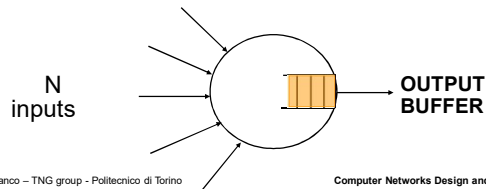


Scheduling and QoS scheduling

Andrea Bianco
Telecommunication Network Group
firstname.lastname@polito.it
<http://www.telematica.polito.it/>

Scheduling algorithms

- Scheduling: choose a packet to transmit over a link among all packets stored in a given buffer (multiplexing point)
- Mainly look at QoS scheduling algorithms
 - Choose the packet according to QoS needs



Output buffered architecture

- Advantage of OQ (Output Queued) architectures
 - All data immediately transferred to output buffers according to data destination
 - It is possible to run QoS scheduling algorithms independently for each output link
- In other architectures, like IQ or CIOQ switches, problems become more complex
 - Scheduling to satisfy QoS requirements and scheduling to maximize the transfer data from inputs to outputs have conflicting requirements

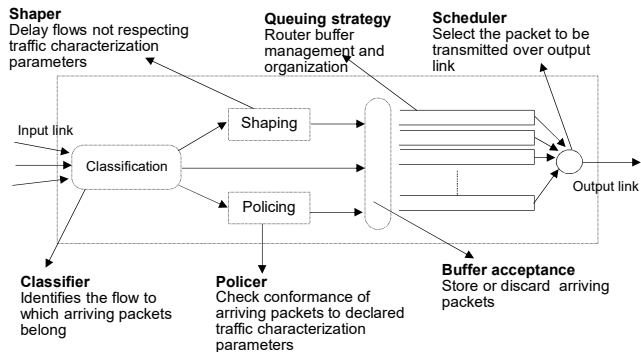
QoS scheduling algorithms

- Operate over multiplexing points
- Micro or nano second scale
- Easy enough to be implemented in hardware at high speed
- Regulate interactions among flows
 - Single traffic relation (1VP/1VC)
 - Group of traffic relations (more VC/1VP o more VC with similar QoS needs)
 - QoS classes
- Strictly related and dependent from buffer management techniques
- To simplify and make the problem independent, assume infinite capacity buffers
- Choice of the scheduler may have implications on CAC

Andrea Bianco – TNG group - Politecnico di Torino

Computer Networks Design and Management - 4

QoS-capable router



Andrea Bianco – TNG group - Politecnico di Torino

Computer Networks Design and Management - 5

QoS scheduling algorithms: properties

- Flow isolation
 - “mis-behaving” (non conformant) flows should not damage “well-behaved” (conformant) flows
 - PER-FLOW queuing, which implies resource partitioning
 - scheduler chooses from which queue to transmit the packet
 - Related to fairness
- End-to-end statistical or deterministic guarantees
 - Bit rate
 - Equal for all flows (useful for best effort traffic)
 - Specific for each flow
 - Delay
 - Losses

Andrea Bianco – TNG group - Politecnico di Torino

Computer Networks Design and Management - 6

QoS scheduling algorithms classification

- Work-conserving scheduler
 - Always transmit a packet as long as there is at least a packet available in switch buffer
 - Optimal performance in terms of throughput
- Non-work-conserving scheduler
 - May delay packet transmission
 - No transmission even if there are packets stored in buffers
 - Reduced throughput
 - Better guarantees on delay jitter
 - Reduced buffer size
 - In theory appealing approach, not much used in practice

Scheduling discipline property

- Theorem
 - The sum of mean queuing delays received by a set of multiplexed connections, weighted by their share of the link load is independent of the scheduling algorithm
- A scheduling algorithm can reduce a connection mean delay only at the expense of increasing the delay of another connection
- A work-conserving scheduler can only reallocate delays among connections
- A non work-conserving scheduler can only provide a mean queuing delay larger than a work conserving discipline

