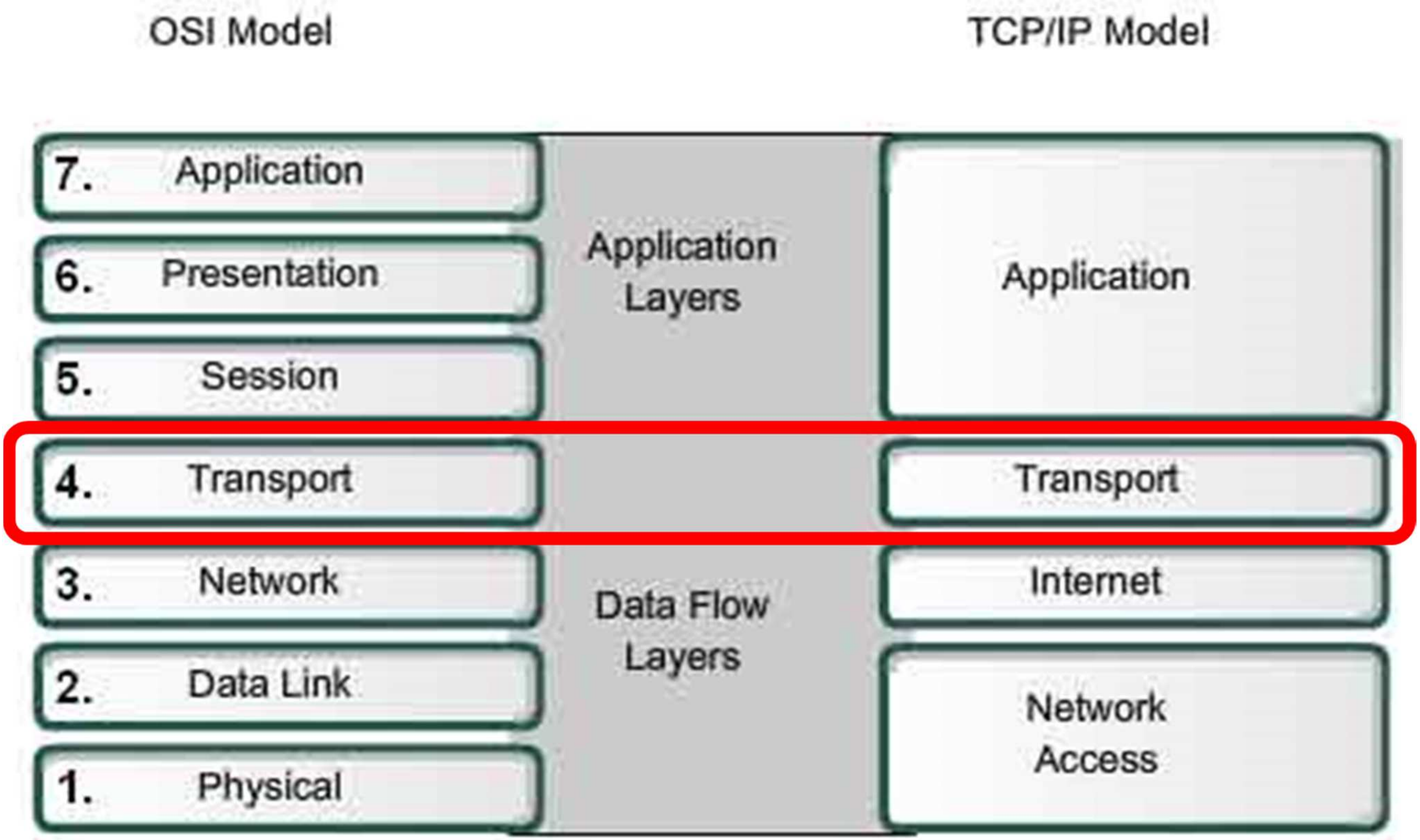
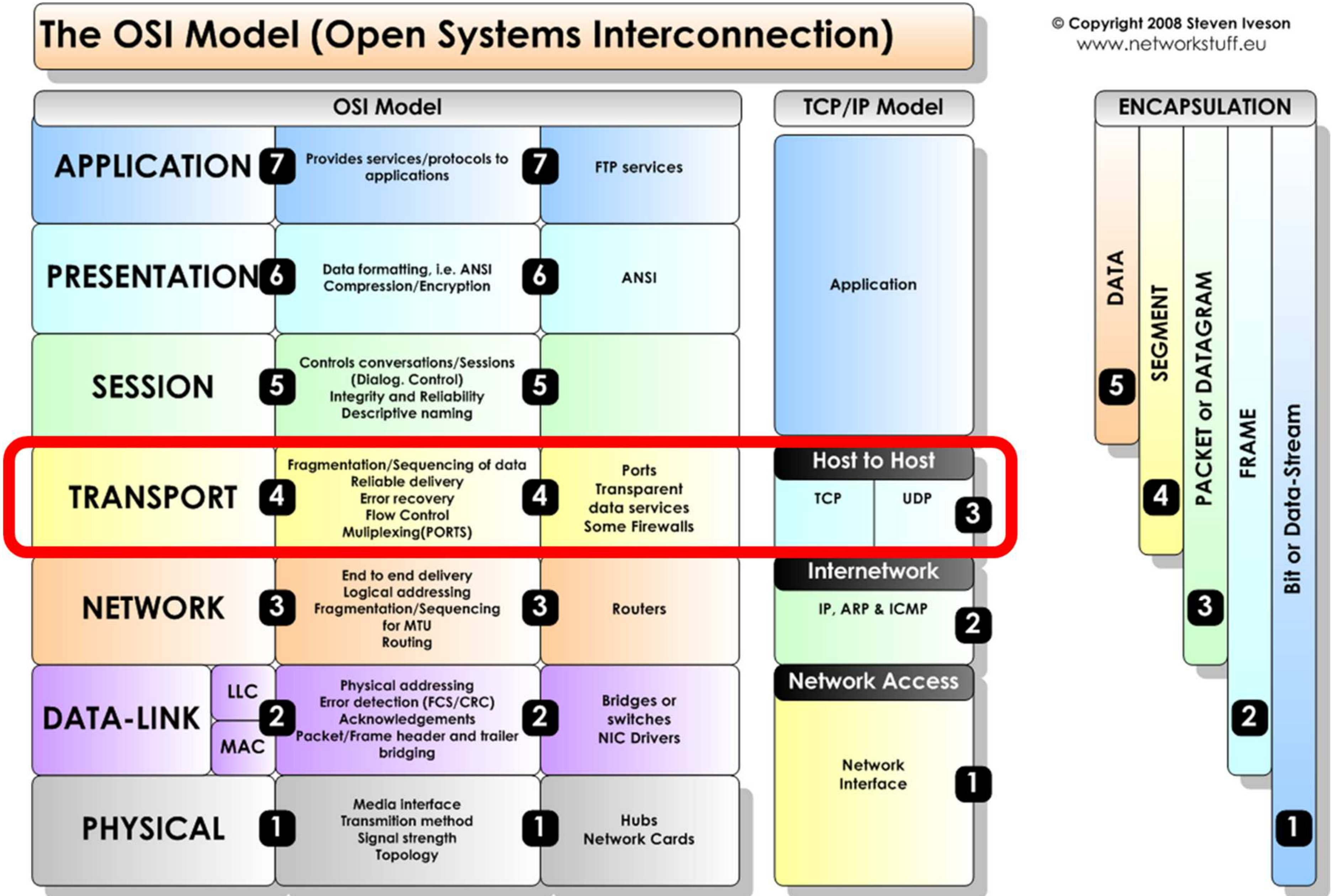

