Session 2D

TCP Header: Control Bits

Reference: Ref1: TCP/IP Illustrated-Volume 1: Chapter 12: TCP

Mouli Sankaran

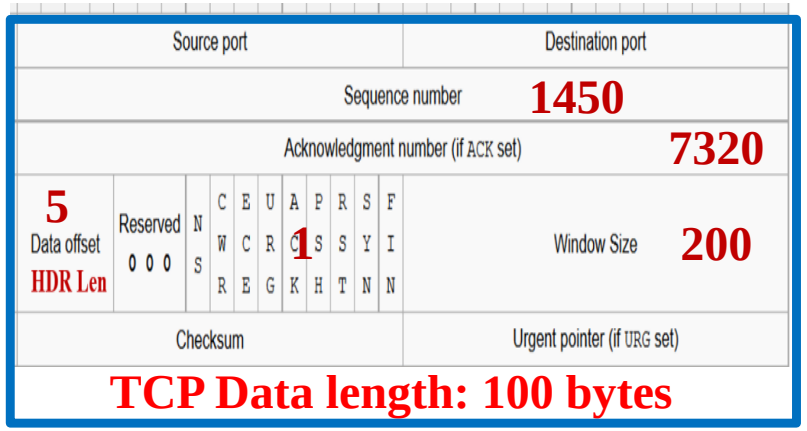
Session 2D: Focus

- Quiz 1 – TCP Header Fields
- TCH Header: Control bits or Flags
  - PSH: Push
  - URG: Urgent Flag and Urgent Pointer
  - Congestion Control (CWG and ECE flags)

Course page where the course materials will be posted as the course progresses:

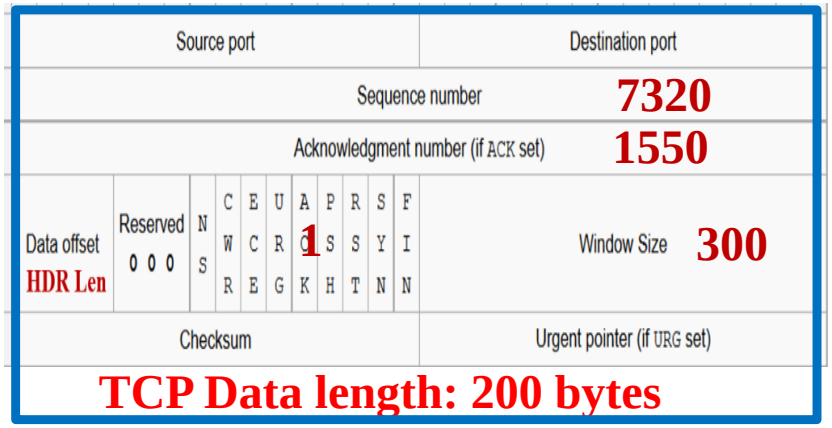
Quiz 1a: TCP Header Fields

If the host is generating the below TCP **Segment 2**, after receiving the **Segment 1** on the right, then answer the questions given.



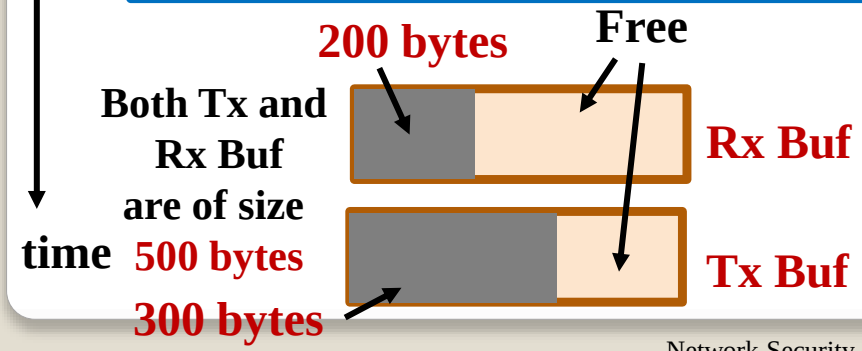
Segment 1

Segment 2



Q1: Fill in the Window size, ACK no. and Seq no. in the Segment 2

Note: Assume no other segment is in transit from this host to the other end as well as from the other end to this host, at this moment.



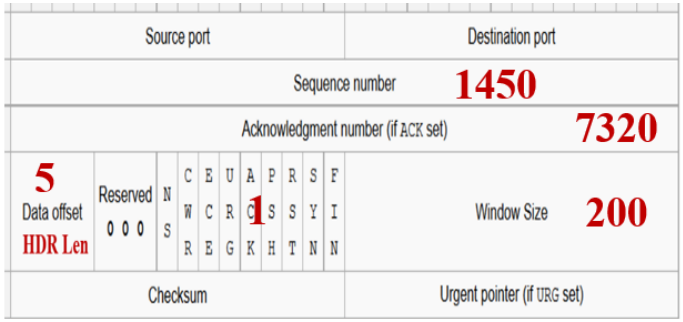
The **status** of **Rx buffer** after **receiving** the above **Segment 1** is shown here.