Work conserving versus non-work conserving schedulers

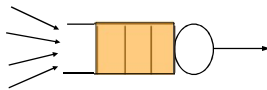
- Work-conserving schedulers disadvantage
 - Multiplexing point increase traffic burstiness
 - This increase packet jitter and buffering requirements to prevent losses
 - Pathological scenarios demonstrate that this phenomena may become worse when the number of crossed nodes increases
- Non work-conserving schedulers have buffering requirements independent of the network depth

Scheduling algorithms goals

- Best-effort traffic scheduler
 - All active flows should obtain the same amount of service
 - Possibly max-min fair
 - No delay guarantees
 - FIFO, PS (Processor Sharing), RR (Round Robin), DRR (Deficit Round Robin)
- QoS scheduler, i.e. scheduler for traffic with QoS requirements
 - Specific bit rate guarantees for each flow
 - Specific delay guarantees for each flow
 - Strict priority, GPS (Generalized Processor Sharing), WRR (Weighted Round Robin), WFQ (Weighted Fair Queuing), EDD (Earliest Due Date)

FIFO

- FIFO (First In First Out) service discipline
 - Also known as FCFS (First Come First Served)
- Single queue
- Data queued according to arrival time and served in order



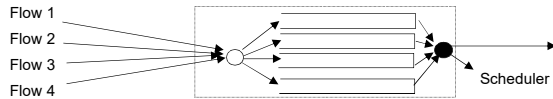
FIFO: properties

- Work-conserving
- Complete sharing of link bit rate and buffer space:
 - no protection against non conformant flows
- All flows observe similar delay performance
 - Suited to best-effort traffic
- Neither bit rate (bandwidth) guarantees nor loss guarantees
 - Performance depend on the amount of ingress data traffic of each flow
- Aggressive flows obtain better performance
 - Unfair

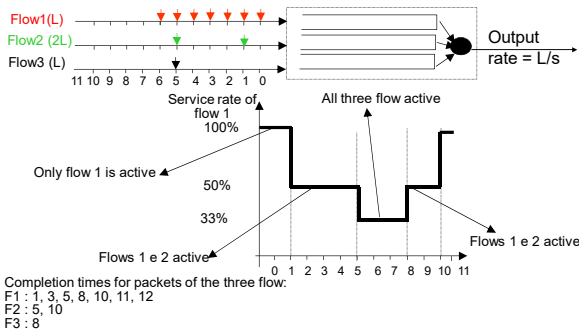
Processor Sharing

- Ideal work-conserving scheduler for best effort
- Each queue served according to a fluid model
- At time t , queue j is served at rate

$$rate[j] = \frac{rate_{link}}{\# activeflows}$$



Processor Sharing: example

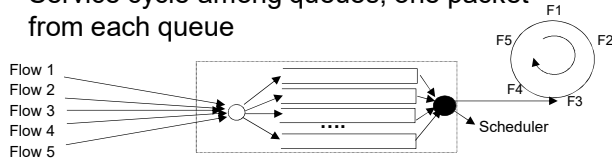


Processor Sharing

- Pros
 - If no data are discarded, a network of PS schedulers provides rates close to a max-min fair allocation
 - Rate of the max-min allocation only downstream from the bottleneck link
 - Fairness does not require congestion control mechanisms
 - If dropping packets, fair dropping must be ensured
- Cons
 - Ideal solution, non practical (packets are not fluids)
 - Devise approximations

Round Robin

- Processor sharing approximation
- Buffer organized in separate queues, one queue for each active flow
 - Each queue is a FIFO queue
- Service cycle among queues, one packet from each queue



Andrea Bianco – TNG group - Politecnico di Torino

Computer Networks Design and Management - 16

Round Robin

- May have some delay bias
- To improve delay fairness, at each serving cycle it is possible to modify queue service order
 - At time 0, queue service order: 1,2,3,...,K
 - At time 1, queue service order: 2,3,...,K,1

