Communication Networks

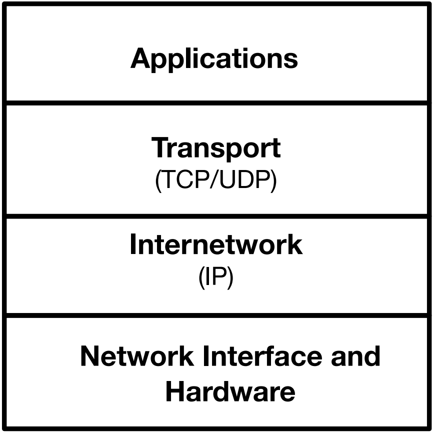
Dr Lina Barakat

School of Computer Science and Electronic Engineering

University of Essex

Email: lina.barakat@essex.ac.uk

TCP/IP Protocol Stack



Internetwork Layer

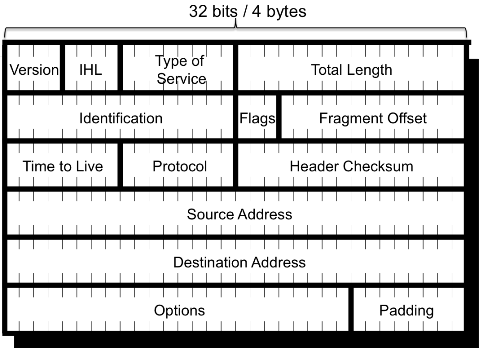
- Also called the internet layer or the network layer
- Provides a *virtual network* view (hides the underlying physical network)
- **Internet Protocol (IP)** is the most important internetwork-layer protocol
- Other internetwork-layer protocols:
  - Internet Control Message Protocol (ICMP)
  - Internet Group Management Protocol (IGMP)
  - Address Resolution Protocol (ARP)
  - Dynamic Host Configuration Protocol (DHCP)

Internet Protocol (IP)

- IP sends messages across the internet: provides a *routing function* that attempts to deliver messages to the right destination
- **IP datagram** is the basic unit of information transmitted in an IP network
  - messages may be divided into fragments, which are sent individually
  - IP adds a header to each fragment to form an IP datagram
- IP is an unreliable protocol: fragments might be lost, arrive out of order, or be duplicated
- Higher layer protocols add reliability to IP

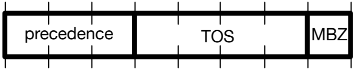
IP Datagram Header

- IP Datagram $=$ IP header $+$ data (from higher-layer protocols)
- An IP header has a minimum length of 20 bytes



IP Datagram Header

- **Version field.** The IP protocol version being used
  - 4 bits
  - e.g. IPv4 is 0100
  - e.g. IPv6 is 0110
- **Internet Header Length (IHL) field.** The length of this IP header in 32-bit words
  - 4 bits
  - e.g. IHL value 6 (0110) $\rightarrow$ header is $6 * 32 = 192$ bits
  - Minimum IHL value is 5 (0101)
- **Service Type field.** Specifies how the datagram should be handled in transmission (the quality of service requested)
  - 8 bits



e.g.

000 (routine)
001 (priority)
010 (Immediate)
...

0000 (normal)
1000 (min delay)
0100 (max throughput)
0010 (max reliability)
0001 (min cost)

IP Datagram Header

- **Total length field.** Total number of bytes/octets (multiples of 8 bits) that the IP datagram (header plus data) takes up
    - 16 bits
    - maximum length: 65,535 octets, minimum length: 20 octets
- **Identification field.** A unique number assigned to a datagram fragment to help reconstructing message from fragments
    - 16 bits
    - Every fragment of the same message has the same identification number
- **Flag fields.** Informational bits
    - 3 bits for 3 flags: Reserved, Don't Fragment (DF), and More Fragments (MF)
    - Reserved bit must be zero
    - DF indicates if fragmentation of this data is allowed (0), or not allowed (1)
    - MF indicates if this is the last fragment of the datagram (0), or there are still more fragments to follow (1)

