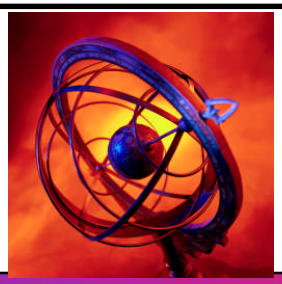




TCP and UDP – Transport Layer

Professor Richard Harris

School of Engineering and Advanced Technology (SEAT)

Objectives

- You will be able to identify the fields for the UDP header and how they are obtained.
- You will be able to identify the fields for the TCP header and show how they are obtained.

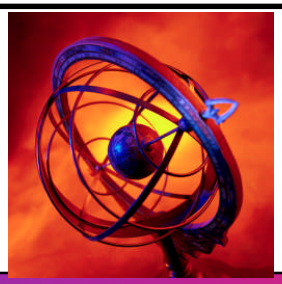
References

- Computer Networks by Andrew S. Tanenbaum
  - Chapter 6 of 4th Edition
- Data Communications and Networking by Behrouz A. Forouzan
  - Chapter 23 of 4th Edition
- Data and Computer Communications by William Stallings,
  - Prentice Hall, 6th Edition
- Telecommunications Protocols Travis Russell
  - McGraw Hill

Slides and slide extracts from Forouzan's book

Presentation Outline

- Overview of UDP
- Overview of TCP


The Internet Transport Protocols: UDP

What is UDP?

The UDP Header format

Applications of UDP

What is UDP?

- **User datagram protocol or (UDP) is the internetworking protocol that is part of the TCP/IP suite. It resides within Layer 4 (Transport Layer) of the Open Systems Interconnection (OSI) model. UDP is defined in RFC 768.**
- **It provides a connectionless service for application-level procedures.**
- **Since it is connectionless, UDP is basically an unreliable service, and hence delivery and duplicate protection are not guaranteed.**
  - **However, this does reduce the overhead of the protocol and adequate in many cases.**

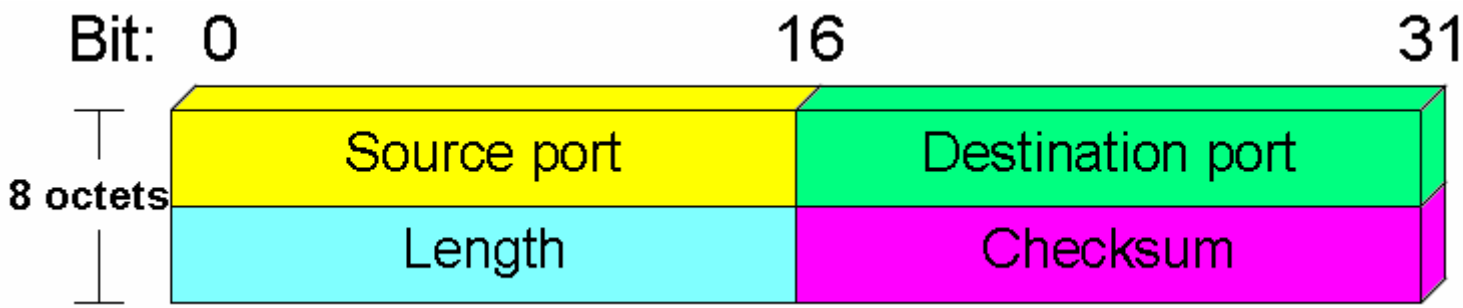
UDP

- **Listed below are examples when the use of a connectionless service is justified.**
  - **Inward data collection: involves the periodic active or passive sampling of data sources, such as sensors, and automatic self-test reports from security equipment or network devices. In a real-time monitoring situation, the loss of an occasional data unit would not cause distress, because the next report should arrive shortly.**
  - **Outward data dissemination: includes broadcast messages to network users, the announcement of a new node or the change of address of a service, and the distribution of real-time clock values.**

UDP

- **Request-response**: applications in which a transaction service is provided by a common server to a number of distributed TS users, and for which a single request-response sequence is typical. Use of the service is regulated at the application level, and lower-level connections are often unnecessary and cumbersome.
- **Real-time applications**: such as voice and telemetry, involving a degree of redundancy and/or real-time transmission requirement. These must not have connection-oriented functions such as retransmission.