Andrea Bianco – TNG group - Politecnico di Torino

Computer Networks Design and Management - 17

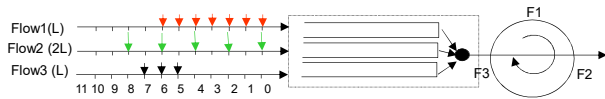
Round Robin: properties

- Relatively easy to implement in hardware
- Guarantees flow isolation
 - Through queue separation
- Service rate of each queue:
 - C/K , for fixed packet size and k flows
 - For variable packet size, some rate unfairness may arise (fair in #packets per flow)
 - Taking into account packet size makes implementation more complex

Andrea Bianco – TNG group - Politecnico di Torino

Computer Networks Design and Management - 18

Round Robin: example

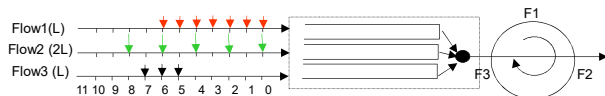


T	In F1	In F2	In F3	Q(F1)	Q(F2)	Q(F3)	Scheduled packets
0	P10[1]	P20[2]	-	P10	P20	-	F1:P10
1	P11[1]	-	-	P11	P20	-	F2:P20
2	P12[1]	P22[2]	-	P11,P12	P22	-	F2:P20 (cont)
3	P13[1]	-	-	P11,P12,P13	P22	-	F1:P11
4	P14[1]	P24[2]	-	P12,P13,P14	P22,P24	-	F2:P22
5	P15[1]	-	P35[1]	P12,P13,P14,P15	P24	P35	F2:P22 (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

- Round robin scheduler working with variable packet size
- Each queue[i] has a deficit counter d[i] associated
- d[i] is increased by a fixed quantum when queue [i] is visited
 - if (length_first_packet of queue[i] > d[i])
 - { packet is kept in queue[i] }
 - else
 - { packet transmitted on output link;
 - d[i]=d[i]- packet_length;
 - if (queue [i] is empty) { d[i]=0; }

Deficit Round Robin: example



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	P12,P13	P22	-	F2:P20(cont)
4	F1	0+1-1	0	0	P13,P14	P22,P24	-	F1:P12
5	F2,F3	0	0+1	0+1-1	P13,P14,P15	P22,P24	P35	F3:P35
6	F1	0+1-1	1	0	P14,P15,P16	P22,P24,P26	P36	F1:P13
7	F2	0	1+1-2	0	P14,P15,P16	P22,P24,P26	P36,P37	F2:P22
8	-	0	0	0	P14,P15,P16	P24,P26	P36,P37	F2:P22(cont)
9	F3	0	0	0+1-1	P14,P15,P16	P24,P26	P36,P37	F3:P36
10	F1	0+1-1	0	0	P15,P16	P24,P26	P37	F1:P14
11	F2,F3	0	0+1	0+1-1	P15,P16	P24,P26	P37	F3:P37
...								

Deficit Round Robin

- The idea is to keep track of queues that were not served in a round (compute deficit) and to compensate in the next round
- Keep an active list of indices of queues that contain at least a packet to avoid examining empty queues
- May be a problem to define the quantum
 - If too small, may need to visit too many times queues before serving a queue
 - If too large, some short term unfairness may arise
- Fair only over a time scale longer than a round time
 - Round time is a function of the number of flows and packet size
 - At a shorter time scale, some flows may get more service
 - Small packet size or high transmission speed reduce the round time

Strict priority

- First attempt to define a QoS capable scheduler
- Buffer partitioned in k queues, k being the number of priority classes
- Each queue is associated with a different priority
- Data unit are stored in a queue according to their priority level
- Higher priority queue is always served. Only if empty, the lower priority is considered
 - Non preemptive service: packet under service finish transmission
- Within each queue, data are served according to a FIFO service discipline

Strict priority algorithm

- Work-conserving
- Easy to implement
- Perfect isolation for high priority queue only, low priority queues may even suffer starvation (if CAC is not adopted on high priority queues)
 - Fair?
- No bit rate, loss and delay guarantees
- No isolation among flows stored in the same FIFO queue, i.e., within the same priority level

