Advanced Computer Networks

Congestion control and scheduling

Prof. Andrzej Duda

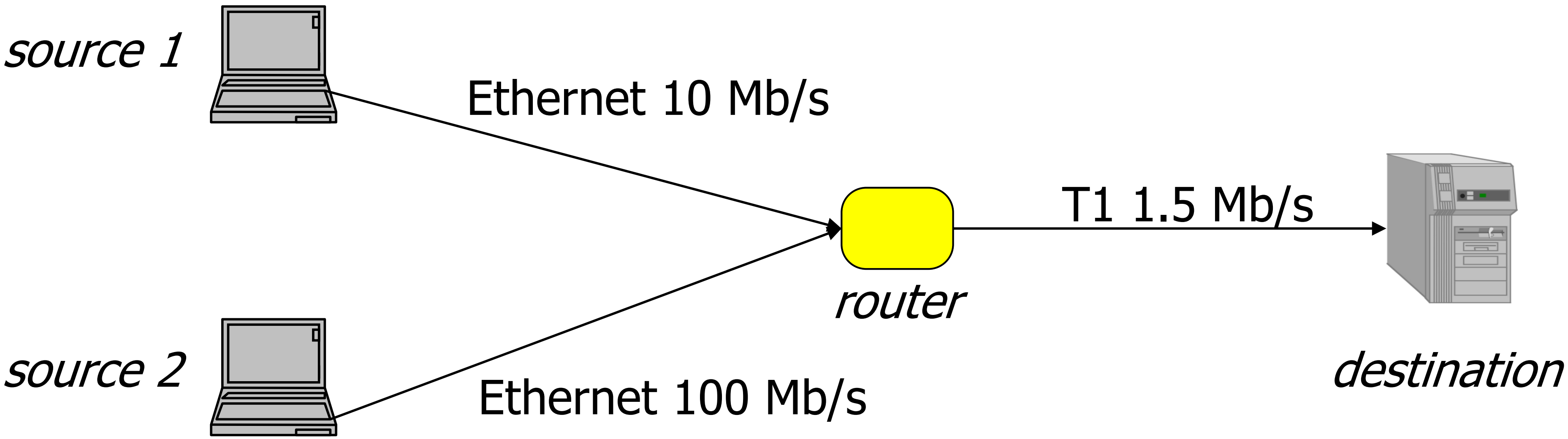
duda@imag.fr

http://duda.imag.fr

Contents

- Objectives of Congestion Control
  - effciency
  - fairness
- Max-min fairness
- Proportional fairness
- Allocation – scheduling disciplines
  - FIFO
  - Priority
  - CBQ
  - Fair queueing
  - DRR

Congestion control

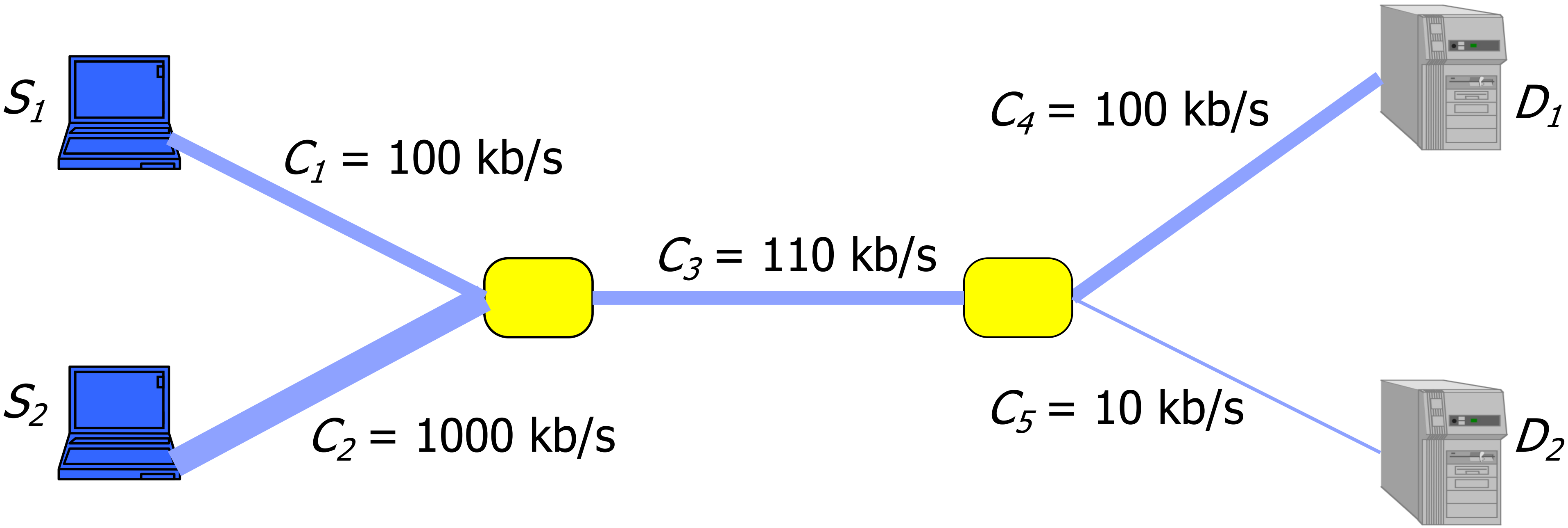


- How to allocate network resources?
  - link capacity
  - buffers at routers or switches
- What to do when the traffic exceeds link capacity?
  - congestion control

Performance criteria

- Efficiency
  - best use of allocated resources
  - max throughput - 100 % utilization
  - min delay - 0 % utilization
- Fairness (équité)
  - fair share to each user
  - different definitions of fairness
    - equal share
    - max-min fairness
    - proportional fairness

Congestion Control - example



- Sources send as much as possible
- Allocation of throughput
  - if the offered traffic exceeds the capacity of a link, all sources see their traffic reduced in proportion of their offered traffic
  - approximately true if FIFO in routers

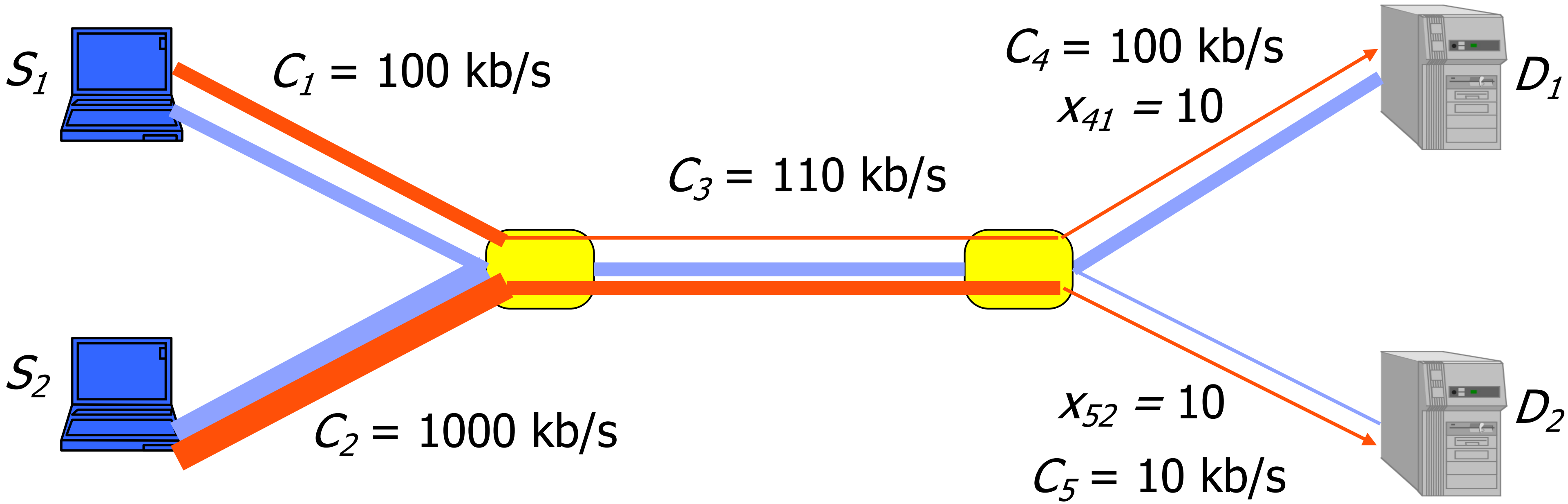
Throughput allocation

- Throughput x_{ls}: source s on link l
- Traffic λ_s: generated by source s
- Allocation

