HO CHI MINH CITY UNIVERSITY OF SCIENCE
FACULTY OF ELECTRONICS AND TELECOMMUNICATIONS
DEPARTMENT OF TELECOMMUNICATIONS AND NETWORKS

COURSE

COMPUTER NETWORKS

Chapter 07

TCP TRANSPORT

Reference: Peter L Dordal, "An Introduction to Computer Networks," Jul 26, 2019

Lecturer: Nguyen Viet Ha, Ph.D. Email: nvha@fetel.hcmus.edu.vn

1 Transmission Control Protocol

1. Transmission Control Protocol

❖TCP is:

- ➢Stream-oriented
  - Application can write data in very small or very large amounts and the TCP layer will take care of appropriate packetization.
- ➢Connection-oriented
  - Established before the beginning of any data transfer.
- ➢Reliable
  - Correct order of delivery
  - Timeout/retransmission mechanism
- ➢Congestion control
  - TCP automatically uses the sliding windows algorithm to achieve throughput relatively close to the maximum available.

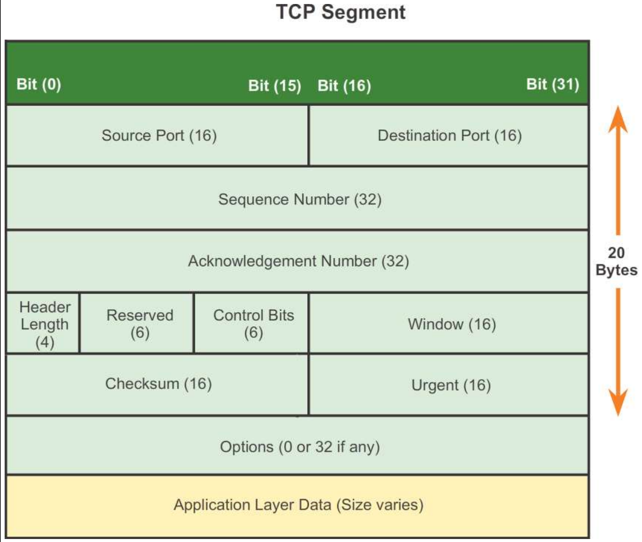
1. Transmission Control Protocol

❖The End-to-End Principle

- ➢It states in effect that transport issues are the responsibility of the **endpoints** (not the core network).
  - **Data corruption**
    - For the first, even though essentially all links on the Internet have link-layer checksums to protect against data corruption, TCP still adds its own checksum.
  - **Congestion**
    - TCP is today essentially the **only layer** that addresses congestion management.

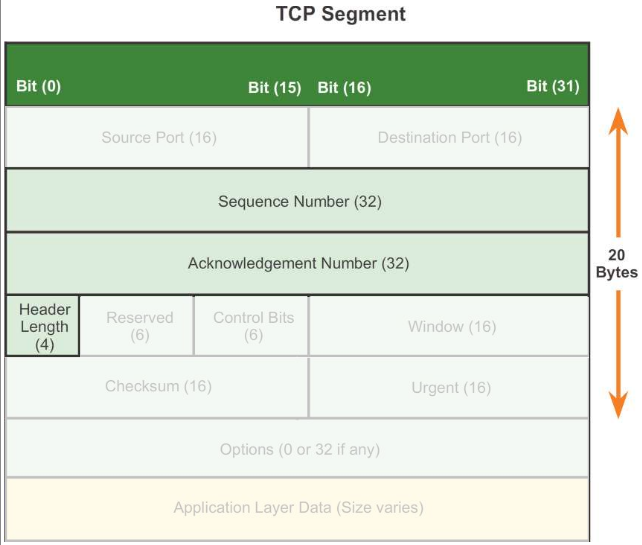
2 TCP Header

2. TCP Header



❖It is traditional to refer to the data portion of TCP packets (PDU – Packet Data Unit) as **segments.**

2. TCP Header



Sequence number (32 bits) - numbering the data, at the byte level.

