Flow Control in Packet-Switched Networks (Controlo de Fluxo em Redes com Comutação de Pacotes)

Modelação e Desempenho de Redes e Serviços

Prof. Amaro de Sousa (asou@ua.pt)

DETI-UA, 2025/2026

Flow control - introduction

In general:

- The total throughput reflects the service quantity supported by a packet-switched network.
- The average delay reflects the service quality provided by a packet-switched network.

The flow control mechanism is a feedback mechanism that establishes a compromise between throughput and average delay aiming to maintain the service quality within acceptable limits

---

- When the traffic load submitted to the network (the offered load) is low, flow control accepts all traffic and:

total throughput = offered load

- When the offered load is too high, flow control rejects part of it and:

total throughput = offered load – rejected load

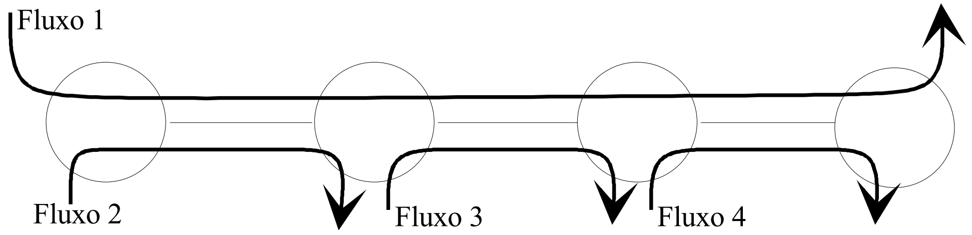
- In general, when the network routing reduces the average delay, flow control accepts more traffic (and vice-versa).

Flow control - introduction

Flow control algorithms should ideally fulfil the following requirements:

- They should set a good trade-off between:
  - the service quantity (given by the throughput, with eventually a minimum guaranteed transmission rate) and
  - the service quality (measured, for example, by the average delay)
- They should guarantee fairness (i.e., a fair treatment) among all traffic flows delivering the required quality of service.

Consider the case in the figure where all links have a capacity of 100:



The resource management is the task of assigning network resources to the different flows. This task can have different objectives:

- Maximum total throughput:

  Flow 1 = 0, Flows 2,3,4 = 100, Total throughput = 0+100+100+100 = 300

- Maximum fairness (i.e., max througput equally assigned to all flows):

  Flows 1,2,3,4 = 50, Total throughput = 50+50+50+50 = 200

Window-based flow control (I)

- Consider a data flow from a source node *A* to a destination node *B*. In a window-based flow control mechanism:
  - For each received data unit, destination node *B* notifies the reception of the data unit by sending a *permission* to source node A:
    - permission information can be sent as a dedicated control packet or piggybacked in a data packet sent in the opposite direction (from *B* to *A*).
  - When the source node *A* receives a permission from node *B*, node *A* is authorized to send another data unit to node *B*.
- A window-based flow control mechanism is always combined with an error control ARQ (Automatic Repeat reQuest) mechanism:
  - in this case, data units are numbered with a SN (*sequence number*) and permissions are also numbered with a AN (*acknowledgment number*) to notify the reception of each data unit without errors.

Window-based flow control (II)

- A window-based flow control of a data flow, from a source node *A* to a destination node *B*, is realized when:
  - there is a limit on the total number of data units that can be transmitted by node *A* and not yet notified with permissions sent by node *B*,
  - the limit is designated by *window size*, or simply, *window*.

- The source node *A* and the destination node *B* can be either two network nodes, or one user terminal and its attached network node or two user terminals that are the end nodes of a traffic flow.

Next, we consider the flow control based on **end-to-end windows**:

- for each packet flow, the flow control mechanism is running between its source node and the destination node (i.e., the intermediate nodes on the routing path(s) of the flow are not involved)
- this is the strategy used by TCP in TCP/IP networks

End-to-end windows (I)

- In window-based flow control, the transmission rate at the source node is reduced when the permissions take longer times to return.
- Therefore, if the routing path of a given packet flow becomes congested, the time difference between the transmission of a packet and the reception of its permission becomes larger (due to the increased delay) making the source node to reduce its transmission rate (and alleviating in this way the path congestion).
- Moreover, the destination node can also delay the sending of permissions in order to explicitly reduce the transmission rate of the source (for example, to avoid overflow of the receiver buffer).