OSI Model
TCP/IP Model
7. Application
6. Presentation
5. Session
4. Transport
3. Network
2. Data Link
1. Physical
Application Layers
Data Flow Layers
Application
Transport
Internet
Network Access


The OSI Model (Open Systems Interconnection)
© Copyright 2008 Steven Iveson
www.networkstuff.eu
OSI Model
TCP/IP Model
ENCAPSULATION
APPLICATION 7
Provides services/protocols to applications
7
FTP services
PRESENTATION 6
Data formatting, i.e. ANSI Compression/Encryption
6
ANSI
SESSION 5
Controls conversations/Sessions (Dialog. Control) Integrity and Reliability Descriptive naming
5
TRANSPORT 4
Fragmentation/Sequencing of data Reliable delivery Error recovery Flow Control Muliplexing(PORTS)
4
Ports Transparent data services Some Firewalls
NETWORK 3
End to end delivery Logical addressing Fragmentation/Sequencing for MTU Routing
3
Routers
DATA-LINK
LLC
MAC
2
Physical addressing Error detection (FCS/CRC) Acknowledgements Packet/Frame header and trailer bridging
2
Bridges or switches NIC Drivers
PHYSICAL 1
Media interface Transmition method Signal strength Topology
1
Hubs Network Cards
Application
Host to Host
TCP
UDP
3
Internetwork
IP, ARP & ICMP
2
Network Access
Network Interface
1
DATA 5
SEGMENT 4
PACKET or DATAGRAM 3
FRAME 2
Bit or Data-Stream 1

Transport-level Protocols

Connection Oriented

Transmission Control Protocol (TCP)

Logical connection

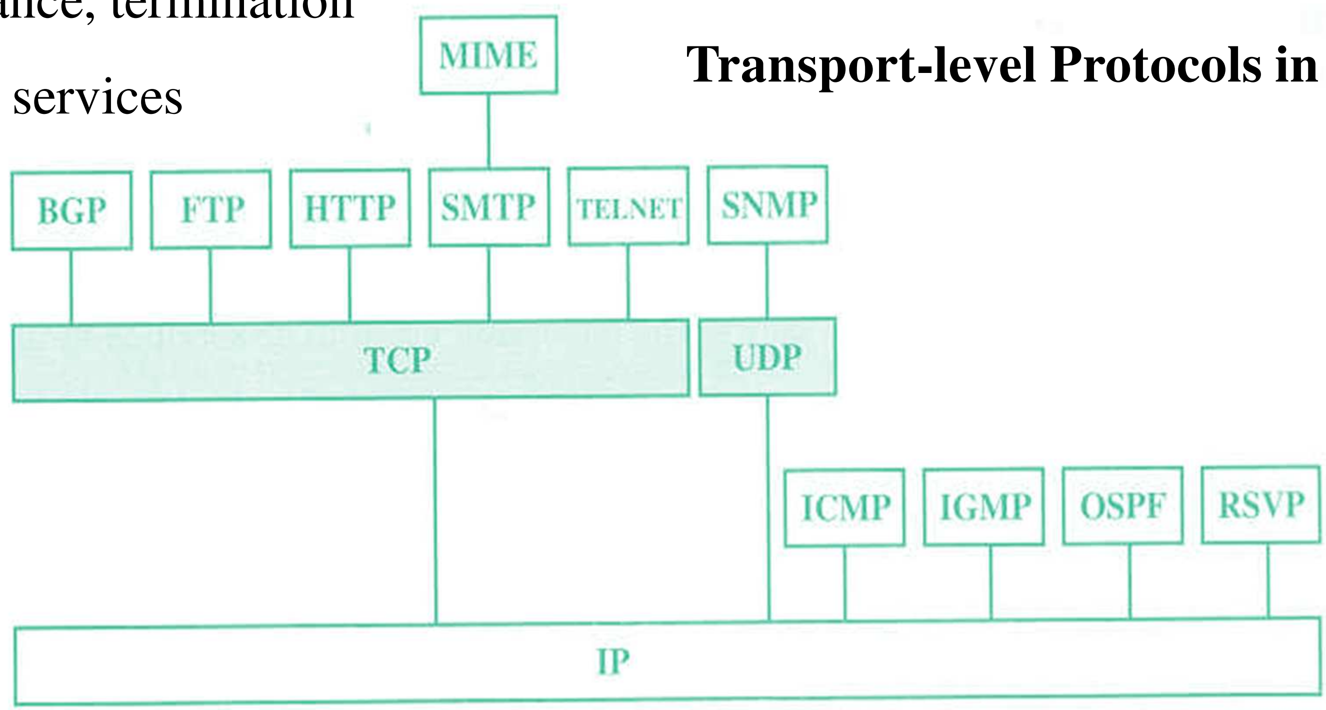
- Establishment
- Maintenance, termination
- Reliable services

