SCTP, Congestion Control and Quality of Service

Unit 4

24-4 SCTP

Stream Control Transmission Protocol (SCTP) is a new transport-layer protocol designed to combine some features of UDP and TCP in an effort to create a protocol for multimedia communication.

- UDP
  - A message oriented protocol
  - Each message is independent of other
    - Desirable feature for IP telephony
  - Unreliable delivery
    - Msg can be lost, duplicated or reached out of order
    - No congestion and flow control
- TCP
  - Byte (stream) oriented, no message boundaries
  - Reliable protocol
    - In sequence delivery, error detection and recovery
- SCTP
  - **Combines best features of UDP and TCP**
  - Reliable message oriented protocol
  - Handles lost, duplicate, and out of order data
  - Has other features like multi-streaming and Multihoming

Table 23.4 Some SCTP applications

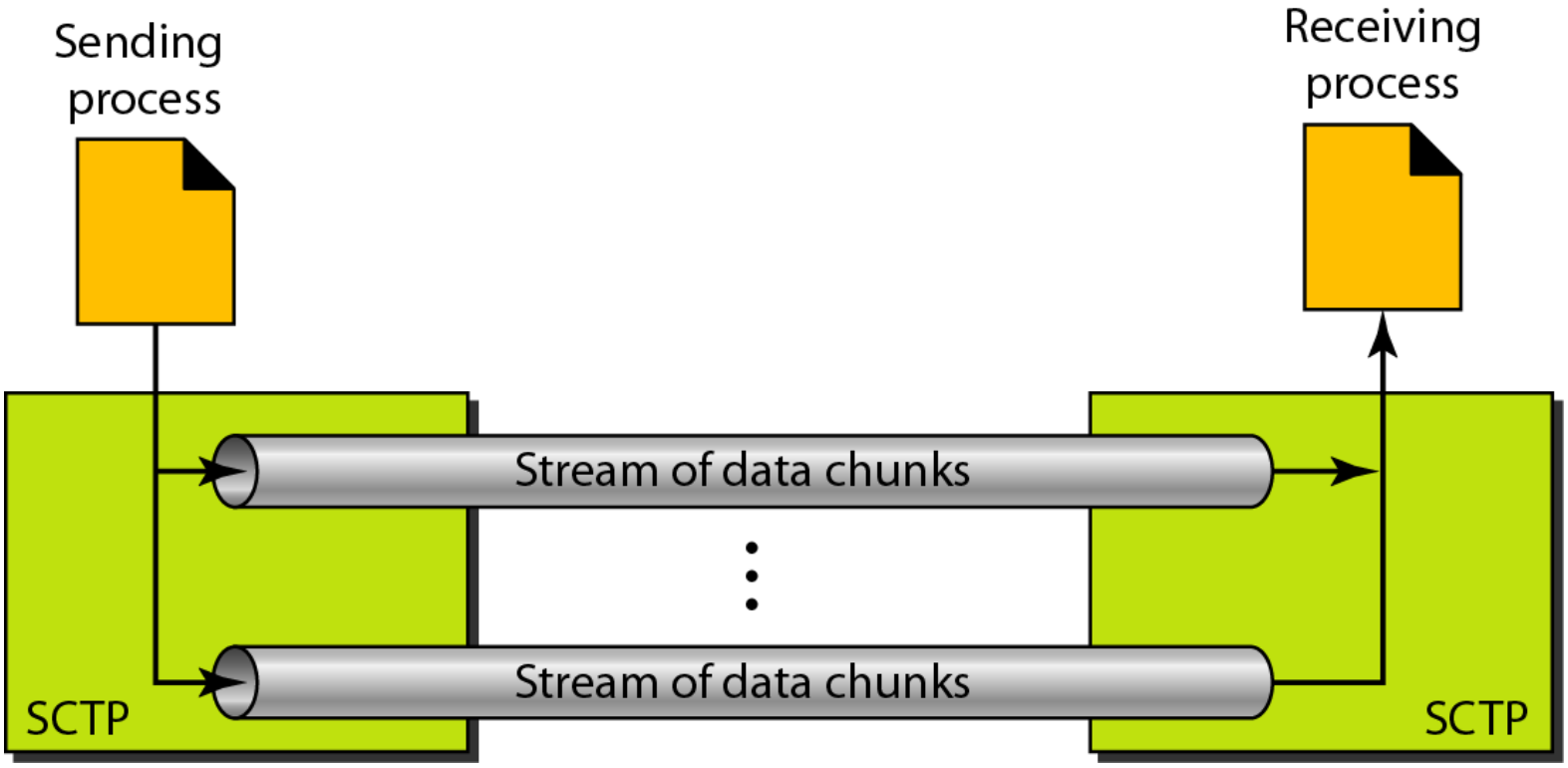
Protocol	*Port Number*	*Description*
IUA	9990	ISDN over IP
M2UA	2904	SS7 telephony signaling
M3UA	2905	SS7 telephony signaling
H.248	2945	Media gateway control
H.323	1718, 1719, 1720, 11720	IP telephony
SIP	5060	IP telephony

SCTP is a message-oriented, reliable protocol that combines the best features of UDP and TCP.

SCTP services

- Process-to-Process communication
- Multiple streams
- Multihoming
- Full duplex communication
- Connection oriented service
- Reliable service

Figure 23.27 Multiple-stream concept

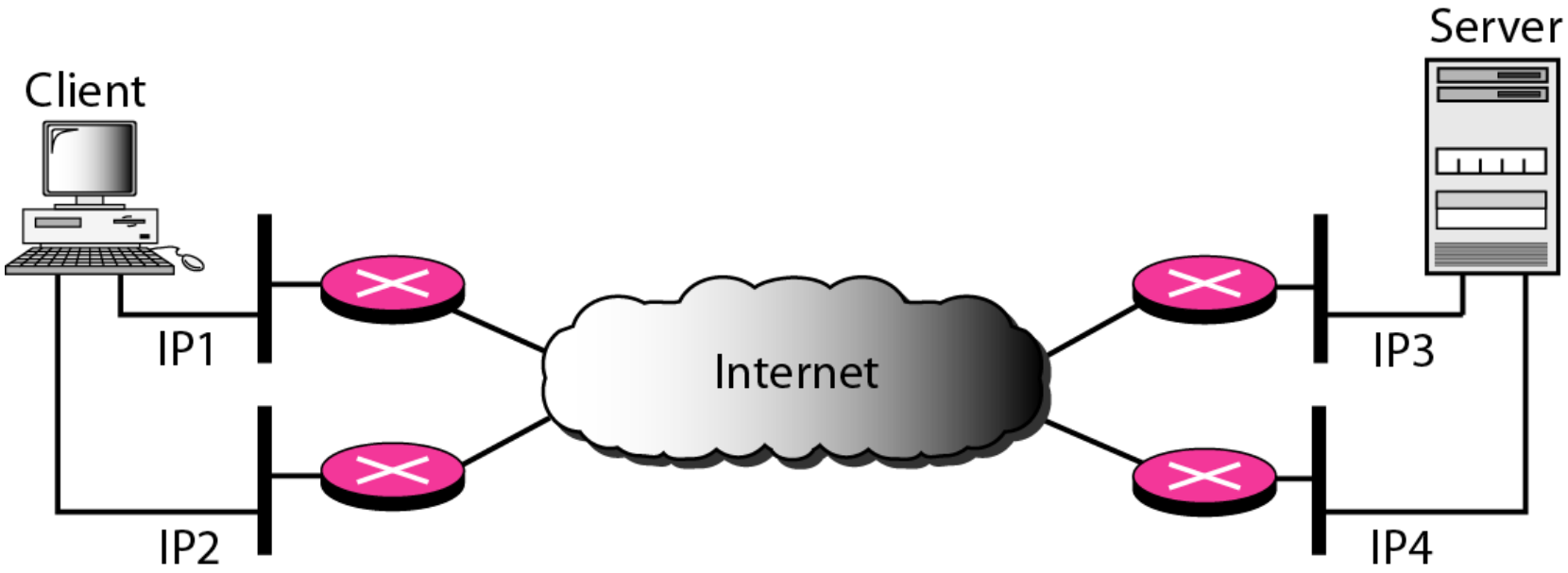


Note

An association in SCTP can involve multiple streams.

- TCP Streaming
  - Single stream, single homing
    - A loss at any point blocks the delivery of rest
      - May be acceptable for text (non real time)
      - Not acceptable for real time apps (audio, video)
- SCTP
  - Multiple streams in each connection
    - Called **Association**
    - If one stream is blocked
      - other stream can still deliver
    - Similar to multi-lane highway
  - Provides for Multi-homing

Figure 23.28 Multihoming concept



Note

SCTP association allows multiple IP addresses for each end.

Endpoints and Associations

- In SCTP

Endpoint (communicating parties)

Association (communicating relationship)

Endpoint may have more than one IP address but at most one port number.

An SCTP application can have one address or set of addresses bind to that socket

eg: **endpoint = [10.1..4.2, 10.1.5.3 : 80]**

- SCTP Features
  - **TSN (Transmission Sequence Number)**
    - Unit of data in SCTP is a **data chunk**
    - Data Transfer is controlled by numbering the data chunks (TSNs)
    - Analogous to sequence number in TCP
    - 32 bit field (randomly initialized) in data chunk header
  - **SI (Stream Identifier)**
    - Several streams in each association
    - Each stream identified by **stream identifier**
    - 16 bit field in data chunk header starting from 0.
  - **SSN (Stream Sequence Number)**
    - Identify data chunks belonging to same stream
    - Ensures ordered delivery within stream

- SCTP Features…
  - Packets
    - Data is carried as **data chunks**
    - Control info is carried as **control chunks**
    - Several control chunks and data chunks together
      - Makes a packet
    - Packet plays same roles as segment in TCP

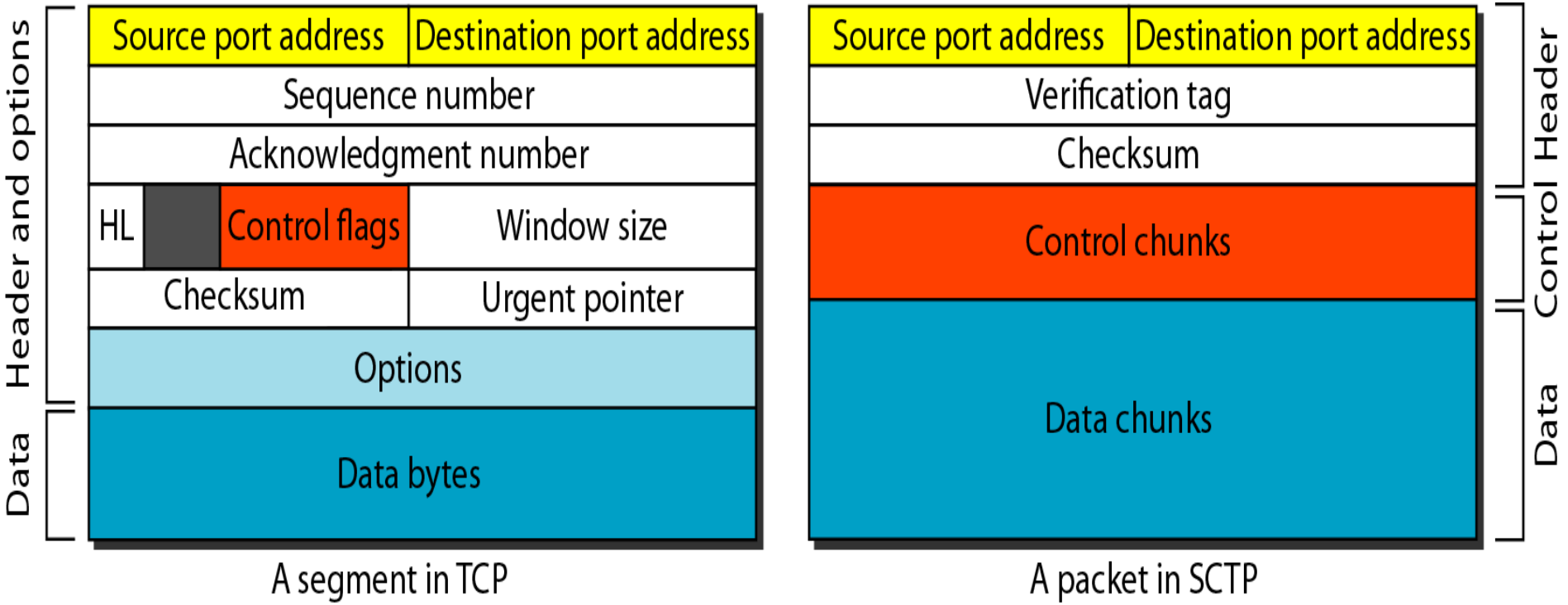
In SCTP, a data chunk is numbered using a TSN.

To distinguish between different streams, SCTP uses an SI.

To distinguish between different data chunks belonging to the same stream, SCTP uses SSNs.

TCP has segments; SCTP has packets.

Figure 23.29 Comparison between a TCP segment and an SCTP packet



- Differences between TCP and SCTP
  - Control info is part of header in TCP
    - In SCTP, it is included in **control chunks and not header**
    - Several type of control chunk, each for different purpose

      (Refer Table – 23.5)
  - Data in TCP segment is treated as one entity.
    - SCTP: a packet can carry several data chunks
    - Each chunk can belong to different stream
  - There is no **options** section in SCTP
  - **SCTP header is short, fixed at 12 bytes**
    - TSN belongs to data chunk, so carried in chunk's header
    - Ack and window size are part of control chunk
    - No need for header length as header size is fixed
    - No equivalent of **urgent** pointer

- Differences between TCP and SCTP…
  - Checksum in SCTP is 32 bits
  - SCTP has **verification** tag, identifies the association
    - Not present in TCP, as it has only one stream
    - Combination of IP, Port defines a connection in TCP
    - SCTP has multi homing (different IP Addresses)
    - Needs a tag to identify an association
  - TCP: Control info (**SYN, FIN)** consumes one seq number
  - SCTP: Control chunks do not use TSN, SI, SSN
  - SCTP packet can carry several data chunks in each packet
    - TSN, SI, SSN define each data chunk
  - SCTP Association
    - May send many packets
    - A packet contains several chunks
    - Chunks may belong to different streams

Table 23.5 Chunks

Type	*Chunk*	*Description*
0	**DATA**	User data
1	**INIT**	Sets up an association
2	**INIT ACK**	Acknowledges INIT chunk
3	**SACK**	Selective acknowledgment
4	**HEARTBEAT**	Probes the peer for liveliness
5	**HEARTBEAT ACK**	Acknowledges HEARTBEAT chunk
6	**ABORT**	Aborts an association
7	**SHUTDOWN**	Terminates an association
8	**SHUTDOWN ACK**	Acknowledges SHUTDOWN chunk
9	**ERROR**	Reports errors without shutting down
10	**COOKIE ECHO**	Third packet in association establishment
11	**COOKIE ACK**	Acknowledges COOKIE ECHO chunk
14	**SHUTDOWN COMPLETE**	Third packet in association termination
192	**FORWARD TSN**	For adjusting cumulative TSN

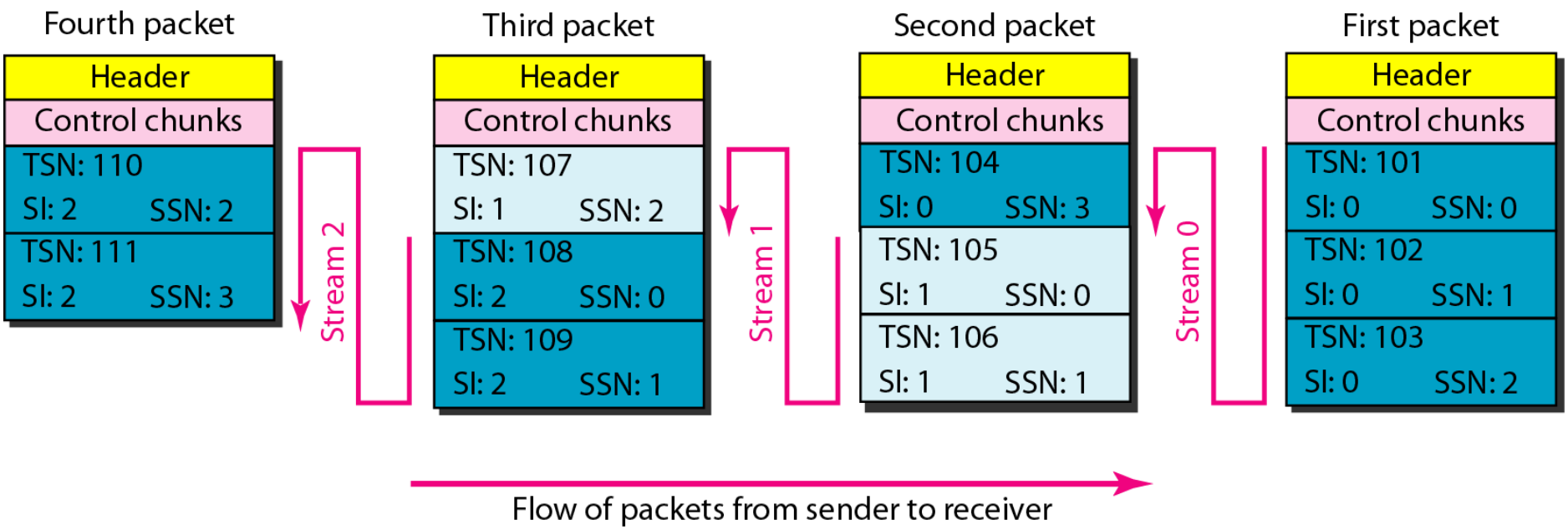
Note

In SCTP, control information and data information are carried in separate chunks.