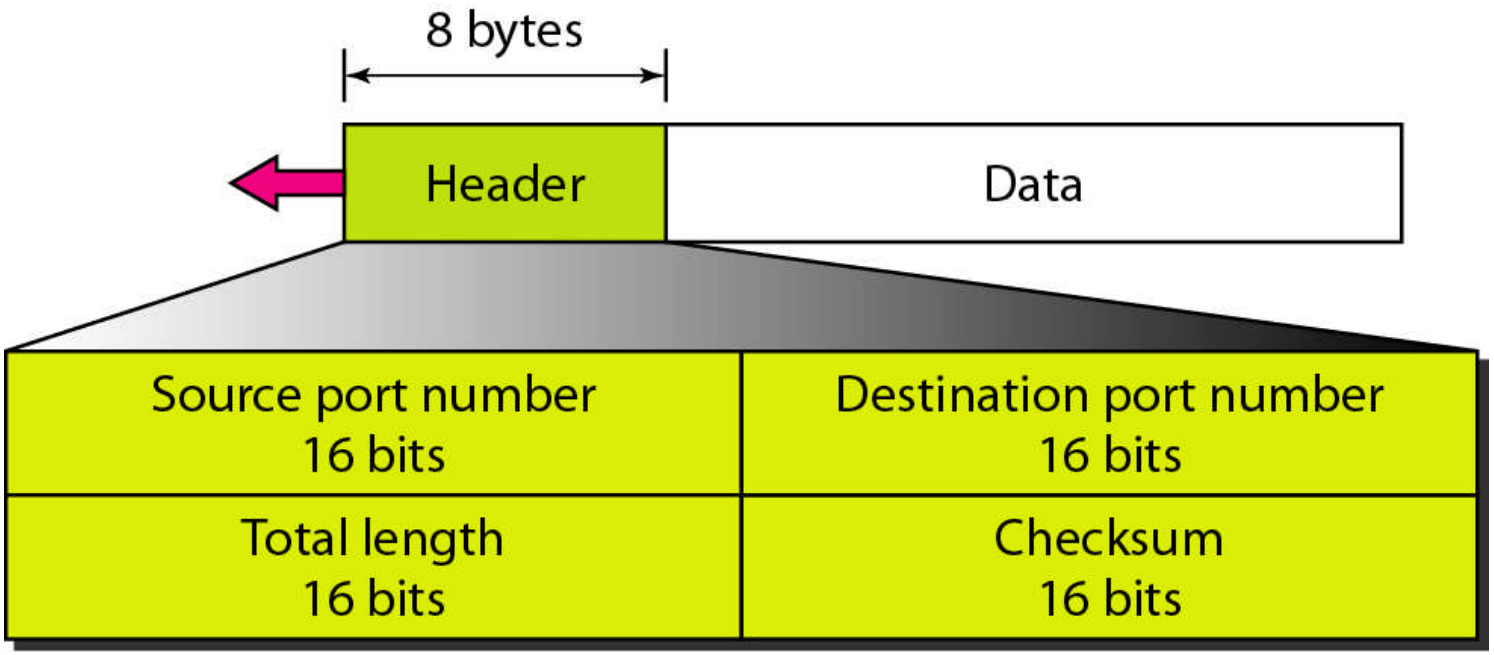
The UDP Header Format



- **Source port:** UDP port of sending host. The sending port value is optional. If not used, it is set to zero.
- **Destination port:** UDP port of destination host. This provides an endpoint for communications.
- **Length:** the size of the UDP message. The minimum UDP packet contains only the header information (8 bytes).
- **Checksum:** verifies that the header is not corrupted. The checksum value is optional. If not used, it is set to zero. It is the same algorithm used for TCP and IP. If an error is detected, the entire UDP segment is discarded and no further action is taken.

The UDP Header Format

- **Because it is connectionless, the role of UDP is to essentially add a port addressing capability to IP.**



UDP (User Datagram Protocol)

- Connectionless service for application level procedures
  - Unreliable
  - Delivery and duplication control not guaranteed
- Reduced overhead
  - e.g. network management and real time communication