Generalized Processor Sharing

- Fluid system used as an ideal reference
- One queue for each flow
- Each queue is served as if it contains a fluid flow, i.e. by an infinitesimal fraction of time
- Each queue j is associated with a weight $w[j]$, normally derived from bit rate requirements
- At time t , queue j is served at rate:

$$rate[j] = rate_{link} \frac{w[j]}{\sum_{i=active\ queues} w[i]}$$

- A queue is active if it contains some fluid
- If the number of active flows decreases, excess bit rate is redistributed in proportion to queue weight
- CAC algorithms must control the rate of served flows, otherwise bit rate guarantees cannot be obtained

GPS properties

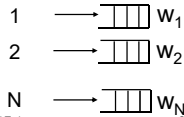
- Work conserving with flow isolation
- Per flow bit rate guarantees
 - When using a single GPS scheduler
 - When using a network of GPS schedulers
- End-to-end delay guarantees for token bucket (r,b) constrained flows
- Provides bounds on buffer size
- Simple jitter delay guarantees $([0, D_{max}])$
- Ideal scheduler, practical approximations needed

GPS approximation

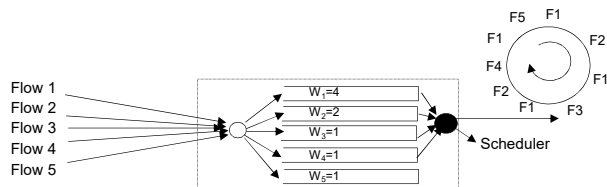
- Frame-based
 - Define a service cycle (frame)
 - Allocate frame portion to each flow
 - Example: WRR (Weighted-Round Robin), WDRR (Weighted Deficit Round Robin)
- Sorted priority
 - Compute a timestamp (tag) and associate it with each packet
 - Packets are ordered for increasing timestamp
 - Examples: Virtual Clock, WFQ (Weighted Fair Queuing), SCFQ

WRR: Weighted Round Robin

- GPS approximation
- Buffer partitioned in N queues
 - each queue served according to a FIFO discipline
- A weight $w_i \propto$ requested bit rate is associated with each queue
- A service cycle among queues is executed, each queue being served proportionally to its weight, i.e., w_i per cycle
- Cycle length is the summation of the weights (possibly normalized)



WRR: Weighted Round Robin



- If all flows are active
 - F1 obtains 4/9 of the link bit rate
 - F2 obtains 2/9
 - F3, F4 and F5 obtain 1/9

WRR: properties

- Work-conserving
- Flow isolation guaranteed
- For each queue i:
 - bit-rate = $w_i / (\sum_j w_j) \text{link_rate}$
 - if all packets are of the same size
- Easy to implement (for a small number of flows)
- Define a service cycle

WRR: problems

- Service cycle (and fairness) may become long when
 - Many flows are active
 - Flows have very different weights
 - On a 45Mbit/s link, 500 flows with weight 1 and 500 flows with weight 10
 - Service time of one cell (48 bytes) 9.422us
 - A cycle requires $500+500*10=5500$ service time=51.82ms
- Service provided to flows may be bursty
 - Avoidable, but complex
- For each variation of the number of active flows (departure, arrival) service cycle must be redefined
 - How to deal with the remaining part of the cycle?
- To deal with variable packet size may use WDRR, Deficit Round-Robin extended to weight support
- Note. WRR may be exploited in best effort scenario
 - May use weights in WRR to compensate for variable packet size for best effort traffic (requires knowledge of flow average packet size)

Sorted priority approximation to GPS

- Per-flow queuing
- Data (cells) served on the basis of negotiated rate and cell arrival time
 - Each data has a tag (urgency) assigned
- Data are inserted in a Sorted Priority Queue on the basis of data tag
- Data are served according to tag ordering
- Several algorithms: virtual clock, WFQ o PGPS, SCFQ