Consider an Example

- Appln need to send 11 msgs in 3 streams
- Stream 1 : 4 msgs, stream 2: 3 msgs, stream 3: 4 msgs
- Each data chunk needs 3 identifiers (TSN, SI, SSN)
- Data chunk of a stream need not be continugous
  - This diagram shows them as contiguous

Figure 23.30 Packet, data chunks, and streams



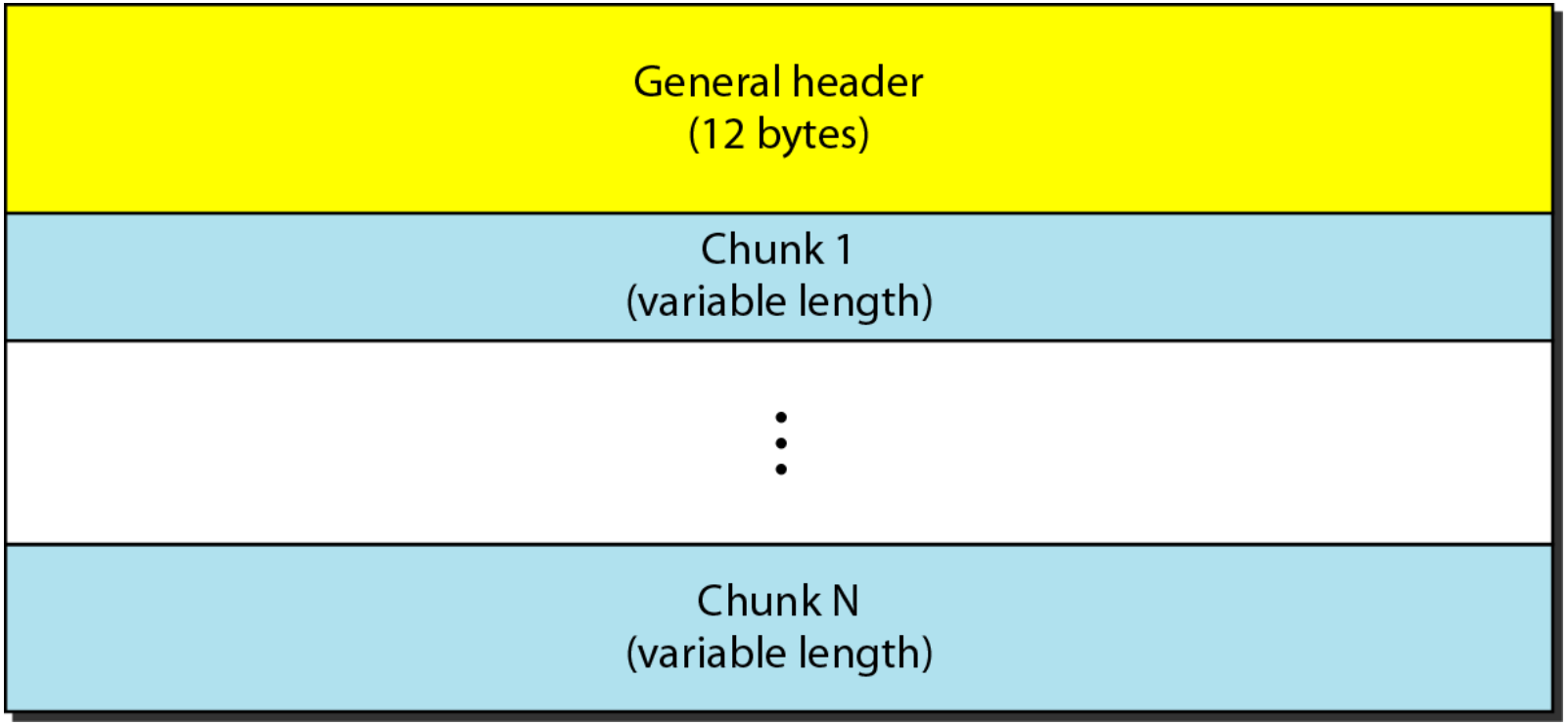
Note

Data chunks are identified by three items: TSN, SI, and SSN.
TSN is a cumulative number identifying the association;
SI defines the stream;
SSN defines the chunk in a stream.

Note

In SCTP, acknowledgment numbers are used to acknowledge only data chunks;
control chunks are acknowledged by other control chunks if necessary.

Figure 23.31 SCTP packet format



Note

In an SCTP packet, control chunks come before data chunks.

Figure 23.32 General header

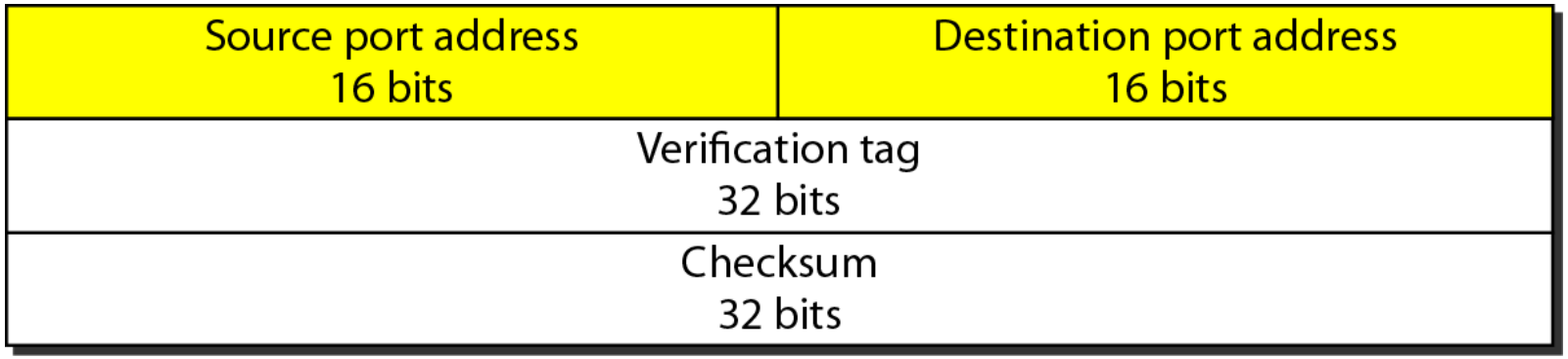
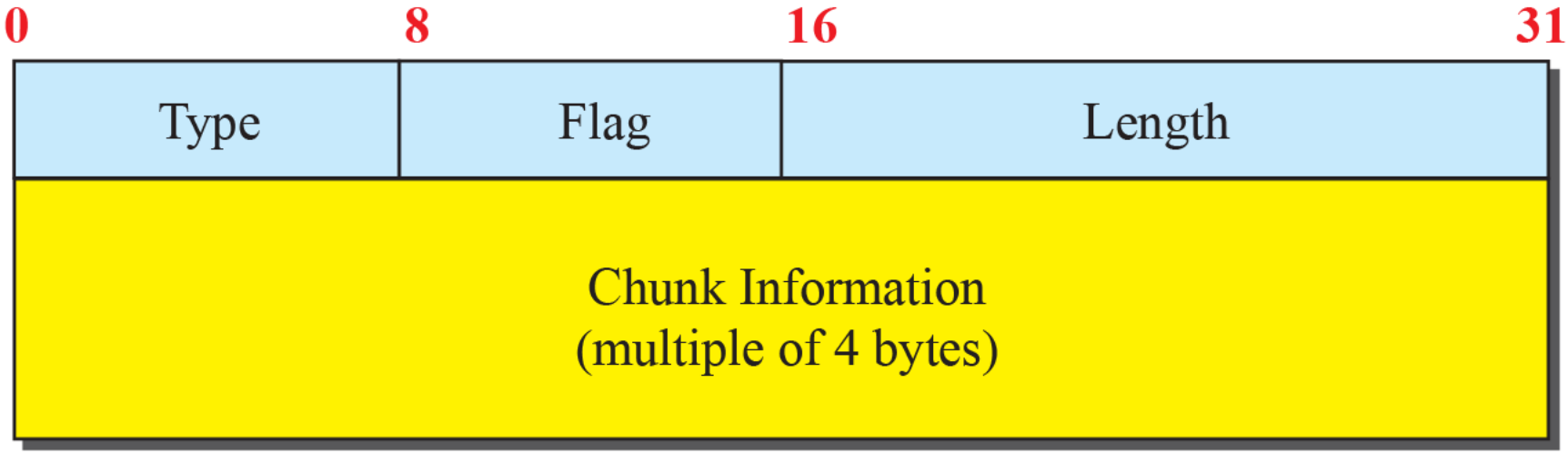


Figure 24.44 : Common layout of a chunk



SCTP Associations

- Like TCP, SCTP is connection-oriented
- A connection-oriented protocol is one that requires a setup procedure to establish the communication relationship (and state) between two parties
- To establish this state, both sides go through a specific set of exchanges
  - TCP uses a 3-way handshake (SYN, SYN/ACK, ACK)
  - SCTP uses a 4-way handshake (we examine this later)

SCTP Association II

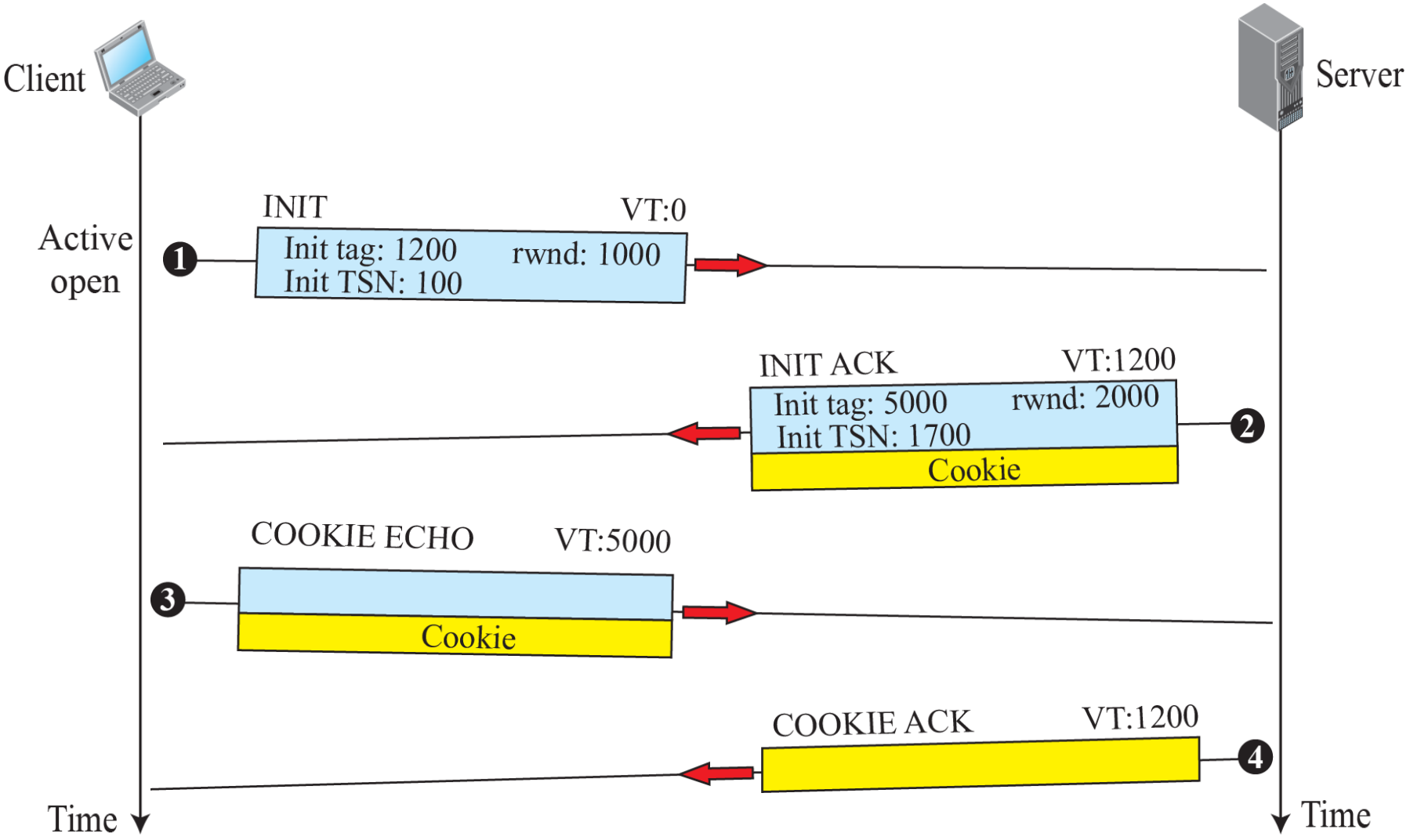
- In TCP, the communication relationship between two endpoints is called a “connection”
- In SCTP, this is called an “association” this is because it is a broader concept than a single connection (i.e. multi-homing)
- An SCTP association can be represented as a pair of SCTP endpoints:

  assoc = { [10.1.61.11 : 2223], [161.10.8.221, 120.1.1.5 : 80]}

SCTP Association III

- An SCTP endpoint may have multiple associations
- Only one association may be established between any two SCTP endpoints

Figure 24.45: Four-way handshaking



Note

No other chunk is allowed in a packet carrying an INIT or INIT ACK chunk.
A COOKIE ECHO or a COOKIE ACK chunk can carry data chunks.

Note

In SCTP, only DATA chunks consume TSNs; DATA chunks are the only chunks that are acknowledged.