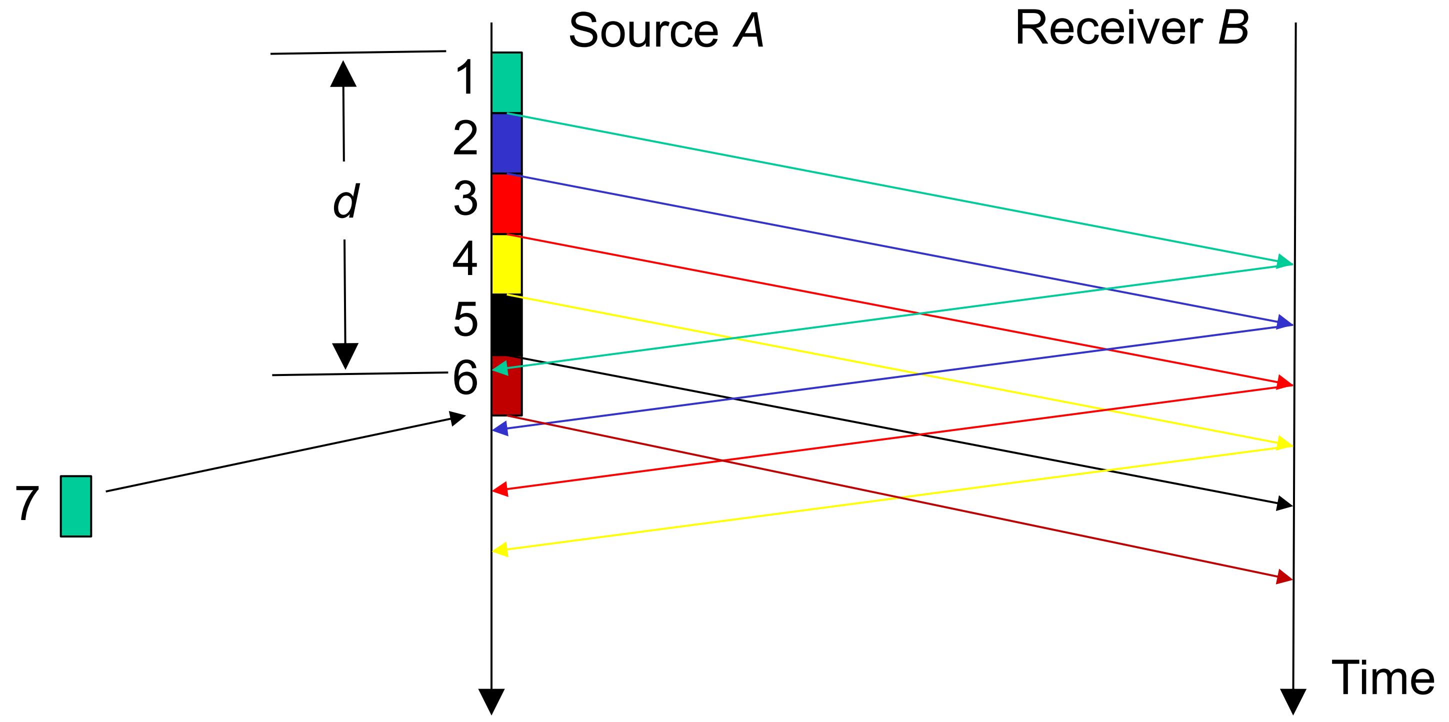
End-to-end windows (II)

- Consider the window size given by *W*, in number of packets (it can be in other units as Bytes in TCP protocol).
  - When a packet is received by the destination node, it sends a permission to the source node to enable the sending of one more packet.
- Consider the total round trip delay given by *d* and the average transmission time of each packet given by *X* (i.e., the maximum throughput available in the network is 1/*X*, in packets/second):
  - ✓ If $d \leq WX$, the transmission of *W* packets takes more time than the round trip time; so, the source can send packets at the rate of 1/*X* packets/second.
  - ✓ If $d > WX$, the flow control is active since the round trip time is high, and *W* packets are sent before the reception of the first permission.

