Chapter 2: ROAD MAP

- Transport Layer Introduction
- Port Address
- UDP
- TCP
- Socket Programming using TCP and UDP
- SCTP
- RTP
- TCP in wireless network
- Quality of services

Chapter 2: ROAD MAP

- Transport Layer Introduction
- Port Address
- UDP
- TCP
- Socket Programming using TCP and UDP
- SCTP
- RTP
- TCP in wireless network
- Quality of services

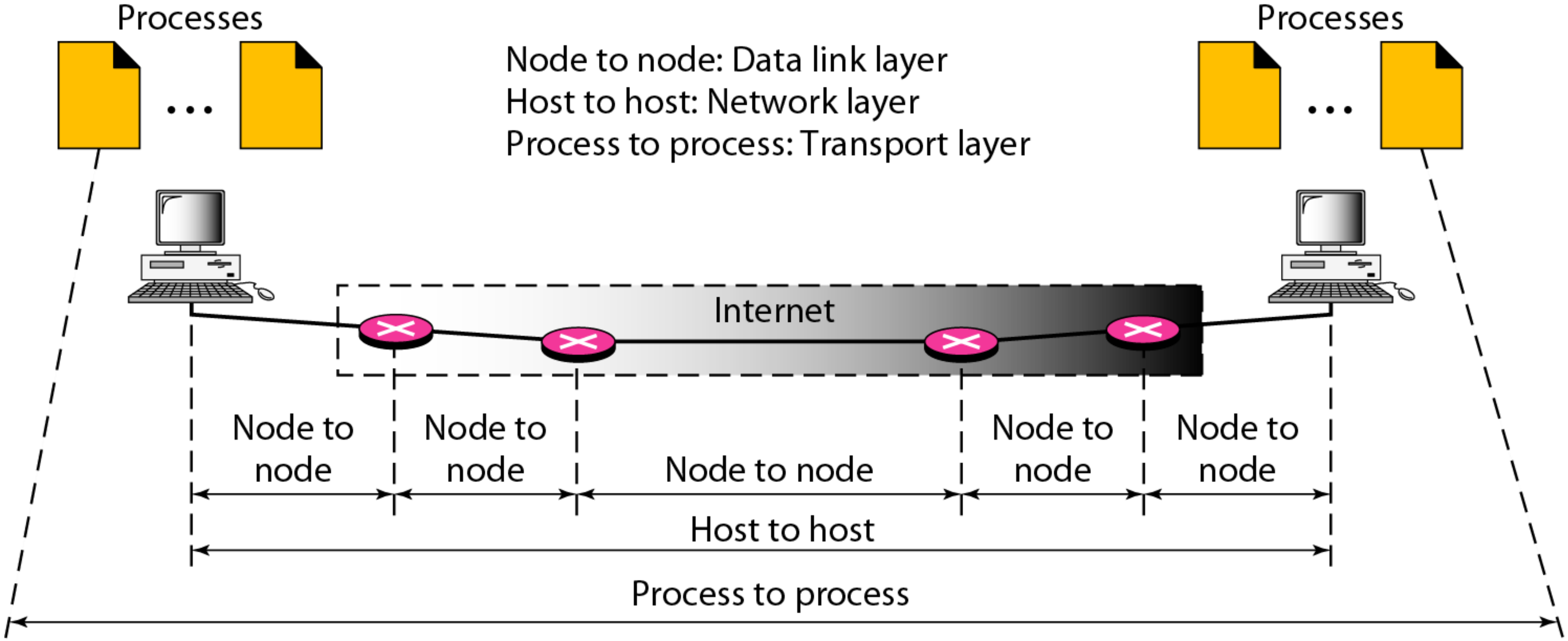
Transport Layer

Services provided by transport layer

- Process to Process delivery
- Connection less as well as connection oriented data delivery
- Error control
- multiplexing/demultiplexing
- reliable data transfer
- flow control
- congestion control

- learn about transport layer protocols in the Internet:
  - UDP: connectionless transport
  - TCP: connection-oriented transport
  - STCP :Combination of TCP and UDP
  - RTP : Real time transport protocol

Process to Process Data Delivery



Addressing

Data Link Layer

- Mac Address (48 bit)
- Physical address
- NIC Card

Network Layer

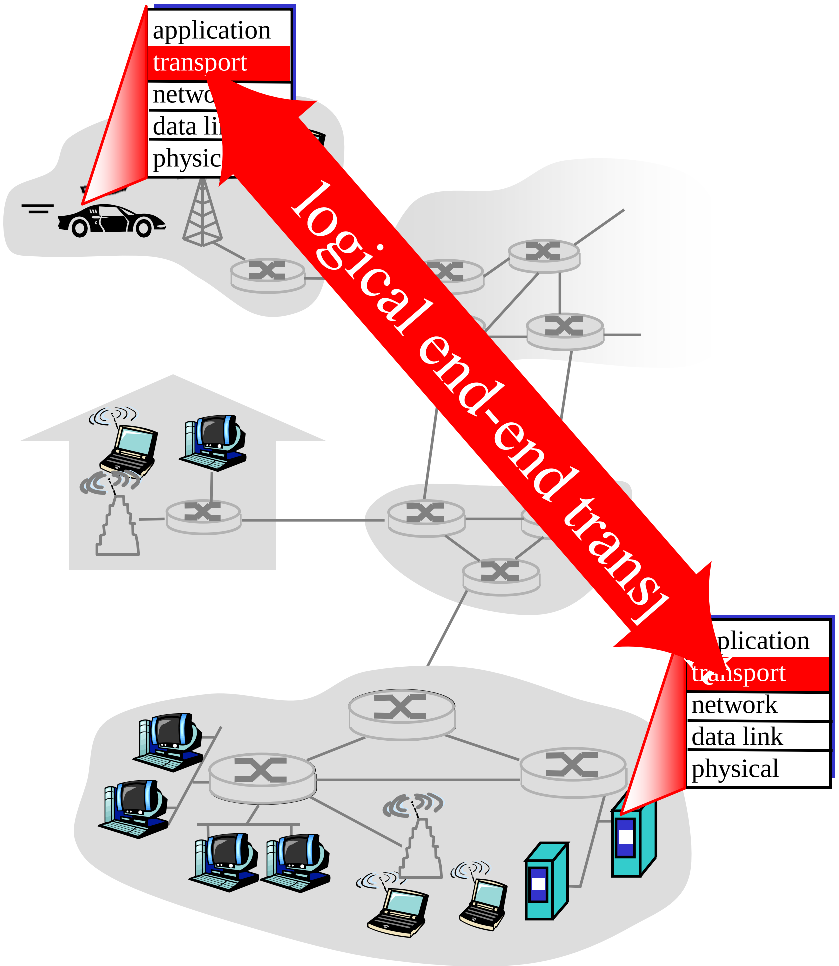
- IP Address (32 or 128 bit)
- Logical Address
- Machine

Transport Layer

- Port Address (16 bit)
- Logical Address
- Application

Transport services and protocols

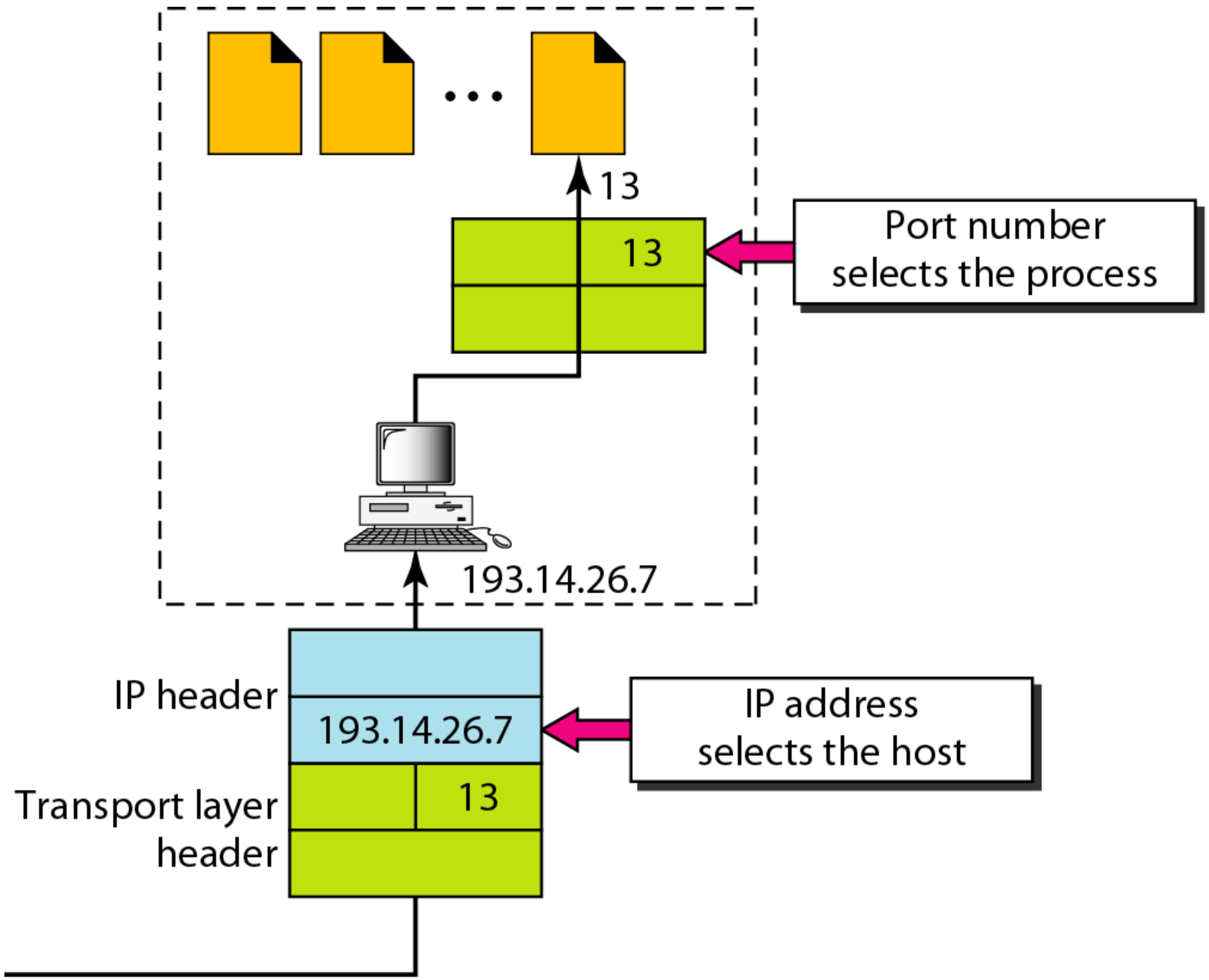
- provide *logical communication* between app processes running on different hosts
- transport protocols run in end systems
  - send side: breaks app messages into segments, passes to network layer
  - rcv side: reassembles segments into messages, passes to app layer
- more than one transport protocol available to apps
  - Internet: TCP and UDP and SCTP



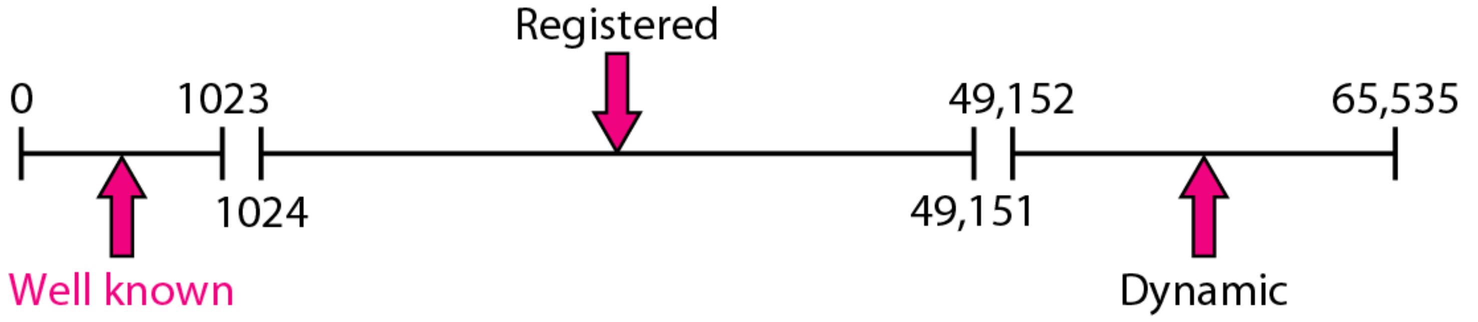
Chapter 2: ROAD MAP

- Transport Layer Introduction
- Port Address
- UDP
- TCP
- Socket Programming using TCP and UDP
- SCTP
- RTP
- TCP in wireless network
- Quality of services

IP addresses versus port numbers



PORT ranges by IANA *(Internet Assigned Number Authority)*



Port Ranges by IANA

Well Known

- From 0-1023
- Assigned & controlled by IANA
- These are port no.s for servers ex. FTP(20,21),SMTP (25)

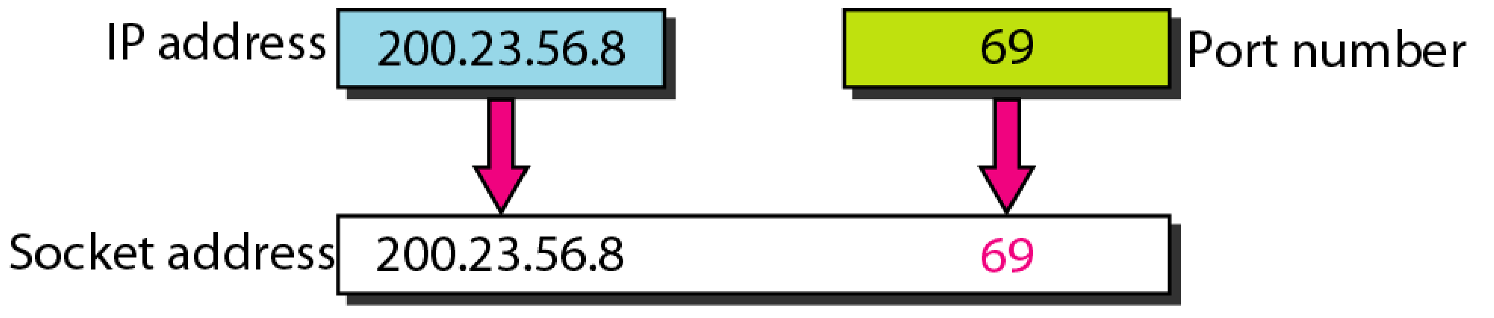
Registered

- From 1024-49151
- Not assigned & controlled by IANA
- Can only be registered with IANA
- Ex. MySQL(3306), MongoDB (27017)

Dynamic

- From 49152-65535
- Nighter controlled nor registered by IANA
- They can be used by any client Program(Process)

Socket address



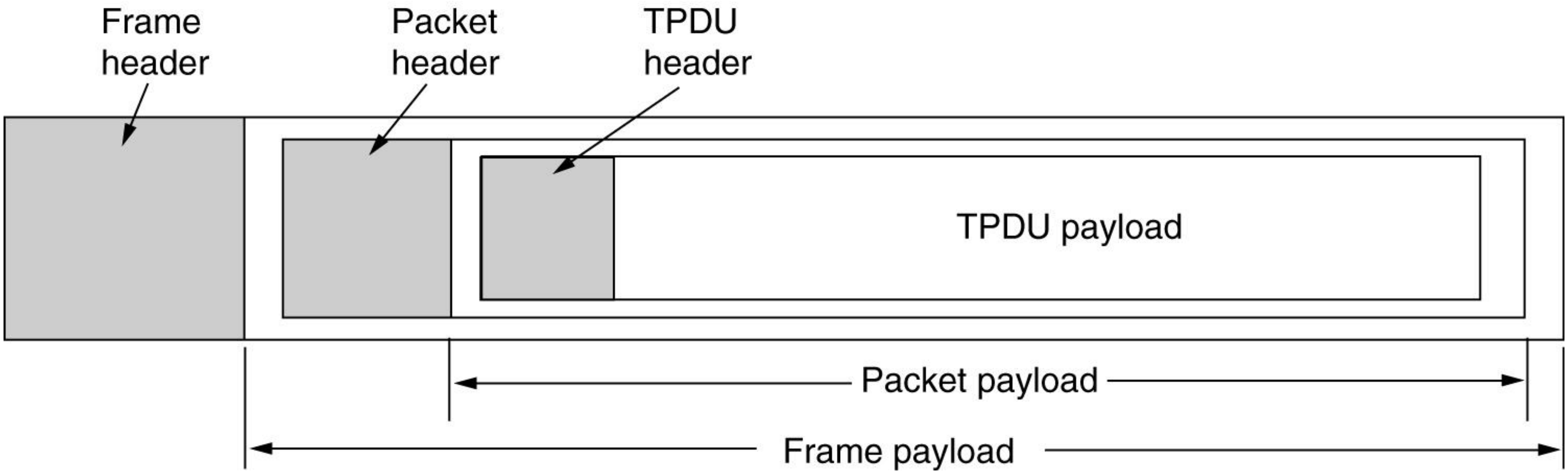
Transport Service Primitives

Primitive	Packet sent	Meaning
LISTEN	(none)	Block until some process tries to connect
CONNECT	CONNECTION REQ.	Actively attempt to establish a connection
SEND	DATA	Send information
RECEIVE	(none)	Block until a DATA packet arrives
DISCONNECT	DISCONNECTION REQ.	This side wants to release the connection

The primitives for a simple transport service.

Transport Service Primitives (2)

The nesting of TPDUs, packets, and frames.

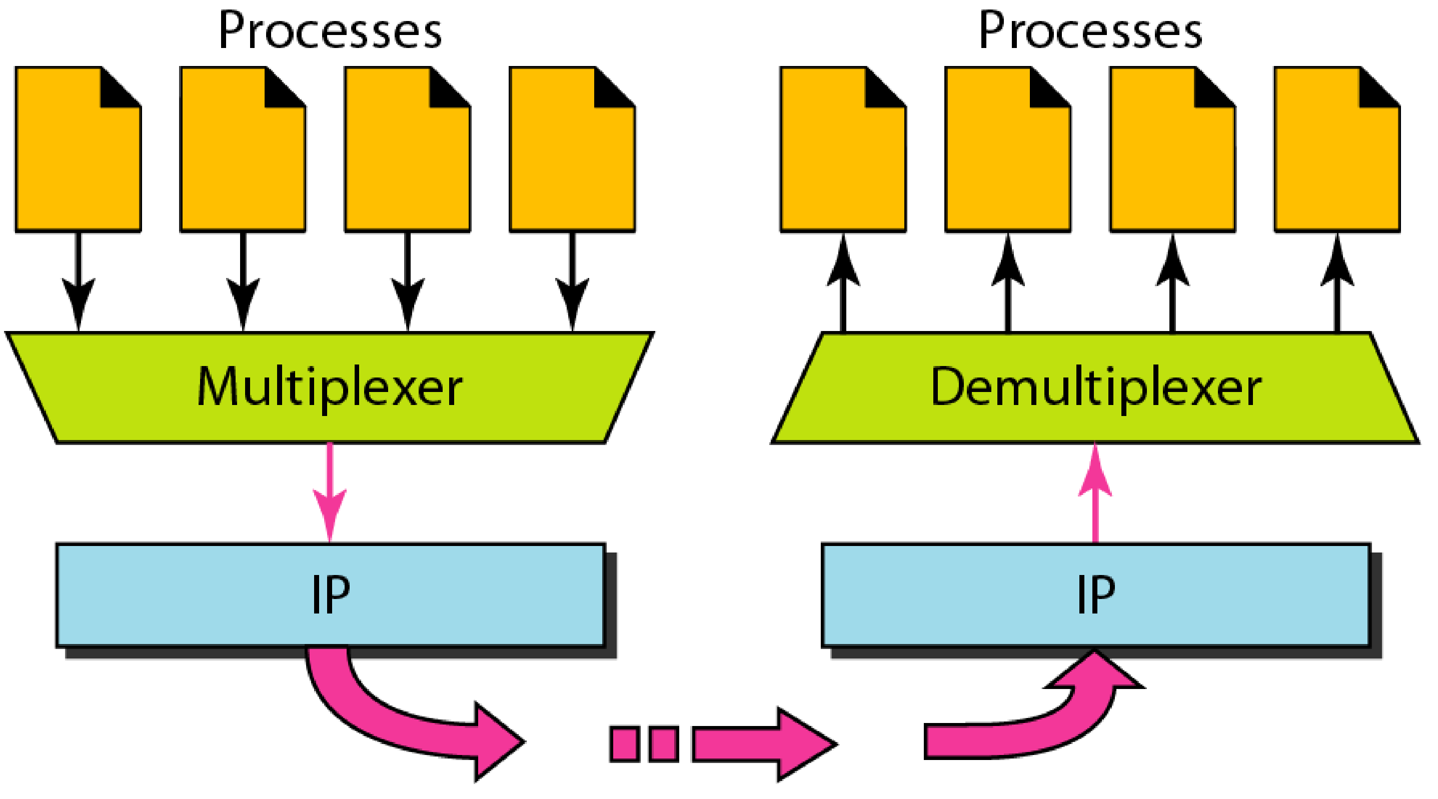


Berkeley Sockets

The socket primitives for TCP.

Primitive	Meaning
SOCKET	Create a new communication end point
BIND	Attach a local address to a socket
LISTEN	Announce willingness to accept connections; give queue size
ACCEPT	Block the caller until a connection attempt arrives
CONNECT	Actively attempt to establish a connection
SEND	Send some data over the connection
RECEIVE	Receive some data from the connection
CLOSE	Release the connection

Multiplexing and demultiplexing



Multiplexing/demultiplexing

Multiplexing at send host:

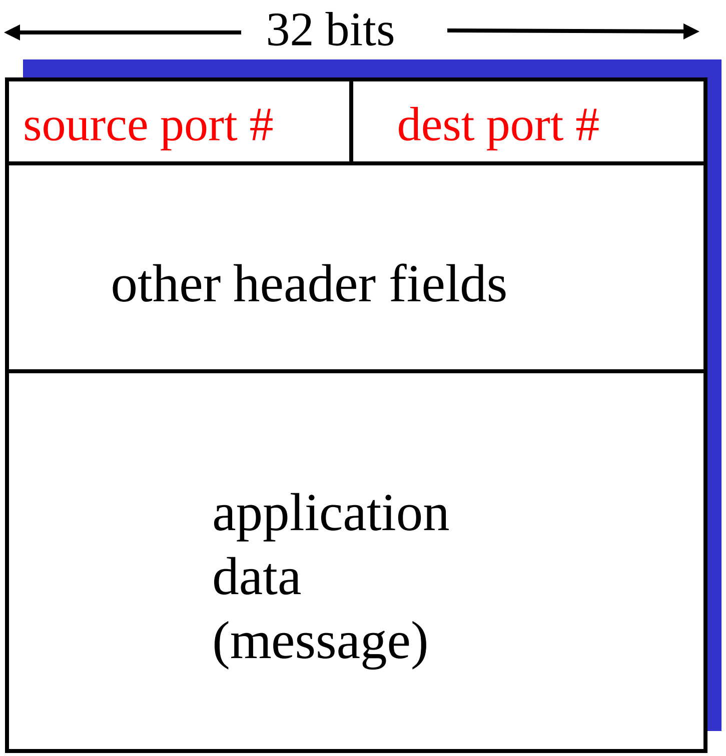
gathering data from multiple sockets, enveloping data with header (later used for demultiplexing)

Demultiplexing at rcv host:

delivering received segments to correct socket

How demultiplexing works

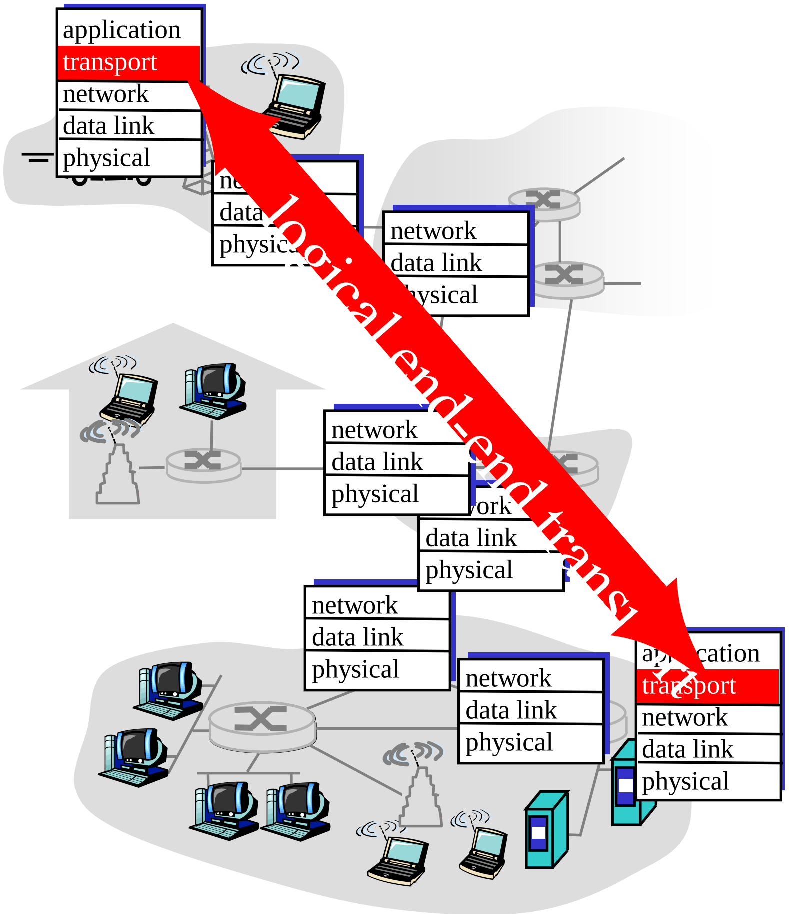
- host receives IP datagrams
  - each datagram has source IP address, destination IP address
  - each datagram carries 1 transport-layer segment
  - each segment has source, destination port number
- host uses IP addresses & port numbers to direct segment to appropriate socket



TCP/UDP segment format

Internet transport-layer protocols

- reliable, in-order delivery (TCP)
  - congestion control
  - flow control
  - connection setup
- unreliable, unordered delivery: UDP
  - Faster data delivery
- Stream Control Transmission Protocol (SCTP):
  - Faster and reliable data delivery



Chapter 2: ROAD MAP

- Transport Layer Introduction
- Port Address
- UDP (User datagram protocol)
- TCP
- Socket Programming using TCP and UDP
- SCTP
- RTP
- TCP in wireless network
- Quality of services

USER DATAGRAM PROTOCOL (UDP)

The User Datagram Protocol (UDP) is called a connectionless, unreliable transport protocol. It does not add anything to the services of IP except to provide process-to-process communication instead of host-to-host communication.