Figure 24.46 : Association termination

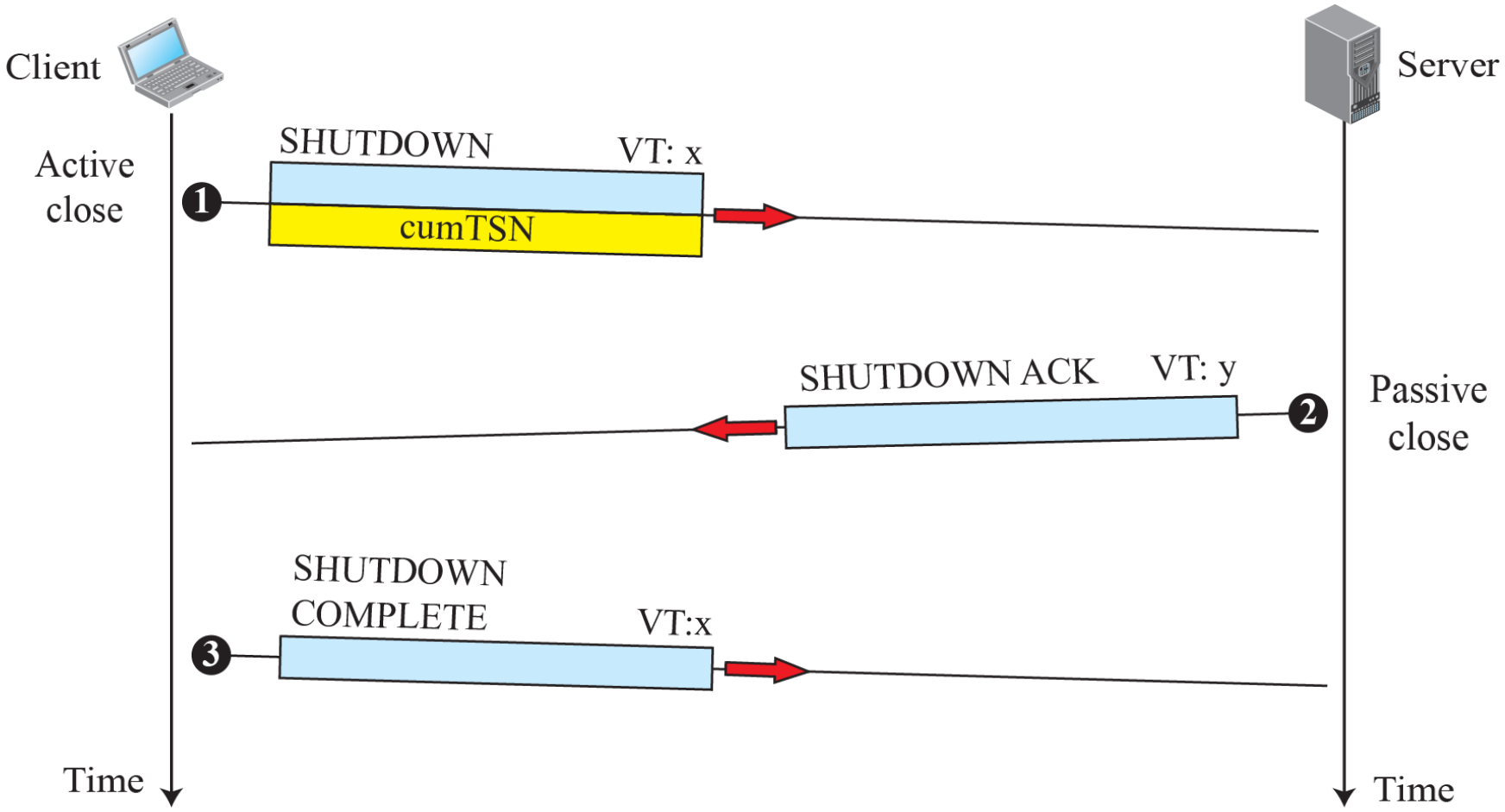
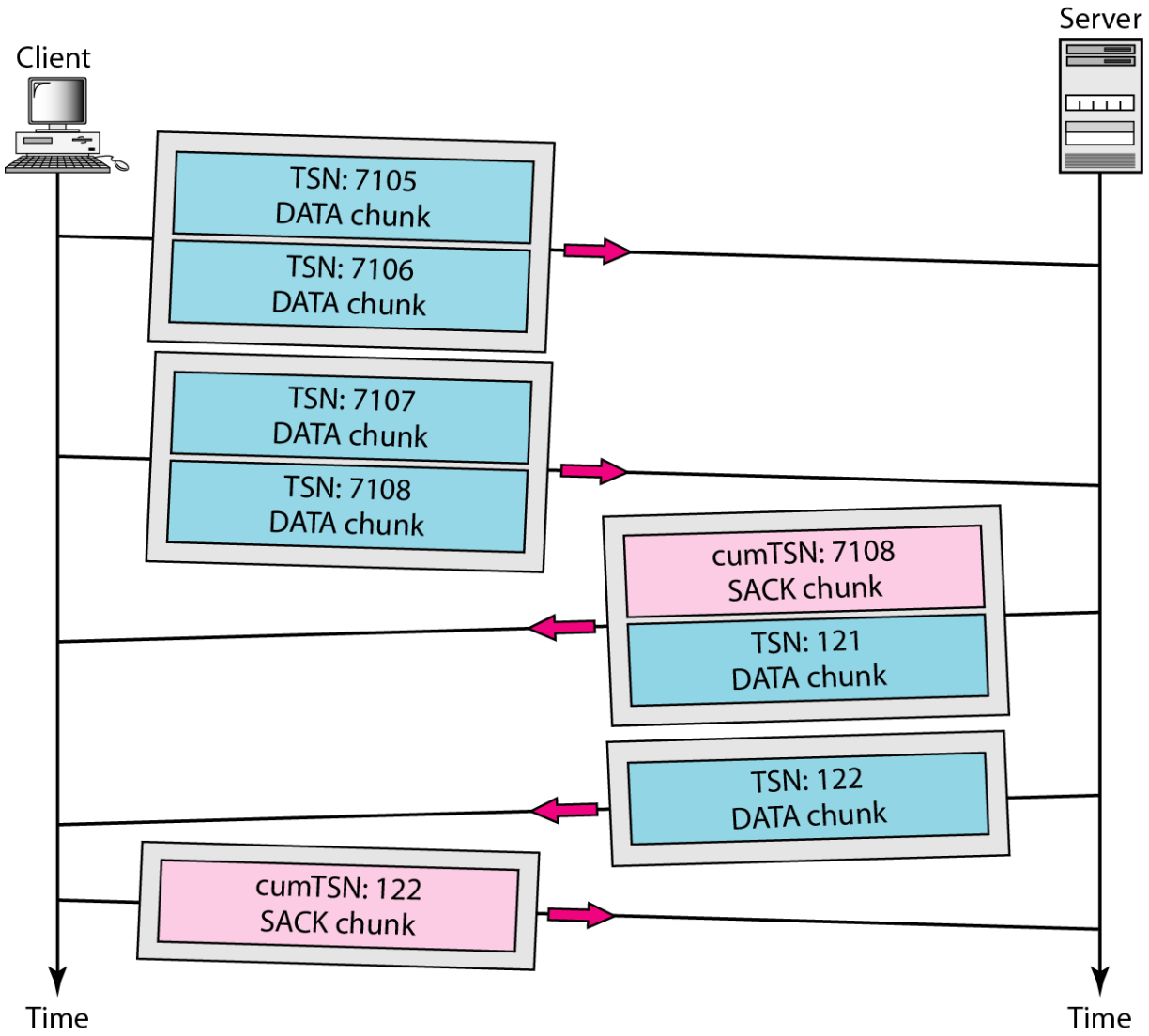


Figure 23.34 Simple data transfer



Note

The acknowledgment in SCTP defines the cumulative TSN, the TSN of the last data chunk received in order.

18.3.4 Congestion Control

Congestion control is a mechanism for improving performance. Later we will discuss congestion at the transport layer.

Now we will see **congestion at the network layer** is not explicitly addressed in the Internet model, the study of congestion at this layer may help us to better understand the cause of congestion at the transport layer and find possible remedies to be used at the network layer.

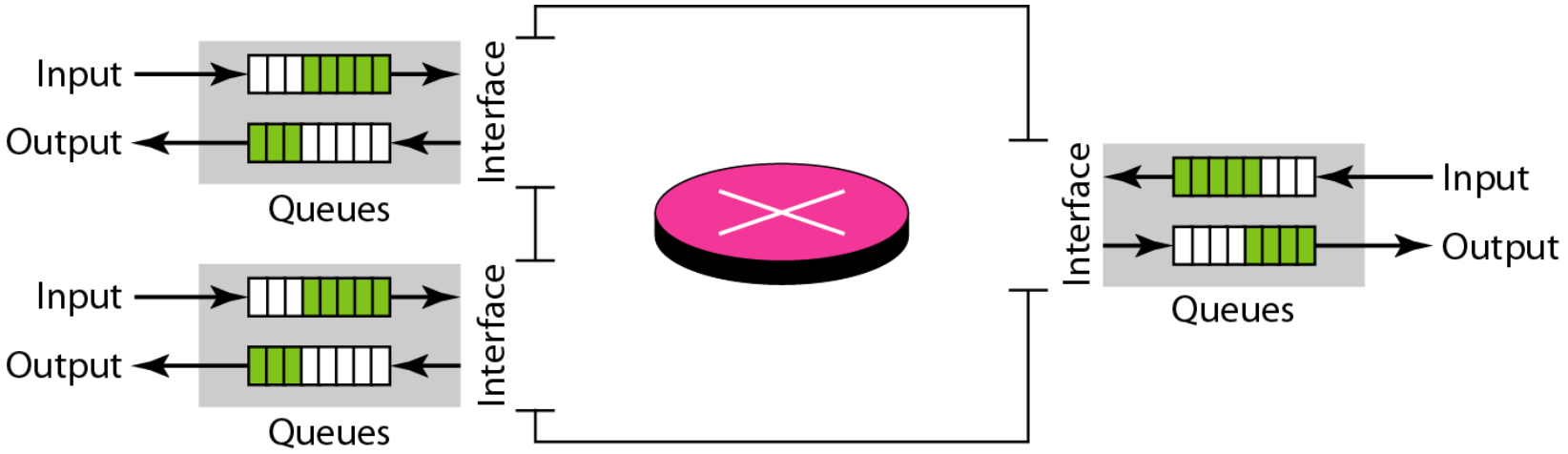
Congestion at the network layer is related to two issues, throughput and delay.

Congestion Control

- 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.**

- Congestion
  - Load on the network is more than capacity
  - Load: Number of packets sent to network
  - Capacity: Number of packets it can route
- Why Congestion
  - Consider Road traffic
    - Despite multi lanes, congestion happens
      - Accident
      - Abnormality in traffic flow (oversize vehicle) etc
  - Network congestion
    - Routers/switches have queues (buffers)
    - These can overflow

Figure 24.3 Queues in a router

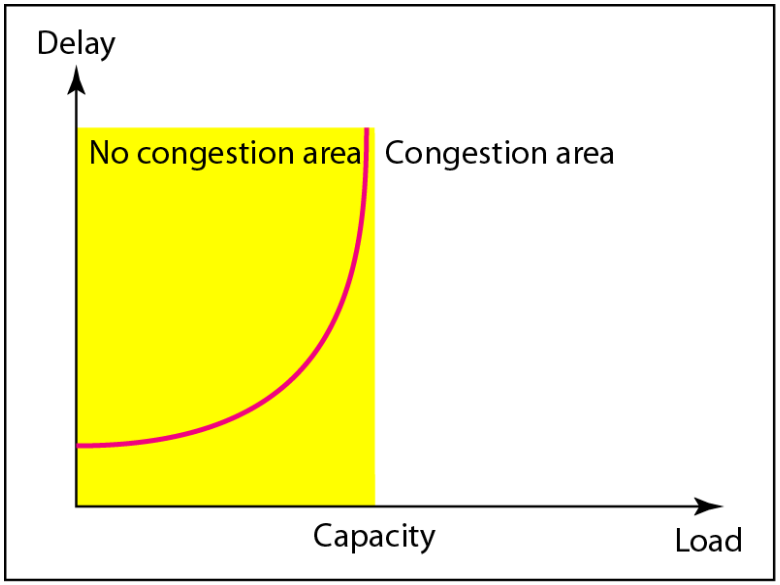


- Routing Steps on packet arrival
  - Packet is put at the end of i/p queue
  - Packet is processed once it reaches front of queue
  - Next hop/link is determined as per routing tables
  - Packet is put in output queue
    - Waits for its turn to be sent
- **Congestion**
  - **Packet arrival rate is faster than processing**
  - **Packet processing is faster than packet departure rate**

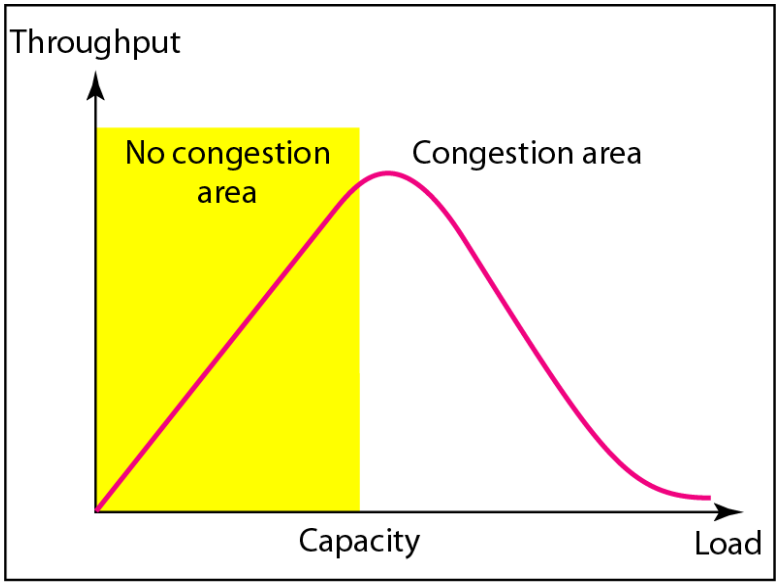
Packet delay and throughput as functions of load

- When the load is much less than the capacity of the network, the delay is at minimum.(Prop. delay + Processing delay). When the load reaches the capacity, the delay increases sharply(queuing delay is added). When the load is greater than the capacity delay becomes infinite.
- When the load is much less than the capacity of the network, the throughput increases proportionally with load. When the load reaches the capacity, we expect the throughput to remain constant, but it declines sharply. (discarding of packets by routers). When the load exceeds the capacity, routers full, discard packets, source retransmits, but packets never reaches the destination.

Figure Packet delay and throughput as functions of load



a. Delay as a function of load



b. Throughput as a function of load

- When pkt is delayed, source retransmits
  - Causes more congestion

24-3 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).

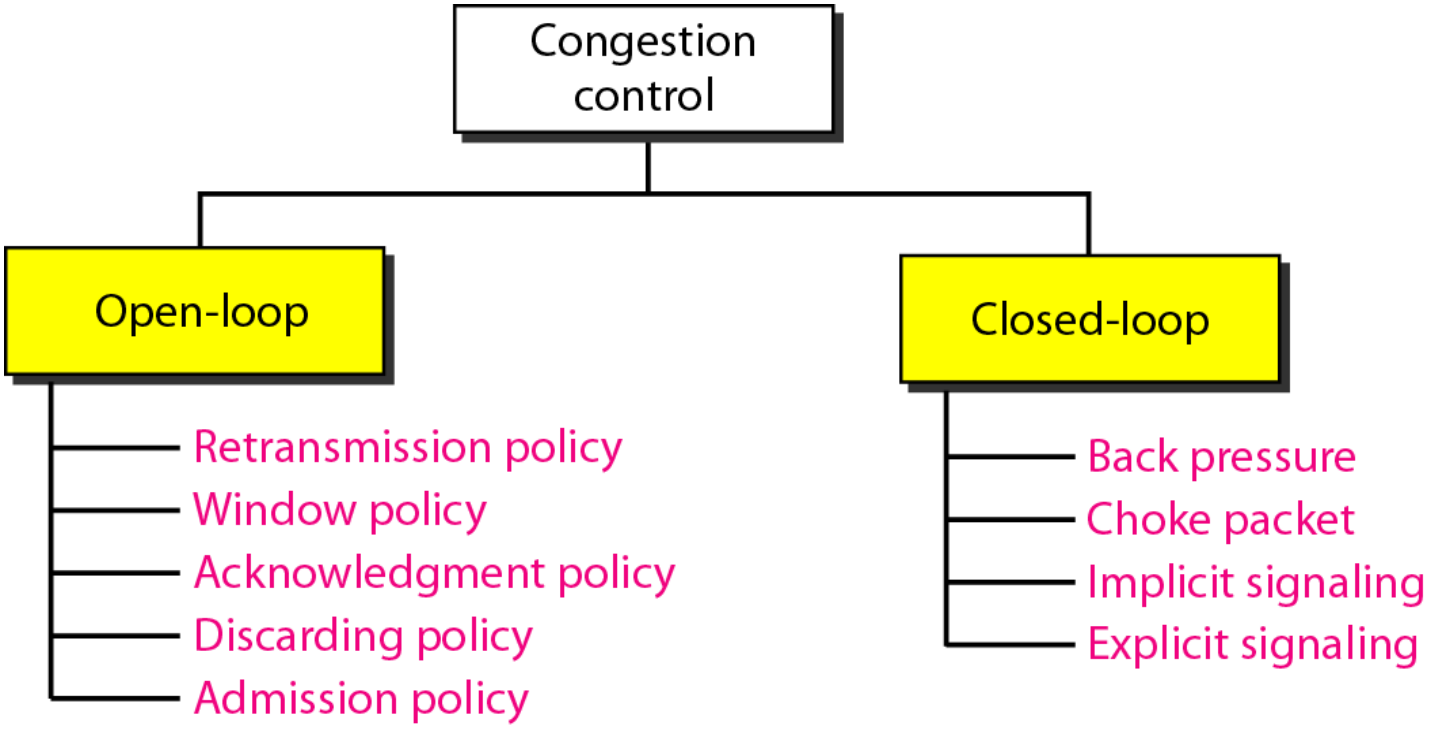
Topics discussed in this section:

Open-Loop Congestion Control

Closed-Loop Congestion Control

- Open loop congestion control
  - Prevent congestion before it occurs
    - Prevention is better than cure
- Close loop congestion control
  - Reactive action
  - Take action to alleviate it after it happens

Figure 24.5 Congestion control categories



- Retransmission Policy
  - Sender resends
    - When it believes packet is lost or corrupted
  - In general, It further increases congestion
    - Congestion feeds on itself
  - A good policy can prevent congestion
    - Selection of good policy and good retransmission timer, Optimizes efficiency and Prevents congestion

- Window Policy
  - Type of window used at sender affect congestion
  - Selective repeat is better than Go-back-N
    - Retransmitting all the N packets will cause congestion
    - Go Back N is also inefficient

      - Some packets may have arrived
    - Selective repeat retransmits only specific packets that are lost or corrupted.

