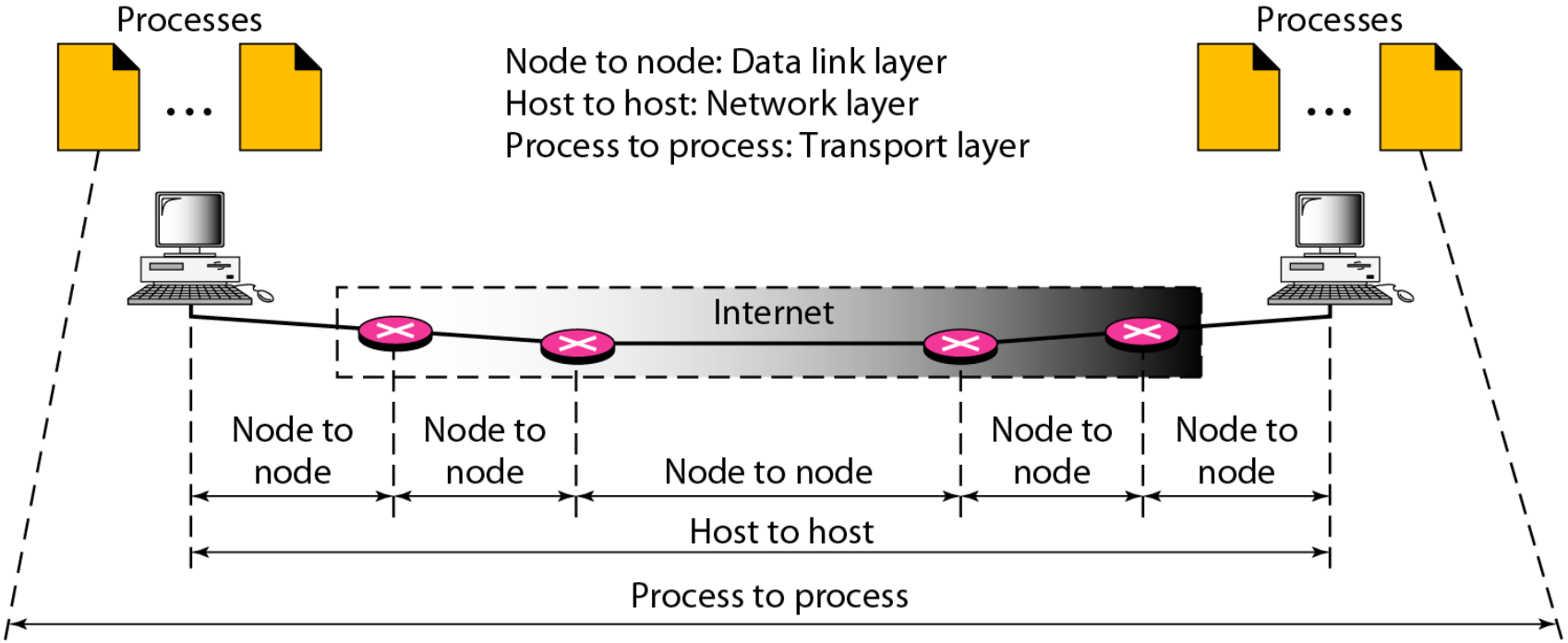
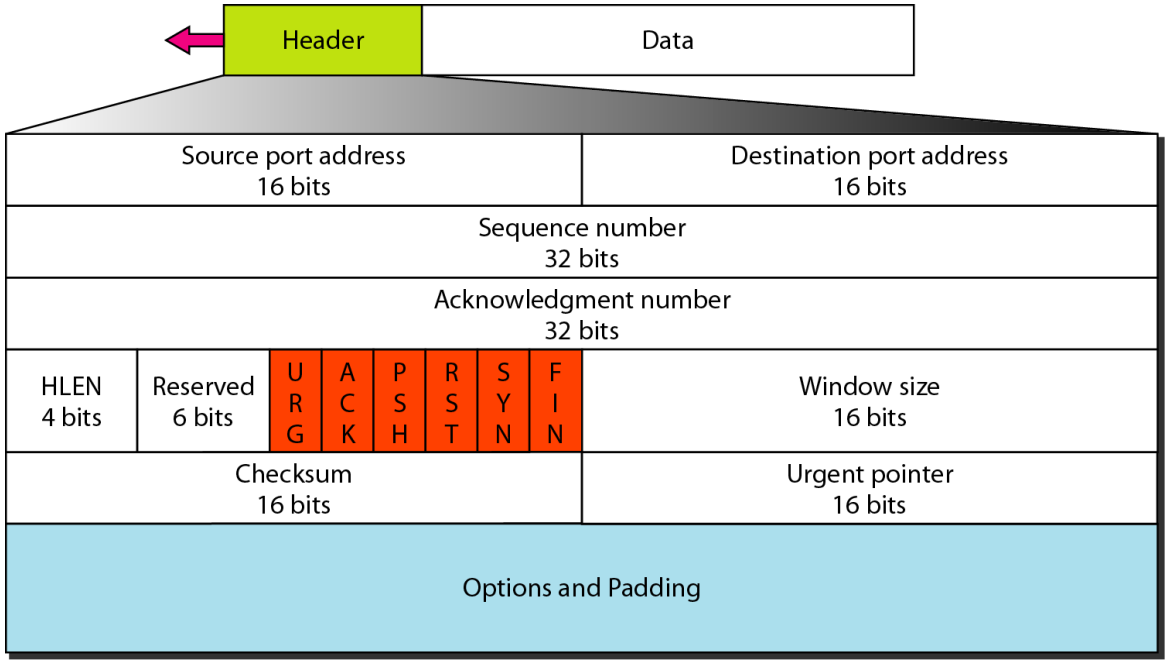
Congestion Control and Quality of Service


Processes
Processes
Node to node: Data link layer
Host to host: Network layer
Process to process: Transport layer
Internet
Node to node
Node to node
Node to node
Node to node
Node to node
Host to host
Process to process

TCP segment format

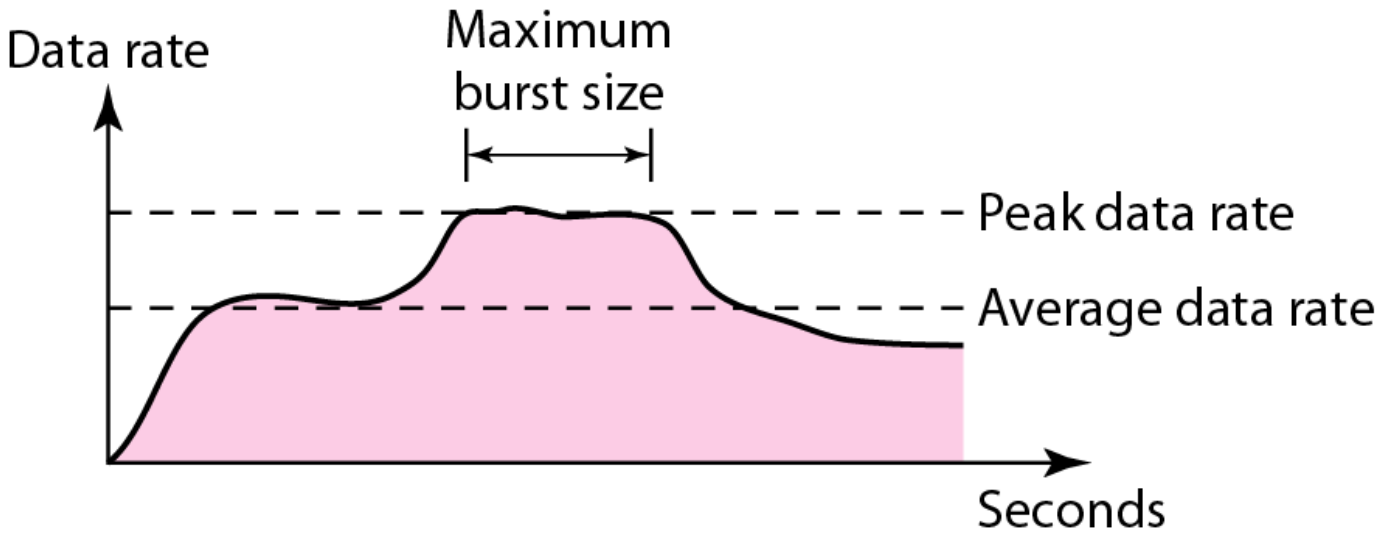


- A router may receive data from more than one sender.
- Even if the large buffer size it may be overwhelmed with data send by the sender and hence dropping of some segments happen.
- More segments loss means retransmission of the segments that worsening the congestion and collapse the communication.