IP Datagram Header

- **Fragment offset field.** Specifies where in the original message the fragment in this datagram starts
  - 13 bits
  - It is the position of this fragment from the start of the original message counted in 8 bytes units
- **Time to live field (8 bits).** How long this datagram is allowed to remain in the system before being deleted
- **Protocol field (8 bits).** The host-to-host transport layer protocol being used by the message sent over IP, e.g. TCP: 6
- **Header checksum field (16 bits).** Used to ensure integrity of the header in transmission

IP Datagram Header

- **Source field (32 bits)**. IP address of sending/source host, e.g. $137.73.9.232$
- **Destination field (32 bits)**. IP address of destination host (intended recipient)
- **Options field**. Optional arguments usable by IP processing software
- **Padding field**. 0s to make header up to multiple of 32 bits

Fragmentation

- Each network has a limit on the size of message it can carry: **the Maximum Transmission Unit (MTU)** of the network
- The sender fragments messages to fit the local MTU
- In IPv4, the fragments may be further fragmented on moving from one network to another with smaller MTU
- Fragments are reassembled at the destination

Fragmentation Algorithm

1. The DF flag bit is checked to see if fragmentation is allowed
    - If DF is set, but the message needs to be fragmented to move onto a network with a lower MTU, an error message is sent by ICMP
    - ICMP is part of IP and used to report errors and other info regarding IP processing back to the datagram's source
    - ICMP code 4: fragmentation needed and DF set
2. Based on the MTU value, the data field is split into two or more parts
    - every newly created data portion must be a multiple of 8 bytes (64 bits) except the last one
3. Each data portion is placed in an IP datagram.
4. Each of the fragmented datagrams is forwarded as a normal IP datagram
    - The fragments can traverse different routers to the intended destination

Fragmentation Algorithm

The **headers** of fragmented datagrams are minor modifications of the original datagram's header:

- The MF flag bit is set in all fragments except the last
- The fragment offset field in each is set to the location this data portion occupied in the original datagram, relative to the beginning of the original datagram
- If options were included in the original datagram, some of the options may be copied to each of the fragments depending on the Copied flag in each option field
- The header length field of the fragment is set
- The total length field of the fragment is set
- The header checksum field is re-calculated

Transport Layer

- Transmission Control Protocol (TCP)
- Key Features:
  - **Connection oriented**: implements mechanisms to setup and tear down a full duplex connection between end points
  - **Reliable**: implements mechanisms to guarantee error free and ordered delivery of information
  - **Flow and Congestion controlled**: implements mechanisms to control traffic

Ports and Sockets

- A process identifies itself to the TCP protocol by one or more **ports**
- A port is a 16-bit number used to identify to which application/process the message should be delivered
- Some ports are reserved for specific applications
  - e.g. FTP: 20/21, HTTP: 80, SMTP (e-mail): 25
- A **socket** is the combination of a host's IP address and a port number
  - E.g. $137.73.9.232 : 8080$
  - Every communication in TCP is between two sockets, i.e. two hosts using particular ports

TCP Connection Set-up

Connection is set up via a **handshake** that involves three steps:

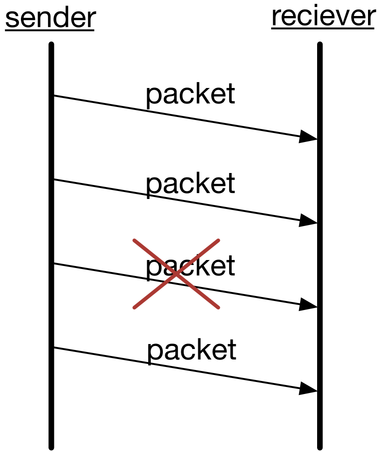
- Sender sends a synchronise message (SYN) to the receiver
- Receiver sends a message back acknowledging the synchronise (SYN ACK), and giving permission for communication to take place
- Sender sends a message acknowledging the acknowledgement (ACK)

TCP Connection tear-down

Connection is closed via a **handshake** that involves three steps:

- Sender sends a finalise message (FIN) to the receiver
- Receiver responds with an acknowledgement of the finalise (FIN + ACK)
- Finally, the sender responds with an acknowledgement of the acknowledgement (ACK)