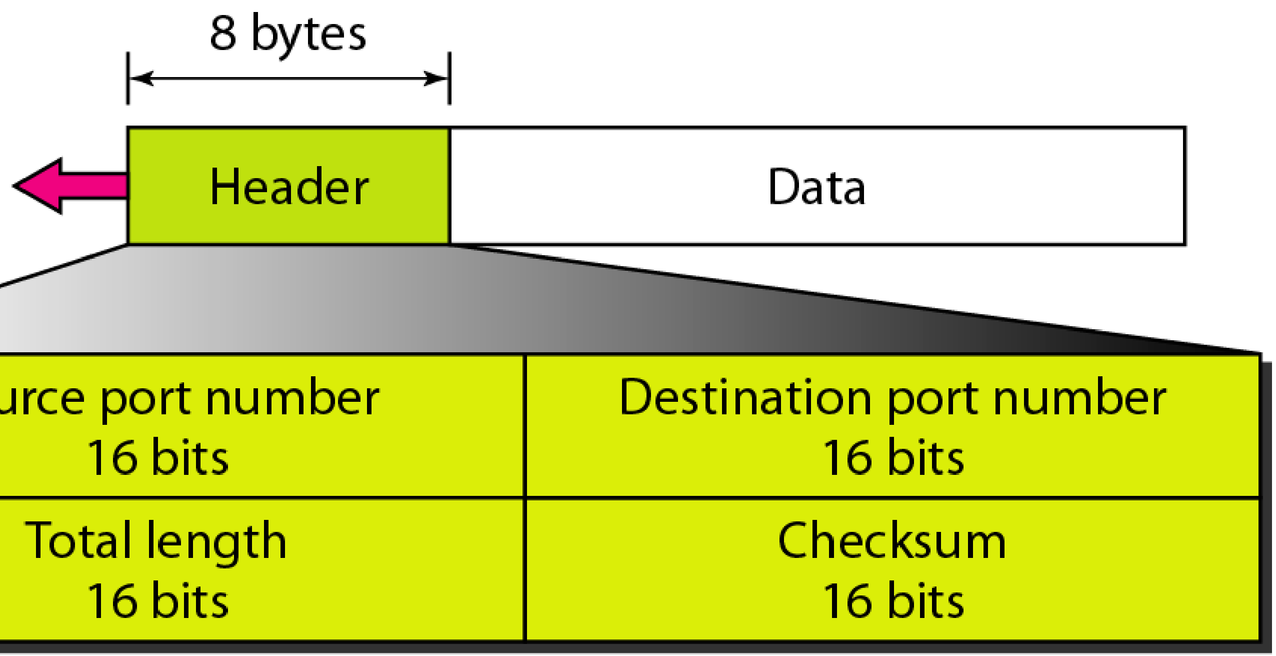
Topics discussed in this section:

Well-Known Ports for UDP
User Datagram
Checksum
UDP Operation
Use of UDP

Well-known ports used with UDP

Port	*Protocol*	*Description*
7	Echo	Echoes a received datagram back to the sender
9	Discard	Discards any datagram that is received
11	Users	Active users
13	Daytime	Returns the date and the time
17	Quote	Returns a quote of the day
19	Chargen	Returns a string of characters
53	Nameserver	Domain Name Service
67	BOOTPs	Server port to download bootstrap information
68	BOOTPc	Client port to download bootstrap information
69	TFTP	Trivial File Transfer Protocol
111	RPC	Remote Procedure Call
123	NTP	Network Time Protocol
161	SNMP	Simple Network Management Protocol
162	SNMP	Simple Network Management Protocol (trap)

User datagram format (UDP Header Format)



UDP Pseudo Header

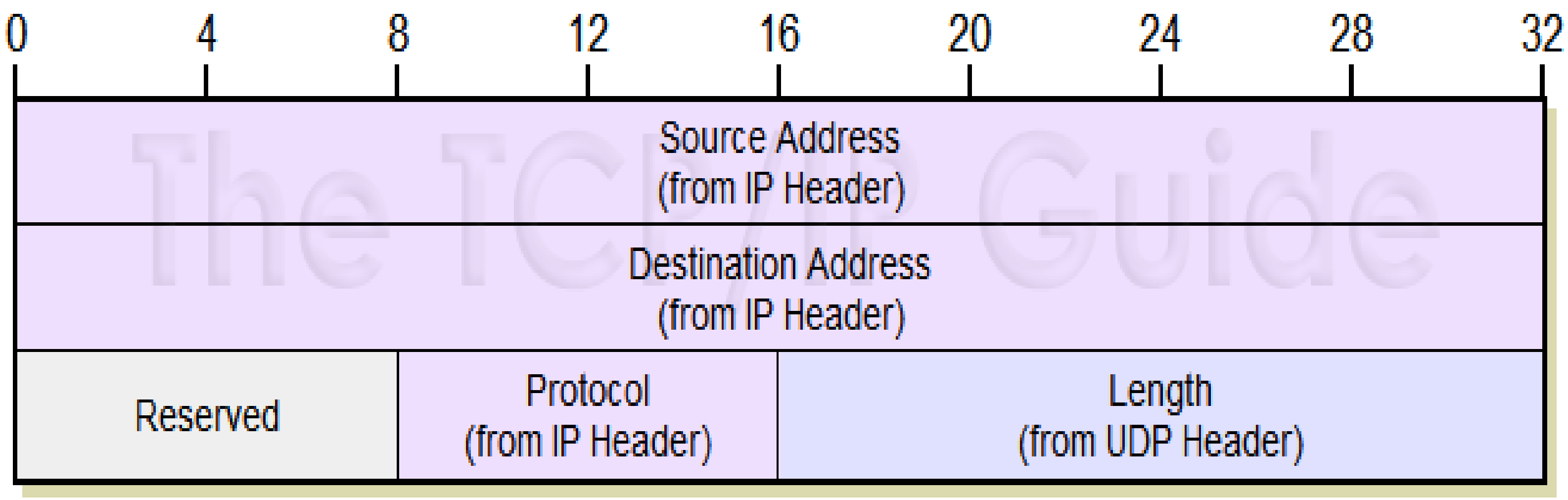
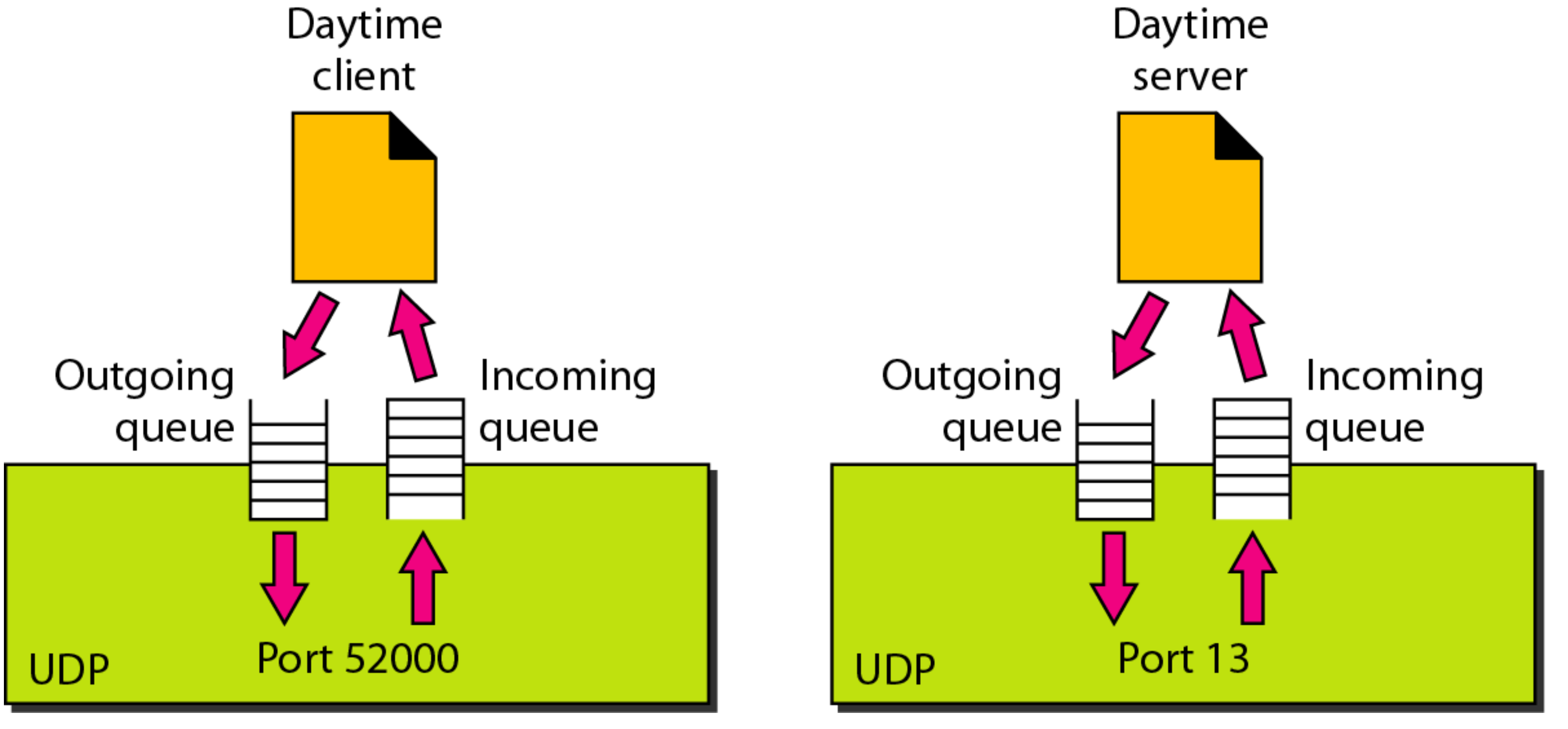


Figure : UDP Pseudo Header Format

UDP Operations

- Connectionless service
- No Flow and error control except checksum
- Encapsulation and Decapsulation of messages in IP datagram
- Queing

Queues in UDP



Uses of UDP

- Simple Request reply communication
- Suitable for process with internal flow and control mechanisms. Eg. TFTP
- The Real-Time Transport Protocol
- Used in route updating protocol like Routing Information Protocol(RIP)
- Remote Procedure Call(RPC)
- Suitable for Multicasting. Multicasting capability is inbuilt in UDP software's

Chapter 2: ROAD MAP

- Transport Layer Introduction
- Port Address
- UDP
- TCP (Transmission control protocol)
- Socket Programming using TCP and UDP
- SCTP
- RTP
- TCP in wireless network
- Quality of services

TCP (Transmission control protocol)

TCP is a connection-oriented protocol; it creates a virtual connection between two TCPs to send data. In addition, TCP uses flow and error control mechanisms at the transport level.

Topics discussed in this section:

TSP Vs UDP
TCP Services
TCP Stream delivery
Segment (TCP Header)
A TCP Connection
Flow Control
Error Control

UDP v/s TCP

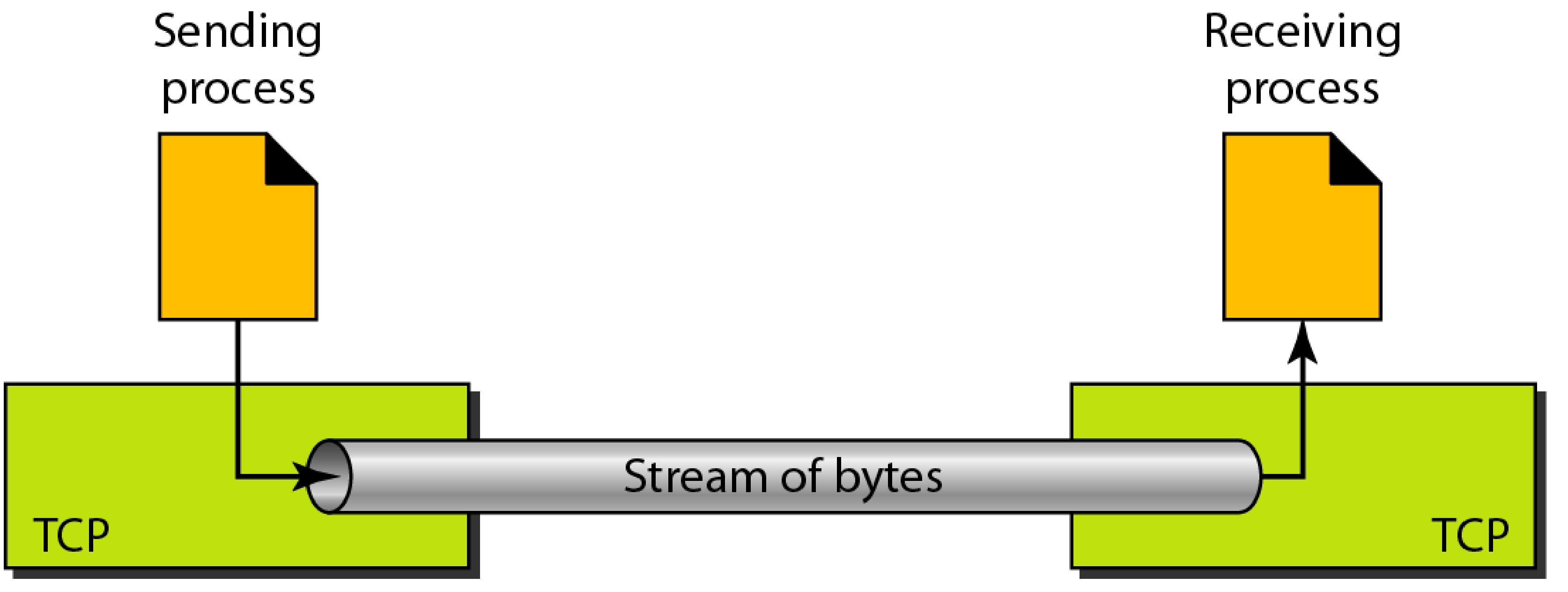
Characteristics/ Description	UDP	TCP
General Description	Simple High speed low functionality "wrapper" that interface applications to the network layer and does little else	Full-featured protocol that allows applications to send data reliably without worrying about network layer issues.
Protocol connection Setup	Connection less data is sent without setup	Connection-oriented; Connection must be Established prior to transmission.
Data interface to application	Message base-based is sent in discrete packages by the application.	Stream-based; data is sent by the application with no particular structure
Reliability and Acknowledgements	Unreliable best-effort delivery without acknowledgements	Reliable delivery of message all data is acknowledged.
Retransmissions	Not performed. Application must detect lost data and retransmit if needed.	Delivery of all data is managed, and lost data is retransmitted automatically.
Features Provided to Manage flow of Data	None	Flow control using sliding windows; window size adjustment heuristics; congestion avoidance algorithms
Overhead	Very Low	Low, but higher than UDP
Transmission speed	Very High	High but not as high as UDP
Data Quantity Suitability	Small to moderate amounts of data.	Small to very large amounts of data.

The TCP Service Model

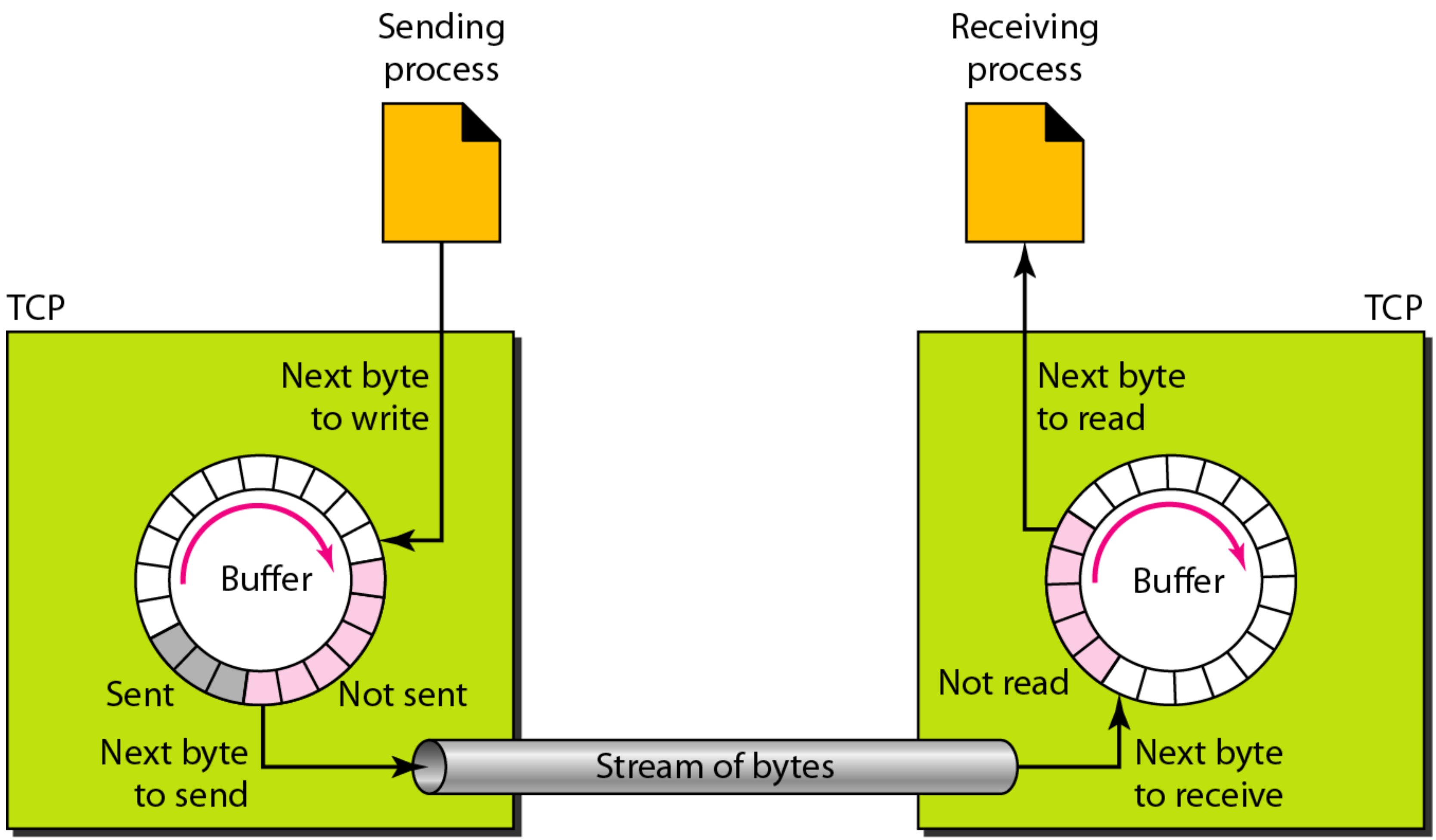
Some assigned ports.

Port	Protocol	Use
21	FTP	File transfer
23	Telnet	Remote login
25	SMTP	E-mail
69	TFTP	Trivial File Transfer Protocol
79	Finger	Lookup info about a user
80	HTTP	World Wide Web
110	POP-3	Remote e-mail access
119	NNTP	USENET news

