Communication Networks

MAP-TELE

2011/12

José Ruela

Network basic mechanisms

Traffic Control

Flow, Congestion and Admission Control

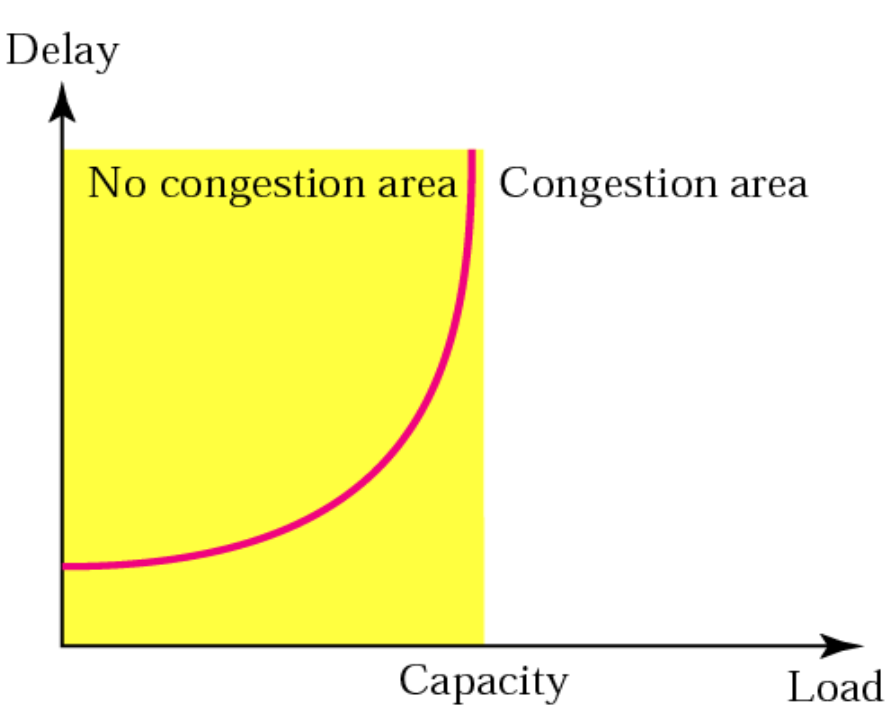
Flow control

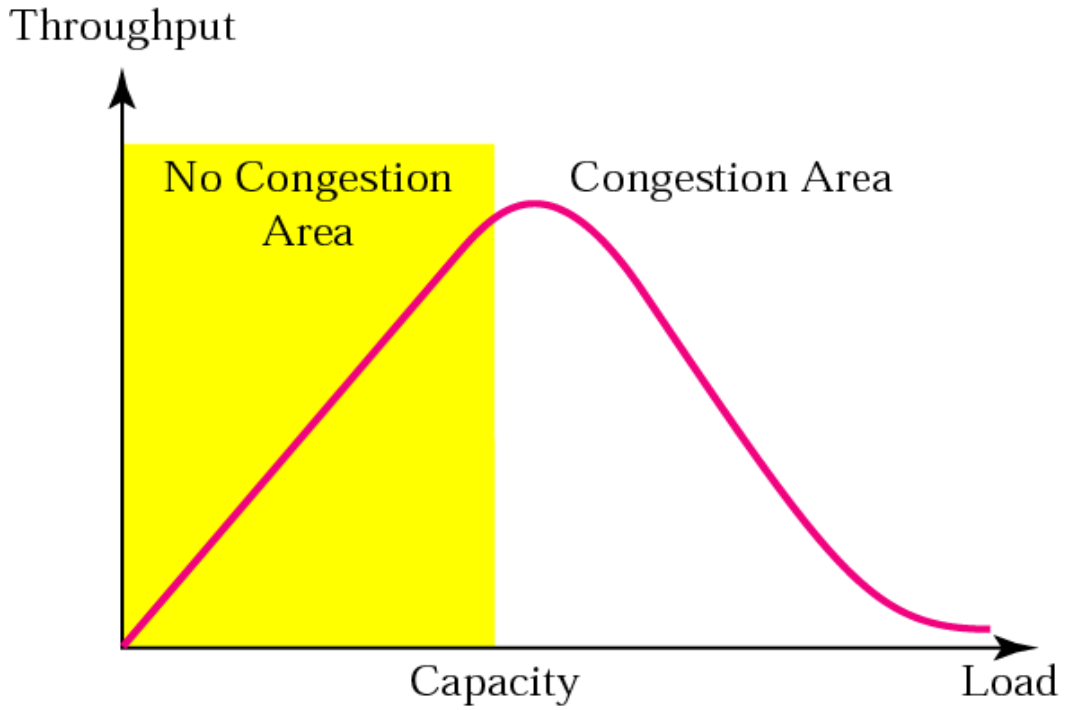
- In order to characterize the scope and goals of flow control and to distinguish it from other traffic control techniques (e.g., admission and congestion control) it is useful to first analyse some typical flow control scenarios
- In the simplest scenario, a source (sender) and a destination (receiver) are directly connected over a point-to-point link (channel)
  - The goal is to match the sending rate to the rate that the receiver can handle – the rate may have to be reduced due to buffer and/or processing constraints at the receiver
  - Data Link protocols provide flow control either explicitly or as a side effect of the acknowledgement and error control mechanisms (*stop and wait* or *sliding window*)
- Similarly, adjacent nodes on a network path also play a sender and a receiver role – in this case, if required, flow control is performed on a hop-by-hop basis, either at the Data Link or at the Network layer (e.g., in X.25 networks there was provision for flow control per Virtual Circuit)

Flow control and congestion control

- When the source and the destination are attached to a network, the receiver must still be capable of controlling the rate of the sender – this is performed on an end-to-end basis, typically at the Transport layer (e.g., TCP)
- The throughput that a network can offer to a flow depends on the link rates along the path and on the amount of transmission resources that the network nodes can (dynamically) allocate to the flow
  - Flow control by the destination will be effective when the network can sustain the source rate – in this case, the goal is to match the target receiving rate
    - End-to-end flow control is influenced by the network behaviour – round trip time and flow throughput are affected by changing network conditions and traffic loads
  - If the bottleneck is not the receiver but the network (due to slow links or overloaded nodes), the throughput will in practice be limited by the network
    - Either the senders are capable of adapting their rates to match the network conditions or packets will be lost when buffer overflow starts to occur in some nodes
    - In case of congestion, additional measures must be taken and senders are forced to reduce their rates based on explicit or implicit information of the network state
- In conclusion, we may say that flow control is a set of techniques that allow regulating the rate of a traffic source (sender) so that it matches the available service rate at the destination (receiver) and at the network

