Quality of Service (QoS) fundamentals

QoS fundamentals

- Definitions
  - Throughput, delay/RTT, packet loss, jitter, availability, …
- Application layer requirements
- Network QoS mechanisms
  - Classification
  - Packet Scheduling
  - Queue management
- Traffic Contract
  - Traffic Contract Specification: Leaky bucket, token bucket
  - Traffic Conditioning: Shaping and Policing
- Signaling and CAC

Definitions

Introduction to QoS

- A network supports Quality of Service (QoS) if it provides reliable transport and guarantees some performance figures, e.g.:
  - Throughput (b/s and pps)
  - Average delay, RTT
  - Packet Loss Ratio
  - Jitter (delay variation)
  - Availability
  - Other contexts
    - Link: BER (Bit Error Rate) and PER (Packet Error Rate)
    - Circuit Switching: Blocking probability, Set-up delay, etc
    - Application layer: Interactivity = f(RTT)
- Different applications have different QoS requirements

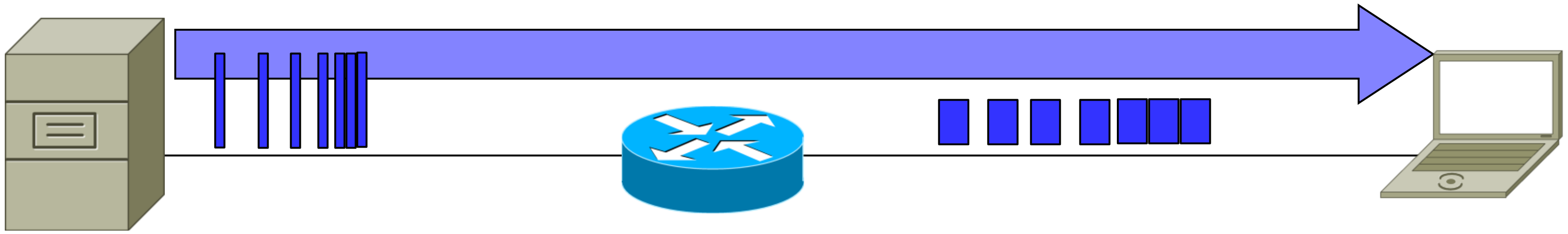
QoS in packet switching networks

- **WAN Technologies**
  - **FR: <Tc, CIR, EIR>**
  - **ATM**
    - **Traffic: PCR,SCR,MBS,MCR**
    - **QoS: CLR,CTD,CDV**
    - **Categories: CBR,VBR-RT,VBR-nRT,ABR,GFR,UBR**
    - **Traffic Control:**
      - CAC
      - UPC
      - Priority Control
      - Flow Control (EFCN, discard)
      - Congestion Control (preventive, reactive)
  - **IP**
  - **MPLS**

- **LAN Technologies**
  - **IEEE 802.1p**
    - ✓ Extension of 802.1q (tag VLAN added to the MAC layer). The VLAN tag has two parts: VLAN ID (12-bit) and Prioritisation (3-bit), this last one is defined in the 802.1p standard.
    - ✓ Priority Queuing
    - ✓ No reservations
- **MAN**
- **SAN**
- **Home/Multimedia**
- **Industrial**
- **Access**

Throughput

- *Throughput:* is the rate (bits/time unit) of *successful* message delivery over a communication channel or network.
  - *"instantaneous":* rate in a relatively short time interval (1ms-10ms-1s), yet longer than the packet's transmission time
  - *average:* rate over a longer period of time (transfer,1min-…)
  - It depends on the level/layer where we measure

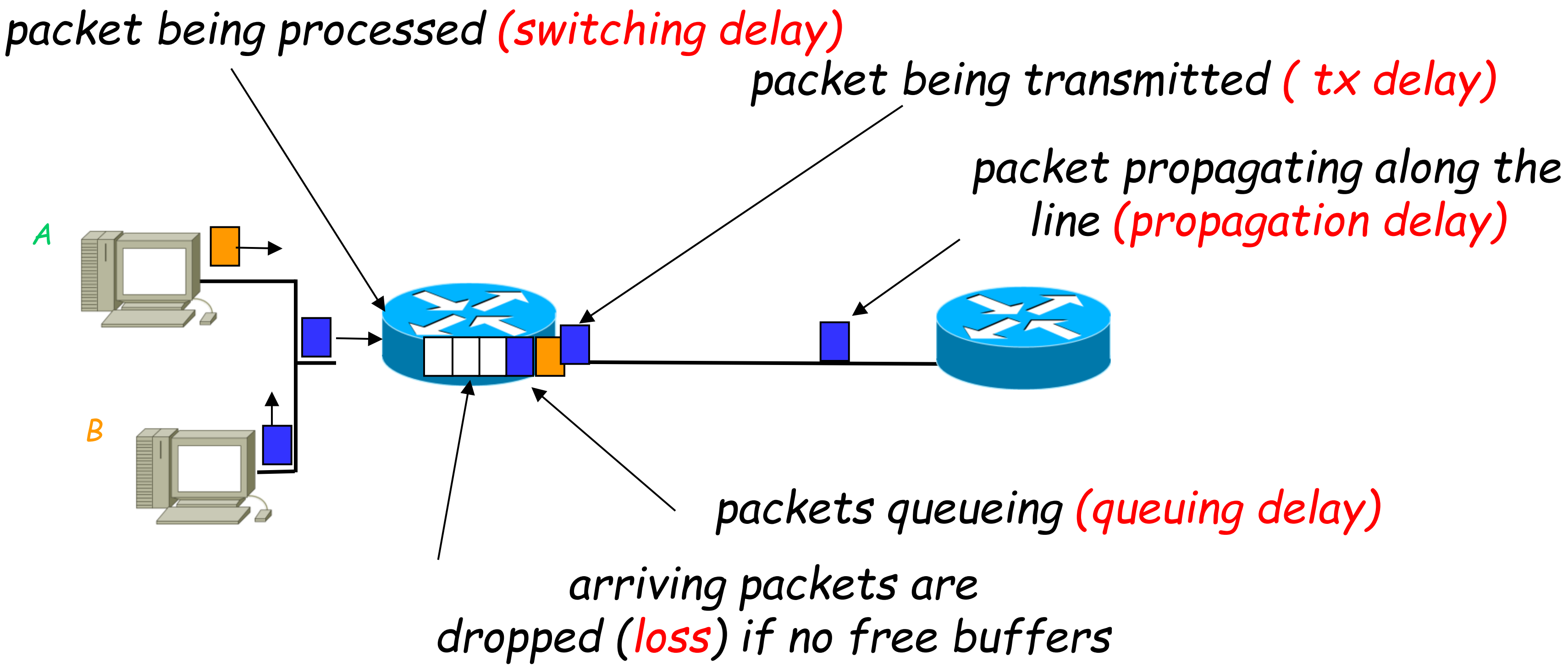


- goodput is the application level throughput, i.e. the number of useful information bits delivered by the network to a certain destination per unit of time.

Delay / RTT / Packet loss

Packets *queue* in router buffers

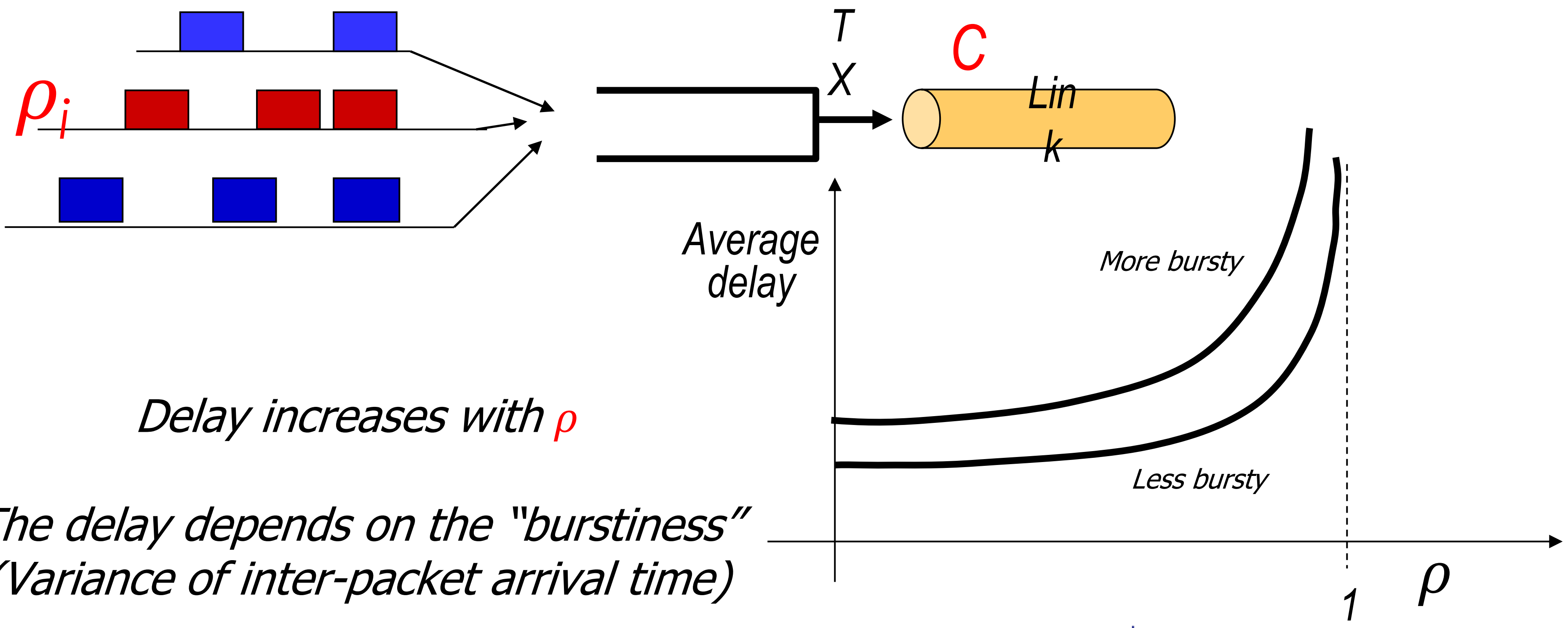
- **If packet arrival rate to link temporarily exceeds the output link capacity then packets have to queue, waiting for turn**


packet being processed (switching delay)
packet being transmitted (tx delay)
packet propagating along the line (propagation delay)
A
B
packets queueing (queuing delay)
arriving packets are dropped (loss) if no free buffers

Adapted from: Kurose & Ross

Queueing delay

- Per-flow rate: $\rho_i C = \lambda_i T_i C$
  - arrival rate x transmission time x Link Capacity
- Aggregate rate: $\rho C = \Sigma_i \rho_i C$



Delay increases with ρ

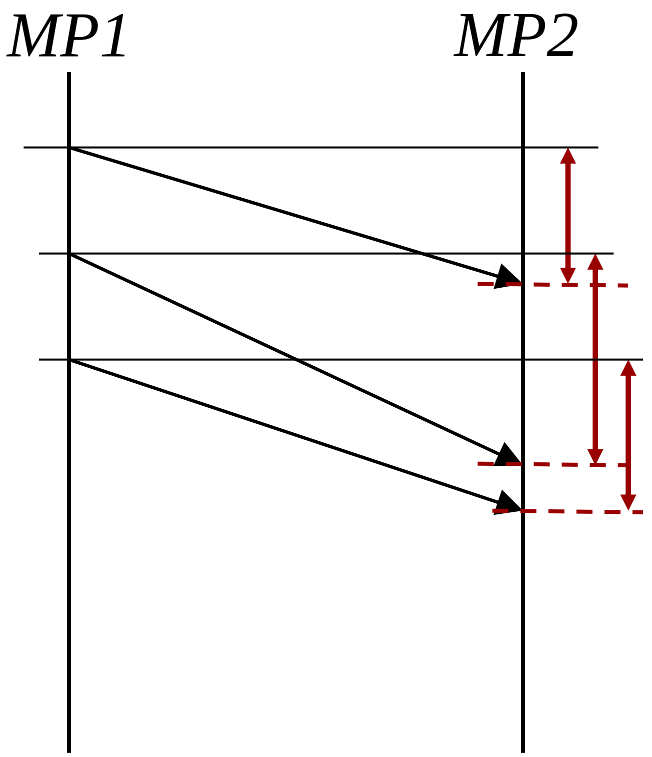
The delay depends on the "burstiness" (Variance of inter-packet arrival time)

Jitter

- **Variability of packet delays within the same packet stream**

RFC3393: A definition of the IP Packet Delay Variation (ipdv) can be given for packets inside a stream of packets. The ipdv of a pair of packets within a stream of packets is defined for a selected pair of packets in the stream going from measurement point MP1 to measurement point MP2. The ipdv is the difference between the one-way-delay of the selected packets.

ITU-T Y.1540 <u>Internet protocol data communication service - IP packet transfer and availability performance parameters</u>



IPDV

- *Instantaneous (2 packets): ipdv = delay(p1)-delay(p2)*
- *Variance = E((ipdv-E(ipdv))^2)*
- *Maximum*

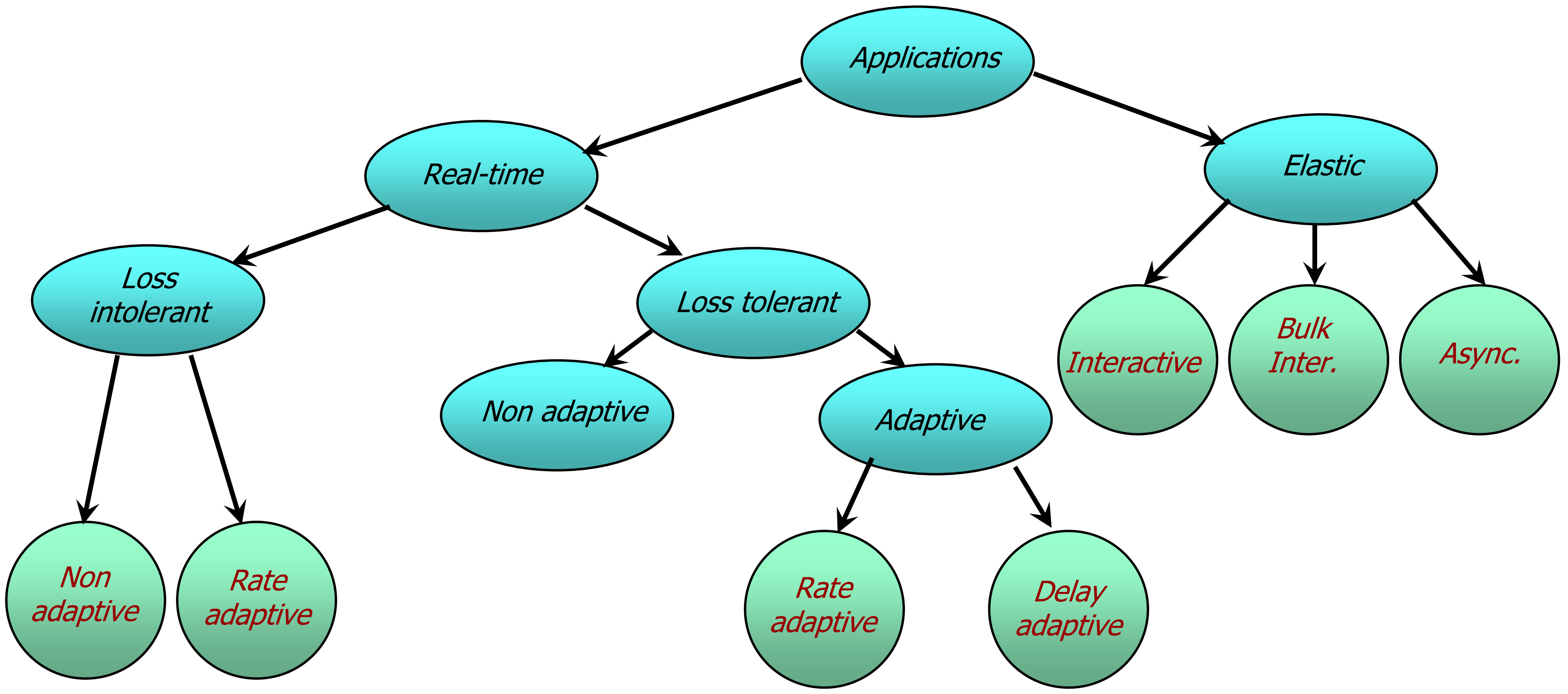
Other QoS parameters: Availability

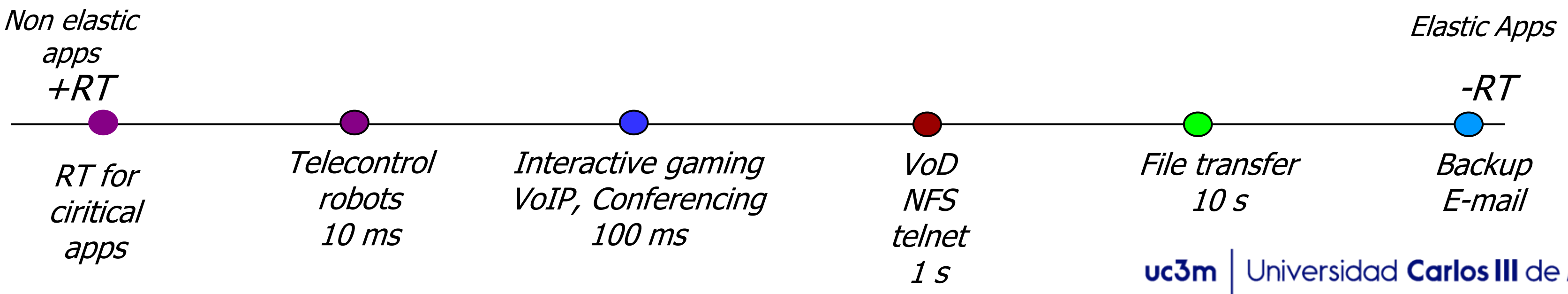
- Fraction of time that a circuit/system/link/network works according to its specification
  - ✓ A = MTBF/(MTBF+MTTR) = 1-U = 1-MTTR/(MTBF+MTTR)

Availability	
99,900 % = "Three Nines"	(9 hours average out of service in a year)
99,990 % = "Four Nines"	(53 min. average out of service in a year)
99,999 % = "Five Nines"	(5 min. average out of service in a year)

Application layer requirements

Taxonomy of networked applications

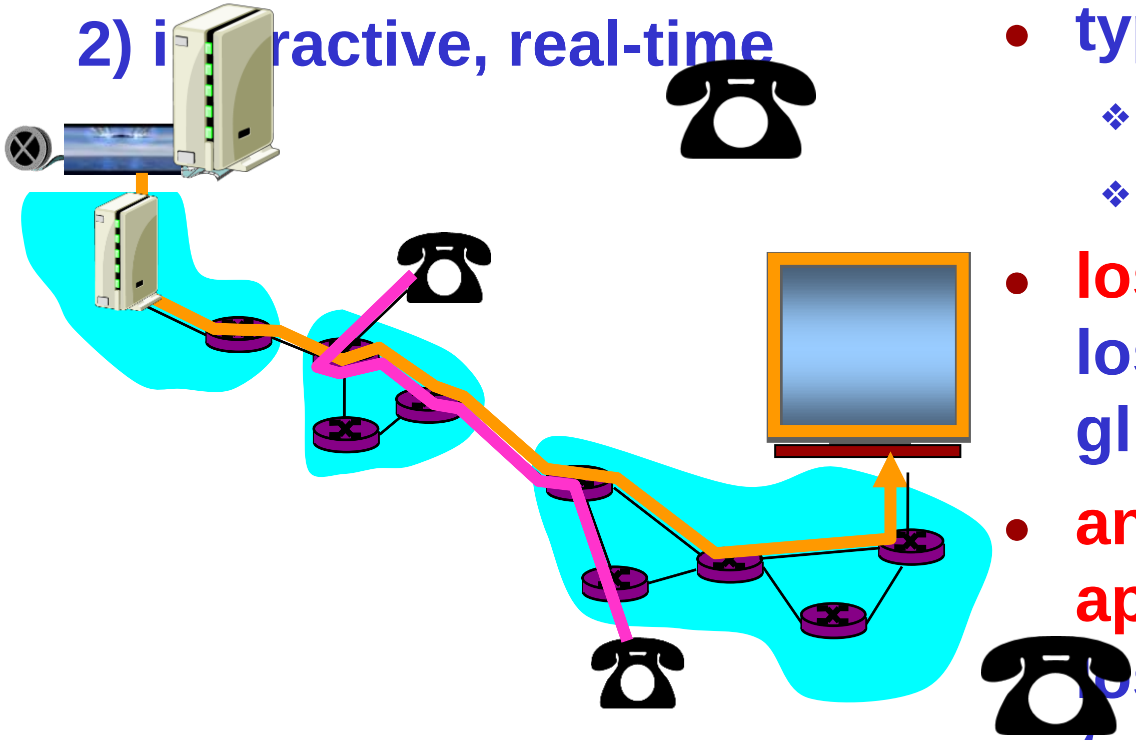




Internet QoS

- **Today : best-effort**
  - **QoS tools**
    - ✓ **LOSS**
      - End-to-end
        - Retransmission
        - FEC
        - Time-bounded retransmission
      - Network
        - Over-provisioning
    - ✓ **DELAY**
      - End-to-end
        - Buffering
      - Network
        - Over-provisioning

Multimedia applications



Classes of MM applications:

1) stored/live streaming

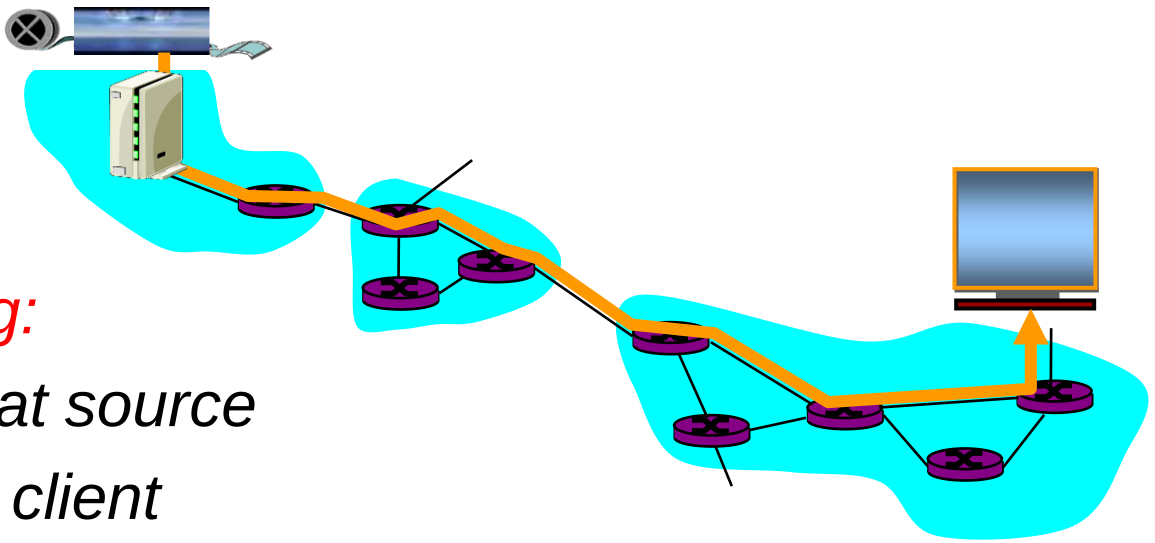
2) interactive, real-time

Fundamental characteristics:

- typically **delay sensitive**
  - end-to-end delay
  - delay jitter
- **loss tolerant:** infrequent losses cause minor glitches
- **antithesis of file transfer applications**, which are loss *intolerant* but delay *tolerant*.