Virtual Clock

- Time Division Multiplexing emulation
- Each flow j has an assigned service rate r_j
- To each data k of length L_j^k belonging to flow j , a tag (label, urgency, auxiliary virtual clock) is assigned
 - Tag represents the data finishing service time (starting service time + service time) in a TDM system serving flow j at rate r_j :

$$\text{Aux VC}_j^k = \text{Aux VC}_j^{k-1} + \frac{L_j^k}{r_j}$$

Scheduling in OQ architectures

Virtual Clock scheduling

Example

	0	1	2	3	4	5	6	7	8	9
$r_1=1/3$	□ 3	□ 6	□ 9	□ 12	□ 15	□ 18				
$r_2=1/3$	○ 3	○ 6	○ 9	○ 12	○ 15	○ 18				
$r_3=1/3$	△ 3			△ 6			△ 9			△ 12

Service order: □ 3 ○ 3 △ 3 □ 6 ○ 6 △ 6 □ 9 ○ 9 △ 9

Andrea Bianco – TNG group - Politecnico di Torino Computer Networks Design and Management - 34

Virtual Clock: example 1

T	AV(F1) Q(F1)	AV(F2) Q(F2)	AV(F3) Q(F3)	Scheduled flow
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	- -	F2
3	9+3 6,9,12	9+3 6,9,12	3+3 3	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			

Andrea Bianco – TNG group - Politecnico di Torino Computer Networks Design and Management - 35

Virtual Clock scheduling

Problem:

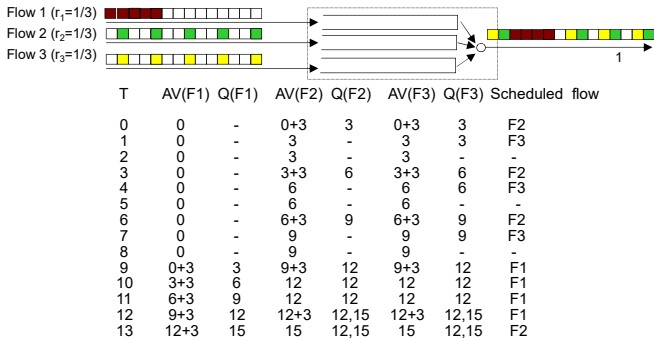
	0	1	2	3	4	5	6	7	8	9	10	11	12	13
$r_1=1/3$	□ 3			□ 6			□ 9			□ 12			□ 15	
$r_2=1/3$	○ 3		○ 6			○ 9			○ 12			○ 15		
$r_3=1/3$									△ 3	△ 6	△ 9	△ 12	△ 15	

Service order: □ 3 ○ 3 □ 6 ○ 6 □ 9 ○ 9 △ 3 △ 6 △ 9 □ 12 ○ 12

Andrea Bianco – TNG group - Politecnico di Torino Computer Networks Design and Management - 36

ag. 12

Virtual Clock: problem



Virtual Clock

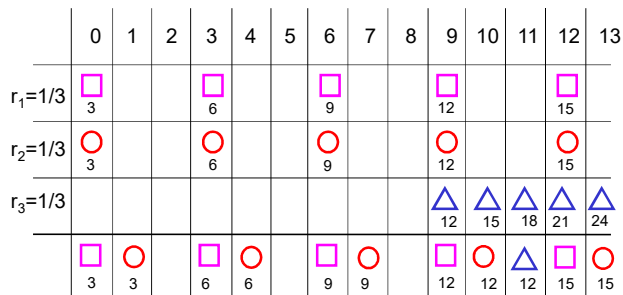
- Long term fairness with some problems
 - Inactive flows “gain time” and get more service in the future, penalizing, and even starving, other active flows (even conformant flows)
 - Clock of different flows proceed independently
- Modify the tag computation, taking into account system real time:

$$\text{Aux VC}_j^k = \max(\text{Aux VC}_j^{k-1}, a_j^k) + \frac{L_j^k}{r_j}$$

– where a_j^k is the arrival time of cell k of flow j

Virtual Clock scheduling

Problem solved



Modified Virtual Clock

Another problem

Time	0	1	2	3	4	5	6	7	8	9	10	11	12	13
$r_1=1/3$	□ 3	□ 6	□ 9	□ 12	□ 15	□ 18	□ 21	□ 24	□ 27	□ 30	□ 33	□ 36	□ 39	
$r_2=1/3$	○ 3	○ 6	○ 9	○ 12	○ 15	○ 18	○ 21	○ 24	○ 27	○ 30	○ 33	○ 36	○ 39	
$r_3=1/3$													△ 15	△ 18
	□ 3	○ 3	□ 6	○ 6	□ 9	○ 9	□ 12	○ 12	□ 15	○ 15	□ 18	○ 18	△ 15	△ 18