- Acknowledgement policy
  - Receiver ack policy affects congestion
  - Not acknowledging each pkt slows sender
    - Helps prevent congestion
  - Receiver sends acknowledgement
    - Either on piggyback to the data
    - Or on expiry of special timers
    - Or N pkts have been received
  - Acknowledgement also adds to network load
    - Fewer acks imply less load on network

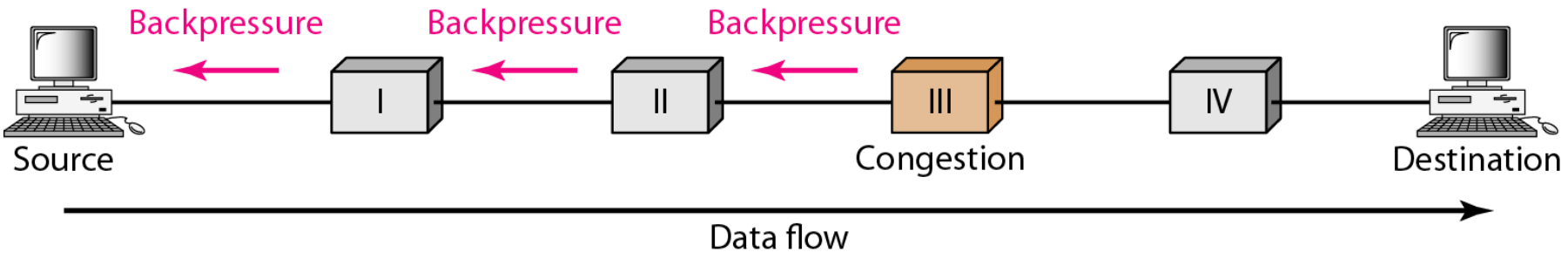
- Discarding policy
  - A good discarding policy prevents congestion
    - Should not harm the transmission integrity
  - Example: audio transmission
    - Discard less sensitive packet
    - Quality of sound is preserved
    - Congestion is prevented if it was likely to occur

- Admission Policy
  - Applicable when dealing with QoS mechanism
  - Can prevent congestion in virtual circuit n/w
  - Switches/router must check resources
    - Before admitting new flows
    - Should deny admission when possibility of congestion

Closed loop Congestion Control

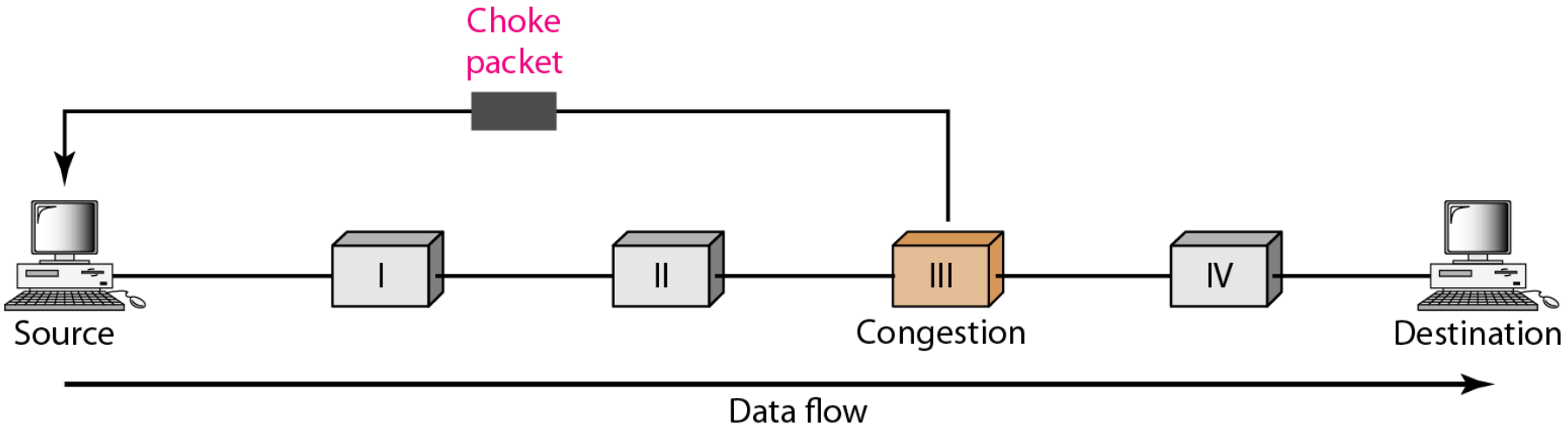
- Take action to alleviate it after it happens
- Backpressure
- Choke packet
- Implicit signaling
- Explicit signaling

Figure 24.6 Backpressure method for alleviating congestion



- Congested node stops receiving from the immediate upstream node
- Can be done virtual circuit network
  - X.25 network used this technique
- Can not be applied in datagram network like IP
  - Why?

Figure 24.7 Choke packet



- Packet send to the source to inform that there is congestion
- Which ICMP error follows this ?

- Implicit Signaling
  - There is no communication by congested node
  - Source somehow guesses the congestion
    - E.g. no acks for a while for multiple pkts sent
    - Delay in receiving ack is interpreted as congestion
  - Used in TCP
- Explicit Signaling
  - Different from choke packet
  - No separate packet, but piggybacked on data
  - Used in Frame Relay and ATM networks

TCP congestion control:

- *Actual window size = min (cwnd, rwnd)*
- *TCP sender detects congestion either by timeout (more severe) or by three duplicate acks (less severe)*
- *TCP Taho treat both events as same, whereas TCP Reno treat them differently.*
- *There are 3 stages: slow start, congestion avoidance and congestion detection (fast recovery)*

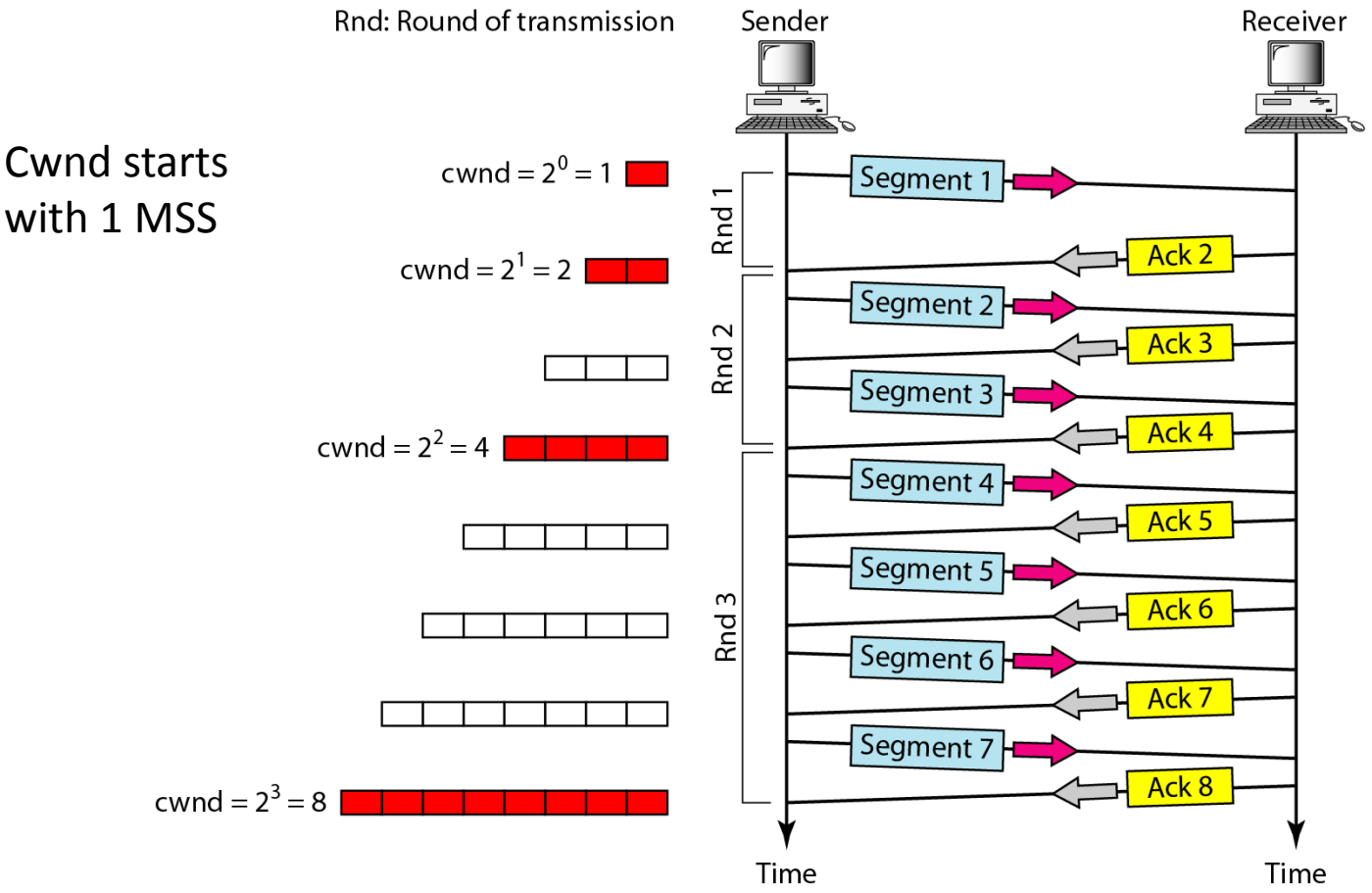
TCP congestion control: additive increase multiplicative decrease

- *approach:* sender increases transmission rate (window size), probing for usable bandwidth, until loss occurs
  - *additive increase:* increase **`cwnd`** by 1 MSS every RTT until loss detected
  - *multiplicative decrease:* cut **`cwnd`** in half after loss

Slow start: Exponential increase

- Cwnd starts with one MSS, increase by I MSS for each acknowledgement.
- Starts slowly but grows exponentially.
- We assume rwnd is very large and the sender is dictated by cwnd only.

Figure 24.8 Slow start, exponential increase



Note

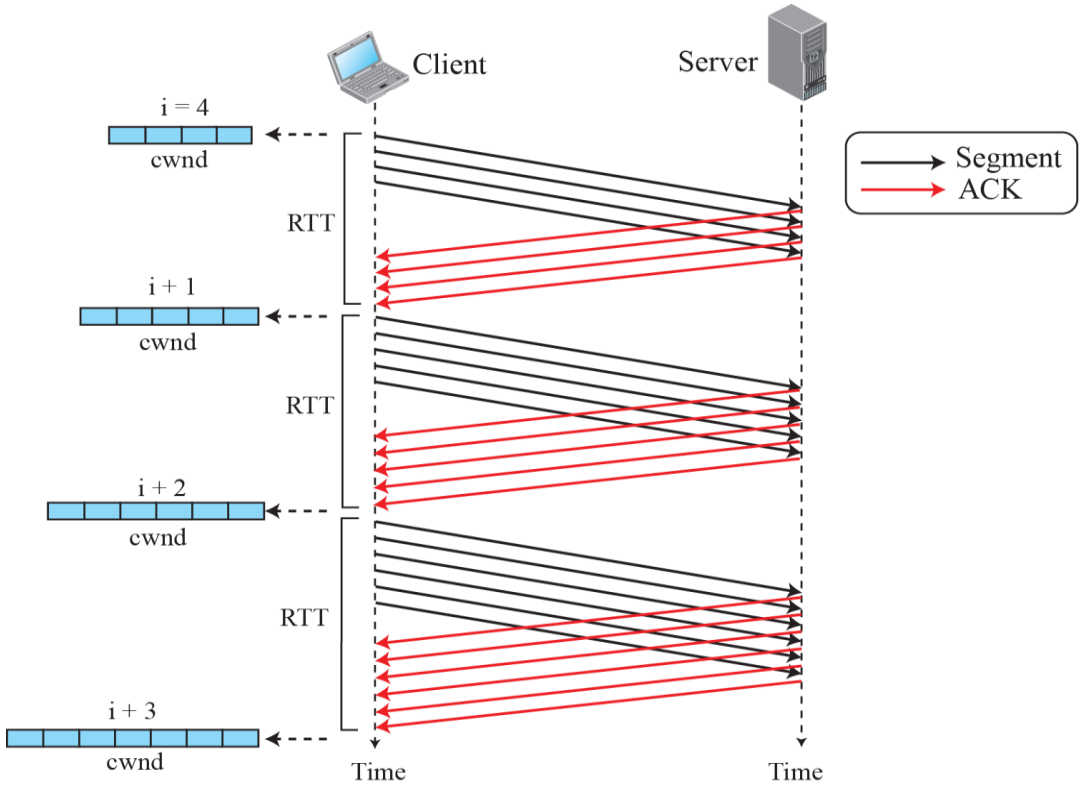
In the slow-start algorithm, the size of the congestion window increases exponentially until it reaches a threshold.

Slow start becomes still slower in case of delayed Acks

Congestion avoidance – additive increase

- When cwnd reaches ssthresh, then slow start phase stops and additive increase phase begins.
- For one RTT, it increases the cwnd by 1

Figure 24.9 Congestion avoidance, additive increase



After the threshold value cwnd increases additively

Note

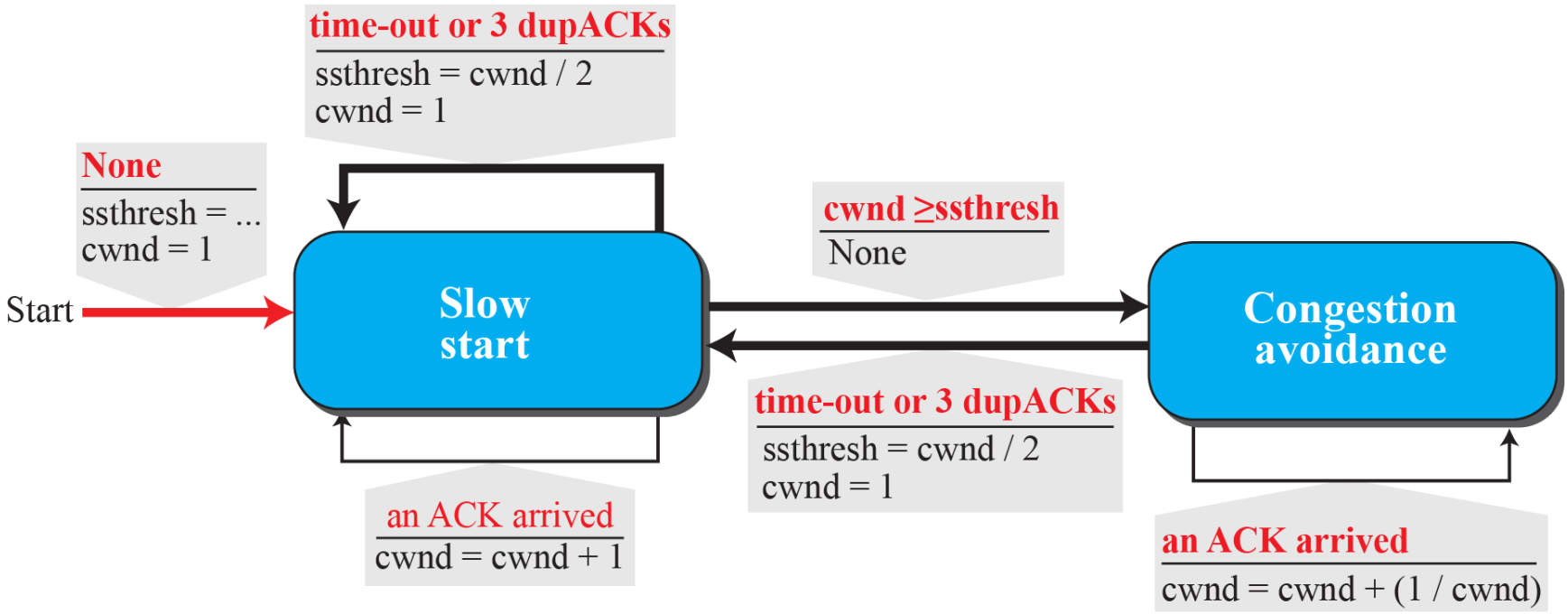
In the congestion avoidance algorithm, the size of the congestion window increases additively until
congestion is detected.

Congestion detection – fast recovery

- Optional in TCP – old versions did not use this, newer versions try to use this.
- There are three versioons of TCP – TCP Taho, TCP Reno and new Reno TCP.

Figure 24.31: FSM for Taho TCP

Uses only two algorithms – slow start and congestion avoidance



Example 24.9

Figure 24.32 shows an example of congestion control in a Taho TCP. TCP starts data transfer and sets the ssthresh variable to an ambitious value of 16 MSS. TCP begins at the slow-start (SS) state with the cwnd = 1. The congestion window grows exponentially, but a time-out occurs after the third RTT (before reaching the threshold). TCP assumes that there is congestion in the network. It immediately sets the new ssthresh = 4 MSS (half of the current cwnd, which is 8) and begins a new slow start (SA) state with cwnd = 1 MSS. The congestion grows exponentially until it reaches the newly set threshold. TCP now moves to the congestion avoidance (CA) state and the congestion window grows additively until it reaches cwnd = 12 MSS.

At this moment, three duplicate ACKs arrive, another indication of the congestion in the network. TCP again halves the value of ssthresh to 6 MSS and begins a new slow-start (SS) state. The exponential growth of the cwnd continues. After RTT 15, the size of cwnd is 4 MSS. After sending four segments and receiving only two ACKs, the size of the window reaches the ssthresh (6) and the TCP moves to the congestion avoidance state. The data transfer now continues in the congestion avoidance (CA) state until the connection is terminated after RTT 20.