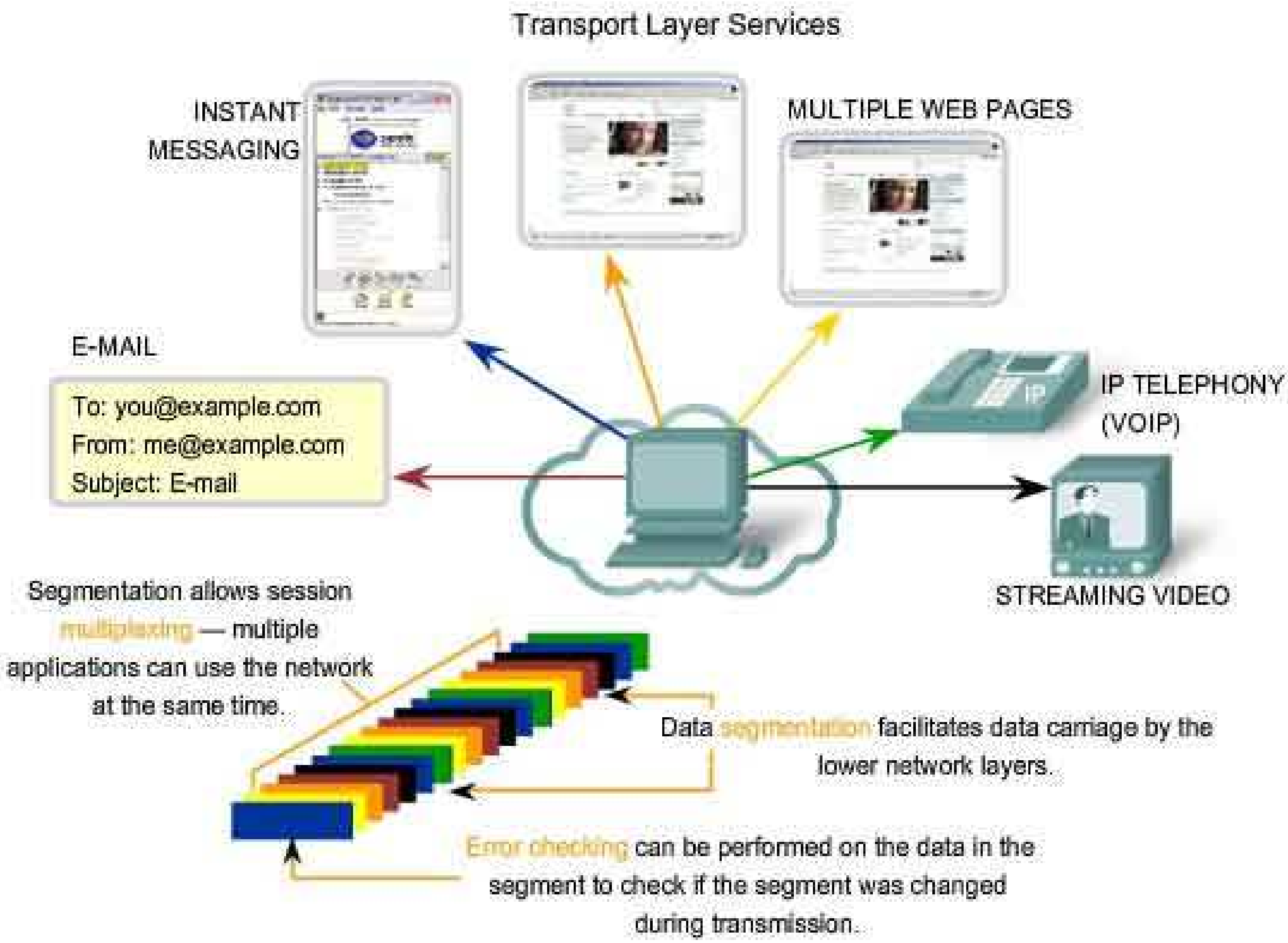
Connectionless

User Datagram Protocol (UDP)

- Connectionless
- ‘Best-effort’ delivery

Transport-level Protocols in Context




Transport Layer Services
INSTANT MESSAGING
MULTIPLE WEB PAGES
E-MAIL
To: you@example.com
From: me@example.com
Subject: E-mail
IP TELEPHONY (VOIP)
STREAMING VIDEO
Segmentation allows session multiplexing — multiple applications can use the network at the same time.
Data segmentation facilitates data carriage by the lower network layers.
Error checking can be performed on the data in the segment to check if the segment was changed during transmission.

