3. Process to Process TCP and UDP

Rekayasa Internet

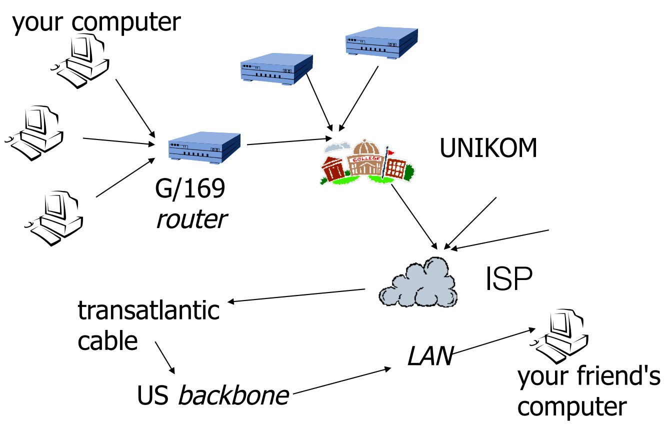
Teknik Komputer

Susmini I. Lestariningati, M.T

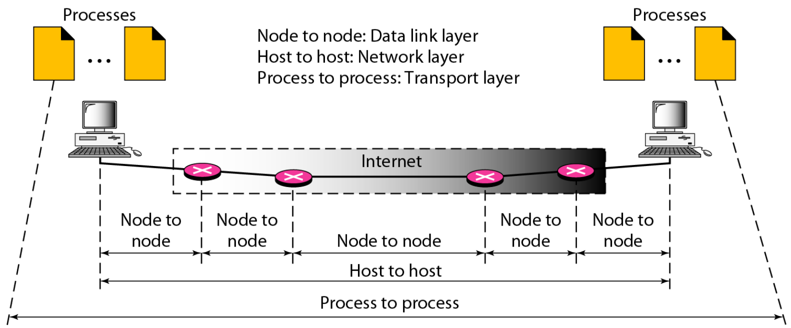
Reminder from last lecture

- IP sends data from place to place. TCP or UDP sit above it at either end.
- When you use the internet you use addresses like http://facebook.com or lestariningati@yahoo.com
- These addresses must then be converted to an IP address e.g. 144.32.100.24
- This means that data (packets) can get from A to B.
- But what happens if data is lost, how do we know where they are going to and how can we put packets back together into data?

Emailing a friend



Types of Data Deliveries



Routing - Network Layer

- How do packets know where to go?
- This problem is known as routing.
- The oldest (and easiest) solution is static routing.
- Each computer has a table saying where to go to get to each other computer.
- On a Local Area Network (LAN) list all machines on your subnet and the address of the external router for everything else.
- Most machines only need to know how to get to their nearest router.

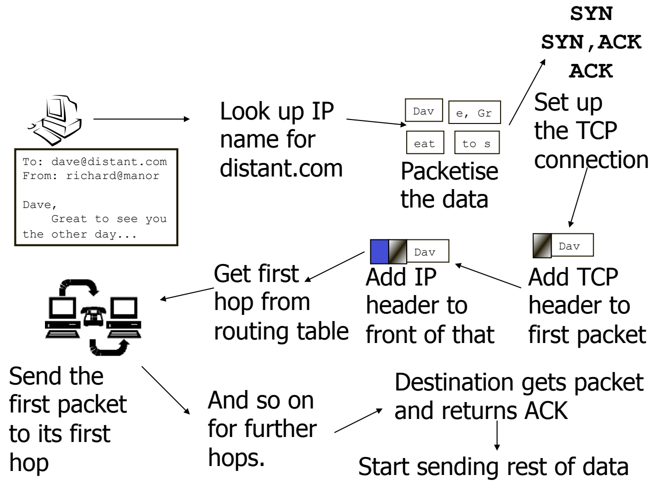
Introduction - Transport Layer

- On a single device, people can use multiple services such as e-mail, the web, and instant messaging to send messages or retrieve information.
- Applications such as e-mail clients, web browsers, and instant messaging clients allow people to use computers and networks to send messages and find information.
- Data from each of these applications is packaged, transported, and delivered to the appropriate server daemon or application on the destination device.
- The processes described in the OSI Transport layer accept data from the Application layer and prepare it for addressing at the Network layer. The Transport layer is responsible for the overall end-to-end transfer of application data. The role of the Transport layer is encapsulating application data for use by the Network layer.