Effects of network congestion





Flow control and admission control

- In a network that supports QoS guarantees, agreements must be established between end-users and the network before traffic is submitted to the network
  - A source must characterize its traffic (described by traffic parameters) and specify QoS goals (described by QoS parameters)
- An *admission control* mechanism is necessary to determine whether a new traffic flow can be accepted by the network, that is, whether the network may commit resources to meet performance objectives
- In case a flow is accepted, the sender must regulate (flow control) its traffic so that it conforms with the contract (and thus should receive the expected service with high probability) and the network verifies whether traffic is compliant to give it the promised treatment
- This process is typical of *open loop* control schemes

Closed loop vs. open loop control

- Flow control schemes can be classified as *open loop* or *closed loop*
- Open loop control acts on flows that have been accepted by the network through a negotiation process
  - The source describes its behaviour (traffic parameters) and the network reserves the adequate resources in case it can accept a flow (admission control)
  - The source shapes outgoing flows so that they are compliant with the agreed parameters (auto-regulation) and the network polices incoming flows
- Closed-loop schemes are used when flows do not get enough resources due to a variety of reasons, e.g., resource reservation is not necessary or not feasible (not supported by the network or reservation is not adequate to the type of traffic) or the network overbooks resources (statistical multiplexing)
  - The source should dynamically adapt the sending rate in order to match its current share of network resources (*service rate*) determined by the network model of resource sharing
  - The source needs to get information about the network state (feedback), so that it can react to network changes and adjust the sending rate so that an optimum equilibrium point is reached – this feedback may be implicit or explicit

Closed loop flow control

- Existing closed loop strategies are based on rather different criteria
- First generation schemes simply try to match the sending rate at the source to the service rate at the destination (receiver)
  - Examples are *on-off*, *stop and wait*, and *static window*
- Second generation schemes also take into account the network service rate and thus are an instrument of congestion control
  - The network may provide *explicit* feedback information or the source may derive such information in an *implicit* way (based on some events or performance measurements)
  - Control may be performed *hop-by-hop* or *end-to-end* and can be based on dynamically adapting the *sender window* or the *sender rate*
    - In window based schemes, a sender window used for congestion control is decoupled from the window advertised by the receiver – both are necessary, since the latter is still used for flow control by the receiver
    - In rate based schemes the sender rate is directly controlled, while in window based schemes the sender rate is indirectly controlled by modifying the sender window

On-off

- In *on-off* flow control the receiver sends to the transmitter an *off* signal to stop transmission and an *on* signal to resume transmission
  - An example is the XON/XOFF protocol used to control serial input-output devices, e.g., CTRL Q (ASCII 0x11) and CTRL S (ASCII 0x13), respectively
  - IEEE 802.3x defines a PAUSE command (carried on MAC control frames) for flow control in full-duplex operation – one station requests that the other station stops transmitting frames for a specified period of time (no *on* signal is required)
  - HDLC provides a *Receiver Not Ready* (RNR) Supervisory frame for *stop and go* flow control that bypasses the window mechanism (RNR for *stop* and RR/REJ for *go*) – *stop and go* should not be confused with *stop and wait*
- Before the *off* signal produces effect, the sender may have continued to transmit data during a round trip delay interval
  - The receiver either provides buffers to handle data in transit or simply discards frames if it runs out of buffers – in HDLC, depending on the situation, a RR or a REJ should be sent to resume transmission
  - Buffer overflow may occur when the *off* signal is lost
  - If the *on* signal is lost, the sender will be blocked and therefore a time-out mechanism is required (HDLC provides a solution to this problem)
- The mechanism is primarily used when the round trip delay is small

Stop and wait

- *Stop and wait* is primarily a basic acknowledgement and error control mechanism (ARQ), but it can also be considered a form of flow control since the sender must pause transmission until an ACK is received
- When $a = \tau / T_f$ is large, the sender rate will be severely reduced
- The receiver may further reduce the sender rate by delaying ACKs, but this may trigger a time-out and retransmission by the sender and thus should be used with care
- As a flow control mechanism, *stop and wait* does not provide the receiver with the means to tightly control the sender rate, since it strongly depends on the propagation delay
  - As the propagation delay increases, the sender rate rapidly decreases, even if no flow control is intended by the receiver
  - *Stop and wait* flow control on an end-to-end basis over a network is even worse, not only because the round trip time is much higher than in small distance point-to-point links but because it is highly variable

Static window (node-to-node)

- *Sliding window* protocols are also primarily used for error control and provide a much higher efficiency than *stop and wait*, which can even reach 100% if $W \geq 1 + 2a$
- Static windows are typically used in Data Link layer protocols
  - Control is more flexible than *stop and wait*, since it is possible to configure a value for the window size (W) up to the maximum value allowed by the protocol
  - It has the problem that W is usually measured in frames (not bytes), and thus small frames will lead to a low efficiency
- They inherently provide flow control as a side effect, since a sender will have to stop after sending W frames without receiving an ACK
  - If $W < 1 + 2a$, flow control may occur (if new frames are ready for transmission), even when not necessary or not intended by the receiver
  - If $W \geq 1 + 2a$, the receiver will not be able to control the rate of the sender, unless ACKs are delayed
    - This is the reason why in HDLC a *stop and go* mechanism is also provided

Static window (end-to-end)

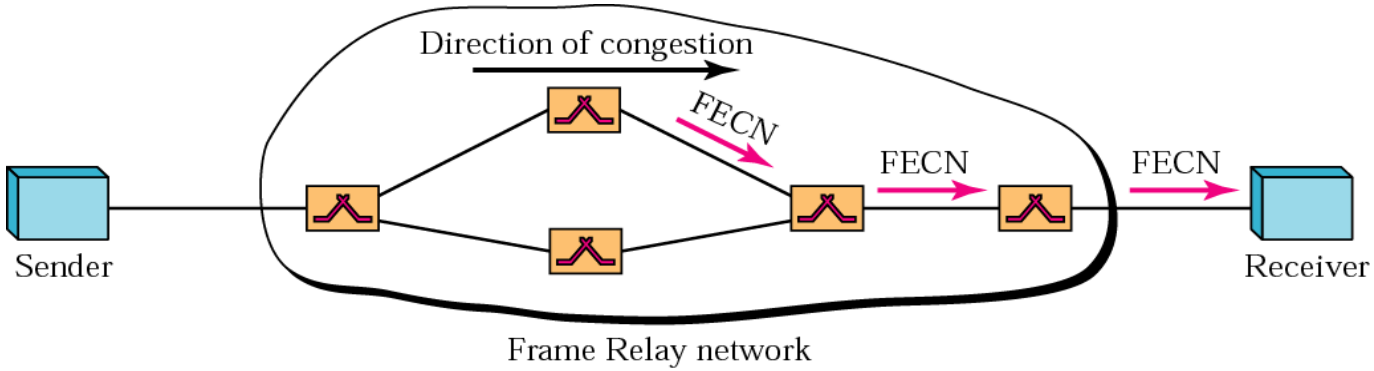
- When static windows are used end-to-end over a network they raise similar problems as *stop and wait*, in spite of being more flexible
- Assuming that the window offered by the receiver is W and that the round trip time is RTT, the source rate R will be at most W / RTT
- If the bottleneck rate along the path is *r*, to keep the bottleneck fully utilized (not wasting resources), the condition W / RTT $\geq$ r should hold
  - If R = W / RTT < r, the bottleneck link will be idle for part of RTT
  - If R = W / RTT > r, packets will queue up and will be lost if there is no sufficient buffer space at the bottleneck node
- It is not easy to tune the window size, since the bottleneck rate on the network and the round trip delay change from session to session and even during the same session
- In TCP, the receiver window is not fixed and the current value is sent with each ACK as an updated credit to the sender
  - In this way the sender rate automatically adapts to the current receiver window and round trip time, but not to congestion in the network