Transmission Control Protocol (TCP)

- Connection oriented
- Reliable byte stream over unreliable IP
- IP may be transported over many different network technologies
- RFC 793
- Each end: a socket
- Socket is a pair of: IP address + port number
- Multiple connections may be active on a socket
- Full duplex connections
- Point to point links
- No multicast or broadcast
  - Other protocols used for these
- May buffer information to increase amount sent
  - PUSH flag requests that buffered data be sent

Communicating computers must agree on a port number

`Server' opens selected port and waits for incoming messages

`Client' selects local port and sends message to selected port

Services provided by many computers use reserved, *well-known* port numbers:

TFTP, DNS, Echo

Other services use *dynamically assigned* port numbers

Port	Name	Description
7	echo	Echo input back to sender
9	discard	Discard input
11	systat	System statistics
13	daytime	Time of day (ASCII)
17	quote	Quote of the day
19	chargen	Character generator
37	time	System time (seconds since 1970)
53	domain	DNS
69	tftp	Trivial File Transfer Protocol (TFTP)
123	ntp	Network Time Protocol (NTP)
161	snmp	Simple Network Management Protocol (SNMP)

Transmission Control Protocol (TCP)

TCP general features

- *Connection oriented*: Application requests connection to destination and then uses connection to deliver data to transfer data
- *Point-to-point*: A TCP connection has two endpoints
- *Reliability*: TCP guarantees data will be delivered without loss, duplication or transmission errors
- *Full duplex*: The endpoints of a TCP connection can exchange data in both directions simultaneously
- *Stream interface*: Application delivers data to TCP as a continuous *stream*, with no record boundaries; TCP makes no guarantees that data will be received in same blocks as transmitted
- *Reliable connection establishment*: *Three-way handshake* guarantees reliable, synchronized startup between endpoints
- *Graceful connection termination*: TCP guarantees delivery of all data after endpoint shutdown by application

TCP Services

Reliable communication between pairs of processes (applications)

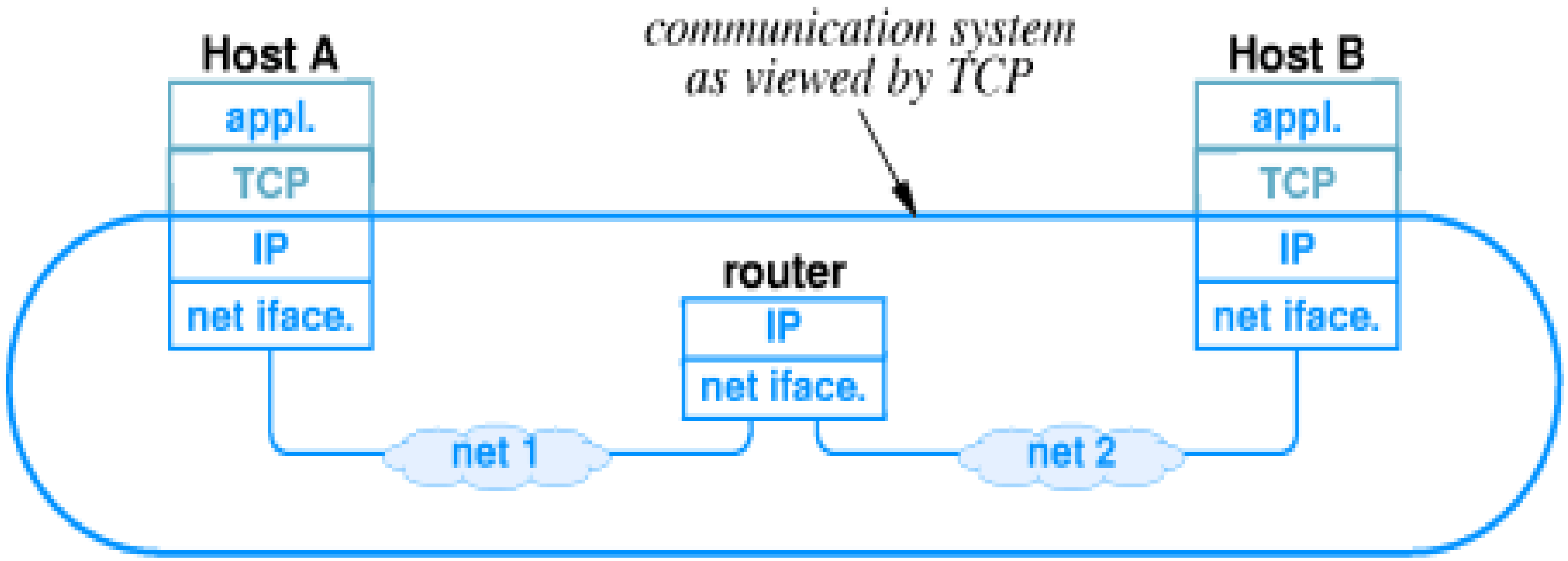
Across variety of reliable and unreliable networks and internets

Two labeling facilities:

- Data stream push
  - TCP user can require transmission of all data up to push flag
  - Receiver will deliver in same manner
  - Avoids waiting for full buffers
- Urgent data signal
  - Indicates urgent data is upcoming in stream
  - User decides how to handle it

TCP uses IP for data delivery

- Endpoints are identified by ports (sockets)
    - -16-bit number
    - --System Ports range (0-1023): “well-known ports” which provide well-known services; assignment through IANA
      https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml
    - --User Ports range (1024-49151) are available for assignment through IANA
    - --Dynamic Ports range (49152-65535) have been specifically set aside for local and dynamic use and cannot be assigned through IANA.
- Allows multiple connections on each host
- Ports may be associated with an application or a process
- IP treats TCP like data and does not interpret any contents of the TCP message



TCP services at the boundary with the process (application level) are defined using the concepts of abstract service primitives (TCP ASPs) and service-access points (SAPs). The primitives are implemented using the segment header fields or passing some parameters at the IP level.