Figure 24.32: Example of Taho TCP

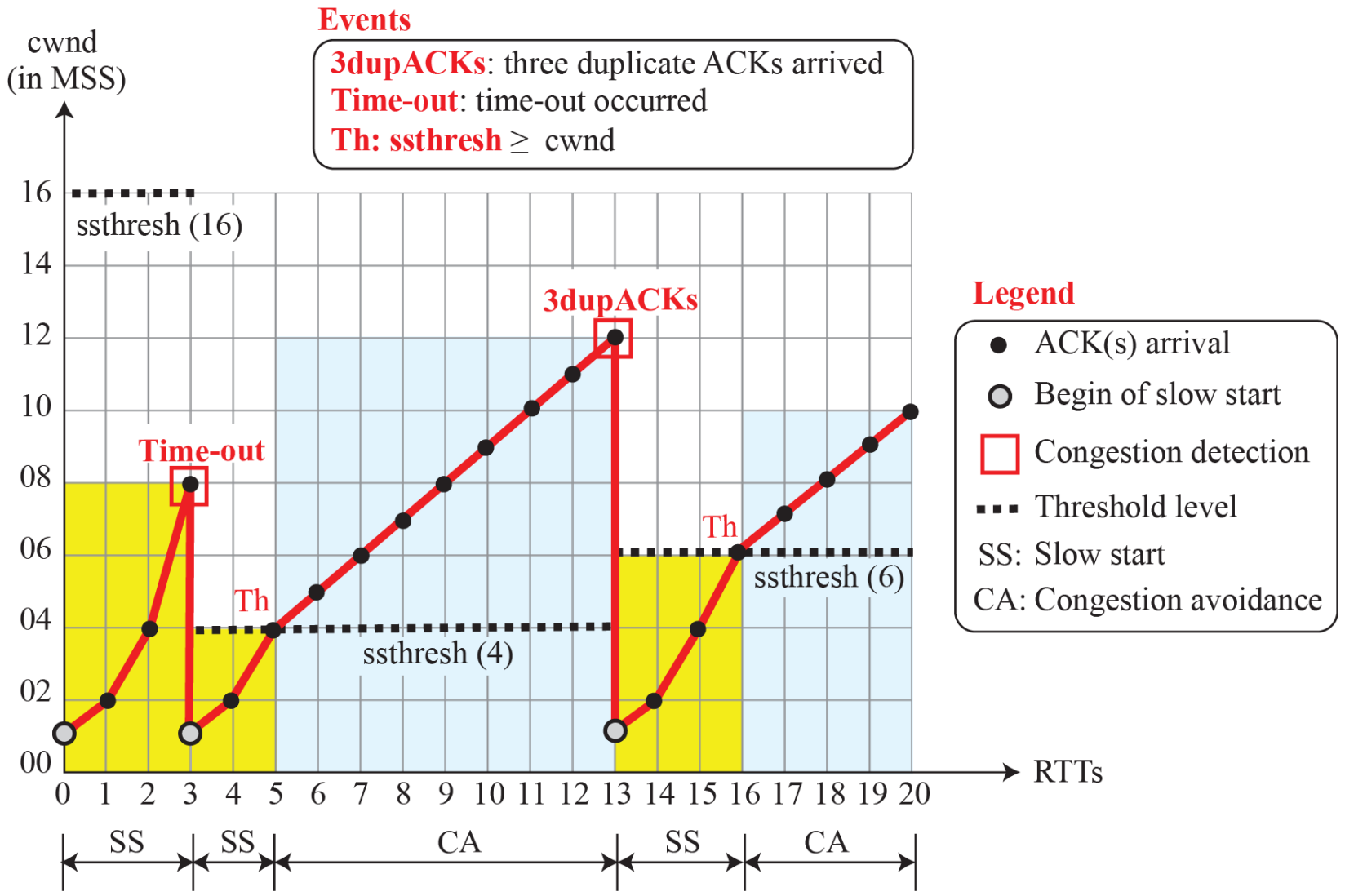
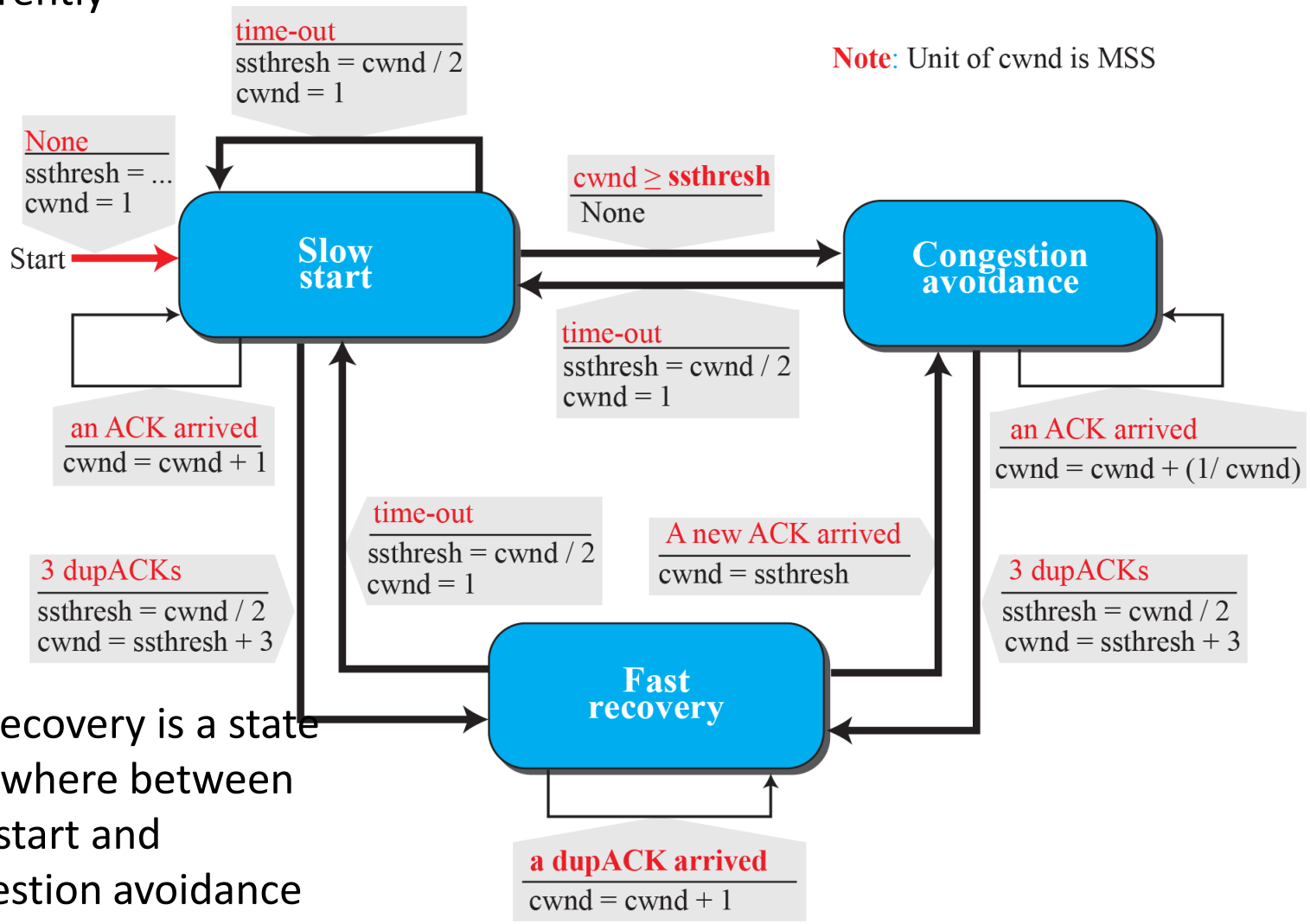


Figure 24.33: FSM for Reno TCP

3 states – fast recovery is added, 3 duplicate acks and timeout are treated differently



Fast recovery is a state somewhere between slow start and congestion avoidance

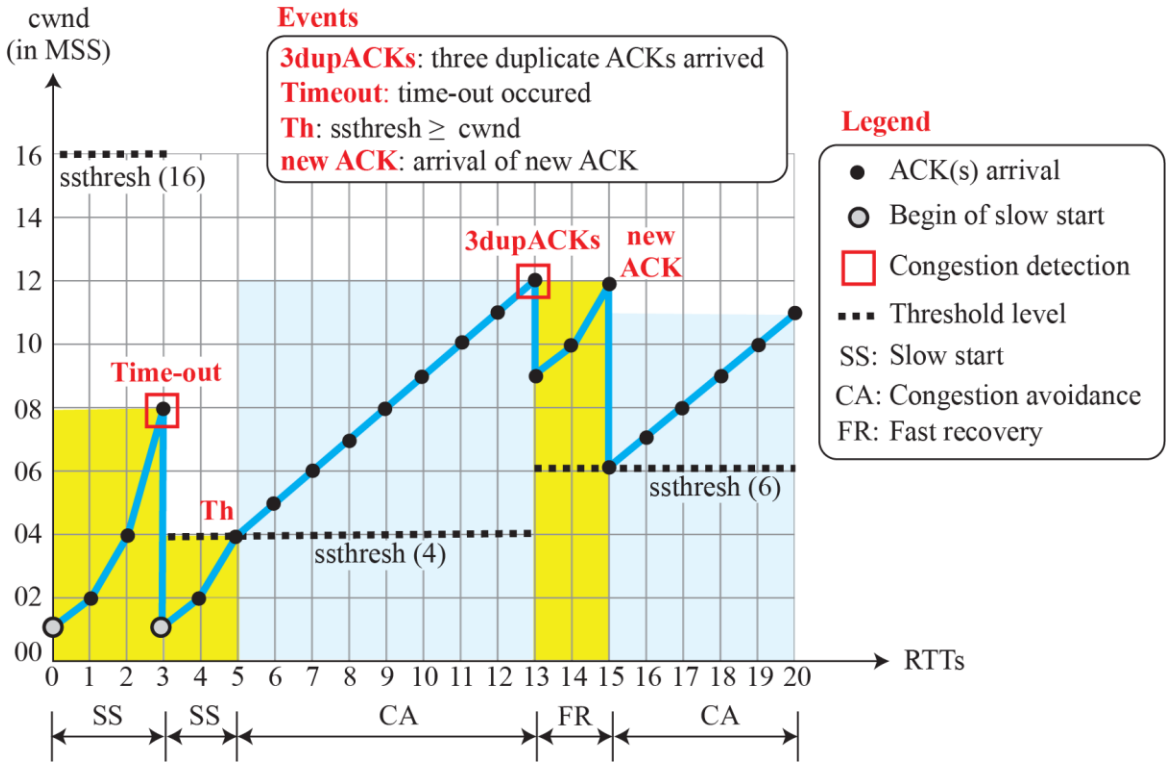
Fast recovery

- It behaves like slow start – cwnd grows exponentially but cwnd starts with ssthresh + 3 MSS
- 3 events may occur
- If duplicate acks continue to arrive, stays in same state, cwnd grows exponentially.
- If time out occurs, go to slow start (thinks that there is real congestion)
- If new ack arrives, moves to congestion avoidance state, but cwnd = ssthresh value

Example 24.10

Figure 24.34 shows the same situation as Figure 3.69, but in Reno TCP. The changes in the congestion window are the same until RTT 13 when three duplicate ACKs arrive. At this moment, Reno TCP drops the ssthresh to 6 MSS, but it sets the cwnd to a much higher value (ssthresh + 3 = 9 MSS) instead of 1 MSS. It now moves to the fast recovery state. We assume that two more duplicate ACKs arrive until RTT 15, where cwnd grows exponentially. In this moment, a new ACK (not duplicate) arrives that announces the receipt of the lost segment. It now moves to the congestion avoidance state, but first deflates the congestion window to 6 MSS as though ignoring the whole fast-recovery state and moving back to the previous track.

Figure 24.34: Example of a Reno TCP

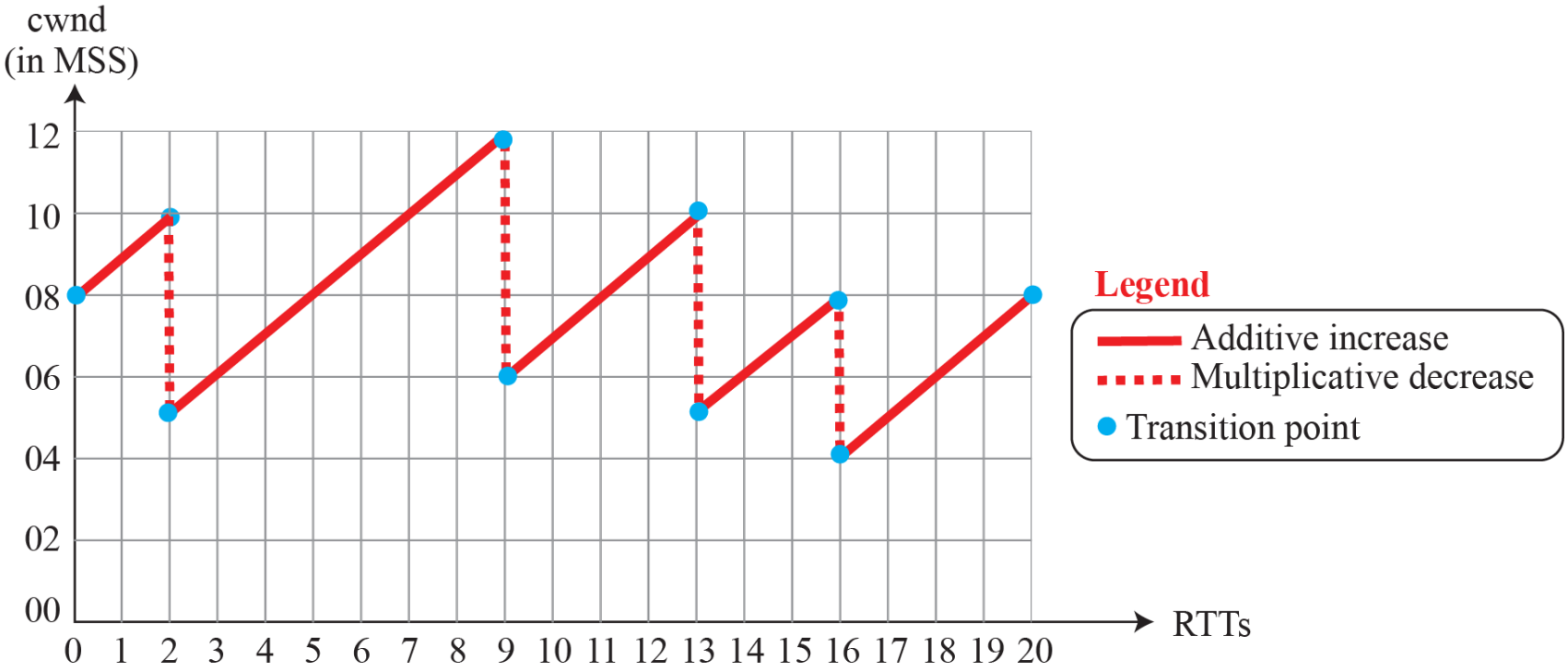


New Reno TCP

- More optimized than Reno TCP
- When it receives 3 duplicate acks, it checks whether more than one segment is lost in the current window.

Figure 24.35: Additive increase, multiplicative decrease (AIMD)

Out of the three versions, TCP Reno is most popular today.
Except slow start phase, it uses AIMD.



24-1 DATA TRAFFIC

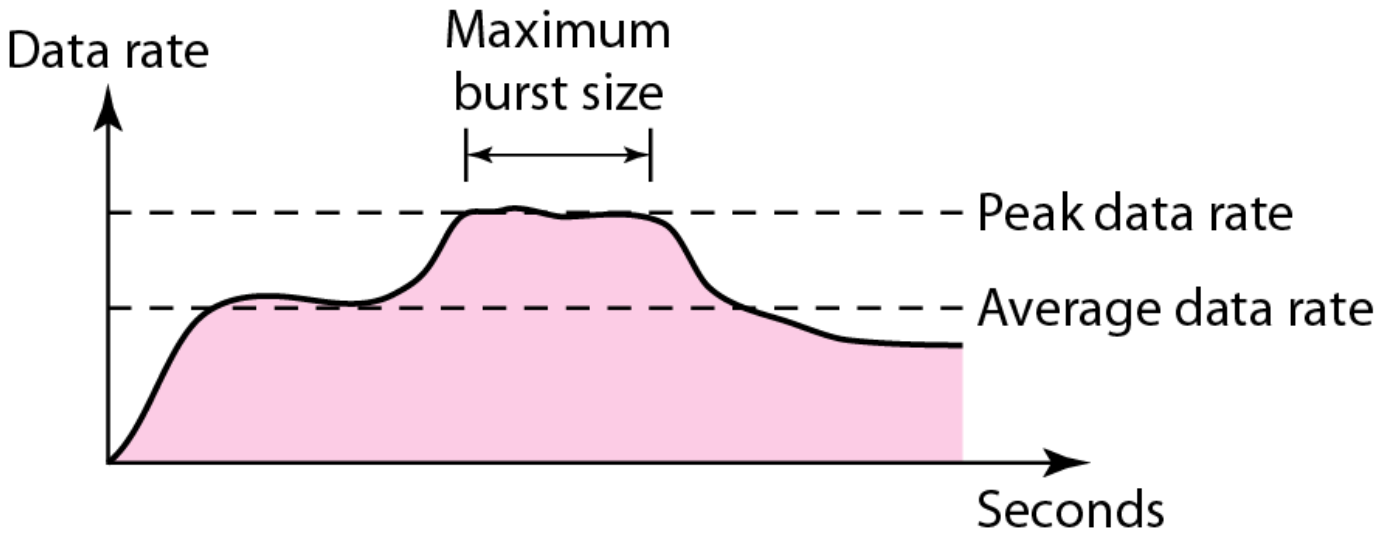
- Main focus of congestion control and QoS
  - Data traffic.
- Congestion control
  - Avoid traffic congestion.
- QoS
  - Create an environment for the traffic.
- First we should understand the data traffic itself.