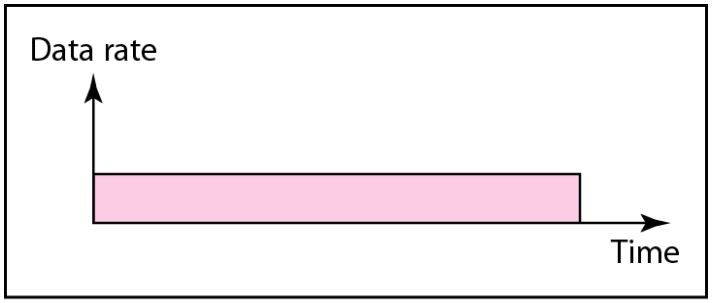
Data Traffic

- *The main focus of congestion control and quality of service is data traffic.*
- *In congestion control we try to avoid traffic congestion.*
- *In quality of service, we try to create an appropriate environment for the traffic.*

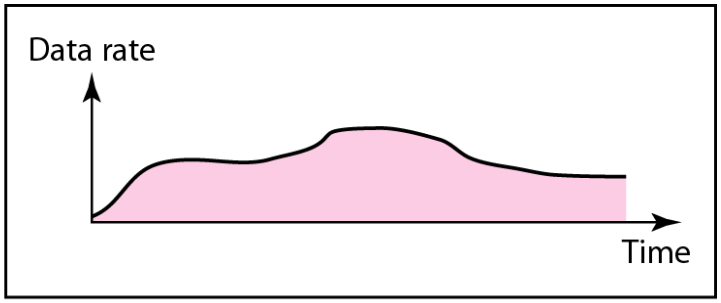
Traffic descriptors



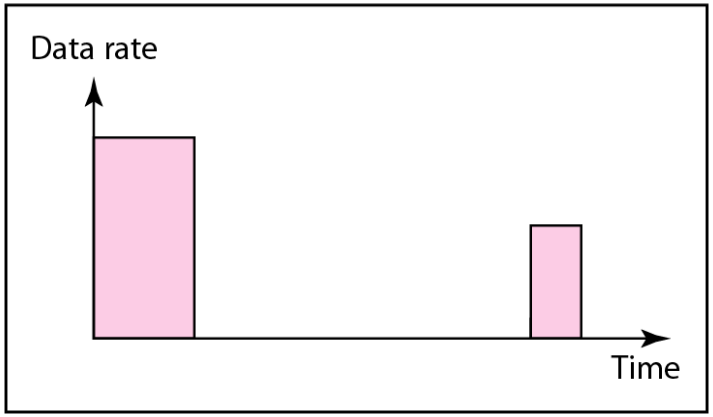
Three traffic profiles



a. Constant bit rate



b. Variable bit rate

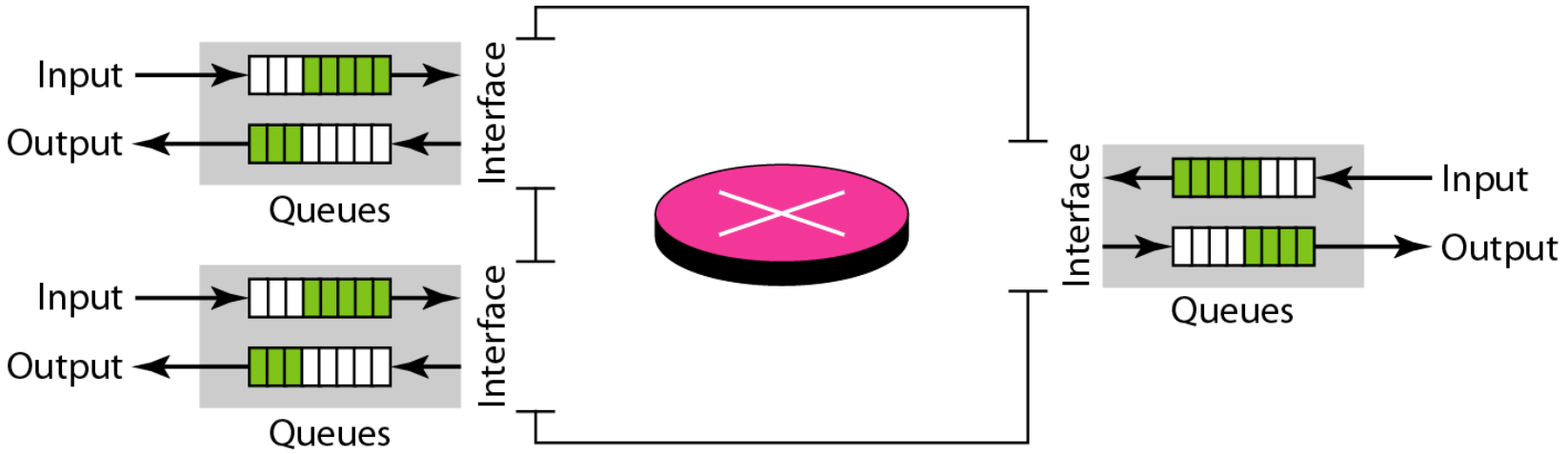


c. Bursty

Congestion

- *Congestion in a network may occur if the load on the network*
  - *The number of packets sent to the network is greater than the capacity of the network, the number of packets a network can handle.*
- *Congestion control refers*
  - *to the mechanisms and techniques to control the congestion and*
  - *keep the load below the capacity.*

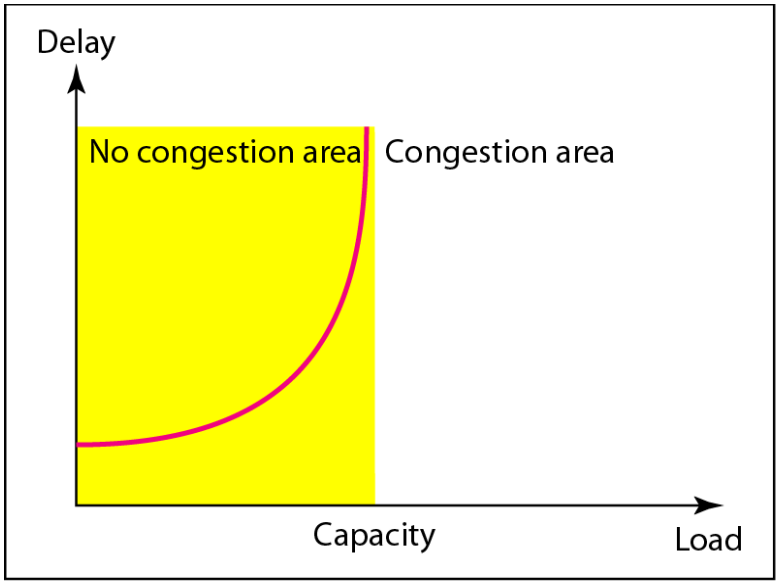
Queues in a router



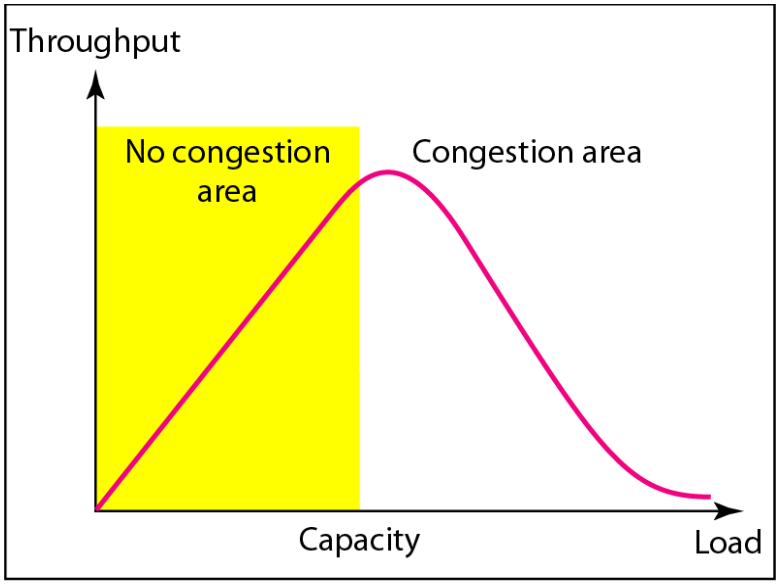
- First, if the *rate of packet arrival is higher* than the packet processing rate, the input queues become longer and longer.