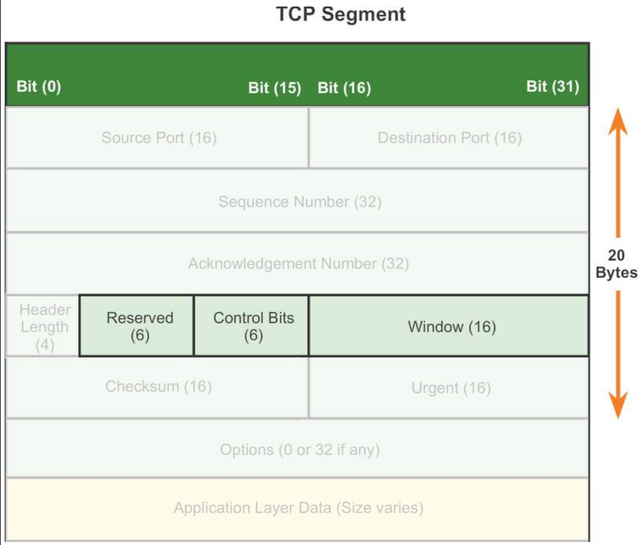
- The first byte of the **current** data payload.

Acknowledgement number (32 bits) - Indicates the data that has been received.

- The first byte of the **next** data payload.

Header length (4 bits) - Indicates the length of the TCP segment header.

2. TCP Header

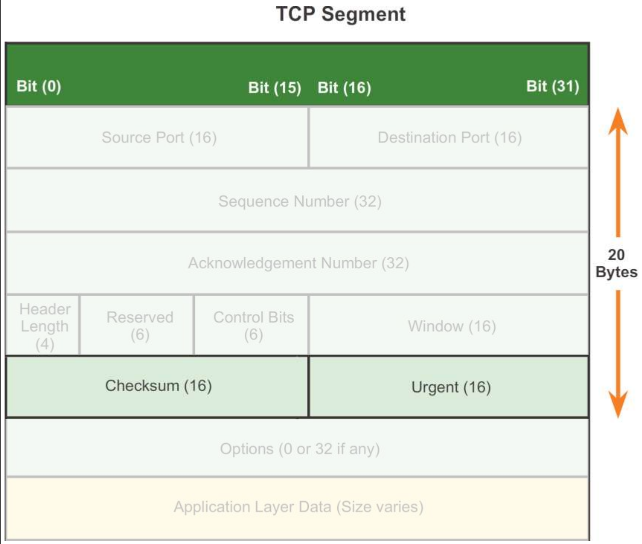


Reserved (6 bits) - is reserved for the future.

Control bits (6 bits) - Includes bit codes, or flags, that indicate the purpose and function of the TCP segment.

Window size (16 bits) - Indicates the number of segments that can be accepted at one time.

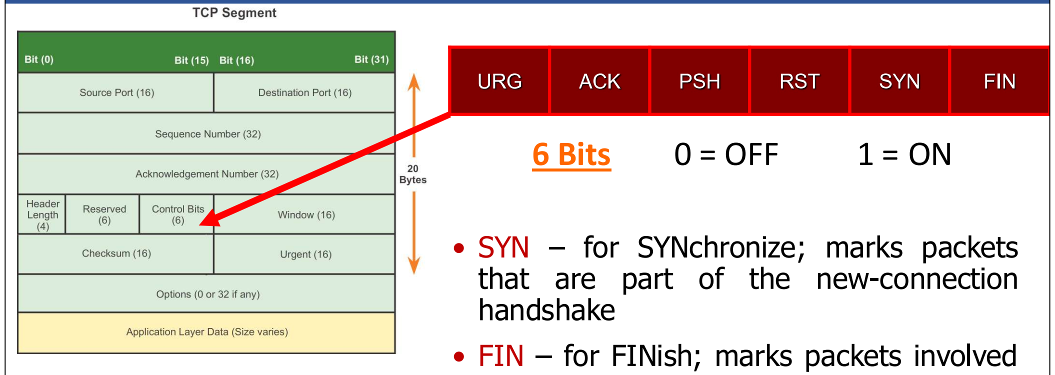
2. TCP Header



Checksum (16 bits) - Used for error checking of the segment header and data.