Adapted from: Kurose & Ross

Streaming Stored Multimedia



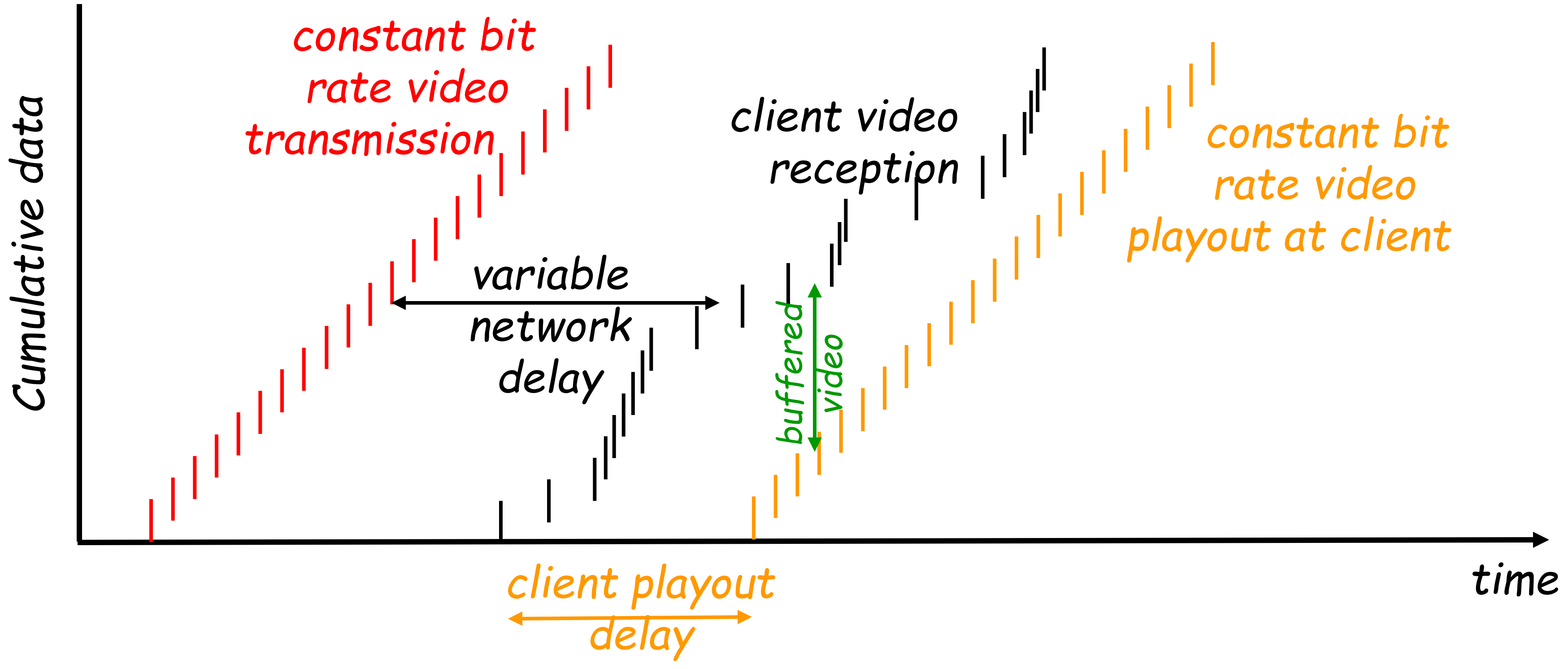
Stored streaming:

- *media stored at source*
- *transmitted to client*
- *streaming: client playout begins before all data has arrived*

- *timing constraint for still-to-be transmitted data: in time for playout*

Adapted from: Kurose & Ross

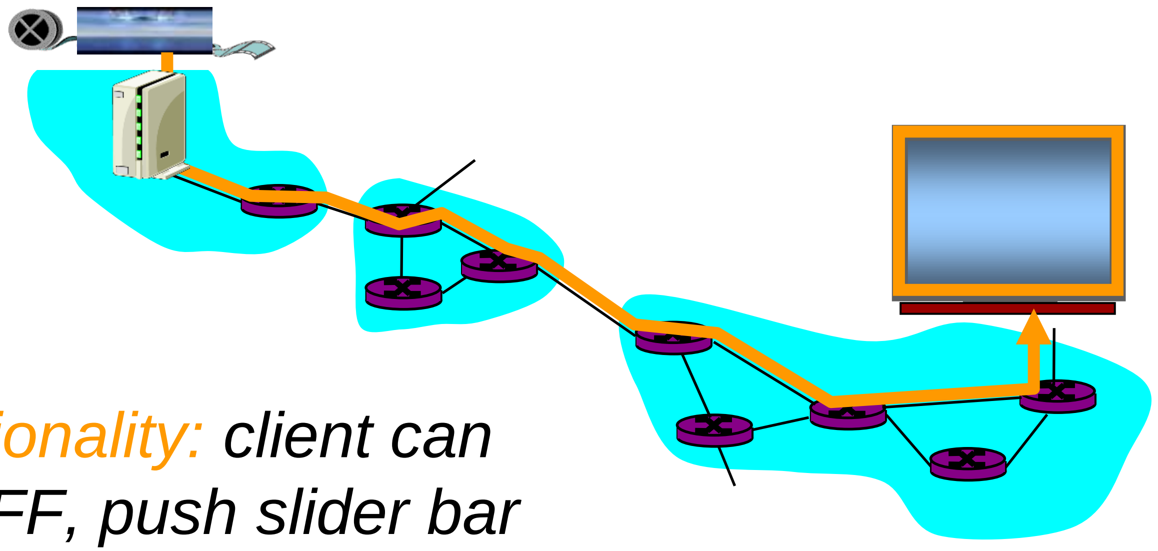
Streaming Multimedia: Client Buffering



- **client-side buffering, playout delay compensate for network-added delay or jitter**

Adapted from: Kurose & Ross

Streaming Stored Multimedia: Interactivity



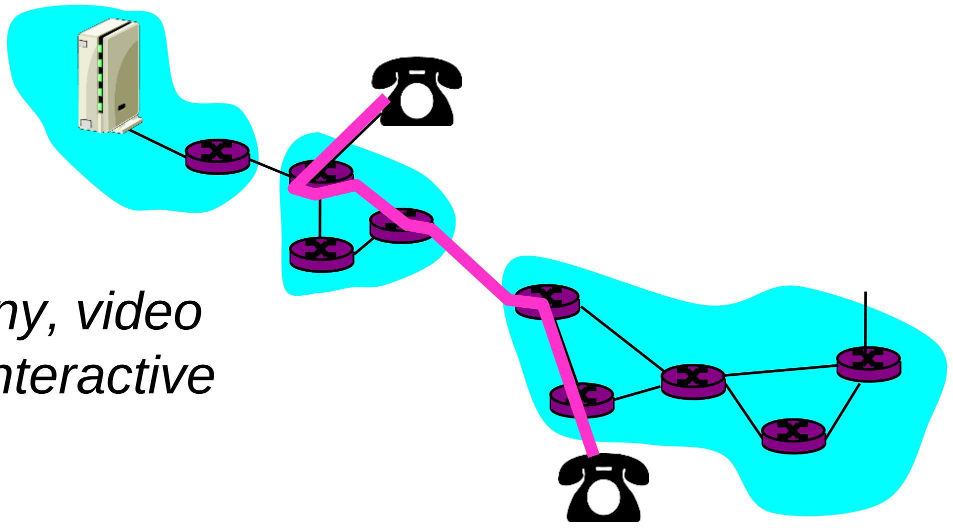
- *VCR-like functionality: client can pause, rewind, FF, push slider bar*
  - *5 sec initial delay OK*
  - *1-2 sec until command effect OK*

RTSP

Dynamic Adaptive Streaming over HTTP (DASH)

Adapted from: Kurose & Ross

Real-Time Interactive Multimedia



- *Applications: IP telephony, video conference, distributed interactive games*

- **Guranteed rate**
- **End-end delay requirements:**
  - **E.g.: audio: < 150 msec good, < 400 msec OK**
    - ✓ **includes application-level (packetization) and network delays**
    - ✓ **higher delays noticeable, impair interactivity**

Adapted from: Kurose & Ross

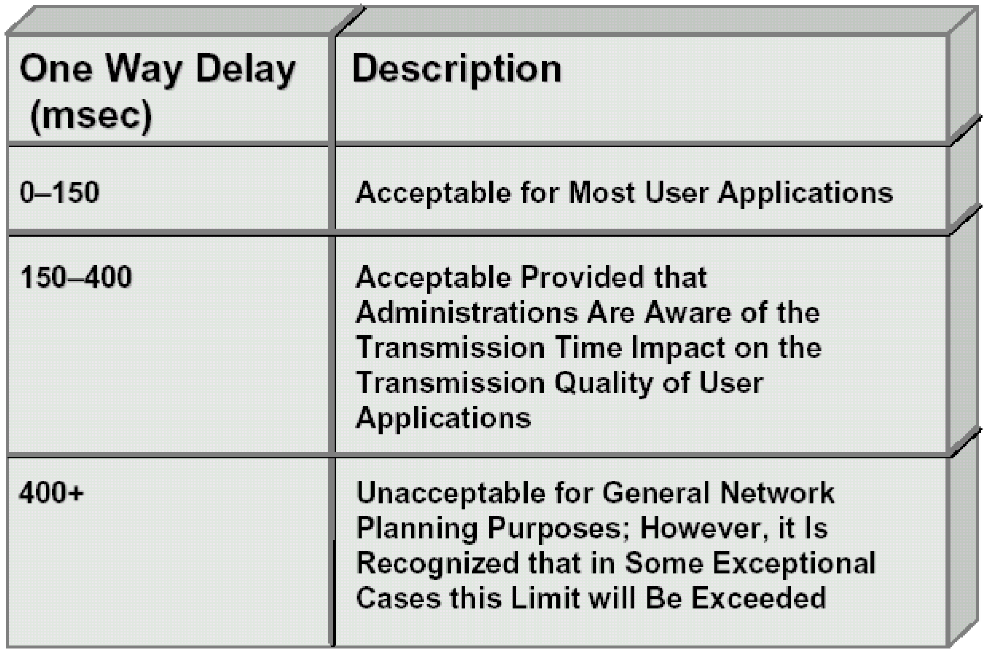
Example: VoIP (1/2)

- Voice over IP (VoIP)
  - Real time application
    - ✓ Not delay tolerant
  - Loss tolerant
    - ✓ E.g. 1% packet loss; even more if FEC (Forward Error Correction)
  - Adaptive
    - ✓ Packet size, codec, rate, playout delay may change during the call
- Performance affected by
  - Bandwidth =f(codec)
  - Delay, Jitter and Packet loss

Example: VoIP (2/2)

- **QoS index, in terms of:**
  - **Call setup time**
  - **One way delay (e.g., recommendation ITU G.114)**
  - **Packet loss**

ITU G.114



One Way Delay (msec)	Description
0–150	Acceptable for Most User Applications
150–400	Acceptable Provided that Administrations Are Aware of the Transmission Time Impact on the Transmission Quality of User Applications
400+	Unacceptable for General Network Planning Purposes; However, it Is Recognized that in Some Exceptional Cases this Limit will Be Exceeded

Multimedia Over Today's Internet

Internet's IP: "best-effort service"

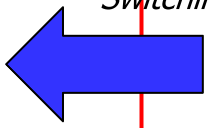
- ***no*** **guarantees on delay, loss**

But you said multimedia apps requires QoS and level of performance to be effective!

Two ways to provide QoS:

- To introduce mechanisms at network level to guarantee the QoS

- To use application-level techniques to mitigate (as best possible) effects of delay, loss,..



Adapted from: Kurose & Ross

Network QoS mechanisms

Packet Scheduling

Scheduling Algorithms

- **Work-conserving**
  - FCFS
  - PQ
  - RR, WRR
  - GPS, Bitwise-RR (theoretical)
  - WFQ, PGPS
  - DRR
  - Virtual Clock
  - Delay-EDD
  - CBQ
  - Worst-case Fair WFQ (WF2Q)
  - SCFQ (Self-Clocked FQ)
  - Start Time FQ
  - Core State FQ
  - …

- ***Non-Work-conserving***
  - *Jitter-EDD*
  - *Stop-and-Go*
  - *HRR (Hierarchical RR)*
  - *Rate-Controlled Static priority*

Goals:

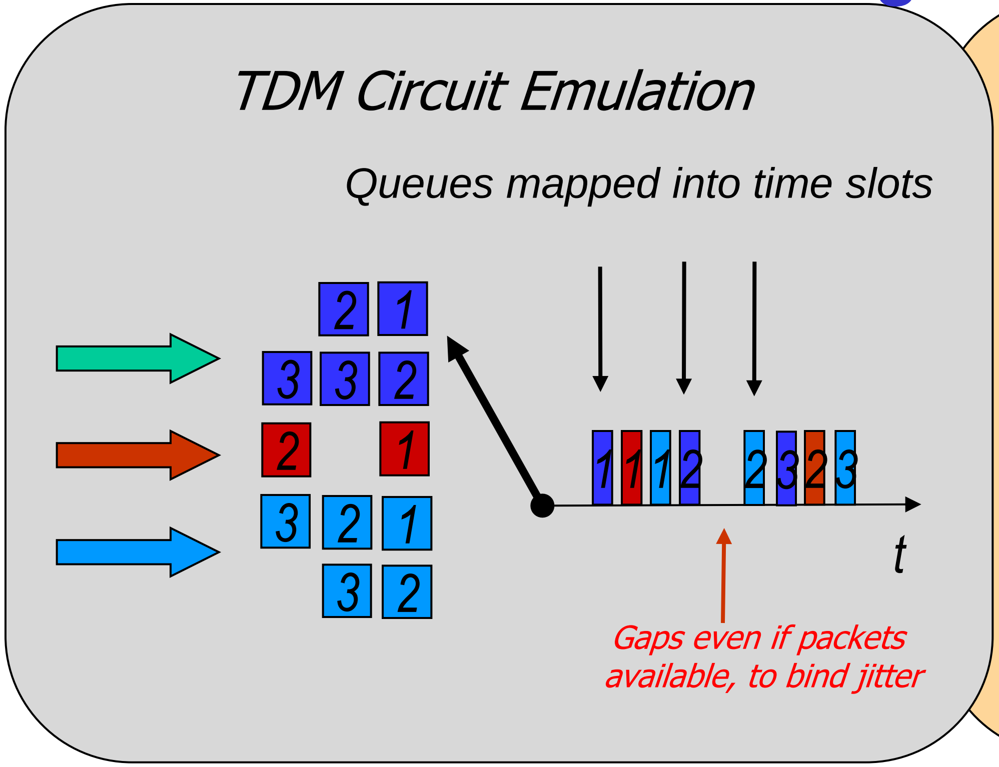
Fairness

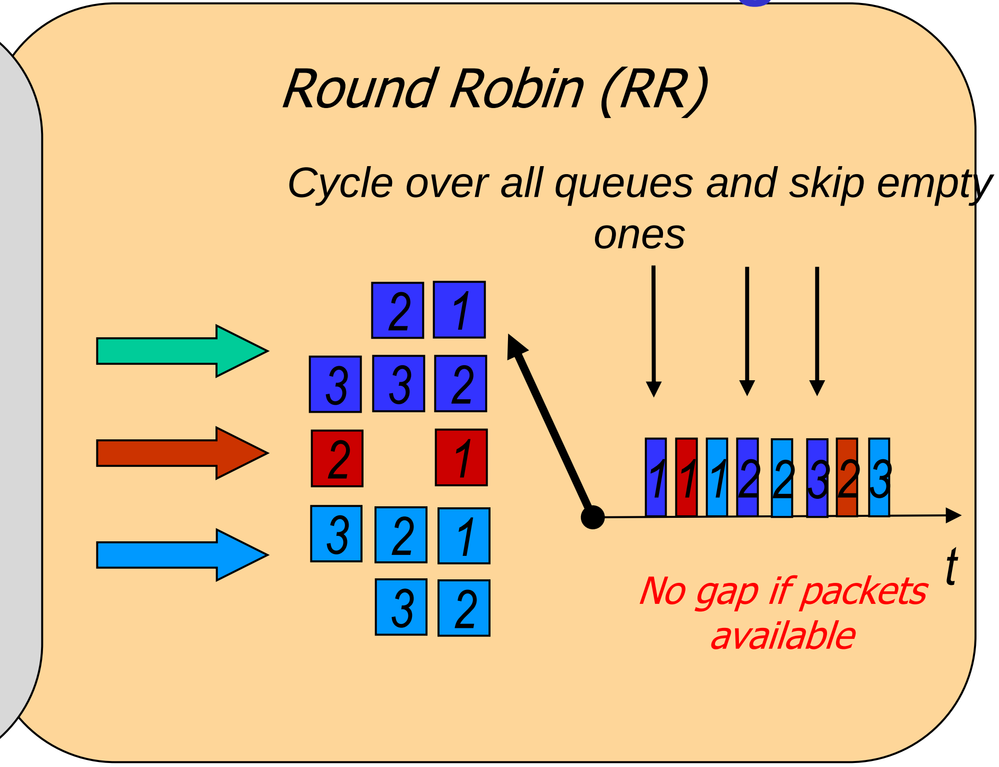
Traffic Control:
- *Rate (avg,peak,mbs)*