- Second, if the *packet departure rate is less* than the packet processing rate, the output queues become longer and longer.

Packet delay and throughput as functions of load



a. Delay as a function of load



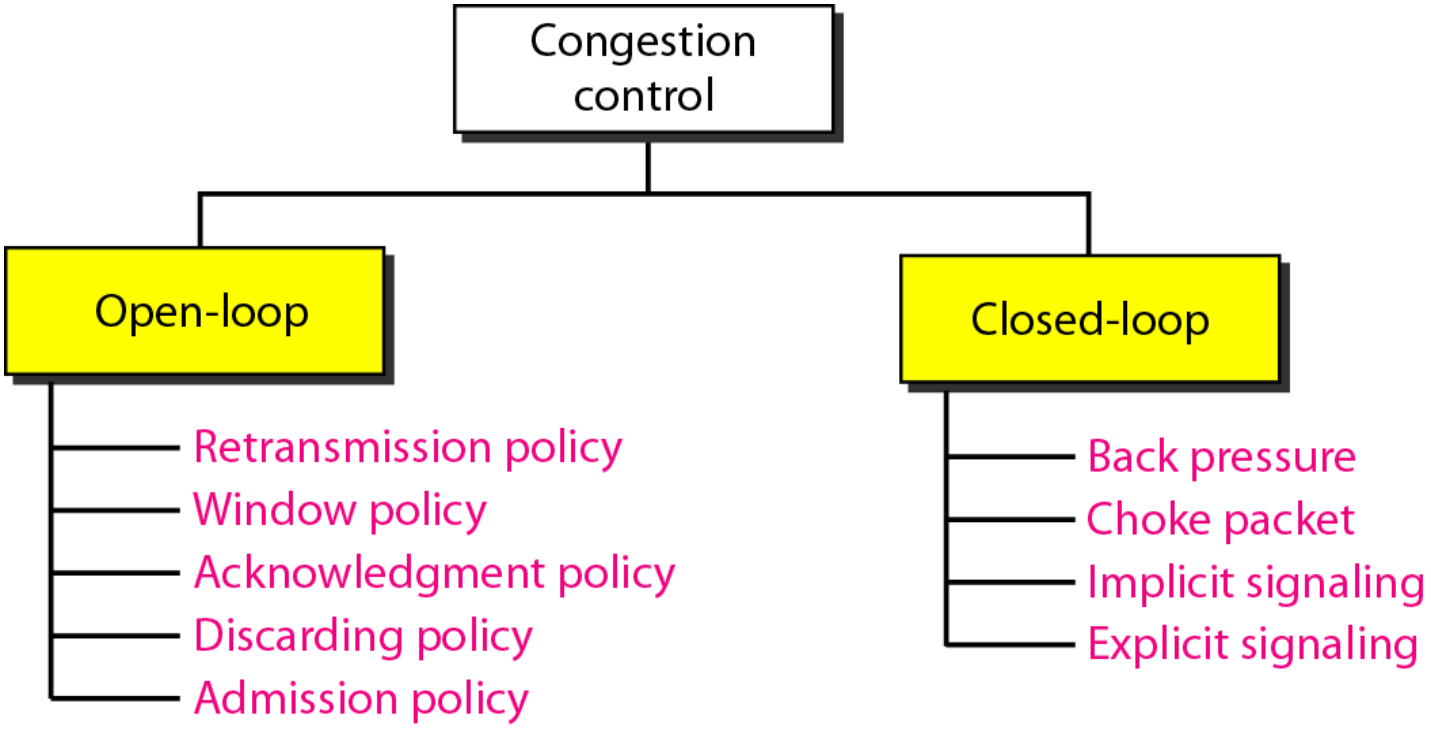
b. Throughput as a function of load

- **Congestion control involves two factors that measure the performance of a network:**
  - ***delay* and *throughput*.**

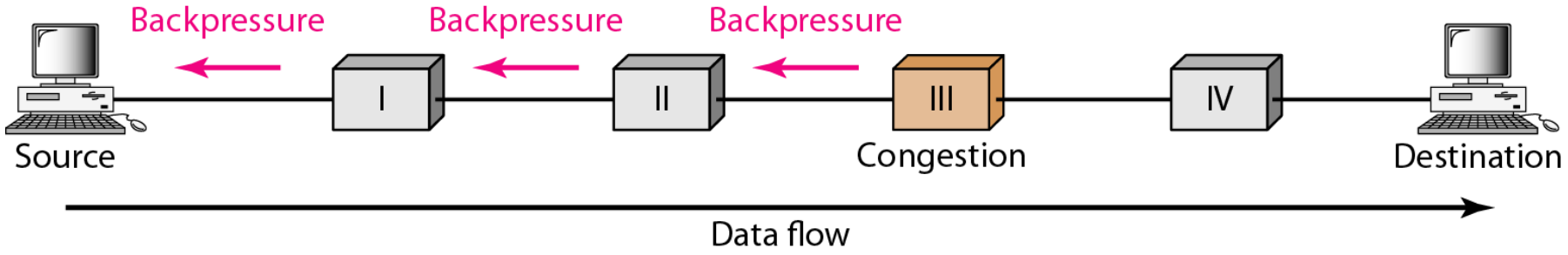
Congestion Control

- Congestion control refers to techniques and mechanisms that can either ***prevent congestion,*** before it happens, or ***remove congestion***, after it has happened.
- In general, we can divide congestion control mechanisms into two broad categories:
  - ***Open-loop congestion control (prevention) and***
  - ***Closed-loop congestion control (removal).***

Congestion control categories

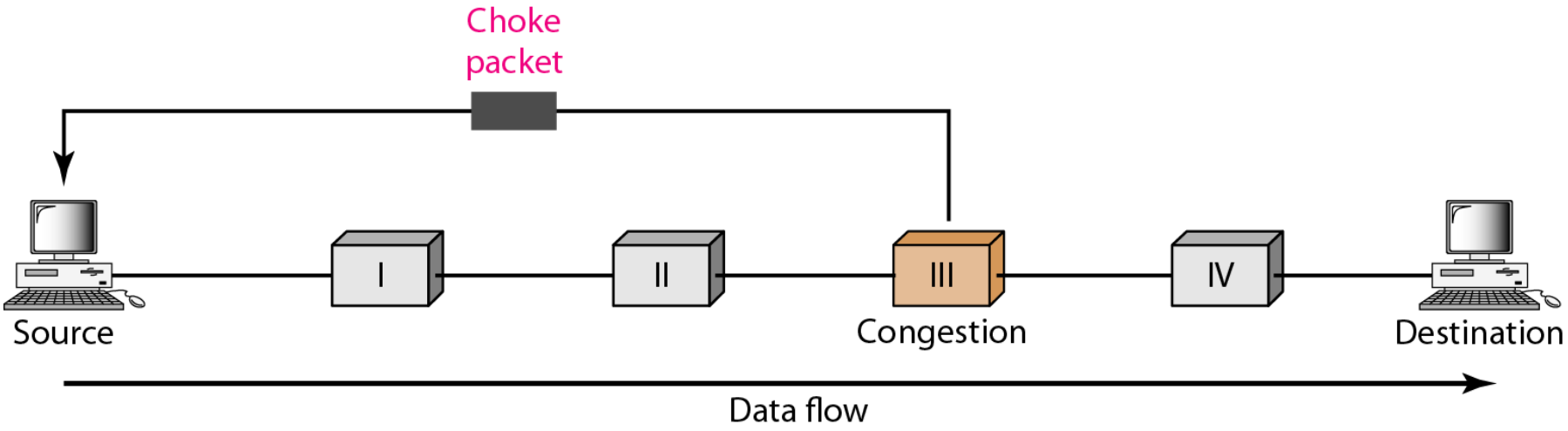


Backpressure method for alleviating congestion



- In this technique a congested node stops receiving data from the immediate upstream node or nodes.
- This may cause the upstream node or nodes to become congested, and they, in turn, reject data from their upstream nodes or nodes.
- The backpressure technique can be applied only to virtual circuit networks, in which each node knows the upstream node from which a flow of data is corning.

Choke packet



- A choke packet is a packet sent by a node to the source to inform it of congestion.
- In the choke packet method, the warning is from the router, which has encountered congestion, to the source station directly.
- The intermediate nodes through which the packet has travelled are not warned.