Items Passed to IP

TCP passes some parameters down to IP

- Precedence of segments
- Normal delay/low delay
- Normal throughput/high throughput
- Normal reliability/high reliability
- Security

Also, TCP Protocol:

Breaks application messages into *segments* (TCP data units)

Each segment has an (at least) 20 byte header

Segments are sized based on:

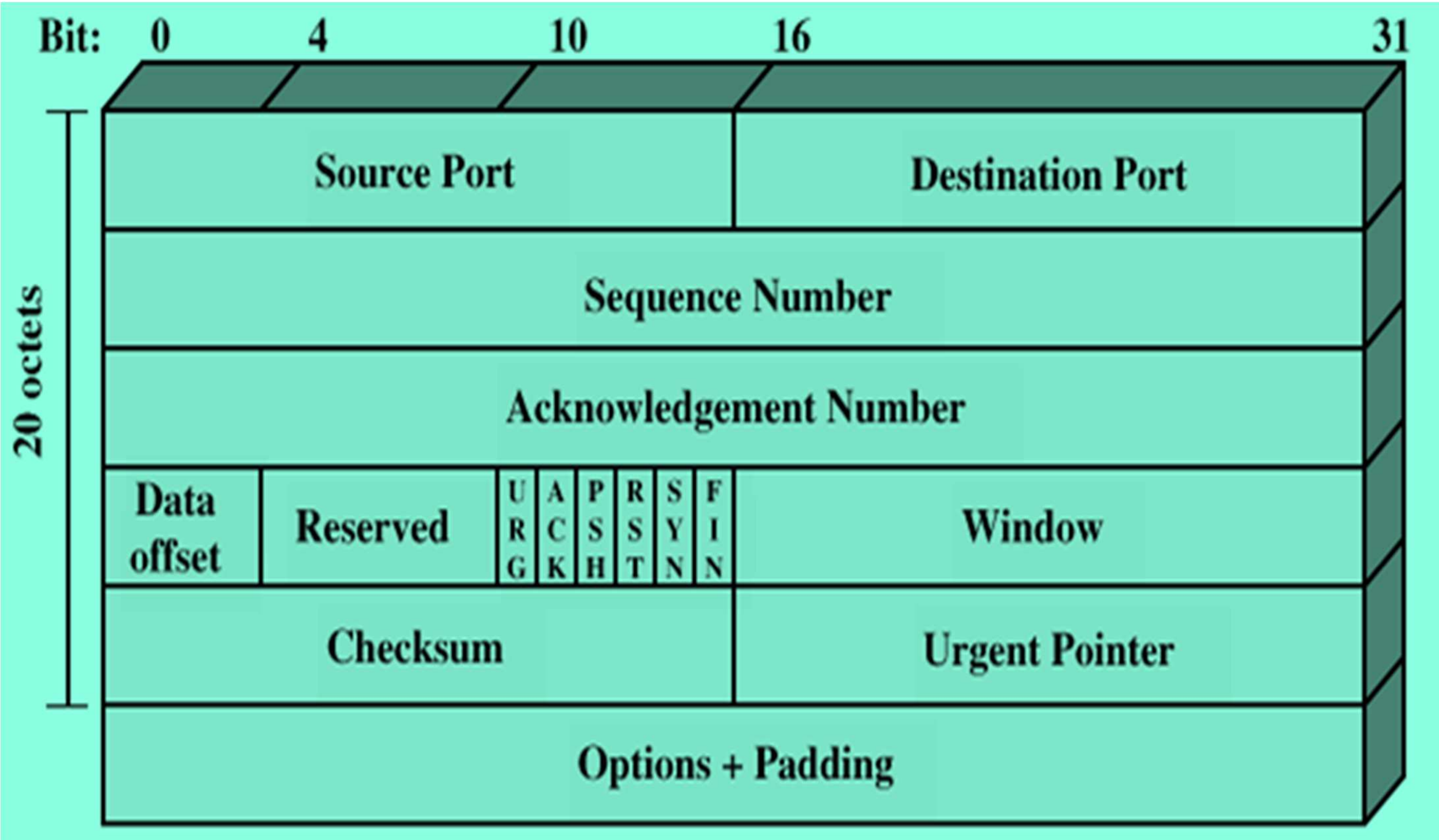
being less the 64koctets (the path Maximum Transmission Unit (MTU))

Segments may be *fragmented* during transmission if MTU smaller than packet

TCP Header Fields

Source port – source TCP user
Destination port – destination TCP user
Sequence number – sequence number of the first data octet in this segment
Acknowledgement number – piggybacking acknowledgement, contains sequence number of next data octet the destination TCP entity expects to receive
Data offset – number of words (32 bit) in this header (header flexible length)
Reserved – for future use & developments

TCP Header



Bit:
0
4
10
16
31
20 octets
Source Port
Destination Port
Sequence Number
Acknowledgement Number
Data offset
Reserved
URG
ACK
PSH
RST
SYN
FIN
Window
Checksum
Urgent Pointer
Options + Padding