Andrea Bianco - TNG group - Politecnico di Torino Computer Networks Design and Management - 40

Virtual Clock

- Even the modified version of Virtual clock can lead to unfairness
- Clocks of flows are now synchronized by the system time
- However, tags may overcome the system time when flows get excess bandwidth
- Excess bandwidth must be redistributed among flows to ensure work conserving property but reallocation must not penalize flows in the future

Andrea Bianco - TNG group - Politecnico di Torino Computer Networks Design and Management - 41

WFQ (Weighted Fair Queueing) or PGPS (Packetized GPS)

- Algorithms that try to approximate GPS behavior
 - The minimum amount of service that can be provided cannot be smaller than the service time of a cell, since no preemption is admitted
- At time τ , the transmitted packet is the packet whose service would finish first in the GPS system if no other packets arrive after τ
 - Need to emulate the GPS system

Andrea Bianco - TNG group - Politecnico di Torino Computer Networks Design and Management - 42

ag. 14

WFQ or PGPS

Example:

- 1 flow with negotiated rate 0.5
 - 10 fixed size packets arrive at rate 1 starting at time 1
- 10 flows with negotiated rate 0.05
 - 1 packet arrives at time 1

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	Time	
P1	P2	P3	P4	P5	P6	P7	P8	P9	P10												Ideal fluid system
										P11										GPS	
										P20											
P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	P16	P17	P18	P19	P20	WFQ service order	

Andrea Bianco – TNG group - Politecnico di Torino Computer Networks Design and Management - 43

WFQ o PGPS

- Tag computation
 - Tag should represent the finishing service time of data in the GPS system
 - However, it is fundamental to compute the tag when data unit are received at buffer input
 - Future should be known, since the data finishing service time in the ideal system depends on flow activation in the future
 - The problem is trivial if all flows are always active, since service rate are fixed

Andrea Bianco – TNG group - Politecnico di Torino Computer Networks Design and Management - 44

WFQ or PGPS

- Tag computation:

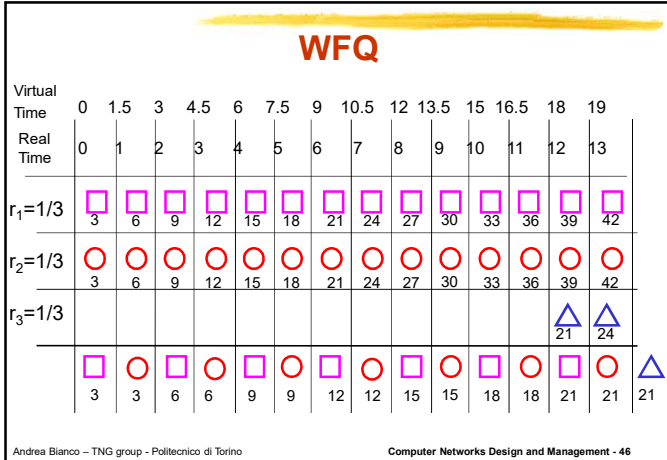
$$F_j^k = \max \{ F_j^{k-1}, V(a_j^k) \} + \frac{L_j^k}{\phi_j}$$
- $V(t)$ is the system virtual time or system potential (k active flows):