Traffic Descriptor: Qualitative values that represent a data flow.

- Traffic Descriptor
  - Average Data Rate
    - Number of bits sent / duration (seconds)
  - Peak Data Rate

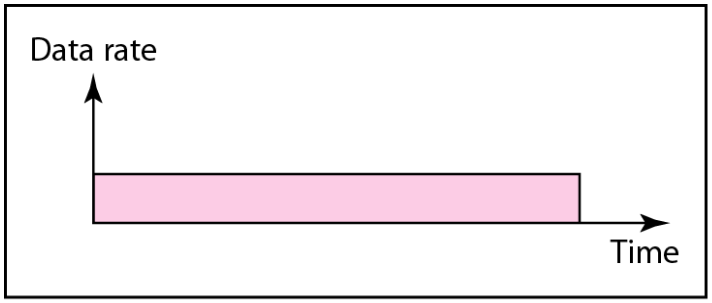
    . Maximum data rate of the traffic
    - Peak BW data traffic needs
  - Maximum Burst Rate
    - Maximum Length of time traffic is generated at peak rate
  - Effective Bandwidth
    - What network needs to allocate for traffic flow
    - A (complex) function of all 3 values above

Figure 24.1 Traffic descriptors

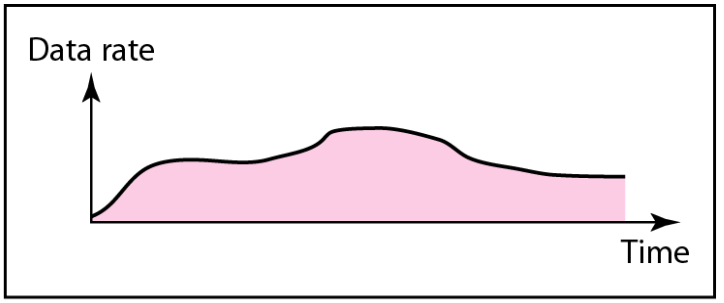


- Traffic Profiles
  - Constant Bit Rate
    - Data rate does not change
    - Average and peak is the same
    - Best traffic profile for a network to handle
  - Variable bit rate
    - Rate of data flow changes with time
    - Changes are typically smooth
    - Average and peak rates are different
    - Maximum burst size is usually a small value
  - Bursty traffic
    - Rate changes from 0 to high value
    - Average and peak rate are quite different
    - **Most difficult traffic profile for network to handle**
    - **One of the main causes of congestion**
    - **Maximum burst size is significant and needs to reshape**

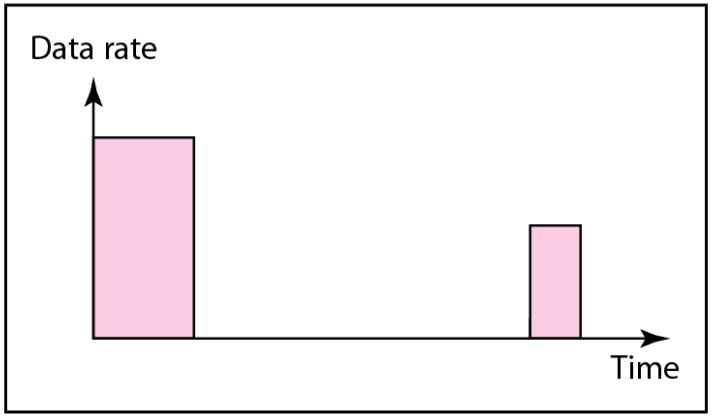
Figure 24.2 Three traffic profiles



a. Constant bit rate



b. Variable bit rate



c. Bursty

30.30.1 Definitions

We can give informal definitions of the four characteristics:

Reliability is a characteristic that a flow needs in order to deliver the packets safe and sound to the destination.

Source-to-destination delay is another flow characteristic.

Jitter is the variation in delay for packets belonging to the same flow.

Different applications need different bandwidths.

30.30.2 Sensitivity of Applications

Now let us see how various applications are sensitive to some flow characteristics. Table 30.1 gives a summary of application types and their sensitivity.

Table 30.1: Sensitivity of applications to flow characteristics

Application	*Reliability*	*Delay*	*Jitter*	*Bandwidth*
FTP	High	Low	Low	Medium
HTTP	High	Medium	Low	Medium
Audio-on-demand	Low	Low	High	Medium
Video-on-demand	Low	Low	High	High
Voice over IP	Low	High	High	Low
Video over IP	Low	High	High	High

30.30.3 Flow Classes

Based on the flow characteristics, we can classify flows into groups, with each group having the required level of each characteristic. The Internet community has not yet defined such a classification formally.

Although the Internet has not defined flow classes formally, the ATM protocol does. As per ATM specifications, there are five classes of defined service.

a. Constant Bit Rate (CBR). Eg: telephone traffic

b. Variable Bit Rate-Non Real Time (VBR-NRT). Eg. Multimedia e-mail
c. Variable Bit Rate-Real Time (VBR-RT). Eg. Interactive video (sensitive to jitter)
d. Available Bit Rate (ABR). (File transfer and e-mail
e. Unspecified Bit Rate (UBR). Internet traffic

30-2 FLOW CONTROL TO IMPROVE QoS

Although formal classes of flow are not defined in the Internet, an IP datagram has a ToS field that can informally define the type of service required for a set of datagrams sent by an application.

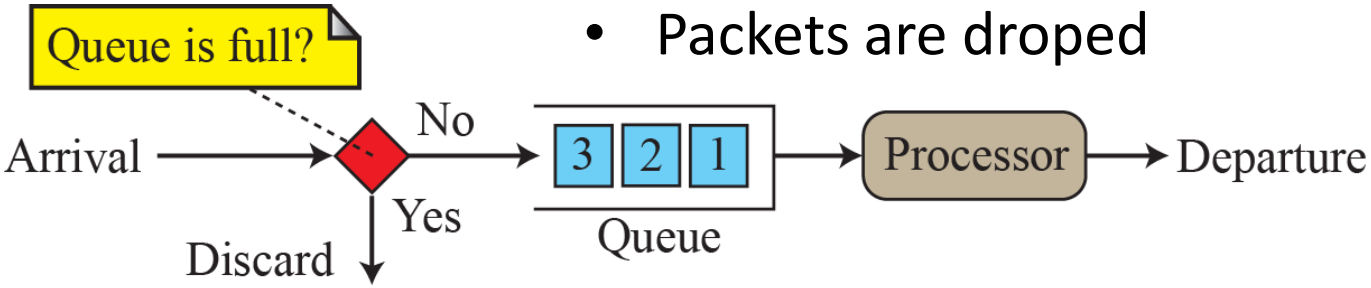
If we assign a certain type of application a single level of required service, we can then define some provisions for those levels of service. These can be done using several mechanisms.

30.2.1 Scheduling

Treating packets (datagrams) in the Internet based on their required level of service can mostly happen at the routers. It is at a router that a packet may be delayed, suffer from jitters, be lost, or be assigned the required bandwidth. A good scheduling technique treats the different flows in a fair and appropriate manner. Several scheduling techniques are designed to improve the quality of service. We discuss three of them here: FIFO queuing, priority queuing, and weighted fair queuing.

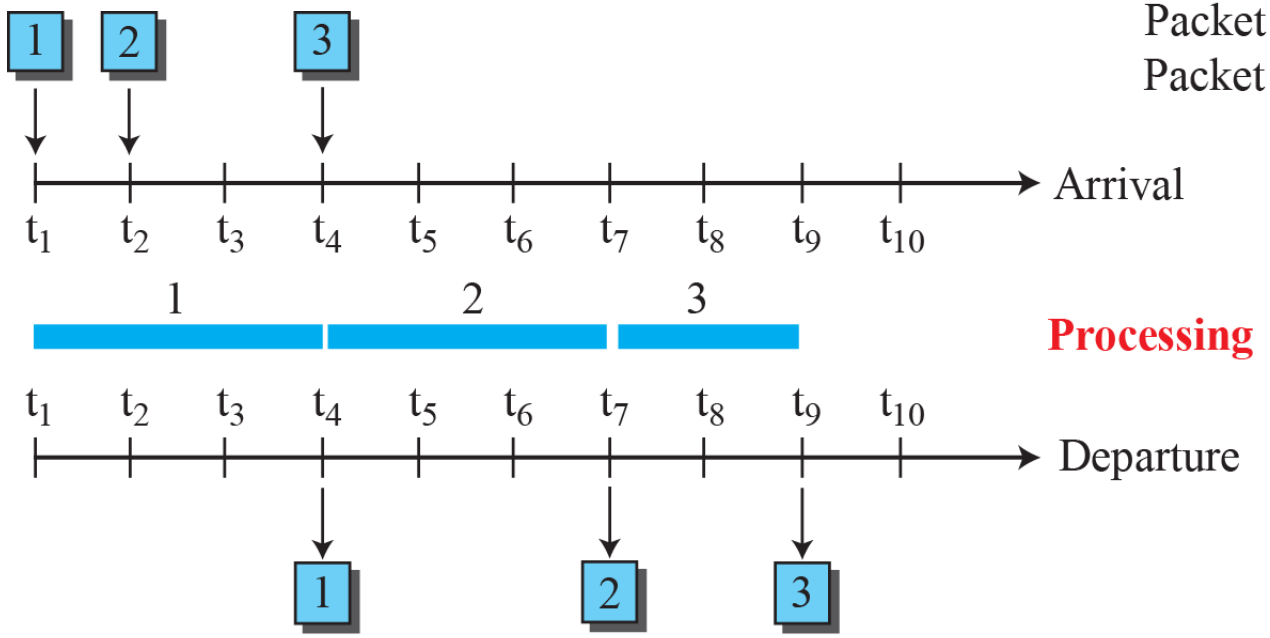
Figure 30.1 : FIFO queue

- If arrival rate is higher than processing rate
  - Packets are droped



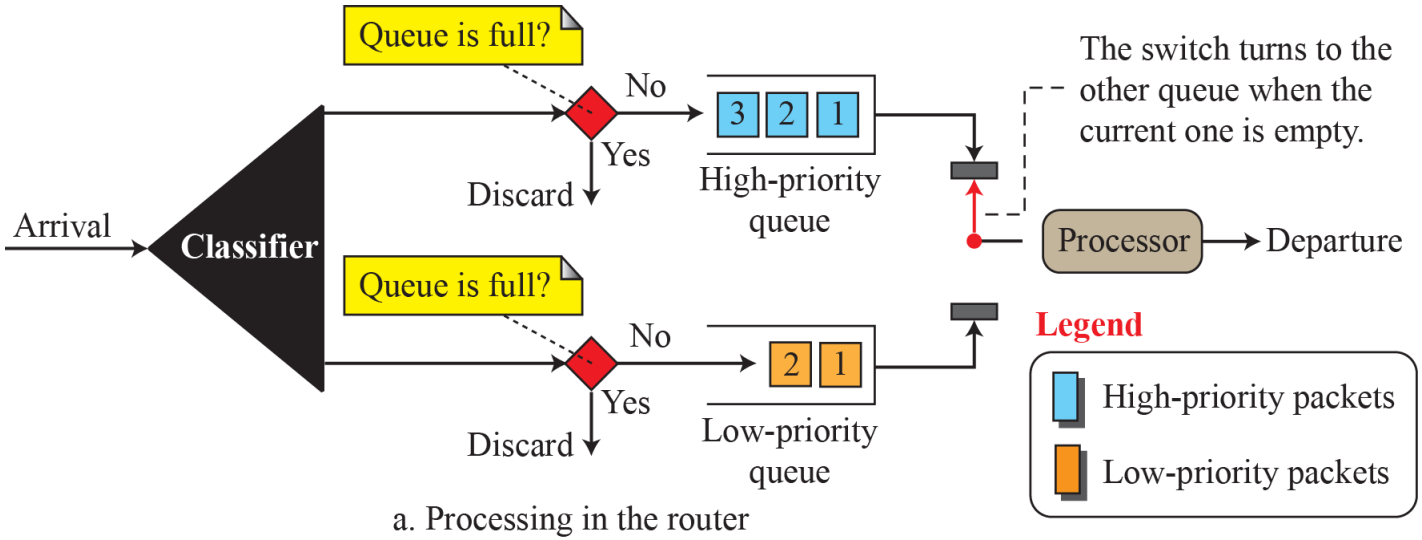
a. Processing in the router

Required processing time
Packet 1: three time units
Packet 2: three time units
Packet 3: two time units



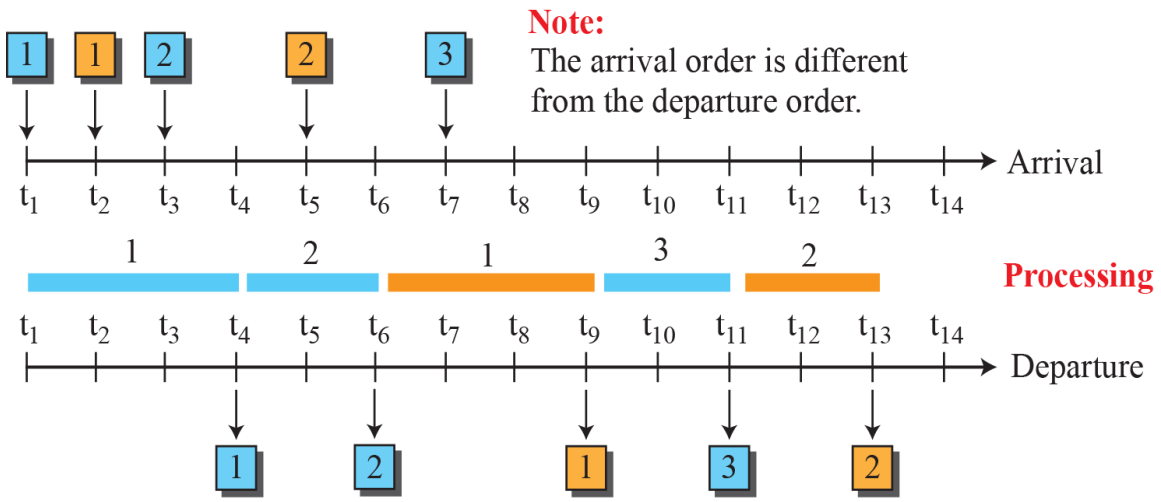
b. Arrival and departure time

Figure 30.2 : Priority queuing



a. Processing in the router

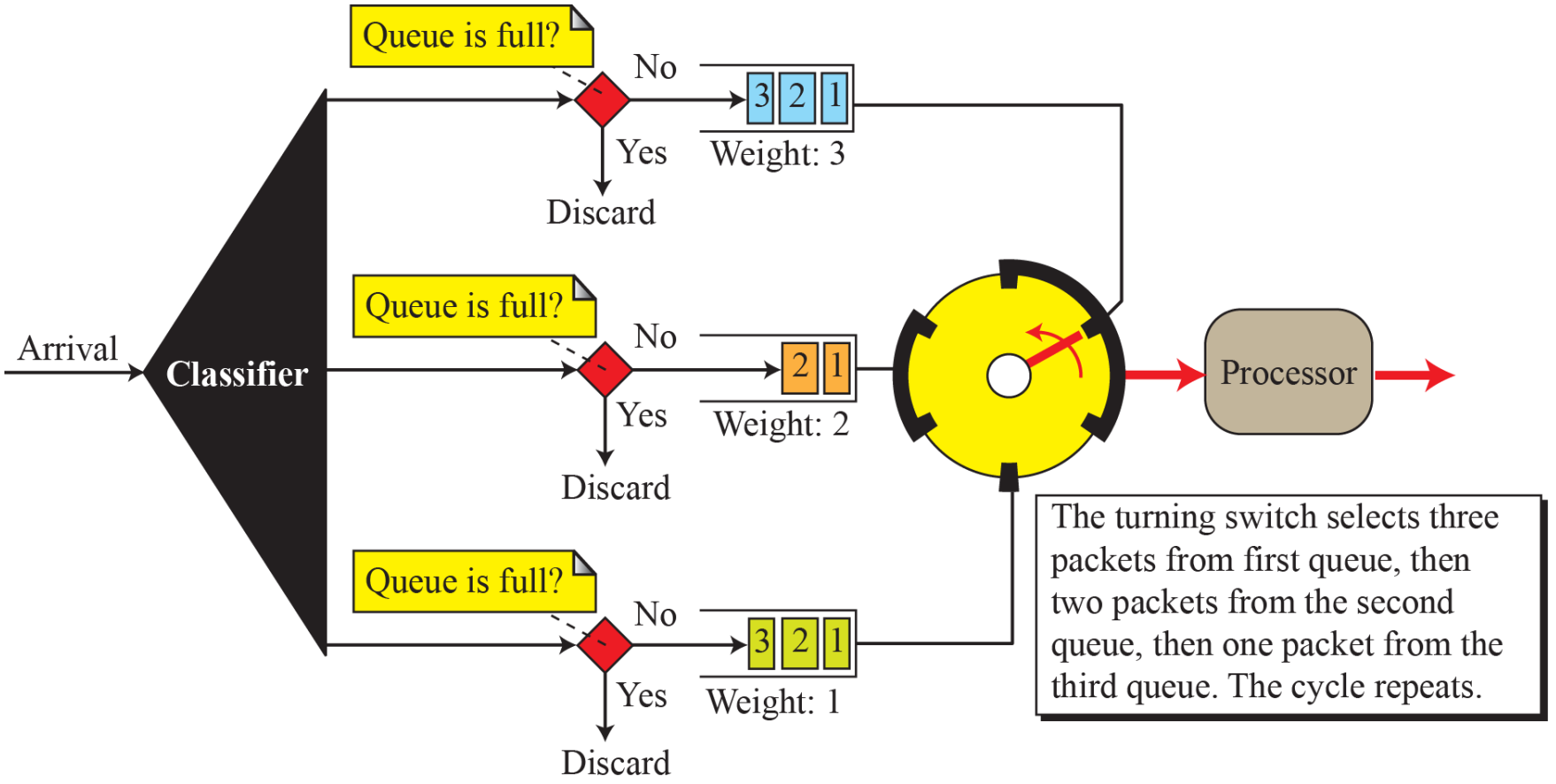
- Time sensitive traffic is put in high priority Q
  - Provides less delay
- Can lead to starvation