DECbit flow control

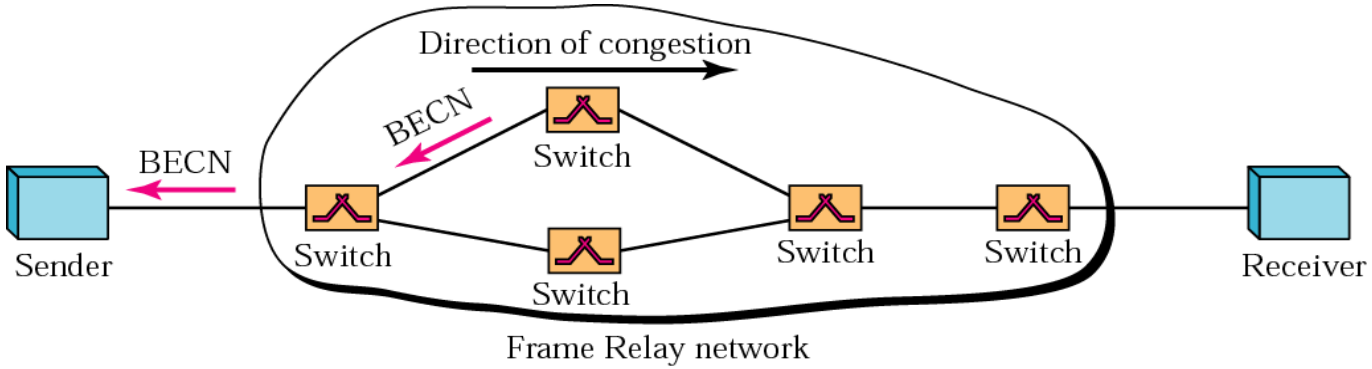
- The DECbit scheme is performed end-to-end, based on a dynamic window and uses explicit feedback from the network
  - Every packet carries on its header a bit that can be set by any router on the path to indicate congestion (once set, it is not reset by routers down the path)
  - The bit is copied by the receiver on ACKs sent back to the source
- A router computes the bandwidth demand from each source and the mean aggregate queue length *Q* over queue regeneration cycles (a busy period followed by an idle period)
  - If Q > 1, the congestion bit is set on flows whose demand exceeds the fair share and, if Q > 2, the congestion bit is set for all flows
- A source uses the congestion bit received on successive ACKs to change the window, based on an *additive increase multiplicative decrease* policy
  - The source counts the number of congestion bits set over the past and present window size – if less than 50% are set, the window is increased by 1 and if more than 50% are set, the window is multiplied by a factor less than 1 (0.875 is the recommended value)
  - In equilibrium, the window oscillates around the optimum size, and adapts to a new optimal point if the bottleneck rate or the RTT on the network change

Explicit notifications in Frame Relay

- A *Forward Explicit Congestion Notification* (FECN) bit is set on frames that cross a congested node towards the receiver – the use of this information by the receiver is left open, but could be coupled with window based flow control



- A *Backward Explicit Congestion Notification* (BECN) bit is set by a congested node on frames travelling in the reverse direction towards the source – it is up to the sender to decide how to adapt its rate based on this notification



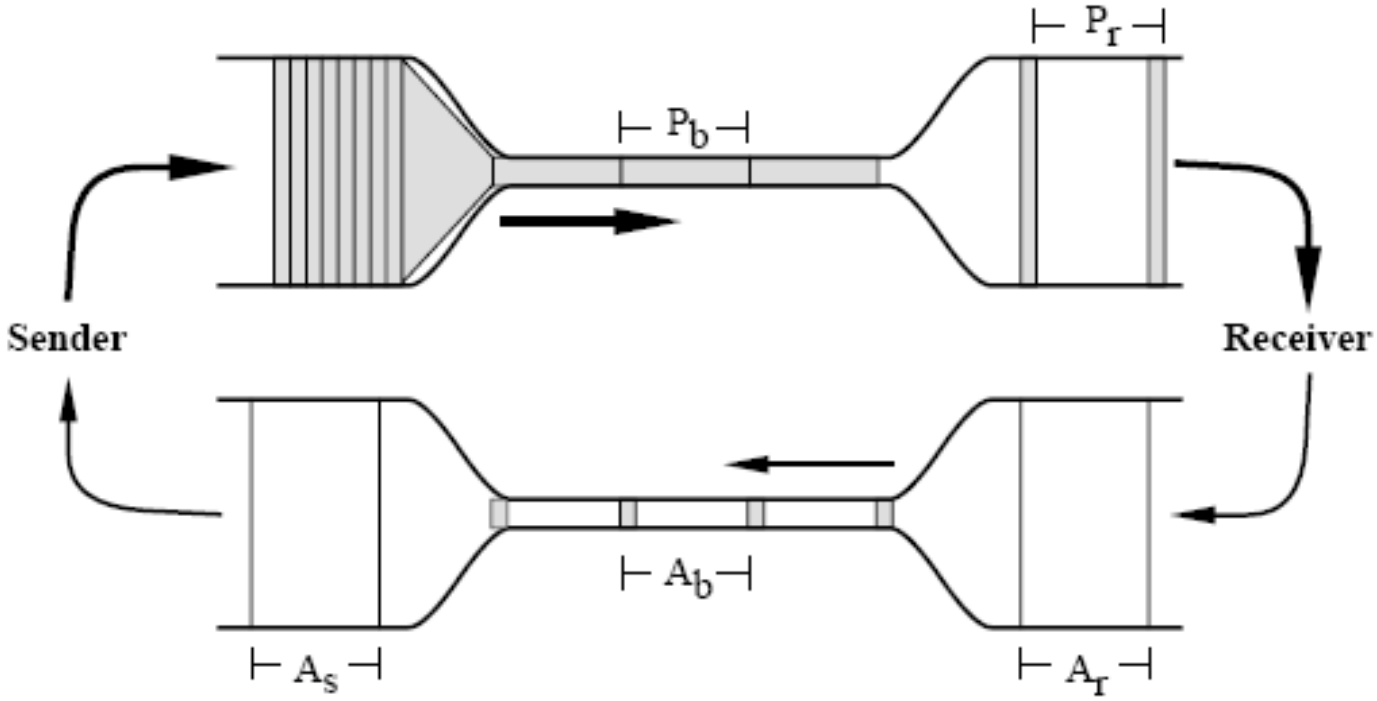
TCP – flow and congestion control windows