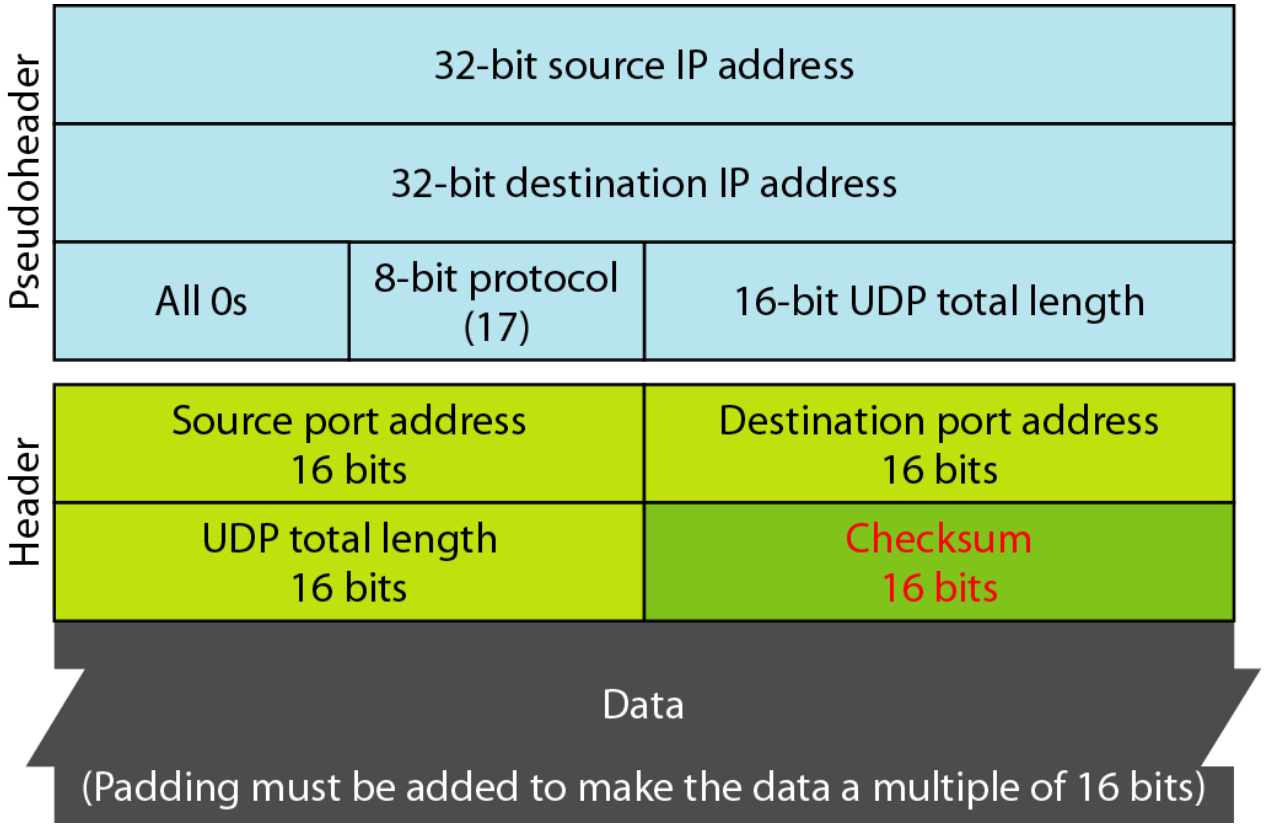
Applications of UDP

- Client-server situation
  - The client sends a short request to the server and expects a short reply back. If either the request or reply is lost, the client can just time out and try again
  - DNS (Domain Name System). A program that needs to look up the IP address of some host name, for example, www.massey.ac.nz, can send a UDP packet containing the host name to a DNS server. The server replies with a UDP packet containing the host's IP address.
  - No setup is needed in advance and no release is needed afterward. Just two messages go over the network
- Real time transmission

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)