Our example:

Allocation	Our example
$x_{11} = \min\ (\lambda_1, C_1)$	$x_{11} = 100$
$x_{22} = \min\ (\lambda_2, C_2)$	$x_{22} = 1000$
$x_{3i} = \min\ (x_{ii}, C_3\, x_{ii}/(x_{11} + x_{22}))$	$x_{31} = 110 \times 100/1100 = 10$
	$x_{32} = 110 \times 1000/1100 = 100$
$x_{41} = \min\ (x_{31}, C_4)$	$x_{41} = 10$
$x_{52} = \min\ (x_{32}, C_5)$	$x_{52} = 10$
throughput $\vartheta = x_{41} + x_{52}$	throughput $\vartheta = 20$ kb/s

Congestion Control - example

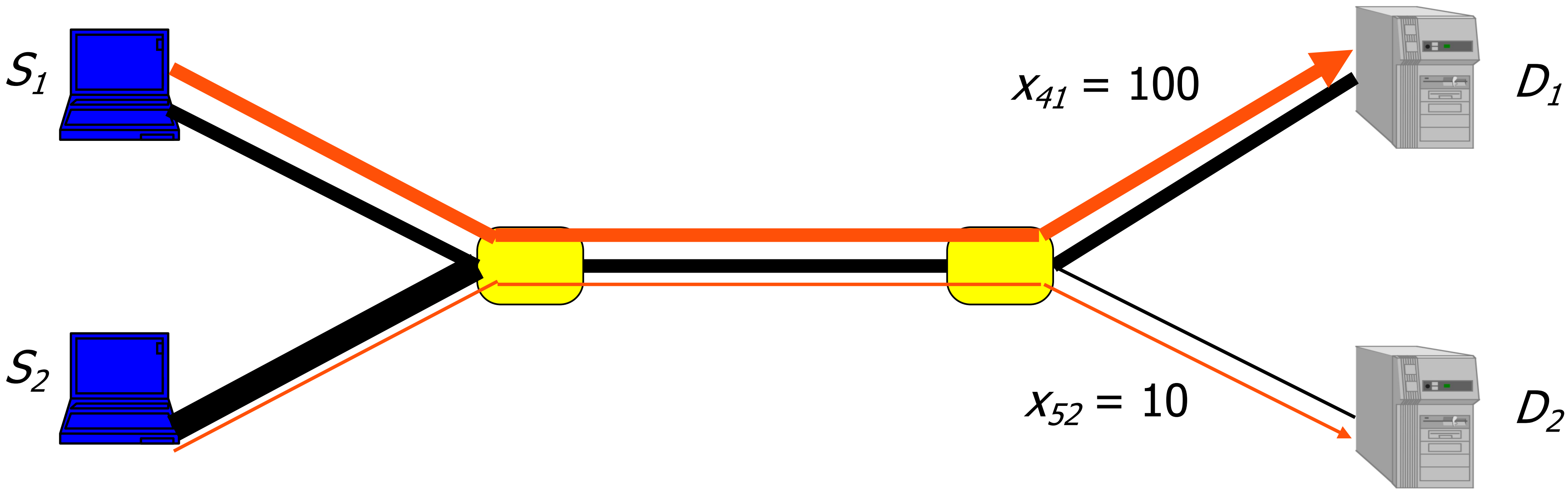


- S_1 sends 10 kb/s because it is competing with S_2 on link 3
- S_2 is limited on link 5 anyway

Congestion Control - exemple

- How to increase throughput?
  - if S_2 is aware of the global situation and if it would cooperate
  - S_2 reduces x_{22} to 10 kb/s, because anyway, it cannot send more then 10 kb/s on link 5
  - x_{31} = 100 kb/s and x_{41} = 100 kb/s without any penalty for S_2
  - throughput is now ϑ = 110 kb/s

Congestion Control - exemple



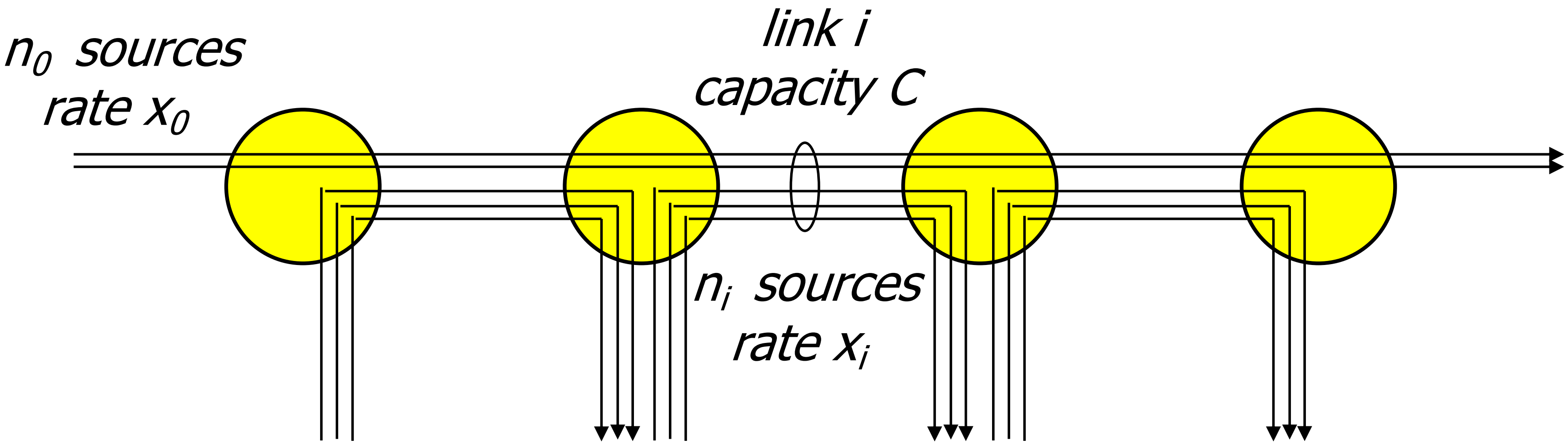
- Optimal use of resources

Efficiency criterion

- In a packet network, sources should limit their sending rate by taking into consideration the state of the network. Ignoring this may put the network into congestion collapse
  - network resources are not used efficiently
  - performance indices perceived by sources are not satisfactory
- One objective of congestion control is to avoid such inefficiencies

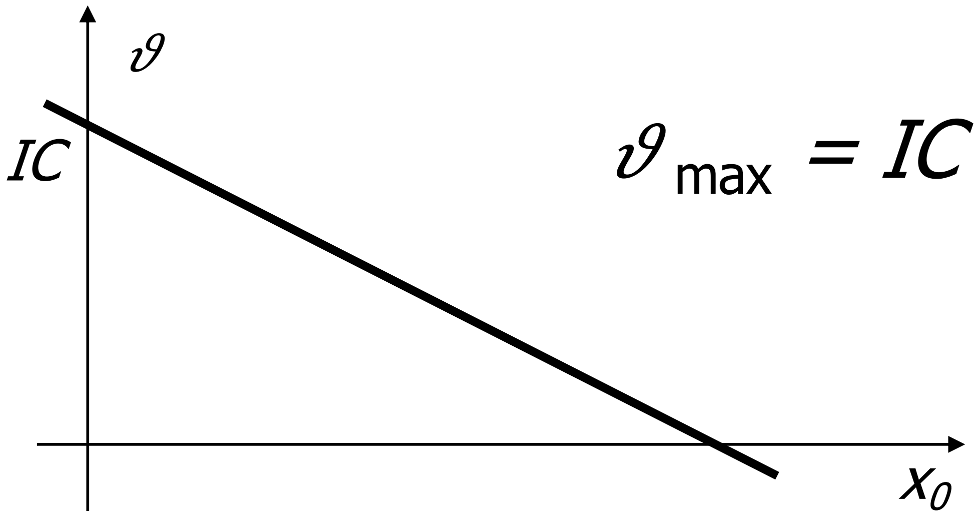
Efficiency versus Fairness

- Parking lot scenario
  - link capacity: C
  - n_i sources, rate x_i, $i = 1, \ldots, I$
  - traffic on link i : $n_0\, x_0 + n_i\, x_i$