- TCP congestion control is performed end-to-end, based on a dynamic *sender window* and uses implicit information of the network state
- In addition, the sender must respect the window advertised by the receiver
  - For a given amount of packets in transit each ACK received means that a packet left the network and that a new packet can be sent (window rotation)
  - Since the pace at which ACKs are received (and thus new packets can be sent) is also determined by the network behaviour, TCP is said to be *self-clocking*
- However, the network dynamics is rather complex – the number of active sessions and available capacity changes over time and the network may become congested
- The basic idea of TCP congestion control is to let each source independently determine how much capacity it can use and adapt its rate to match network conditions, that is, a source must implement a dynamic *congestion window* that is based on the perceived level of congestion at the network
- The congestion window (*cwnd*) is used for flow control imposed by the sender, while the advertised window (*rwnd*) is used for flow control imposed by the receiver and thus the transmission window is the minimum of the two

$$\textit{current sender window} = \textit{min (cwnd, rwnd)}$$

TCP self-clocking

- P_b represents the minimum packet spacing on the slowest link (bottleneck) and thus, at the receiver, $P_r = P_b$
- Assuming the same processing time for all packets, the spacing between ACKs generated by the receiver is $A_r = P_r = P_b$, and considering similar conditions on the reverse path (and that ACKs occupy less resources than data packets), then $A_s = A_b = A_r = P_b$
- If, after an initial burst, packets are sent only in response to ACKs, the sender packet spacing will match the packet time on the slowest link



From Van Jacobson, “Congestion Avoidance and Control”, Proc. SIGCOMM’88, August 1988

TCP – control algorithms

- TCP implements four algorithms, which are usually combined in two groups – *slow start* and *congestion avoidance* on the one hand, and *fast retransmit* and *fast recovery* on the other

- The TCP sender maintains the state of the two windows (*cwnd* and *rwnd*) and the value of a *slow start threshold* (*ssthresh*), which allows deciding whether to use *slow start* (window below the threshold) or *congestion avoidance* (window above the threshold)

- The sender also keeps a variable *FlightSize*, which is the amount of data that has been sent but not yet acknowledged (its value is limited by the current window size)

- In the description of the algorithms and updating of variables it is also considered the *Sender Maximum Segment Size* (SMSS), which is the size of the largest segment that the sender can transmit

TCP slow start and congestion avoidance

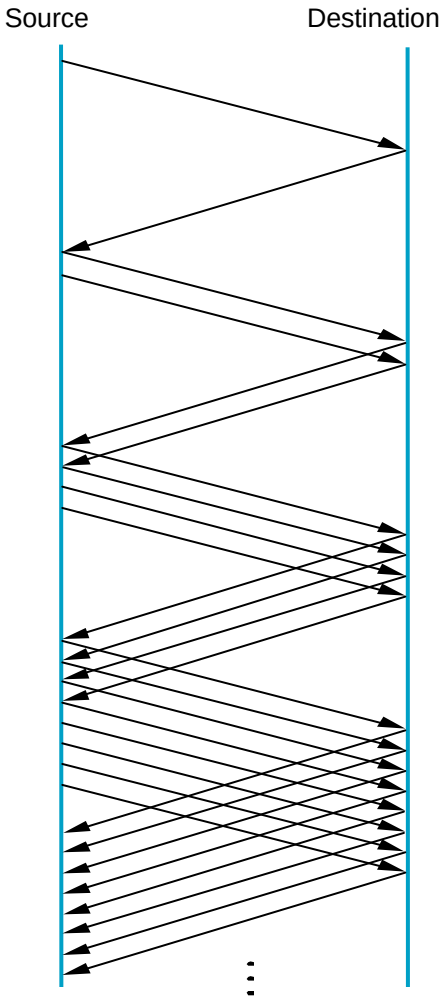
- TCP assumes that loss of segments is due to network congestion and the sender uses two indications of packet loss – time-out and receipt of duplicate ACKs, which trigger different recovery actions
- When starting a connection or after a time-out, *cwnd* is set to one segment (SMSS) and the sender enters the *slow start* mode
- In *slow start* mode, *cwnd* is incremented by at most SMSS bytes for each ACK received – this in fact means that *cwnd* is doubled every round trip time (*exponential increase*)
- When *cwnd* reaches *ssthresh*, the sender switches to the *congestion avoidance* mode and *cwnd* is incremented by one full-sized segment per round trip time (*linear increase*)
  - To achieve this, *cwnd* may be updated on every incoming non-duplicate ACK using the formula

$$cwnd = cwnd + SMSS * SMSS / cwnd$$

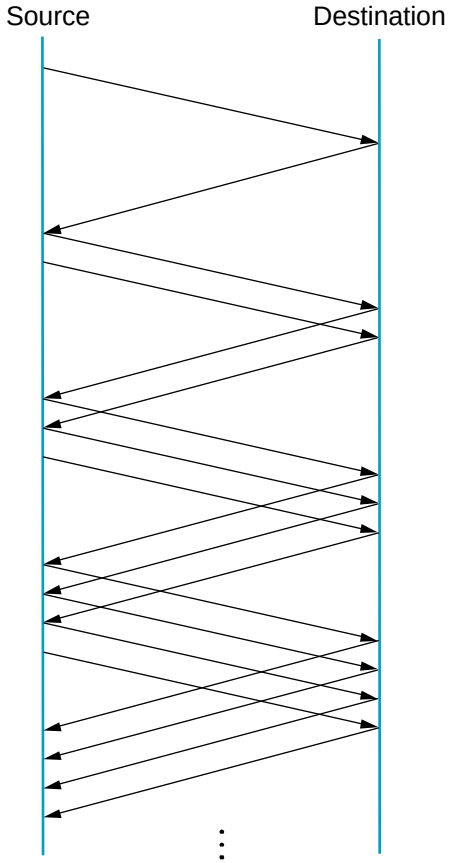
- *Congestion avoidance* continues until congestion is detected
  - By increasing *cwnd*, TCP is probing the network and at some time losses will start to occur and the congestion window will have to be reduced