Pseudo-header for Checksum



Checksum Calculation

153.18.8.105		
171.2.14.10		
All 0s	17	15

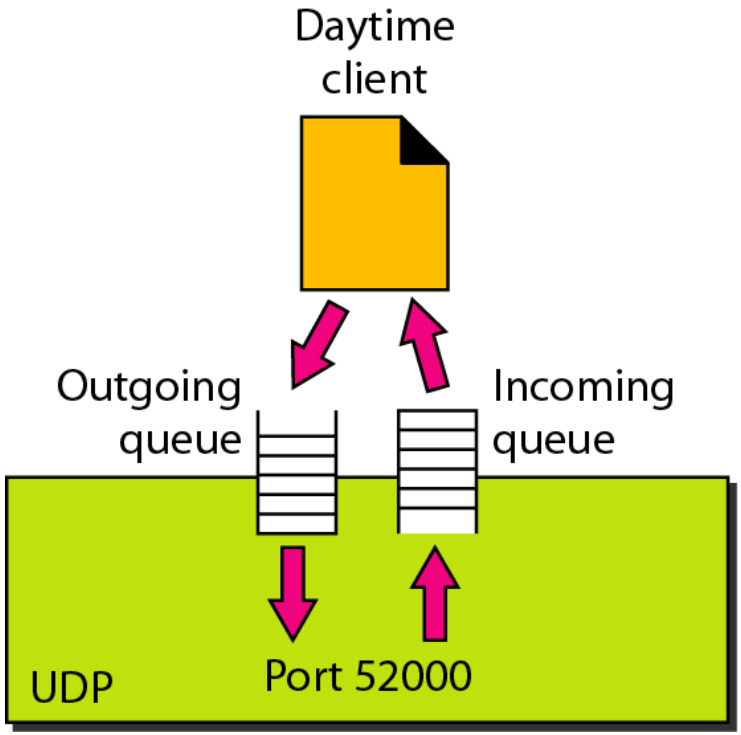
1087	13
15	All 0s

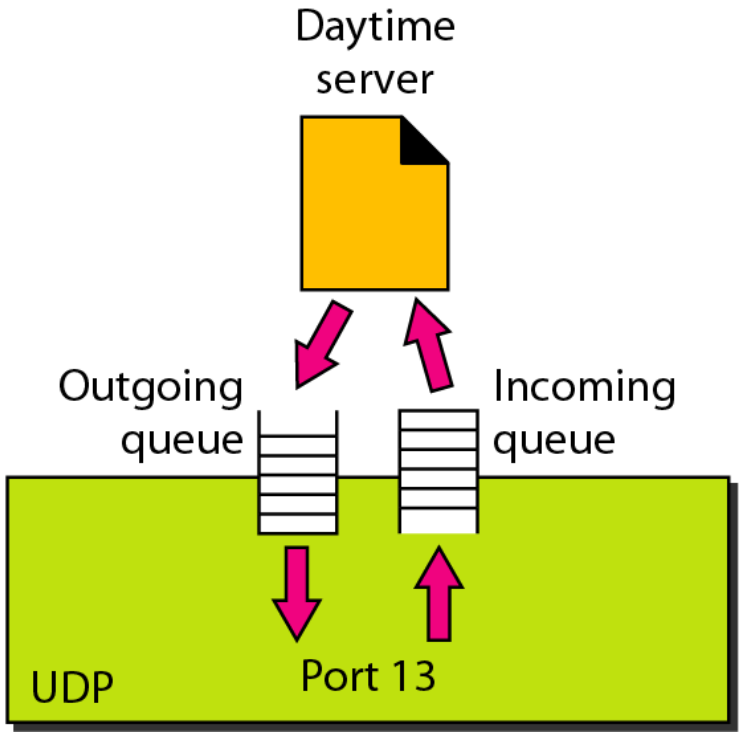
T	E	S	T
I	N	G	All 0s

10011001 00010010	153.18
00001000 01101001	8.105
10101011 00000010	171.2
00001110 00001010	14.10
00000000 00010001	0 and 17
00000000 00001111	15
00000100 00111111	1087
00000000 00001101	13
00000000 00001111	15
00000000 00000000	0 (checksum)
01010100 01000101	T and E
01010011 01010100	S and T
01001001 01001110	I and N
01000111 00000000	G and 0 (padding)
10010110 11101011	Sum
01101001 00010100	Checksum

The above figure shows the checksum calculation for a very small user datagram with only 7 bytes of data. Since the number of bytes of data is odd, padding is added for checksum calculation. The pseudo-header as well as the padding will be dropped when the user datagram is delivered to IP.

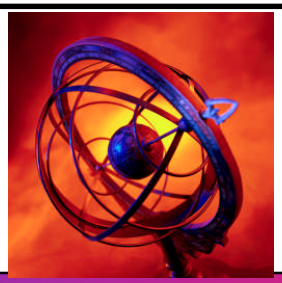
Queues in UDP





Some Tutorial Questions

- Consider the following Hex dump of the UDP header:
  - 06 32 00 0D 00 1C E2 17
- Determine
  1. Source port number
  2. Destination port number
  3. Total length of UDP header
  4. Is the packet directed from client to server or vice versa?
  5. What is the client process?
- What is the largest possible UDP datagram?
- What is the smallest possible UDP datagram?