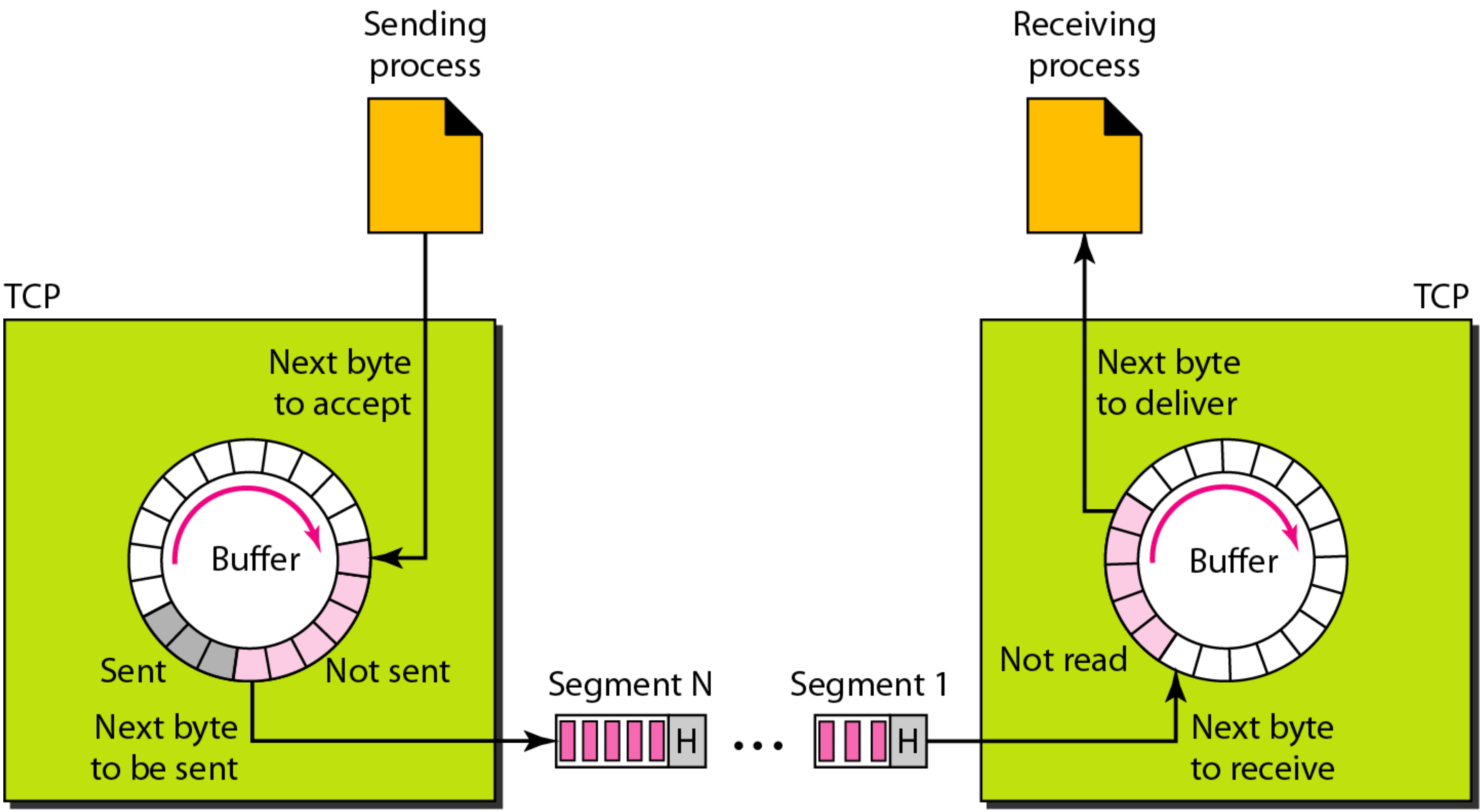
Stream delivery



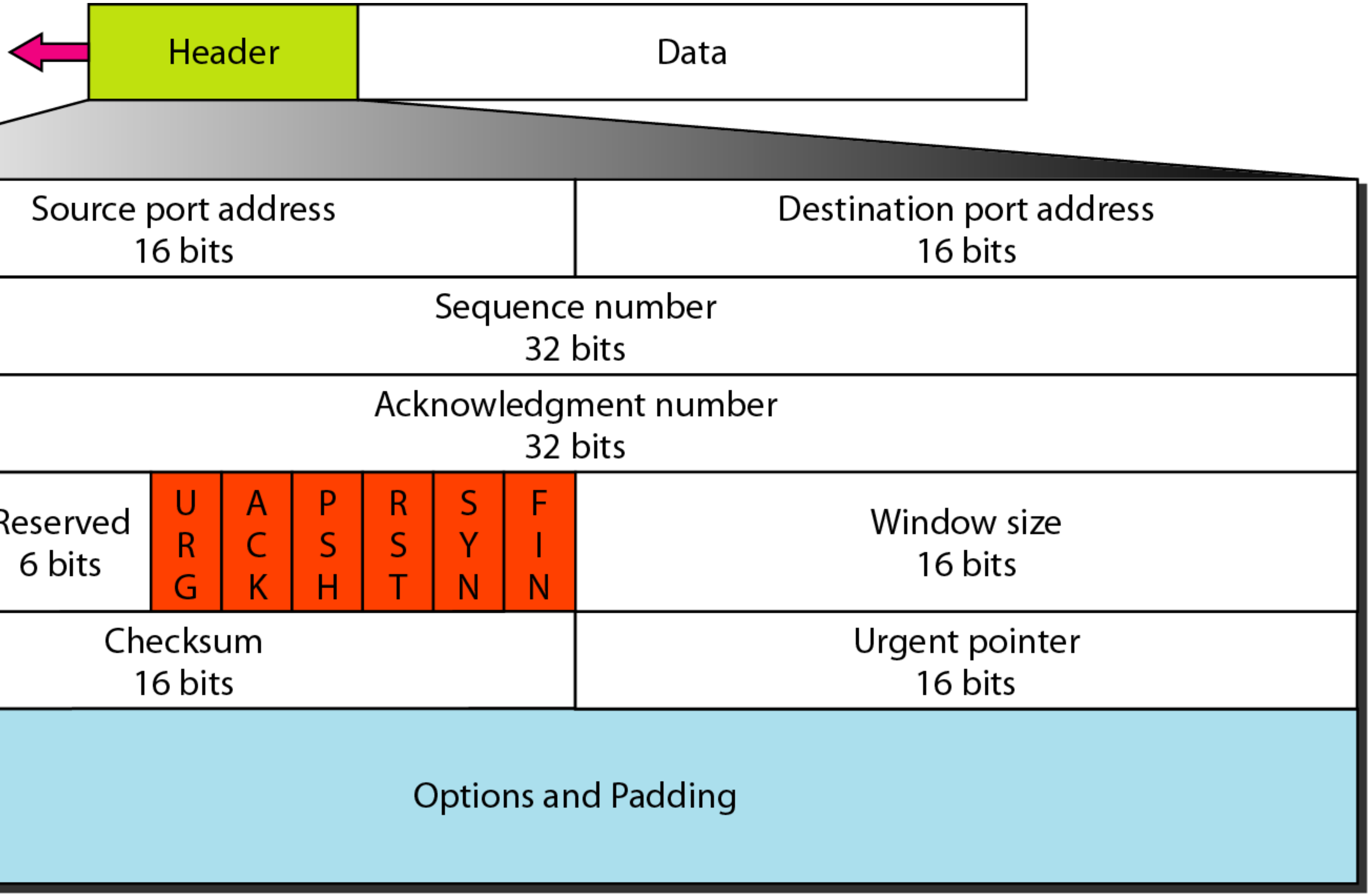
Sending and receiving buffers



TCP segments

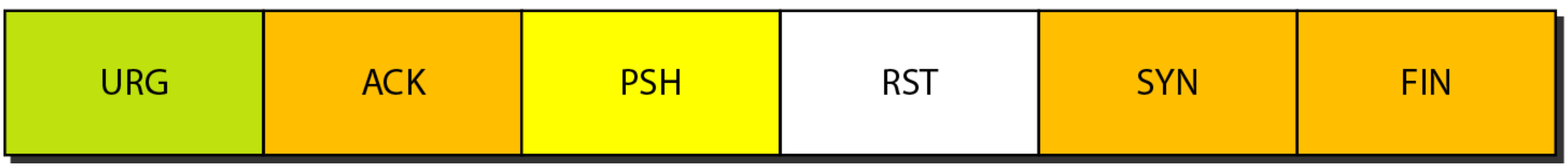


TCP segment format (TCP Header)



Control field

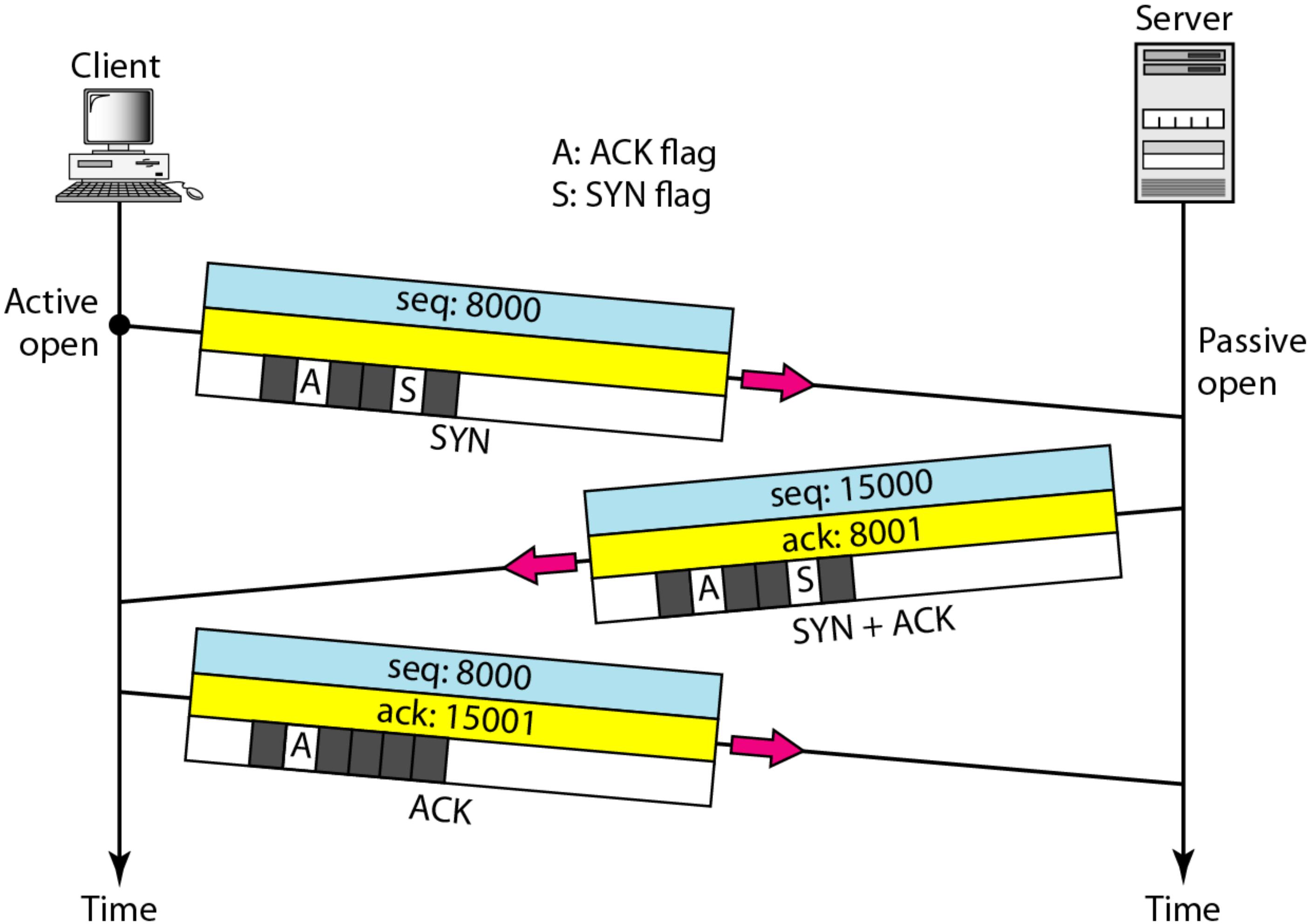
URG: Urgent pointer is valid
ACK: Acknowledgment is valid
PSH: Request for push
RST: Reset the connection
SYN: Synchronize sequence numbers
FIN: Terminate the connection



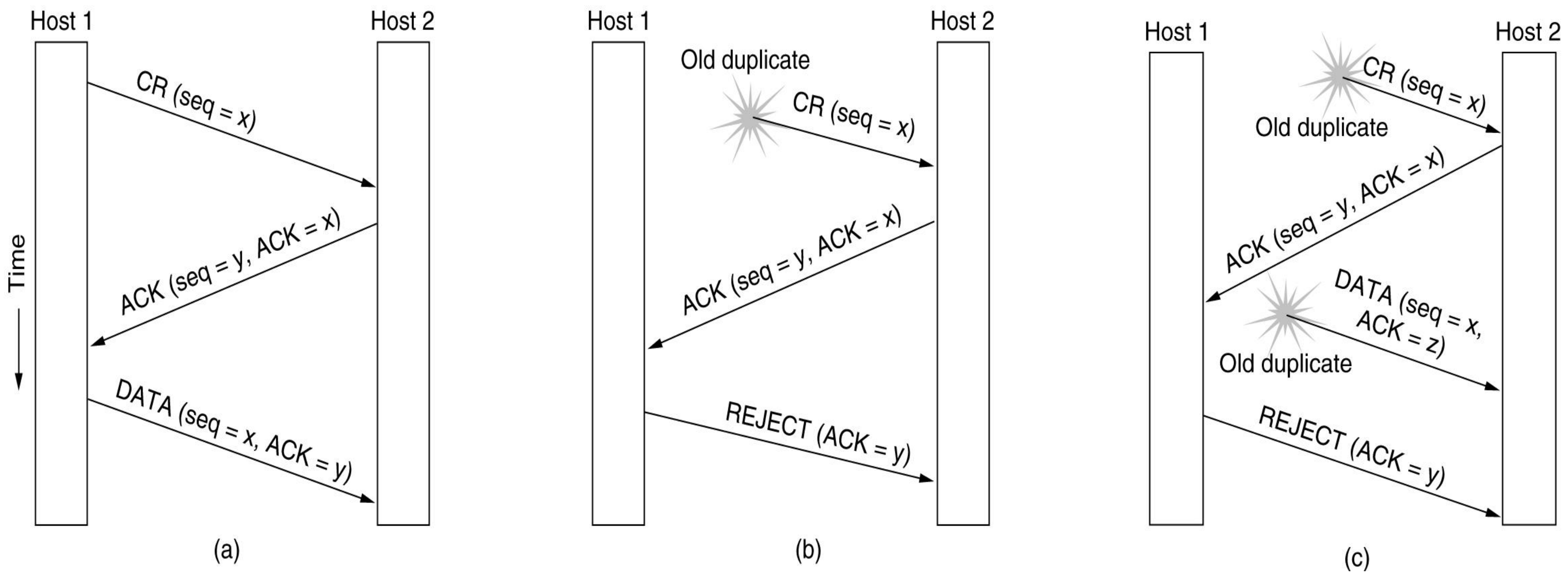
Description of flags in the control field

Flag	*Description*
URG	The value of the urgent pointer field is valid.
ACK	The value of the acknowledgment field is valid.
PSH	Push the data.
RST	Reset the connection.
SYN	Synchronize sequence numbers during connection.
FIN	Terminate the connection.

TCP Connection establishment using three-way handshaking



Connection Establishment (3)



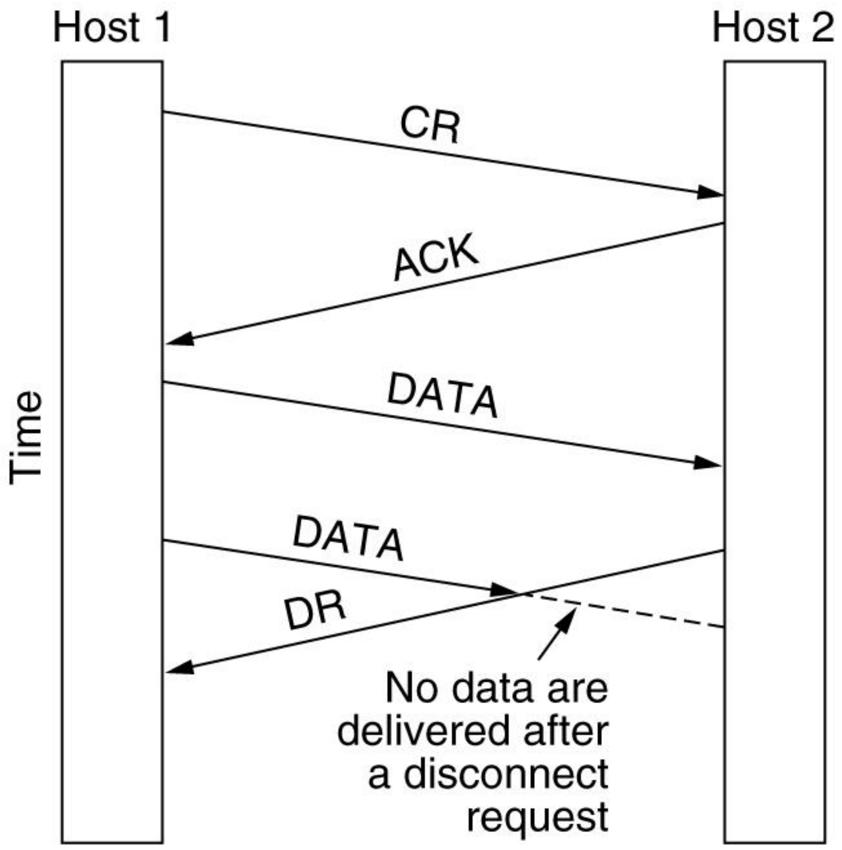
Three protocol scenarios for establishing a connection using a three-way handshake. CR denotes CONNECTION REQUEST.

(a) Normal operation,

(b) Old CONNECTION REQUEST appearing out of nowhere.

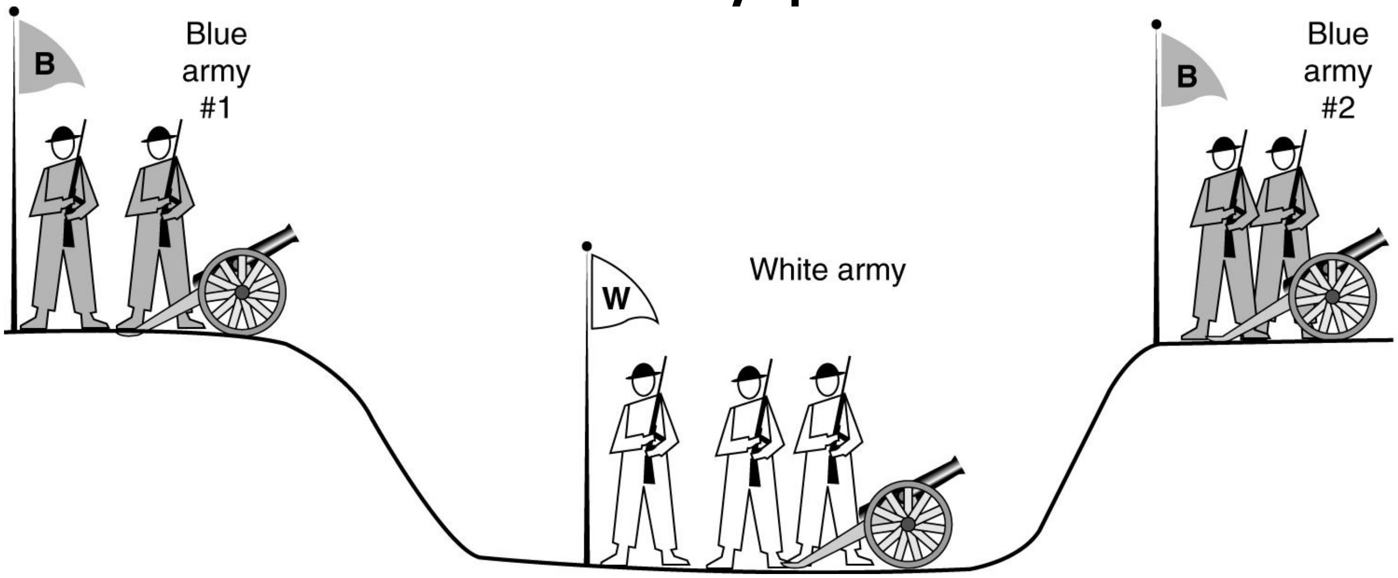
(c) Duplicate CONNECTION REQUEST and duplicate ACK.

Connection Release



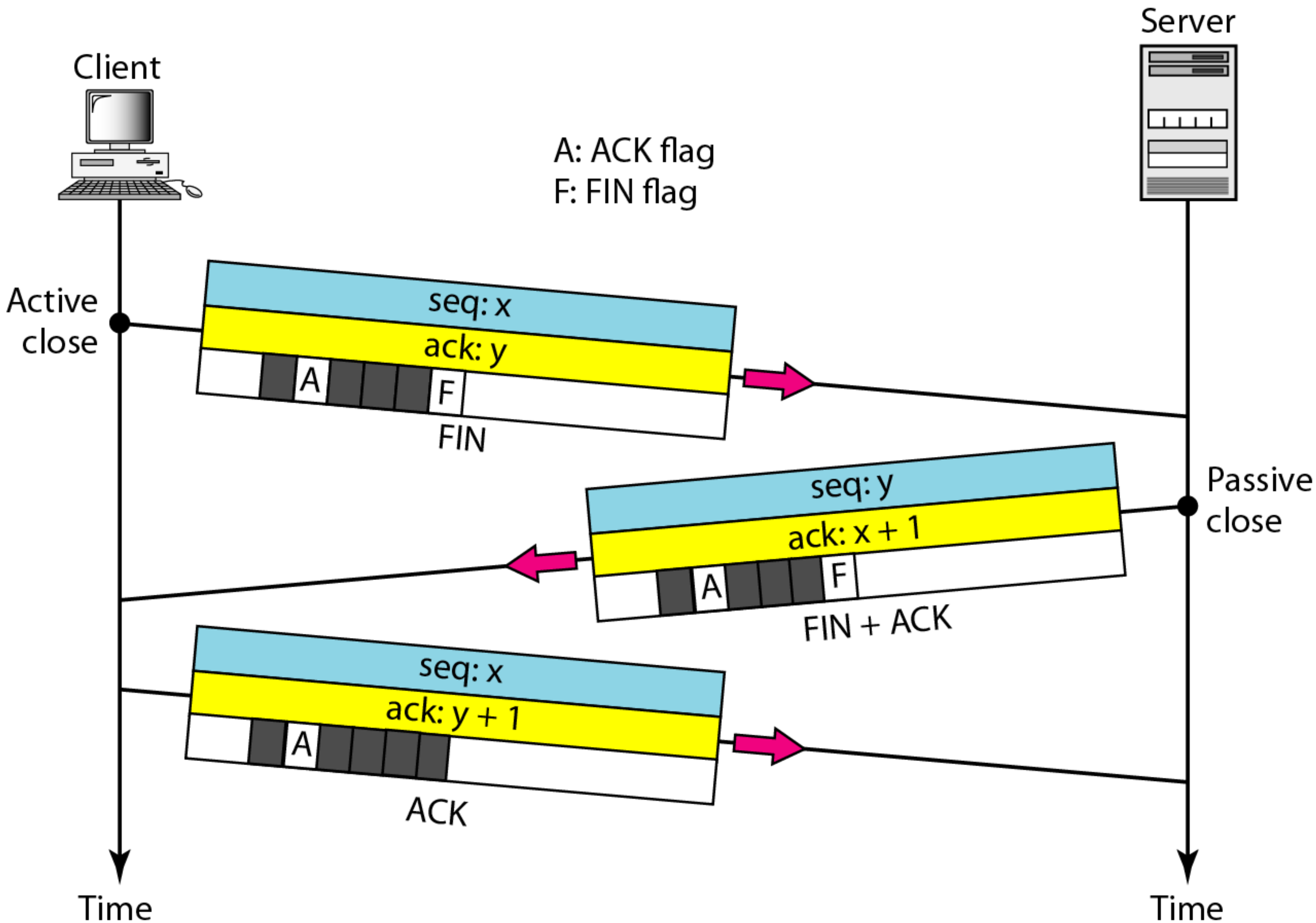
Abrupt disconnection with loss of data.

Connection Release (2)

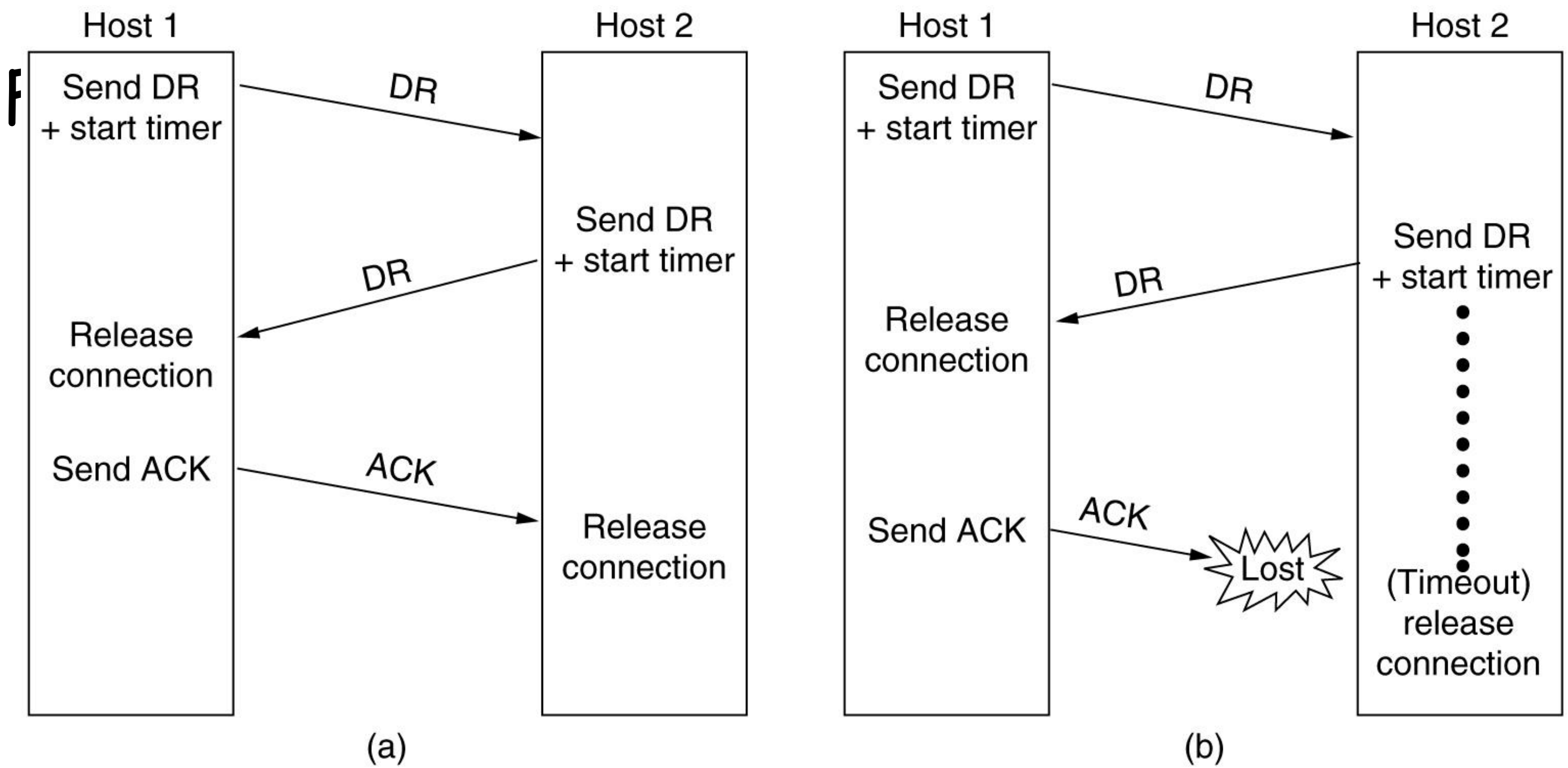


The two-army problem.