Therefore, the sending rate is given by: $r = \min\left\{\frac{1}{X}, \frac{W}{d}\right\}$

Illustration of end-to-end windows (I)

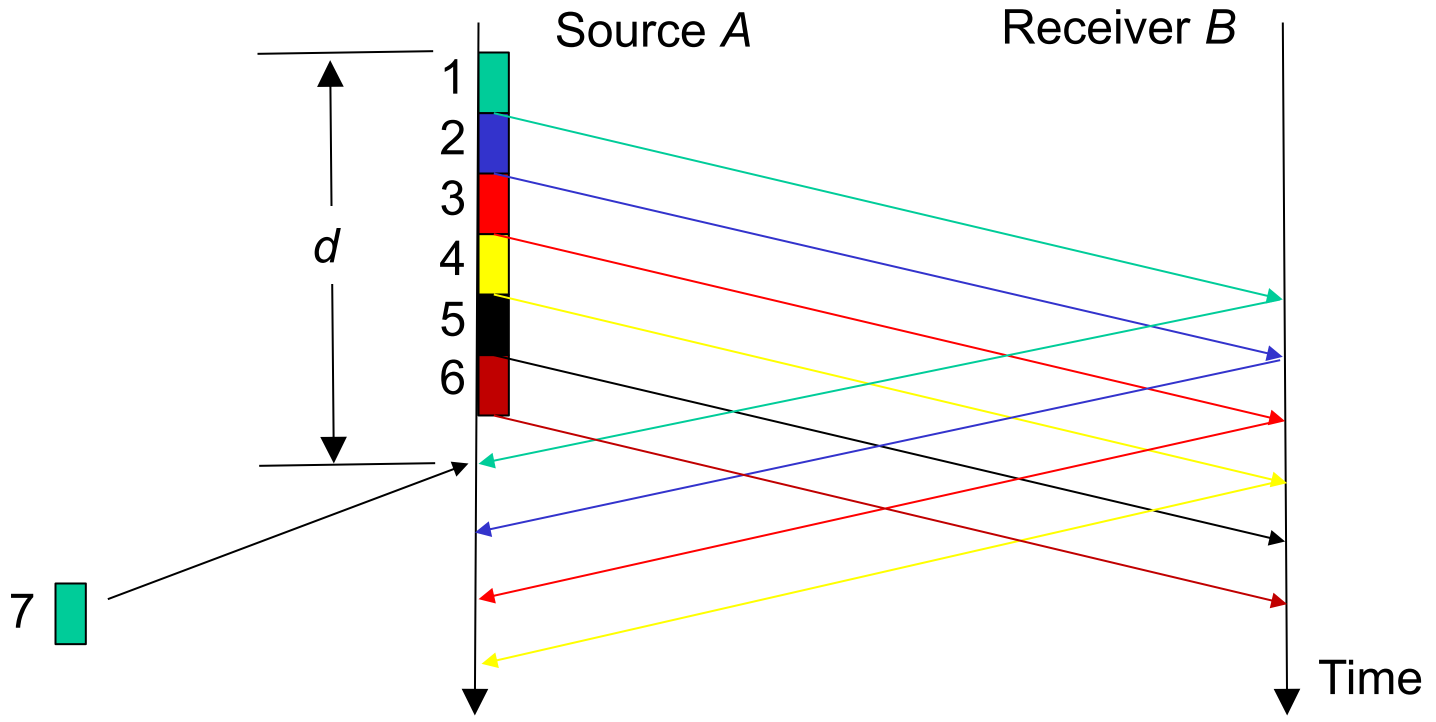
Consider W = 6 packets from source node A to receiver node B.