QoS: (abs. or relative)
- *Delay*
- *Jitter*
- *Loss*

Scheduling: rate allocation examples

- **Non-work-conserving**
- **Work-conserving**

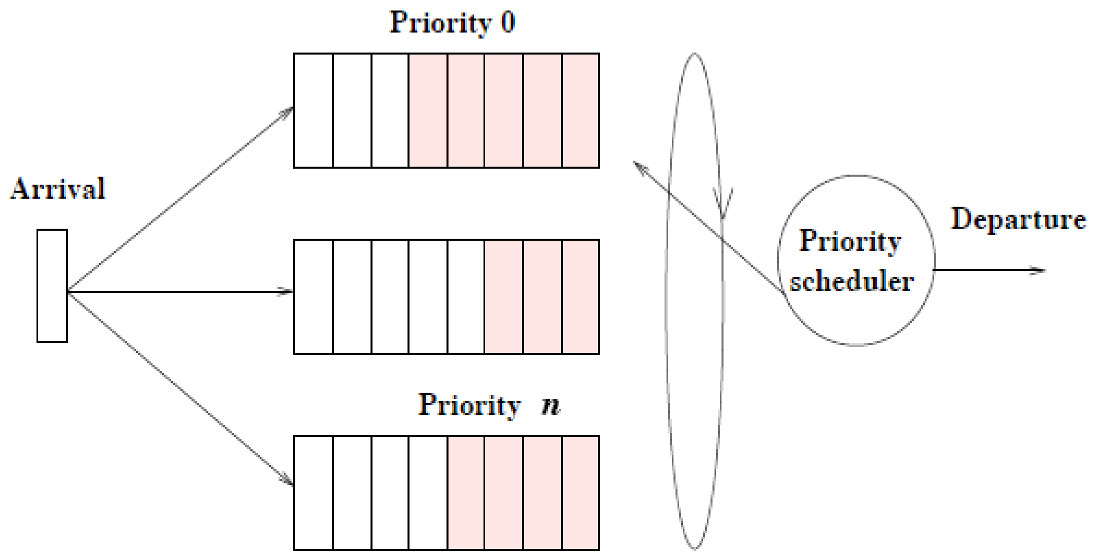

TDM Circuit Emulation
Queues mapped into time slots
2 1
3 3 2
2 1
3 2 1
3 2
1 1 1 2 2 3 2 3
t
Gaps even if packets available, to bind jitter


Round Robin (RR)
Cycle over all queues and skip empty ones
2 1
3 3 2
2 1
3 2 1
3 2
1 1 1 2 2 3 2 3
t
No gap if packets available

Fixed bandwidth

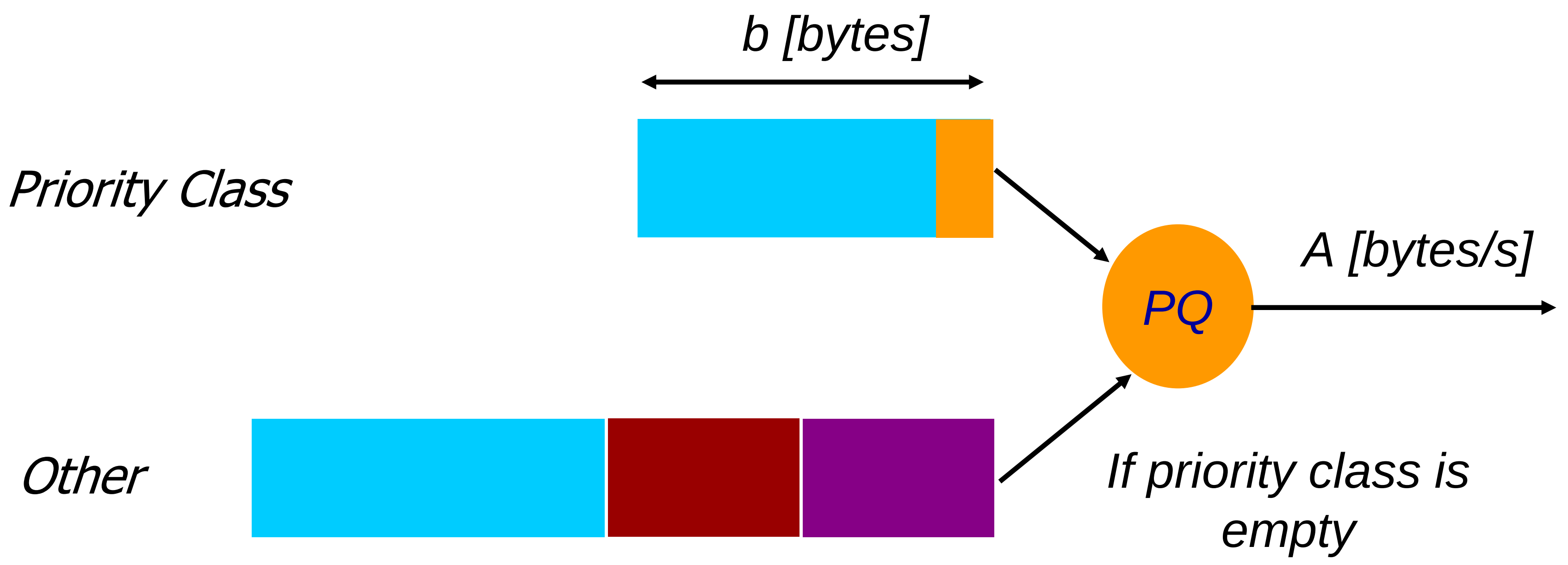
Variable bandwidth with guaranteed minimum

Priority Queueing



- Priority i is served only if all lines j<i are empty.
  - The highest priority has the least delay, highest throughput, and lowest loss.
  - Potential of starvation of lower priority classes (i.e., the server will be unable to serve the lower class because it is always busy serving the higher class).
  - Packets from a given priority queue are served on a First Come First Served (FCFS) basis.

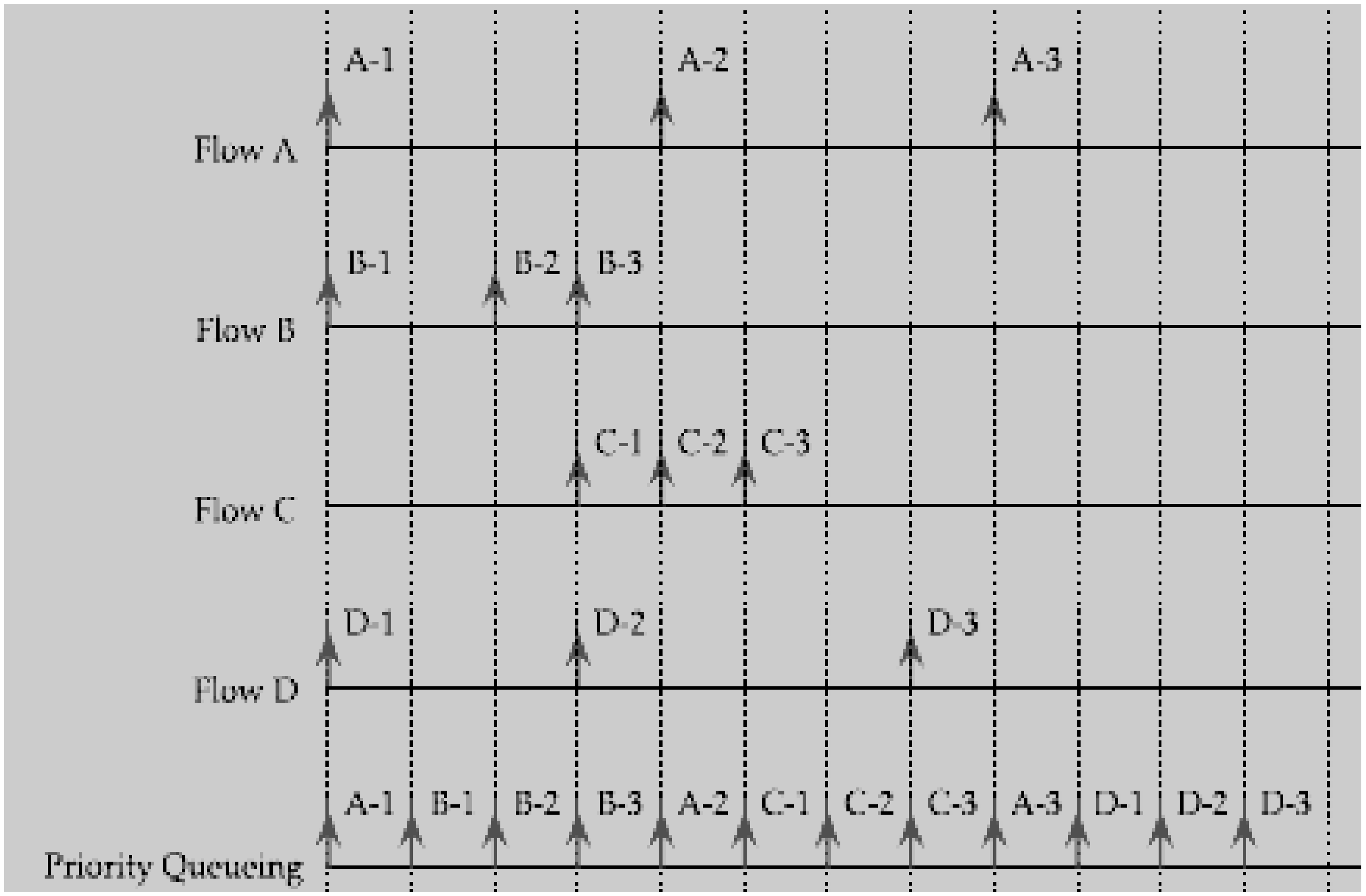
Priority Queueing



Max_Queueing_Delay_Priority = MTU/A + b/A

Max_Queueing_Delay_NoPriority = ∞

Example: Priority Queueing



Max-min fair sharing

- Objective: maximize the share of the sessions that demand fewer resources
  - ❖ The scheme allocates the smallest of all demands from all flows first. Remaining resources are distributed equally among competing flows.
- Principles:
  - ❖ All flows receive at least the *fair share* ((total capacity)/(total number of flows))
  - ❖ The unused *fair share* capacity is divided equally between all flows requiring resources above the *fair share*.
    - ✓ Independently of how many resources are demanded
      - ➢ Note that in FCFS the allocation is proportional to demand

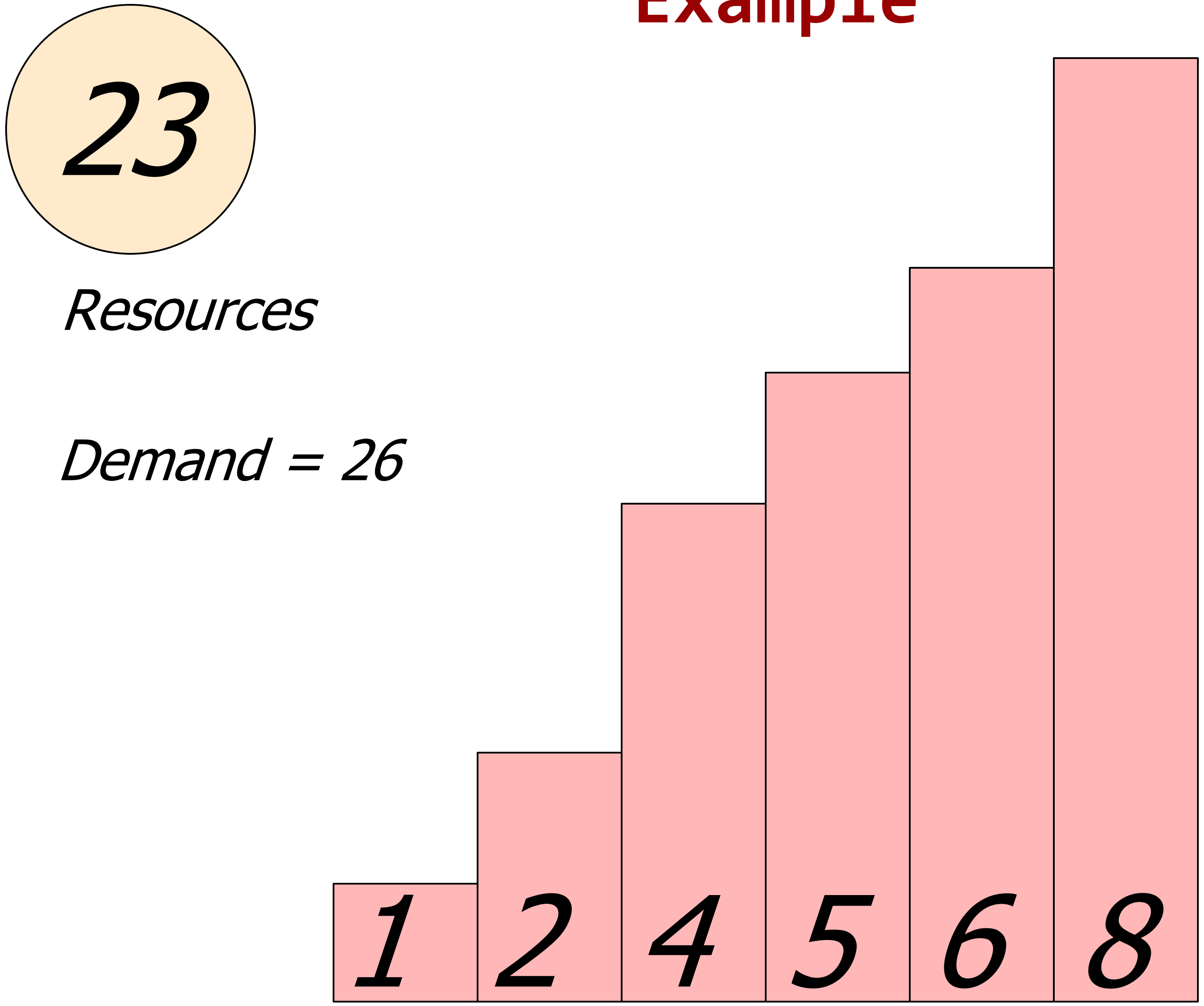
max-min fair share calculation

1. Calculate the initial fair share
   - ❖ Fair share= (total capacity)/(total number of flows)
2. For all flows that have a demand equal to, or less than the fair share, allocate the flows' actual demand
3. Remove the flows satisfied and subtract the allocated capacitites from the total available capacity
4. Repeat steps 2 and 3 for the remaining (unsatisfied) flows with a new

Fair_share =(remaining capacity)/(remaining number of flows)

until all remaining demands are larger than the current fair share or no unsatisfied demand remains.