Maximal throughput

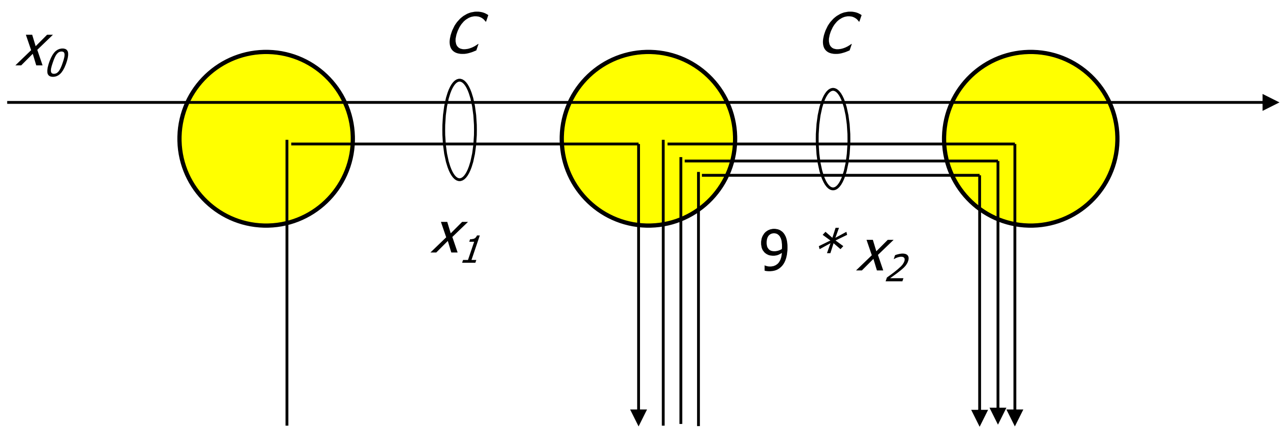
- For given n_0 and x_0, maximizing throughput requires that
  - $n_i\, x_i = C - n_0\, x_0$
- Total throughput, measured at the network output
  - $\vartheta = n_0\, x_0 + \sum n_i\, x_i = n_0\, x_0 + \sum (C - n_0\, x_0) =$
    $= n_0\, x_0 + I(C - n_0\, x_0) = IC - (I - 1)\, n_0\, x_0$



$\vartheta_{\max} = IC$ for $x_0 = 0$!

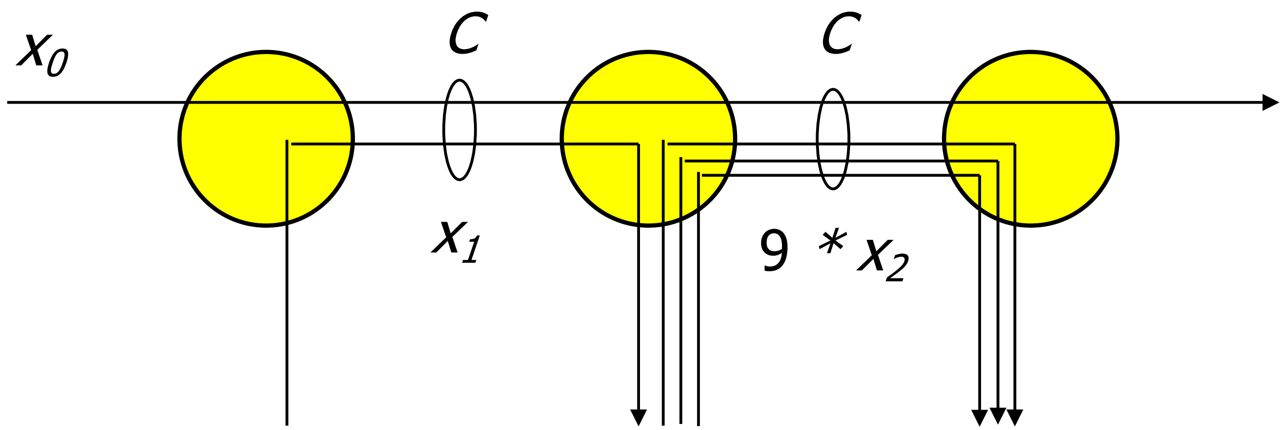
Maximum throughput

- Example
  - $I = 2$, $n_0 = n_1 = 1$, $n_2 = 9$
- The value of x_0 for maximum throughput?
  - 1: C?
  - 2: $2C$?
  - 3: $0.1\ C$?
  - 4: None of the above?



Maximum throughput

- Find $x_0\ x_1\ x_2$ such that:
  - $x_0 + x_1 \leq C$ -> $x_0 + x_1 = C$
  - $x_0 + 9x_2 \leq C$
  - Maximize $x_0 + x_1 + 9x_2$ -> $x_0 + x_1 + 9x_2 = 2C$
  - $9x_2 = C$
  - $x_0 = 0$, $x_1 = C$, $x_2 = C/9$



Pareto Efficiency (Optimality)

- A feasible **allocation of rates x_i** is called **Pareto-efficient** iff increasing one source must be done at the expense of decreasing some other source

- For a feasible allocation x_i', for every i :

- if $x_i' > x_i$ then $x_j' < x_j$

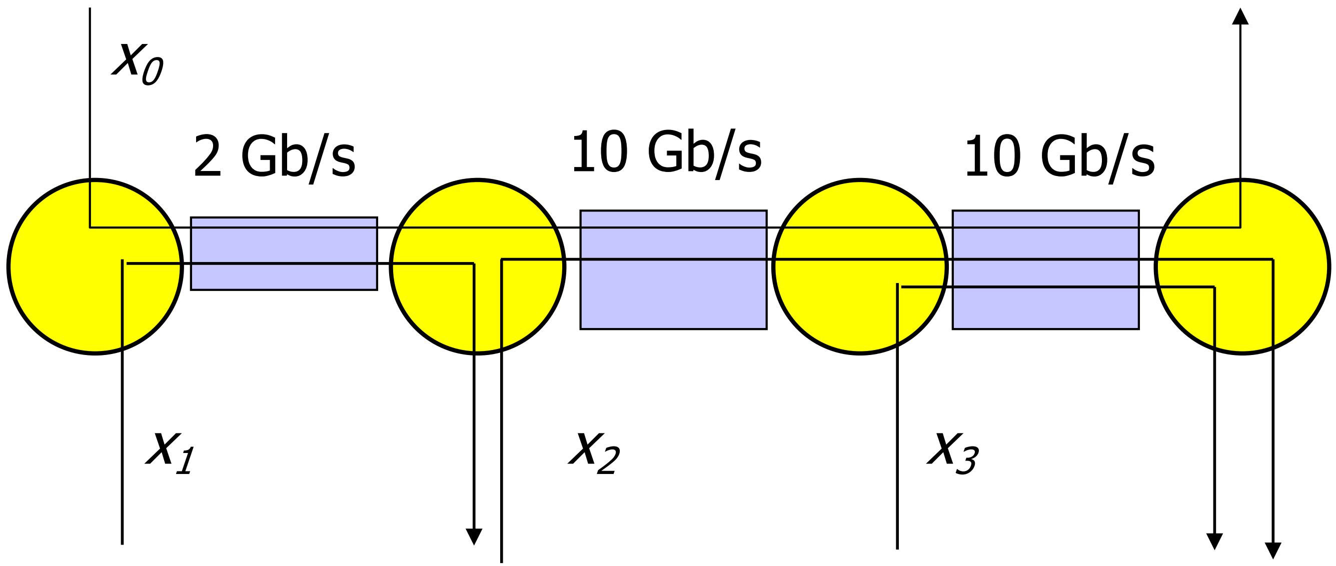
- **Every source has a bottleneck link** (i.e., for every source *i* there exists a link, used by *i*, which is saturated)