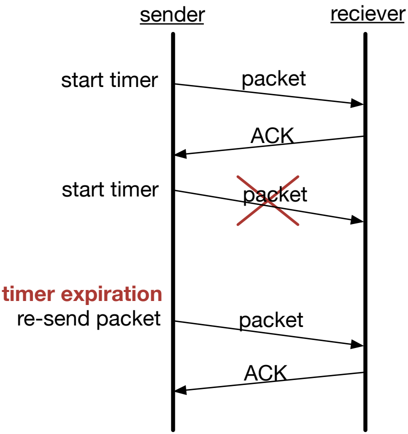
Problem with Unreliable Protocols: Lack of Feedback



TCP Concepts for Reliability and Flow Control

Basic Reliability: Positive Acknowledgment with Retransmission (PAR)

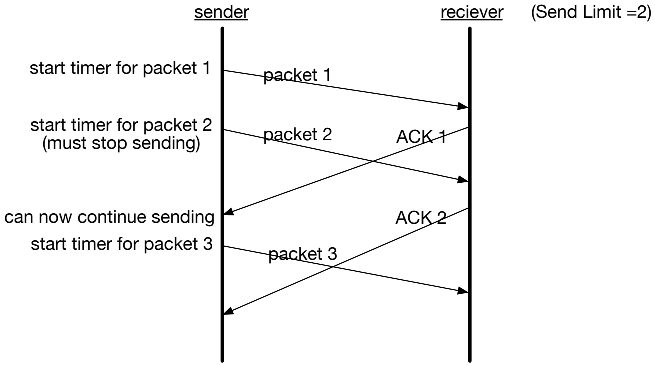
- Drawback: sender cannot send a second message until the first has been acknowledged



TCP Concepts for Reliability and Flow Control

Improved PAR: with Message Identification and Send Limits

- multiple messages can be sent without waiting for acknowledgment
- ensures better use of the network bandwidth (**efficiency**)
- *send limit* specifies maximum number of unacknowledged messages allowed from sender at one time (**basic flow control**)



TCP Concepts for Reliability and Flow Control

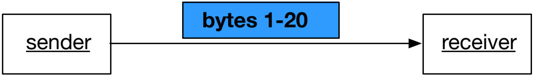
TCP's Sliding Window System: Similar Conceptually to Improved PAR, but with Some Adjustments

- TCP is a stream oriented protocol
- Every byte in a message sent from a sender to a reciever has a **sequence number**
  - Sender communicates its **initial sequence number (ISN)** in synchronisation
  - Sequence number for 1st byte of message is ISN+1, for 2nd byte is ISN+2, ...
- TCP divides continuous byte stream into **segments**
  - a TCP segment only carries the sequence number of the first byte in the segment

TCP Concepts for Reliability and Flow Control

TCP's Sliding Window System: Similar Conceptually to Improved PAR, but with Some Adjustments

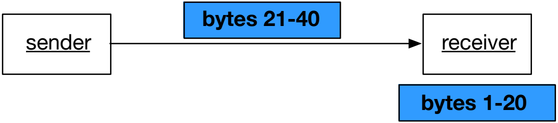
- **Reliability**. The receiver sends an **acknowledgement** back to the sender for every segment it receives
  - An acknowledgement states that the receiver has received all data in the message before a given sequence number
  - A segment is retransmitted on **time out**, or on acknowledgment **repetition**



TCP Concepts for Reliability and Flow Control

TCP's Sliding Window System: Similar Conceptually to Improved PAR, but with Some Adjustments

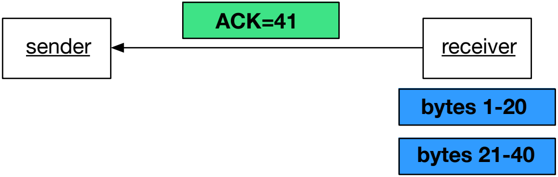
- **Reliability**. The receiver sends an **acknowledgement** back to the sender for every segment it receives
  - An acknowledgement states that the receiver has received all data in the message before a given sequence number
  - A segment is retransmitted on **time out**, or on acknowledgment **repetition**