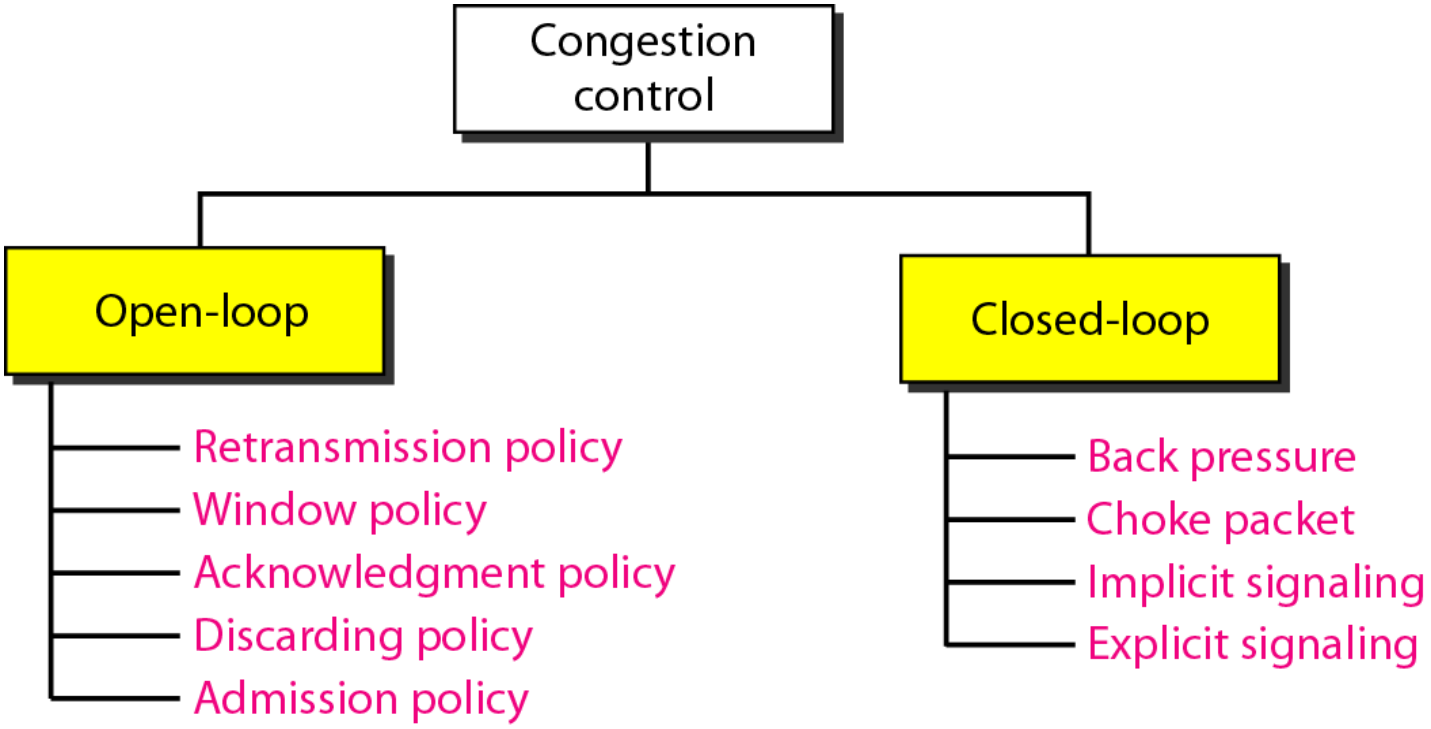
Implicit Signaling

- In implicit signaling, there is no communication between the congested node or nodes and the source.
- The *source guesses that there is a congestion* somewhere in the network from other symptoms.
- The delay in receiving an acknowledgment is interpreted as congestion in the network
  - The source should slow down.

Explicit Signaling

- The node that experiences congestion can *explicitly send a signal to the source* or destination.
- The explicit signaling method, however, is different from the choke packet method.
  - In the choke packet method, *a separate packet is used for this purpose*
  - In the explicit signaling method, *the signal is included in the packets that carry data.*

Congestion control categories



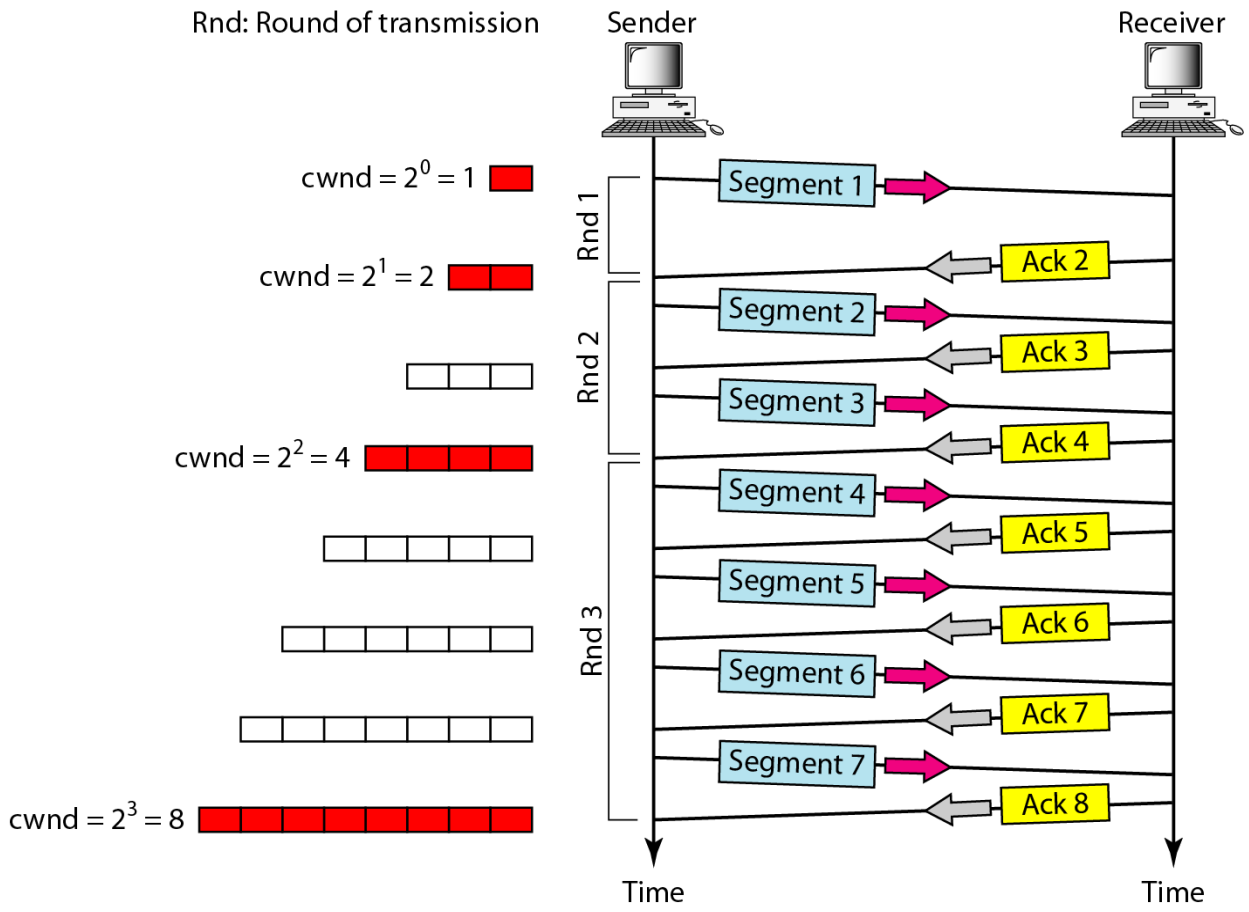
Congestion window and receive window

- To control the number of segments to transmit TCP uses congestion window(***cwnd***) and receiver window (***rwnd***)
  - *cwnd* size is controlled by the sender
  - *rwnd* determines the available buffer space in the receiver
    - ***rwnd = buffer size – number of waiting bytes to be pulled***
- The ***cwnd* and *rwnd*** variable together define the size of the send window in TCP
- Actual size of the window = **min*(cwnd, rwnd)***

Congestion Policy: Prevention

- Based on three phases:
  - Slow start
  - Congestion avoidance and
  - Congestion detection.
- In the slow-start phase, the sender starts with a very slow rate of transmission, but increases the rate rapidly to ***reach a threshold.***
- When the threshold is reached, the data rate is reduced to avoid congestion.
- Finally if congestion is detected, the sender goes back to the slow-start or congestion avoidance phase based on how the congestion is detected.