Pareto Efficiency (Optimality)

- State of resource allocation in which there is no alternative state that would make some people better off without making anyone worse off
- In the case of multipe flows, it means that giving higher rate to a flow cannot reduce the throughput of other flows

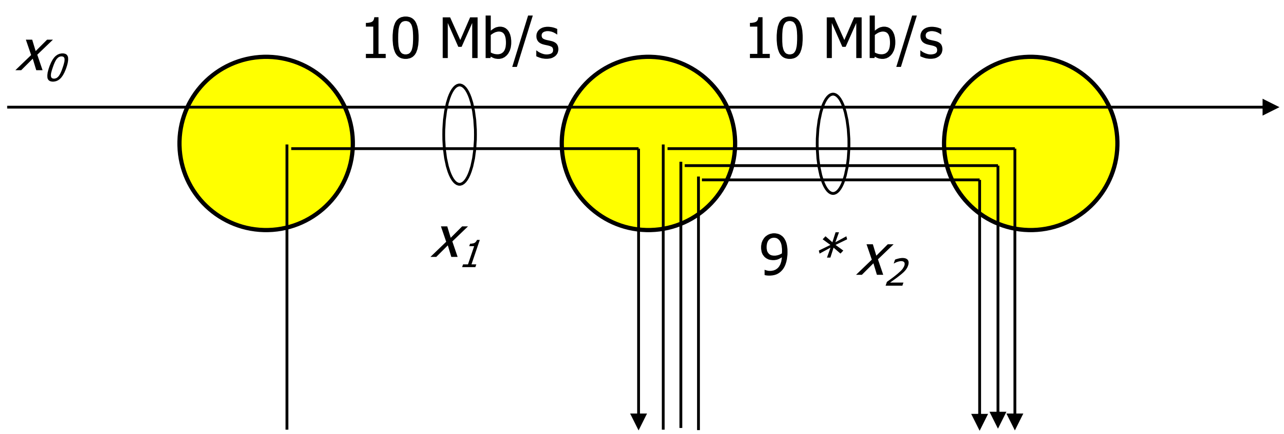
Allocation Pareto-Efficient?

- $x_0 = 1,\ x_1 = 1,\ x_2 = 2\ x_3\ = 7$?
- $x_0 = 1,\ x_1 = 1,\ x_2 = 4.5\ x_3\ = 4.5$?
- Both?
- None?
- I don't know?



Pareto-Efficient?

- $x_0 = 0$, $x_1 = 10$, $x_2 = 10/9$?
- $x_0 = 0.55$, $x_1 = 9.45$, $x_2 = 1.05$?
- $x_0 = 1$, $x_1 = 9$, $x_2 = 1$?



Pareto Efficiency

- The Pareto efficient allocations are the ones that **use the resources maximally**
- Maximal efficiency means Pareto optimality.
- Maximizing total throughput is Pareto optimal, but it means shutting down some flows (x_0) this is at the expense of fairness.

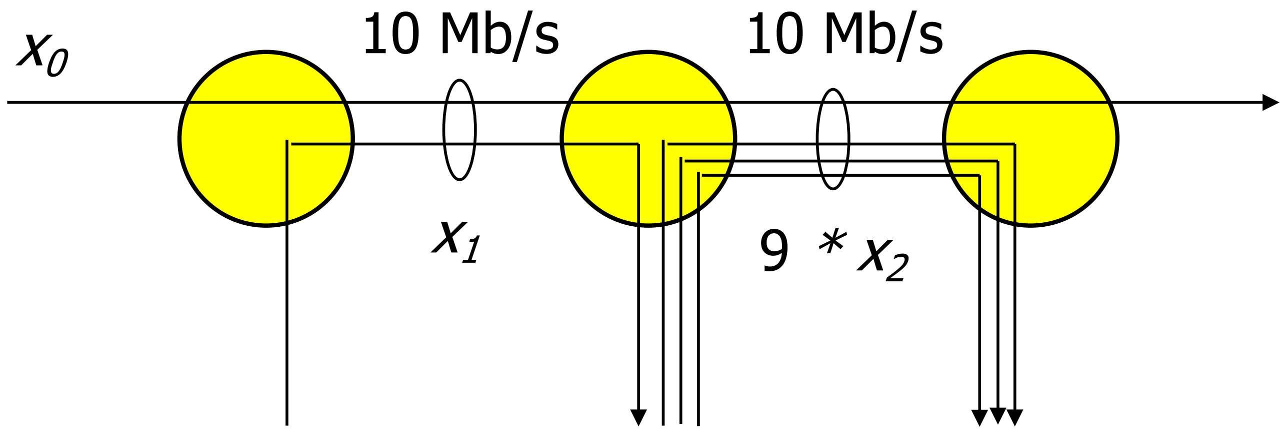
- Are there Pareto-efficient allocations that are fair? What is fairness?
- Egalitarianism (give each flow the same part) is not Pareto-efficient

Fairness

- Maximizing network throughput as a primary objective may lead to large unfairness
  - some sources may get a zero throughput
- Fairness criterion – **equal share to all**
  - let allocate the same share to all sources (egalitarianism), e.g., for $n_i = 1$
    - $x_i = C/2$
    - $\vartheta_{fair} = (I+1)C/2$
  - roughly half of maximal throughput

Fair (equal share)?

- $x_0 = x_1 = x_2 = 0.5$?
- $x_0 = x_1 = x_2 = 1$?
- $x_0 = x_1 = x_2 = 10/9$?

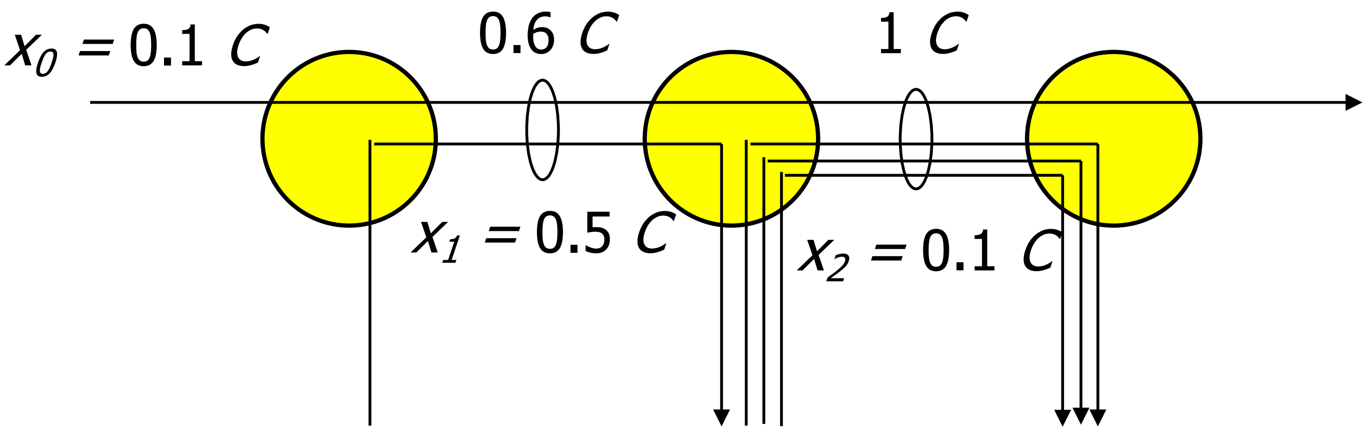


Equal share fairness

- Consider the parking lot scenario for any values of n_i
  - equal share on link i
    - $x_i = C / (n_0 + n_i)$, $i = 1, \ldots, I$
  - let decrease x_0 to increase ϑ (we have seen that this maximizes throughput)
    - $x_0 = \min C / (n_0 + n_i)$,
  - example
    - $I = 2$, $n_0 = n_1 = 1$, $n_2 = 9$
    - link 2: $x_2 = C / (1 + 9) = 0.1\ C$
    - link 1: $x_1 = C / (1 + 1) = 0.5\ C$
    - $x_0 = \min (0.5\ C, 0.1\ C) = 0.1\ C$
- Allocating equal shares is not a good solution
  - some flows can get more