Urgent pointer (16 bits) - Indicates if data is urgent.

TCP Connection Establishment



6 Bits 0 = OFF 1 = ON

- SYN – for SYNchronize; marks packets that are part of the new-connection handshake
- FIN – for FINish; marks packets involved in the connection closing
- RST – for ReSeT; indicates various error conditions
- ACK – indicates that the header Acknowledgment field is valid; that is, all but the first packet.
- PSH – for PuSH; marks "non-full" packets that should be delivered promptly at the far end.
- URG – for URGent; part of a now-seldom-used mechanism for high-priority data.

TCP Connection Establishment

❖PSH:

- ➢If A sends a series of small packets to B, then B has the option of assembling them into a full-sized I/O buffer before releasing them to the receiving application.
  - However, if A sets the **PSH** bit on each packet, then B should release each packet immediately to the receiving application.

TCP Connection Establishment

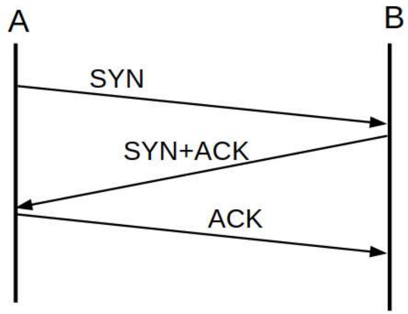
❖URG:

- ➢In telnet connection, A sent a large amount of data to B. Suddenly, A wishes to abort that processing by sending the interrupt character CNTL-C.
- ➢**Under normal conditions**, the application at B would have to finish processing all the pending data before getting to the CNTL-C.
- ➢However, if the **URG bit** is set, and the **TCP header's Urgent Pointer** field points to the CNTL-C in the current packet, the receiving application then skips ahead in its processing of the arriving data stream until it **reaches the urgent data**.

3 TCP Connection Establishment

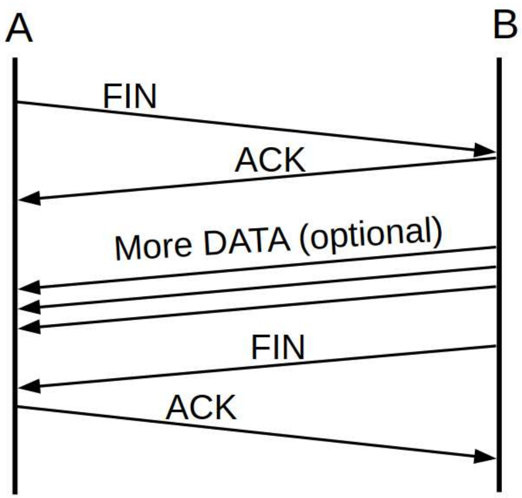
3. TCP Connection Establishment

❖TCP connections are established via an exchange known as the **three-way handshake**.



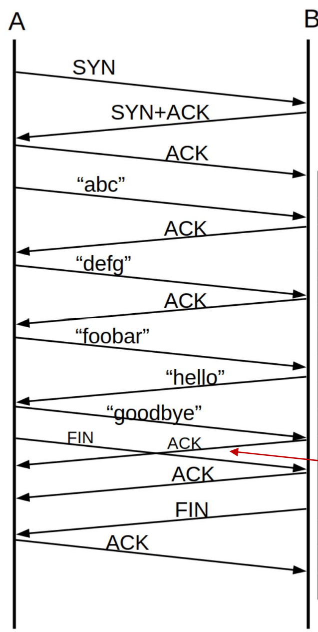
3. TCP Connection Establishment

❖Close the connection: **two-way FIN/ACK handshakes**.



3. TCP Connection Establishment

❖Example of a full exchange of packets in a representative connection.