$d \leq WX$ (the transmission of the 6 first packets takes a longer time than the round trip delay d) $\rightarrow$ the 7th packet can be transmitted right after the transmission of the 6th packet

Illustration of end-to-end windows (II)

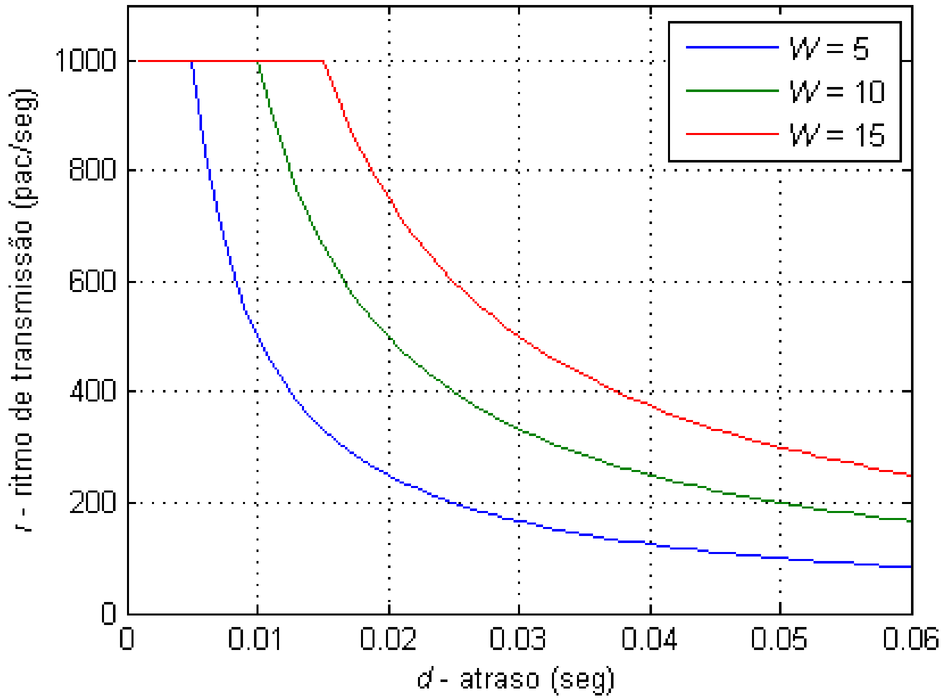
Consider *W* = 6 packets from source node *A* to receiver node *B*.



d > *WX* (the transmission of the 6 first packets takes less time than the round trip delay *d*) $\rightarrow$ the 7th packet can only be transmitted when source *A* receives the permission of the 1st packet

Illustration of end-to-end windows (III)

Example: X = 1 msec. and W = 5, 10 or 15 packets.



$$r = \min\left\{\frac{1}{X}, \frac{W}{d}\right\}$$

- ✓ For values $d \leq WX$, the source sends packets at the maximum rate $r = 1/10^{-3} = 1000$ (in pacekts/second)
- ✓ For values $d > WX$, flow control is active, and the source sends packets at the rate $r = W/d$ (in packets/second)

Ideal window size

- The window size represents a trade-off between throughput (i.e., the sending rate) and average delay:
- on one hand, the window size should be small to limit the number of packets travelling in the network avoiding as much as possible packet delays and link congestion;
- on the other hand, the window size should be large enough to let sources transmit at maximum throughput when traffic load is not high.
- Ideally, the source node of each flow should be able to use the maximum available throughput provided by the network when no other flow is active (i.e., no other source is sending packets).
- Since the source node can send packets at full rate when $d \le WX$, then, the ideal window size (in number of packets) should be

$$W = \left\lceil \frac{d}{X} \right\rceil$$

where $\lceil z \rceil$ represents the lowest integer higher than z and d is the minimum round-trip delay provided by the network.

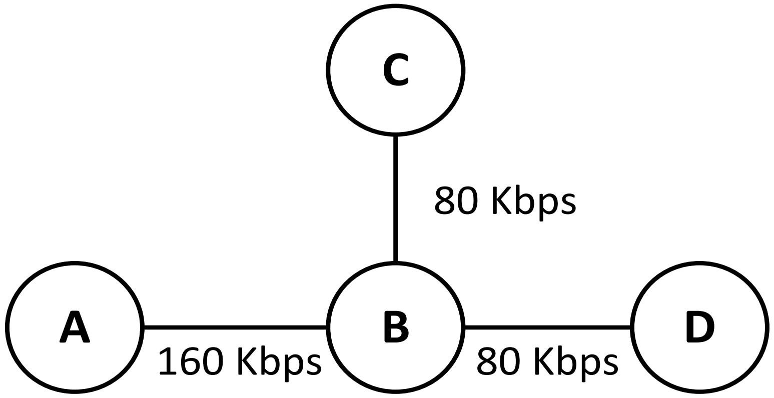
Example 1

Consider the packet switching network of the figure below. For all links, consider that the propagation delay is 10 msec. on each direction.