Congestion window increase

- Exponential increase during *slow start* phase



- Linear increase during *congestion avoidance* phase



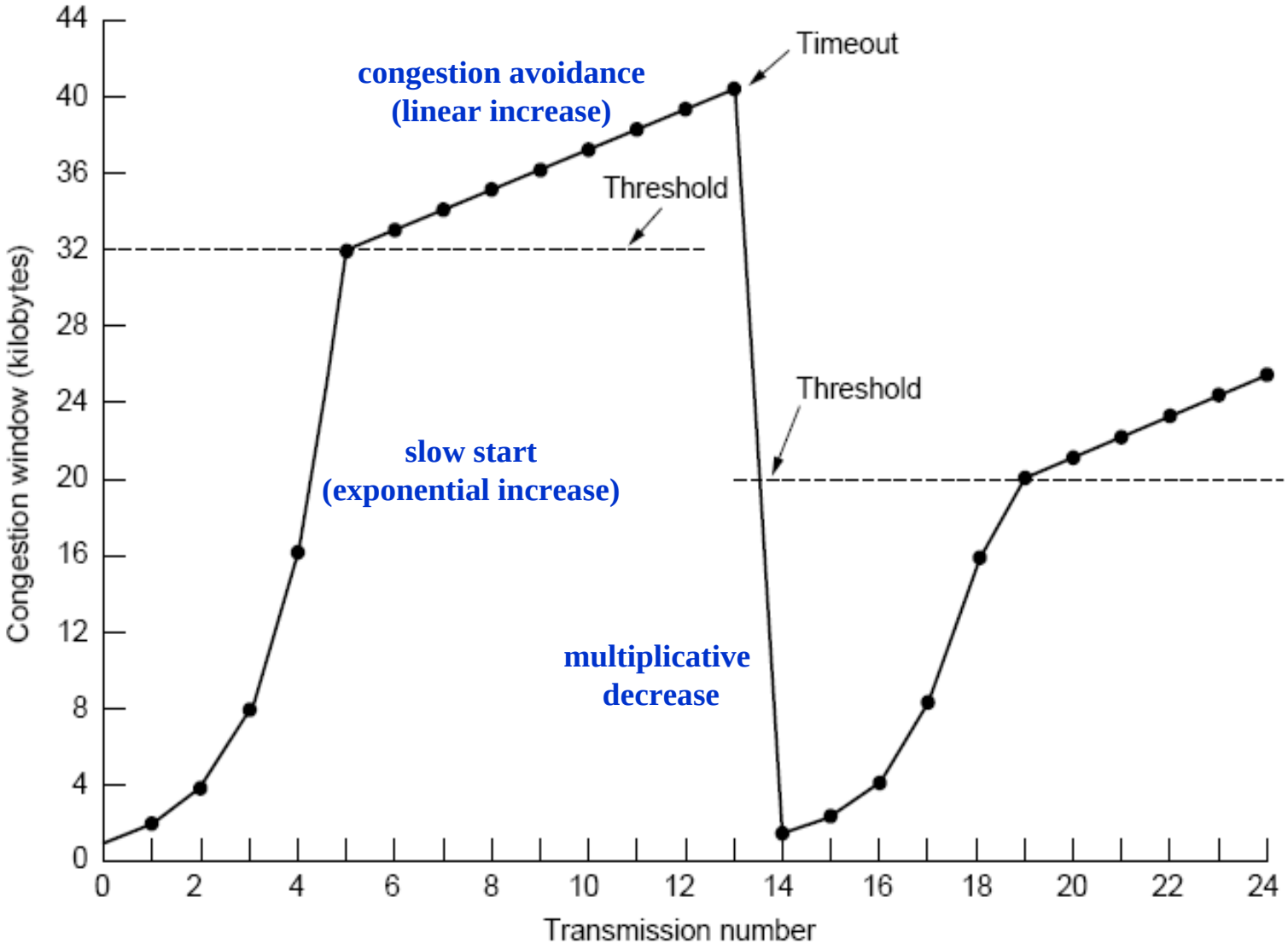
TCP congestion recovery

- A time-out or the receipt of duplicate ACKs are used as loss indications and thus as triggers for recovery
  - A TCP receiver must send a duplicate ACK when an out-of-order packet arrives
- If a time-out occurs, *ssthresh* is reduced to a value bounded by half of the current window size (*multiplicative decrease*), *cwnd* is set to one segment and the sender enters the *slow start* mode
  - Update of *ssthresh* is performed according to RFC 5681 (as per RFC 2581)

$$ssthresh = max\ (FlightSize\,/\,2,\ 2 * SMSS)$$

- The reception of three duplicate ACKs (without a time-out occurring) is a strong indication that a packet loss may have occurred but also that the receiver continued to receive additional packets – in this case TCP performs *fast retransmit* and *fast recovery* (and not *slow start*)
  - When the third duplicate ACK is received, *ssthresh* is also set to the value above
  - The lost segment is retransmitted and *cwnd* is set to *ssthresh* plus 3 * SMSS (the congestion window is inflated with the three segments that left the network)
  - For each additional duplicate ACK received, *cwnd* is incremented by SMSS (inflated with the additional segment that left the network) and a new segment is transmitted, if allowed by the current updated window – *min* (*cwnd*, *rwnd*)
  - When an ACK that really acknowledges new data is received, *cwnd* is set to the value of *ssthresh* updated at the start of recovery (thus deflating the window)

TCP – slow start and congestion avoidance



ATM end-to-end rate control scheme

- The ATM Forum defined an end-to-end flow control mechanism to be used with the ATM *Available Bit Rate* (ABR) service
  - This scheme allows sources to dynamically adapt their rate, based on explicit information sent by the network
- Control information sent by the network may be an *explicit rate* (ER), a *congestion indication* (CI) or both
- Depending on the information received, the source may keep its current rate, perform an *additive increase* or, in case of congestion indication, a *multiplicative decrease*
- An ABR source sends special *Resource Management* (RM) cells on a regular basis (typically once every 32 cells)
  - The RM cells carry the current *allowed cell rate* (ACR) and a maximum requested rate – *peak cell rate* (PCR)
  - The switches along the path process the RM cells and calculate the rate they can support, based on the current state
  - The explicit rate (ER) is carried on RM cells sent back by the receiver to the source on the opposite direction (ER is the minimum rate calculated by the switches and is determined by the bottleneck link on the path)

Open loop flow control

- Open-loop flow control is applied by a network to identifiable flows
  - Depending on the service model, the incoming flows may be treated by the network separately or as aggregates, based on a set of traffic classes
- A flow has to be admitted (accepted) before traffic can be sent to the network and thus the process involves an establishment and a data transmission phase
- During the establishment phase, an agreement (contract) must be negotiated between the user and the network
  - A source describes its behaviour by means of a set of traffic parameters (*traffic descriptor*) and specifies its QoS requirements by means of QoS parameters
  - The proposed values for the parameters are then negotiated
  - Based on an *admission control* mechanism, the network determines whether the new flow can be accepted, that is, whether it may commit resources to meet the negotiated QoS
  - In case the flow is accepted, the network reserves the appropriate resources
- Once a flow is accepted, the sender must regulate (shape) its traffic so that it complies with the contract
  - This is a form of flow control that is enforced solely by the sender, based on the values of traffic parameters that have been agreed with the network