The Internet Transport Protocols: TCP

The TCP Service Model

The TCP Segment Header

TCP Connection Establishment

TCP Connection Release

TCP Connection Management Modeling

TCP Transmission Policy

TCP Congestion Control

TCP Timer Management

What is TCP? - 1

- TCP is the internetworking protocol that is part of the TCP/IP suite. It resides within Layer 4 (Transport Layer) of the Open Systems Interconnection (OSI) model. TCP is defined in RFC 793.
- It provides an end-to-end transport of data units using connection-oriented services across multiple packet switching networks.
- Because it is connection-oriented, TCP provides reliable data transfer through the use of credit-based flow and error control techniques. This technique is somewhat different from the sliding-window flow control found in X.25 and HDLC.

What is TCP? - 2

- In essence, TCP separates acknowledgments from the management of the size of the sliding window.
- Although this credit-based mechanism is used for end-to-end flow control, it is also used to assist in inter-network congestion control.
- This is accomplished by reducing the data flow of data onto the Internet until congestion eases.

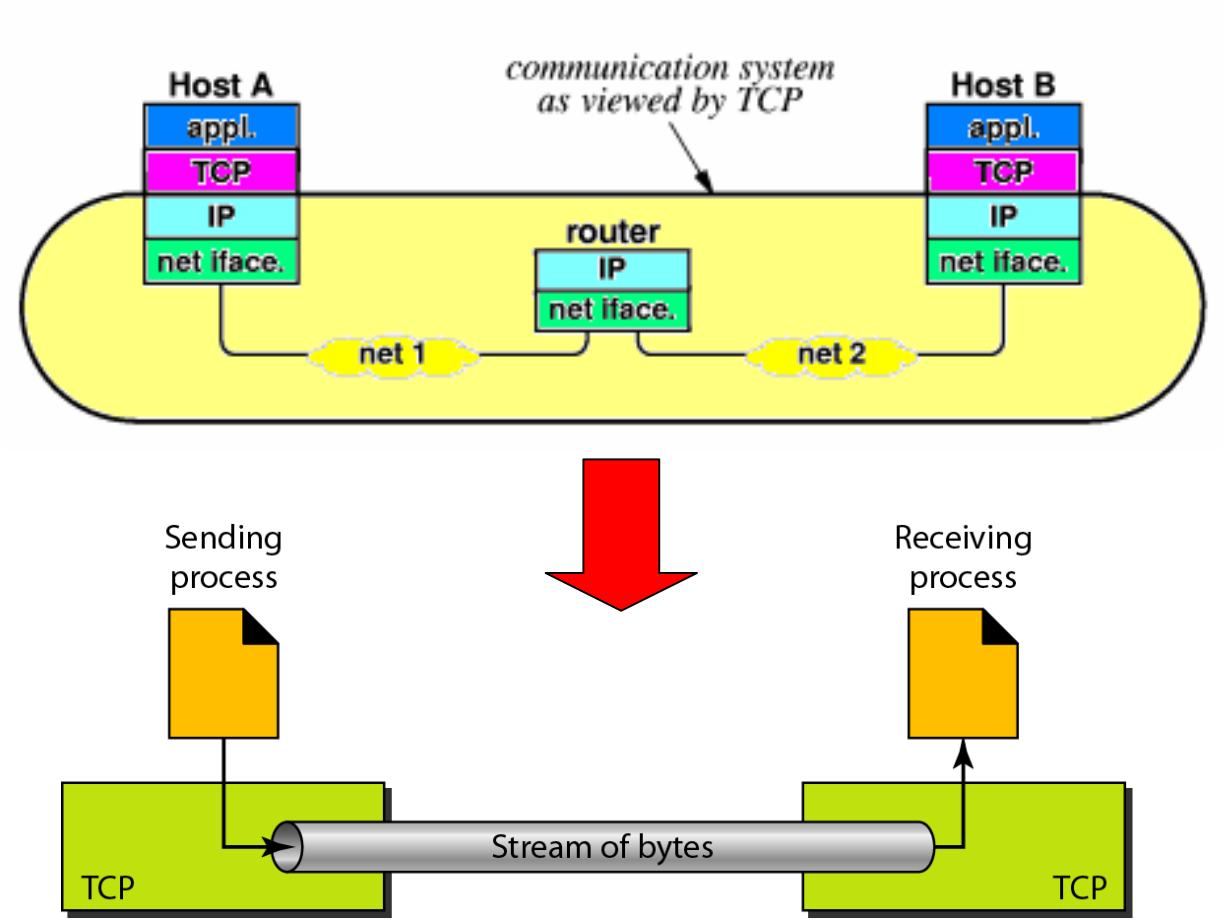
TCP Components

- Like any protocol standard, TCP is specified in two parts:
  - TCP services
  - The protocol format and mechanisms