TCP and UDP - Transport Layer

- Once we've got our IP packet safely to its destination what happens next?
- Having stripped off the header, the first thing we find is another header.
- The second header provides information on which port to enter the machine on and where to send the reply.
- It also provides a checksum to check the data is valid.
- There's TCP and UDP on Transport Layer

Journey of an email



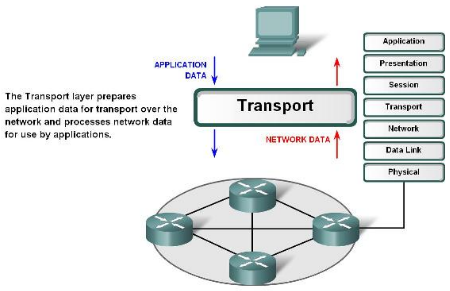
Transport Layer

The Transport layer also encompasses these functions:

a) Enables multiple applications to communicate over the network at the same time on a single device

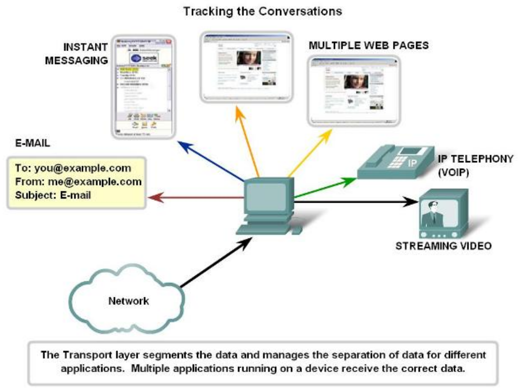
b) Ensures that, if required, all the data is received reliably and in order by the correct application

c) Employs error handling mechanisms



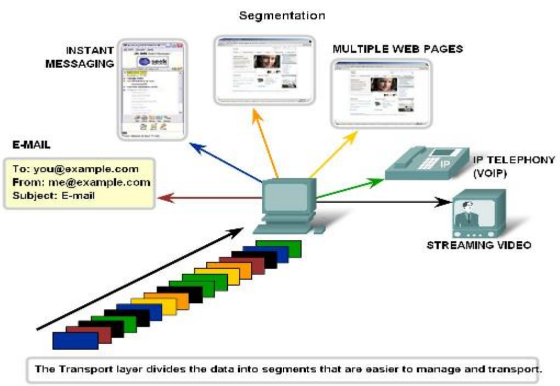
Purpose of Transport Layer

1. **Tracking the individual communication between applications on the source and destination hosts:**



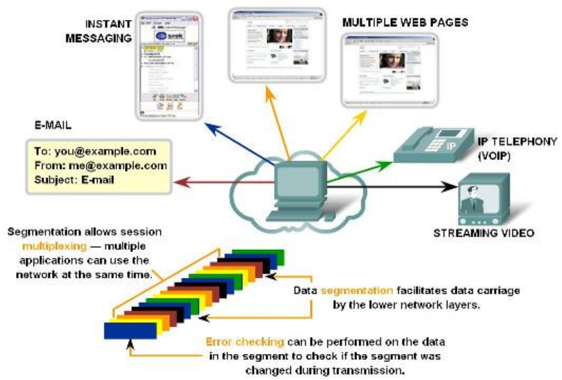
- Any host may have multiple applications that are communicating across the network. Each of these applications will be communicating with one or more applications on remote hosts. It is the responsibility of the Transport layer to maintain the multiple communication streams between these applications.