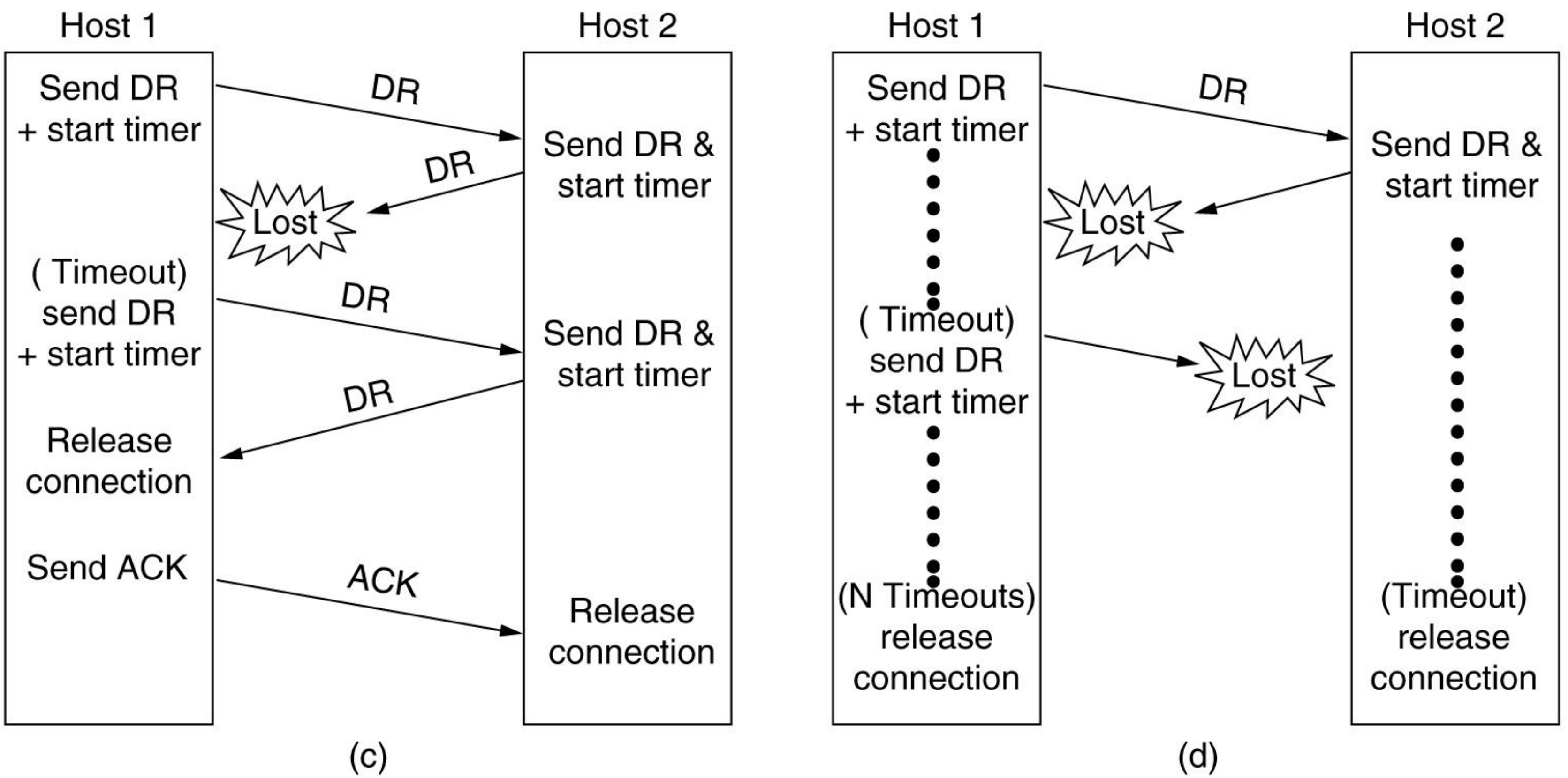
TCP Connection termination using three-way handshaking



Connection Release (3)

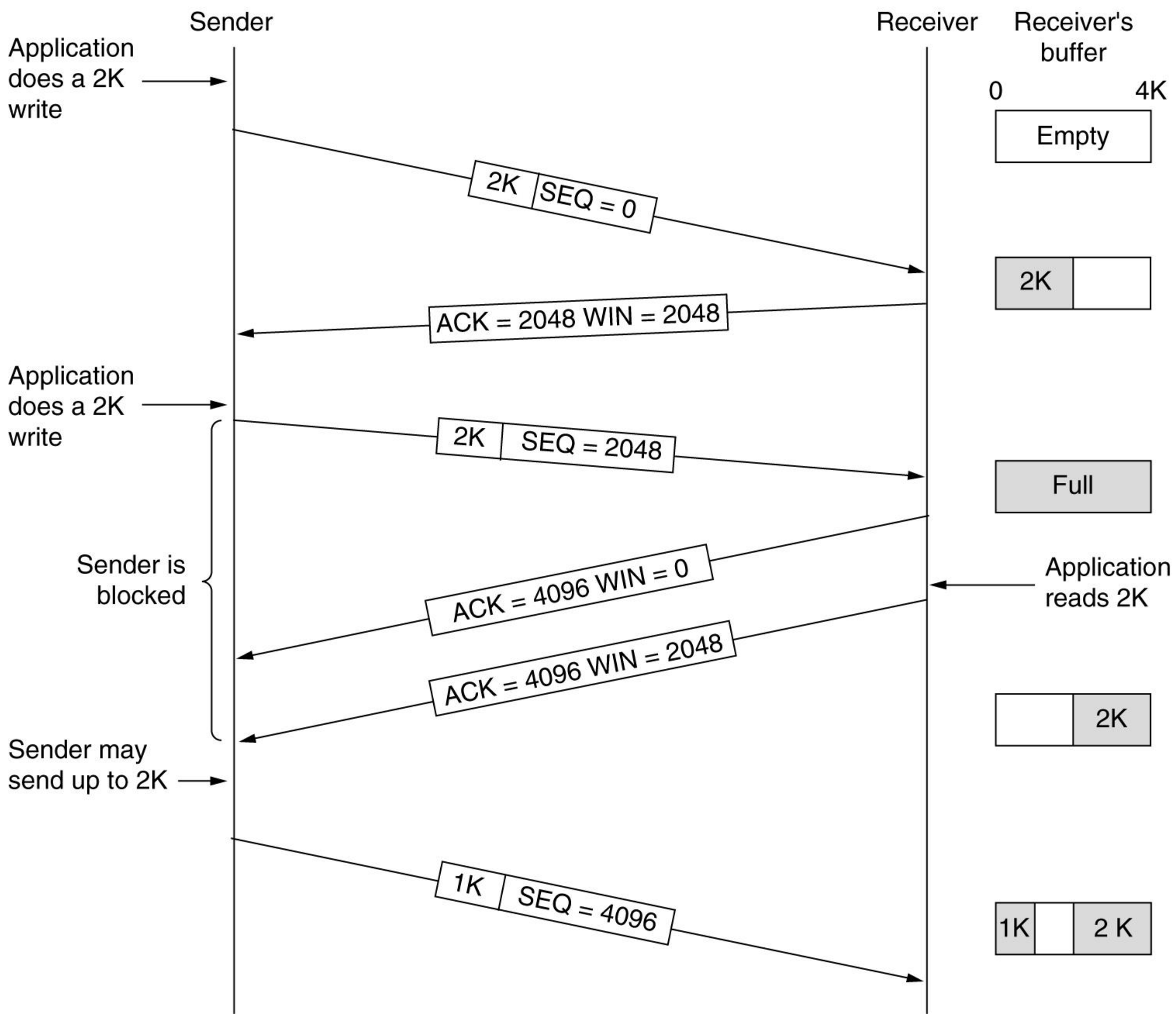


Connection Release (4)

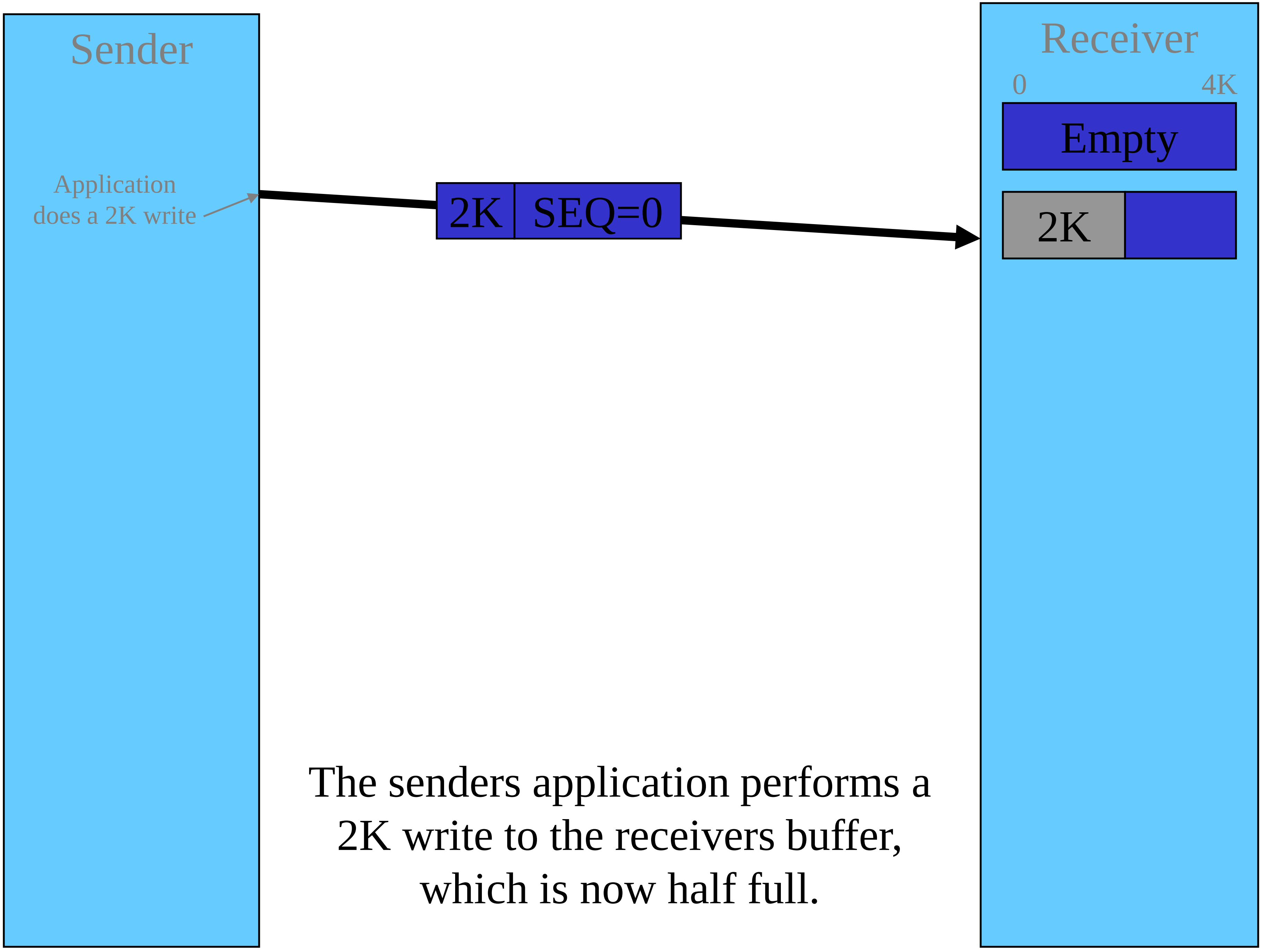


(c) Response lost. (d) Response lost and subsequent DRs lost.

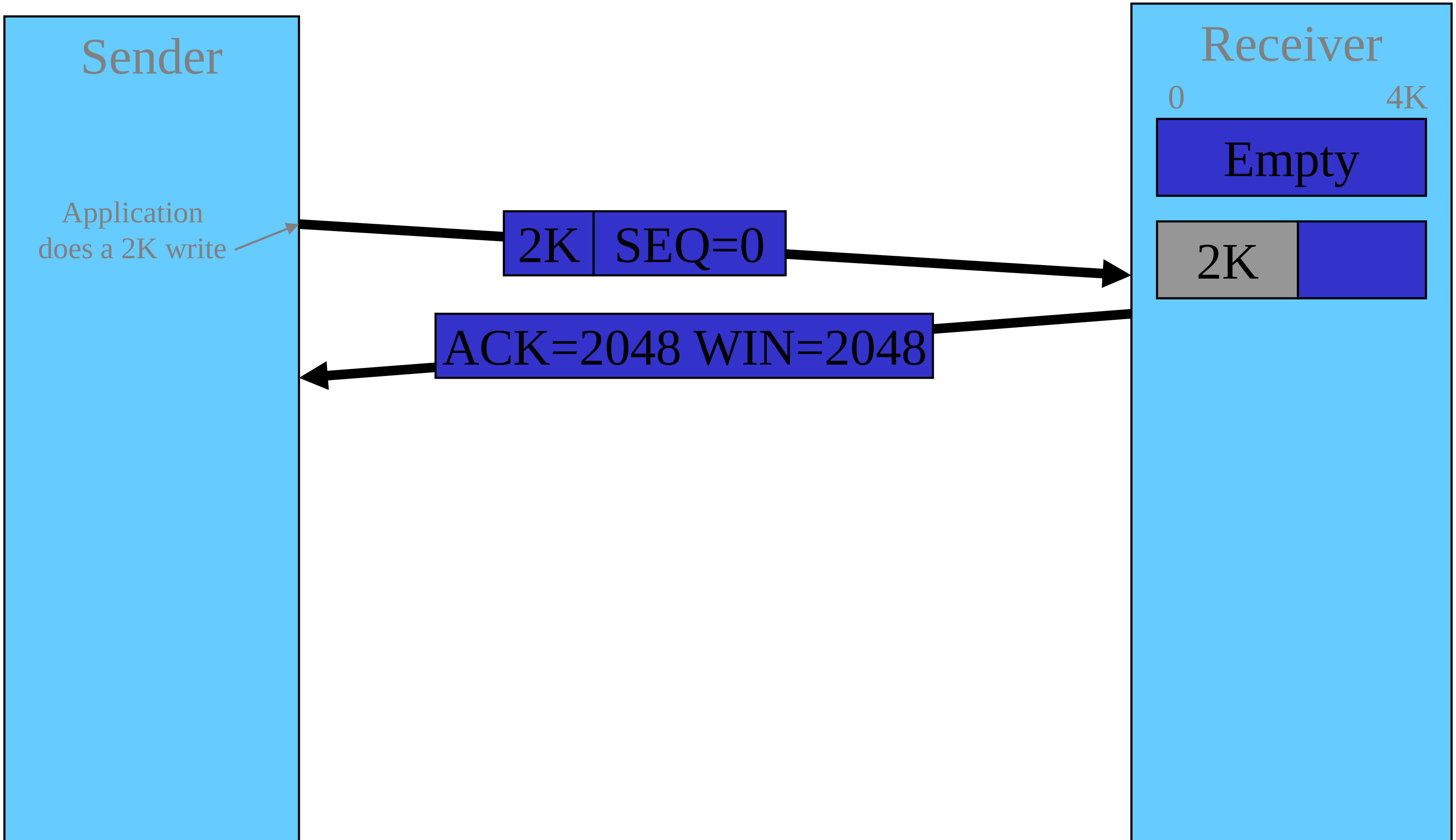
TCP Transmission Policy(Flow control)



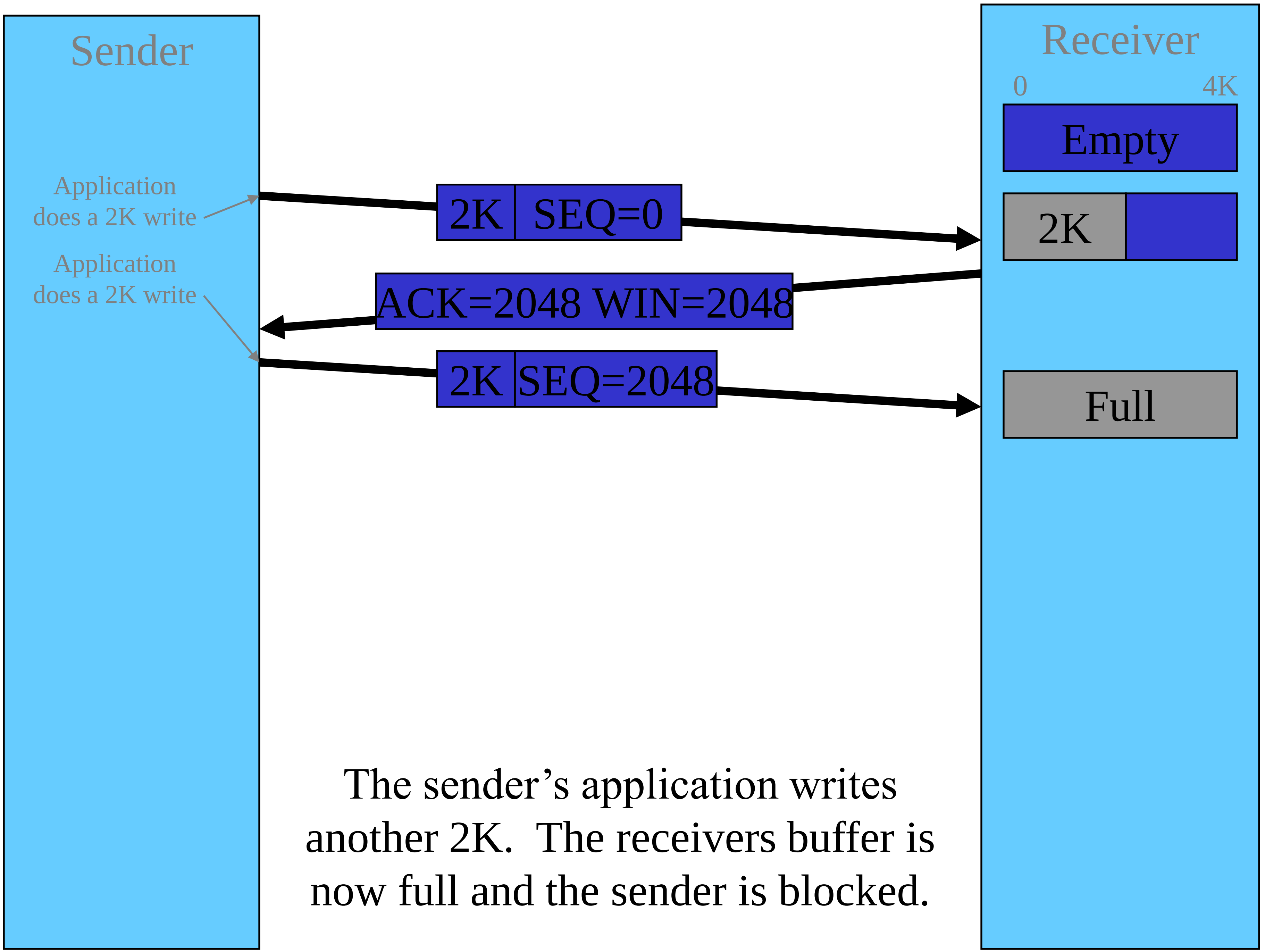
Window management in TCP.



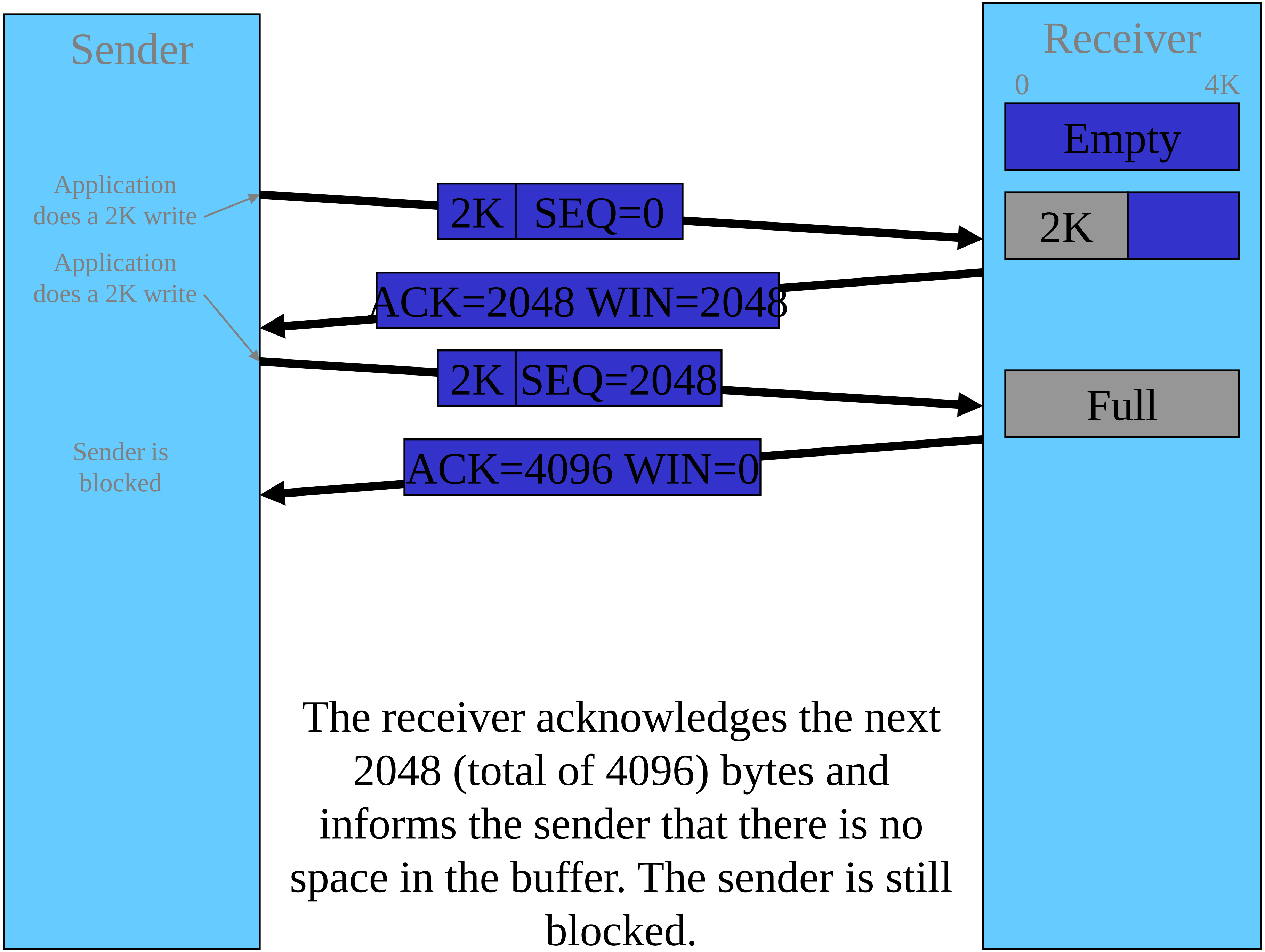
The senders application performs a 2K write to the receivers buffer, which is now half full.



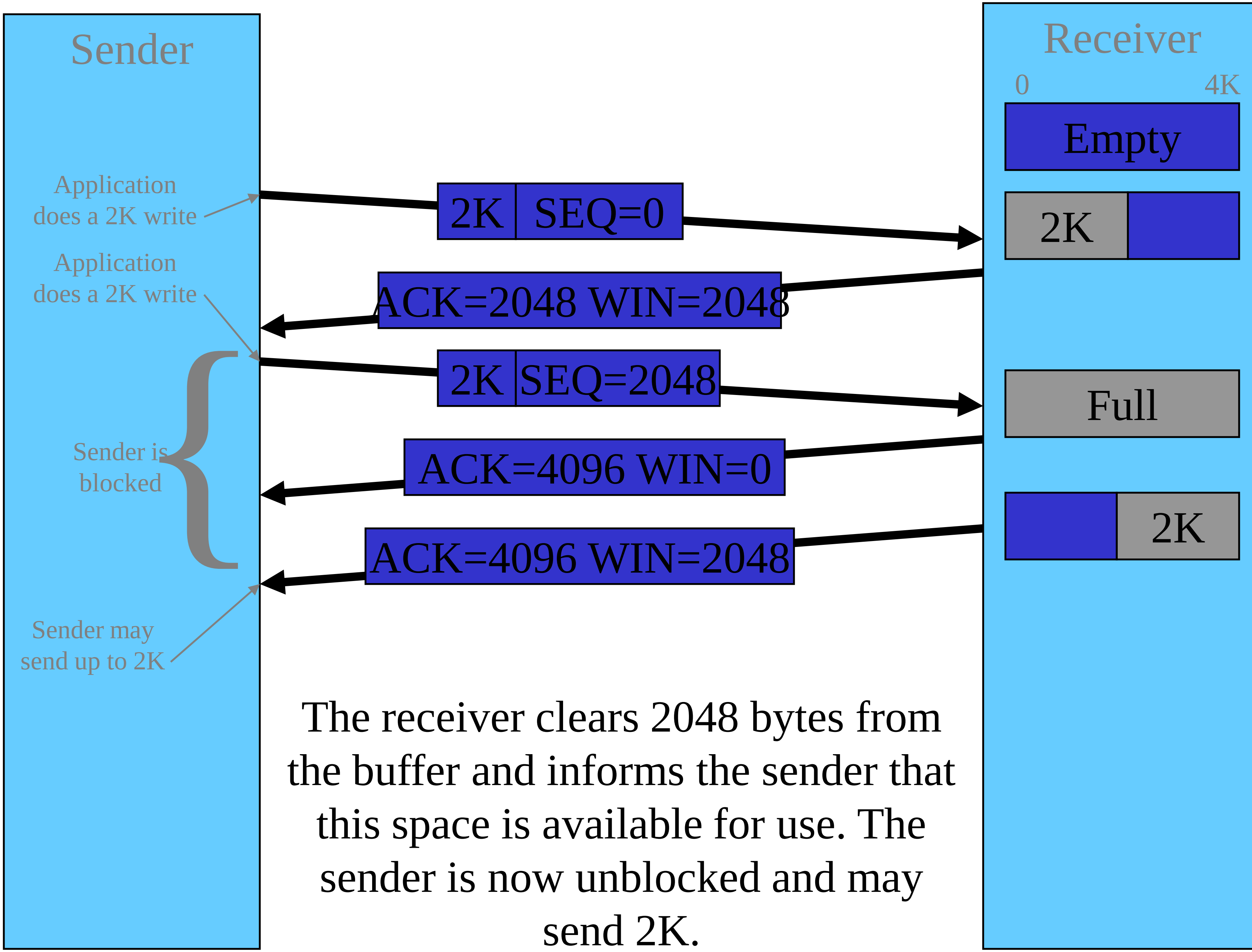
The receiver acknowledges the first 2048 bytes and informs the sender that there is space in the buffer for 2048 bytes.