Flags – 6 bits:

URG – urgent pointer field significant

ACK – acknowledgement field significant

PSH – Push function

RST – reset connection

SYN – synchronize the sequence numbers (used in connection establishment)

FIN – no more data from sender (used in connection termination)

Window – flow control credit allocation in octets (window size)

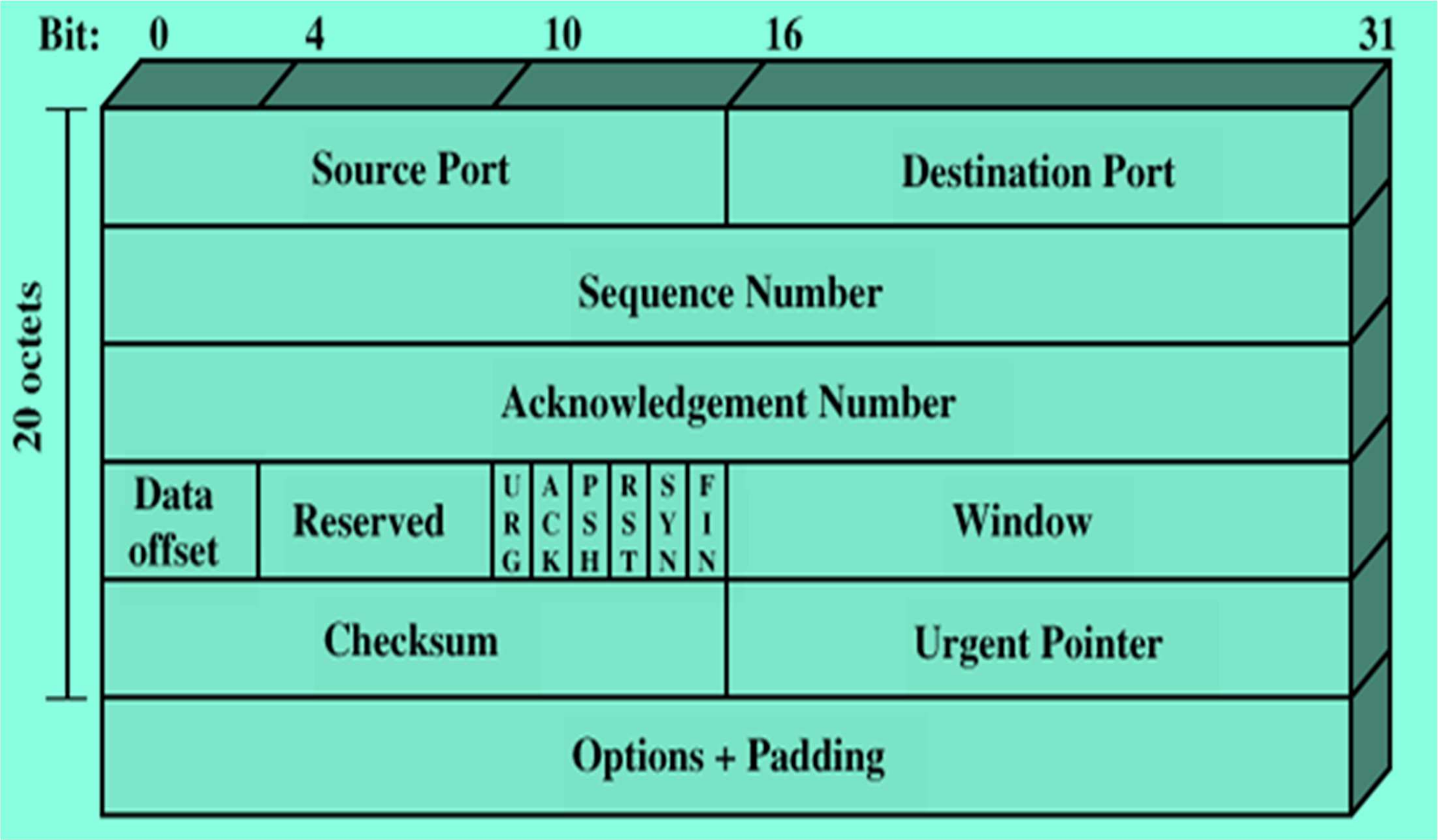
Checksum – ones complement of the sum modulo 2^{16}-1 of all 16-bit words in the segment (plus eventually pseudo-header)