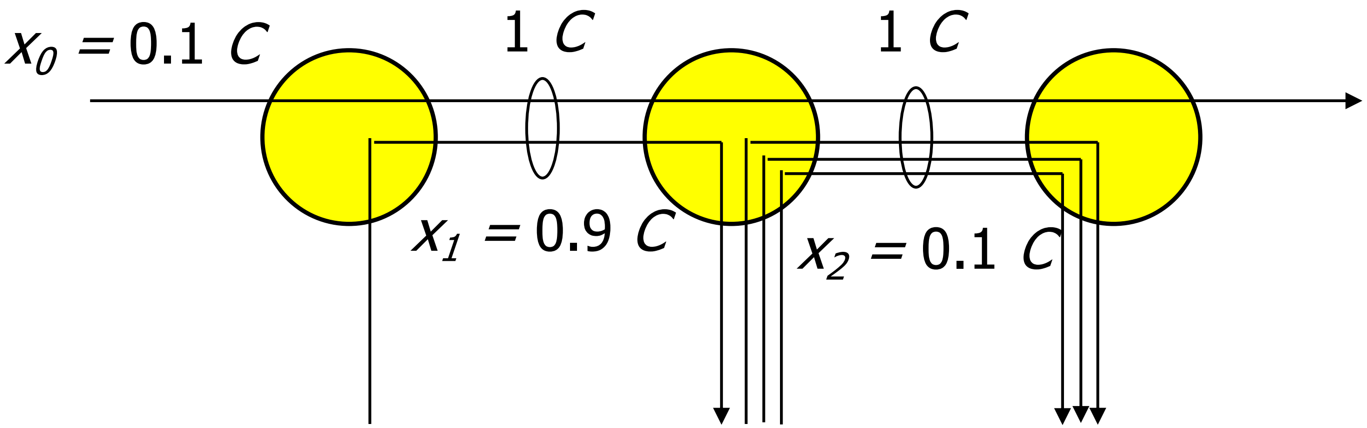
Example

- Problem
  - link 1: 0.6 C
    - underutilized
  - link 2: 1 C



Max-Min Fairness

- We can increase x_1 without penalty for other flows
  - $x_0 = 0.1\ C$, $x_1 = 0.9\ C$, $x_2 = 0.1\ C$
- This allocation is Pareto-efficient!



Max-Min Fairness

- Allocating resources in an equal proportion is not a good solution since some sources can get more that others without decreasing others' shares
- Max-Min fair allocation
  - Min: because of the fairness on bottleneck links
  - Max: because we can increase throughput whenever possible
- For every source *i* , increasing its rate must force the rate of some other (not richer) source *j* to decrease
- An allocation is max-min fair if *any rate increase contradicts fairness*
- Max-min fair allocation is Pareto-efficient (converse is not true)

Progressive filling

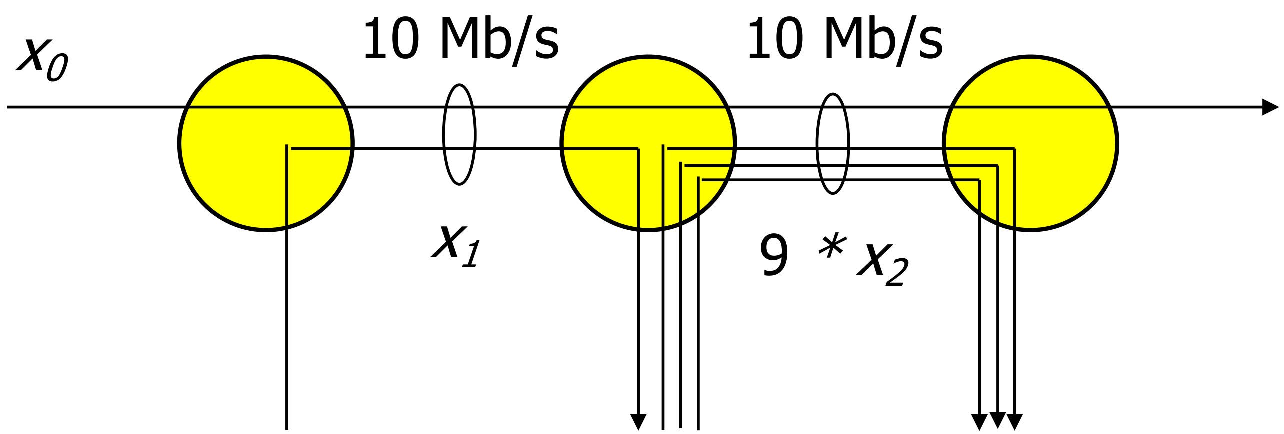
- Bottleneck link *l* for source *s*
  - link *l* is saturated: $\sum x_i = C$
  - source *s* on link *l* has the maximum rate among all sources using that link
- Progressive filling allocation
  - $x_i = 0$
  - increase x_i equally until $\sum x_i = C$
  - rates for the sources that use this link are not increased any more
    - all the sources that do not increase have a bottleneck link (Min)
  - continue increasing the rates for other sources (Max)

Example

- Parking lot scenario
  - $x_i = 0$
  - $x_i = d$ until $n_0\, x_0 + n_i\, x_i = C$
  - bottleneck link for $d_1 = \min\,(C\,/\,(n_0 + n_i))$, source 0 or i
    - $x_0 = \min\,(C\,/\,(n_0 + n_i)\,)$
  - increase other sources
    - $x_i = (C - n_0\, x_0\,)\,/\,n_i$
- In our example
  - $x_0 = 0.1\ C$, $x_2 = 0.1\ C$
  - $x_1 = 0.9\ C$

Max-Min Fair?

- $x_0 = 0$ $x_1 = 10$, $x_2 = 10/9$?
- $x_0 = 1$ $x_1 = 9$ $x_2 = 1$?



Exercise

- $C = 10$
- We have four flows with demands of 2, 2.6, 4, 5
- What is the Max-min allocation to flows?

Exercise

- Two sources 1 and 2 share a capacity link C. The flow x_i of source i is limited by
  - $x_i \leq r_{i,}$ i = 1, 2
- Let C = 9 Mb/s, r_1 = 3 Mb/s, r_2 = 8 Mb/s
- Find x_i assuming the allocation is max-min

Proportional Fairness

- Equal share fairness and Max-min fairness
  - per link only
  - do not take into account the number of links used by a flow
  - flows x_0 benefit from more network resources than flows x_i
- Another fairness
  - give higher throughput to flows that use less resources
  - give smaller throughput to flows that use more resources
- Proportional fairness

Proportional Fairness

- An allocation of rates x_s is *proportionally fair* if and only if, for any other feasible allocation y_s we have (S sources)

$$\sum_{s=1}^{S} \frac{y_s - x_s}{x_s} \leq 0$$

- Any change in the allocation must have a negative average change
- Parking lot example with $n_s = 1$
  - max-min fair allocation $x_s = C/2$ for all s
  - let decrease x_0 by δ: $y_0 = C/2 - \delta$, $y_s = C/2 + \delta$, $s = 1, \ldots, I$
  - average rate of change is positive - not proportionally fair for $I \geq 2$!

$$\left(\sum_{s=1}^{I} \frac{2\delta}{c} \right) - \frac{2\delta}{c} = \frac{2(I-1)\delta}{c}$$

Proportional Fairness

- There exists one unique proportionally fair allocation. It is obtained by maximizing

$$J(\vec{x}) = \sum_{s} \ln(x_s)$$

over the set of feasible allocations for all sources *s*

Parking lot example

- For any choice of x_0 we should set x_i such that
  - $n_0 x_0 + n_i x_i = C$, $i = 1, \ldots, I$
- Maximize

$$f(x_0) = n_0 \ln(x_0) + \sum_{i=1}^{I} n_i(\ln(C - n_0 x_0) - \ln(n_i))$$

over the set $0 \le x_0 \le C / n_0$.