The network is supporting 2 packet flows: A→D with packets of average size 1000 Bytes and C→D with packets of average size 500 Bytes.

Both flows are controlled by end-to-end windows and in both cases the permissions are sent through dedicated packets of size 100 Bytes.

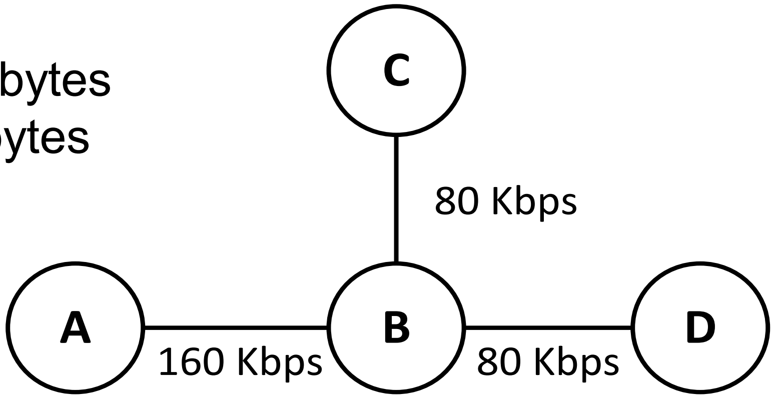
Determine the ideal windows size (in number of packets) of each flow enabling each flow source to transmit at full rate when the other flow is not active.



Example 1 - resolution

A $\rightarrow$ D with packets of average size 1000 bytes
C $\rightarrow$ D with packets of average size 500 bytes

$$W = \left\lceil \frac{d}{X} \right\rceil$$



$$W_{AD} = \left\lceil \frac{\frac{8\times 1000}{160000} + 0.01 + \frac{8\times 1000}{80000} + 0.01 + \frac{8\times 100}{80000} + 0.01 + \frac{8\times 100}{160000} + 0.01}{\frac{8\times 1000}{80000}} \right\rceil = \left\lceil \frac{0.205}{0.1} \right\rceil = 3 \text{ packets}$$

$$W_{CD} = \left\lceil \frac{\frac{8\times 500}{80000} + 0.01 + \frac{8\times 500}{80000} + 0.01 + \frac{8\times 100}{80000} + 0.01 + \frac{8\times 100}{80000} + 0.01}{\frac{8\times 500}{80000}} \right\rceil = \left\lceil \frac{0.16}{0.05} \right\rceil = 4 \text{ packets}$$

Limitations of the end-to-end window-based flow control

1. It does not allow to guarantee a minimum throughput for each flow: when the number of flows increases, the throughput obtained by each flow decreases.
2. It does not allow to guarantee a minimum delay for each flow. Consider n flows with window sizes W_1, …, W_n. The total number of packets and permissions in the network is: $\sum_{i=1}^{n} W_i$

   and the total number of packets is $\sum_{i=1}^{n} \beta_i W_i$ where β_i is a value between 0 and 1.

   By the Little's theorem, the average packet delay is $T = \frac{\sum_{i=1}^{n} \beta_i W_i}{\lambda}$

   where λ is the throughput (in packets/second) of all flows. So:

   - when the number of flows increases, the throughput can increase only up to a maximum value provided by the network capacity;
   - after this maximum value is reached, the average delay increases linearly with the number of flows and, therefore, the average packet delay can become arbitrarily large.

Transmission rate control

- The flow control of the network can assign to each flow a given maximum transmission rate appropriate to its requirements of QoS (quality of service).
- This transmission rate can be defined in the establishment of a virtual circuit (for example, when establishing a Label Switched Path in MPLS networks).
- Next, we consider two methods for transmission rate control:
  - Window-based rate control
  - Leaky bucket algorithm, as used for example by the RSVP protocol on the Integrated Services (IntServ) architecture of IP networks

Window-based transmission rate control (I)

- Let us assume that a transmission rate of *r* packets/second was assigned to a particular flow (from a given source node to a given destination node).