Slow start, exponential increase



- This algorithm is based on the idea that the size of the congestion window *(cwnd)* starts with one maximum segment size (MSS).

Slow Start

- Slow start cannot continue indefinitely. There must be a threshold to stop this phase.
- The sender keeps track of a variable named ***ssthresh (slow-start threshold).*** When the size of window in bytes reaches this threshold, slow start stops and the next phase starts.
- In most implementations the value of ***ssthresh*** is 65,535 bytes.

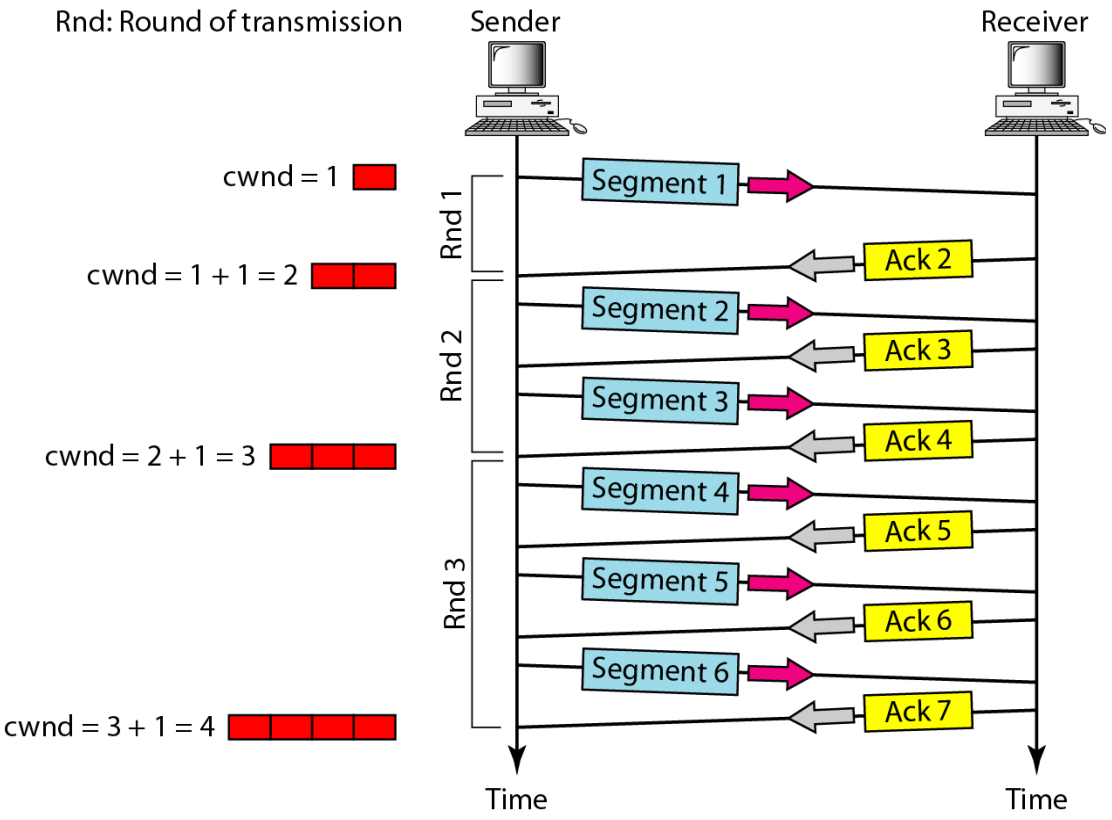
Congestion Avoidance

- To avoid congestion before it happens, one must slow down this exponential growth.
- TCP defines another algorithm called congestion avoidance, which undergoes an additive increase instead of an exponential one.
- When the size of the congestion window reaches the slow-start threshold, the slow-start phase stops and the additive phase begins.

Congestion Avoidance

- In this algorithm, each time the whole window of segments is acknowledged (one round), the size of the congestion window is increased by 1.

Congestion avoidance, additive increase



Congestion detection: Multiplicative decrease

- If congestion occurs, the congestion window size must be decreased.
- The only way the sender can guess that congestion has occurred is by the need to retransmit a segment.
- However, retransmission can occur in one of two cases:
  - when *a timer times out* or
  - when *three ACKs* are received.

Time-Out Phase

- If a time-out occurs, there is a stronger possibility of congestion
- A segment has probably been dropped in the network, and there is no news about the sent segments.
- In this case TCP reacts strongly
  - It sets the value of the threshold to one-half of the current window size *cwnd*.
  - It sets *cwnd* to the size of one segment.
  - It starts the slow-start phase again.

Three ACKs are received

- There is a weaker possibility of congestion
- A segment may have been dropped, but some segments after that may have arrived safely since *three ACKs* are received.
- This is called fast transmission and fast recovery.

Three ACKs are received

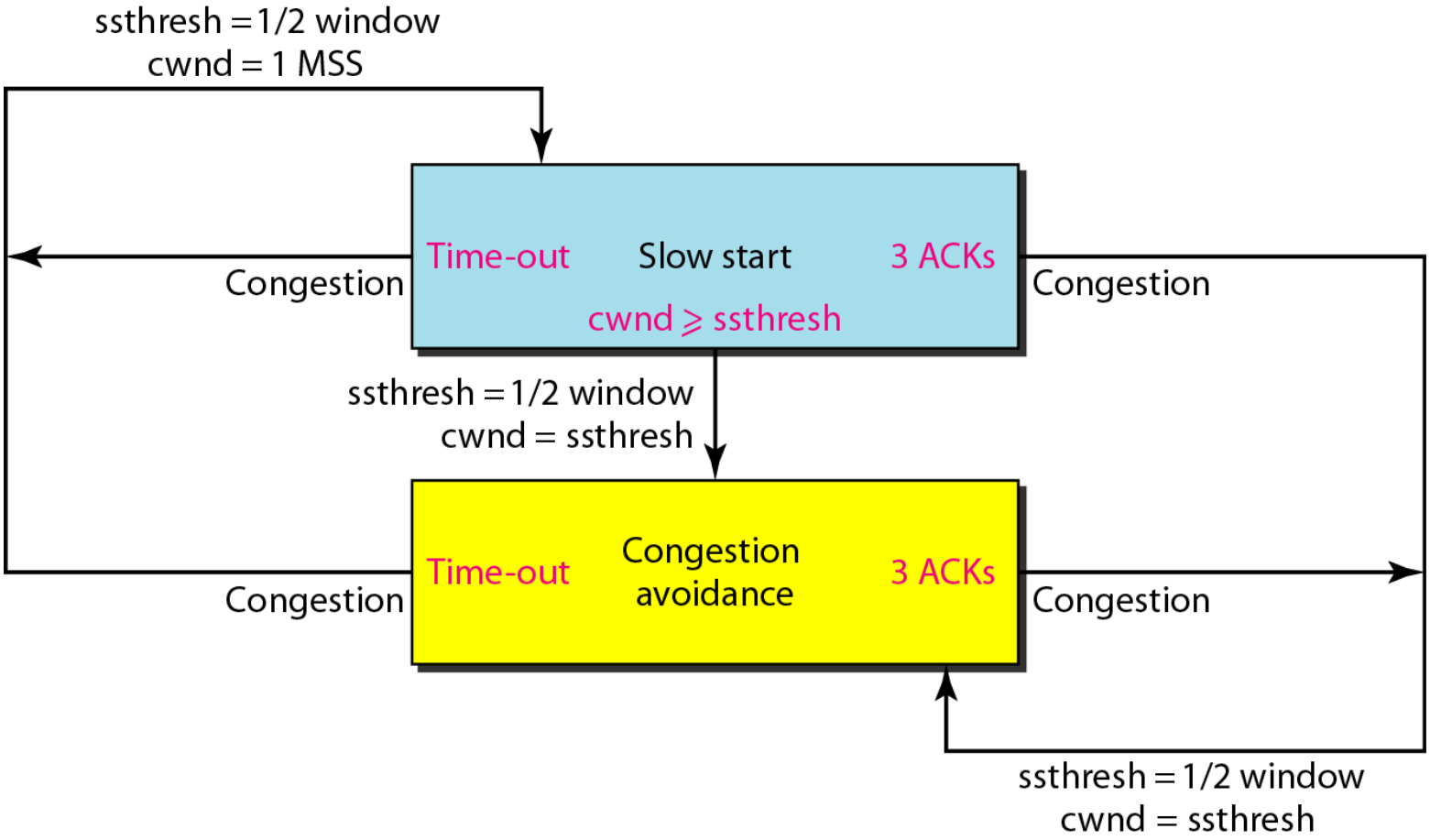
- In this case, TCP has a weaker reaction:
  - It sets the value of the threshold to one-half of the current window size
  - It sets *cwnd* to the value of the threshold (some implementations add three Segment sizes to the threshold)
  - It starts the congestion avoidance phase.