The **status** of **Tx buffer** after **completing** the above **Segment 2,** which is ready to be sent, is shown here.

Quiz 1b: TCP Header Fields contd. …

Q2: The Rx buf was empty before receiving the Segment 1. (Yes/No) **ANS2: NO. It had 100 bytes.**

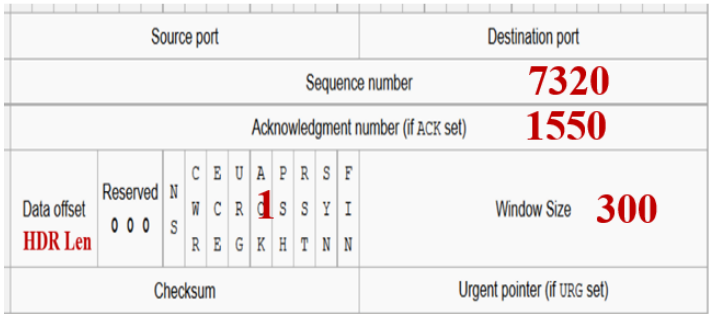
Q3: The Tx buf was full prior to preparing the Segment 2. (Yes/No) **ANS3: Yes.**



TCP Data length: 100 bytes

Segment 1

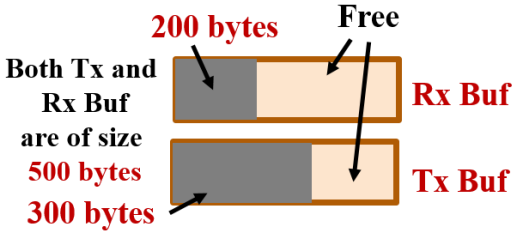
Segment 2



TCP Data length: 200 bytes

Q4: What can you say about the status of the Rx buf of the receiver after receiving this Segment 2. **ANS4: It will be full**

Note for Q4: Assume that the application receiving this data on the other end has not read any data from the moment Segment1 was sent out, till the Segment 2 is received.



The **status** of **Rx buffer** after **receiving** the above **Segment 1** is shown here.

Note: Remember that nothing can be said about the total size of the Tx and Rx bufs at the receiver, because it need not be same as what is on this machine.

The **status** of **Tx buffer** after **preparing** the above **Segment 2** is shown here.

time

TCP Segment Structure

Flags – Control Bits

TCP Header: With 9 Control Bits

Ref: From Wikipedia

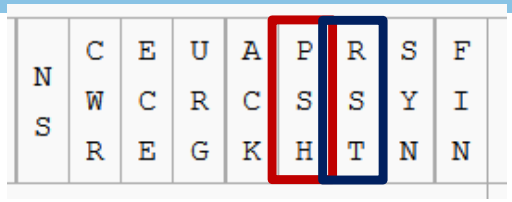
TCP Header

Offsets	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Source port																Destination port															
4	32	Sequence number																															
8	64	Acknowledgment number (if ACK set)																															
12	96	Data offset				Reserved 0 0 0				NS	CWR	ECE	URG	ACK	PSH	RST	SYN	FIN	Window Size														
16	128	Checksum																								Urgent pointer (if URG set)							
20 ...	160 ...	Options (if *data offset* > 5. Padded at the end with "0" bytes if necessary.) ...																															

- You can see a difference in the number of flags defined. There are 9 flags (control bits) including the NS bit. We will not study about **all** the control bits here. They are shown here for completeness.
- Currently **nine bit fields** are defined for the TCP header, although some older implementations understand only the last six of them.
- One or more of them can be turned on at the same time.

NS: This is not covered.

TCP Header Field: PSH (PUSH)