The sender's application writes another 2K. The receivers buffer is now full and the sender is blocked.



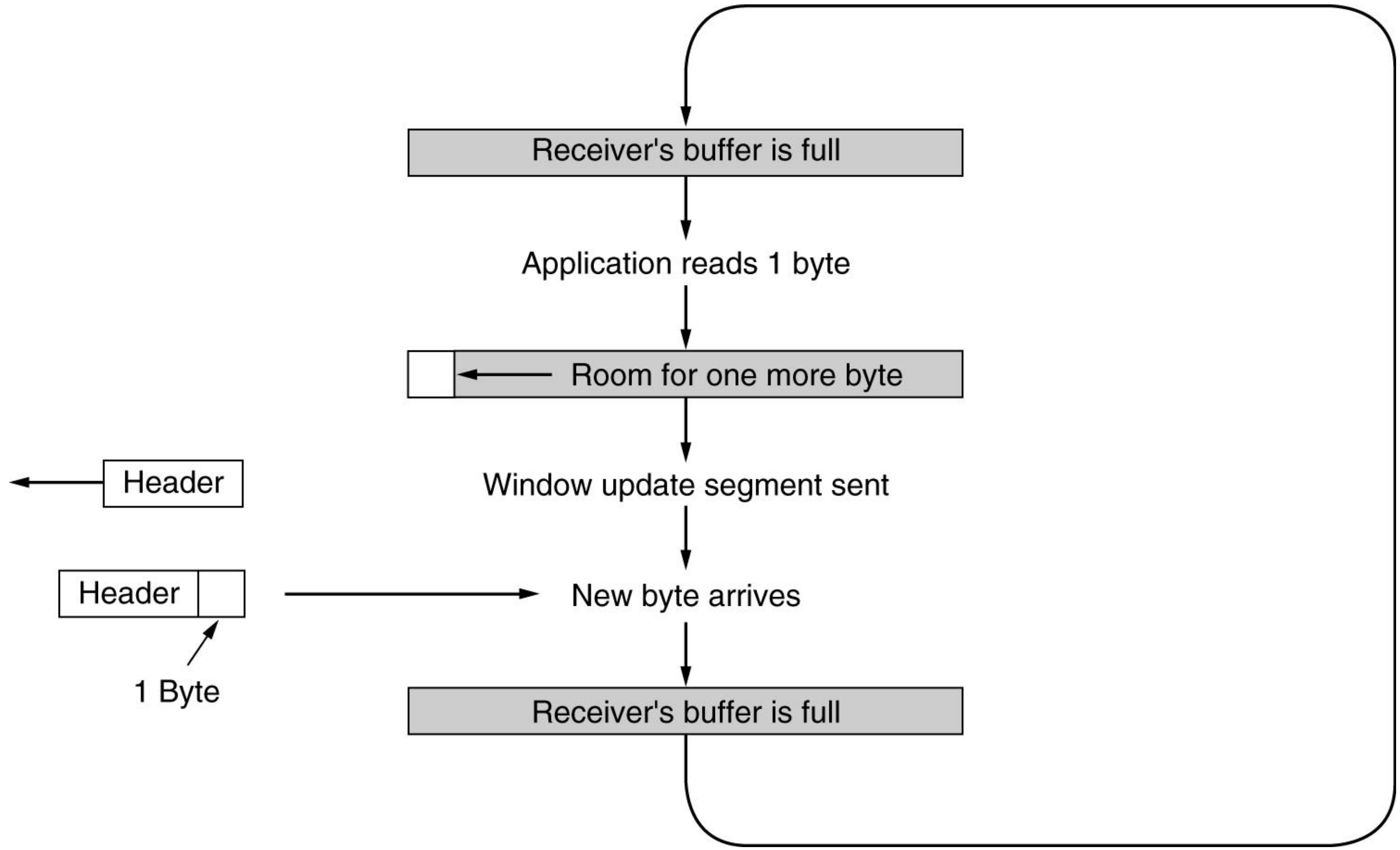
The receiver acknowledges the next 2048 (total of 4096) bytes and informs the sender that there is no space in the buffer. The sender is still blocked.



The receiver clears 2048 bytes from the buffer and informs the sender that this space is available for use. The sender is now unblocked and may send 2K.

The sender's application writes another 1K. The receivers buffer now has 1K of space available.

Silly window syndrome Problem



Silly window syndrome.

Solution to Silly window syndrome Problem

- There are two solutions

1. Nagle's solution
2. Clark's solution

Nagle's algorithm

Purpose is to allow the **sender** TCP to make efficient use of the network, while still being responsive to the sender applications.

Idea:

If application data comes in byte by byte, send first byte only. Then *buffer all application data till until ACK for first byte comes in*.
If network is slow and application is fast, the second segment will contain a lot of data.
Send second segment and buffer all data till ACK for second segment comes in.
An exception to this rule is to always send (not wait for ACK) if enough data for half the receiver window or MSS(Maximum segment size) is accumulated.

Clark's algorithm

Purpose is to prevent the **receiver** from sending a window update for 1byte.

Idea:

- Receiver is forced to wait until it has a decent amount of space available

- The receiver should not send a window update until it can handle the maximum segment size it declared when the connection was established or until its buffer is half empty, whichever is smaller

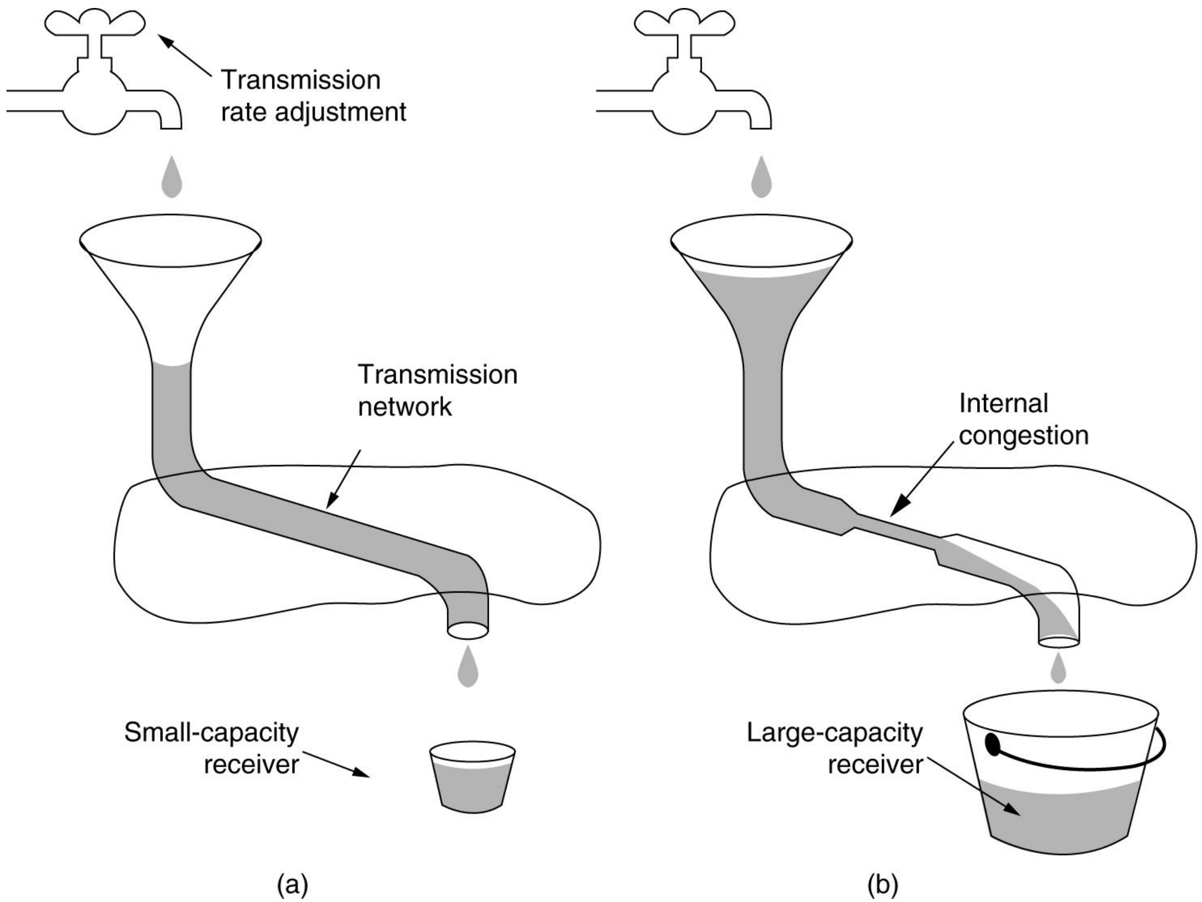
TCP congestion control

We looked at how TCP handles flow control. In addition we know the congestion happens. The only real way to handle congestion is for the sender to reduce sending rate.

So how does on detect congestion ?
In old days, packets were lost due to transmission errors and congestion. But nowadays, transmission errors are very rare (except for wireless). So, TCP assumes a lost packet as an indicator of congestion.

So does TCP deal with congestion ?
It maintains an indicator of network capacity, called the ***congestion window***

TCP Congestion Control



(a) A fast network feeding a low capacity receiver.
(b) A slow network feeding a high-capacity receiver.

TCP congestion control

In essence TCP deals with two potential problems separately:

Problem		*Solution*
Receiver capacity	→	**Receiver window** (rwnd)
Network capacity	→	**Congestion window** (cwnd)

Each window reflect the number of bytes the sender may transmit. The sender sends the *minimum of these two sizes*. This size is the ***effective window***.

TCP Congestion Control – 3 Stages

TCP uses these stages in updating cwnd.

1. *Slow start*: Initial state. Rapidly grow cwnd
2. *Congestion avoidance*: Slowly grow cwnd.

} Control amount of data injected into network

3. *Congestion detection : (Multiplicative decrease)*

TCP Congestion Control - Slow start

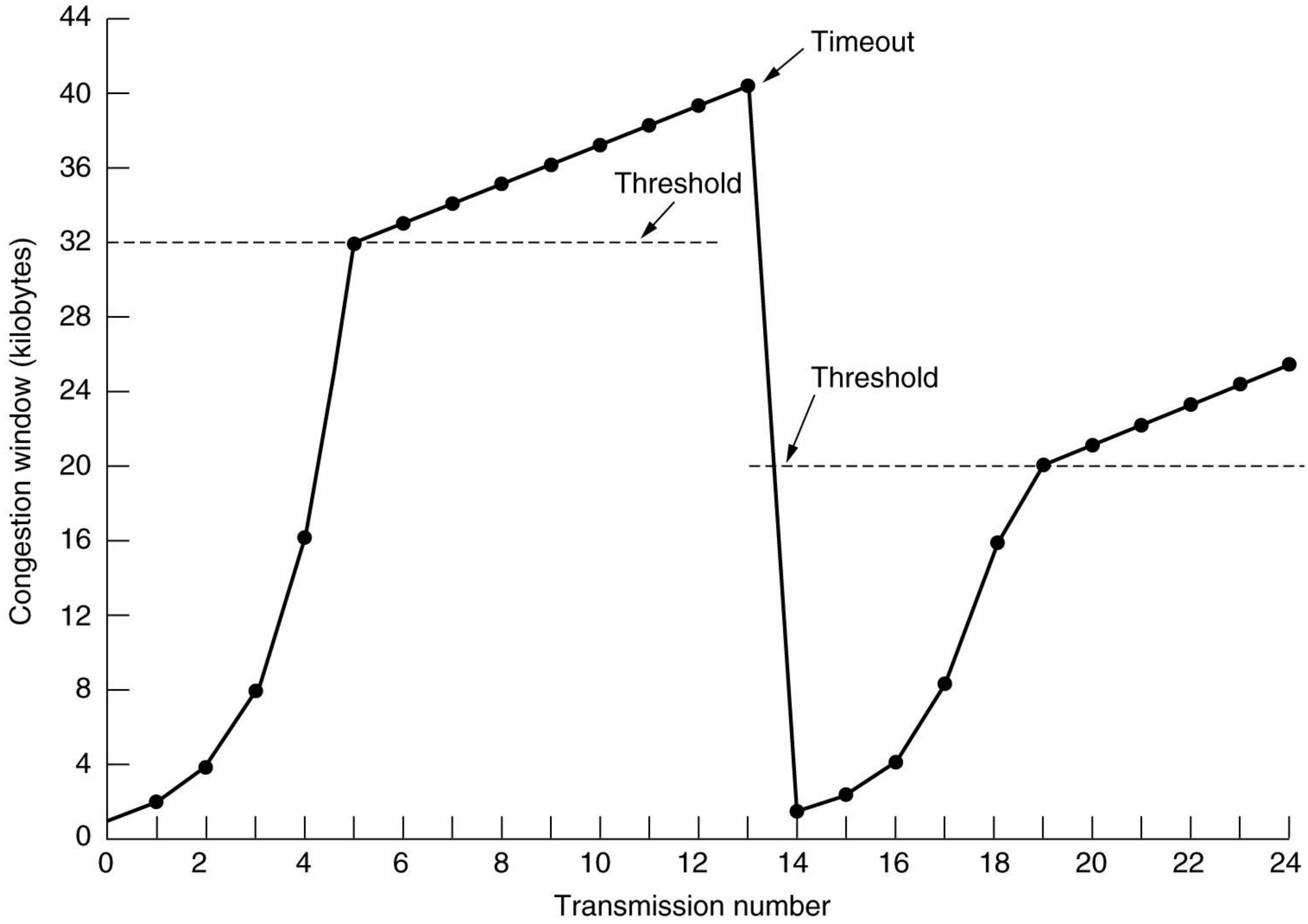
- When connection is established , the sender initializes the congestion window to the size of the maximum segment in use on the connection.

- It then sends the one maximum segment

- If this segment is acknowledged before timeout occurs then it doubles the segment size

- This is continued until the timeout occurs or receivers window size is reached

TCP Congestion Control



An example of the Internet congestion algorithm.

TCP Congestion Control-Congestion Avoidance

- When the size of congestion window reaches the slow start threshold, the slow start phase stops and the additive phase begins.

TCP Congestion Control-Congestion Detection

- If congestion occurs the congestion window size must be decreased.
- That means when a timer time outs or when 3 Acks are received the size of the threshold is dropped to ½ (multiplicative decrease)

Chapter 2: ROAD MAP

- Transport Layer Introduction
- Port Address
- UDP
- TCP
- Socket Programming using TCP and UDP
- SCTP
- RTP
- TCP in wireless network
- Quality of services

Socket programming

Goal: learn how to build client/server application that communicate using sockets

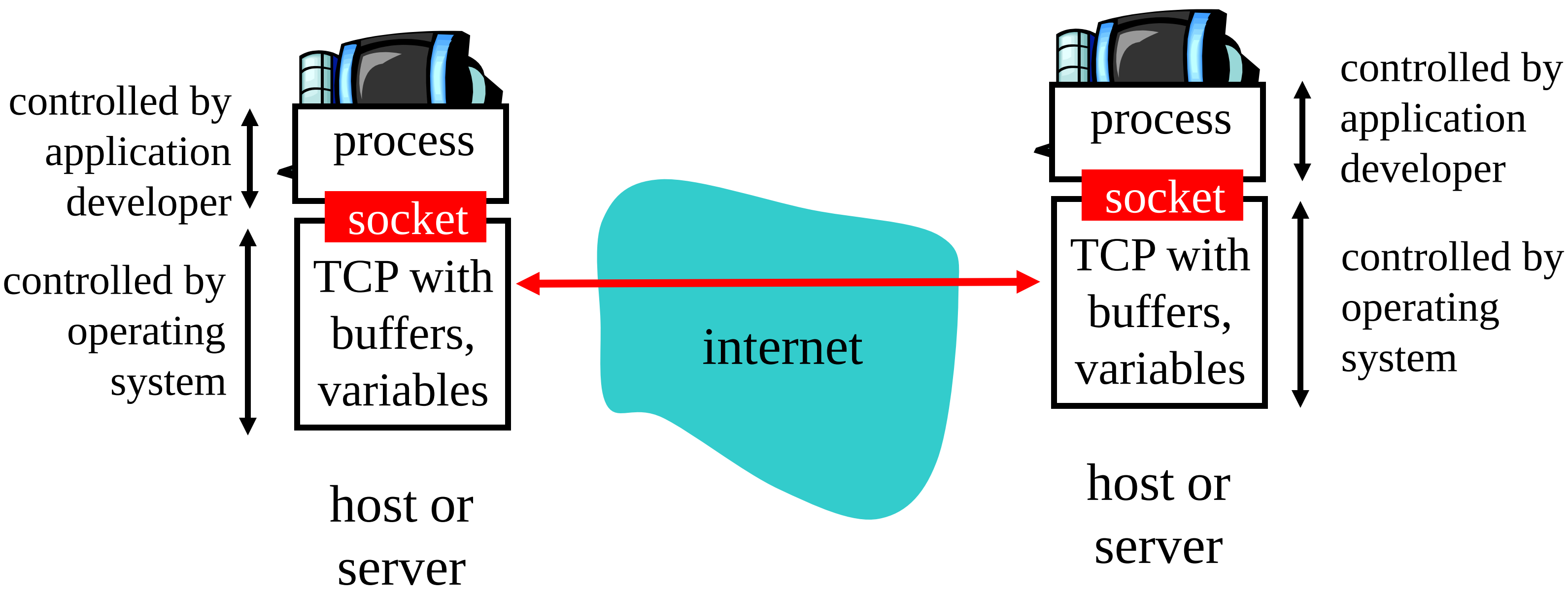
Socket API

- client/server paradigm
- two types of transport service via socket API:
  - unreliable datagram (UDP)
  - reliable, byte stream-oriented (TCP)

Socket-programming using TCP

Socket: a door between application process and end-end-transport protocol (UCP or TCP)

TCP service: reliable transfer of **bytes** from one process to another



Socket programming *with TCP*

Client must contact server

- server process must first be running
- server must have created socket (door) that welcomes client's contact

Client contacts server by:

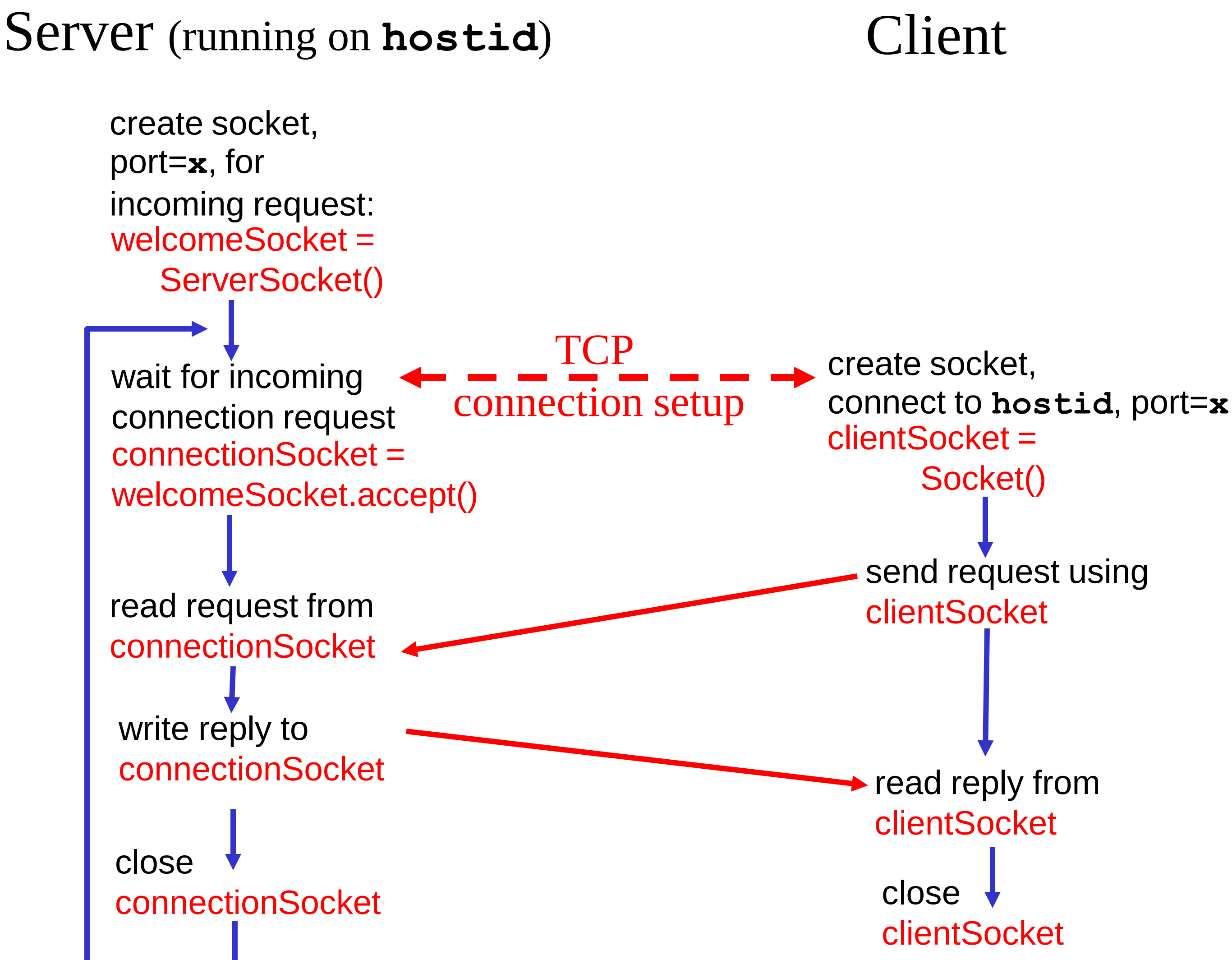
- creating client-local TCP socket
- specifying IP address, port number of server process
- When client creates socket: client TCP establishes connection to server TCP

- When contacted by client, server TCP creates new socket for server process to communicate with client
  - allows server to talk with multiple clients
  - source port numbers used to distinguish clients

application viewpoint

TCP provides reliable, in-order transfer of bytes ("pipe") between client and server

Client/server socket interaction: TCP



Socket programming with TCP

Example client-server app:

1) client reads line from standard input (**inFromUser** stream) , sends to server via socket (**outToServer** stream)

2) server reads line from socket

3) server converts line to uppercase, sends back to client

4) client reads, prints modified line from socket (**inFromServer** stream)

Example: Java client (TCP)

```
import java.io.*;
import java.net.*;
class TCPClient {

    public static void main(String argv[]) throws Exception
    {
        String sentence;
        String modifiedSentence;
```

Create input stream

```
        BufferedReader inFromUser =
          new BufferedReader(new InputStreamReader(System.in));
```

Create client socket, connect to server

```
        Socket clientSocket = new Socket("hostname", 6789);
```

Create output stream attached to socket

```
        DataOutputStream outToServer =
          new DataOutputStream(clientSocket.getOutputStream());
```

Example: Java client (TCP), cont.