TCP Concepts for Reliability and Flow Control

TCP's Sliding Window System: Similar Conceptually to Improved PAR, but with Some Adjustments

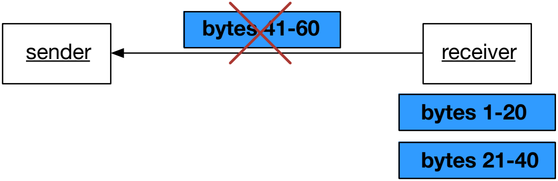
- **Reliability**. The receiver sends an **acknowledgement** back to the sender for every segment it receives
  - An acknowledgement states that the receiver has received all data in the message before a given sequence number
  - A segment is retransmitted on **time out**, or on acknowledgment **repetition**



TCP Concepts for Reliability and Flow Control

TCP's Sliding Window System: Similar Conceptually to Improved PAR, but with Some Adjustments

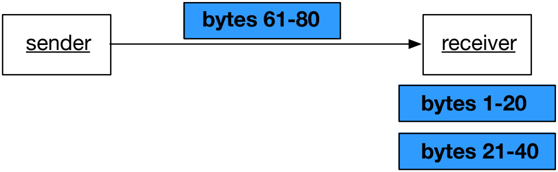
- **Reliability**. The receiver sends an **acknowledgement** back to the sender for every segment it receives
  - An acknowledgement states that the receiver has received all data in the message before a given sequence number
  - A segment is retransmitted on **time out**, or on acknowledgment **repetition**



TCP Concepts for Reliability and Flow Control

TCP's Sliding Window System: Similar Conceptually to Improved PAR, but with Some Adjustments

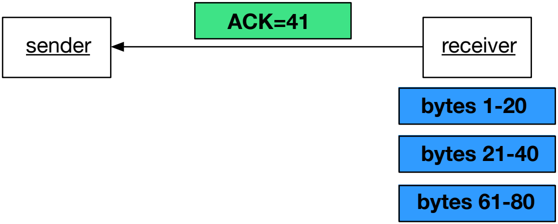
- **Reliability**. The receiver sends an **acknowledgement** back to the sender for every segment it receives
  - An acknowledgement states that the receiver has received all data in the message before a given sequence number
  - A segment is retransmitted on **time out**, or on acknowledgment **repetition**



TCP Concepts for Reliability and Flow Control

TCP's Sliding Window System: Similar Conceptually to Improved PAR, but with Some Adjustments

- **Reliability**. The receiver sends an **acknowledgement** back to the sender for every segment it receives
  - An acknowledgement states that the receiver has received all data in the message before a given sequence number
  - A segment is retransmitted on **time out**, or on acknowledgment **repetition**



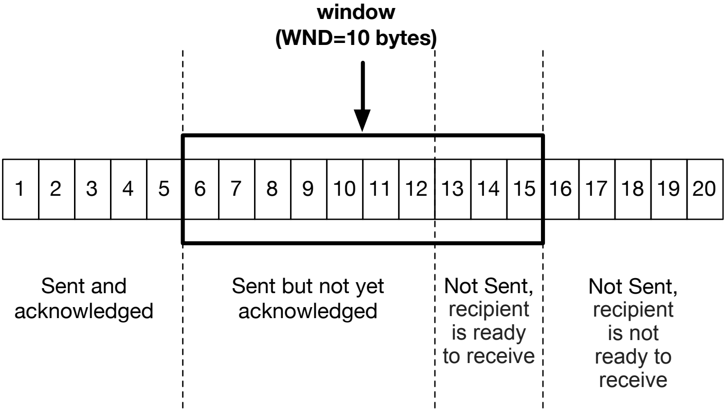
TCP Concepts for Reliability and Flow Control

TCP's Sliding Window System: Similar Conceptually to Improved PAR, but with Some Adjustments

- **Flow Control**. The amount of data that the receiver can receive is called the **window size (WND)**
  - It is the maximum number of unacknowledged bytes allowed from sender at one time
  - Determined initially by the receiver when the connection is established, but can vary during data transfer
  - Each ACK message will include the window size that the receiver is ready to deal with at that particular time
  - **Usable Window**: amount of the window that the sender is still allowed to send at any point in time