TCP Services

- TCP provides two facilities for labelling data: push and urgent.
  - Data stream push: Normally, TCP decides when sufficient data have accumulated to form a segment for transmission.
    - However, the TCP user can require that TCP transmits all outstanding data up to and including that labelled with a push flag.
    - On the receiving end, TCP will deliver this data to the user in the same manner.
  - Urgent data signalling: This provides a means of informing the destination TCP user that significant or "urgent" data is in the upcoming data stream.
    - However, it is up to the receiving end to determine appropriate action.

The TCP Service Model



The TCP Service Model

- **Connection oriented**
  - Reliable connection startup, duplicate packets used in previous connections will not appear to be valid responses or otherwise interfere with the new connection
  - Graceful connection shutdown, TCP guarantees to deliver all the data reliably before closing connection
- **Point-to-point communication**, each TCP connection has exactly two endpoints and the TCP service is obtained through a socket, each socket has a socket number consisting of the IP address and a 16-bit number local to the host known as port
- With complete reliability, TCP guarantees that the data sent across a connection will be delivered exactly as sent.
- Full duplex communication, allows data to flow in either direction at any time.

TCP Service Primitives and Parameters

- Similar to IP, TCP services are expressed in terms of:
  - Primitives: the function that is to be performed
  - Parameters: used to pass data and control information
- The TCP primitives and parameters are more complex because of the richer set of services provided by TCP.
- TCP provides two primitives at the interface:
  - TCP service request primitive: issued by a TCP user to TCP
  - TCP service response primitive: issued by TCP to a local TCP user

TCP Service Request Primitives

- **TCP Service Request {**
  - **Unspecified Passive Open**
  - **Fully Specified Passive Open**
  - **Active Open**
  - **Active Open with Data**
  - **Send**
  - **Allocate**
  - **Close**
  - **Abort**
  - **Status**
  - **}**

TCP Service Request Primitives

- **Unspecified Passive Open:** listen for connection attempt at specified security and precedence from any remote destination.
- **Fully Specified Passive Open:** listen for connection attempt at specified security and precedence from specified destination.
- **Active Open:** request connection at a particular security and precedence to a specified destination.
- **Active Open with Data:** request connection at a particular security and precedence to a specified destination and transmit data with the request.

TCP Service Request Primitives

- **Send**: transfer data across named connection.
- **Allocate**: issue incremental allocation for receive data to TCP.
- **Close**: Close connection gracefully.
- **Abort**: Close connection abruptly.
- **Status**: Query connection status.

TCP Service Response Primitives

- **TCP Service Response {**
  - **Open ID**
  - **Open Failure**
  - **Open Success**
  - **Deliver**
  - **Closing**
  - **Terminate**
  - **Status Response**
  - **Error**
  - **}**