Example

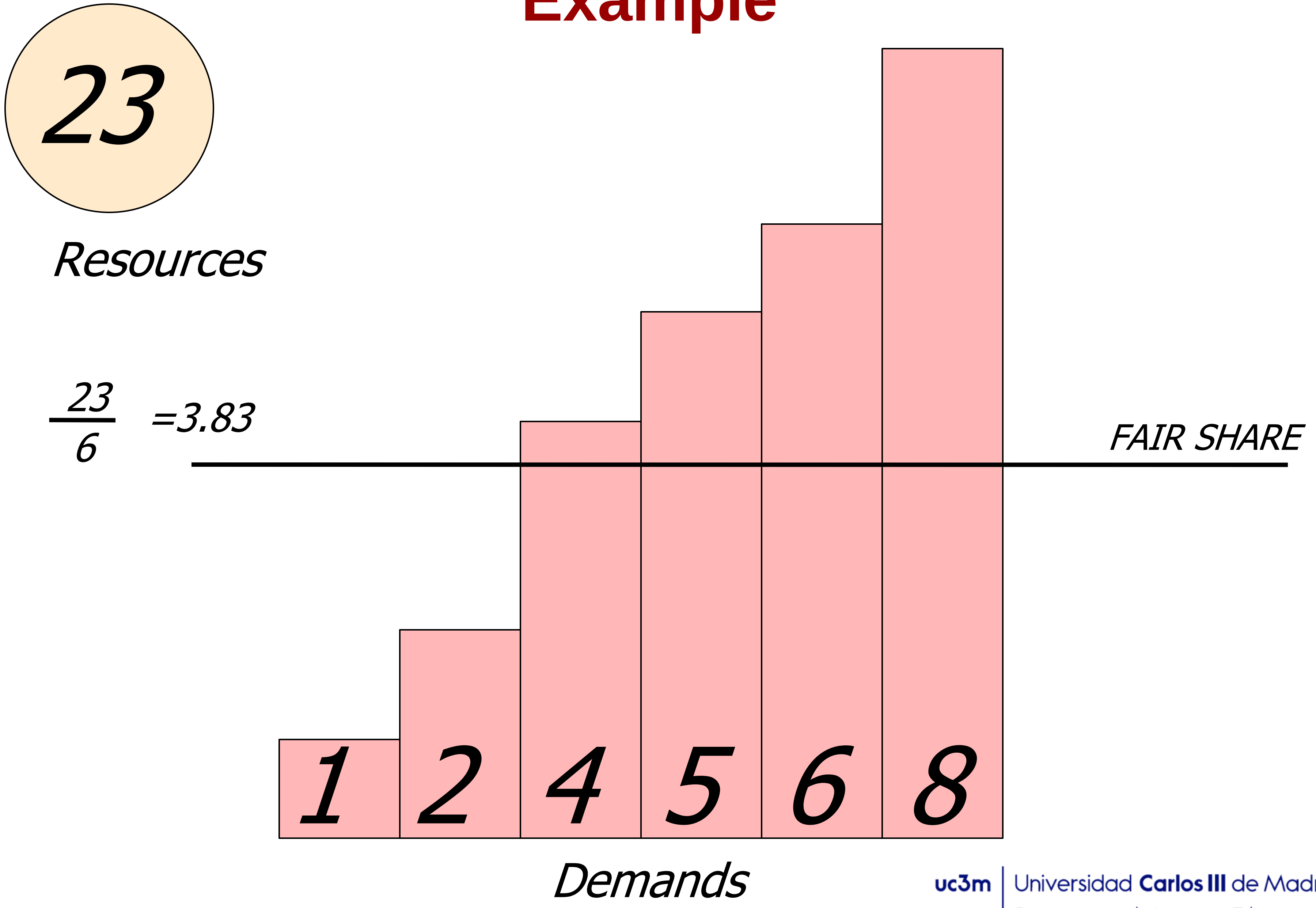


Resources

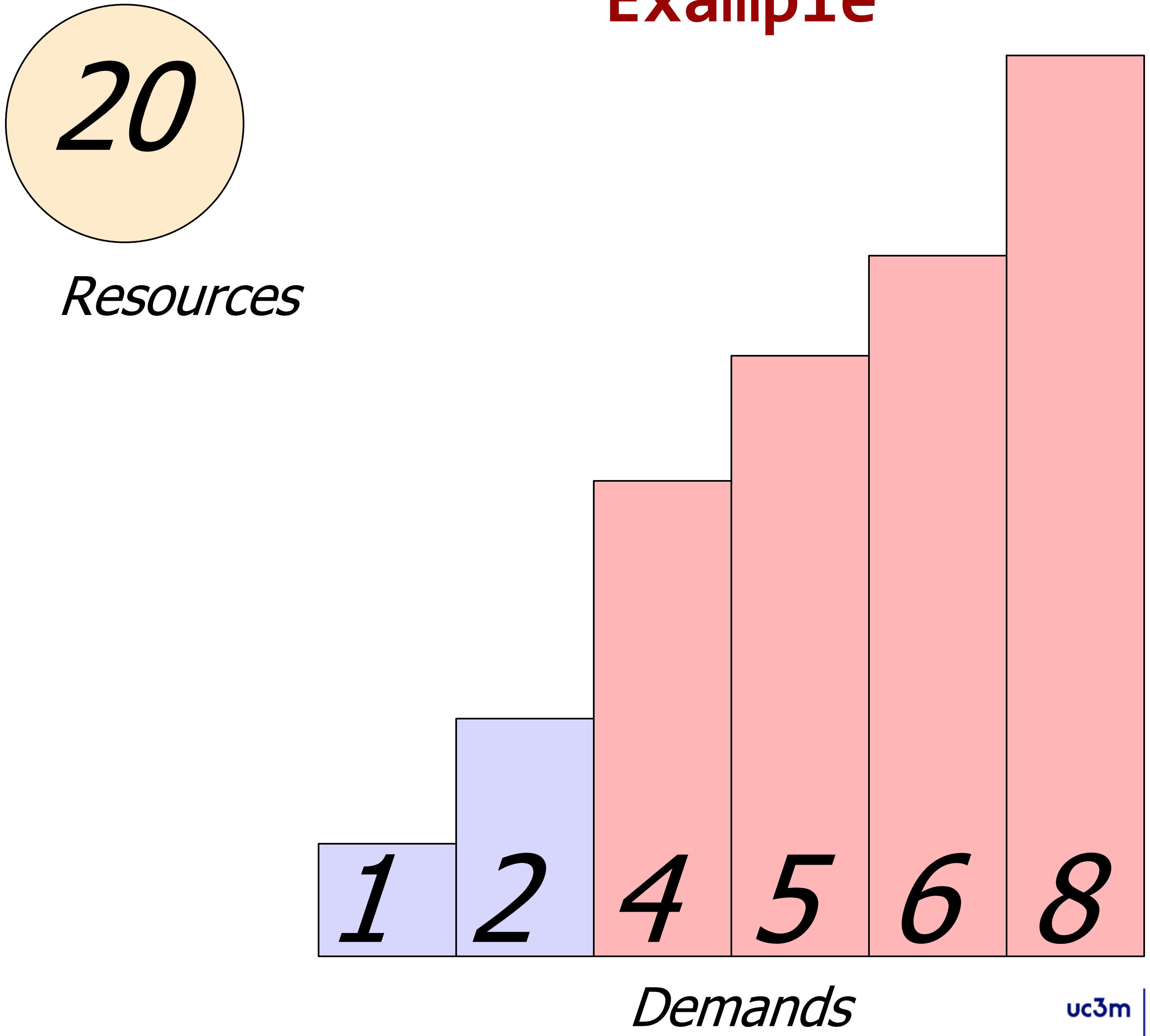
Demand = 26

Demands

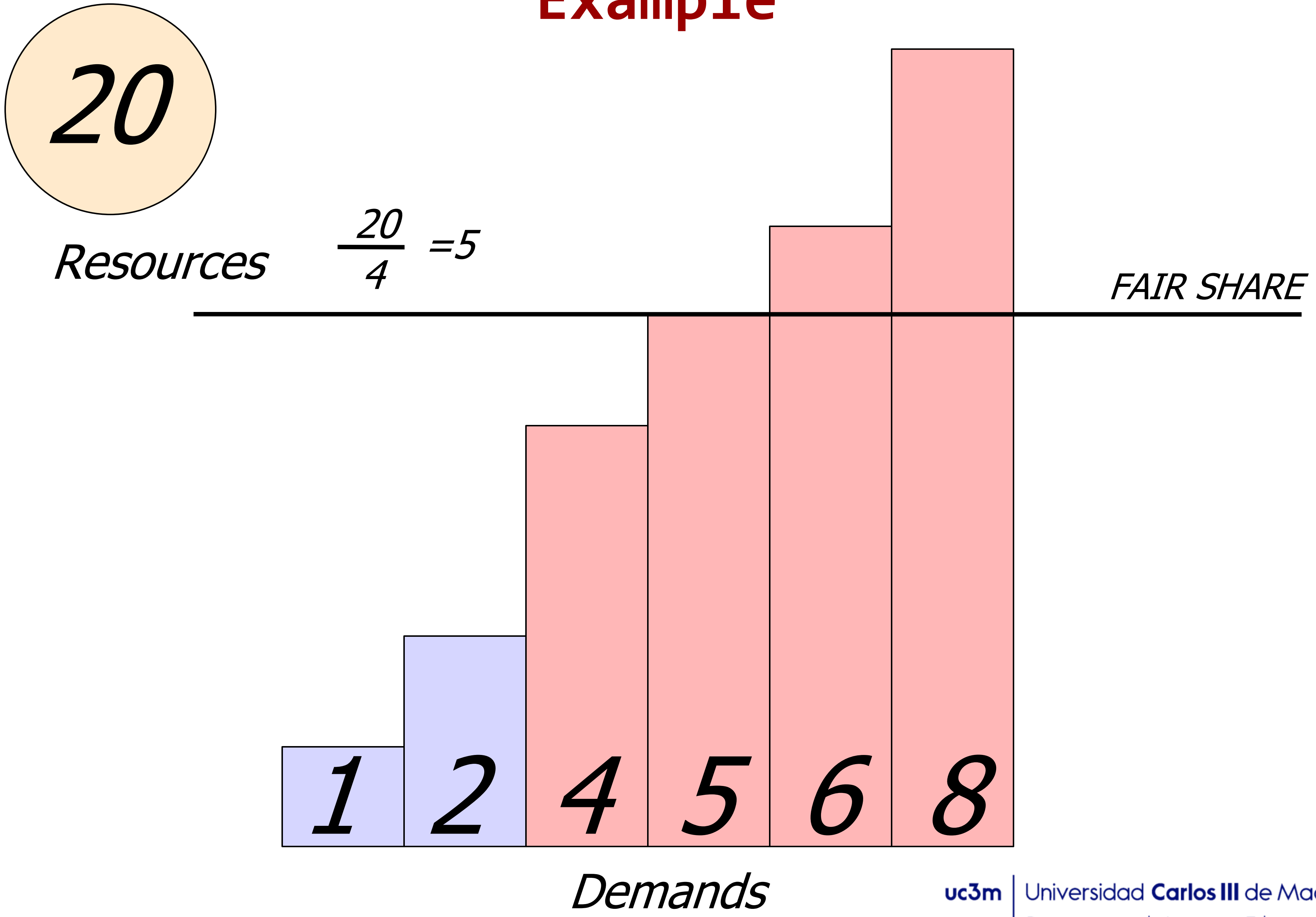
Example



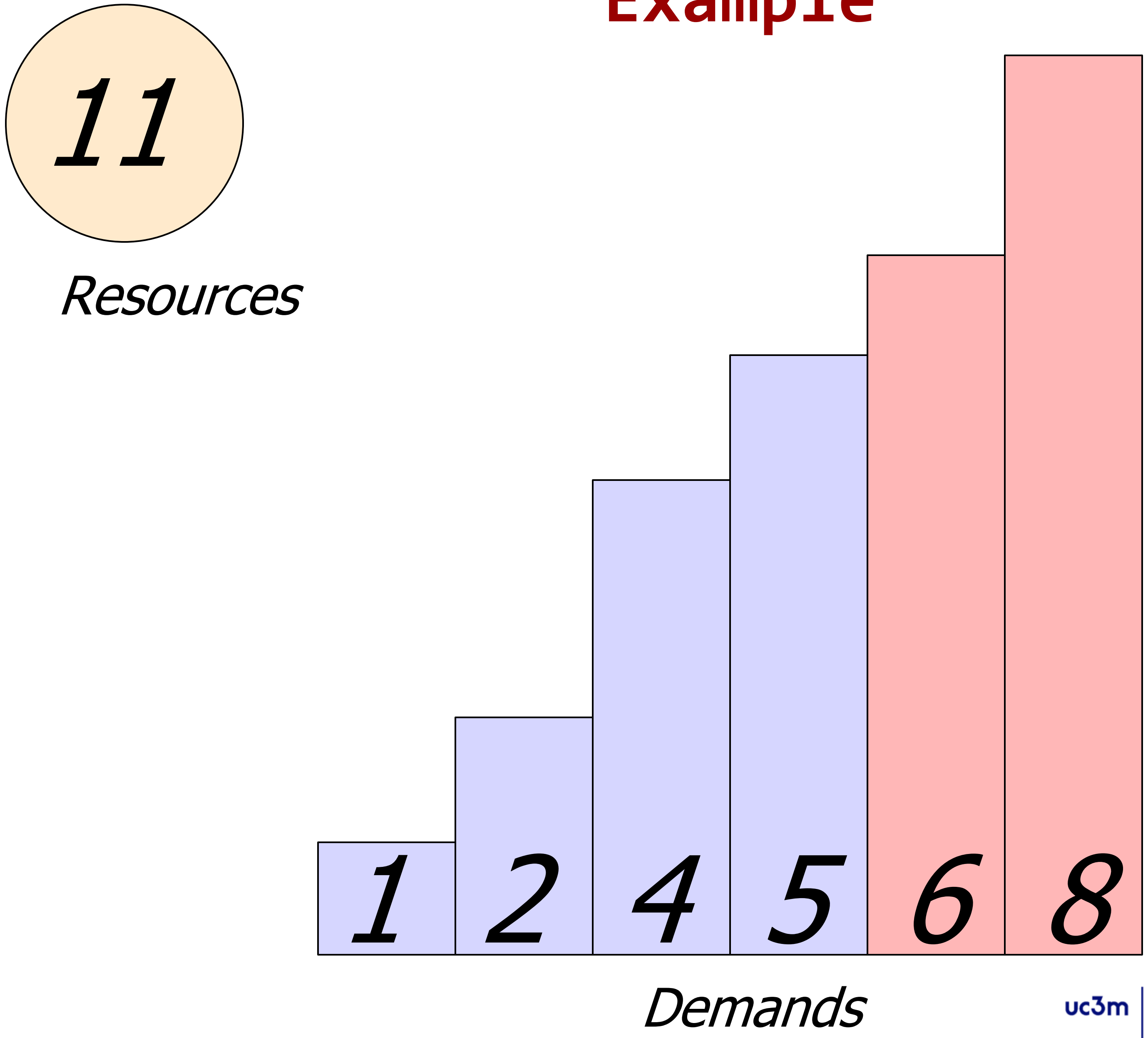
Example



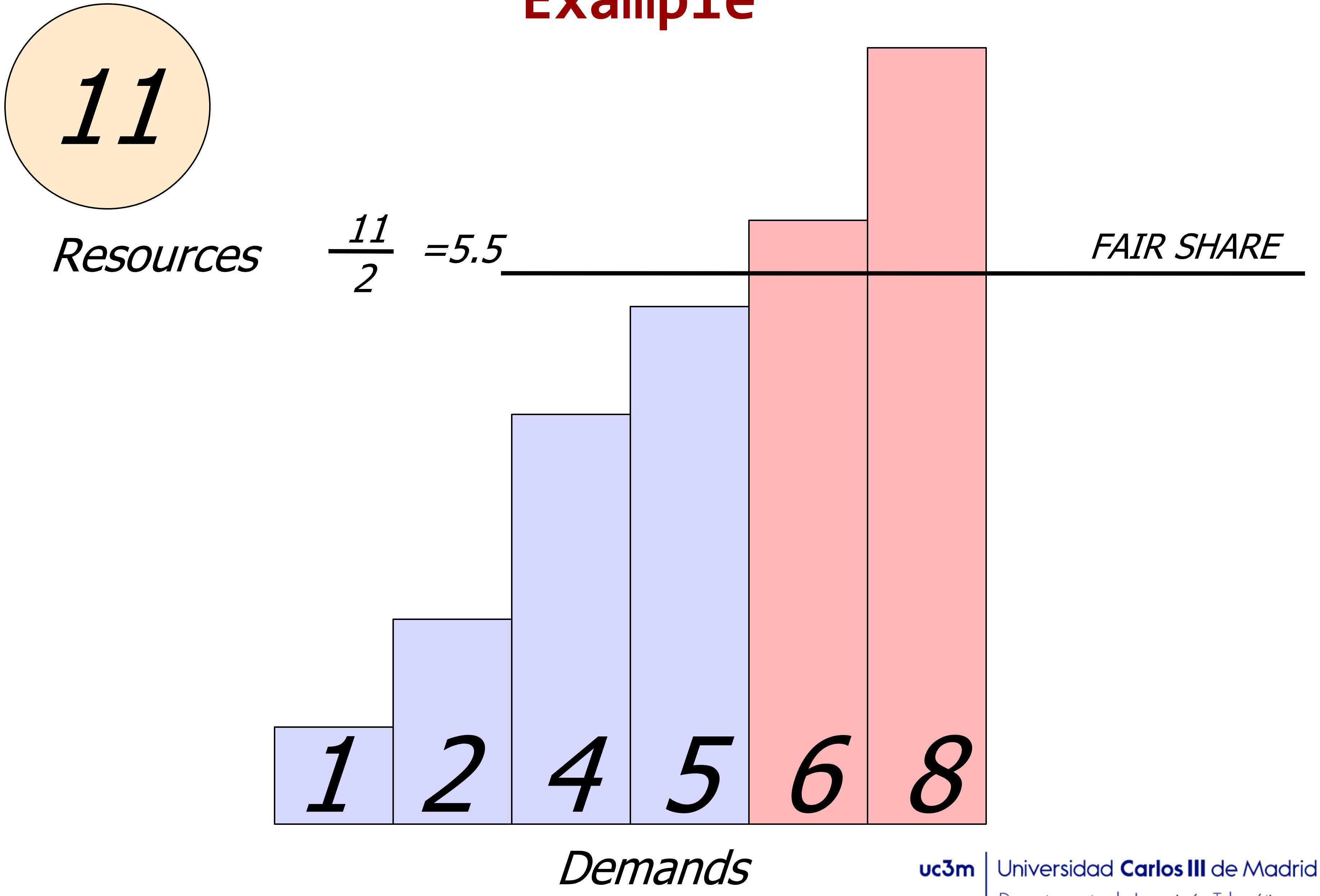
Example



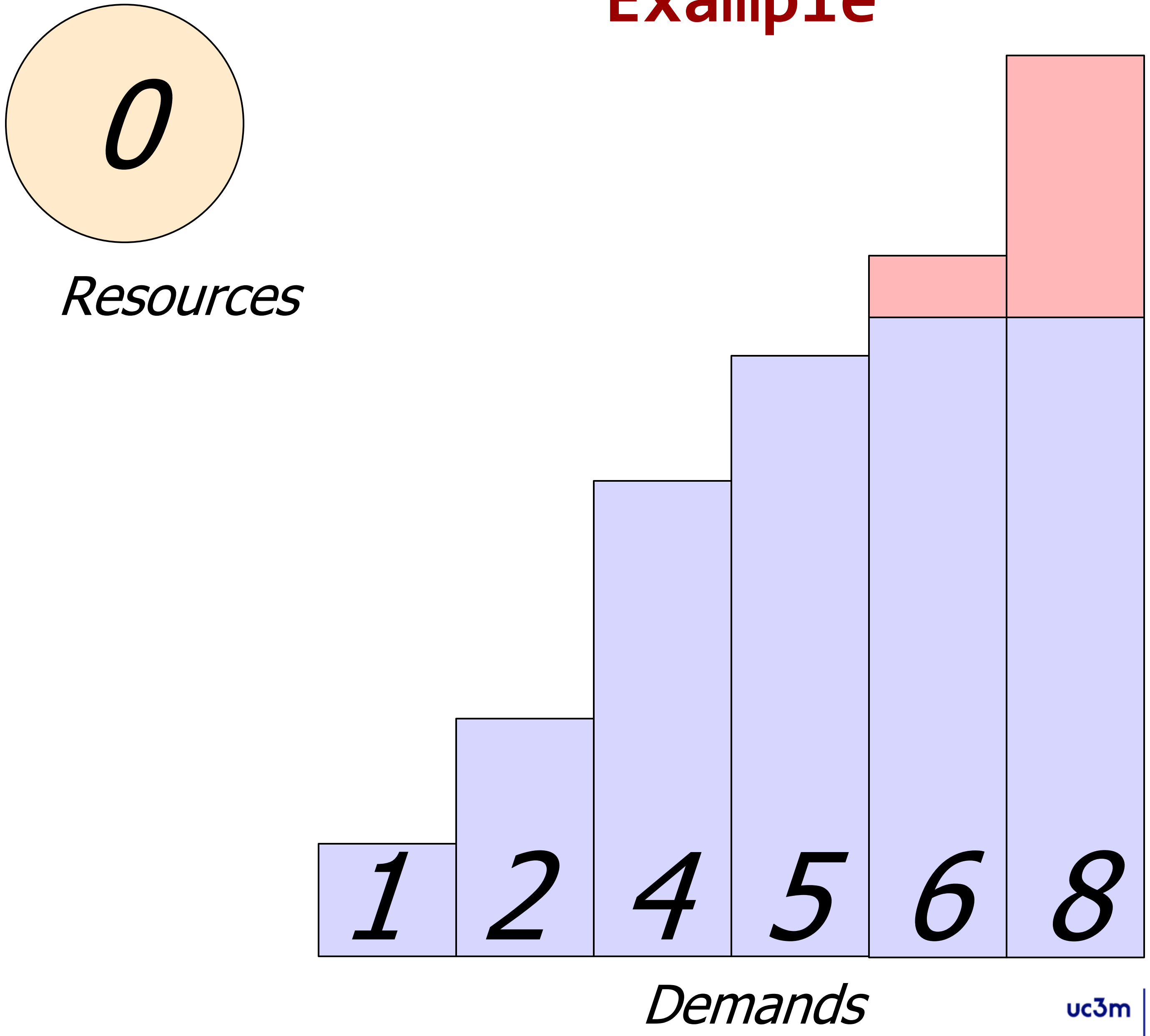
Example



Example



Example



Properties of max-min fair share

- All flows that demand less than the fair share will be satisfied.
- Unused resources from these flows will be re-distributed to remaining, more 'greedy' sources, which request more than their fair share.
- Distribution of the remaining resources is according to the actual demand of the greedy sources.
- If some flows should be allocated a bigger share of resources than others, associate a weight to each flow to indicate the relative sharing. The max-min fair sharing can be extended by simply normalizing the demand with the corresponding weight for the flow.

Scheduling of Packets: GPS

- Generalized Processor Sharing (GPS) (Parek & Gallager 93)
  - ❖ Ideal fair-queuing algorithm
  - ❖ Based on fluid model (not real), work conserving
  - ❖ Provides exact max-min weighted fair share
  - ❖ Distributes bandwidth SIMULTANEOUSLY between different sessions
    - ✓ Each session i has its own queue of sufficient size
      - ➢ For isolation of badly-behaving flows to achieve fairness!
    - ✓ Each session is assigned a weight (with a global scope) w_i (normally $0 \le w_i \le 1$)
    - ✓ At every instant t, all non-empty queues are served simultaneously, in proportion to their weight

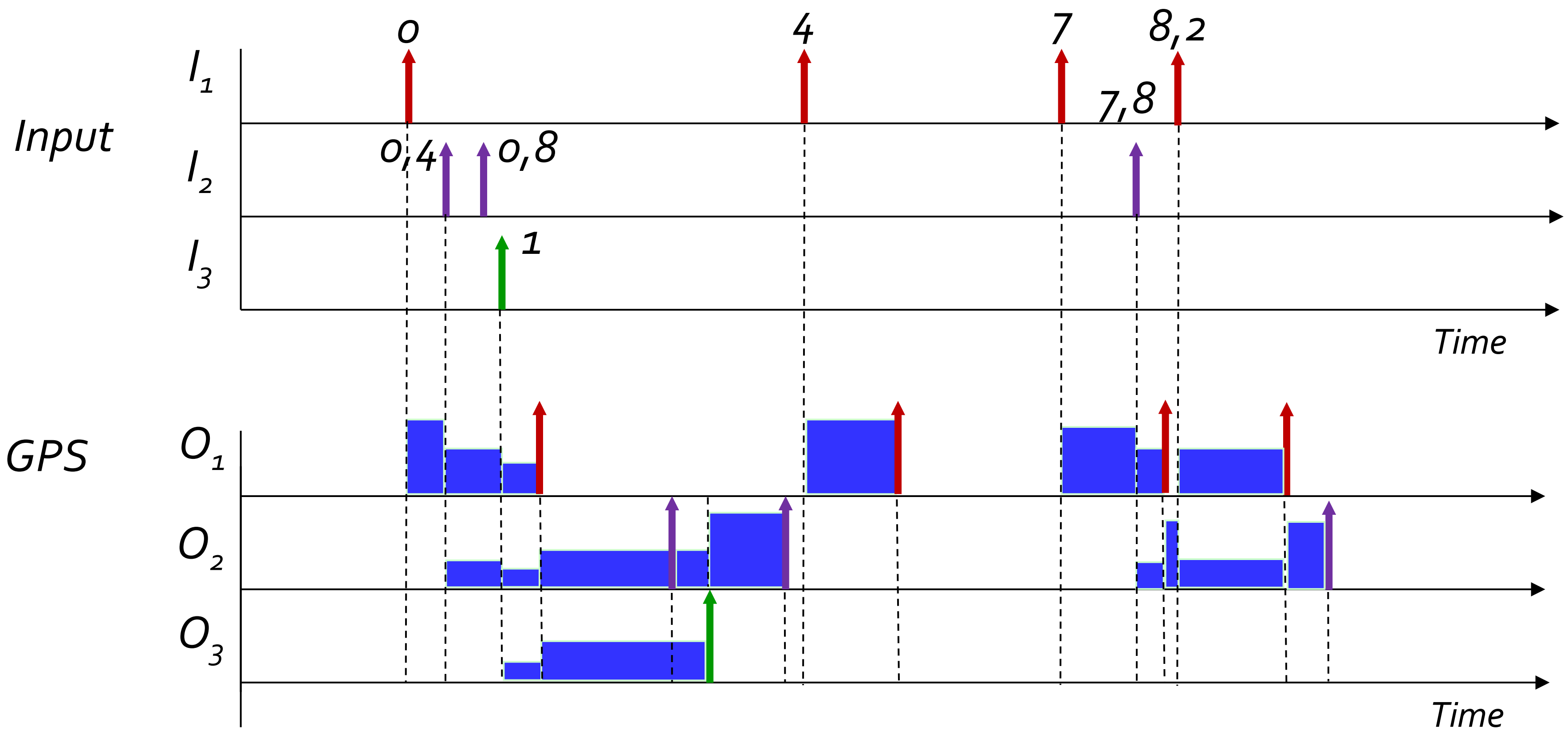
$$R_i = A \cdot \frac{w_i}{\sum_{j \in B_t} w_j}$$

R_i: assigned bandwidth
A: total link bandwidth
B_t: set of active flows

    - ✓ Worst case bandwidth assignment: Receive A * w_i

GPS - Example

- *Example: w_1=0,5; w_2=0,25; w_3=0,25*



GPS Scheduling: Guarantees

- Assumption: Traffic generated by a token bucket (r_i, b_i)
- Guarantee for the assigned bandwidth if
  - $r_i < A * w_i = R_i$
- Guarantee for the queuing delay:

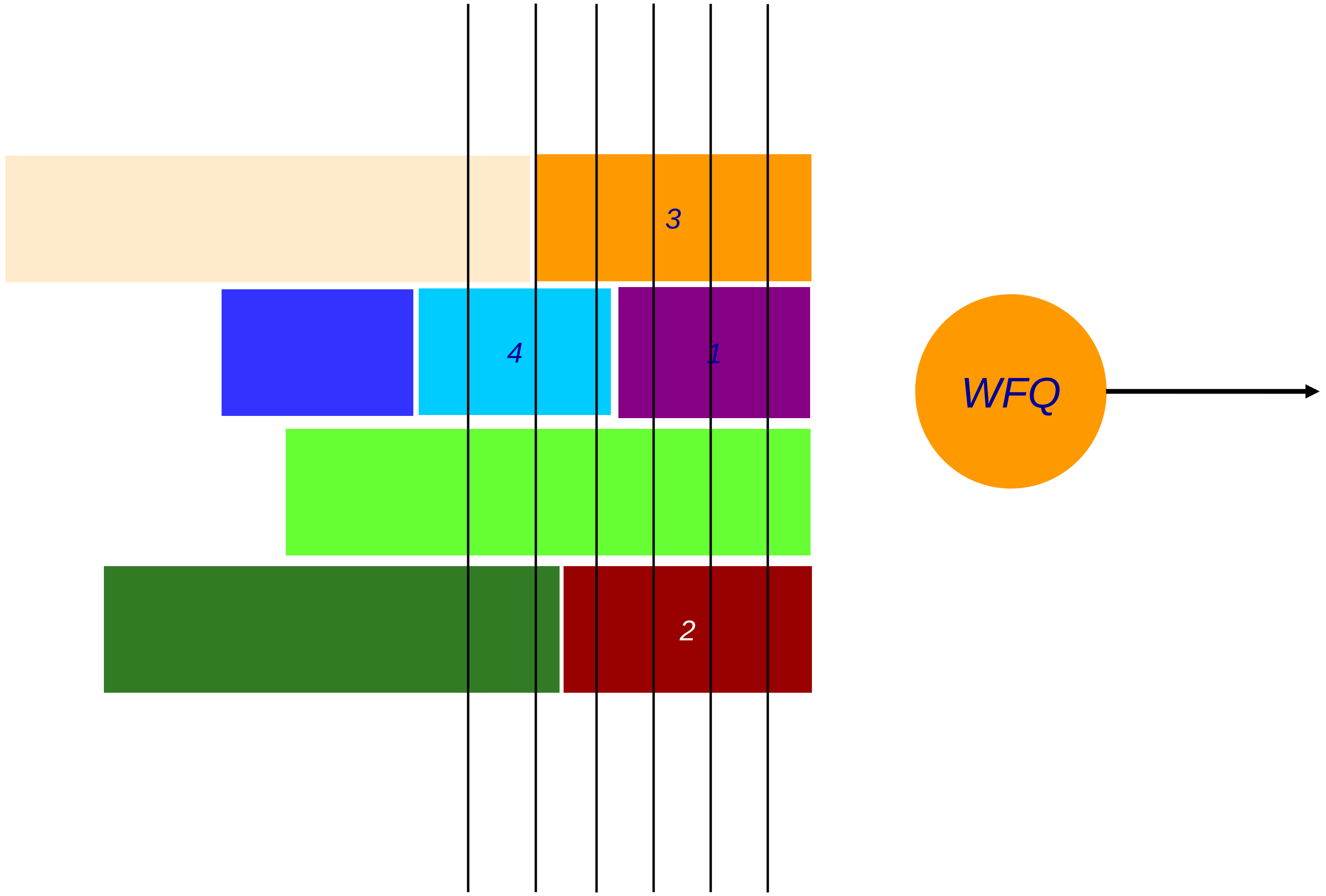
$$\max\_delay_{GPS} = \frac{b_i}{A * w_i}$$

- Token bucket + GPS (with adequate parameters) = guarantee of worst-case bandwidth and delay
  - Necessary to assign a weight w_i to each flow
    - ✓ Used to calculate a bandwidth assignment $A * w_i = R_i$
  - Delay depends directly on the assigned bandwidth
    - ✓ Since it determines the time it takes to empty the buffer

Packet Scheduler for GPS: WFQ

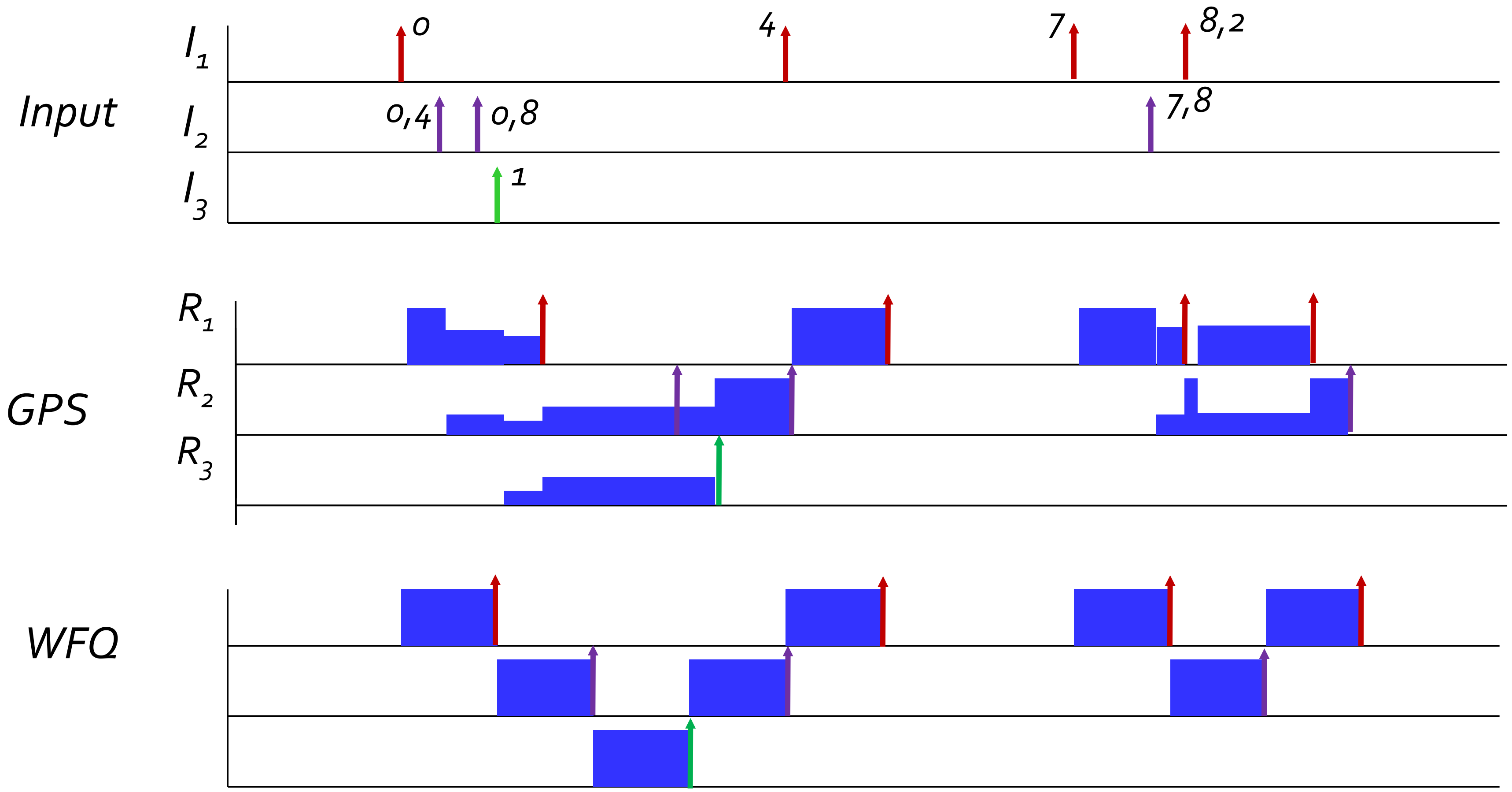
- In reality it is not possible to apply a continuous model like GPS
- WFQ (Weighted Fair Queuing) Demers, Keshav, Shenker, 89
  - ❖ Work conserving algorithm that allows the emulation of GPS in a discrete environment (packet-by-packet version of GPS).
  - ❖ Algorithm:
    - ✓ From all available packets at a given moment select that packet whose processing would have been completed first under GPS scheme and send it.
      - ➢ Tries to implement a Bitwise round robin
  - ❖ Recalculation of GPS departure times is needed on each event
    - ✓ Packet arrival & packet departure
    - ✓ Requires to manage an ordered list of packets
  - ❖ Visit all queues in round robin order.

WFQ



WFQ Scheduling: Example

- **Example: w1 = 0,5; w2 = 0,25; w3 = 0,25**



The worst-case delay in WFQ

Developing the theory for GRSA (Guaranteed Rate Scheduling Algorithm), and considering that the generated traffic is shaped according to a token bucket
Due to the packetization effect, the worst-case delay for WFQ is slightly larger than for GPS:

(Parek & Gallager 93)

$$\max\_queueing\_delay_i = \frac{b_i}{R_i} + \frac{C_{total_i}}{R_i} + D_{total}$$

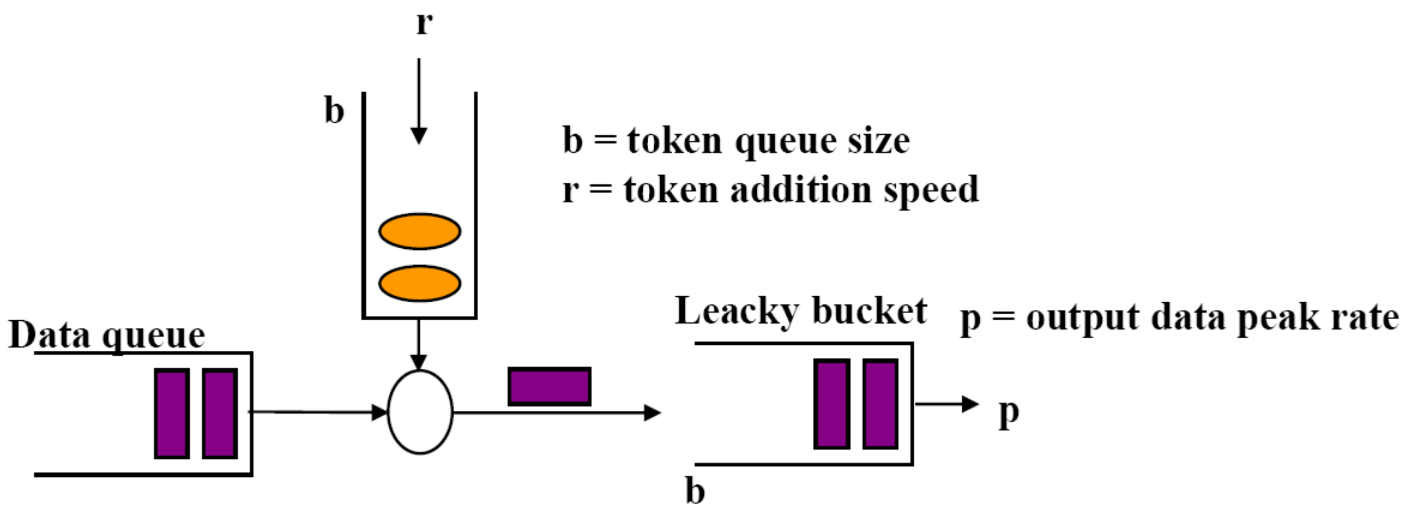
$$C_{total_i} = (K-1)M_i \quad D_{total} = \sum_{k=1}^{K} (M^k / A^k)$$

K = # hops
A^k = channel bandwidth of link k
M_i = max. packet size of flow i
R_i = guaranteed reserved rate for flow i
b_i = token bucket buffer size
M^k = max. packet size in link k (MTU)

Explanation:
- *The first term is the delay bound of a GPS system.*
- *The second item is the extra delay a packet may experience, if it arrives just after it would have been served under GPS system (Queueing delay at each Kth hop).*
- *The third item reflects the fact that packets are served one-by-one, so a packet may have to wait until the current packet is served (in the worst case MTU/A per hop).*

WFQ packet scheduling

- Model extension: Token Bucket + Leaky Bucket



- Limiting the peak rate improves delay estimation

$$\max\_queueing\_delay = \frac{(b_i - M_i)(p_i - R_i)}{R_i(p_i - r_i)} + \frac{(M_i + C_{tot})}{R_i} + D_{tot} \quad (p_i > R_i \geq r_i)$$

$$\max\_queuing\_delay = \frac{(M_i + C_{tot})}{R_i} + D_{tot} \quad (R_i \geq p_i \geq r_i)$$

Round Robin Family

Round Robin

- The scheduler maintains one queue for each flow.
- Each incoming packet is placed in an appropriate queue.
- Queues are served in a round robin fashion
  - One packet from each nonempty queue in turn
  - Empty queues are skipped over.

Weighted Round Robin

- n packets are served per turn
- n is adjusted to allocate a specific fraction of link bandwidth to that queue.

If packet sizes are variable there is a fairness problem.

Deficit Round Robin

- A variable is initialized to represent the number of bits to be served from each queue.

Source: Jha Hassan

Deficit Round Robin (DRR)

- Deficit Round Robin(DRR) (Shreedhar, Varghese 1995)

- Initialization()

```
 for (i=0; i < n; i++){
  Quantum[i] = Queue Quantum[i];
  DC[i] =0;
 }
 Enqueuing();
```

- Enqueuing()

```
 while ( (packet=read_packet())!=null ){
   i = Compute_Class(packet);
   if ( ! ExistsInActiveList(i) ) {
           InsertInActiveList(i);
           DC[i]=0;
   }
   Enqueue(packet,Queue[i]);
 }
 Dequeuing();
```

Deficit Round Robin (DRR) (2)

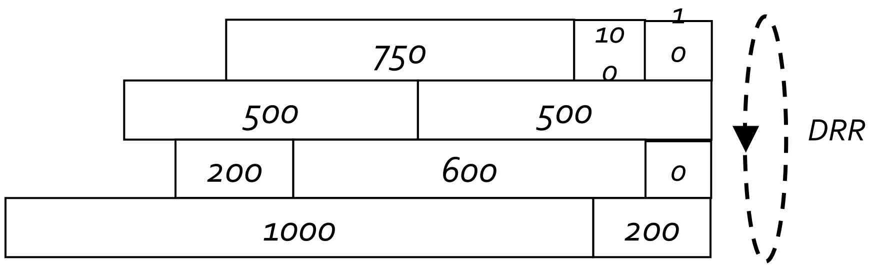
- Dequeuing ()

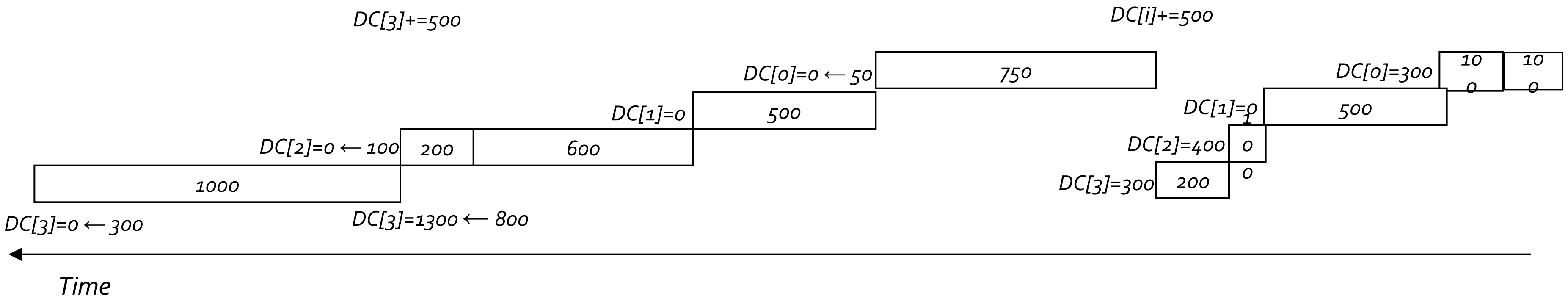
```
while (true) {
  if  ( ! IsEmptyActiveList() ) {
       i = RemoveHeadActiveList();
       DC[i] = DC[i] + Quantum[i];
       while ( (DC [i] > 0) && ! IsEmpty(Queue[i])  ){
            packet_size = sizeof( Head(Queue[i]) );
            if ( packet_size <= DC[i] ) {
               SEND( DeQueueHead(Queue[i]) );
               DC[i] = DC[i] - packet_size;
            else
               break;
        }
        if  ( IsEmpty(Queue[i]) )
             DC[i]=0;
        else
             InsertInActiveList(i);    /* at the end */
 } Enqueuing();
}
```

uc3m | Universidad Carlos III de Madrid
Departamento de Ingeniería Telemática

Deficit Round Robin (DRR): Example

`For all i, Quantum[i]=500`





uc3m Universidad Carlos III de Madrid
Departamento de Ingeniería Telemática

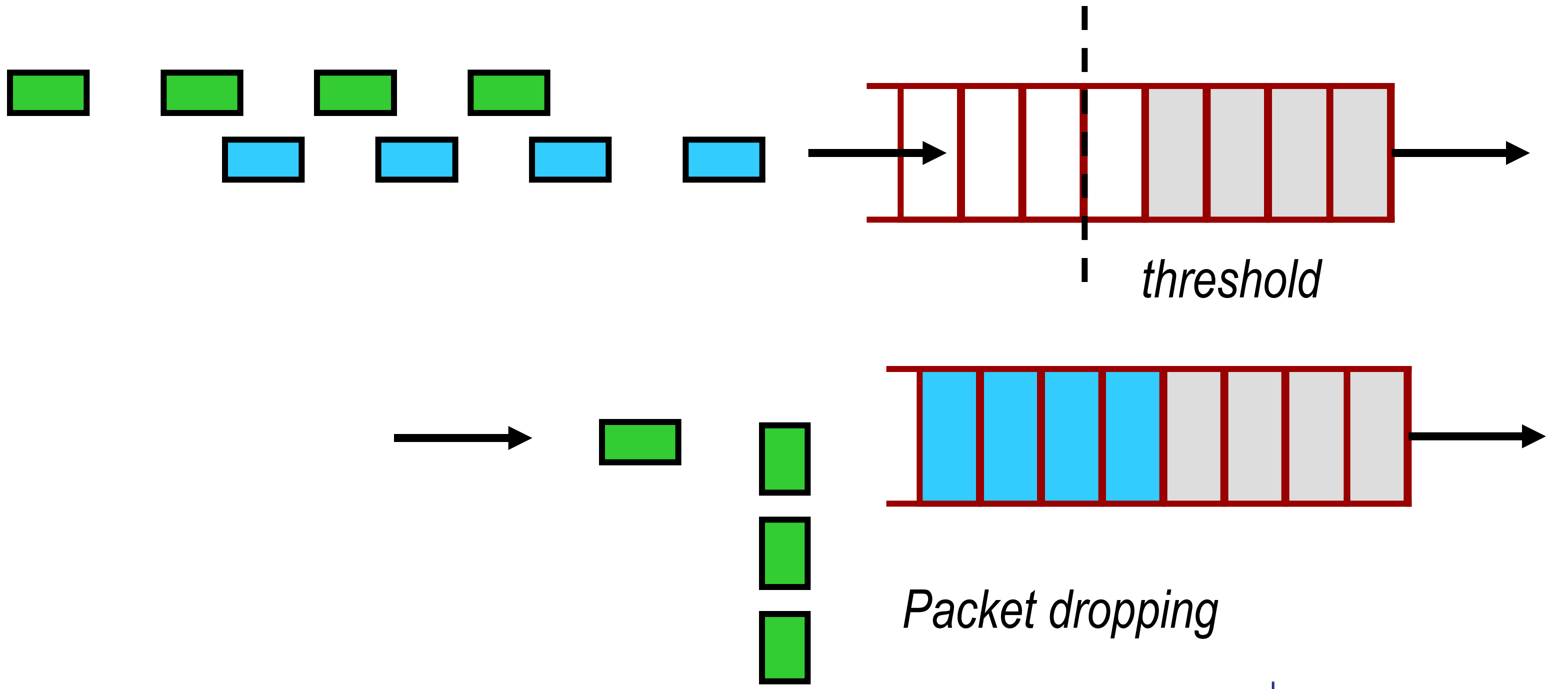
Network QoS mechanisms

Queue Management

Objective of a packet buffer: absorbing transient packet bursts !
What happens if the buffer is always full ?