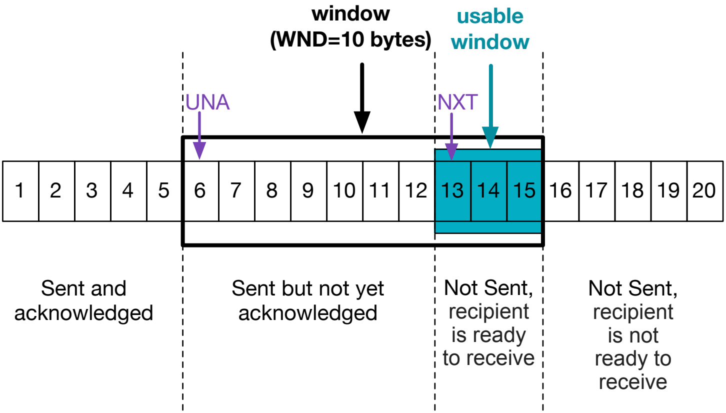
TCP Concepts for Reliability and Flow Control

TCP's Sliding Window System: Similar Conceptually to Improved PAR, but with Some Adjustments



TCP Concepts for Reliability and Flow Control

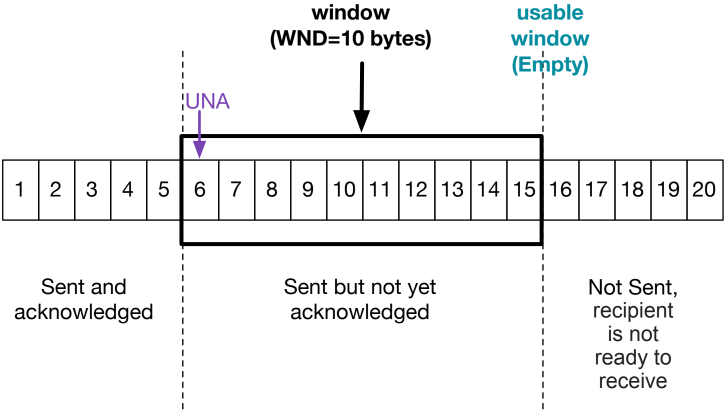
TCP's Sliding Window System: Similar Conceptually to Improved PAR, but with Some Adjustments



usable window size = UNA + WND -NXT

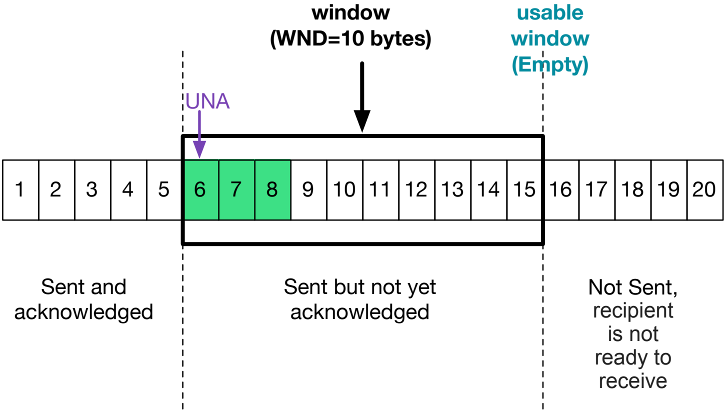
TCP's Sliding Window System: Example

Sending bytes in the usable window:



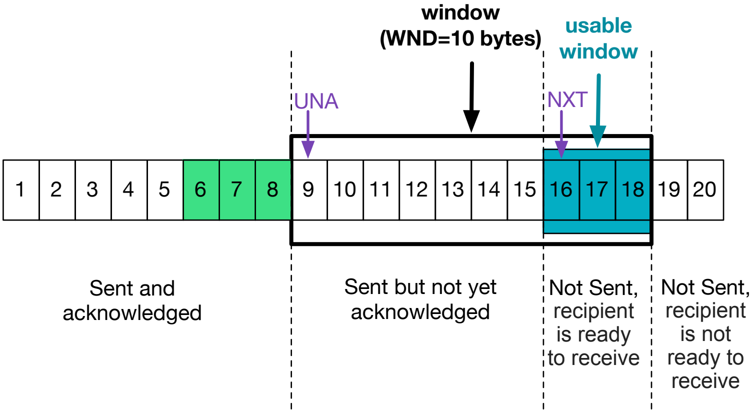
TCP's Sliding Window System: Example

Processing acknowledgments (ACK=9)