- A simple idea for a rate control algorithm is to let the source node transmit at most one packet on each 1/*r* seconds.

- Nevertheless, this approach forces significant delays at the source node when the packet generation is highly bursty (as it is usual on data transmission services).

- In this case, it is better to let the source node transmit at most *W* packets on each *W*/*r* seconds, as it allows the transmission of bursts of *W* packets.

- This is the idea of the window-based transmission rate control mechanism.

Window-based transmission rate control (II)

If a given flow was assigned with: (i) an average transmission rate of *r* packets/second and (ii) a window of *W* packets, the method works as follows:

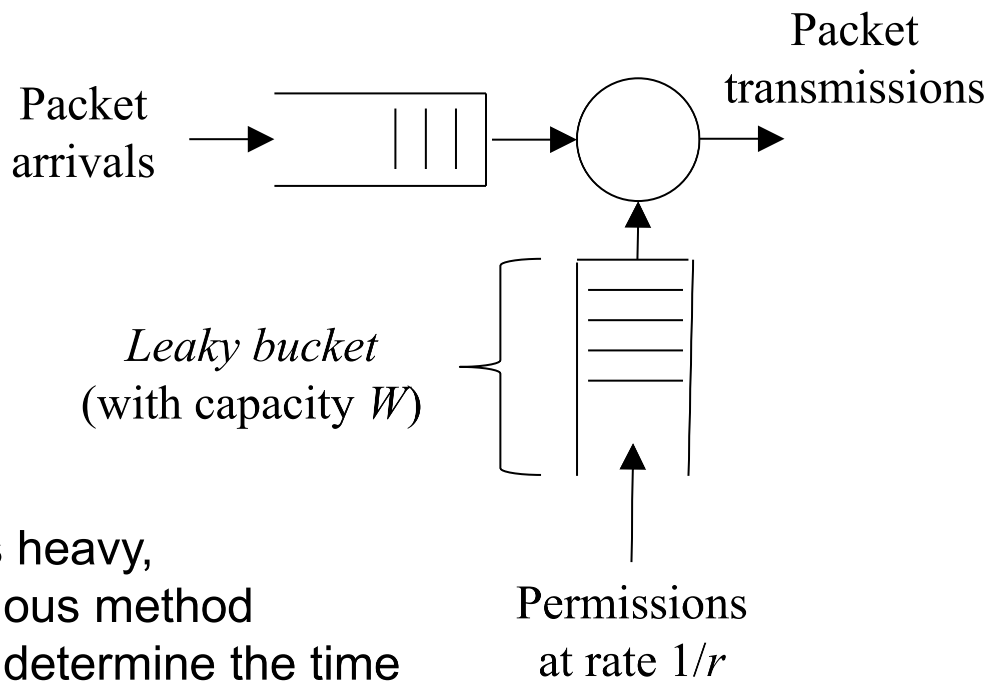
1. The source node maintains a counter *x* that indicates, at each time instant, the number of packets of the assigned window that it can still transmit (*x* is initially set with *W*).
2. Whenever a packet is transmitted, the counter *x* is decreased and, after *W*/*r* seconds, increased again (it required a timer for each transmitted packet).
3. The packets can be transmitted only if $x > 0$ (the required number of timers is *W*).

Note: This method is similar to the window-based flow control. The difference lies in the fact that in the window-based flow control the counter is incremented when a permission is received.

Disadvantage: this method is computationally hard since it requires *W* timers for each flow to work properly.