Note

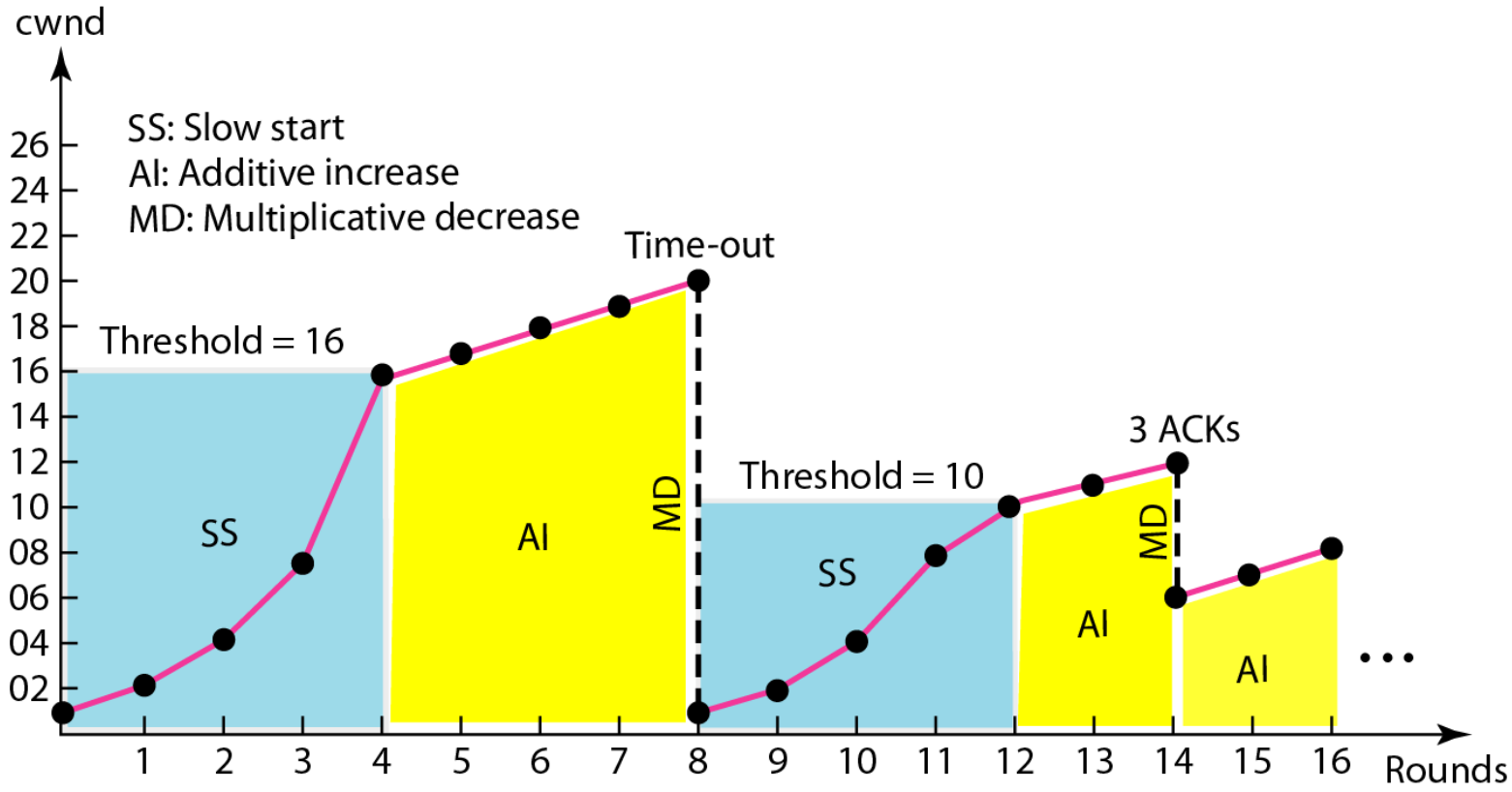
An implementation reacts to congestion detection in one of the following ways:

- **If detection is by time-out, a new slow start phase starts.**
- **If detection is by three ACKs, a new congestion avoidance phase starts.**

TCP congestion policy summary



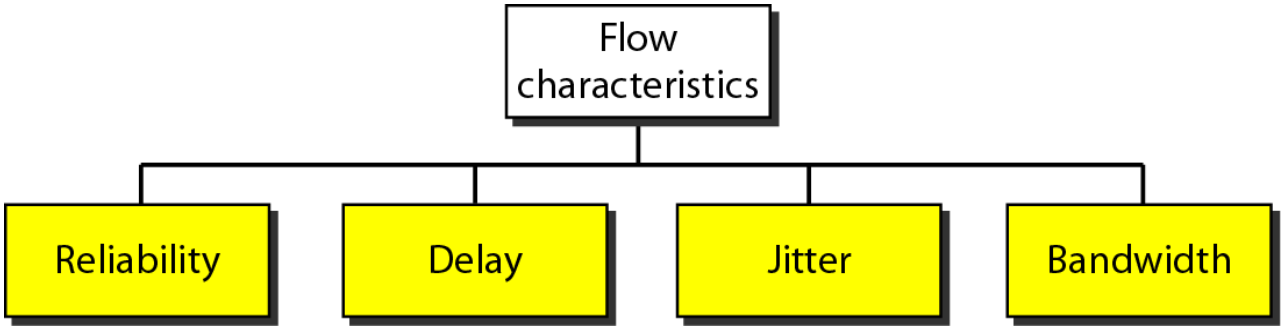
Congestion example



Quality of Service (QoS)

- ***Quality of service (QoS) is an internetworking issue that has been discussed more than defined.***
- ***We can informally define quality of service as something a flow seeks to attain.***

Flow characteristics

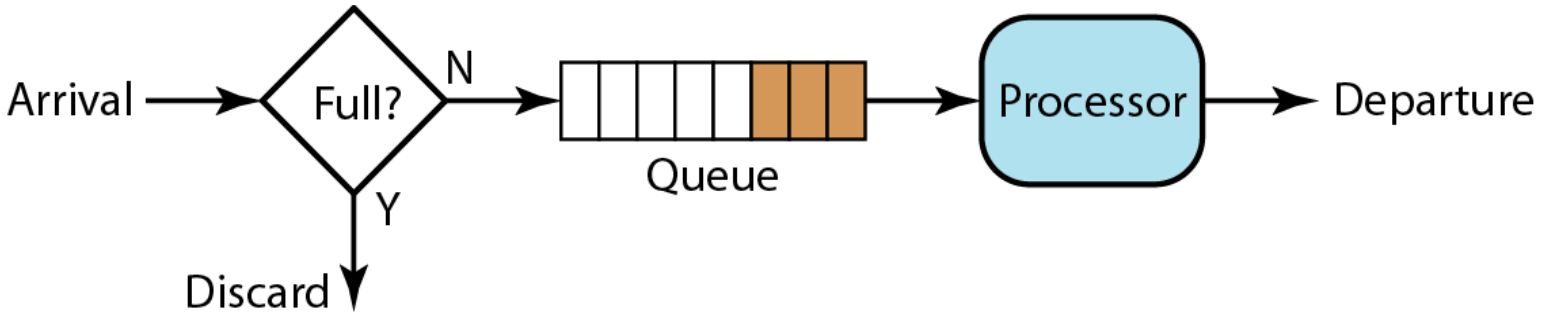


- **Jitter:** It is defined as the variation in the packet delay. High jitter means the difference between delays is large; low jitter means the variation is small.
  - For example:
    - If four packets depart at times 0, 1, 2, 3 and arrive at 20, 21, 22, 23, all have the same delay, 20 units of time.
    - On the other hand, if the above four packets arrive at 21, 23, 21, and 28, they will have different delays: 21,22, 19, and 24.
    - **For applications such as audio and video, the first case is completely acceptable; the second case is not.**