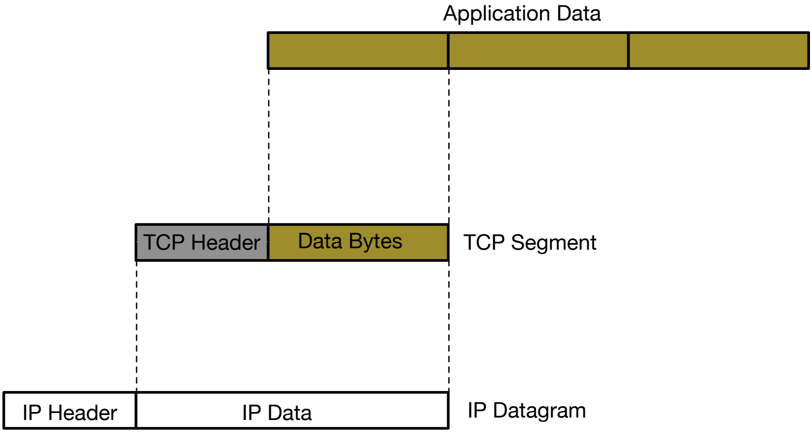


TCP's Sliding Window System: Example

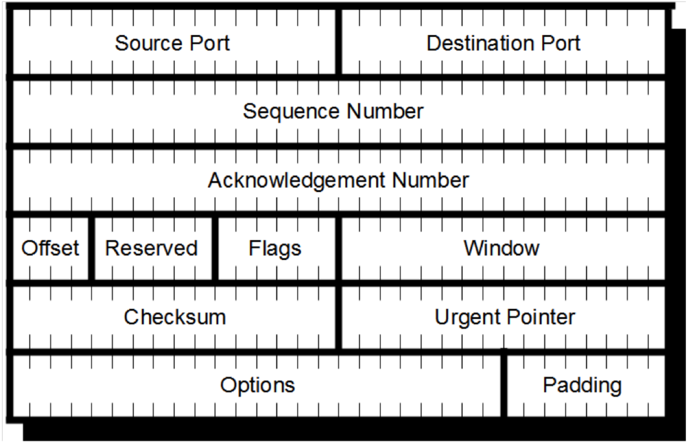
Processing acknowledgments (ACK=9)



TCP and IP


Application Data
TCP Header
Data Bytes
TCP Segment
IP Header
IP Data
IP Datagram

TCP Header



Source Port
Destination Port
Sequence Number
Acknowledgement Number
Offset
Reserved
Flags
Window
Checksum
Urgent Pointer
Options
Padding

TCP Header

- Source Port: the port of the segment sender
- Destination Port: the port of the receiver
- Sequence Number: the sequence number of the start of the segment, or the Initial Sequence Number (if a synchronise message)
- Acknowledgement Number: the sequence number before which the receiver has all the message data
- Data Offset: The length of the TCP header in 32-bit words
- Reserved: Not currently used (all 0s)

TCP Header

- URG Flag: Marks that this message contains urgent data
- ACK Flag: Marks that this is an acknowledgement
- PSH Flag: Marks that this data was pushed
- RST Flag: Marks that this is a reset message
- SYN Flag: Marks that this is a synchronise message or acknowledgement of a synchronise
- FIN Flag: Marks that this is a finalise message or acknowledgement of a finalise

TCP Header

- Window: The current acceptable window size (sent in acknowledgement messages)
- Checksum: A checksum over the segment, used to check for corruption
- Urgent Pointer: Position of where the urgent data ends inside the segment
- Options: Various TCP options (varies in length)

Useful Resources

- TCP (RFC 793)
- TCP and IP (RFC 879)
- http://www.tcpipguide.com