Packet dropping

- **Forced dropping**
  - **Buffer overflow**
- **Preventive dropping**
  - **Before overflow**



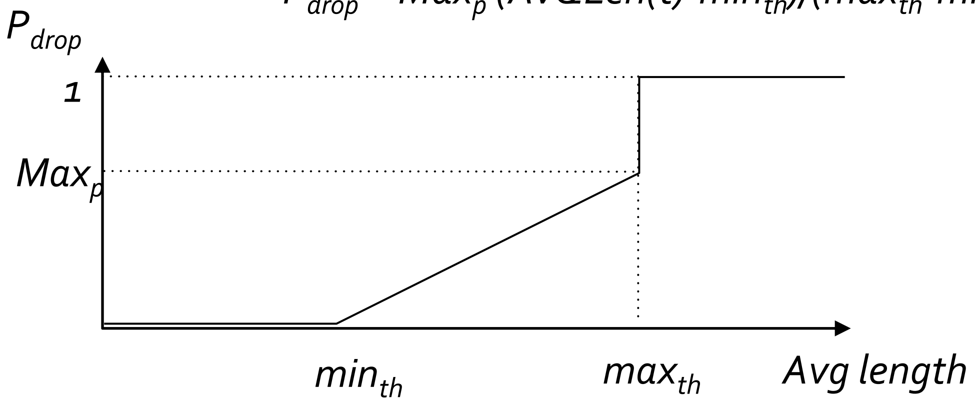
Packet dropping

- **Tail-drop**
  - **Arriving packet is discarded**
- **Head-drop**
  - **First packet is discarded**
  - **Slightly improves TCP performance**
- **Problems**
  - **Not fair dividing between flows**
    - Tail drop: larger flows fill and monopolize queue
  - **TCP global Synchronization**
    - Oscillation between under-utilization and congestion
  - **High mean delay if queue always full**
- **Solution: Active Queue Management (AQM)**
  - **Monitor the queue size proactively and starts marking and dropping packets before severe congestion occurs.**

Queue management: RED (Floyd, van Jacobson '98)

- Random Early Detection (RED): drop packets with probability that depends on the average queue size
  - ❖ **Average queue length calculation: exponentially weighted average, estimated every interval**
    - ✓ **EWMA (exponential weighted moving average)**
      - ➢ AvQLen(t) = (1-w)*AvQLen(t-1) + w *QLen(t), example w = 0,003
  - ❖ **Packet drop probability**

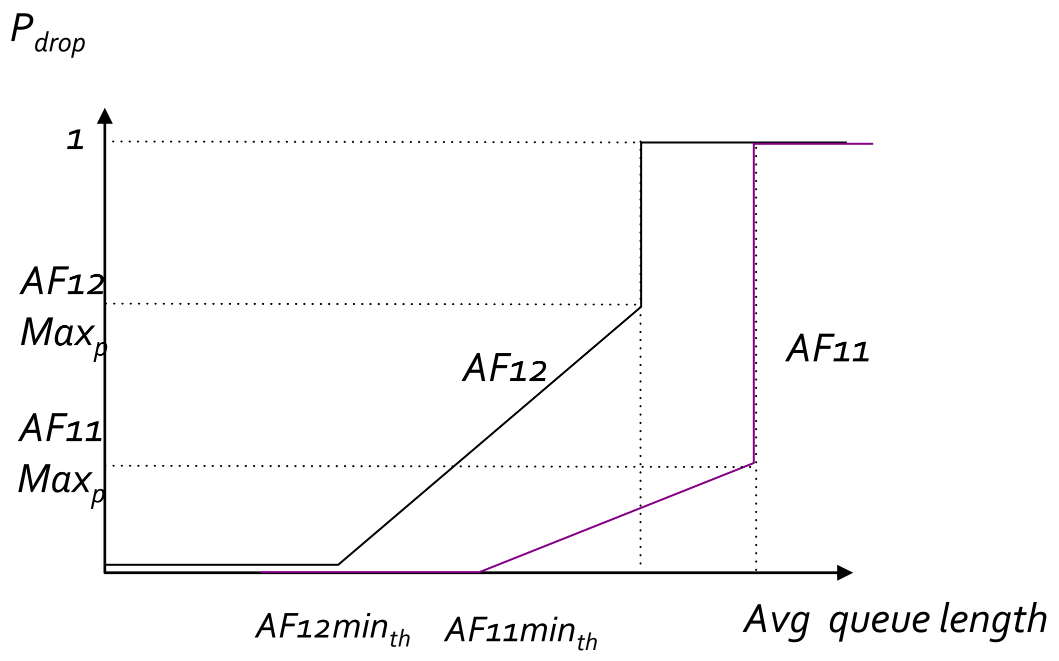
$$P_{drop} = Max_p\,(AvQLen(t)-min_{th})/(max_{th}-min_{th})$$



Effects Avoid permanent overflow, reduce average latency, absorb instantaneous bursts, lower loss, spread losses over multiple sources, reduces synchronisation

WRED (Weighted RED)

- Allows to differentiate packet drop probability between traffic classes



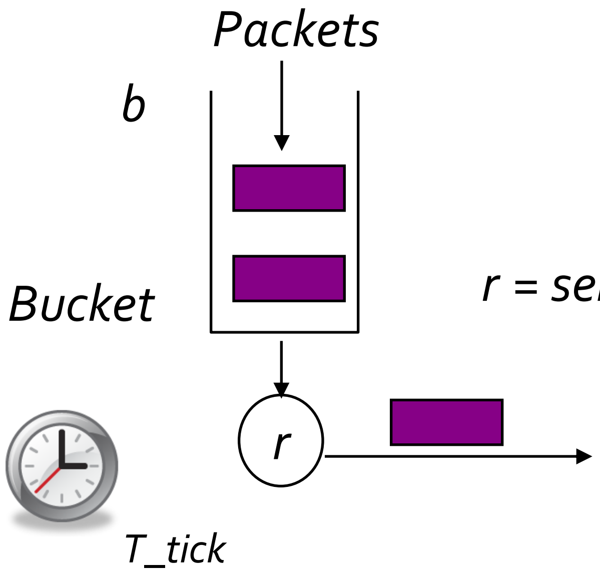
- Used in DiffServ Model

Traffic Contract

Traffic Contract Specification

QoS Contract: Models for Source Traffic

- Leaky Bucket (Turner 1986)
- Send data at regular intervals to the network
    - ❖ Credit to send n data bytes in a T_tick time interval
    - ❖ Data rate never larger than r (bytes / s) = n / T_tick
    - ❖ Buffering possible up to a limited amount of b bytes (bucket size)

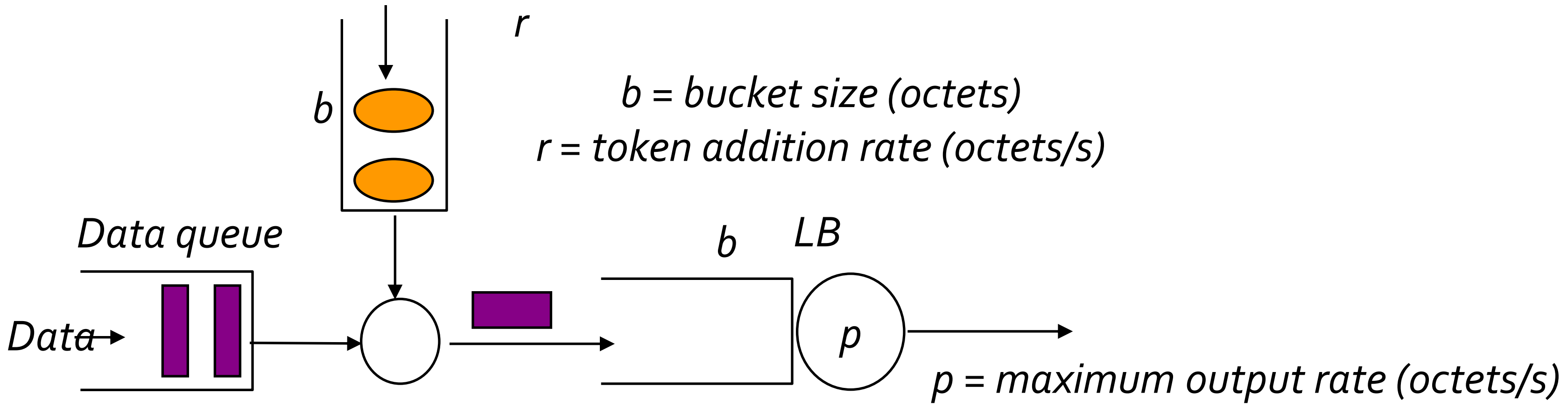


b = bucket size
r = sending rate to the network

r = n / T_tick
(n = max bytes_per_tick)

QoS Contract: Models for Source Traffic

- Token Bucket ("Credit" system: tokens)
  - ❖ Generate token periodically with rate "r"
  - ❖ Send a packet if enough tokens in the bucket
  - ❖ Sending below average rate allows for accumulation of tokens
  - ❖ Maximum number of tokens limited (b)
    - ✓ Allows for modeling bursty traffic
    - ✓ But limits the size of a burst
  - ❖ Peak rate p to limit the speed at which the bucket is emptied


r
b
b = bucket size (octets)
r = token addition rate (octets/s)
Data queue
Data
b
LB
p
p = maximum output rate (octets/s)

Leaky bucket and Token bucket

Input

Leaky Bucket

r=2MB/s

b=1MB

TB

r=2MB/s

b=250KB

TB

r=2MB/s

b=500KB

TB

r=2MB/s

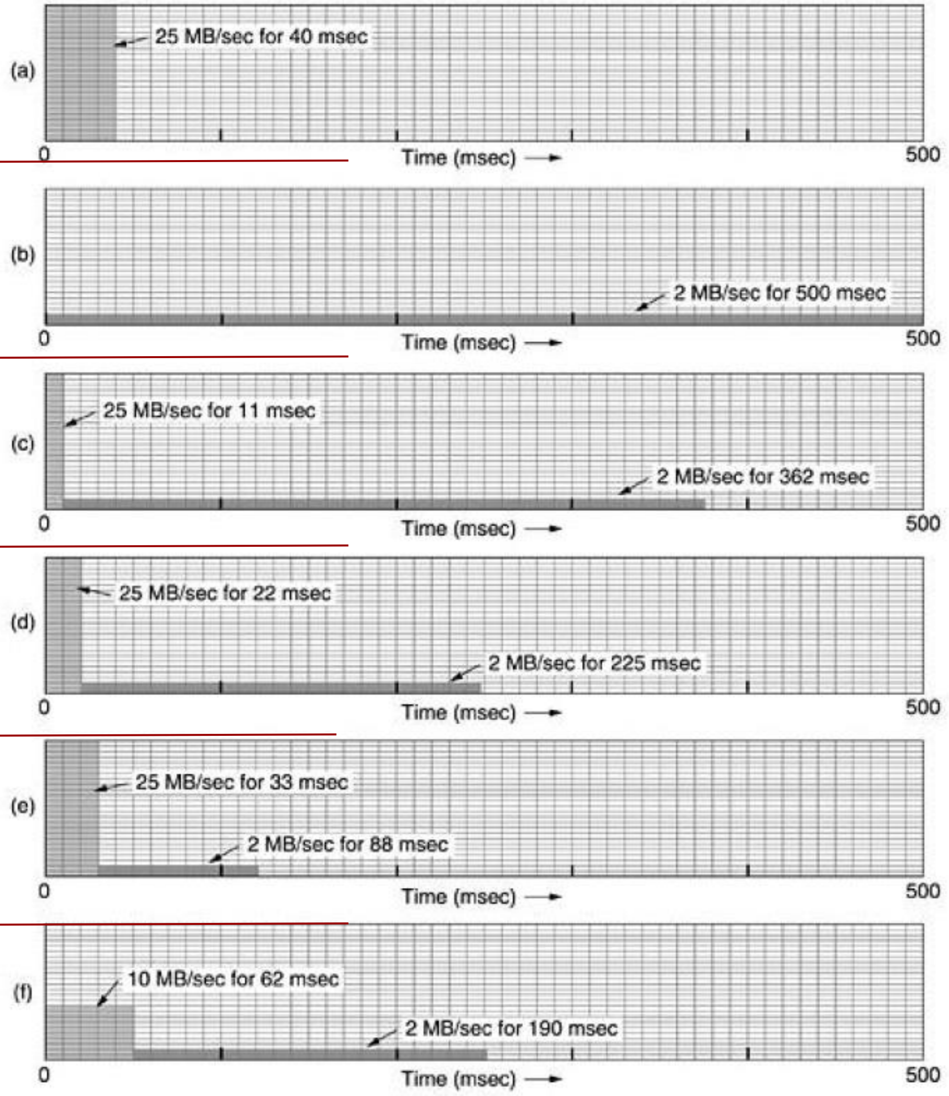
b=750KB

TB+LB

r=2 MB/s

b=500KB

p= 10MB/s



$$p \bullet T\_burst = r \bullet T\_burst + b$$

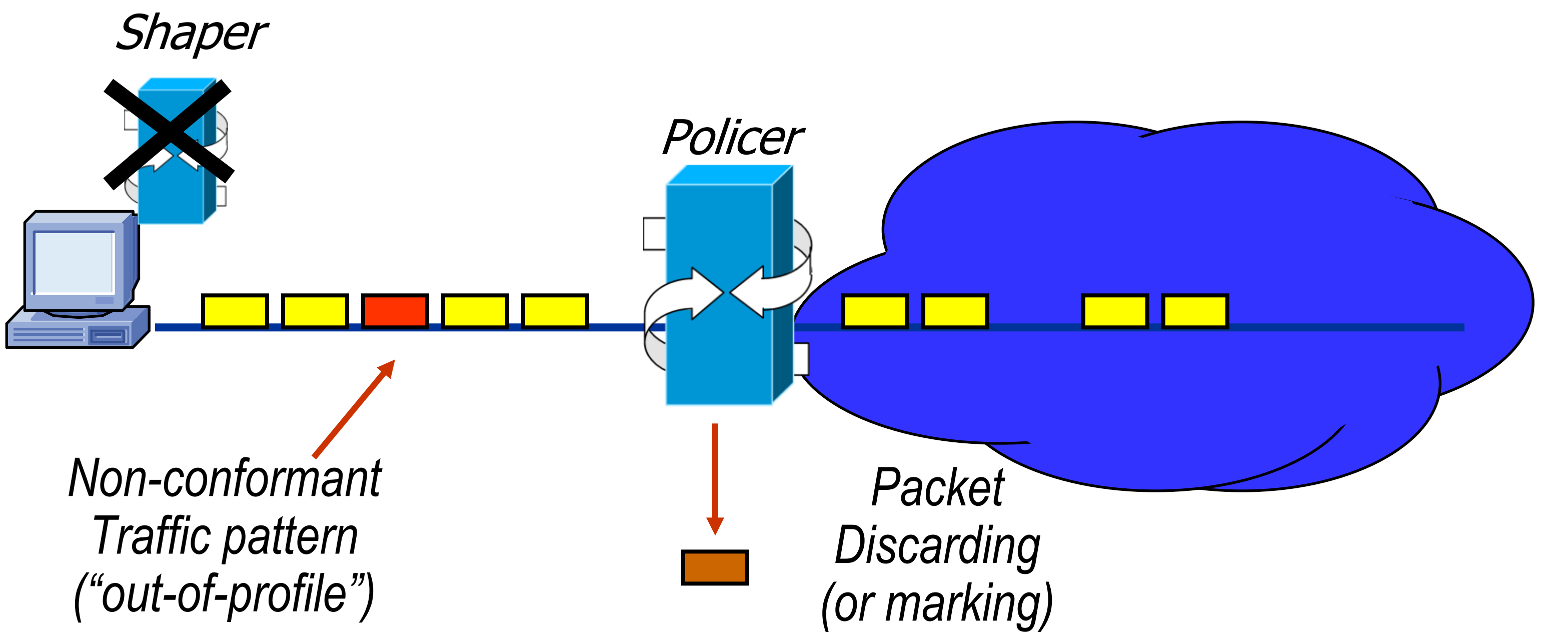
Source: Computer Networks. A. Tanembaum 3ª Ed

Traffic Contract

Traffic Conditioning

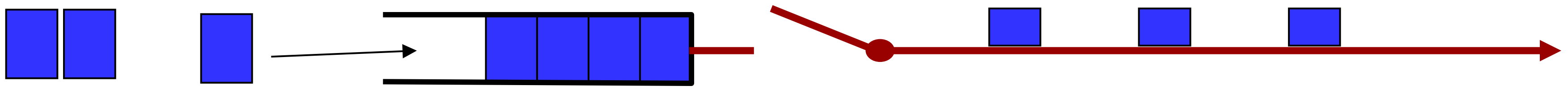
Traffic Conditioning

- ***Objective: enforce a Traffic Contract***
  - ***Policer :*** **operator side**
  - ***Shaper*** **: client side**

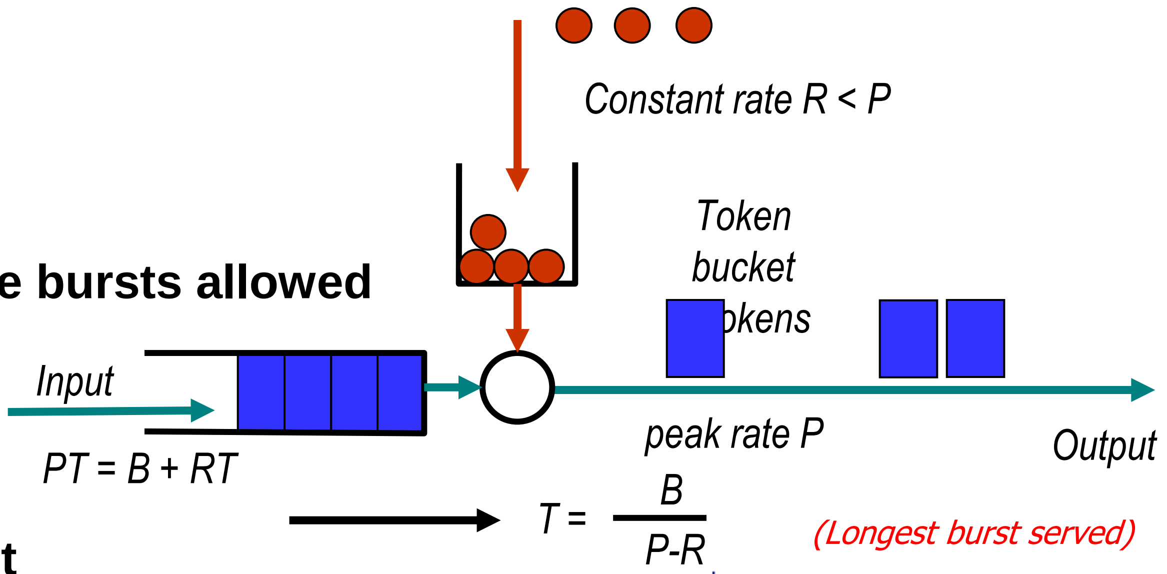

Shaper
Policer
Non-conformant Traffic pattern ("out-of-profile")
Packet Discarding (or marking)

Some commonly adopted policers

- **Leaky bucket**
  - **As shaper**
    - **Transforms an impulsive input into a regular flow**
    - **May add delay**
  - **As dropper/marker**
    - **Drop if bucket full**



- **Token bucket**
  - **As shaper:**
    - **Smooths traffic, some bursts allowed**
    - **May add delay**
  - **As dropper/marker**
    - **Drop/mark if no credit**



$$T = \frac{B}{P-R}$$

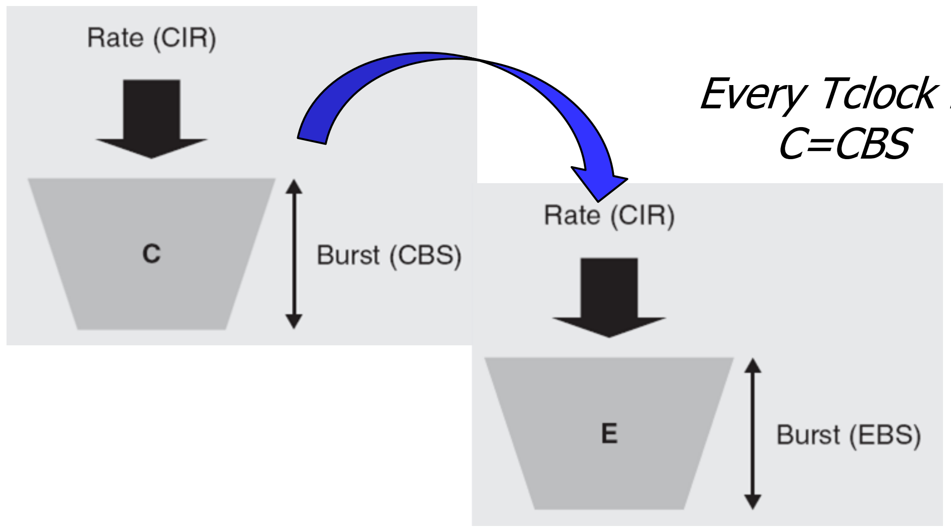
RFC 2697: Single Rate Three Color Marker

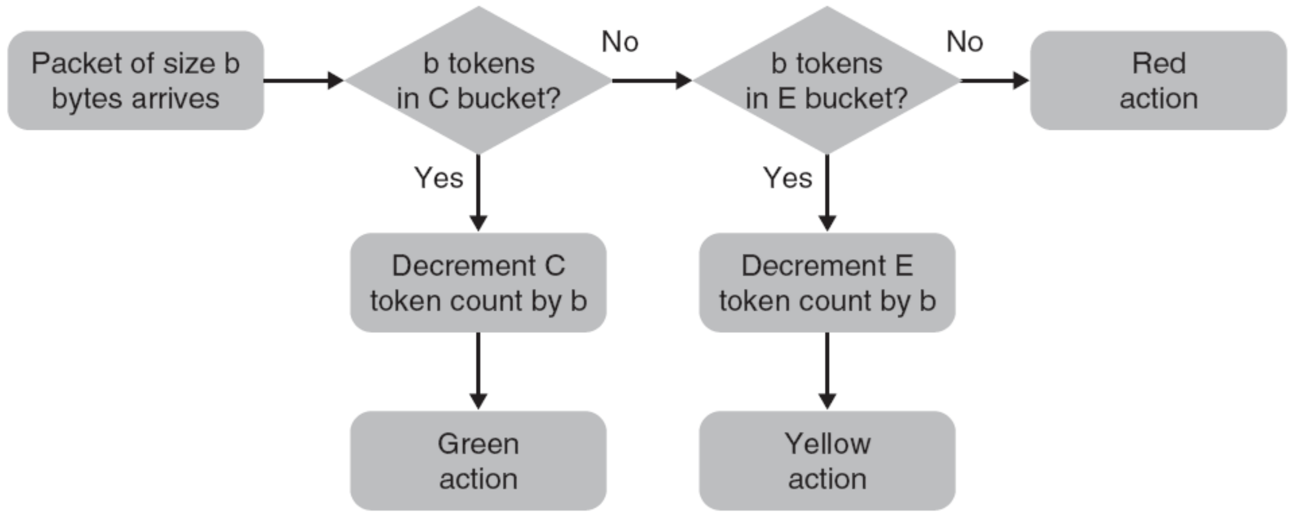
- SR-TCM uses two token buckets. The buckets are defined as C (committed or conform) and E (excess) with maximum burst sizes CBS and EBS respectively.
  - Both buckets are filled with tokens at the same rate CIR = N bits/Tclock
  - But E only increments every Tclock if C is full (C=CBS).