Mechanisms for open loop flow control

- The whole process involves a set of complementary mechanisms
- In the first place, it is necessary to select the traffic parameters to be negotiated
  - The network prescribes a set of parameters and a source selects the values that best describe its traffic characteristics
- The traffic descriptors have three main purposes
  - They are the basis for the negotiation, which will ultimately lead to accepting or rejecting the flow – the network performs *admission control* and only accepts a flow in case the negotiated performance objectives can be met
  - The agreed values are used as input to a traffic regulator (*shaper*) at the source so that traffic sent to the network conforms with the contract (*traffic shaping*)
  - The same values are used as inputs to a *policer* at the network edge, which will accept compliant packets and will drop (or delay) non compliant packets (*traffic policing*), in order to avoid congestion or that misbehaving users may degrade the QoS provided to well-behaved users
- Network nodes must implement *scheduling* and *buffer management* disciplines to give the flows a differentiated treatment in order to meet performance objectives

Traffic descriptors

- Traffic descriptors must have a set of desirable properties
  - They must be representative, that is, they must describe the flow as accurately as possible, so that the resources reserved by the network match the requirements
  - They must be easily verifiable (e.g., for shaping or policing)
  - They must be usable, that is, the source must be able to describe traffic and the network to perform admission control in an easy and fast way
  - They must be preservable – the network must be able to preserve traffic characteristics along the path (to match the reserved resources) or to recalculate the needed resources if it modifies the traffic characteristics
- Different traffic descriptors have been proposed for use in real networks (such as ATM and IP)
- Three basic parameters are usually considered – *average rate*, *peak rate* and *maximum burst size* – and can be combined in different ways to form a traffic descriptor

Average rate

- To define the *average rate* of a source it is necessary to specify the period of time for the averaging process
  - For a given time period, the average rate is the amount of data (bits) generated by the source divided by its duration
- An average rate mechanism requires two parameters (a time window and the number of bits that can be sent in the window) – two commonly used mechanisms are the jumping window and the moving (sliding) window
- In a *jumping window*, the averaging process is performed over consecutive (non overlapping) time intervals (windows) and the process is reinitialized on each interval – the measured average rate is very sensitive to the choice of the starting time, as exemplified
  - A burst that overlaps two windows (end of a window and start of the next one) could be accepted (provided the average rates measured on each window were compliant), even if the same burst were not accepted in case it did occur within a single window with the same duration – and similarly with a set of smaller bursts (overlapping two windows or within a single window, respectively)
- In a *moving window*, the averaging process is performed over all time intervals with duration equal to the defined window and thus control is quite tight and independent of the starting time (but more complex to implement)

Peak rate

- Informally the *peak rate* is the maximum instantaneous rate of a connection (or flow)
- In networks with fixed size packets the peak rate is defined as the inverse of the minimum inter-packet spacing T (e.g., *peak cell rate* in ATM)
- In networks with variable size packets, the peak rate must be specified for a time window over which it should be measured – that is, the peak rate is the highest average rate over all intervals of the specified duration
  - The choice of the window must be such that it gives a useful meaning to the peak rate and allows distinguishing it from the average rate
- The peak rate is very sensitive to even small deviations in traffic patterns, which is easily exemplified in the case of ATM
  - Violation of the peak cell rate definition (that is, the observation of an inter-cell spacing less than T) may easily occur
    - Due to asynchronous time division multiplexing in the source or the access network, delay jitter is introduced and thus the temporal relation between cells is changed (cell clumping may occur) at the input of the core network
    - Peak cell rates are not in general submultiples of the link rate – the nominal inter-cell spacing T that corresponds to a given peak rate is not a multiple of the cell time, and thus for some cells the observed inter-cell spacing will be less than T
  - The compliance of the ATM peak cell rate is thus coupled with a *cell delay variation tolerance*

Maximum burst size

- In order to control the *average rate* of a flow in a meaningful way (from the point of view of resource allocation), the averaging period should not be too short (in practice, average and peak rate would not be distinguished) nor too long, since in this case control would be too loose (it would not be possible to tightly control the duration of bursts)
  - The long term average rate will always be bounded by the average rate controlled over shorter intervals (but the reverse is not true)
- One way of achieving a tighter control over the average rate is to control the duration of bursts that may be sent at the peak rate
- A *linear bounded arrival process* (LBAP) limits the number of bits transmitted in any interval t to