Create input stream attached to socket

```
BufferedReader inFromServer =
  new BufferedReader(new
  InputStreamReader(clientSocket.getInputStream()));

sentence = inFromUser.readLine();
```

Send line to server

```
outToServer.writeBytes(sentence + '\n');
```

Read line from server

```
modifiedSentence = inFromServer.readLine();

System.out.println("FROM SERVER: " + modifiedSentence);

clientSocket.close();

  }
}
```

Example: Java server (TCP)

```
import java.io.*;
import java.net.*;

class TCPServer {

  public static void main(String argv[]) throws Exception
   {
     String clientSentence;
     String capitalizedSentence;

     ServerSocket welcomeSocket = new ServerSocket(6789);

     while(true) {

        Socket connectionSocket = welcomeSocket.accept();

       BufferedReader inFromClient =
         new BufferedReader(new
         InputStreamReader(connectionSocket.getInputStream()));
```

Create welcoming socket at port 6789 → `ServerSocket welcomeSocket = new ServerSocket(6789);`

Wait, on welcoming socket for contact by client → `Socket connectionSocket = welcomeSocket.accept();`

Create input stream, attached to socket → `BufferedReader inFromClient = ...`

Example: Java server (TCP), cont

Create output stream, attached to socket

Read in line from socket

Write out line to socket

End of while loop, loop back and wait for another client connection

```
            DataOutputStream  outToClient =
              new DataOutputStream(connectionSocket.getOutputStream());

            clientSentence = inFromClient.readLine();

            capitalizedSentence = clientSentence.toUpperCase() + '\n';

            outToClient.writeBytes(capitalizedSentence);
        }
    }
}
```

Socket programming *with UDP*

UDP: no "connection" between client and server

- no handshaking
- sender explicitly attaches IP address and port of destination to each packet
- server must extract IP address, port of sender from received packet

UDP: transmitted data may be received out of order, or lost

application viewpoint

UDP provides unreliable transfer of groups of bytes ("datagrams") between client and server

Client/server socket interaction: UDP

Server (running on **`hostid`**)

create socket,
port= x.
serverSocket =
DatagramSocket()

read datagram from
serverSocket

write reply to
serverSocket
specifying
client address,
port number

Client

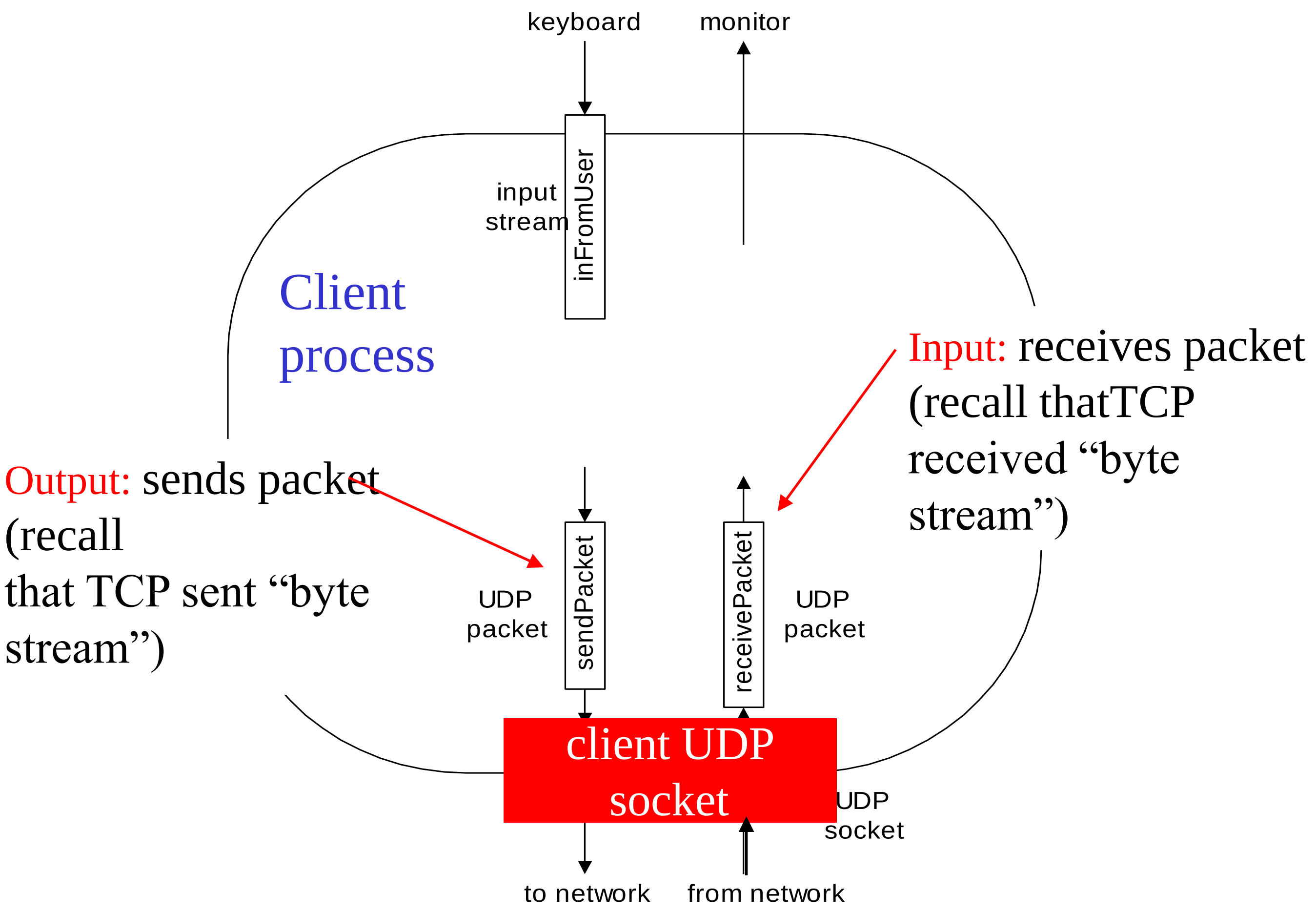
create socket,
clientSocket =
DatagramSocket()

Create datagram with server IP and
port=x; send datagram via
clientSocket

read datagram from
clientSocket

close
clientSocket

Example: Java client (UDP)



Example: Java client (UDP)

```
import java.io.*;
import java.net.*;

class UDPClient {
    public static void main(String args[]) throws Exception
    {

      BufferedReader inFromUser =
        new BufferedReader(new InputStreamReader(System.in));

      DatagramSocket clientSocket = new DatagramSocket();

      InetAddress IPAddress = InetAddress.getByName("hostname");

      byte[] sendData = new byte[1024];
      byte[] receiveData = new byte[1024];

      String sentence = inFromUser.readLine();

      sendData = sentence.getBytes();
```

Create input stream → BufferedReader inFromUser

Create client socket → DatagramSocket clientSocket

Translate hostname to IP address using DNS → InetAddress IPAddress

Example: Java client (UDP), cont.

Create datagram with data-to-send, length, IP addr, port

```
DatagramPacket sendPacket =
  new DatagramPacket(sendData, sendData.length, IPAddress, 9876);
```

Send datagram to server

```
clientSocket.send(sendPacket);

DatagramPacket receivePacket =
  new DatagramPacket(receiveData, receiveData.length);
```

Read datagram from server

```
clientSocket.receive(receivePacket);

String modifiedSentence =
   new String(receivePacket.getData());

System.out.println("FROM SERVER:" + modifiedSentence);
clientSocket.close();
}
}
```

Example: Java server (UDP)

Create datagram socket at port 9876

Create space for received datagram

Receive datagram

```
import java.io.*;
import java.net.*;

class UDPServer {
  public static void main(String args[]) throws Exception
    {

      DatagramSocket serverSocket = new DatagramSocket(9876);

      byte[] receiveData = new byte[1024];
      byte[] sendData  = new byte[1024];

      while(true)
        {

          DatagramPacket receivePacket =
             new DatagramPacket(receiveData, receiveData.length);
           serverSocket.receive(receivePacket);
```

Example: Java server (UDP), cont

```
        String sentence = new String(receivePacket.getData());

        InetAddress IPAddress = receivePacket.getAddress();

        int port = receivePacket.getPort();

                String capitalizedSentence = sentence.toUpperCase();

        sendData = capitalizedSentence.getBytes();

        DatagramPacket sendPacket =
           new DatagramPacket(sendData, sendData.length, IPAddress,
                              port);

        serverSocket.send(sendPacket);
      }
    }
  }
```

Get IP addr port #, of sender

Create datagram to send to client

Write out datagram to socket

End of while loop, loop back and wait for another datagram

Chapter 2: ROAD MAP

- Transport Layer Introduction
- Port Address
- UDP
- TCP
- Socket Programming using TCP and UDP
- SCTP (Stream control transmission protocol)
- RTP
- TCP in wireless network
- Quality of services

SCTP

Stream Control Transmission Protocol (SCTP) is a new reliable, message-oriented transport layer protocol. SCTP, however, is mostly designed for Internet applications that have recently been introduced. These new applications need a more sophisticated service than TCP can provide.

Topics discussed in this section:

SCTP Services and Features
Packet Format
An SCTP Association
Flow Control and Error Control

Note

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

Comparison

- UDP: Message-oriented, Unreliable
- TCP: Byte-oriented, Reliable
- SCTP
  - Message-oriented, Reliable
  - Other innovative features
    - Association, Data transfer/Delivery
    - Fragmentation,
    - Error/Congestion Control

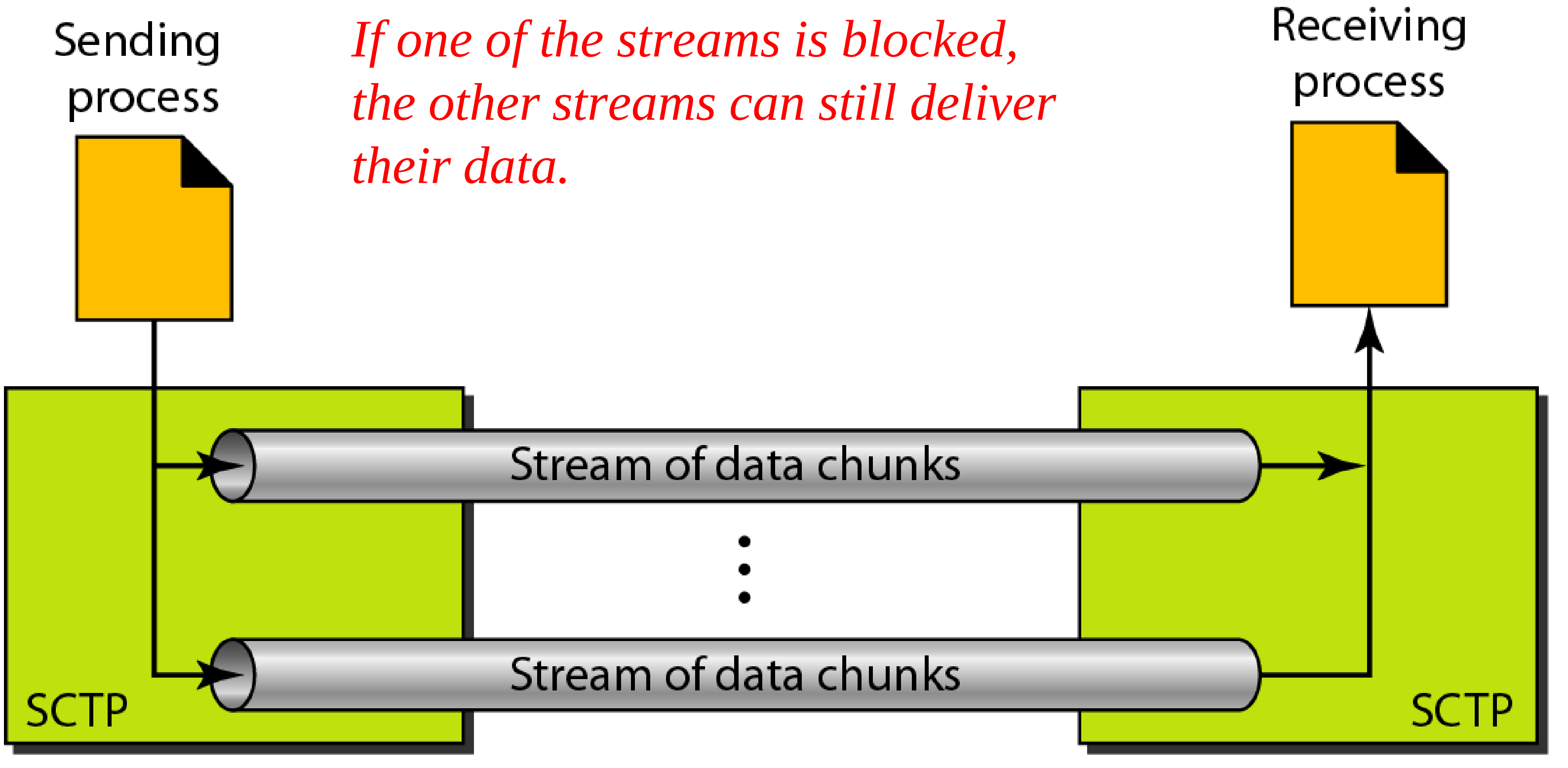
Some SCTP applications

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

Servicesof SCTP

- ✓ Process-to-Process Communication
- ✓ Multiple Streams
- ✓ Multihoming
- ✓ Full-Duplex Communication
- ✓ Connection-Oriented Service
- ✓ Reliable Service

Multiple-stream concept



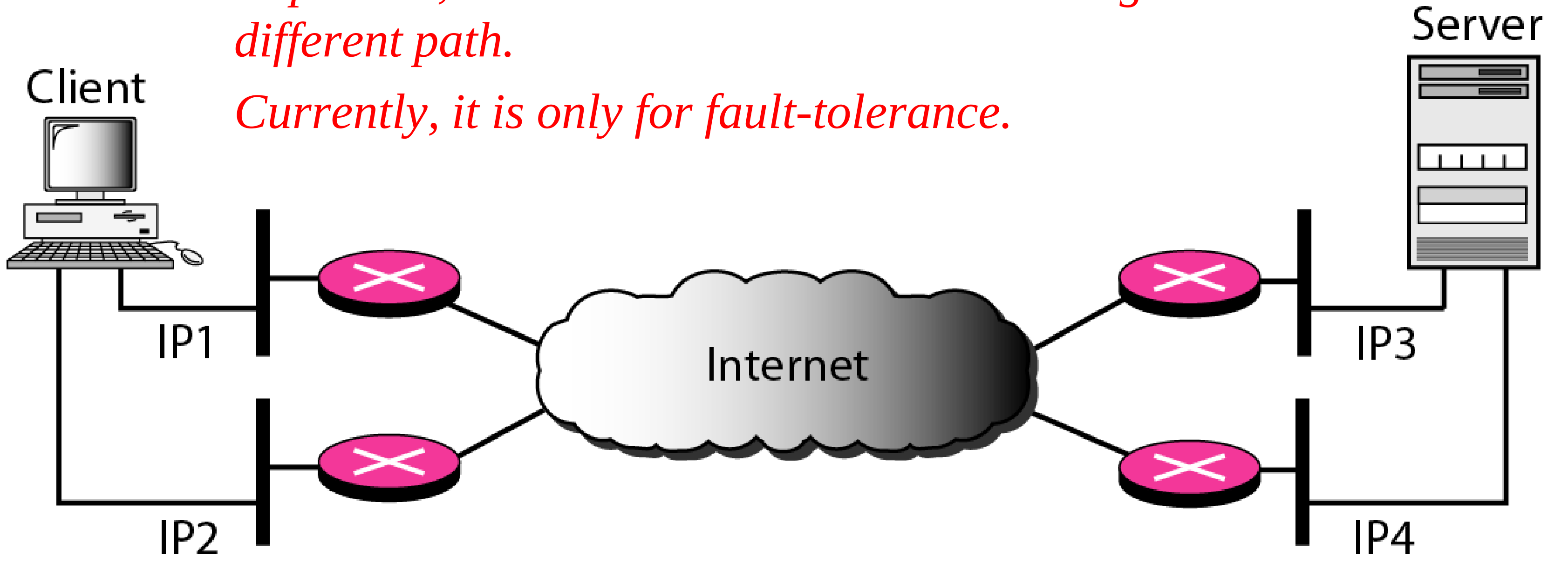
Note

An association in SCTP can involve multiple streams.

Multihoming concept

At present, SCTP does not allow load sharing between different path.

Currently, it is only for fault-tolerance.



SCTP Features

- ✓ Transmission Sequence Number (TSN)
- ✓ Stream Identifier (SI)
- ✓ Stream Sequence Number (SSN)
- ✓ Packets
- ✓ Acknowledgment Number
- ✓ Flow Control
- ✓ Error Control
- ✓ Congestion Control

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.

Comparison between UDP,TCP and SCTP

UDP	TCP	SCTP
Message oriented protocol	Byte oriented protocol	Message oriented protocol
Preserve message boundaries	Does not Preserve message boundaries	Preserve message boundaries
Unreliable	Reliable	Reliable
No congestion and flow control	Have congestion and flow control	Have congestion and flow control
Each message follows different route so no sequencing	Each message follows same route so have in sequence data delivery	have in sequence data delivery
Port no 17	Port no 6	Port no 132

SCTP vs. TCP (1)

- Control information
  - TCP: part of the header
  - SCTP: several types of control chunks
- Data
  - TCP: one entity in a TCP segment
  - SCTP: several data chunks in a packet
- Option
  - TCP: part of the header
  - SCTP: handled by defining new chunk types

SCTP vs. TCP (2)

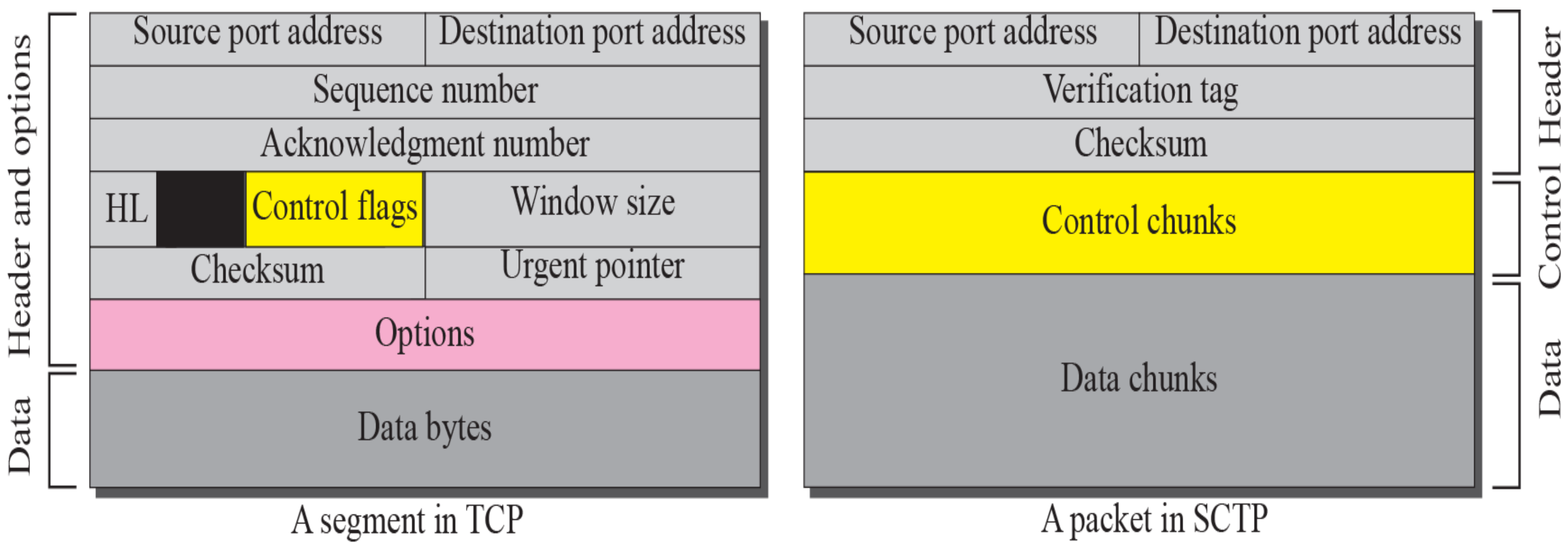
- Mandatory part of the header
  - TCP: 20 bytes, SCTP: 12 bytes
  - Reason:
    - TSN in data chunk's header
    - Ack. # and window size are part of control chunk
    - No need for header length field (∵no option)
    - No need for an urgent pointer
- Checksum
  - TCP: 16 bits, SCTP: 32 bit

SCTP vs. TCP (3)

- Association identifier
  - TCP: none, SCTP: verification tag
  - Multihoming in SCTP
- Sequence number
  - TCP: one # in the header
  - SCTP: TSN, SI and SSN define each data chunk
  - SYN and FIN need to consume one seq. #
  - Control chunks never use a TSN, SI, or SSN number

Comparison between a TCP segment and an SCTP packet

TCP has segments; SCTP has packets.

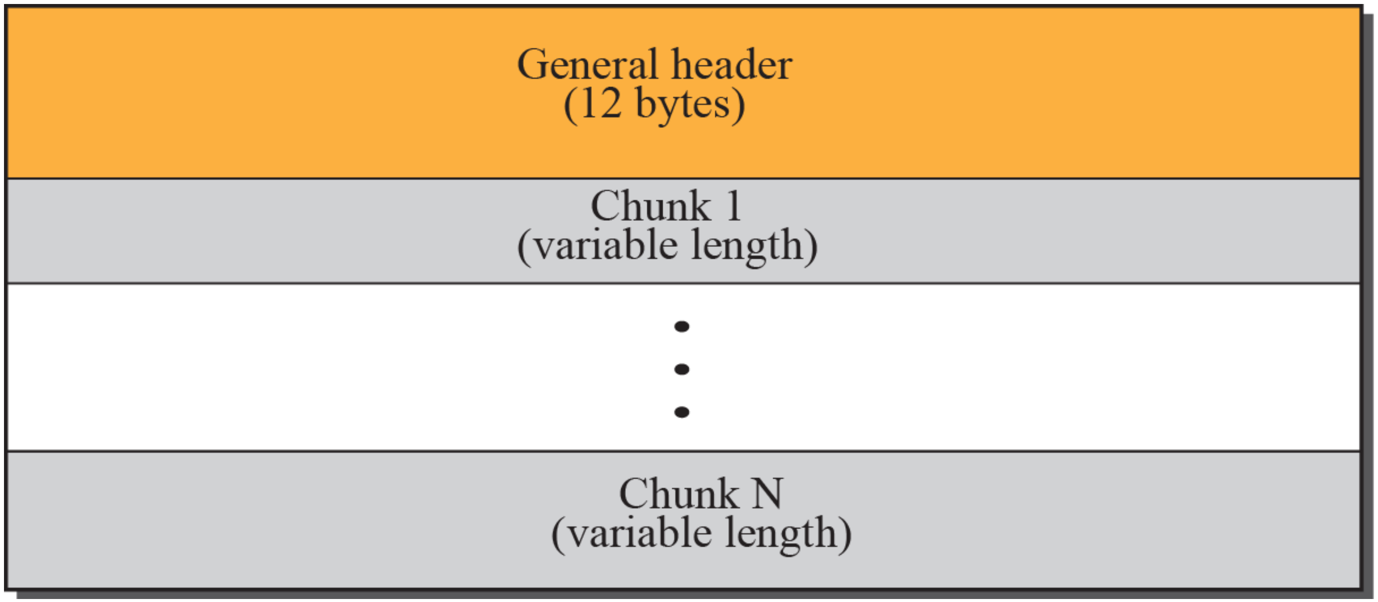


SCTP PACKET FORMAT

- In this section, we show the format of a packet and different types of chunks.
- An SCTP packet has a mandatory general header and a set of blocks called chunks.
- There are two types of chunks:

1. control chunks and
2. data chunks.