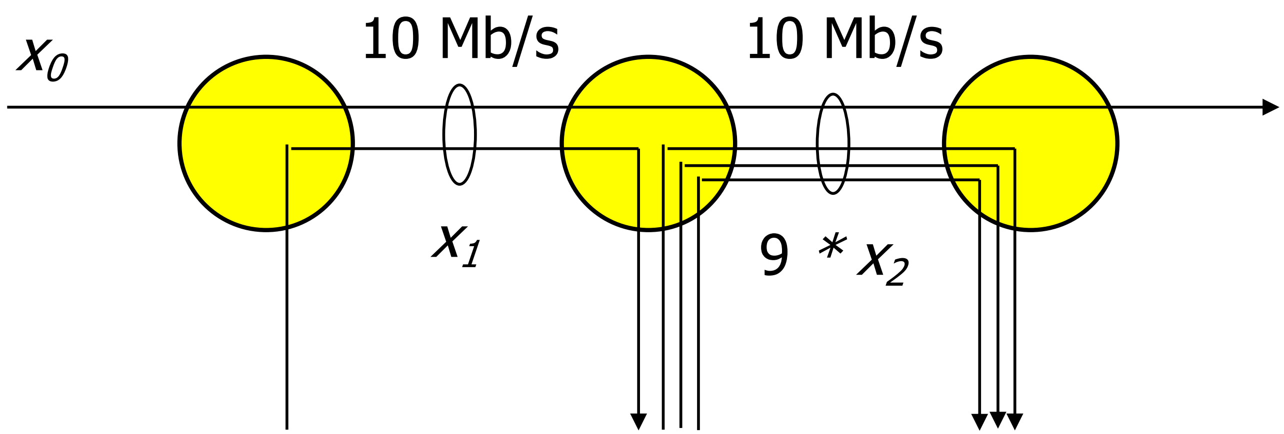
- The maximum is for

$$x_0 = \frac{C}{\sum_{i=0}^{I} n_i} \qquad x_i = \frac{C - n_0 x_0}{n_i}$$

- If $n_i = 1$, $x_0 = C/(I+1)$, $x_i = CI/(I+1)$
- Max-min allocation is $C/2$ for all rates - sources of type 0 get a smaller rate, since they use more network resources

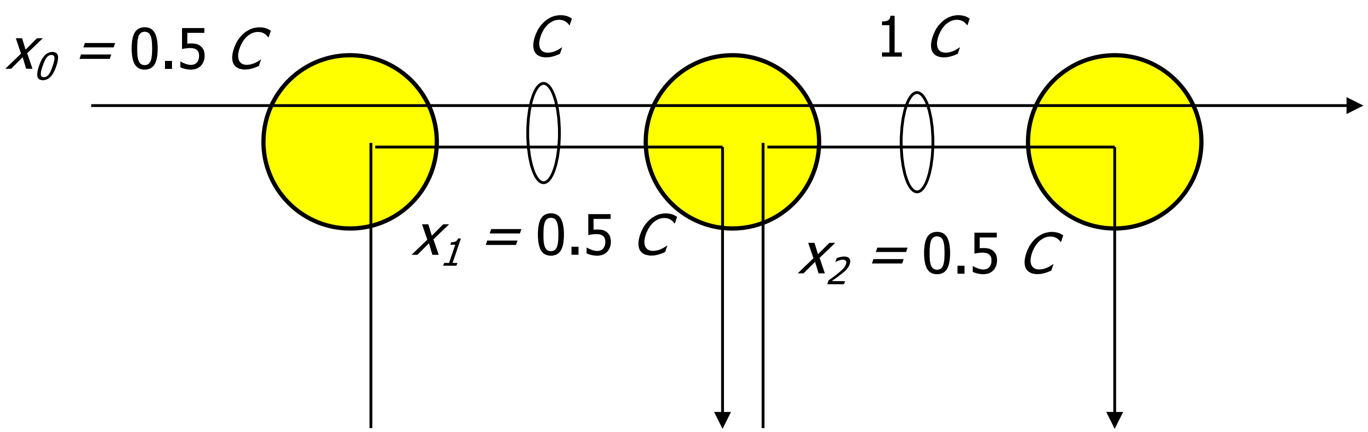
Proportionally Fair?

- $x_0 = 1$ $x_1 = 9$ $x_2 = 1$?
- $x_0 = 0.909$ $x_1 = 9.091$ $x_2 = 1.010$?

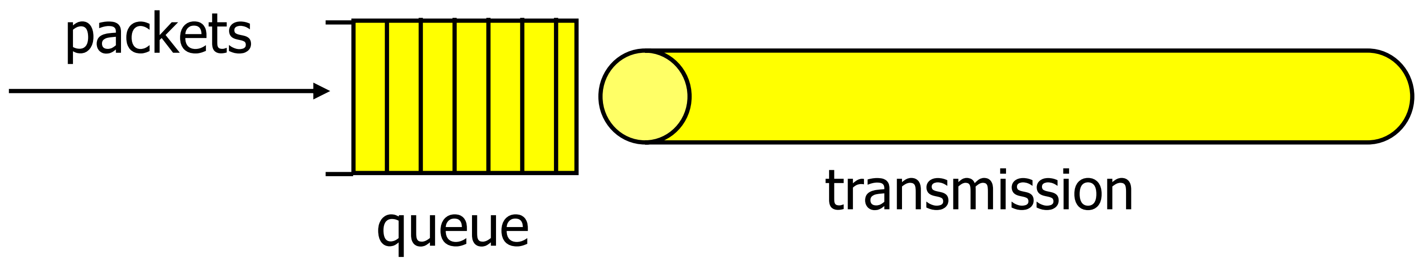


Comparisons

- $I = 2$, $n_i = 1$
- max throughput:
  - $x_0 = 0$, throughput = $2C$
- equal-share and max-min:
  - $x_0 = C/2$, $x_i = C/2$, throughput = $1.5C$
- proportional fairness:
  - $x_0 = C/3$, $x_i = 2C/3$, throughput = $5C/3$



Scheduling strategies



- Scheduler
  - defines the order of packet transmission
- Allocation strategy
  - throughput
    - which packet to choose for transmission
    - when chosen, packet benefits from a given throughput
  - buffers
    - which packet to drop, when no buffers