	A sends	B sends
1	SYN, **seq=0**	
2		SYN+ACK, seq=0, **ack=1** (expecting)
3	ACK, **seq=1**, ack=1 (ACK of SYN)	
4	"abc", **seq=1**, ack=1	
5		ACK, seq=1, **ack=4**
6	"defg", **seq=4**, ack=1	
7		seq=1, **ack=8**
8	"foobar", **seq=8**, ack=1	
9		seq=1, **ack=14**, "hello"
10	**seq=14**, ack=6, "goodbye"	
11,12	**seq=21**, ack=6, FIN	seq=6, **ack=21** ;; ACK of "goodbye", crossing packets
13		seq=6, **ack=22** ;; ACK of FIN
14		seq=6, **ack=22**, FIN
15	**seq=22**, ack=7 ;; ACK of FIN	

3. TCP Connection Establishment

- Each side chooses its **Initial Sequence Number (ISN)**, and sends that in its initial SYN.
  - The third ACK of the three-way handshake is an acknowledgment that the server side's SYN response was received correctly.
  - All further sequence numbers sent are the ISN chosen by that side plus the relative sequence number.
  - It helps with the allocation of a sequence number that does not conflict with other data bytes transmitted over a TCP connection.

3. TCP Connection Establishment

- If B had not been LISTENing at the port to which A sent its SYN, its response would have been **RST** ("reset"), meaning in this context "connection refused".
- Similarly, if A sent data to B before the SYN packet, the response would have been RST.
- RST can be sent by either side at any time **to abort** the connection.

4 Path MTU Discovery

4. Path MTU Discovery

- TCP connections are more efficient if they can keep large packets flowing between the endpoints.
- Once upon a time, TCP endpoints included just 512 bytes of data in each packet that was not destined for local delivery, to avoid fragmentation.
- TCP endpoints now typically engage in **Path MTU Discovery** which almost always allows them to send larger packets.
  - Backbone ISPs are now usually able to carry 1500-byte packets.

4. Path MTU Discovery

- The IPv4 strategy is to send an initial data packet with the IPv4 DONT_FRAG bit set.
  - If the ICMP message Frag_Required/DONT_FRAG_Set comes back, or if the packet times out, the sender tries a smaller size.
  - If the sender receives a TCP ACK for the packet, on the other hand, indicating that it made it through to the other end, it might try a larger size.

4. Path MTU Discovery

- IPv6 has no DONT_FRAG bit.
- Path MTU Discovery over IPv6 involves the **periodic** sending of larger packets; if the ICMPv6 message Packet Too Big is received, a smaller packet size must be used.

5 TCP Flow Control

5. TCP Flow Control

- **TCP Sliding Windows** (are measured in terms of bytes)
  - To improve throughput.
  - In the initial three-way handshake, each side specifies the maximum window size it is willing to accept, in the **Window Size** field of the TCP header.
    - This 16-bit field can only go to 65,535 Bytes.
      - **Window Scale** option that can also be negotiated in the opening handshake to increase the Window Size.
    - The window size included in the TCP header is known as the **Advertised** Window Size.
  - **TCP may either transmit a bulk stream of data**, using sliding windows fully, or it may send slowly generated interactive data.

❖TCP Flow Control

- ➢It is possible for a TCP sender to send data faster than the receiver can process it.
  - ○When this happens, a TCP receiver may reduce the **advertised** Window Size value of an open connection
    - ▪To inform the sender to switch to a smaller window size.

❖Delayed ACKs

- ➢Simply mean that the ACK traffic volume is reduced.
- ➢Because ACKs are cumulative, one ACK from the receiver can in principle acknowledge multiple data packets from the sender.
- ➢Default number of delayed ACKs is 2.
- ➢The maximum ACK delay timeout is 500 *ms*.
  - ○Default is 200 *ms*.

6 TCP Timeout and Retransmission

❖When TCP sends a packet containing user data (this excludes ACK-only packets), it sets a timeout.

- ➢If that timeout expires before the packet data is acknowledged, it is retransmitted.
- ➢If the retransmission loss the sender doubles Timeout.
- ➢Retrying 5 times as the default.

QA

Lecturer: Nguyen Viet Ha, Ph.D.
Ho Chi Minh City University of Science
Faculty of Electronics and Communications
Department of Telecommunication and Networks
Email: nvha@fetel.hcmus.edu.vn