Transmission rate control based of *leaky bucket*

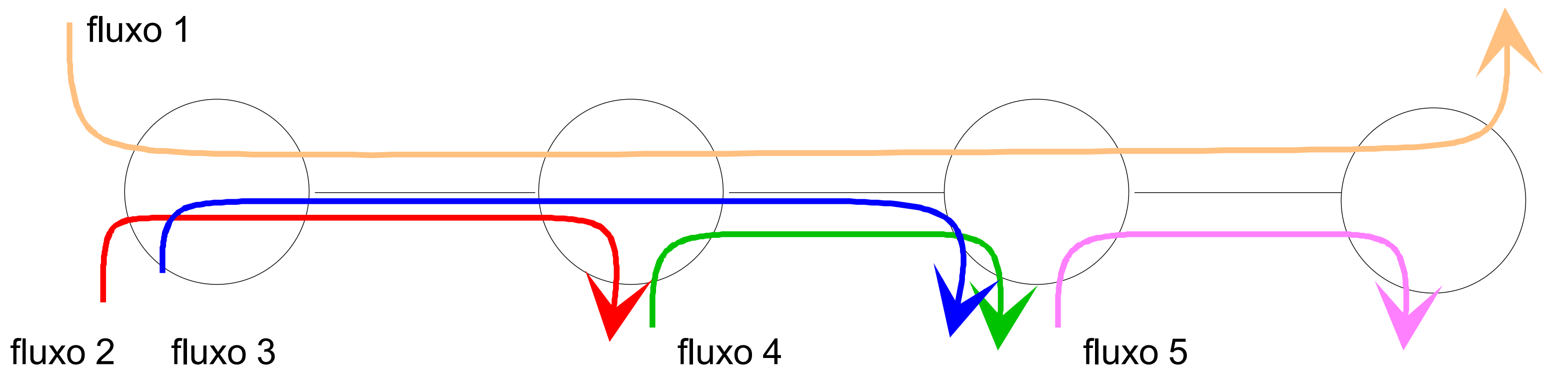
- In this method, the counter is incremented periodically on each 1/*r* seconds, until a maximum of *W* packets.
- The method can be viewed in the following way (*leaky bucket* model):
  - there is a queue of packets to be transmitted
  - there is a queue of permissions (named *leaky bucket*), with a capacity to *W* permissions;
  - a new permission is inserted in the leaky bucket every 1/*r* seconds;
  - for each packet to be transmitted, a permission must be taken out of the leaky bucket and packets wait in the queue if the leaky bucket is empty.



Advantage: this method is much less heavy, in computational terms, then the previous method as it requires only 1 timer per flow (to determine the time instants of the permission insertions in the leaky bucket).

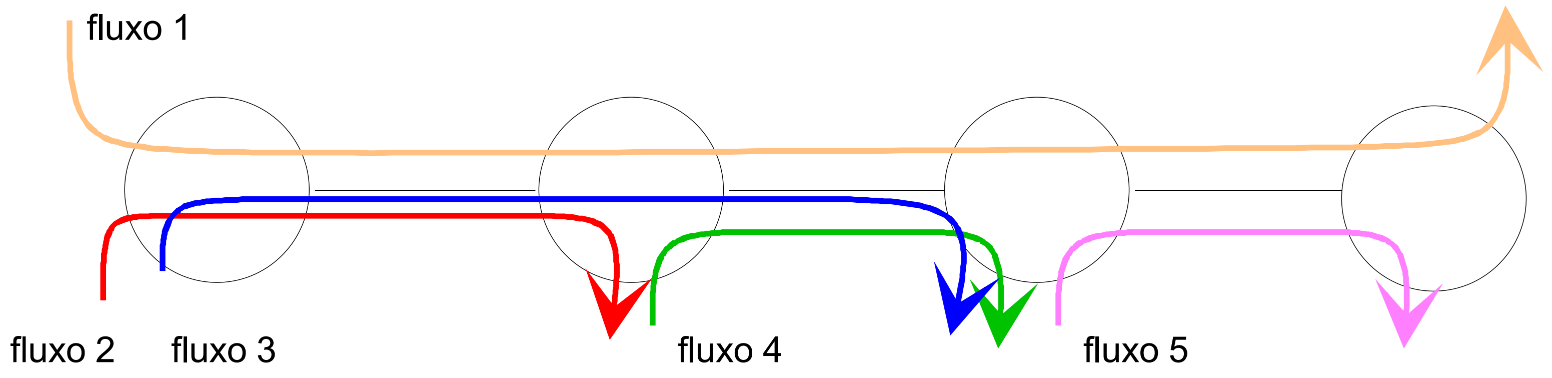
Assignment of transmission rates

- Consider the network of the figure assuming that all links have a capacity of 120 packets/sec.
- A fair assignment of transmission rates to the flows would be to assign a transmission rate of 1/3 × 120 = 40 packets/sec to each flow.
- Nevertheless, there is no reason to constraint the transmission rate of flow 5 to 40 packets/sec, since flow 5 can be assigned with 80 packets/sec without jeopardizing the rates assigned to the other flows.