- The three possible states that are outcomes of the SR-TCM are described using a "traffic light" color scheme.

- If EBS=0, then effectively the output of the marker has only two states and packets will be marked either green or red, and the effective behavior of the SR-TCM is reduced to that of the simple one rate policer ("a single rate two color marker")

RFC 2697: Single Rate Three Color Marker



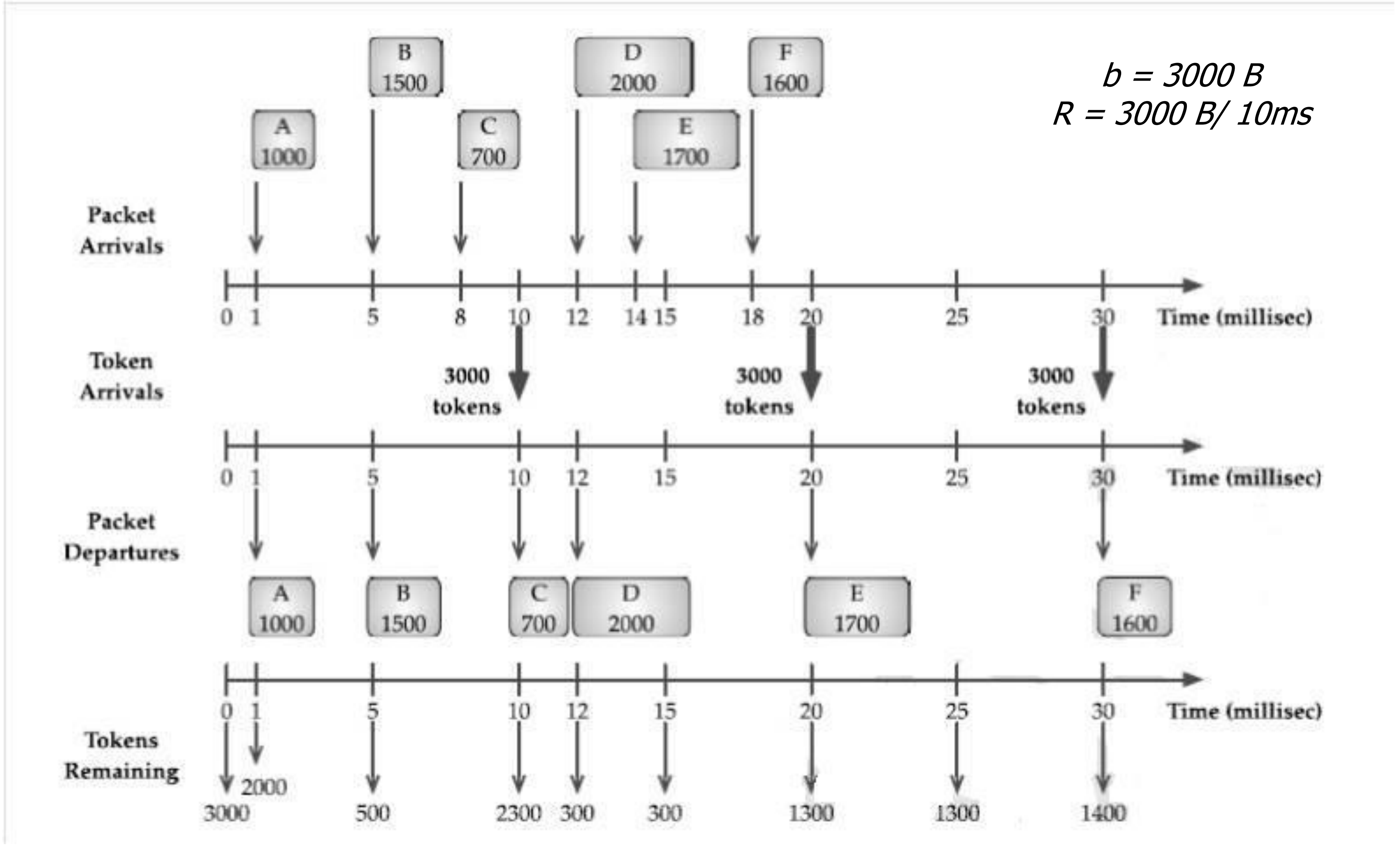


Source: EvansFilsfils

RFC 2697: Single Rate Three Color Marker

- Different actions, such as transmitting, dropping, or marking the packet can then be applied – possibly in combination – depending upon whether the packet has been designated as green, yellow or red by the SR-TCM
- Examples of use:
  - Enforcing a maximum rate for a voice class of traffic, with EBS=0 and applying a green action of transmit and red action of drop. Applied in this way the SR-TCM would enforce a maximum rate of CIR and a burst of CBS on the traffic stream, and any traffic in violation of this would be dropped, which is typical of the conditioning behavior used for the Differentiated Services Expedited Forwarding Per-Hop Behavior.
  - Marking a certain amount of a traffic class as in-contract, and everything above that as out-of-contract, with EBS=0 and applying a green action of transmit and red action of {transmit + mark out-of-contract}. Applied in this way the SR-TCM would enforce a maximum rate of CIR and a burst of CBS on the traffic stream; any traffic in violation of this would not be dropped but would be marked out-of-contract, which is typical of the conditioning behavior used for the Differentiated Services Assured Forwarding Per-Hop Behavior

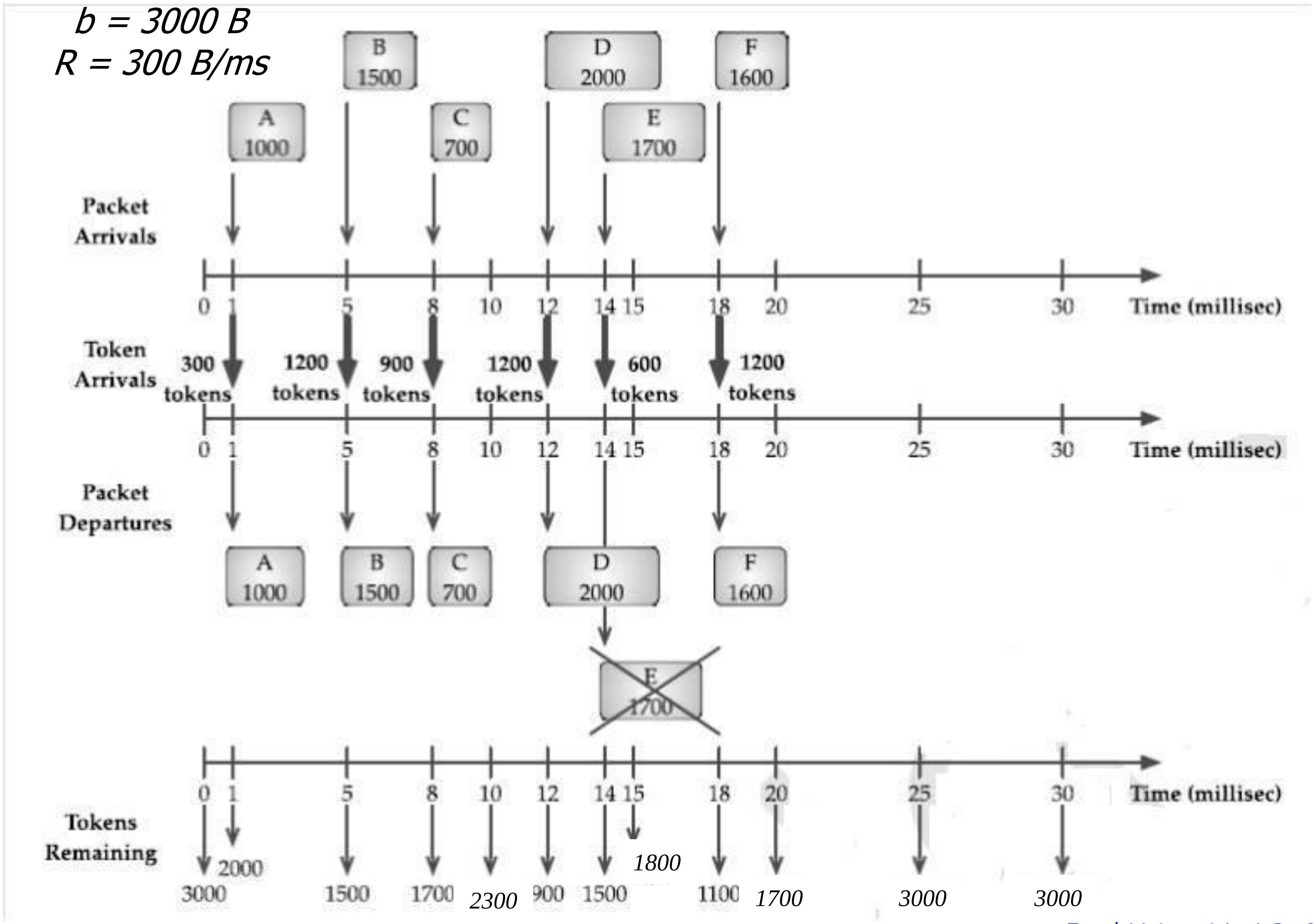
Example: Shaping traffic with token bucket



b = 3000 B
R = 3000 B/ 10ms

Source: MedhiRamasamy

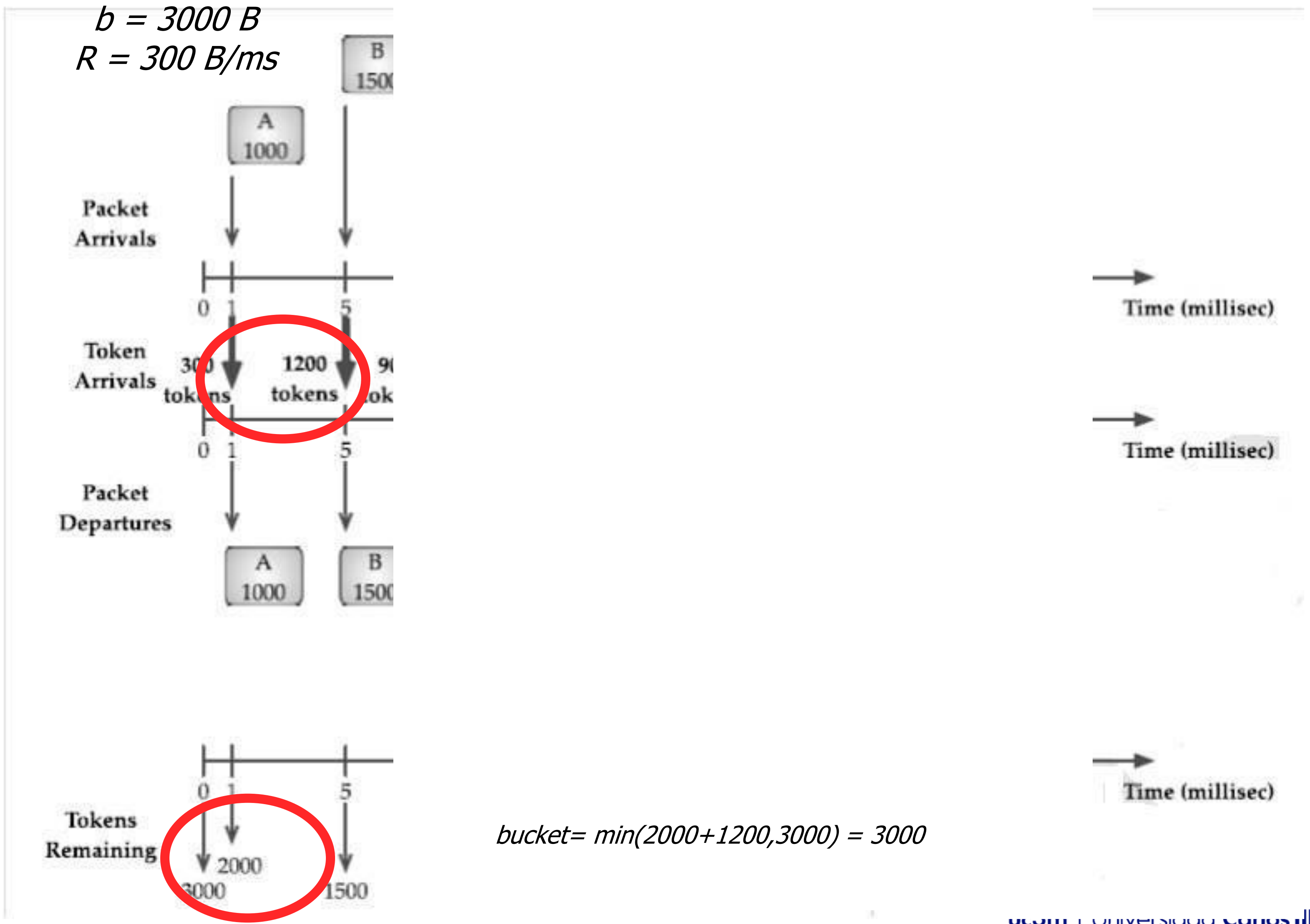
Example: Policing traffic with a token bucket



Source: MedhiRamasamy

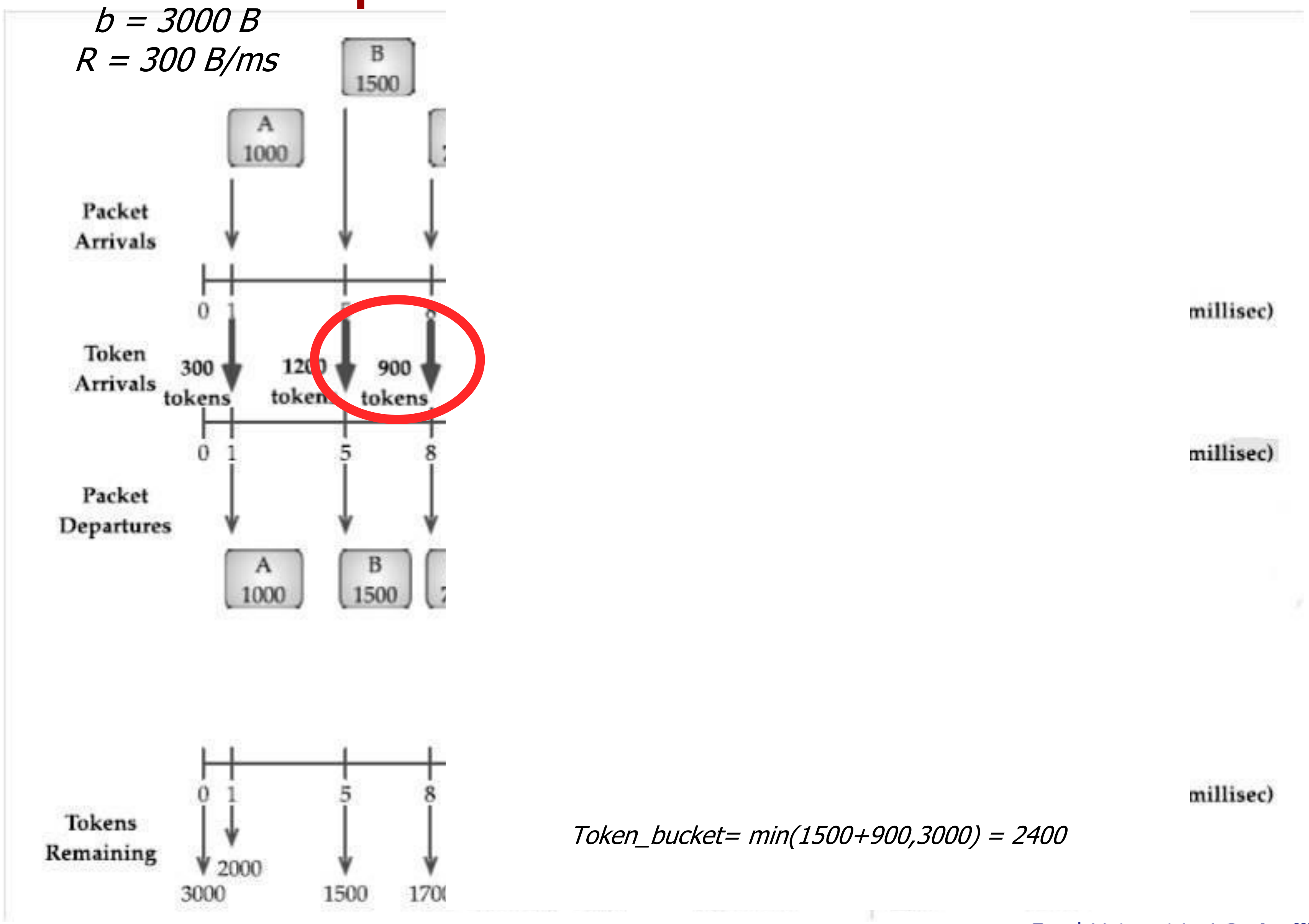
uc3m Universidad Carlos III de Madrid
Departamento de Ingeniería Telemática