TECHNIQUES TO IMPROVE QoS

- ***Four common methods:***
  - ***scheduling***
  - ***traffic shaping***
  - ***admission control***
  - ***resource reservation.***

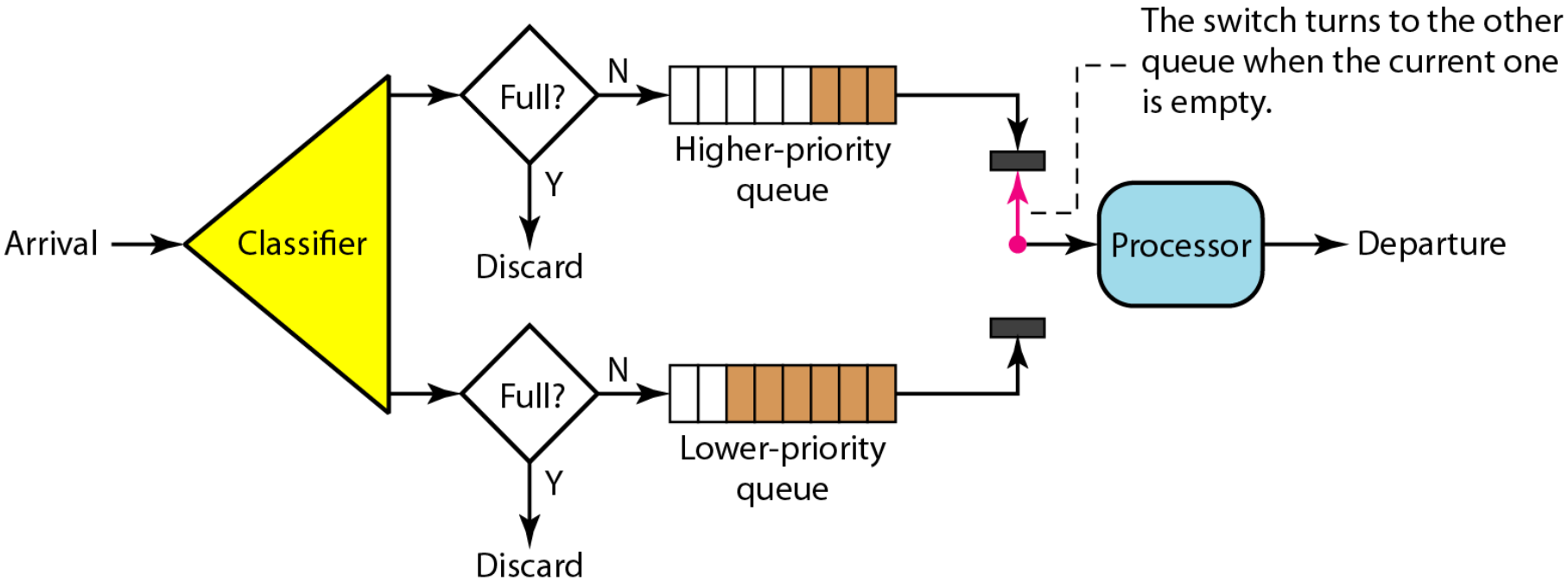
Scheduling: FIFO queue



- In first-in, first-out (FIFO) queuing, packets wait in a buffer (queue) until the node (router or switch) is ready to process them.

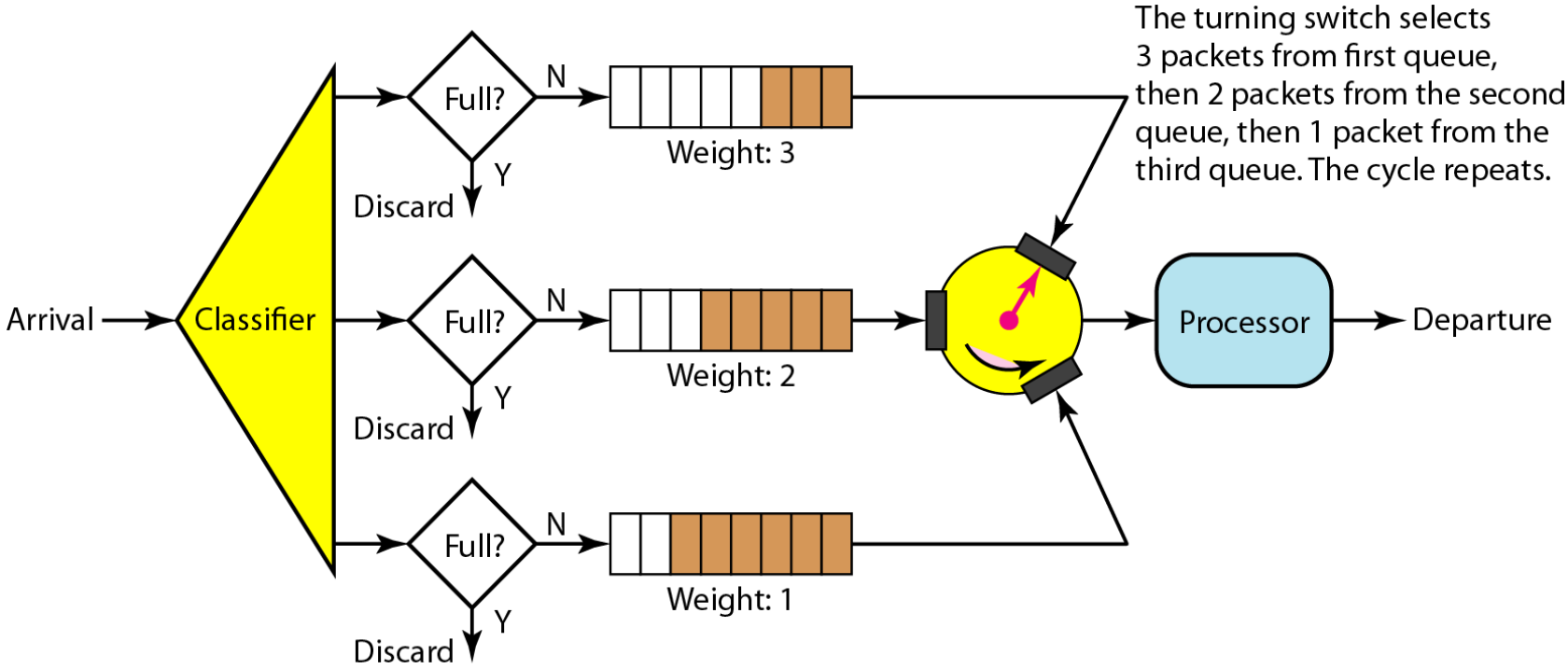
- If the average arrival rate is higher than the average processing rate, the queue will fill up and new packets will be discarded.

Scheduling: Priority queuing



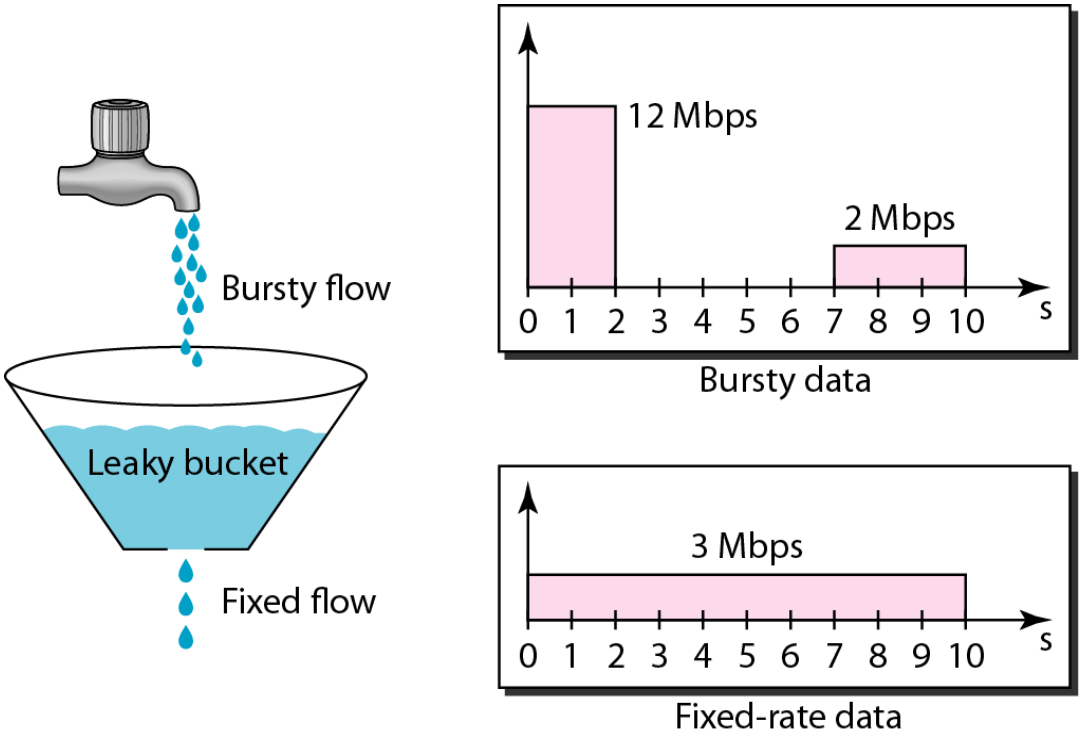
- In priority queuing, packets are first assigned to a priority class. Each priority class has its own queue.
- The packets in the highest-priority queue are processed first. Packets in the lowest-priority queue are processed last.
- Note that the system does not stop serving a queue until it is empty.