$$r * t + b$$

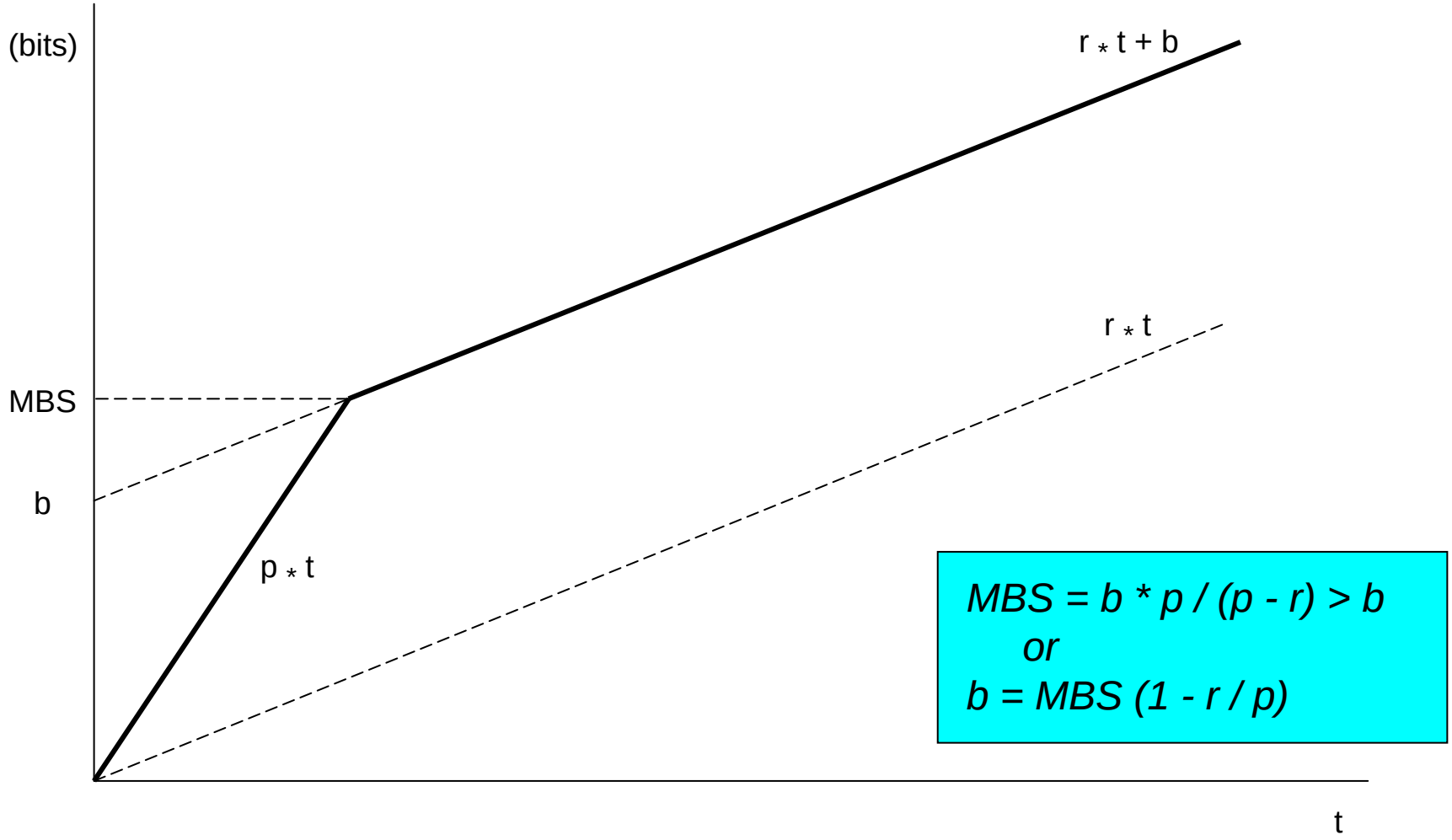
  where r is the average rate to control and b is related (but not equal) to the *maximum burst size* (MBS)
- Assuming that p is the peak rate (and considering a fluid model), then

$$MBS = b * p / (p - r) > b$$

$$b = MBS\ (1 - r / p)$$

- These traffic parameters are usually regulated by a *Leaky Bucket* (p) and a *Token Bucket* (r, b)

LBAP – bounds for a traffic flow (p, r, b)



Traffic Parameters – ATM and IP

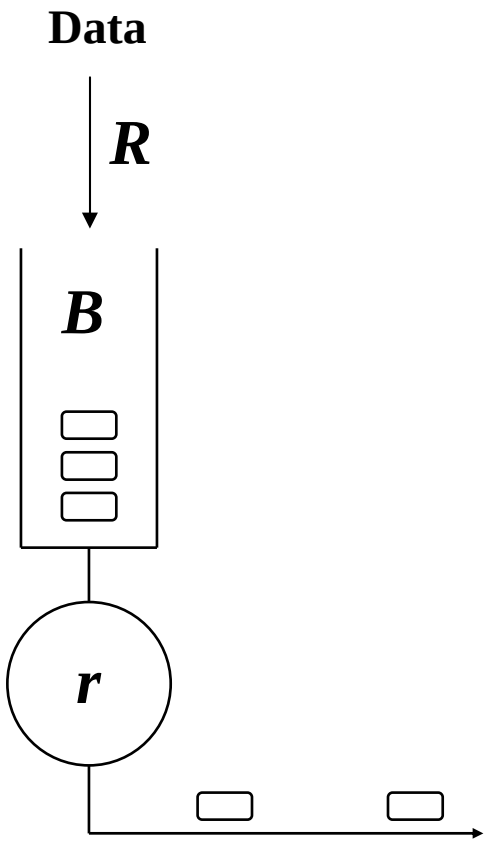
- ATM traffic parameters
  - Traffic parameters are used when negotiating ATM service categories (e.g., Constant Bit Rate, real-time and non real-time Variable Bit Rate)
  - Main parameters
    - PCR – *Peak Cell Rate*
    - CDVT – *Cell Delay Variation Tolerance*
    - SCR – *Sustainable Cell Rate*
    - MBS – *Maximum Burst Size*
- IP FlowSpec
  - Contains information that characterizes the traffic to be submitted to the network (TSpec) and the required service with associated QoS (RSpec)
  - TSpec
    - *p – peak rate*
    - *r – token bucket rate*
    - *b – bucket size*
    - *M – maximum datagram size*
    - *m – minimum policed unit*

Leaky Bucket

- A *Leaky Bucket* controls the maximum instantaneous rate (*leak rate*) *r* of a flow – its operation is easily described in the case of fixed size packets (e.g., ATM cells), but can be generalized
  - It accepts bursty traffic on its input but eliminates bursts on the output
  - The maximum accepted burst size, without losses, depends on the buffer size *B* and the input rate *R*

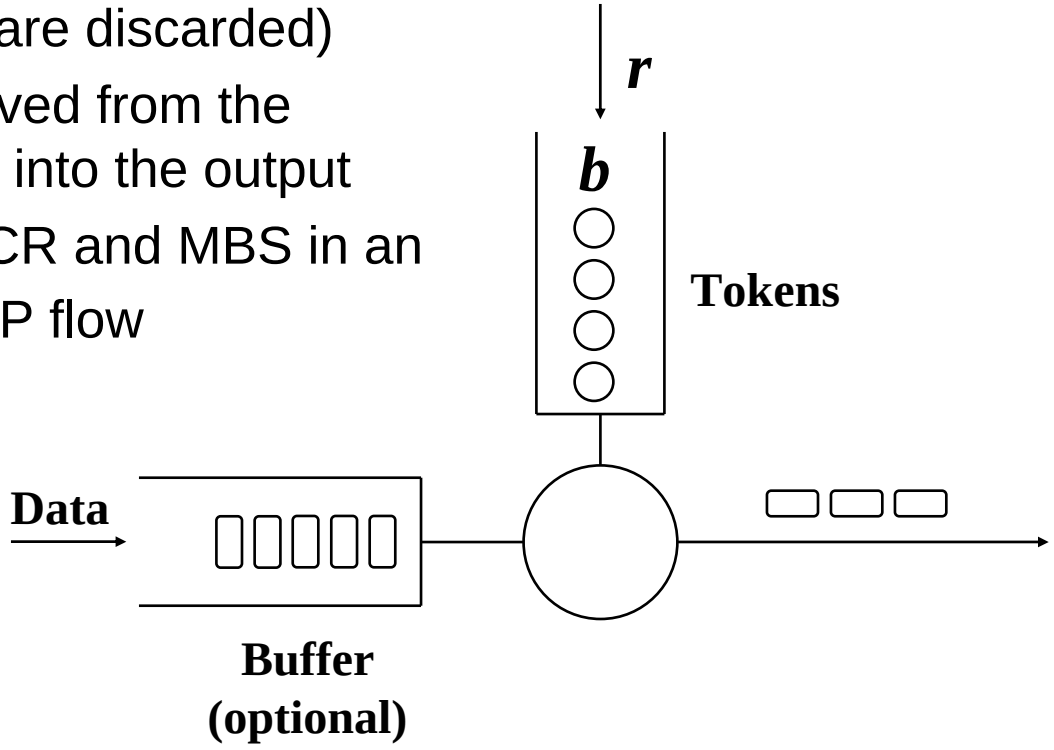
$$MBS = B * R / (R - r) \qquad \text{for } R > r$$