Max-min fairness concept

- This example introduces the max-min fairness concept.
- This concept states that we maximize the rates assigned to the flows that can have lower transmission rates.
- An alternative way to formulate this concept is:
  - we maximize the rates assigned to each flow, guaranteeing that an increase in the rate of flow *i* does not force a decrease of the rate of any flow whose assigned rate is lower than the rate of *i*.



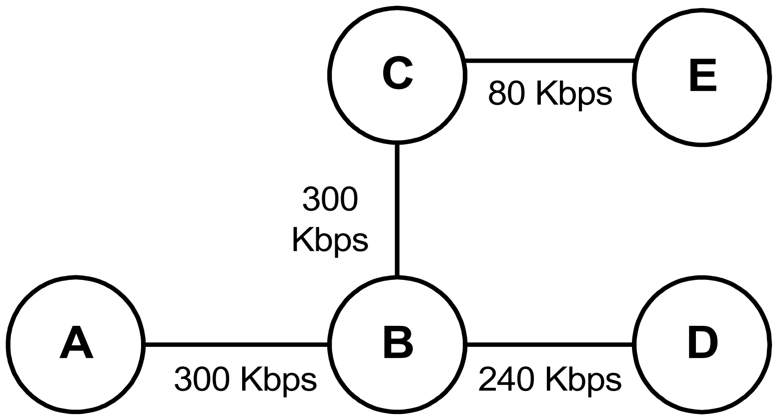
Example 2

Consider the packet switching network presented in the figure below.

The network supports 5 packet flows: from A to B, from A to C, from A to D, from D to B and from B to E.

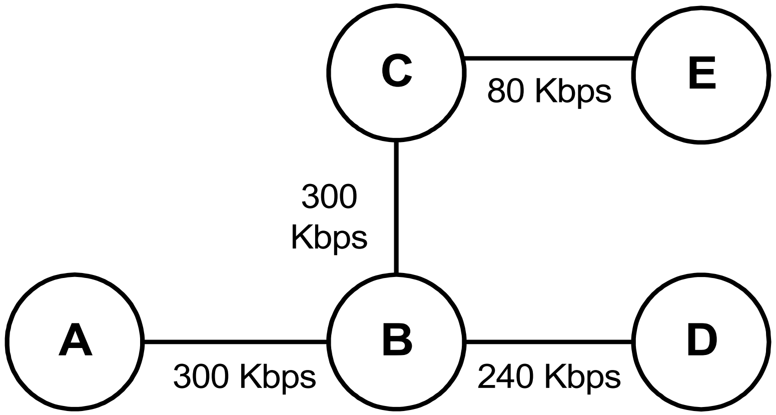
The network is able to control the maximum rate of each flow through an adequate transmission rate control method.

Determine the maximum transmission rate to be assigned to each flow according to the max-min fairness concept.



Example 2 - resolution

5 packet flows:
- from A to B
- from A to C
- from A to D
- from B to D
- from B to E



<u>1st iteration:</u>
- link AB assigns 300/3 = 100 Kbps per flow
- link BC assigns 300/2 = 150 Kbps per flow
- link BD assigns 240/2 = 120 Kbps per flow
- link CE assigns 80/1 = 80 Kbps per flow

Link CE assigns the lowest value: flow B→E is assigned with 80 Kbps.

<u>2nd iteration:</u>
- link AB assigns 300/3 = 100 Kbps per flow
- link BC assigns (300–80)/1 = 220 Kbps per flow
- link BD assigns 240/2 = 120 Kbps per flow

Link AB assigns the lowest value: flows A→B, A→C, A→D are assigned with 100 Kbps.

<u>3rd iteration:</u>
- link BD assigns (240–100)/1 = 140 Kbps per flow

Flow B→D is assigned with 140 Kbps.