Urgent Pointer – pointer to the last octet in a sequence of urgent data

Options – variable field

Padding – to meet segment length of multiple of 32 bit

TCP Header

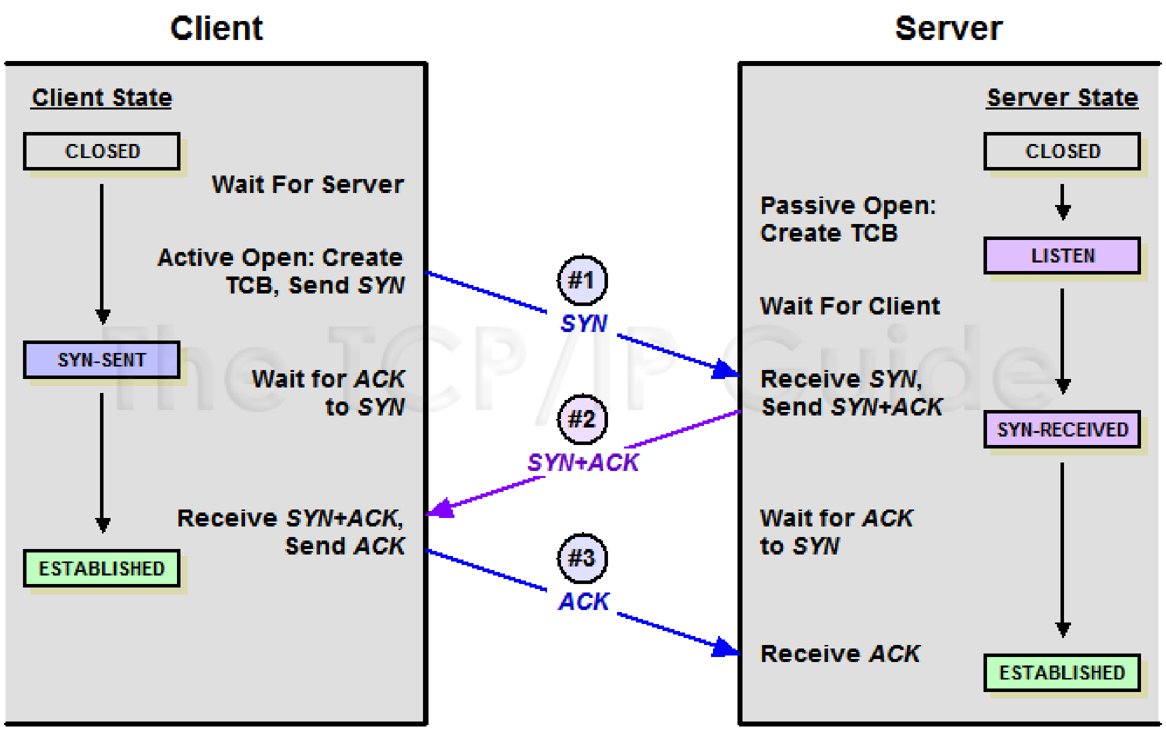


Bit: 0 4 10 16 31
20 octets
Source Port
Destination Port
Sequence Number
Acknowledgement Number
Data offset
Reserved
URG
ACK
PSH
RST
SYN
FIN
Window
Checksum
Urgent Pointer
Options + Padding

TCP Mechanisms

Connection establishment

- Three way handshake used for connection establishment (exchange of SYNs)
- Each TCP connection is established between pair of ports
- One port can connect to multiple destinations (may support multiple connections)



TCP Mechanisms

Connection establishment

- Three way handshake used for connection establishment (exchange of SYNs)
- Each TCP connection is established between pair of ports
- One port can connect to multiple destinations (may support multiple connections)

Data transfer

- Logically, data is considered a stream of octets
- Octets numbered modulo 2^{32}
- Data is transferred over a TCP connection in segments
- Flow control by credit allocation of number of octets
- Data buffered at transmitter and receiver
- Use of *PUSH* flag for forcing transmission of so far accumulated date (end-of-block function)
- User may specify *urgent* data transmission

Connection termination

- Graceful close (normal exchange of FIN info)
  - TCP users issues CLOSE primitive
  - Transport entity sets FIN flag on last segment sent
- Abrupt termination by ABORT primitive
  - Entity abandons all attempts to send or receive data
  - RST segment transmitted (connection Reset)

Implementation Policy Options (allows for possible TCP implementations)

- Send policy
- Deliver policy
- Accept policy
- Retransmit policy
- Acknowledge policy

Send policy

If no *Push* flag or CLOSE indication, a TCP entity transmits at its own convenience

- Used Data buffered at transmit buffer
- TCP entity may construct segment per data batch as provided by user
- May wait for certain amount of data

Policy depends on performance considerations (header overhead/response speed)

Delivery Policy

In absence of *Push*, receiving TCP entity may deliver data to the user at own convenience

- May deliver as each in order segment received
- May buffer data from more than one segment

Policy depends on how promptly user needs data, or how much processing involved (each delivery = application software interrupts)

Acknowledgement policy

- Immediate ACK
- Cumulative ACK

Accept policy

Segments may arrive out of order; options:

- In order
  - Only accept segments in order
  - Discard out of order segments
- In windows
  - Accept all segments within receiver window

Retransmit policy

- TCP maintains queue of segments transmitted but not acknowledged
- Policy depends mainly on receiver acceptance policy (in order or in window)
- TCP will retransmit if not ACKed in given time; retransmission options:
  - First only segment from the queue (necessary one timer for entire queue)
  - Batch – all segments from queue (one timer for entire queue)
  - Individual – one timer for each segment in queue; retransmits the individual segment

TCP Congestion Control

Congestion: a transmission/retransmission takes too much time

Basic idea in congestion avoidance: don't insert a new packet until an old one leaves

Other Basic idea: timeouts associated with retransmission are due to congestion state

Approaches based on re-transmission timer & window size management:

Timers for re-transmission: time-out management

can increase window size if no congestion

TCP Congestion control estimate round trip delay, by observing delay pattern in recent segments, and then sets the timer to a value a little bit greater

Estimations based on:

Simple average

Average Round-Trip Time (ARTT) for a segment *i*, is the simple average of delays for the precedent *k* transmitted segments.