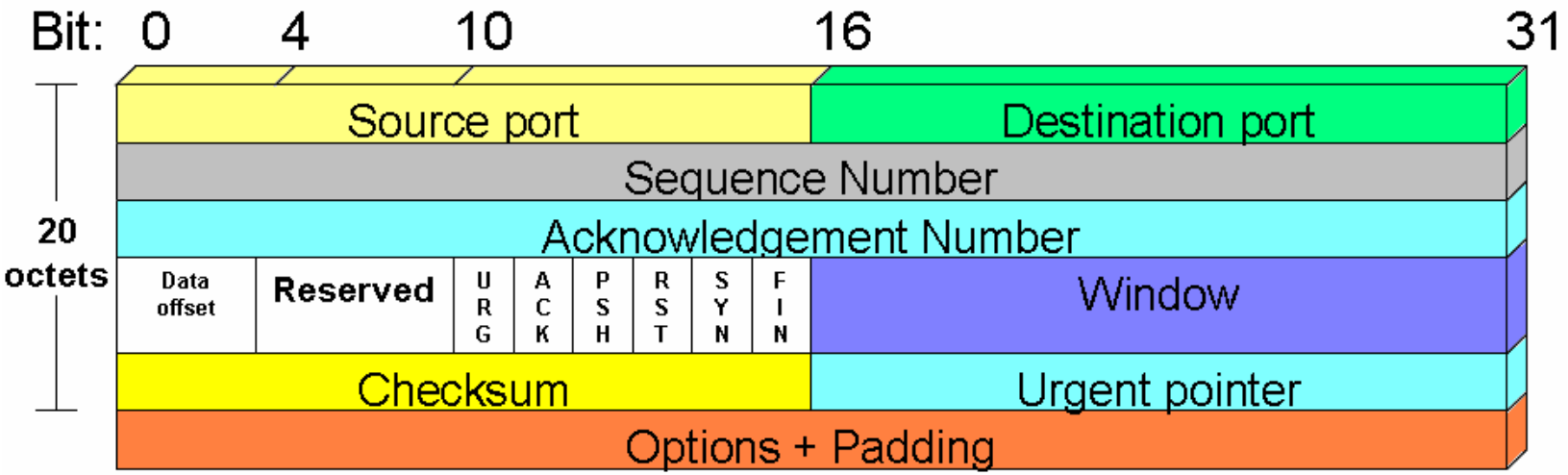
TCP Service Response Primitives

- **Open ID**: informs TCP user of connection name assigned to pending connection requested in an Open primitive.
- **Open Failure**: reports failure of an Active Open request.
- **Open Success**: reports completion of pending Open request.
- **Deliver**: reports arrival of data.
- **Closing**: reports that remote TCP user has issued a Close and that all data sent by remote user have been delivered.

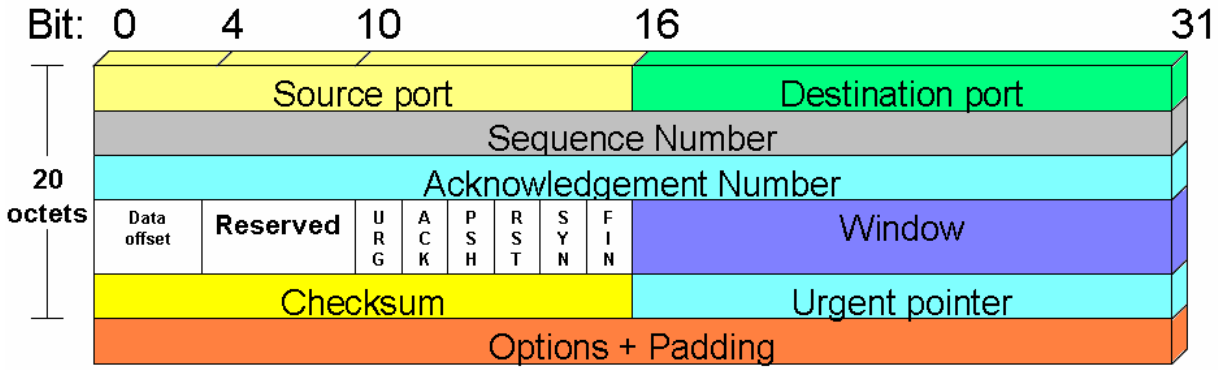
TCP Service Response Primitives

- **Terminate**: reports that the connection has been terminated; a description of the reason for termination is provided.
- **Status Response**: reports current status of connection.
- **Error**: reports service-request or internal error.

The TCP Header Format



The TCP Header Format - 1



- **Source port (16 bits)**: source TCP user.
- **Destination port (16 bits)**: destination TCP user.
- **Sequence number (32 bits)**: sequence number of the first data octet in this segment except when the SYN flag is set. If SYN is set, it is the initial sequence number (ISN) and the first data octet is ISN + 1.
- **Acknowledgement number (32 bits)**: a piggybacked acknowledgment. Contains sequence number of the next data octet that the TCP entity expects to receive.
- **Data offset (4 bits)**: number of 32-bit words in the header.

The TCP Header Format - 2

- **Reserved (6 bits)**: reserved for future use.
- **Flags (6 bits)**:
  - **URG**: urgent pointer field significant.
  - **ACK**: acknowledgment field significant.
  - **PSH**: push function.
  - **RST**: reset the connection.
  - **SYN**: synchronize the sequence numbers.
  - **FIN**: no more data from sender.
- **Window (16 bits)**: flow control credit allocation, in octets. Contains the number of data octets beginning with the one indicated in the acknowledgment field that the sender is willing to accept.