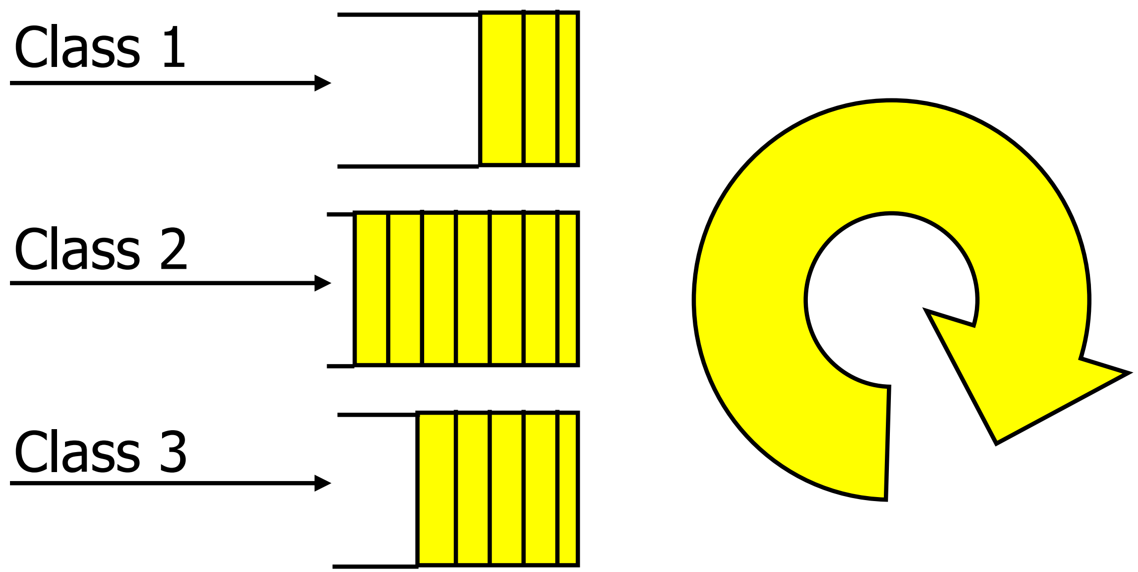
FIFO

- Current state of Internet routers
- Allows to share bandwidth
  - proportionally to the offered load
- No isolation
  - elastic flows (rate controlled by the source eg. TCP) may suffer from other flows
    - a greedy UDP flow may obtain an important part of the capacity
    - real time flows may suffer from long delays
- Last packets are dropped - tail drop
  - TCP adapt bandwidth based on losses
- RED (Random Early Detection) techniques
  - choose a packet randomly before congestion and drop it

Priority Queue

- Several queues of different priority
  - source may mark paquets with priority
    - eg. ToS field of IP
  - packets of the same priority served FIFO
  - non-preemptive
- Problems
  - starvation - high priority source prevents less priority sources from transmitting
  - TOS field in IP - 3 bits of priority
  - how to avoid everybody sending high priority packets?

Class Based Queueing (CBQ)

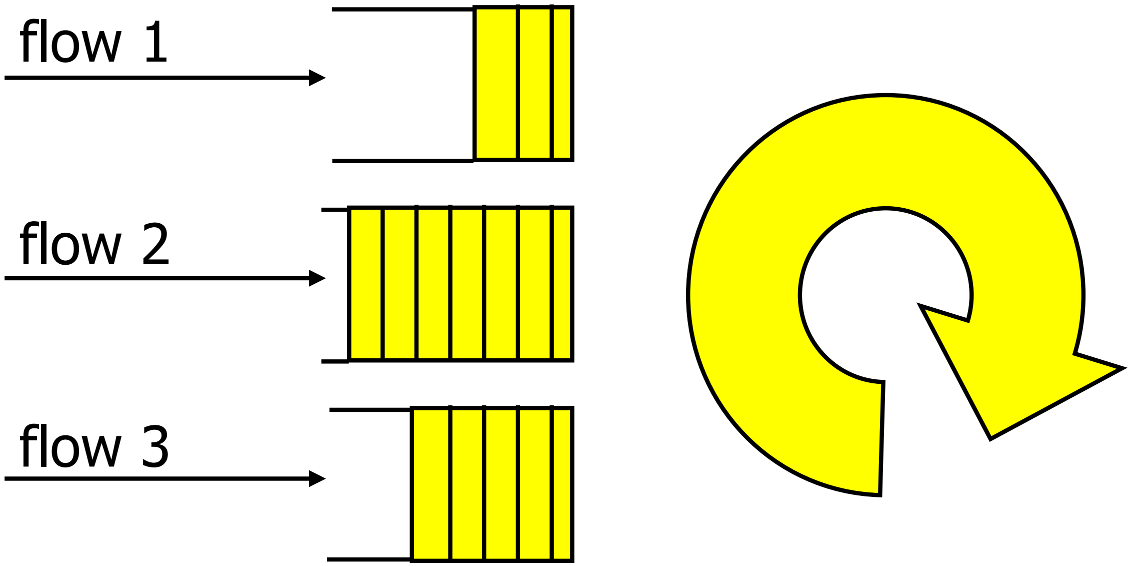


- Also called Custom Queueing (CISCO)
- Each queue serviced in round-robin order
- Dequeue a configured byte count from each queue in each cycle
- Each class obtains a configured proportion of link capacity

Characteristics

- Limited number of queues (CISCO - 16)
- Link sharing for Classes of Service (CoS)
  - based on protocols, addresses, ports
- Method for service differentiation
  - assign different proportions of capacity to different classes
  - not so drastic as Priority Queueing
- Avoids starvation

Per Flow Round Robin

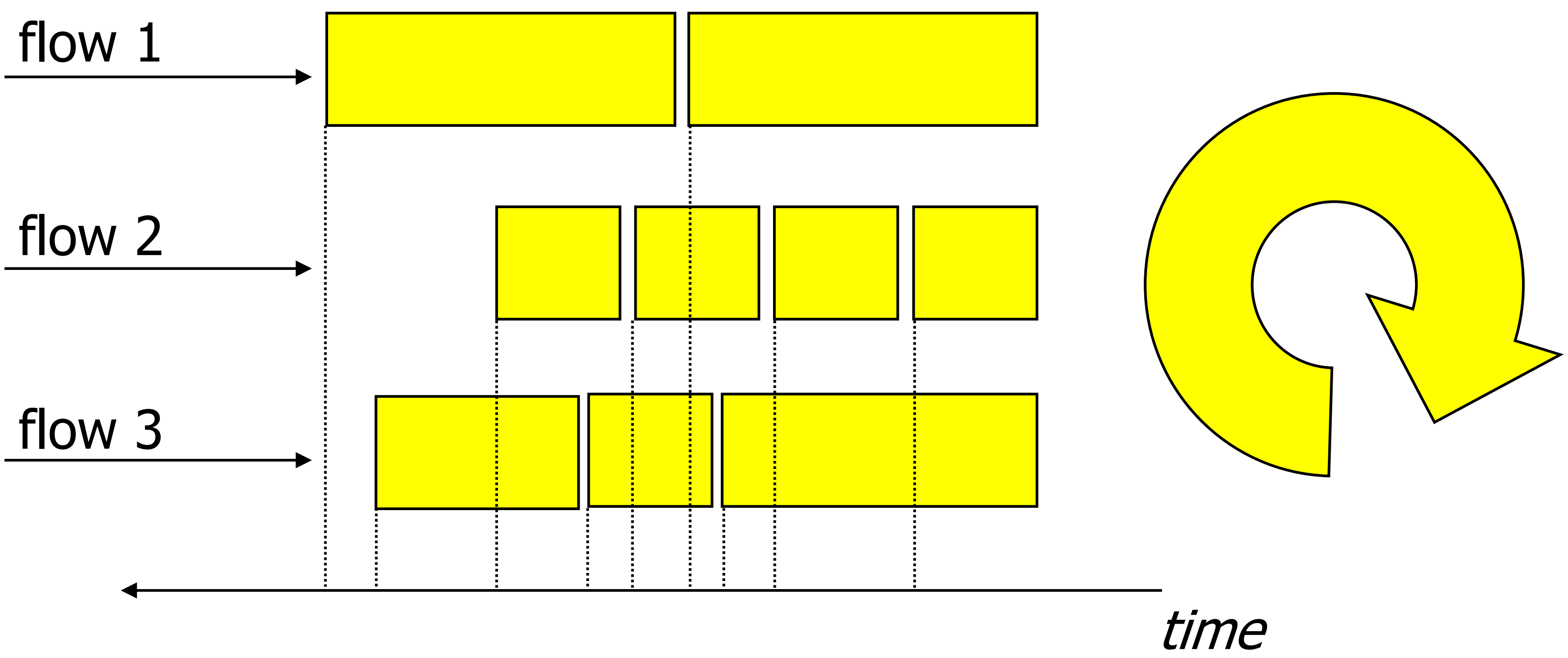


- Similar to Processor Sharing or Time Sharing
  - one queue per flow
  - cyclic service, one packet at a time