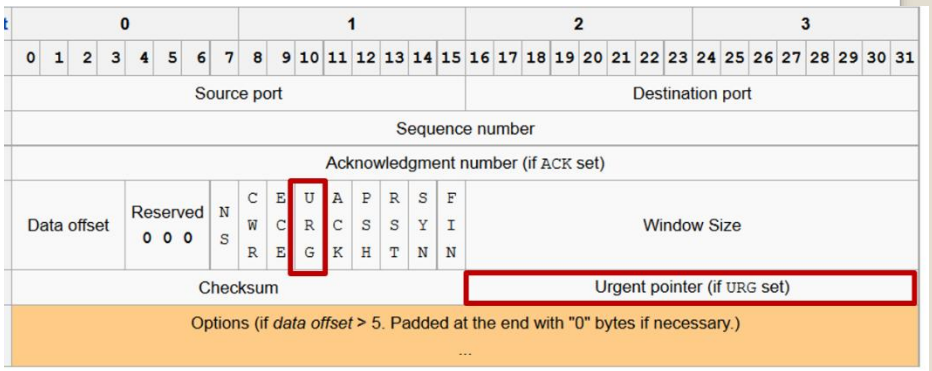


- **PSH: Push**
- It helps when the receiver receives a segment with PSH flag set, the receiving TCP stack should send the data immediately to the upper application layer, without waiting for the "Rx buf" to become full.
- The practical example of this is the **telnet** application where the application sends data in the form of few keystrokes or command lines.
- The telnet will become unusable if it waits for the **Rx buffer to become full** and then sends the data to the Telnet application.
- It should be sent as soon as one line of command has been typed by the user.
- User will be waiting for the response after typing a command.
- This flag has no influence on the protocol stack on the sender side.

RST: **Reset**: Resets the connection, typically used to abort a connection due to errors.

TCP Header Field: URG and Urgent Pointer

- **URG**: Urgent - The Urgent Pointer field is valid only if the URG bit field is set.
- This "pointer" is a positive offset that must be added to the Sequence Number field of the segment to yield the sequence number of the last byte of urgent data

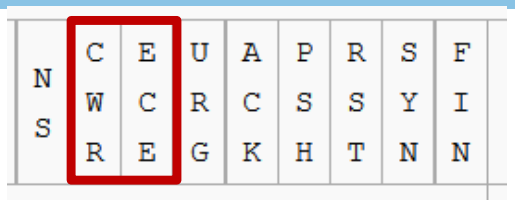


Example: When we press **Ctrl+C** during a telnet session URG flag would be set!!

Ref: Blog on URG and PSH

- The sender will not wait for the entire byte stream to be transmitted which is ahead of the urgent data. It is also called, **out-of-band data**.
- TCP's urgent mechanism is a way for the sender to provide specially marked data to the other end, out of sequence from the rest of the data.
- This also causes the receiving TCP to forward the urgent data on a separate channel to the application, ahead of already buffered data in the "Rx buffer".
- This allows the application to process the data out-of-band.

TCP Header Field: CWR and ECE



ECN: Explicit Congestion Notification (part of IP header).

CE: Congestion Experienced (Part of IP header)

Note: This feature needs to be supported by the network though which the TCP segment passes from the sender to receiver.

- **CWR**: Congestion Window Reduced. the sender reduced its sending rate
- **ECE**: Explicit Congestion Experienced. The sender received an Earlier Congestion Notification
- These two flags work together to enable the sender and the receiver of a TCP connection to be aware that there is a congestion in the network so that the senders can reduce their rates thus avoiding dropping of IP datagrams by the routers en-route.
- When a router detects congestion, rather than dropping packets destined to a receiver, it marks them with the CE flag in the IP header and delivers the packet to the receiver.
- Prior to acknowledging the receipt of the packet, the receiver sets the ECE flag in the TCP header of the ACK and sends it back to the sender.
- The sender having received the ECE marked ACK, responds by **halving the send window** and reducing the data rate thus reducing the congestion.
- During the connection setup this capability is exchanged.

ECN Workflow in Congestion Control

1. **Negotiation During Connection Setup**:

   During the TCP three-way handshake, both endpoints negotiate ECN support using specific bits in the TCP SYN packet:

   - **ECE Flag**: Indicates ECN capability.
   - **CWR Flag**: Reserved for congestion notifications.

2. **Marking by Routers**:

   If a router detects congestion (e.g., its queue is filling up), it:Marks the packet with the CE (11) codepoint in the IP header instead of dropping it.

3. **Receiver's Role**:

   When the receiver detects the CE mark in the packet, it sets the ECE flag in its next acknowledgment (ACK) to notify the sender about congestion.

4. **Sender's Role**:

   Upon receiving the acknowledgment with the ECE flag, the sender: Adjusts its congestion window (reduces the sending rate).