Exponential Average

Gives a better prediction of the next RTT value

Binary exponential Backoff algorithm (see CSMA/CD) may be used for obtaining values for the retransmission time-out values (RTOs)

Exponential RTO Backoff

Since timeout value is probably due to congestion (dropped packet or long round trip), maintaining same RTO is not a good idea

RTO increased each time a segment is re-transmitted

RTO = q*RTO

Commonly q=2

Known as Binary exponential backoff

Window management

TCP Window size may affect transmission parameters; need for managing size:

Used Techniques

if ack is sent in good time, increase window size, if congestion appears, decrease window size and make client wait longer to retransmit

Slow Start

If start using window size as provided by past connection, may cause excessive flow, internet conditions may differ

Solution: sender starting with a smaller window size

Two windows: allowed window and congestion window, sized in segments, no octets

Allwd_wndw = min (cngst_wndw, gained_credit)

cngst_wndw starts with value 1, then increments every acknowledged segment

Dynamic window sizing on congestion

A possible scenario: When first segment lost, means collision sign, so reset value of cngst_wndw to 1 and begin ‘slow start’

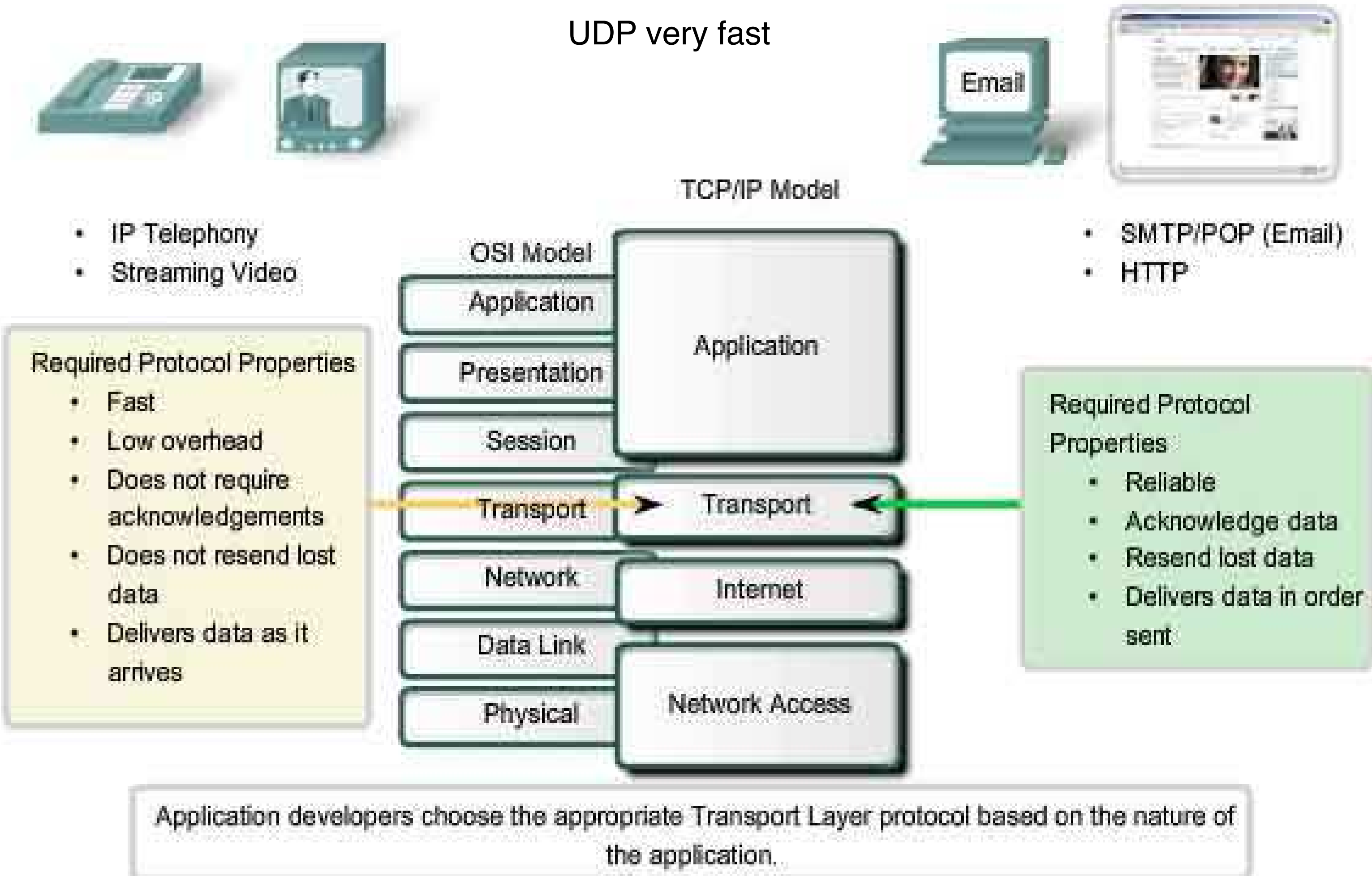
There are more others ….

low overhead: not wait for too long to stabilize connection tcp used in protocols

TCP used in protocols where i need to make sure that information/ data arrives

Transport Layer Protocols

UDP very fast



User Datagram Protocol (UDP)

Connectionless

Less overhead

Used mostly for

real-time applications (voice, video, telemetry), with no need for retransmissions

non-critical functions: inward data collections (monitoring …), outward data collections (broadcasted announcements …)

Specified by RFC 768

Unreliable

Delivery and duplication control not guaranteed

used good for DNS also, but not really reliable

UDP delivers independent messages, called *datagrams* between applications or processes on host computers

`Best effort' delivery - datagrams may be lost, delivered out of order, etc.

Checksum (optionally) guarantees integrity of data

For generality, endpoints of UDP are called *protocol ports* or *ports*

Each UDP data transmission identifies the internet address and port number of the destination and the source of the message (port, socket … as TCP)

UDP header is very simple:

- Port numbers
- Message length
- Checksum (optional, if yes, use same 1s complement checksum as IP)

UDP Header