SCTP packet format



Note

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

General header (Common layout of a chunk)

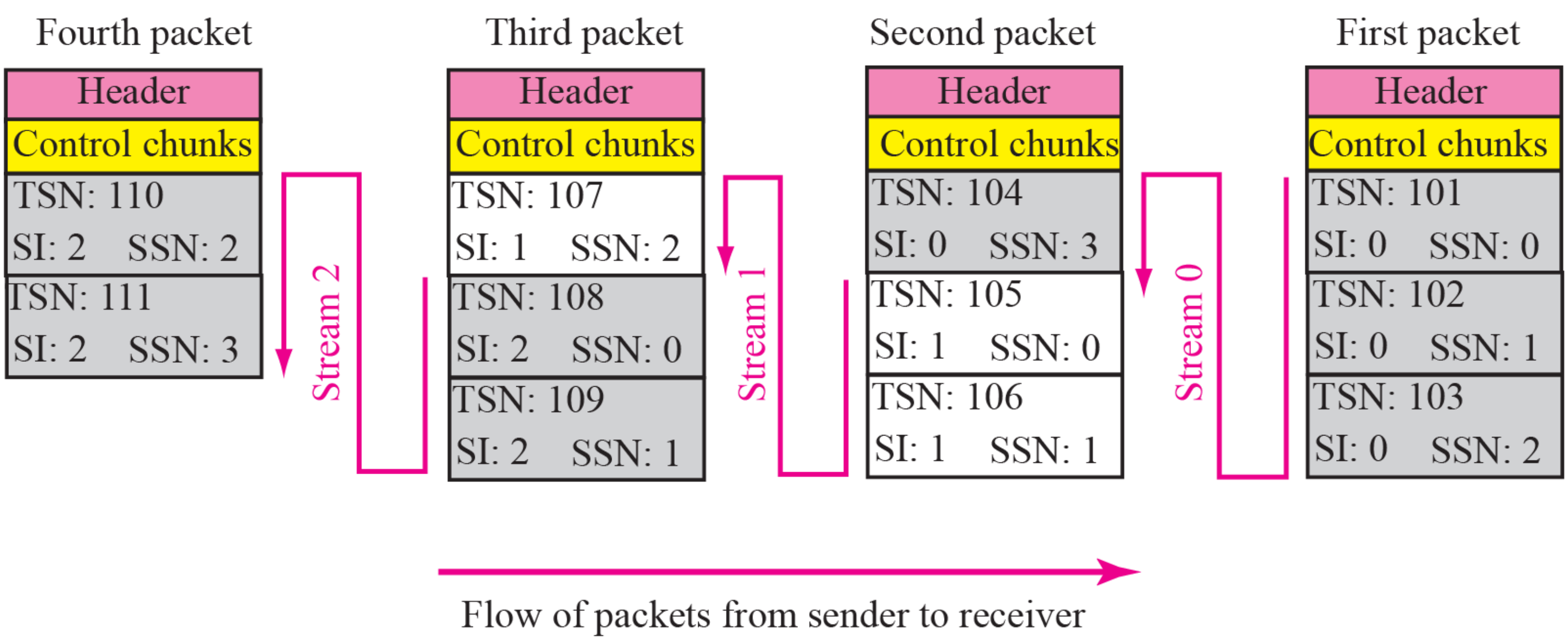
Source port address 16 bits	Destination port address 16 bits
Verification tag 32 bits	
Checksum 32 bits	

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

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

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

Packet, data chunks, and streams



Chapter 2: ROAD MAP

- Transport Layer Introduction
- Port Address
- UDP
- TCP
- Socket Programming using TCP and UDP
- SCTP
- RTP (Real Time Transport Protocol)
- TCP in wireless network
- Quality of services

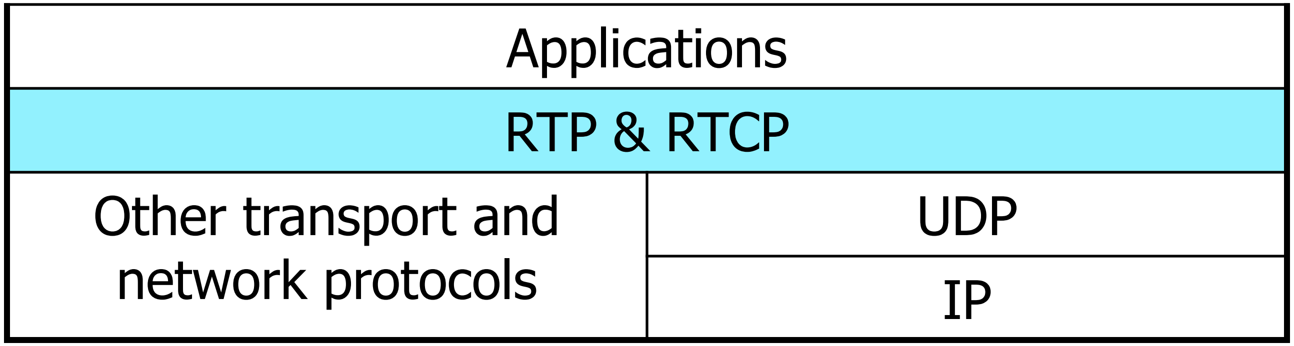
RTP: A Transport Protocol for Real-Time Applications

Introduction

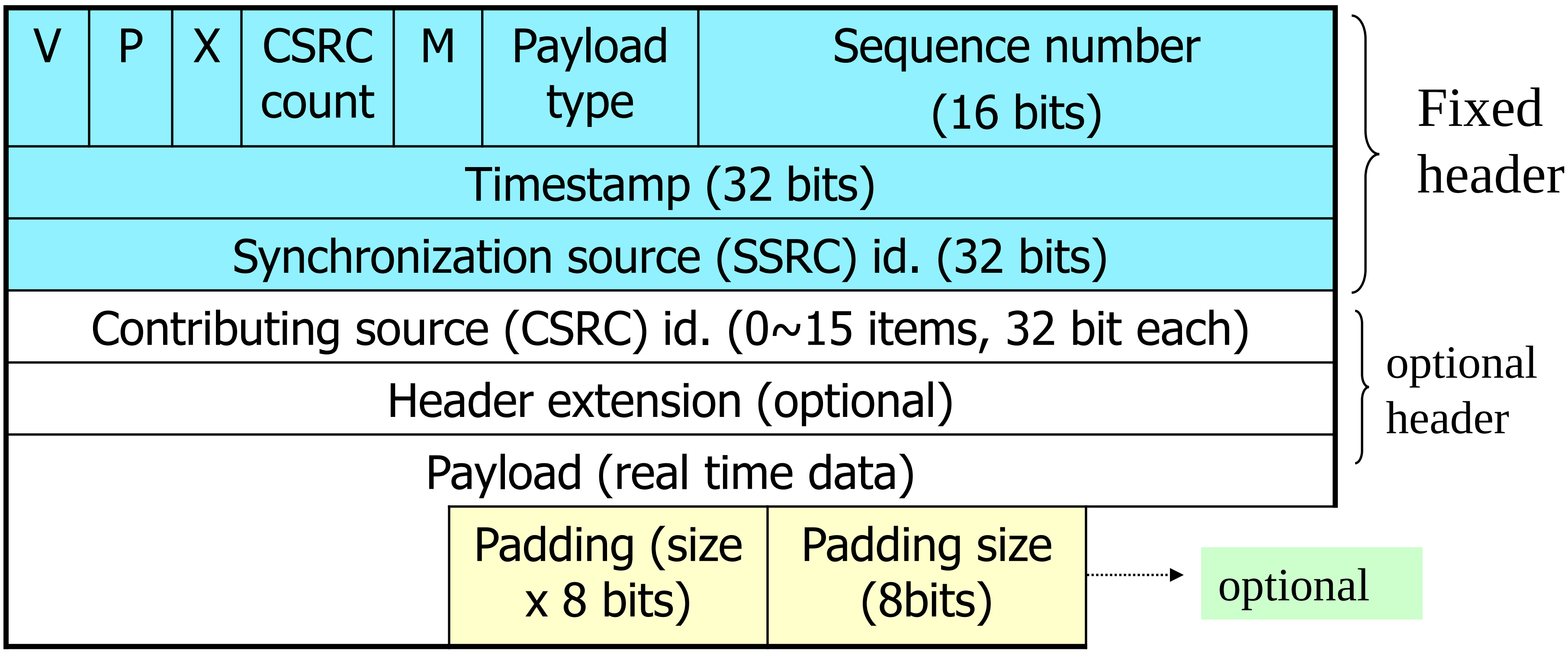
- Internet standard for real-time data
  - Interactive audio, video, and simulation data
- Primarily designed for multi-user multimedia conference
  - Session management
  - Scalability considerations
- Provides end-to-end transport functions for real-time applications
  - Payload type identification
  - Sequence numbering
  - Timestamping
  - Delivery monitoring

Introduction - cont.

- Containing two closely linked parts: data + control
  - RTP: Real-time transport protocol
    - Carry real-time data
  - RTCP: RTP control protocol
    - QoS monitoring and feedback
    - Session control
- Architecture



RTP - packet format



- Version (V, 2bits): =2
- Padding(P, 1bit): If set, last byte of payload is padding size
- Extension(X, 1bit): If set, variable size header extension exists

RTP - header

- CSRC count (4 bits): number of Contributors, max 16 can be possible
- Marker (1 bit): defined in *profile,* mark end of data
- Payload type (7 bits): Audio/Video encoding scheme
- Sequence number: random initial value, increase by one for each RTP packet; for loss detection and seq. restoration
- SSRC: identify source; chosen randomly and locally; **collision** needs to be resolved
- CSRC list: id. of contributing sources, inserted by *mixer*

Chapter 2: ROAD MAP

- Transport Layer Introduction
- Port Address
- UDP
- TCP
- Socket Programming using TCP and UDP
- SCTP
- RTP
- TCP in wireless network
- Quality of Services (QoS)

TCP over Wireless : outline

- TCP over Wireless: Problems
- TCP over Wireless: Solutions/Schemes
  - Split TCP
    1. Indirect TCP
    2. Selective repeat protocol
    3. Mobile TCP
  - TCP-aware link layer
    1. Snoop
    2. WTCP
  - Link layer protocol
  - End-to-end protocol
    1. Selective Acknowledgement
    2. Explicit Loss Notification

TCP over Wireless: Problems

- TCP has been optimized for wired networks.
- Any packet loss is considered to be the result of network congestion and the congestion window size is reduced drastically as a precaution.
- Sources of errors in wireless links:

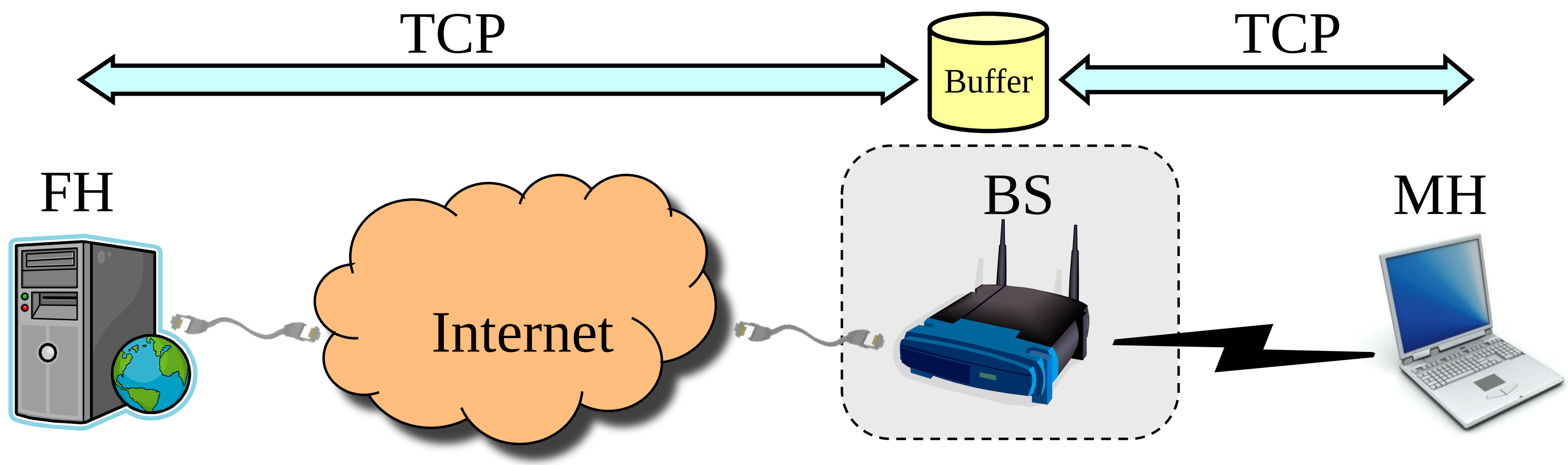
1. Due to hands off between cells
2. Packet losses due to futile transmissions
3. Packet losses due to transmission errors in wireless links

TCP over Wireless : outline

- TCP over Wireless: Problems
- TCP over Wireless: Solutions (Schemes)
  - Split TCP
    1. Indirect TCP
    2. Selective repeat protocol
    3. Mobile TCP
  - TCP-aware link layer
    1. Snoop
    2. WTCP
  - Link layer protocol
  - End-to-end protocol
    1. Selective Acknowledgement
    2. Explicit Loss Notification

Split TCP: Indirect TCP

- I-TCP splits end-to-end TCP connection into two connections
  - Fixed host to BS
  - BS to mobile host
- Two TCP connections with independent flow/congestion control contexts
- Packets buffered at BS

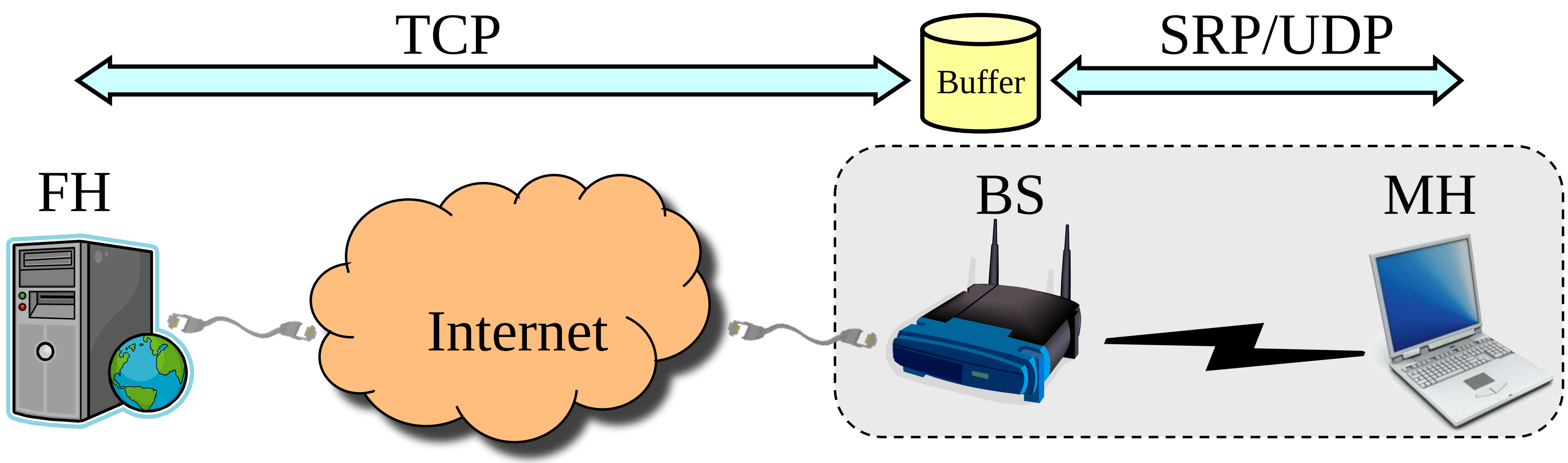


Split TCP: Indirect TCP

- Pros
  - Separates flow and congestion control of wireless and wired
    --higher throughput at sender
- Cons
  - Breaks TCP end-to-end semantics
    - Ack at FH does not mean MH has received the packet
  - BS failure causes loss of data
    - Neither FH nor MH can recover the data
  - On path change, data has to be forwarded to new BS
  - Wireless part is the bottleneck

Split TCP: Selective Repeat Protocol

- Similar to I-TCP but uses SRP/UDP (Selective Repeat Protocol over UDP) over wireless link, Improving End-to-End Performance of TCP over Mobile Internetworks
- Pros
  - Better performance over wireless links
- Cons
  - All cons of I-TCP except last one



Split-TCP: Mobile TCP

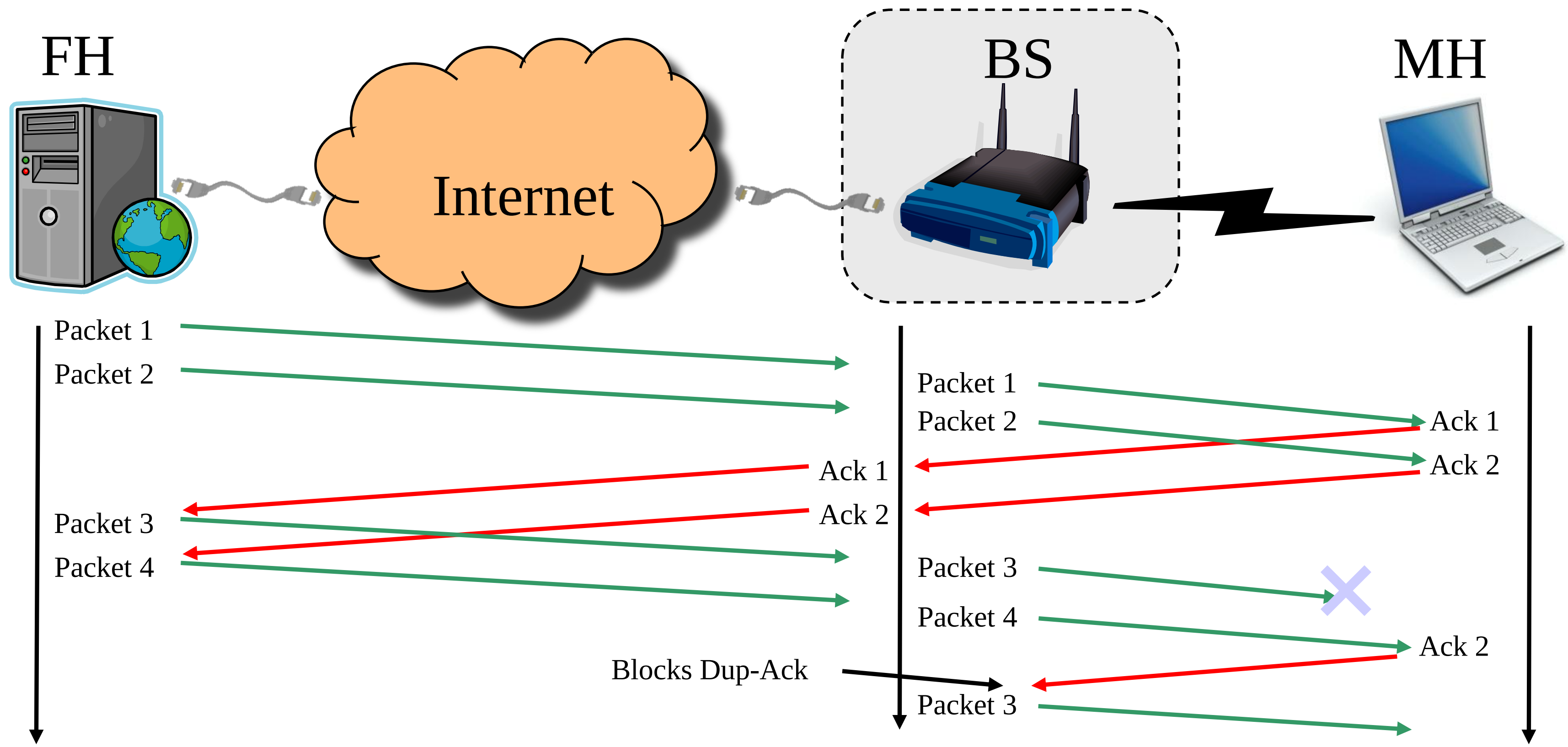
- Similar to I-TCP but tries to keep TCP end-to-end semantics
- No buffering , no retransmission at base station BS.
- BS only monitors all packets and only acks the last packet after it is received by MH
- Pros
  - Data will be recovered eventually after BS failure
  - BS buffer does not overflow
- Cons
  - Worse performance
  - Still not exactly the TCP semantics

TCP over Wireless : outline

- TCP over Wireless: Problems
- TCP over Wireless: Solutions (Schemes)
  - Split TCP
    1. Indirect TCP
    2. Selective repeat protocol
    3. Mobile TCP
  - TCP-aware link layer
    1. Snoop
    2. WTCP
  - Link layer protocol
  - End-to-end protocol
    1. Selective Acknowledgement
    2. Explicit Loss Notification

TCP-aware Link Layers: Snoop

- Link layer is aware of TCP traffic
- BS caches data and monitors acks. Retransmits on duplicate acks and drops duplicate acks



TCP-aware Link Layers: Snoop

- Pros
  - No modification to FH and MH
  - BS only keeps soft state—BS failure does not break TCP
- Cons
  - Does not work with encrypted packets
  - Does not work if data packets and acks traverse different paths
  - Increases RTT—high timeout

Chapter 2: ROAD MAP

- Transport Layer Introduction
- Port Address
- UDP
- TCP
- Socket Programming using TCP and UDP
- SCTP
- RTP (Real Time Transport Protocol)
- TCP in wireless network
- Quality of services (QoS)

Quality of Service

- Requirements
- Techniques for Achieving Good Quality of Service
- Integrated Services
- Differentiated Services

Requirements

1. Reliability
2. Jitter
3. Delay
4. Bandwidth

Requirements

Reliability- *Reliability* is concerned with the ability of a *network* to carry out a desired operation according to its specifications

Jitter- **Jitter** is defined as a variation in the delay of received packets.

Delay- is the amount of time required to transmit packets.

Bandwidth- amount of information that can be transmitted over a **network** in a given amount of time

Requirements

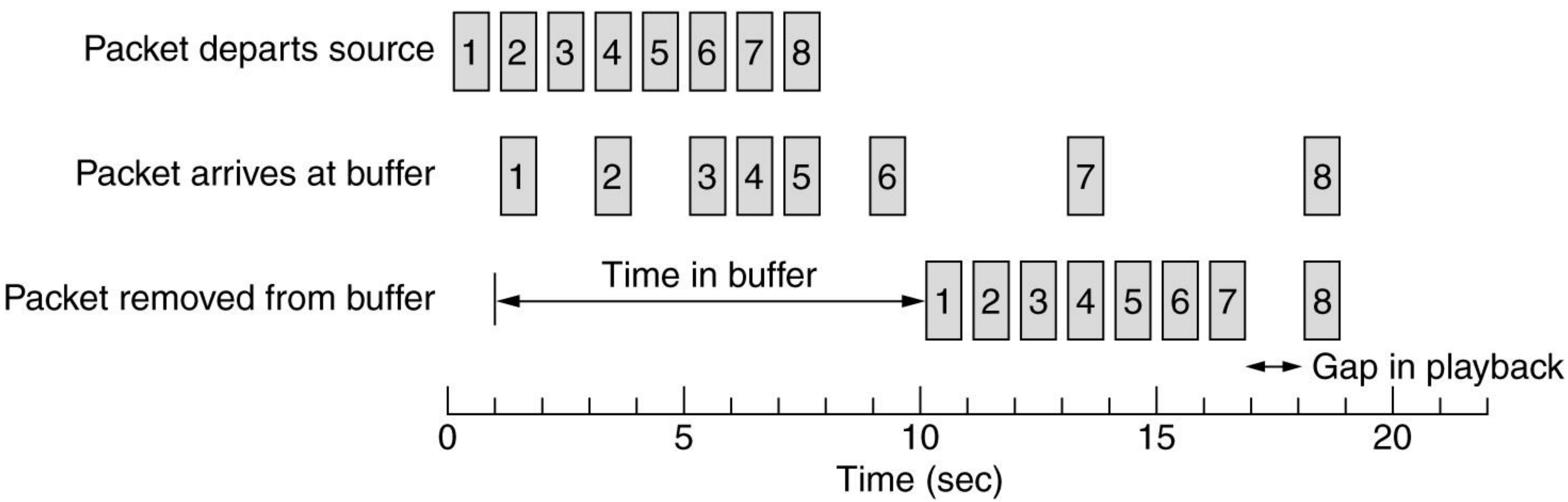
Application	Reliability	Delay	Jitter	Bandwidth
E-mail	High	Low	Low	Low
File transfer	High	Low	Low	Medium
Web access	High	Medium	Low	Medium
Remote login	High	Medium	Medium	Low
Audio on demand	Low	Low	High	Medium
Video on demand	Low	Low	High	High
Telephony	Low	High	High	Low
Videoconferencing	Low	High	High	High

Techniques to achieve Good QoS

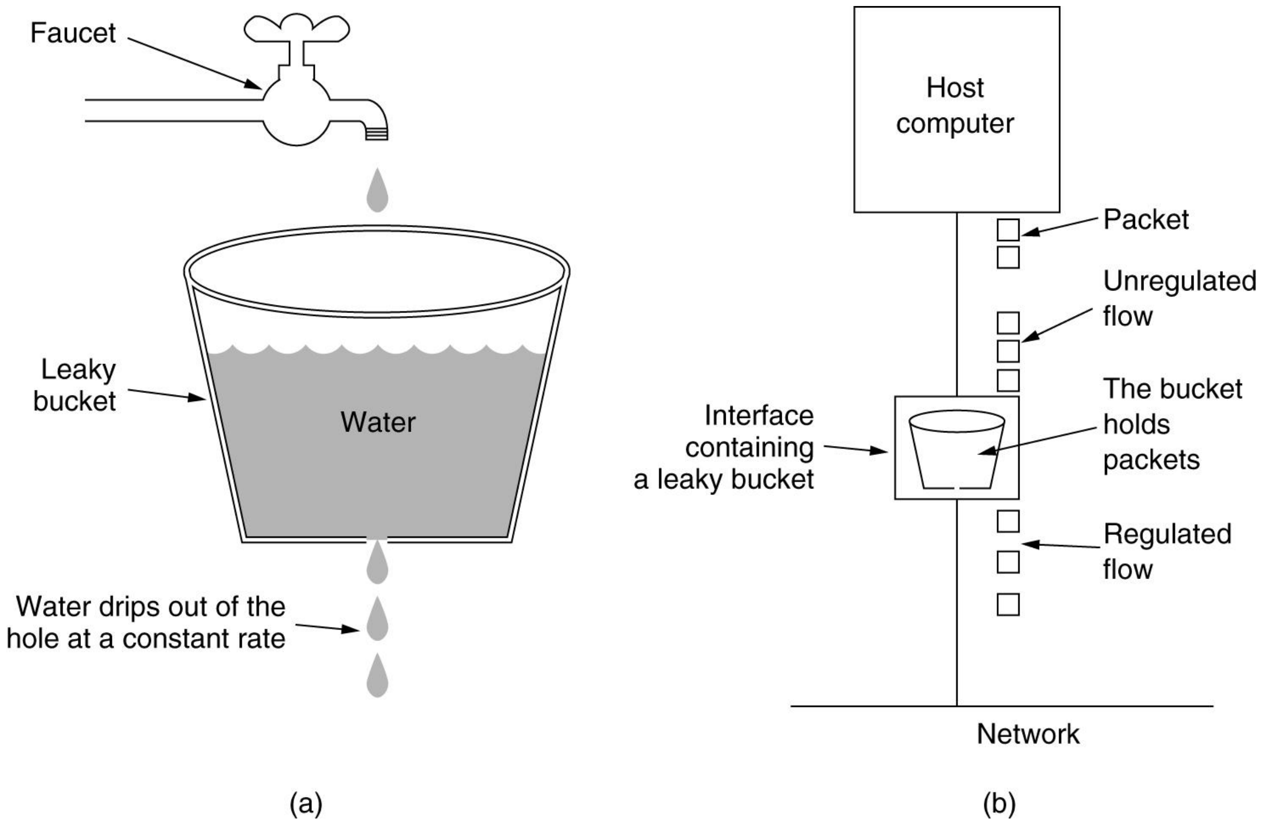
- Buffering
- Traffic Shaping
- Leaky bucket algorithm
- Token bucket algorithm
- Resource reservation
- Admission control
- Packet scheduling

Buffering

Smoothing the output stream by buffering packets.