2. **Segmenting data and managing each piece:**



- As each application creates a stream data to be sent to a remote application, this data must be prepared to be sent across the media in manageable pieces. The Transport layer protocols describe services that segment this data from the Application layer. This includes the encapsulation required on each piece of data. Each piece of application data requires headers to be added at the Transport layer to indicate to which communication it is associated.

3. **Reassembling the segments into streams of application data:**



- At the receiving host, each piece of data may be directed to the appropriate application. Additionally, these individual pieces of data must also be reconstructed into a complete data stream that is useful to the Application layer. The protocols at the Transport layer describe the how the Transport layer header information is used to reassemble the data pieces into streams to be passed to the Application layer.

4. **Identifying the different applications:**

In order to pass data streams to the proper applications, the Transport layer must identify the target application. To accomplish this, the Transport layer assigns an application an identifier. The TCP/IP protocols call this identifier a port number.

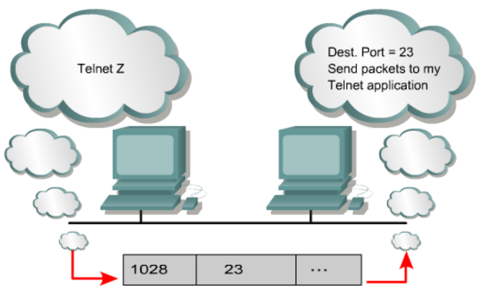
Each software process that needs to access the network is assigned a port number unique in that host. This port number is used in the transport layer header to indicate to which application that piece of data is associated.

What are ports?

- Ports are conceptual “points of entry” into a host computer.
- They do not correspond with real hardware but are an abstraction for convenience.
- Usually a service is associated with a port (e.g. http on port 80).
- Servers “listen on a port” for connection attempts.
- Ports provide one level of internet security.
- Generally, low number ports (<100) are reserved for special services.

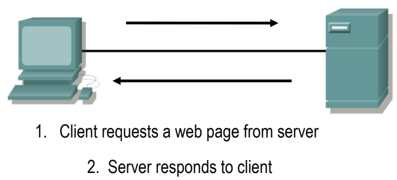
Transport Layer Ports

- Port numbers are used to keep track of different conversations that cross the network at the same time.
- Port numbers identify which upper layer service is needed, and are needed when a host communicates with a server that uses multiple services.



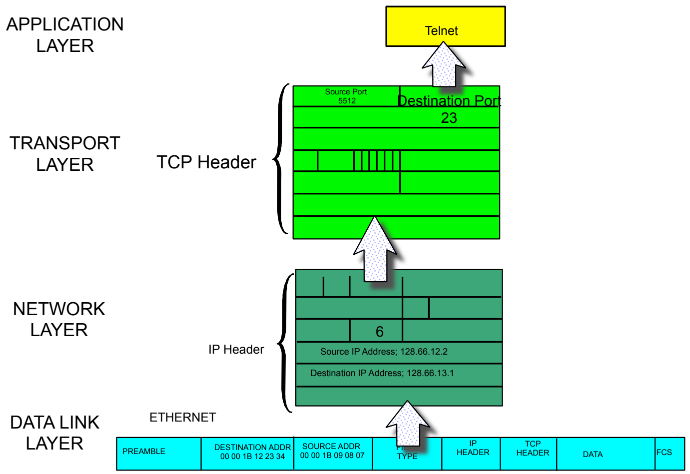
Ports for Clients

- Clients and servers both use ports to distinguish what process each segment is associated with.
- Source ports, which are set by the client, are determined dynamically, usually a randomly assigned a number above 1023.