Weighted fair queuing



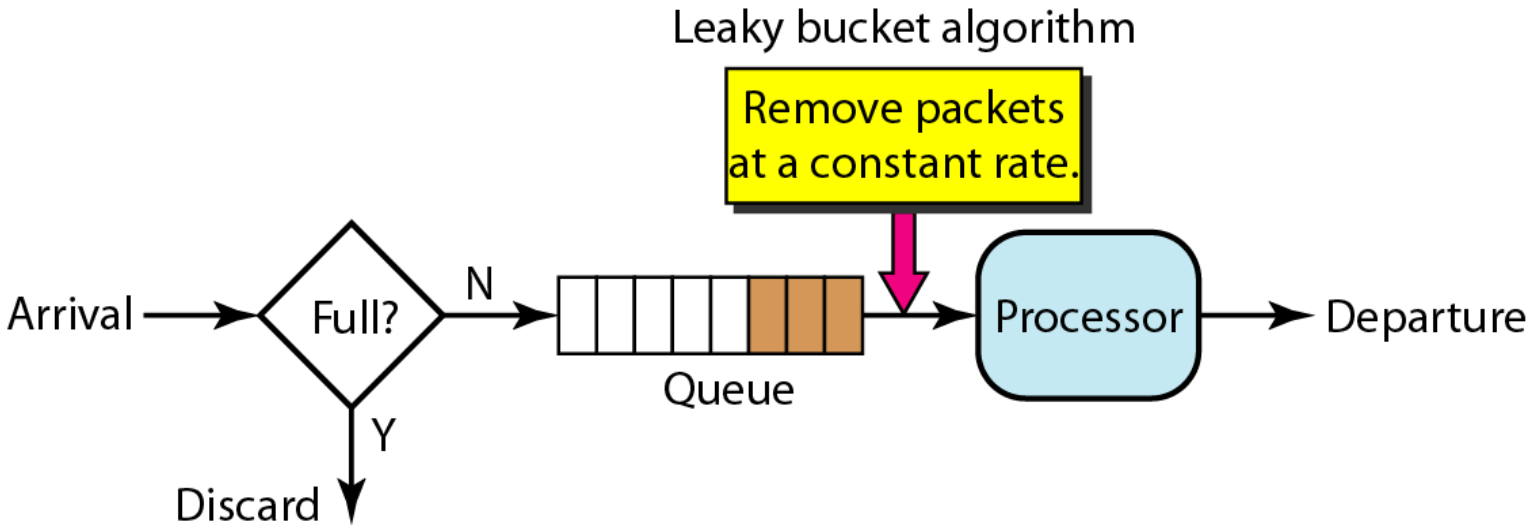
- **In this technique, the packets are still assigned to different classes and admitted to different queues.**
- **The queues are weighted based on the priority of the queues**
  - **Higher priority means a higher weight.**
- **The system processes packets in each queue in a round-robin fashion with the number of packets selected from each queue based on the corresponding weight.**

Traffic Shaping: Leaky bucket



- Conceptually, each host is connected to the network by an interface containing a leaky bucket.
- To send a packet into the network, it must be possible to put more water into the bucket.
- If a packet arrives when the bucket is full, the packet must either be queued until enough water leaks out to hold it or be discarded.

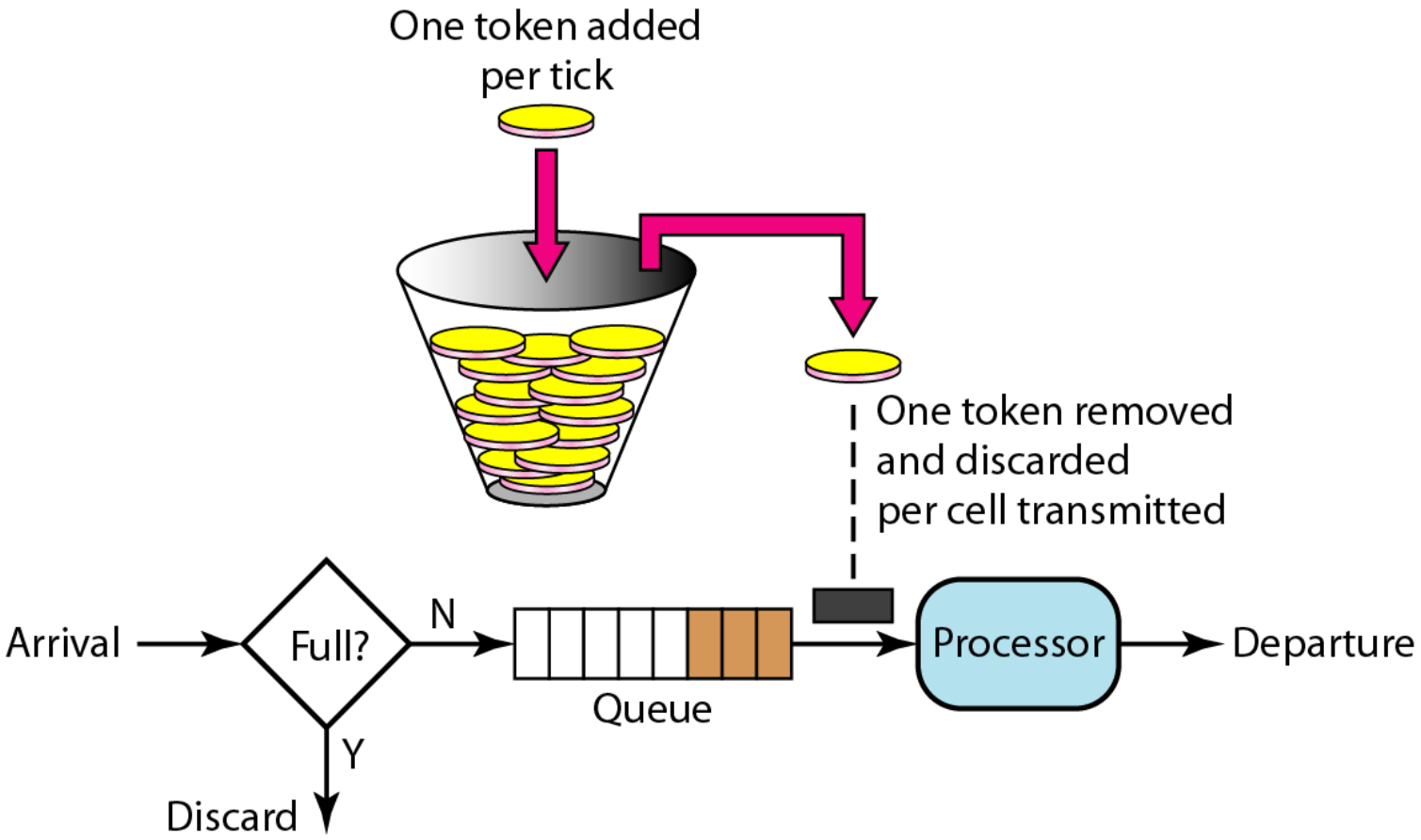
Leaky bucket implementation



Note

A leaky bucket algorithm shapes bursty traffic into fixed-rate traffic by averaging the data rate. It may drop the packets if the bucket is full.

Traffic Shaping: Token bucket



- No more than a fixed number of tokens, can accumulate in the bucket
- If the bucket is empty, we must wait until more tokens arrive before we can send another packet.