b. Arrival and departure time

Figure 30.3 : Weighted fair queuing

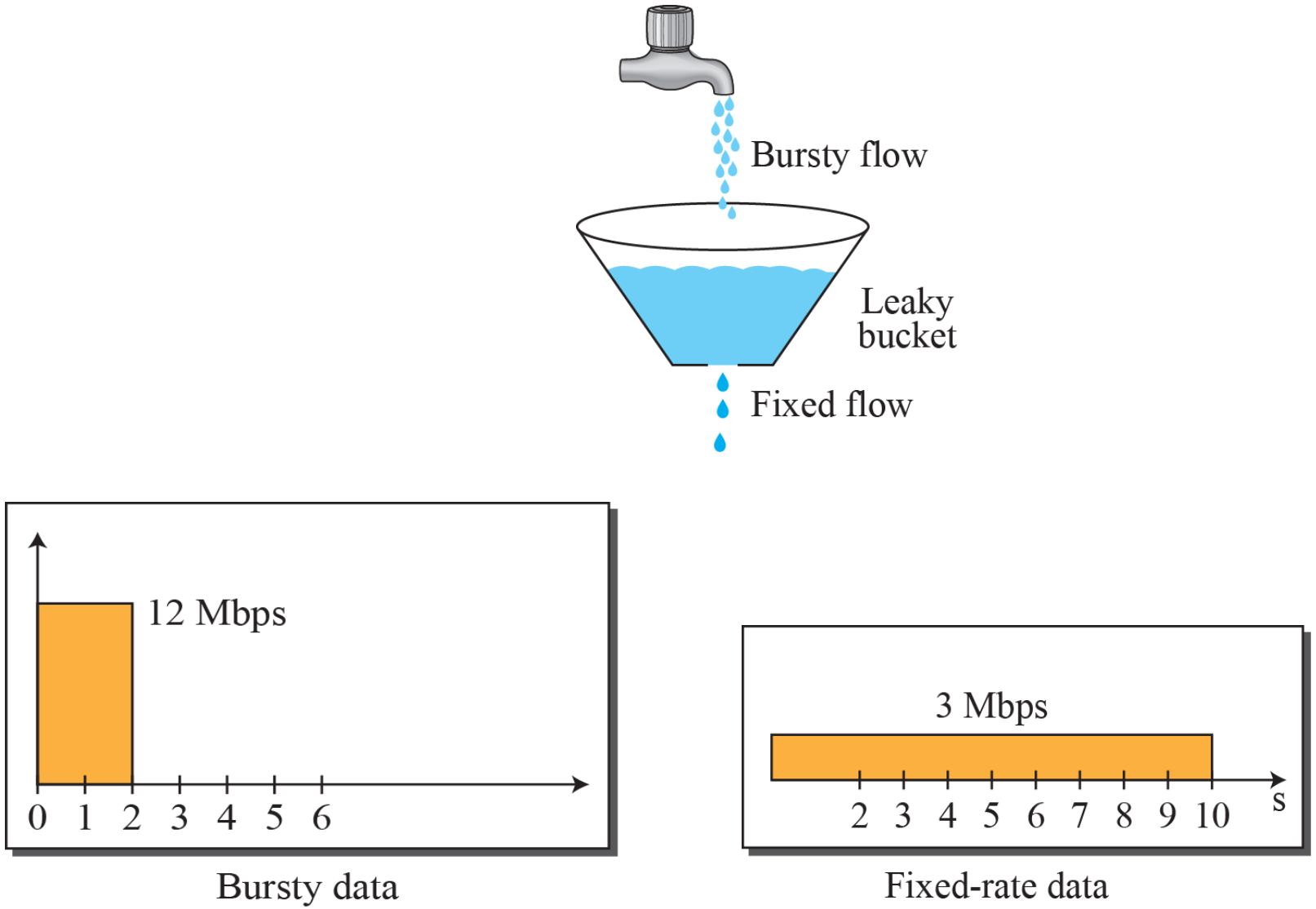
- Addresses starvation



30.2.2 Traffic Shaping or Policing

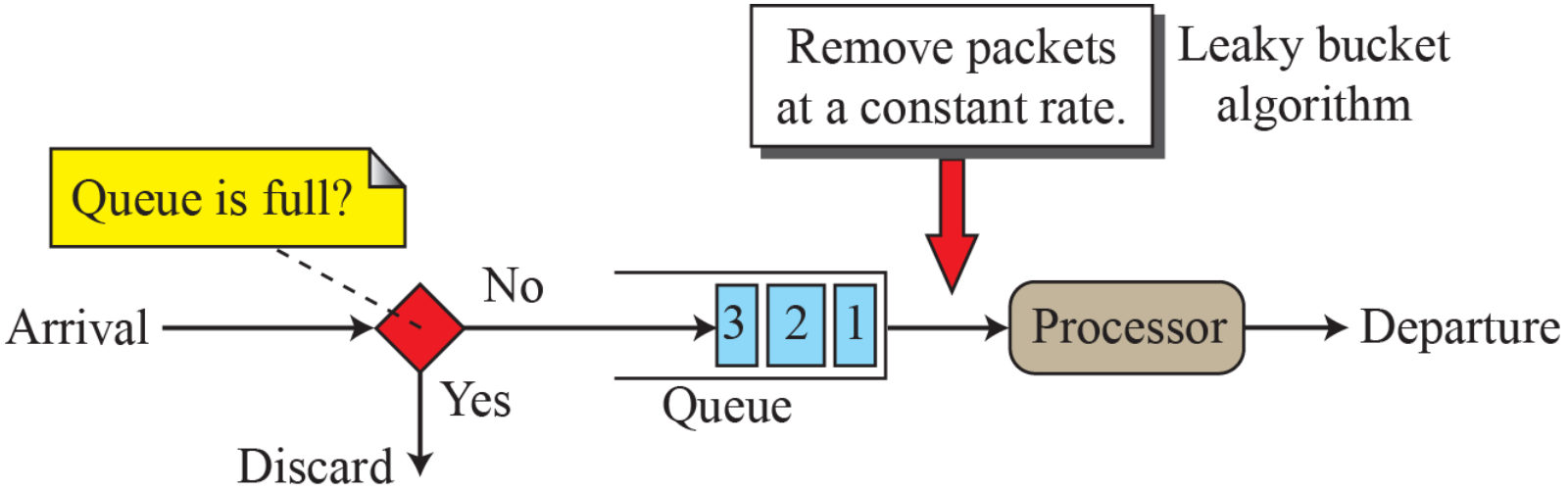
To control the amount and the rate of traffic is called traffic shaping or traffic policing.
The first term is used when the traffic leaves a network; the second term is used when the data enters the network.
Two techniques can shape or police the traffic: leaky bucket and token bucket.

Figure 30.4: Leaky bucket



- **Leaky Bucket**
  - Input rate can vary, o/p rate remains fixed
  - Smoothens out bursty traffic
    - **Shapes into fixed rate traffic** by averaging
    - Helps in preventing congestion
    - Bursty chunks are stored in the bucket and sent out at an average rate
  - **Leaky Bucket Algorithm**
    - Initialize a counter to N at each clock tick
    - Send up to N bytes of data packets if data exists.
      - Decrement the counter with each packet transmission
    - Reset the counter at next clock tick
  - Drops packet is bucket is full
  - No benefit for a host being idle

Figure 30.5: 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.

- **Token Bucket**
  - Leaky bucket is restrictive
    - When a host has no packets
    - Does not get any benefits for later transmissions
  - **Token Bucket Allows for accumulation of credits**
  - **Algorithm**
    - For each clock tick, system adds N tokens
    - One token is consumed for each packet (byte) transmitted
  - Allows bursty traffic as tokens are available

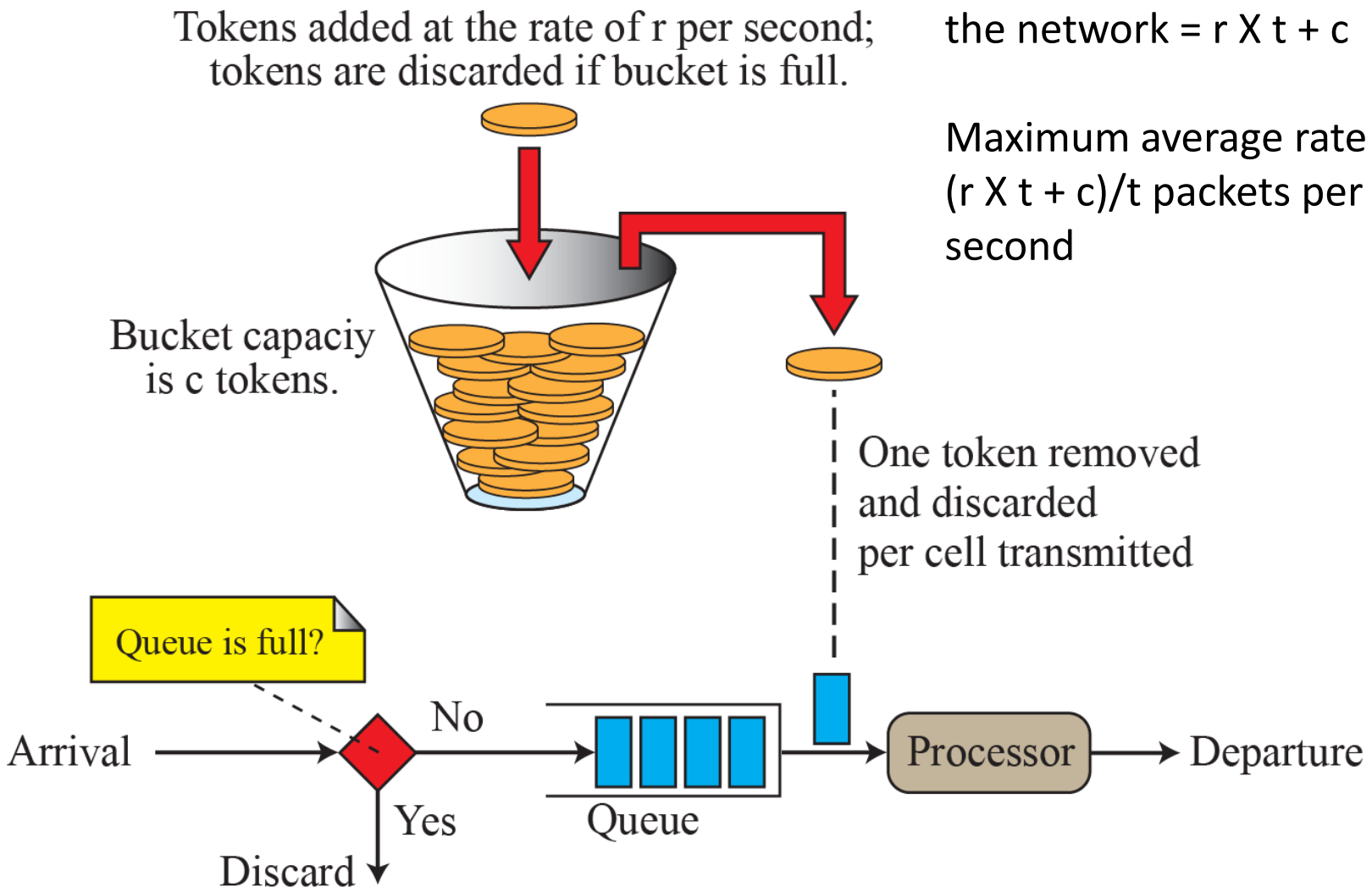
Note

The token bucket allows bursty traffic at a regulated maximum rate.

Figure 30.6 : Token bucket

Maximum number of packets that can enter the network = r X t + c

Maximum average rate = (r X t + c)/t packets per second



Example 30.2

Let assume that the bucket capacity is 10,000 tokens and tokens are added at the rate of 1000 tokens per second. If the system is idle for 10 seconds (or more), the bucket collects 10,000 tokens and becomes full. Any additional tokens will be discarded. The maximum average rate is shown below.

$$\text{Maximum average rate} = (1000t + 10{,}000)/t$$

Token Bucket Algorithm is implemented by a counter.

Counter is initialized to zero.

Each time a token is added, counter is incremented by 1.

Each time a unit of data sent, the counter is decremented by 1

Transmission stops when counter is zero

Combining Token Bucket and Leaky Bucket

- Give credit to idle host and at the same time regulate the traffic.
- Apply leaky bucket after token bucket

30.2.3 Resource Reservation

A flow of data needs resources such as a buffer, bandwidth, CPU time, and so on.
The quality of service is improved if these resources are reserved beforehand.
Below, we discuss a QoS model called Integrated Services, which depends heavily on resource reservation to improve the quality of service.

30.2.4 Admission Control

Admission control refers to the mechanism used by a router or a switch to accept or reject a flow based on predefined parameters.

Before a router accepts a flow for processing, it checks the flow specifications to see if its capacity can handle the new flow.

It takes into account bandwidth, buffer size, CPU speed, etc., as well as its previous commitments to other flows.

30-3 INTEGRATED SERVICES (INTSERV)

IETF developed the Integrated Services (IntServ) model. In this model, which is a flow-based architecture, resources such as bandwidth are explicitly reserved for a given data flow.

In other words, the model is considered a specific requirement of an application in one particular case regardless of the application type.

RFC 2210

Defines how RSVP can be used to make reservations using these classes

- IP Network
  - A best effort service
  - How to implement network that provides QoS to application
    - E.g. real time service with a max delay of 100ms
  - The model is based on **three schemes**
  - The packets are classified as per their service request
  - Uses scheduling to forward the packets according to their flow characteristics
  - Implement admission control
    - Allows some, deny others based on previous commitments

- Model is flow based – all accommodations need to be made before the flow starts
- Need a connection oriented service
- But IP is connectionless
- We use RSVP(Resource Reservation Protocol) on top of IP

30.3.1 Flow Specification

We said that IntServ is flow-based. To define a specific flow, a source needs to define a flow specification, which is made of two parts:

Rspec (resource specification). Rspec defines the resource that the flow needs to reserve (buffer, bandwidth, etc.).

Service specific

Comparatively easy to describe

Tspec (traffic specification). Tspec defines the traffic characterization of the flow, which we discussed before.

30.3.2 Admission

After a router receives the flow specification from an application, it decides to admit or deny the service.
The decision is based on the previous commitments of the router and the current availability of the resource.

30.3.3 Service Classes

Two classes of services have been defined for Integrated Services: guaranteed service and controlled-load service.

Guaranteed Service class

- Specify delay target for guaranteed service
- Designed for real-time traffic that needs a minimum guaranteed minimum end-to-end delay
- Quantitative Type of Service in which the amount of end-to-end delay and data rate is defined by the application eg: VoIP

Controlled -Load service class

- Some delays are acceptable but sensitive to overloaded networks – eg. File transfer, email and internet access
- Qualitative Type of Service
- Expects low loss or no loss

30.3.4 RSVP

Since IP is a connectionless protocol, a new protocol is designed to run on top of IP to make it connection-oriented.

A connection-oriented protocol needs to have connection establishment and connection termination phases.

Before discussing RSVP, we need to mention that it is an independent protocol separate from the Integrated Services model. It may be used in other models in the future.