Example: Policing traffic with a token bucket



bucket= min(2000+1200,3000) = 3000

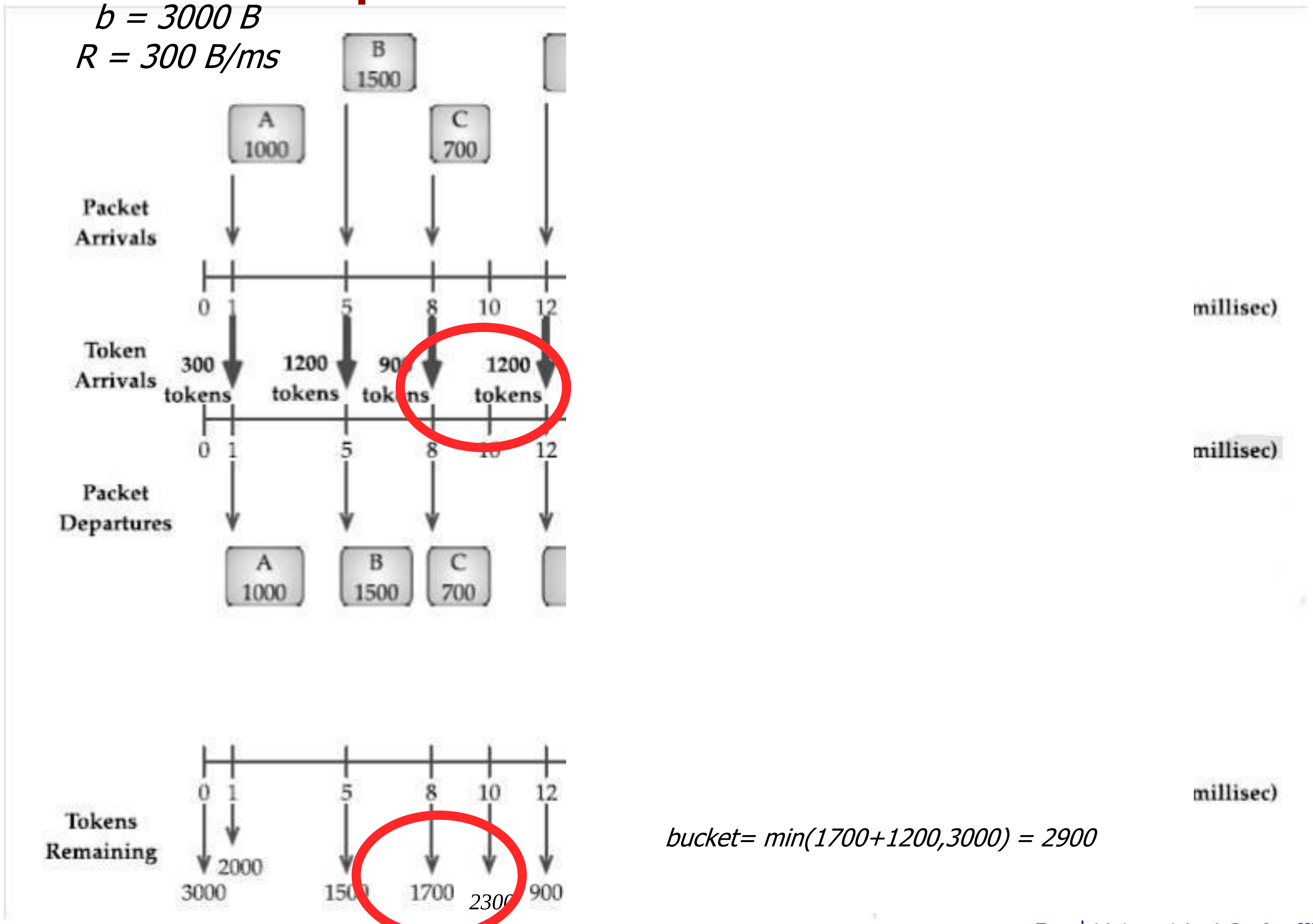
Example: Policing traffic with a token bucket



Token_bucket= min(1500+900,3000) = 2400

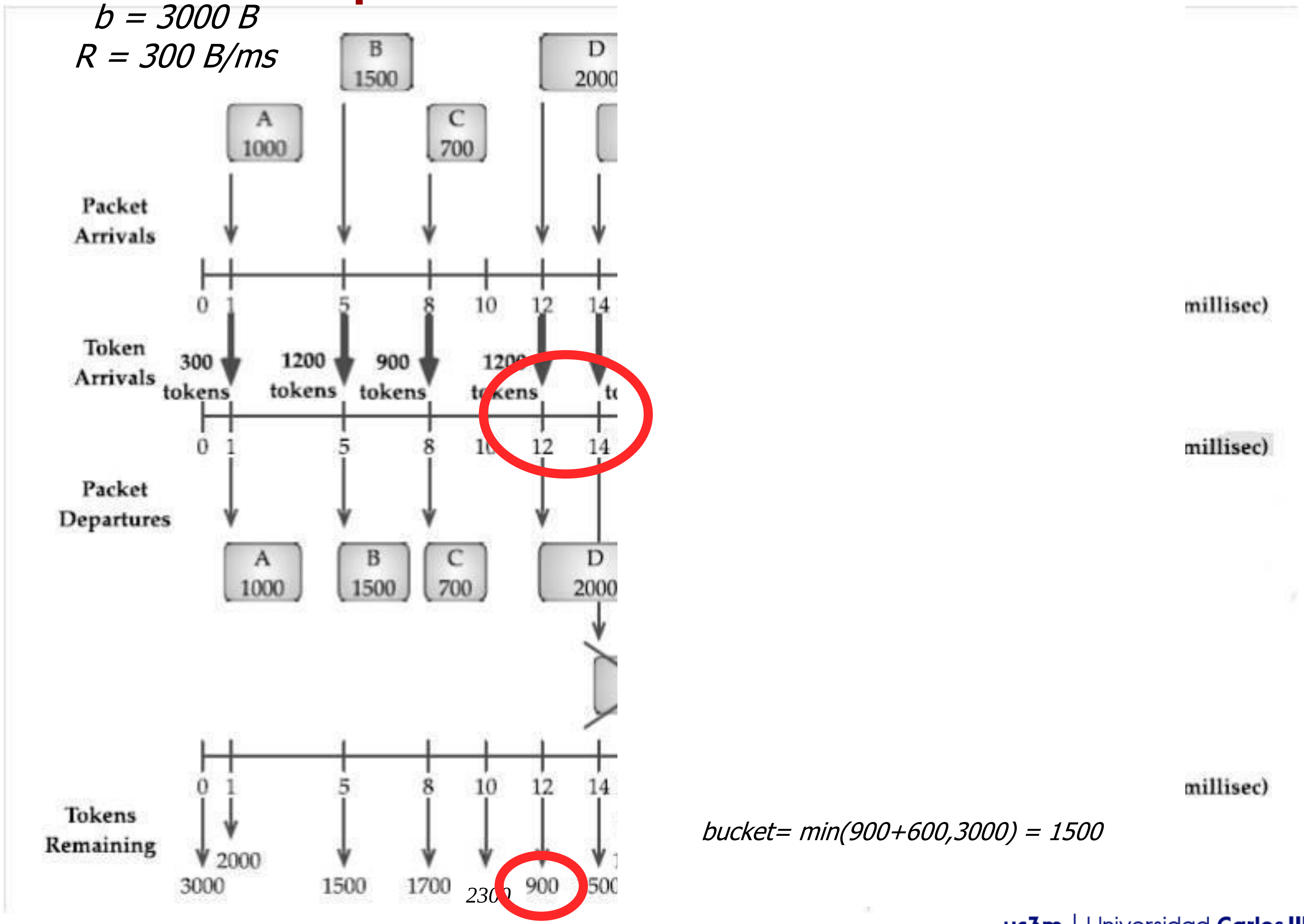
Example: Policing traffic with a token bucket

b = 3000 B
R = 300 B/ms



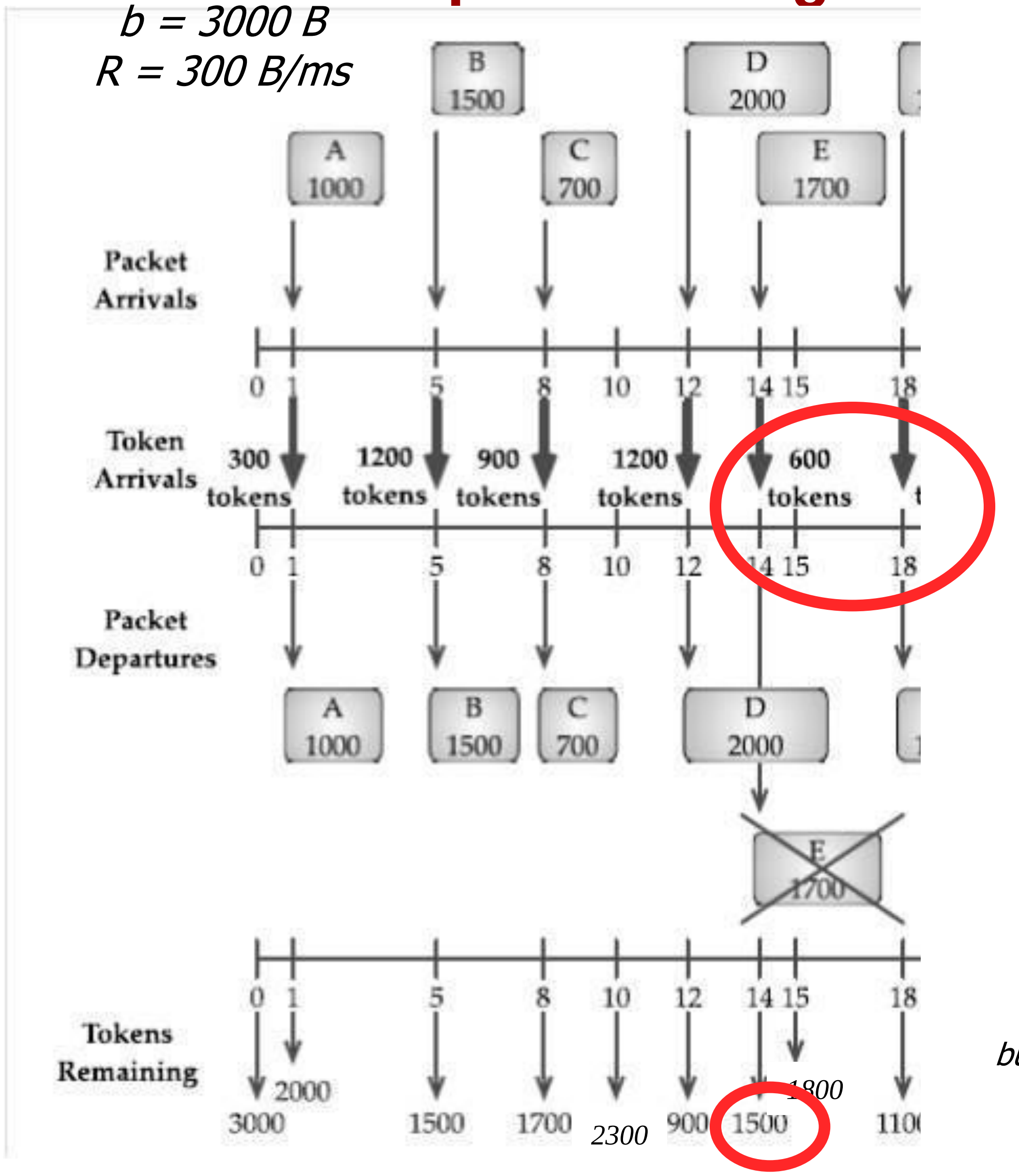
bucket= min(1700+1200,3000) = 2900

Example: Policing traffic with a token bucket



bucket= min(900+600,3000) = 1500

Example: Policing traffic with a token bucket



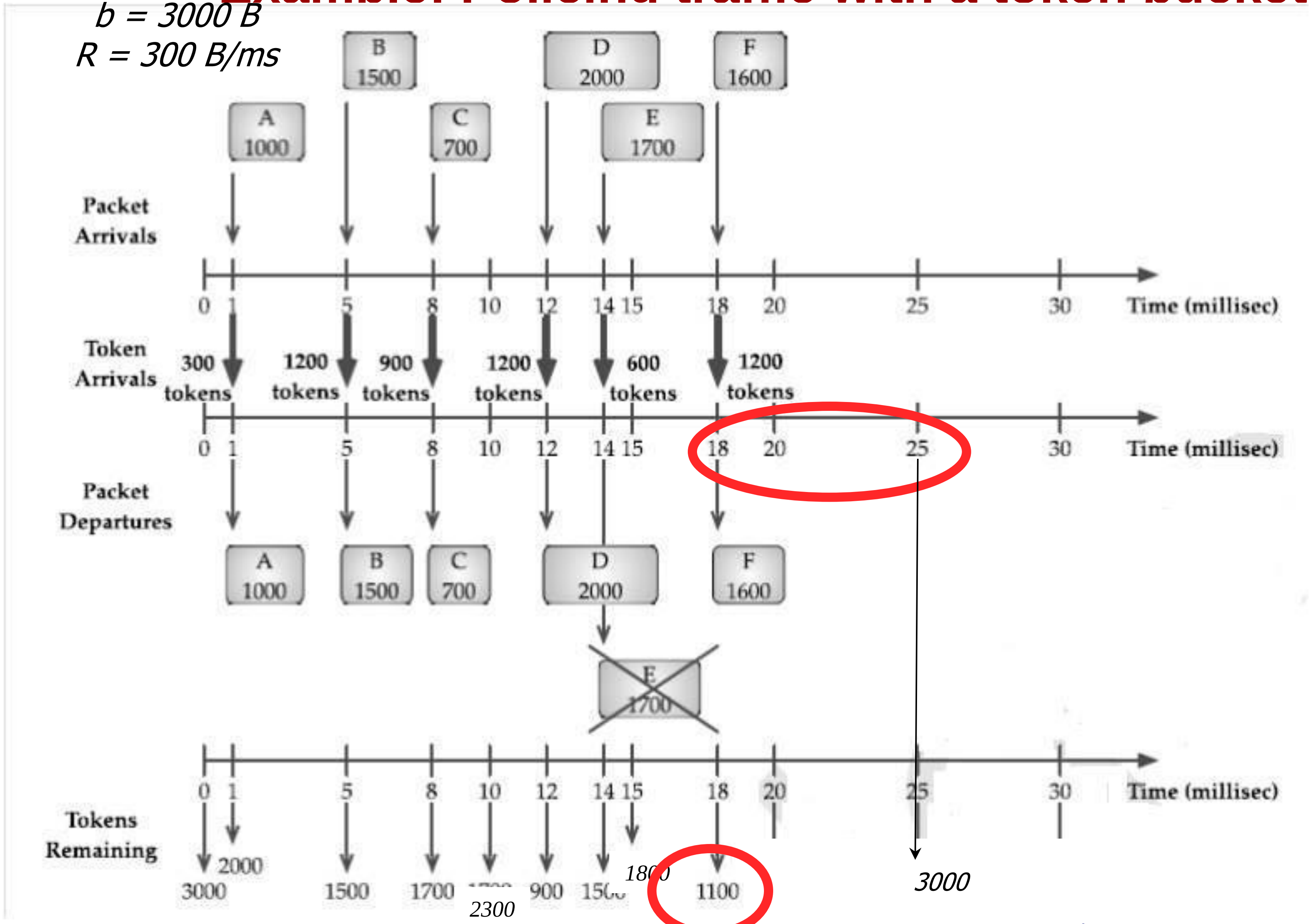
4x300=1200

bucket= min(1500+1200,3000) = 2700

Example: Policing traffic with a token bucket

b = 3000 B

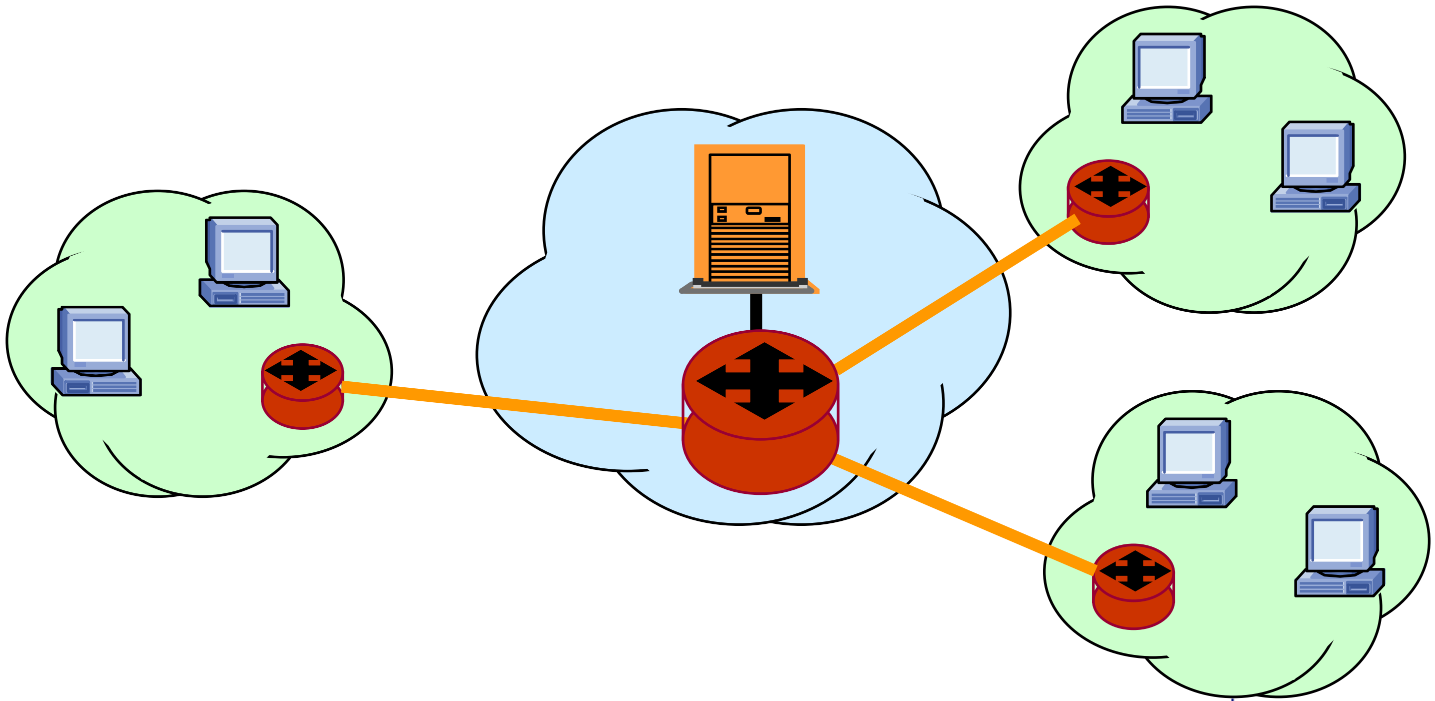
R = 300 B/ms



Signalling and CAC

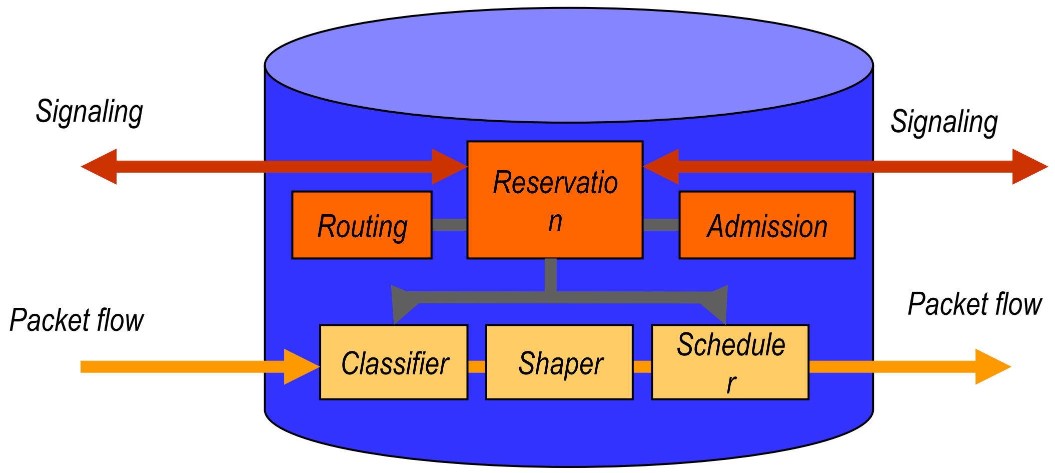
Centralized Admission Control

- Central server (bandwidth broker, PCE, SDN controller)
- ⍰Not very robust
- ⍰Contention solved at signalling phase



Distributed admission control

- Each node decides locally whether to accept a new flow and reserve resources
- Requires reservation protocols (e.g., RSVP)
- More robust, scalable
- Contention problems at signalling phase



Questions

1. Indicate which are the 4 main sources of delay in a packet network. Draw a diagram to locate where each of them takes place.

2. Indicate which Quality of Service mechanisms you know and describe them briefly.

3. Explain the concept of Goodput. Describe in detail the difference between "Non-work-conserving" and "Work-conserving" scheduling algorithms and represent them with an image.

1. What is the problem known as TCP global synchronization? What is the Random Early Detection (RED) queue management technique and how does it contribute to solving it?

References

- Engineering Internet QoS. Sanjay Jha, Mahbub Hassan. Artech House 2002.
- A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Single-Node Case. Abhay K. Parekh, Robert G. Gallager. IEEE ACM Transactions on Networking, Vol. 1, Nº. 3, June 1993.
- A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Multiple Node Case. Abhay K. Parekh, Robert G. Gallager. IEEE ACM Transactions on Networking, Vol. 2, Nº. 2, April 1994
- Computer Networking. A Top-Down Approach Featuring the Internet. James F. Kurose and Keith W. Ross. ISBN 0-201-61274-7, Addison-Wesley
- Internetworking Multimedia. Jon Crowcroft, Mark Handley, Ian Wakeman. UCL Press.
  - https://www.cl.cam.ac.uk/~jac22/out/mm.pdf
- Communication Networking: An Analytical Approach. Anurag Kumar, D. Manjunath, Joy Kuri. Academic Press, 21 May 2004.