The TCP Header Format - 3

- **Checksum (16 bits)**: the ones complement of the sum modulo $2^{16} - 1$ of all the 16-bit words in the segment plus a pseudo-header.
- **Urgent Pointer (16 bits)**: points to the last octet in a sequence of urgent data. This allows the receiver to know how much urgent data are coming.
- **Options (Variable)**: encodes the options requested by the sending user. An example is the option that specifies the maximum segment size that will be accepted.

TCP Mechanisms

- TCP mechanisms are grouped into these three categories:
  - Connection Establishment
  - Data Transfer
  - Connection Termination

TCP Mechanisms

- Connection establishment in a TCP session is initialised through a three-way handshake.
- The following lists the steps for a three-way handshake:
  - The initiating host requests a session by sending out
    - a segment with the synchronization (SYN) flag set to ON
    - SN = X, where X is the initial sequence number (SN).
  - The receiving host acknowledges the request by sending back
    - a segment with both the SYN and acknowledgment (ACK) flags set to ON
    - SN = Y
    - AN = X+1, where AN is the acknowledgment number (AN)

TCP Mechanisms

- Note: The acknowledgment indicates that the receiving host is now expecting to receive a segment from the initiating host beginning with data octet X+1, acknowledging the SYN, which occupied SN = X.

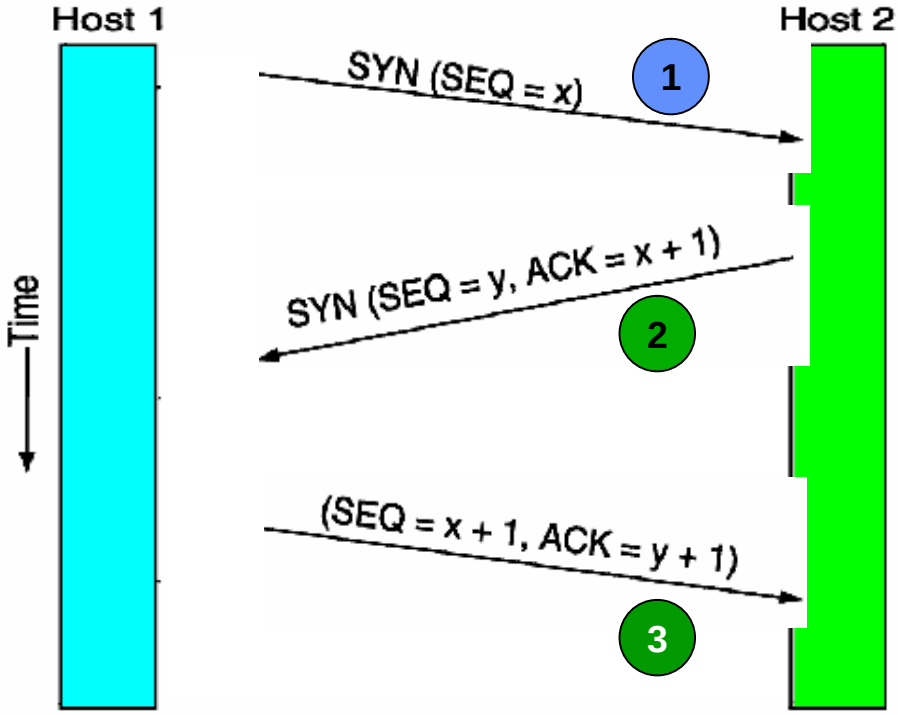
- Finally, the initiating host responds with
  - a segment with both the SYN and ACK flags set to ON
  - SN = X+1
  - AN = Y+1

- Data transfer is viewed logically as a stream of octets, and normally TCP decides when sufficient data have accumulated to form a segment for transmission. However, the data labels PUSH and URGENT can alter this behaviour.

TCP Connection Establishment

TCP connection establishment in the normal case

- Three way handshake
  - Between pairs of ports
  - One port can connect to multiple destinations



TCP Mechanisms

- A graceful close is expected when terminating a connection.
- This is accomplished by sending a CLOSE primitive.
- The transport entity then sets the FIN bit on the last segment it sends out, which also contains the last of the data to be sent on this connection.

Some Common Port Numbers

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
20	FTP, Data	File Transfer Protocol (data connection)
21	FTP, Control	File Transfer Protocol (control connection)
23	TELNET	Terminal Network
25	SMTP	Simple Mail Transfer Protocol
53	DNS	Domain Name Server
67	BOOTP	Bootstrap Protocol
79	Finger	Finger
80	HTTP	Hypertext Transfer Protocol
111	RPC	Remote Procedure Call