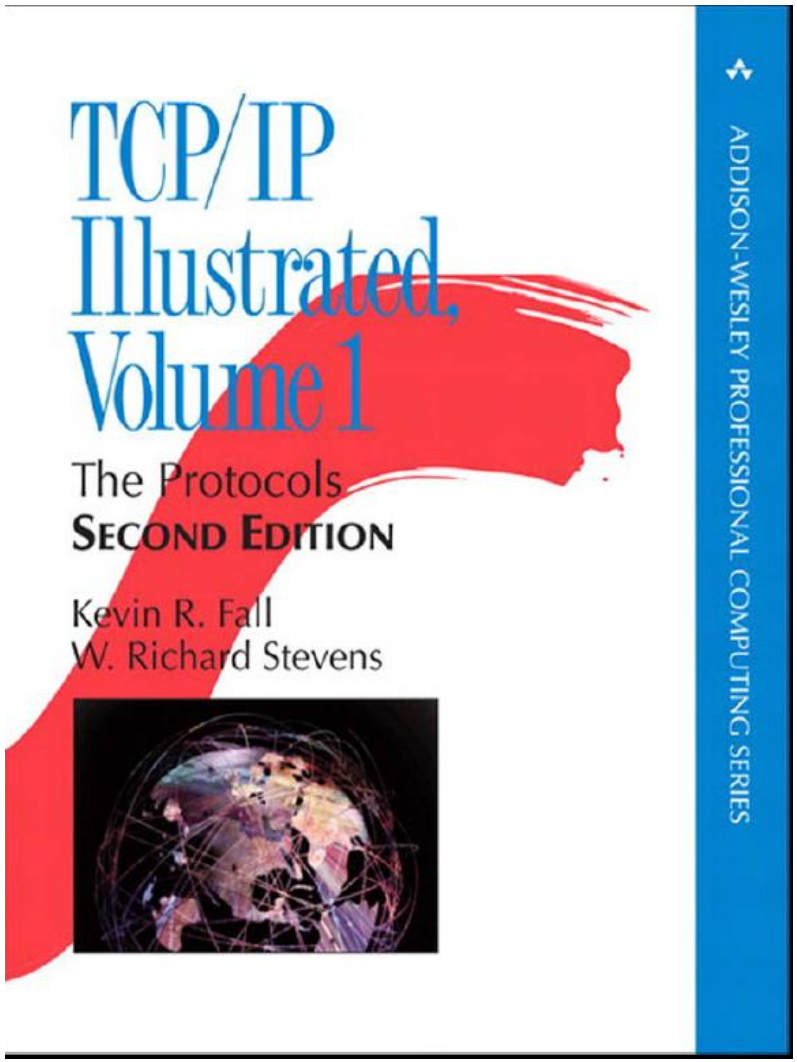
   Sends an acknowledgment with the CWR flag to inform the receiver that it has responded to the congestion.

Session 2D: Summary

- Quiz 1 – TCP Header Fields
- TCH Header: Control bits or Flags
  - PSH: Push
  - URG: Urgent Flag and Urgent Pointer
  - Congestion Control (CWG and ECE flags)

References

Ref 1



Ref 2

TCP Congestion Control: A Systems Approach

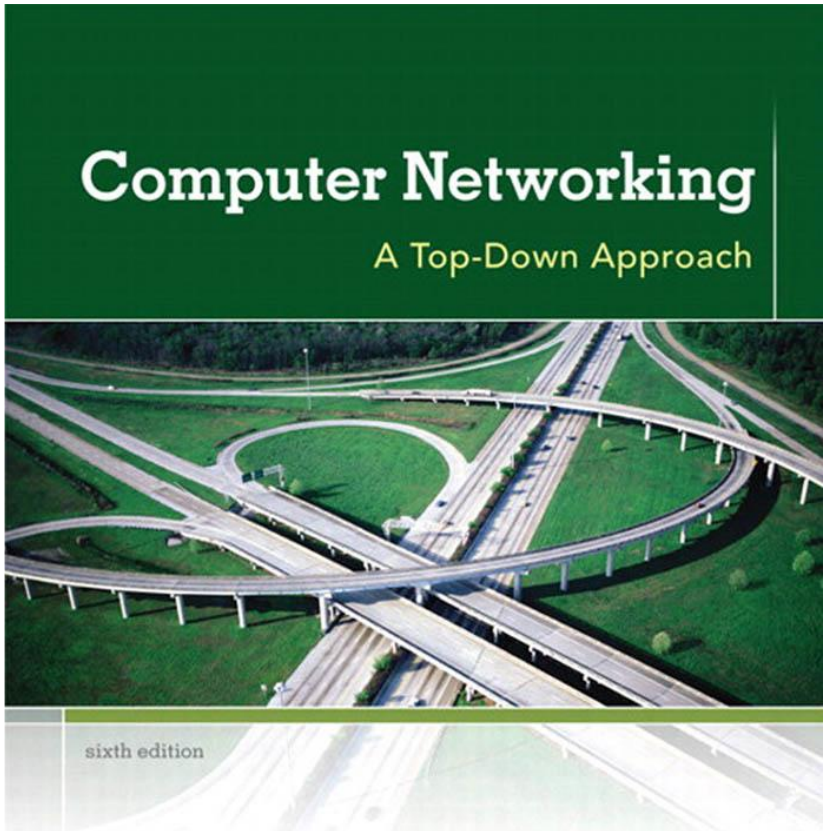
HTTPS://SYSTEMSAPPROACH.ORG

TCP Congestion Control: A Systems Approach

Peterson, Brakmo, and Davie

References

Ref 3



Ref 4