$$V(0) = 0$$

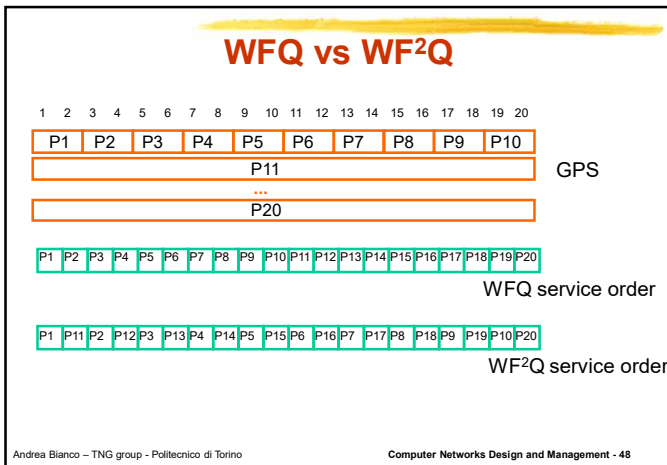
$$\frac{\partial V}{\partial \tau} = \frac{1}{\sum_k \phi_k}$$
- If flows are always active, the virtual time corresponds exactly to the real time

Andrea Bianco – TNG group - Politecnico di Torino Computer Networks Design and Management - 45

Scheduling in OQ architectures



- ### WFQ o PGPS
- Very complex to implement
 - Same properties of GPS
 - WFQ can emulate the ideal GPS system with a time difference bounded by the maximum size packet!
 - Several variations were proposed
 - Indeed, in WFQ packets are never delayed too much, but could be transmitted too early
 - WF²Q
 - improves the similarity of service order to GPS
 - among available packets, the packet with the smallest tag is chosen but only among packets whose service has already started in the ideal GPS system
- Andrea Bianco – TNG group - Politecnico di Torino Computer Networks Design and Management - 47



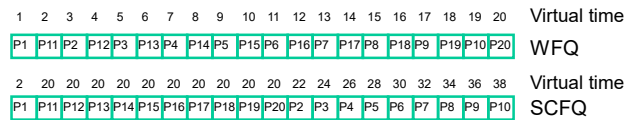
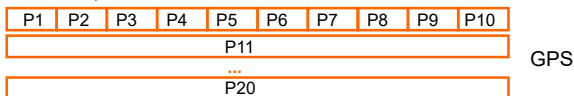
SCFQ

(Self Clocked Fair Queueing)

- Variation of PGPS, simpler to implement
- Does not require emulation of GPS system
- Uses a simplified virtual time
 - Virtual time is set to the tag of the packet being serviced

SCFQ vs WFQ

- 1 flow with negotiated rate 0.5
 - 10 fixed size packets arrive at rate 0.5 starting at time 0
- 10 flows with negotiated rate 0.05
 - 1 packet arrives at time 0



Delay bounds

- Can be computed for token bucket limited flows (R,B)
- Guarantees independent of other flow behavior
- Max delay through n scheduler (excluding fixed delays):

- GPS $\frac{B}{R}$
 - WFQ / PGPS $\frac{B + n \cdot P_{\max}}{R} + \sum_{i=1}^n \frac{P_{\max}}{C_i}$
 - Virtual Clock $\frac{B + n \cdot P_{\max}}{R} + \sum_{i=1}^n \frac{P_{\max}}{C_i}$
 - SCFQ $\frac{B + n \cdot P_{\max}}{R} + \sum_{i=1}^n \frac{k_i \cdot P_{\max}}{C_i}$
- C_i output rate of i -th switch
 • k_i number of flows
 • P_{\max} maximum packet size

- Bandwidth delay coupling

EDD (Earliest Due Date)

- In classical EDD
 - Each packet is assigned a deadline
 - Packets served in deadline order
 - Deadline satisfied only if the scheduler is not overcommitted
- Traffic divided in classes
 - Each class i is characterized by a service deadline d_i
- Scheduler selects, at time t , the packet with the smallest residual time
 - Each packet is time stamped with time t_k on arrival
 - Residual time of a packet = $t_k + d_i - t$
 - the amount of time left before packet service deadline expires
- EDD tends to equalize the probability of violating the delay constraint

