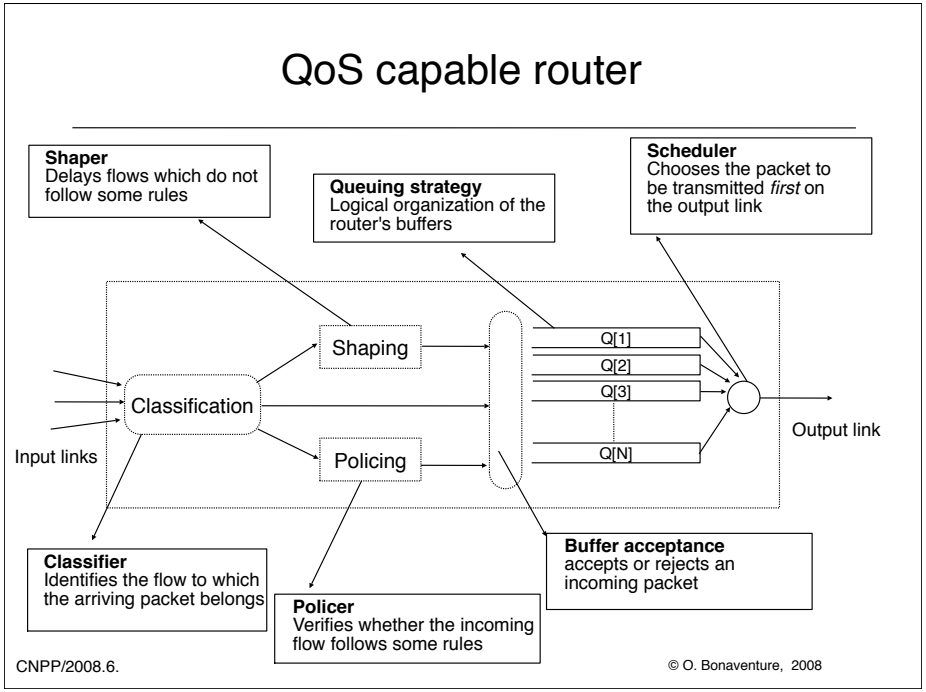


  - discard packets
- What can we do to ensure that packets from interactive streaming application will be sent earlier than packets from batch application ?

What about delay guarantees (2)

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





In practice, the shaper could also be located on the output link, but we don't address this issue here to keep the picture simple and understandable.