Characteristics

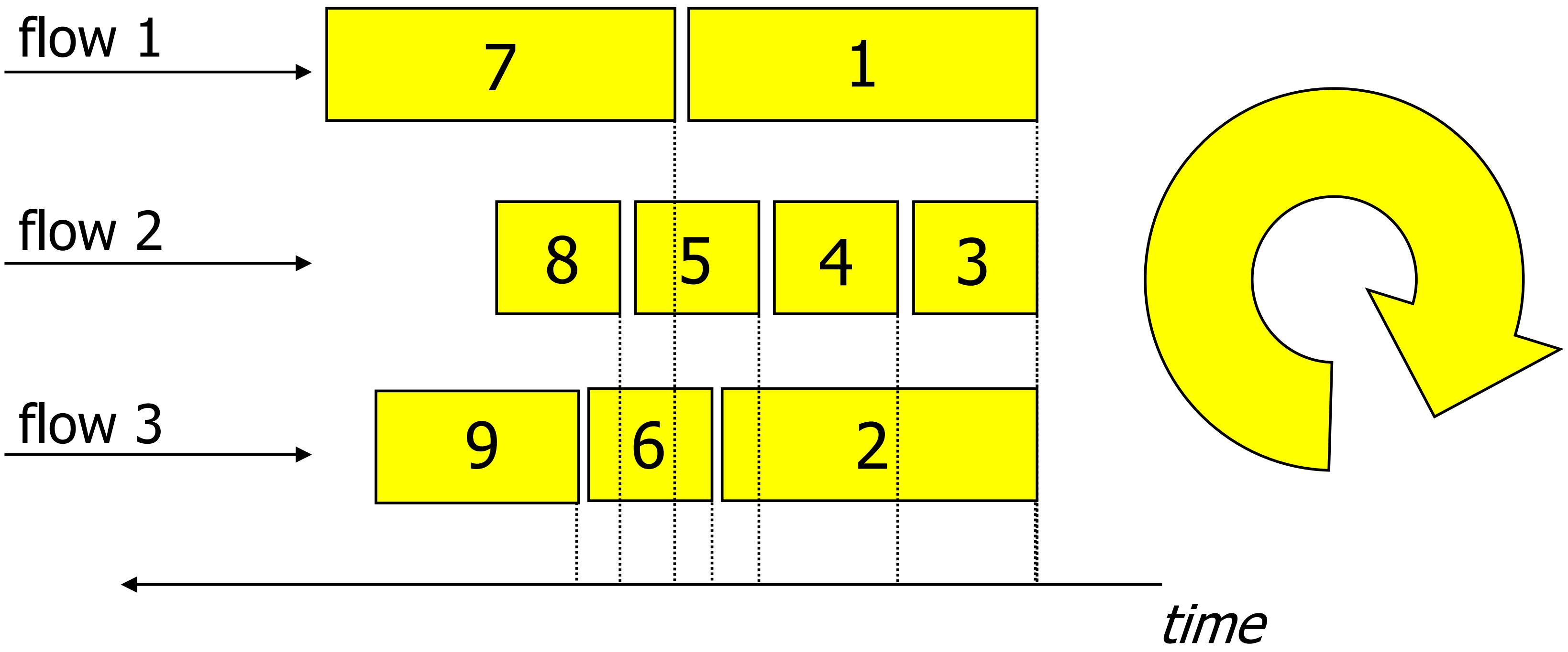
- It modifies the optimal strategy of sources
  - FIFO: be greedy - send as much as possible
  - RR: use your part the best
    - a greedy source will experience high delays and losses
- Isolation
  - good sources protected from bad ones
- Problems
  - flows sending large packets get more
  - cost of flow classification

Fair Queueing



- Round robin "bit per bit"
  - each packet marked with the transmission instant of the last bit
  - served in the order of the instants
  - allocates rates according to local max-min fairness

Start-Time Fair Queueing



- Round robin "bit per bit"
  - each packet marked with the transmission instant of the first bit
  - served in the order of the instants
  - allocates rates according to local max-min fairness

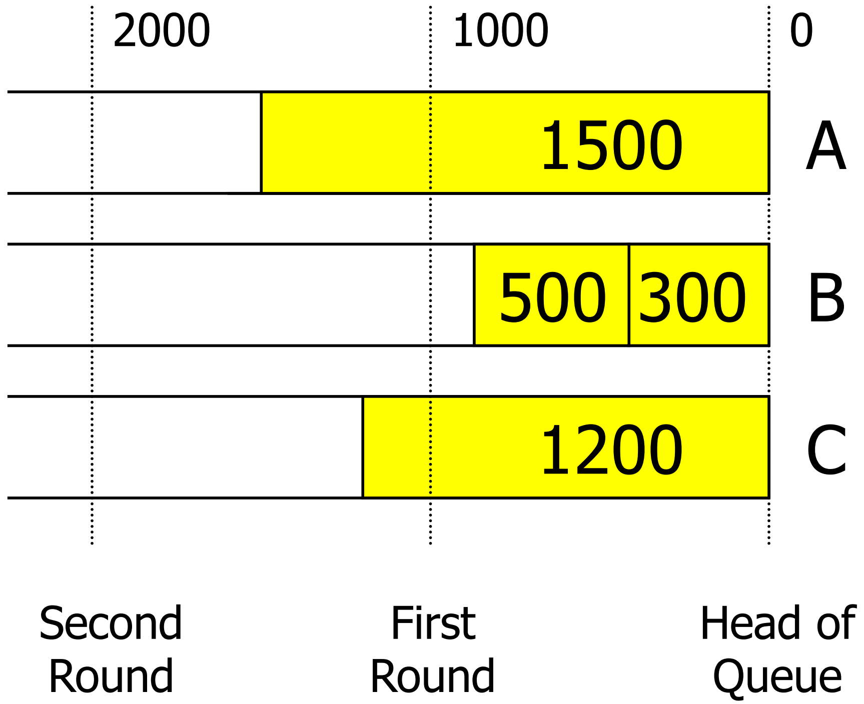
Deficit Round-Robin (DRR)

- A quantum of bits to serve from each connection in order
- Each queue: deficit counter (dc) (to store credits) with initial value zero
- Scheduler visits each non-empty queue, compares the packet at the head to dc and tries to serve one quantum of data
  - if size ≤ (quantum + dc)
    - send and save excess in dc: dc ← quantum + dc – size,
  - otherwise save entire quantum: dc += quantum
  - if no packet to send, reset dc
- Easier implementation than other fair policies
  - O(1)

Deficit Round-Robin

- DRR can handle variable packet size

Quantum size : 1000 bytes



- 1st Round
  - A's dc : 1000
  - B's dc : 200 (served twice)
  - C's dc : 1000
- 2nd Round
  - A's dc : 500 (served)
  - B's dc : 0
  - C's dc : 800 (served)

DRR: performance

- Handles variable length packets fairly
- If weights are assigned to the queues, then the quantum size applied for each queue is multiplied by the assigned weight
- Queues not served during round build up “credits”:
    - only non-empty queues
- Quantum normally set to max expected packet size:
    - ensures one packet per round, per non-empty queue
- Backlogged sources share bandwidth equally
- Simple to implement
    - Similar to round robin

Facts to remember

- In a packet network, sources should limit their sending rate by taking into consideration the state of the network
- Maximizing network throughput as a primary objective may lead to large unfairness
- Objective of congestion control is to provide both efficiency and some form of fairness
- Fairness can be defined in various ways: equal share, max-min, proportional
- Different scheduling algorithms allocate shares of capacity to flows in many ways