- RSVP
  - Providing reservation is easier in connection oriented network
    - At Admission control one can decide
  - Assumptions for connectionless network (IP)
    - Should not disturb the robustness
      - A router can go down and come back
      - Packets continue to get routed
    - Routers do not store any state information
      - Connection oriented network store hard state
      - If a switch goes down, so does the connection
        - RSVP is a signalling protocol to help IP create a flow(a kind of VC) and consequently make a resource reservation

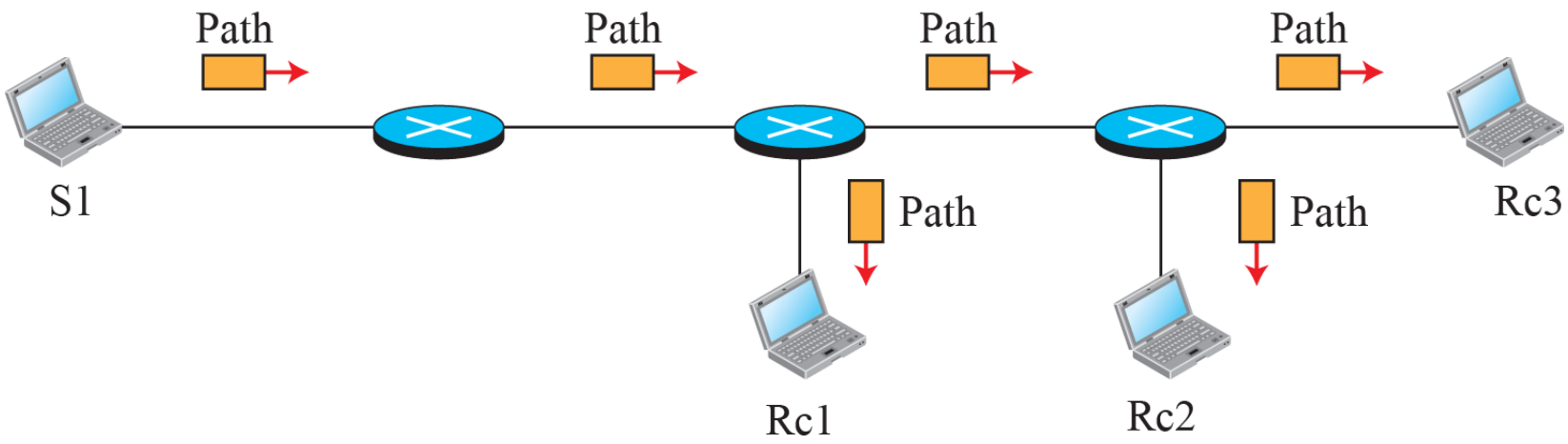
- RSVP…
  - Need to have some soft state
    - Does not need to be explicitly deleted
      - Like in connection oriented network
    - Periodically refreshed (currently refresh time is 30 secs)
    - Router down affects it temporarily
    - Allows new path to be discovered
  - Designed for Multicasting
    - As effectively as unicast flows
    - Many of new (multimedia) applications require
    - Have many more receivers than senders

- RSVP…
  - Takes receiver oriented approach**(Receiver based reservation)**
  - Connected oriented n/w usually sender oriented
    - Phone n/w: sender causes resource reservation
  - Benefit of Receiver oriented and soft state
    - Easy to increase/decrease the resource allocation
    - Periodic refresh for soft state
      - Allows for new reservation of resource requirements

- RSVP…

– Receiver makes the reservation using messages (there are many messages available, we are discussing two here)
  - sender
    - Sends the **PATH message** in the multicast path
    - Intermediary devices stores the necessary information for the receivers
    - A new message is created when the path diverges
  - Receiver sends **RESV message (upstream)**
    - Reservation done at each router
    - Resv message travels towards the sender from the receiver

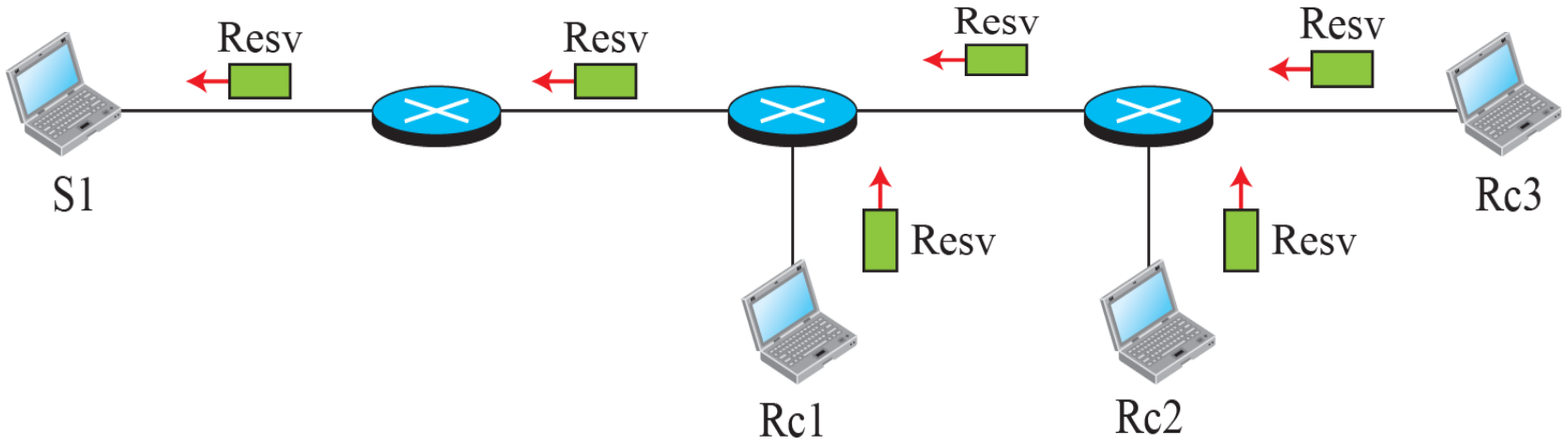
Figure 30.7 : Path messages



- Allows router to figure reverse path for RESV message
- Allows to build a multicast tree

Figure 30.8 : Resv messages

- RESV contains Tspec and Rspec (Receiver rqmnts)
  - Each router allocates the required resources
  - Passes the request to next router
  - If router can't allocate the resource
    - Message is sent to the receiver
  - Resource reservation done when it reaches sender



- Handling router/link failure
  - PATH msgs are sent periodically (30s)
  - Will reach the receiver over new path
  - RESV msg will follow the new path
    - Hopefully establish a new reservation
  - Routers on old path will not get PATH/RESV
    - Resources will be released on timeout
  - RSVP deals quite well with topology change
    - Provided not too frequent

- Reservation Merging
  - Resv travels up the multicast tree
  - Will reach some merging point node
    - Where reservation is already done (previous RESV)
  - Nothing to be done if prev RESV was good enough
    - E.g. prev RESV required delay of 100ms
      - New requires delay of 200ms
    - Prev RESV required BW of 2Mbps
      - New requires BW of 1Mbps
  - When new Resv requires better resources
    - Router will allocated the same
  - Essentially, **Resv requirements are merged**

Figure 30.9 : Reservation merging

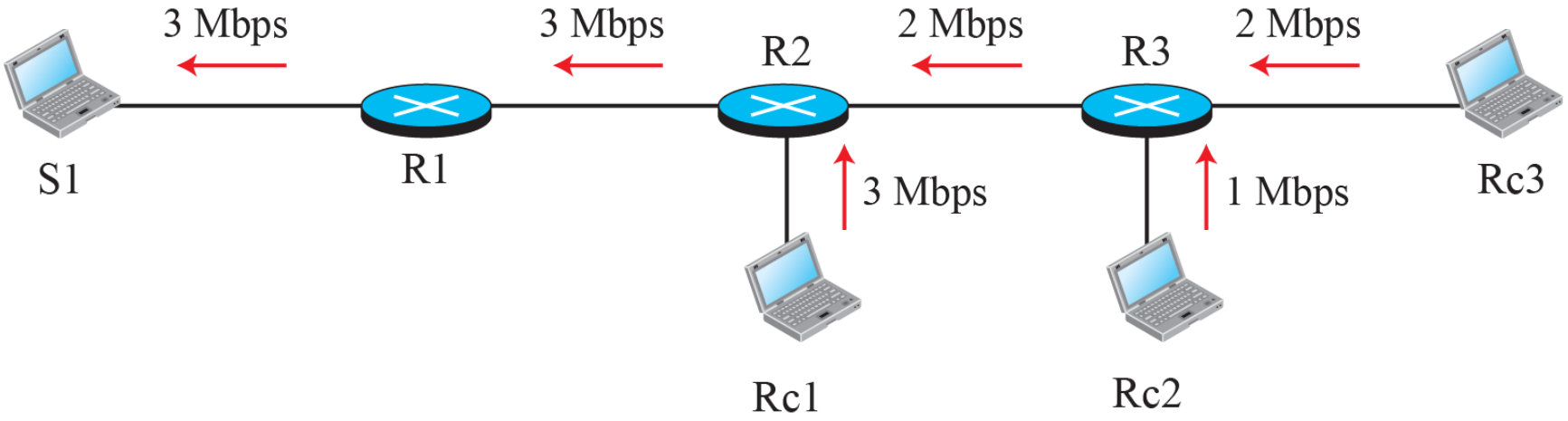
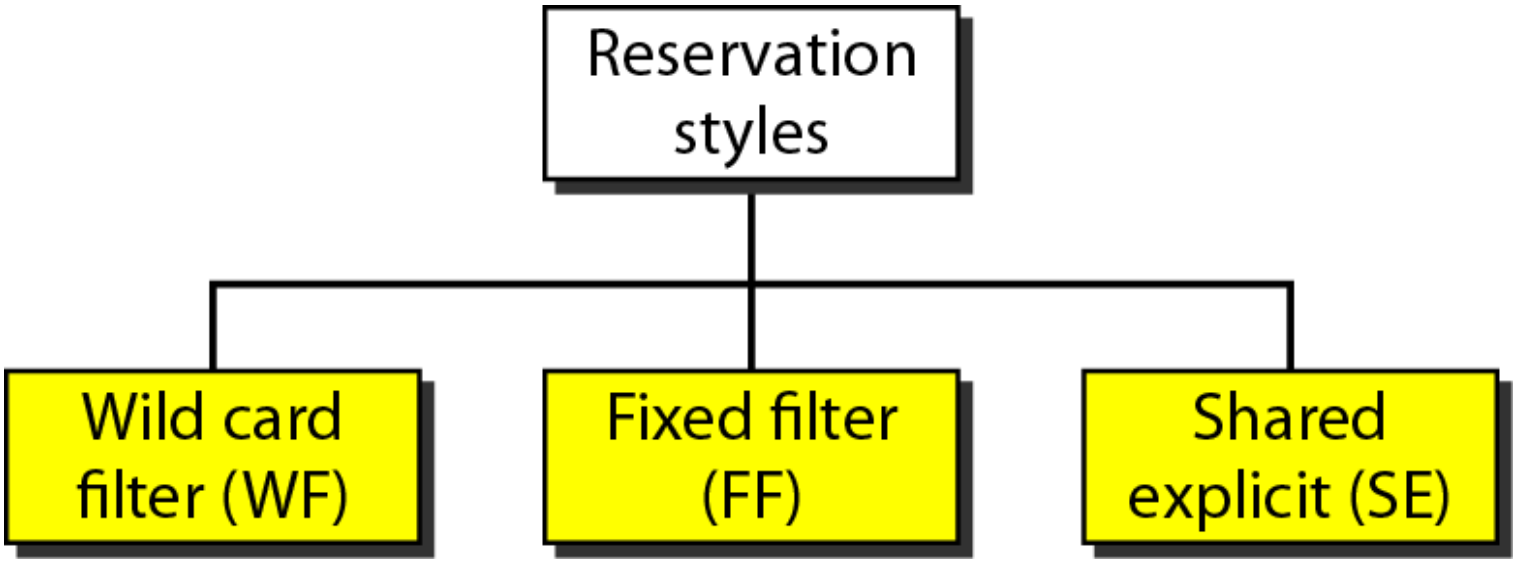


Figure 24.25 Reservation styles

When there is more than one flow, the router need to make reservations for all of them. It uses 3 styles



- Wildcard filter style
  - Single reservation for all senders
  - Reservation based on largest request
    - Works ok when senders do not send at same time
- Fixed Filter Style
  - Created **distinct reservation for each flow**
  - Used when all senders send at same time
- Shared Explicit Style
  - A single reservation for a set of flows
- Soft State
  - Reservation info stored in every node need to be refreshed periodically (currently refresh time is 30 secs)

30.3.5 Problems with Integrated Services

There are at least two problems with Integrated Services that may prevent its full implementation in the Internet: scalability and service-type limitation:

- Issues with IntServ Model
  - Scalability
    - Each router Need to Keep info for each flow
    - Problematic for growing internet (especially for core routers)
  - Service type limitation
    - Provides only two types of service
      - Guaranteed (min end-to-end delay)
      - Control-load (emulates lightly loaded network)
        - » Does not Tolerate packet losses (low loss or no loss)
        - » but some delay is acceptable
    - Newer application may require more type of services

30-3 DIFFERENTIATED SERVICES (DIFFSERV)

In this model, packets are marked by applications into classes according to their priorities.
Routers and switches, using various queuing strategies, route the packets.
This model was introduced by the IETF to handle the shortcomings of Integrated Services.

Differentiated Services is a class-based QoS model designed for IP.

- Changes from Intserv
  - Processing moved from network core to the edge
    - This solved the scalability problem
    - Routers do not store the info about flows
    - Applications/host define type of service needed
  - Per flow service is changed to per class service
    - Routers routes based on class of service (in pkt)
      - Not on flows
      - Solves the service type limitation problem

30.4.1 DS Field

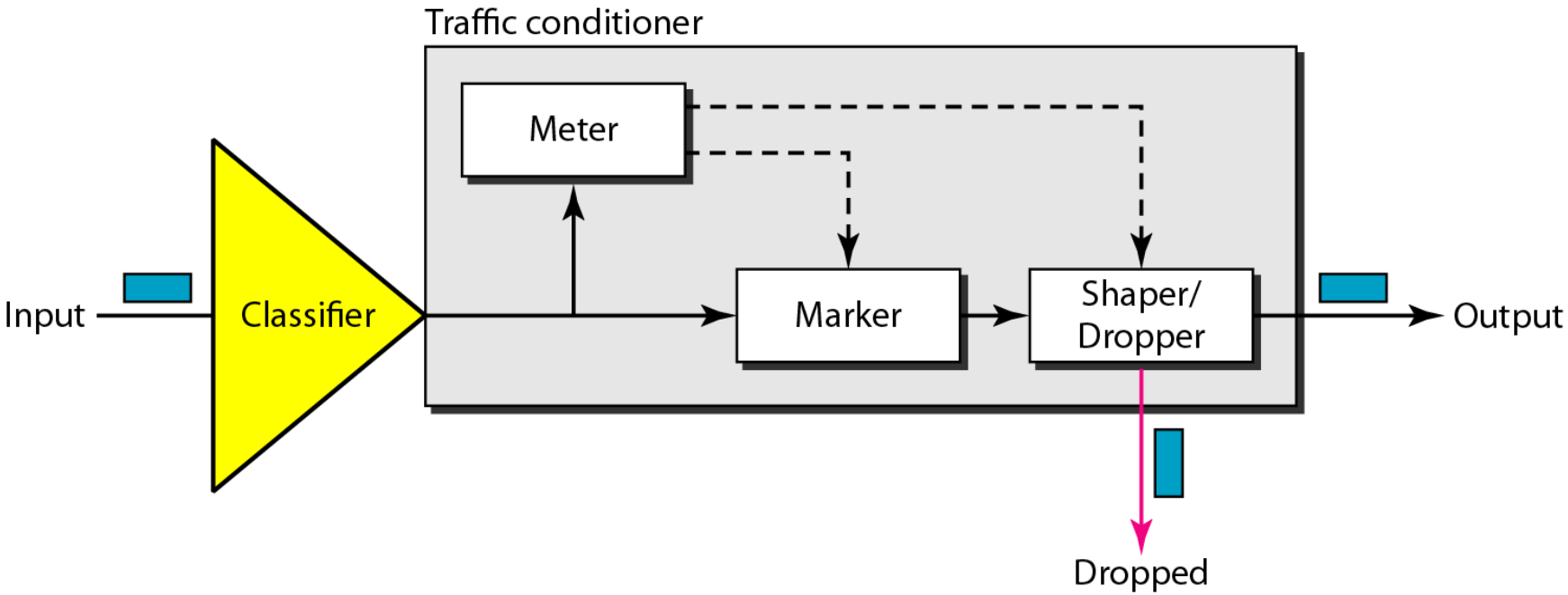
In DiffServ, each packet contains a field called the DS field.
The value of this field is set at the boundary of the network by the host or the first router designated as the boundary router.
IETF proposes to replace the existing ToS (type of service) field in IPv4 or the priority class field in IPv6 with the DS field

Figure 24.26 DS field

- Replaces TOS field in IPv4 or Traffic class field in IPv6
- Value of the field is set at the boundary
  - By the host or first router (boundary router)
- Two fields
  - DSCP: Differentiated Service Code Point
    - Defines per hop behaviour (PHB) (uses 6 bits)
  - CU: Currently Unused

- PHB
  - Defines the behavior of individual router
    - Rather than end to end services
    - Much easier to deal with
  - So far Three types of PHBs are defined
    - **DE PHB : Default PHB**
      - Same as best effort delivery, compatible with TOS
    - **EF PHB: Explicit Forwarding,** provides
      - Low loss
      - Low latency/delay
      - Ensured bandwidth
    - **AF PHB:** Assured Forwarding
      - Forwards packets **as long as traffic does not exceed profile**
      - Some packets may be dropped

Figure 24.27 Traffic conditioners are used to implement DiffServ Model



- Meter checks to see if the flow matches the negotiated traffic profile and give input to other components
- Marker can remark or down-mark a packet
- Shaper reshapes the traffic
- Dropper discards the packet if it severely violates the negotiated profile (Router queue full)

Thank You