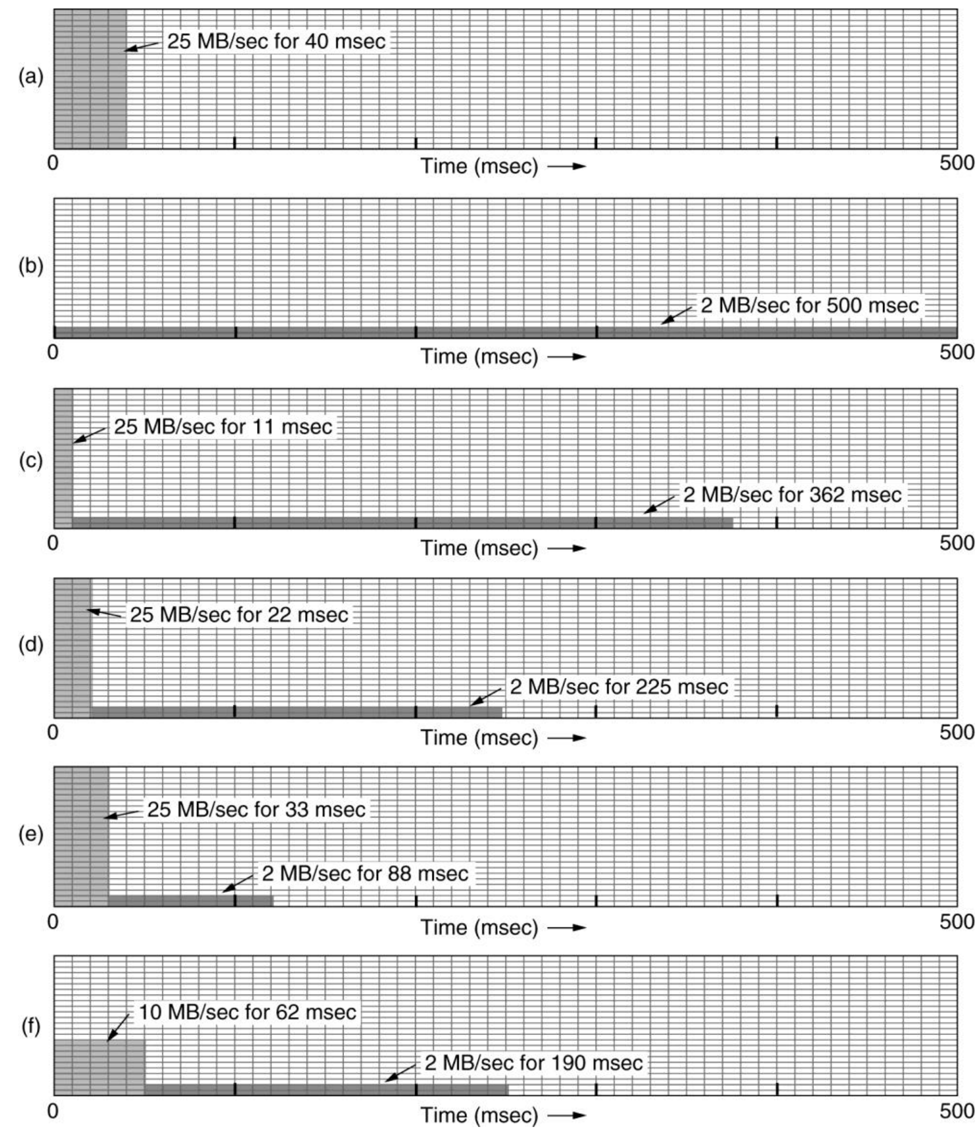


The Leaky Bucket Algorithm



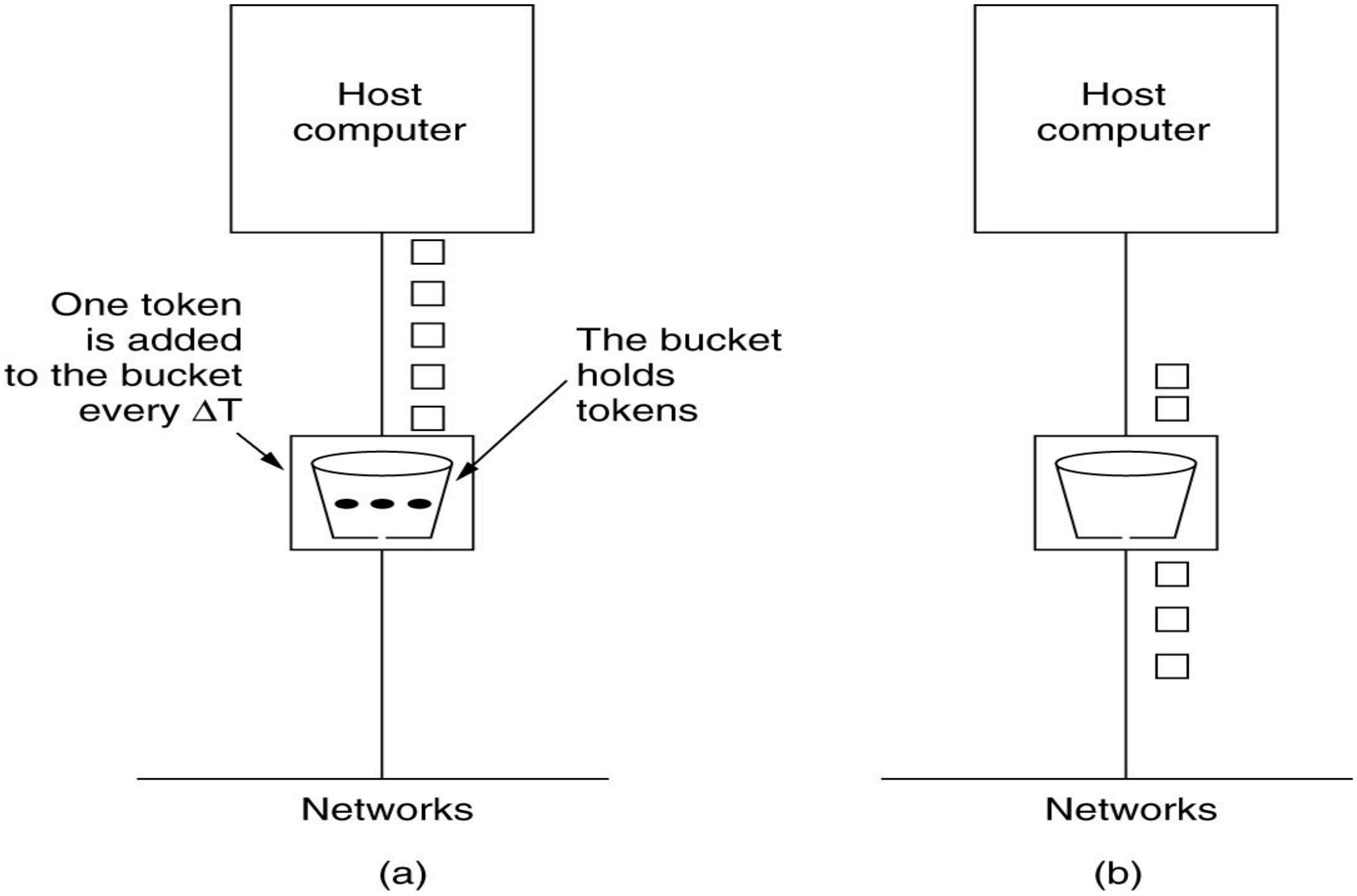
(a) A leaky bucket with water. (b) a leaky bucket with packets.

The Leaky Bucket Algorithm



(a) Input to a leaky bucket. (b) Output from a leaky bucket. Output from a token bucket with capacities of (c) 250 KB, (d) 500 KB, (e) 750 KB, (f) Output from a 500KB token bucket feeding a 10-MB/sec leaky bucket.

The Token Bucket Algorithm



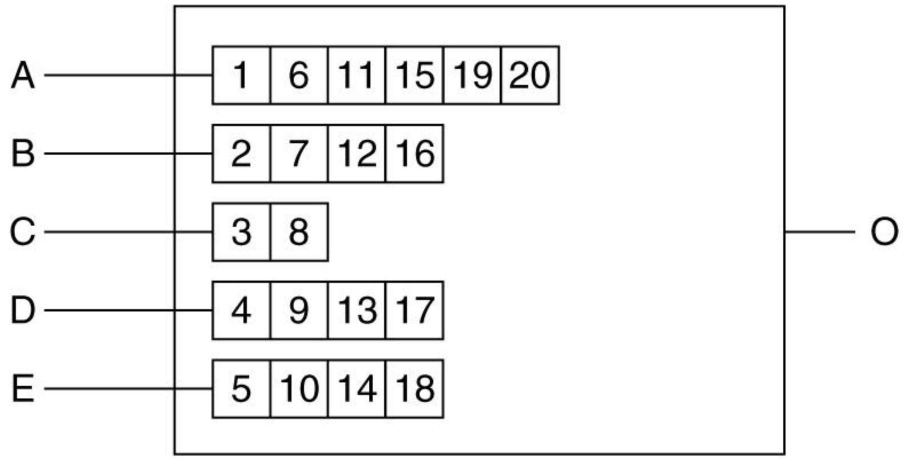
(a) Before. (b) After.

Admission Control

An example of flow specification.

Parameter	Unit
Token bucket rate	Bytes/sec
Token bucket size	Bytes
Peak data rate	Bytes/sec
Minimum packet size	Bytes
Maximum packet size	Bytes

Packet Scheduling



(a)

Packet	Finishing time
C	8
B	16
D	17
E	18
A	20

(b)

(a) A router with five packets queued for line O.

(b) Finishing times for the five packets.

Integrated Services

- Flow based QoS model
- Which means used need to create a flow, a kind of virtual circuit from source to destination and inform all routers about the resource requirement
- This kind of reservation of resources is done by a protocol called RSVP(Resource Reservation Protocol)

Integrated Services

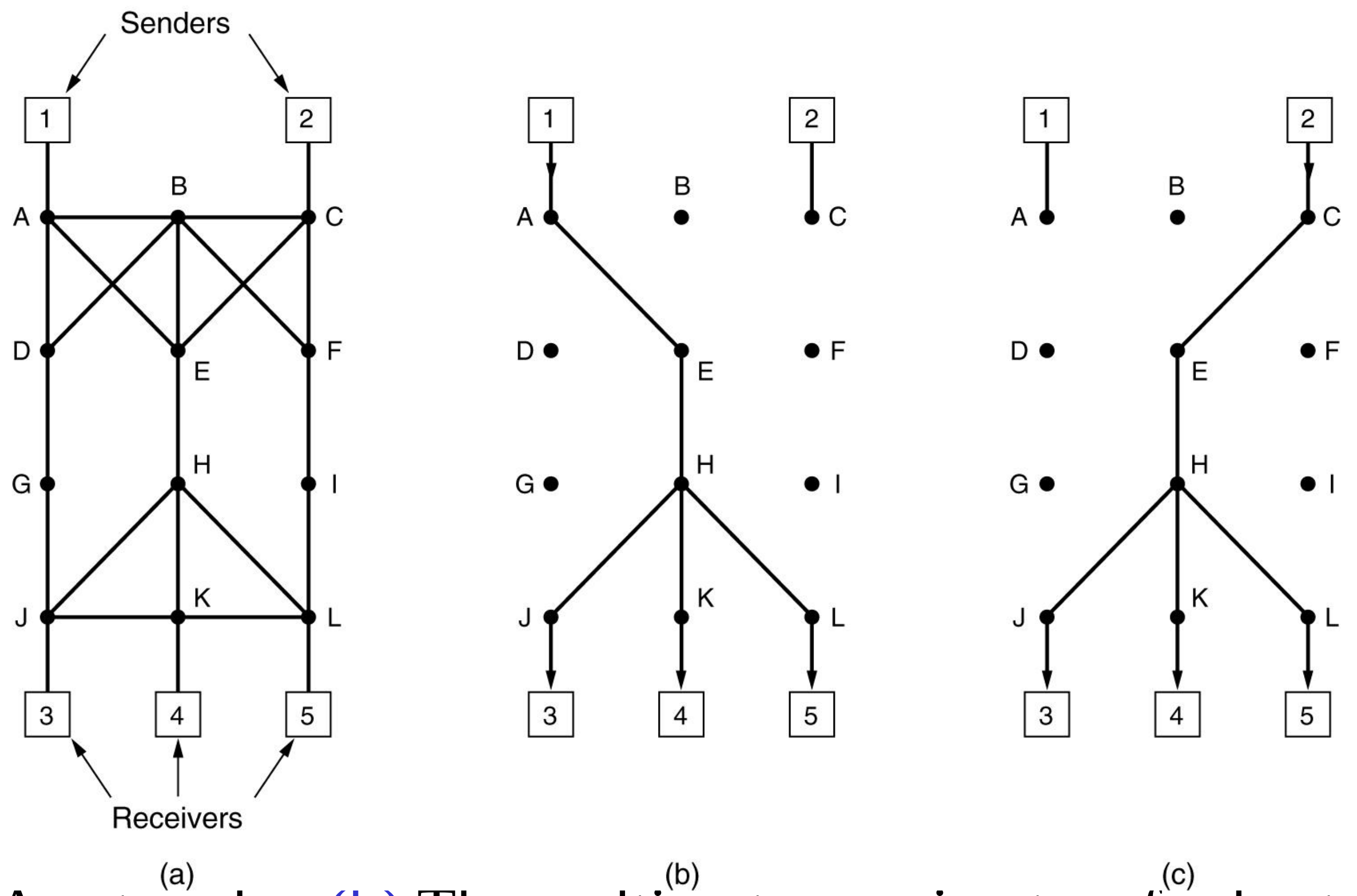
- Resource reservation means reserve how much buffer, bandwidth etc is needed.
- When a router receives flow specification from an application, it decides to admit or deny the service
- Two classes of service is defined for Integrated serviced

1. Guaranteed Service Class(For real time application)
2. Controlled-load Service(For application require reliablility)

RSVP-The Resource ReSerVation Protocol

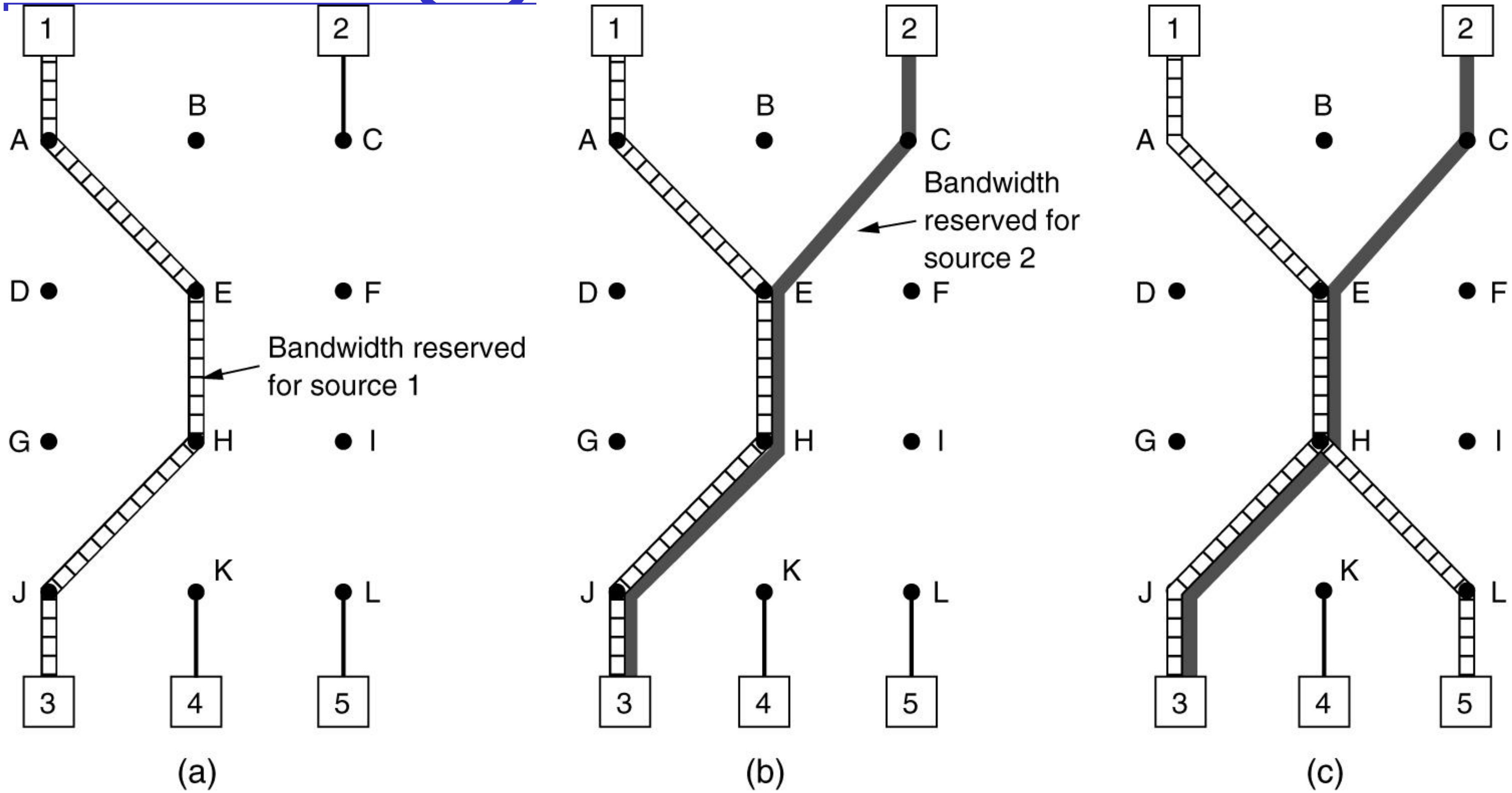
- The **Resource Reservation Protocol** (**RSVP**) is a Transport layer protocol designed to reserve resources across a network for an Integrated service network.
- RSVP operates over an IPV4 or IPV6 and provides resource reservations for multicast or unicast data flows
- RSVP can be used by either host or routersto request or deliver specific levels of quality of service (QoS) for application data streams or flows.
- RSVP defines how applications place reservations and how they can give up the reserved resources once the need for them has ended.

RSVP-The Resource ReSerVation Protocol



(a) A network, (b) The multicast spanning tree for host 1.
(c) The multicast spanning tree for host 2.

RSVP-The ReSerVation Protocol (2)



(a) Host 3 requests a channel to host 1. (b) Host 3 then requests a second channel, to host 2. (c) Host 5 requests a channel to host 1.

Problems with Integrated Services

Scalability: Each router keep information for each flow. So does not possible to scale more

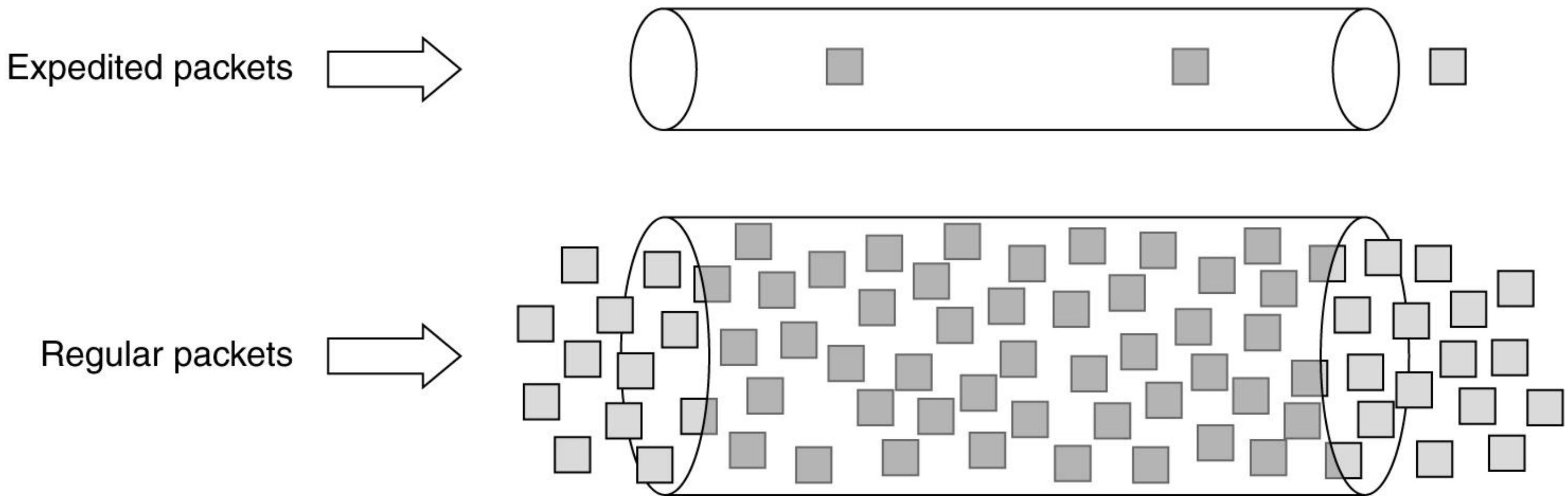
Service Type Limitation: Only two types of services are provided guaranteed and control based

Differentiated Services

- Handles shortcomings of Integrated Services .
- In differentiated model router does not store information about flows.
- No advance reservation is required
- It is a Class based service model
- Each packet contains a field called DS field
- It has two types of models

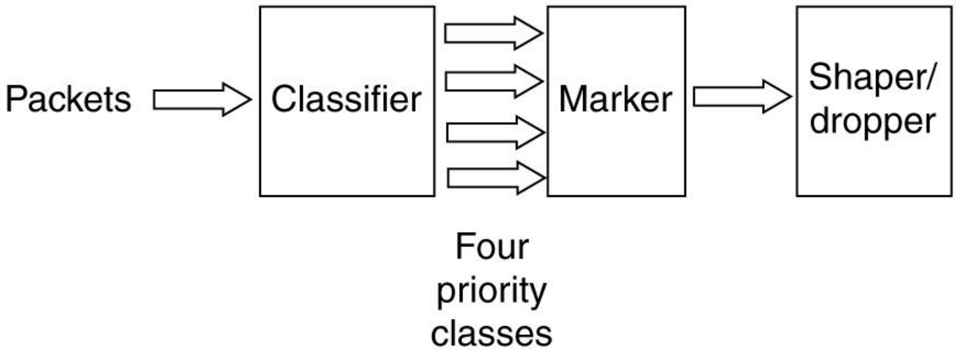
1. Expedited forwarding
2. Assured Forwarding

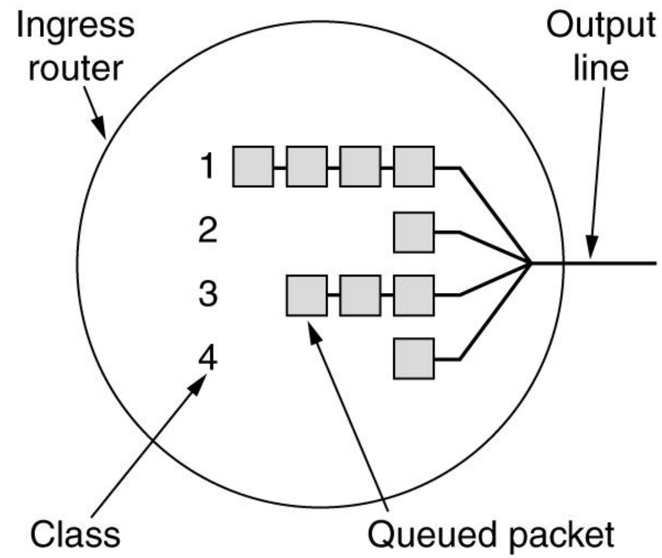
Expedited Forwarding



- In this model two classes of service is available: 1>Regular 2> Expedited
- Expedited packets experience a traffic-free network.

Assured Forwarding





- There are 4 priority classes , each having 3 discard policies like low,medium and high.
- Traffic controller have Classifier,Marker and Shaper/Dropper
- Packet is classified according to priority, then marked according to their class .
- Shaper/dropper filter these packet that may drop or delay the packet.

Thank You .