  - While the buffer is not empty, the output flow is periodic (T = 1 / *r*), with rate *r* (isochronous shaper)
- A Leaky Bucket may control PCR in an ATM connection or *p* in an IP flow

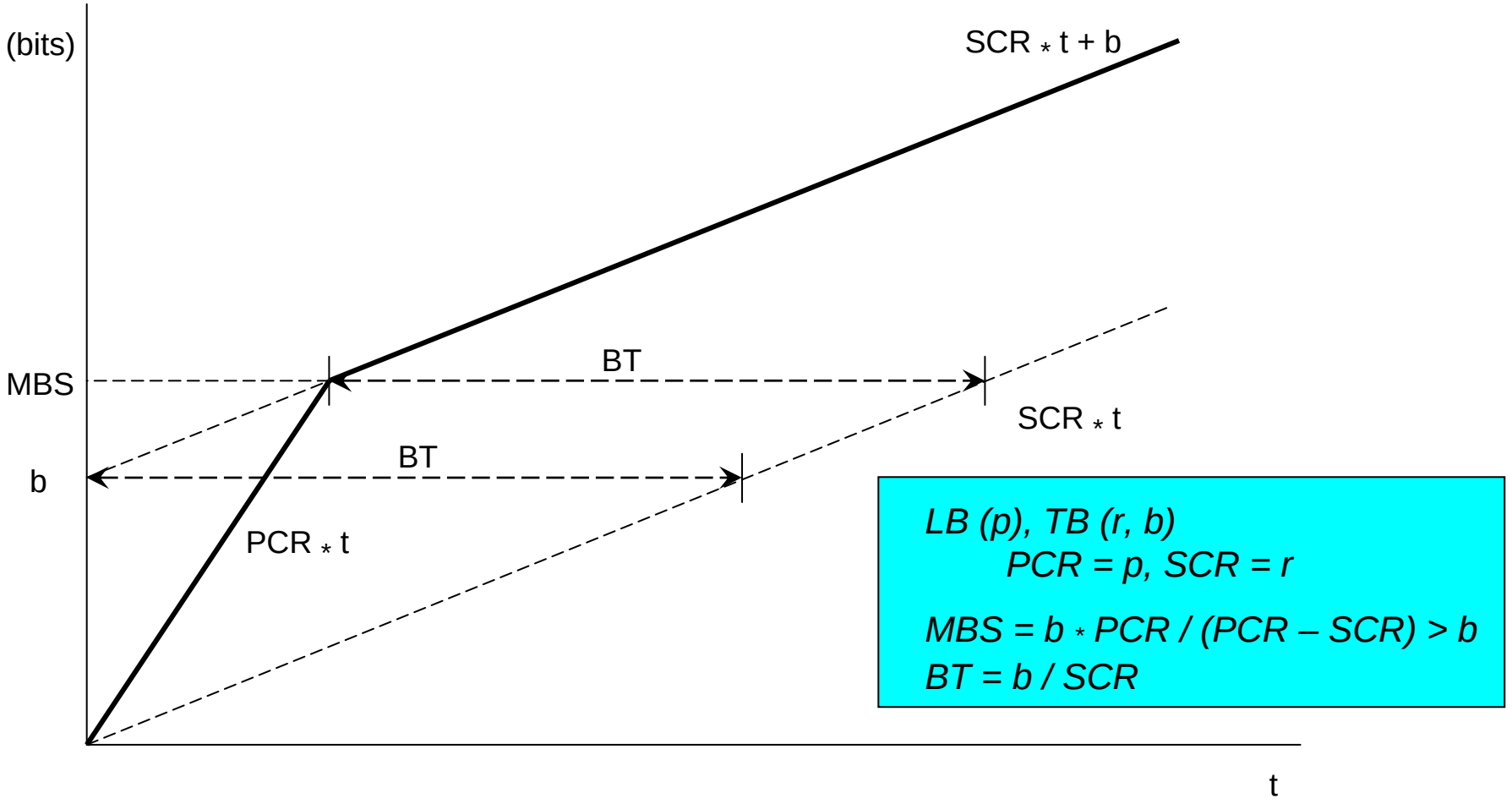


Token Bucket

- A *Token Bucket* controls the average output rate *r* of a bursty flow
  - It allows bursts with an instantaneous rate higher than the average rate, but limits their size
  - The bucket is a container with a capacity of *b* tokens, which are generated with a rate *r* equal to the average rate to be controlled (when the bucket is full, newly generated tokens are discarded)
  - Tokens must be claimed (removed from the bucket) before traffic is allowed into the output
  - A Token Bucket may control SCR and MBS in an ATM connection or (*r*, *b*) in an IP flow



Example – control of ATM traffic parameters



- It is possible to conclude that a Token Bucket (*r*, *b*) is able to control the *Sustainable Cell Rate* (SCR = *r*) and a *Burst Tolerance* (BT = *b* / SCR = *b* / *r*)
- The *Burst Tolerance* only depends on the Token Bucket parameters; it is independent of PCR and MBS and thus applies to different combinations of (PCR, MBS)

Traffic Parameters – Frame Relay

- In Frame Relay a user can negotiate a *Committed Information Rate* (CIR) and a *Committed Burst Size* (CBS)
  - CIR is an average rate that the network guarantees under normal conditions
  - CBS is the maximum amount of information that the network accepts from a user, under normal conditions, during a period T = CBS / CIR
- Frames are transmitted by the user at the link *Access Rate* (AR) and bursts are allowed up to CBS
  - CIR (and CBS) cannot be exceeded on any window of duration T (moving window) – this is more complex to implement than a token bucket
- In fact it is also possible to negotiate an *Excess Burst Size* (EBS)
  - When the amount of information within any window T exceeds CBS, the corresponding frames are marked as eligible for discard provided that CBS + EBS is not exceeded (otherwise, non compliant frames are simply discarded)

Frame Relay window control (CIR, CBS, T)

- The figure shows three compliant frames transmitted at the Access Rate (AR)
  - CBS and CIR are not exceeded during T = CBS / CIR
    - Other patterns would be possible (e.g., the same three frames contiguously sent)