EDD (Earliest Due Date)

- Need to specify the process to assign deadlines
 - Delay EDD and Jitter EDD
- Delay EDD
 - packets belonging to sources obeying a peak rate constraint are assigned a worst case delay (in each node, deadline=expected arrival time+delay bound)
 - CAC must run a schedulability test to check if deadlines can be satisfied
 - Delay bound independent of bandwidth constraint (but need to reserve the peak)
- Jitter EDD
 - Delay jitter regulator in front of a EDD scheduler (non work conserving, see later)
- Issues
 - Interesting to manage delays, difficult to deal with bandwidth guarantees
 - Complex to implement (timers, dealing with real numbers)

Non work-conserving algorithms

- Packets can be scheduled only if eligible
- Eligibility through traffic regulators
 - Rate-jitter regulator
 - Bounds maximum rate
 - Delay jitter regulator
 - Compensates for variable delay at previous hop
- After the regulator use a scheduler (may be FIFO)
- Properties
 - Reduced throughput
 - Worse average delays but
 - Control on delay jitter
 - Reduced buffer size
- Examples
 - Stop and go
 - Hierarchical round robin

Regulators for non work-conserving algorithms

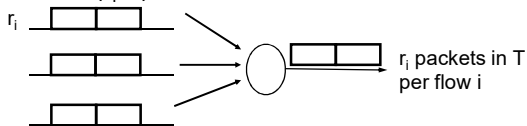
- Rate jitter regulators
 - E.g.: peak rate regulator
 - eligibility time of a packet is the eligibility time of the previous packet plus the inverse of the peak rate (time taken to serve the packet at the peak rate)
- Delay jitter regulators
 - The sum of the queuing delay in the previous switch and the regulator delay is constant
 - Eliminates the delay variability induced by the queuing delay at the previous hop
 - The output stream is a time shifted version of the traffic at input
 - Time shift equal to propagation delay plus delay bound (worst case) at previous switch
 - Burstiness cannot build up
 - Do not protect against misbehaving sources
 - Very complex to implement (it requires clock synchronization)
- Note: by properly selecting the regulator and the scheduler a wide range of work-conserving and non work-conserving schedulers may be emulated

An example of a non work-conserving scheduler: Stop & go

- Framing strategy
 - Time axis divided into frames of length T
- At each switch, the arriving frame of each incoming link is mapped to the departing frame of the output link by a constant delay smaller than T
- Transmission of packets arriving on any link during a frame are postponed to the beginning of the next frame

Stop & go

- Packets on the same frame at the source stay in the same frame throughout the network
- If the traffic is (r_i, T) smooth at source i , it will remain (r_i, T) smooth



Stop & go

- As long as each node can ensure local delay bound, end-to-end delay bound can be guaranteed
- Problem of coupling between delay bounds and bandwidth allocations granularity
 - Assume a fixed packet size P
 - Minimum bandwidth can be P/T
 - Delay bounded by two time frames T
 - Reducing T , reduced the delay but increases the minimum bandwidth
- Generalized stop&go with multiple frame sizes
 - Coupling still exist, but can have low delays for some flows and fine bandwidth granularity for other flows

Andrea Bianco – TNG group - Politecnico di Torino

Computer Networks Design and Management - 58

References

- H. Zhang, "Service Disciplines for Guaranteed Performance Service in Packet-Switching Networks", Proc. IEEE, vol. 83, Oct. 1995, pp.1374-96
- A. Varma, D. Stiliadis, "Hardware Implementations of Fair Queueing Algorithms for Asynchronous Transfer Mode Networks", IEEE Communications Magazine, Dec. 1997, pp. 54-68
- A. Parekh, R. Gallager, "A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: the Single Node Case", IEEE/ACM Trans. On Networking, vol. 1, no.3, June 1993
- S.Keshav, "An engineering approach to computer networking: ATM networks, the Internet and the telephone network", Addison Wesley, 1997

Andrea Bianco – TNG group - Politecnico di Torino

Computer Networks Design and Management - 59