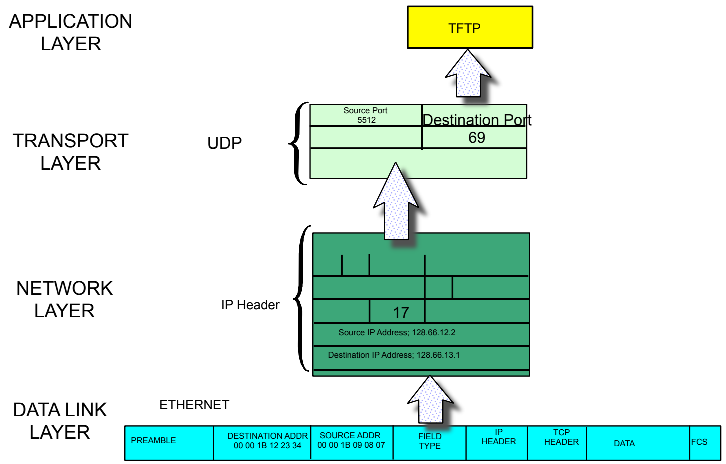


Source Port	Destination Port
1032	80
80	1032

Protocols and Port Numbers



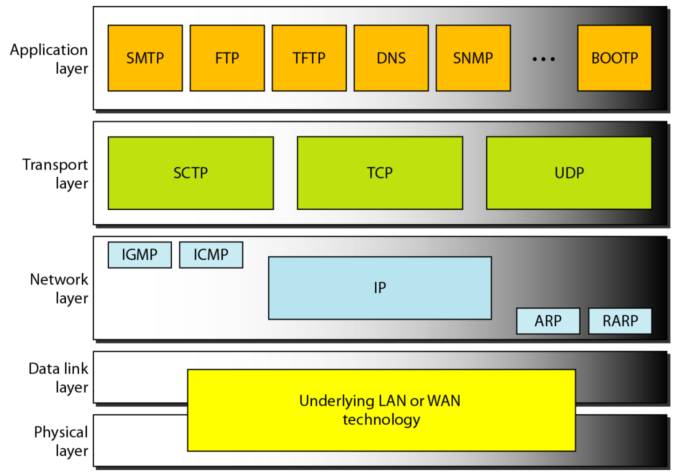
Protocols and Port Numbers



TCP and UDP

- The TCP and UDP based services keep track of the various applications that are communicating. To differentiate the segments and datagrams for each application, both TCP and UDP have header fields that can uniquely identify these applications.These unique identifiers are the port numbers.
- In the header of each segment or datagram, there is a source and destination port. The source port number is the number for this communication associated with the originating application on the local host. The destination port number is the number for this communication associated with the destination application on the remote host.
- Port numbers are assigned in various ways, depending on whether the message is a request or a response. While server processes have static port numbers assigned to them, clients dynamically chooses a port number for each conversation.

Position of UDP, TCP, and SCTP in TCP/IP suite



Socket

- The combination between IP address and port number is called socket and it's unique connection.

```
C:\>netstat

Active Connections

Proto    Local Address          Foreign Address          State
TCP      kenpc:3126             192.168.0.2:netbios-ssn  ESTABLISHED
TCP      kenpc:3158             207.138.126.152:http     ESTABLISHED
TCP      kenpc:3159             207.138.126.169:http     ESTABLISHED
TCP      kenpc:3160             207.138.126.169:http     ESTABLISHED
TCP      kenpc:3161             sc.msn.com:http          ESTABLISHED
TCP      kenpc:3166             www.cisco.com:http       ESTABLISHED

C:\>
```

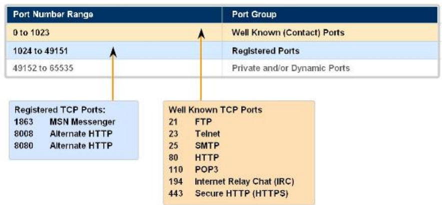
Port Numbers

- **Well Known Ports (Numbers 0 to 1023)** - These numbers are reserved for services and applications. They are commonly used for applications such as HTTP (web server) POP3/SMTP (e-mail server) and Telnet. By defining these well-known ports for server applications, client applications can be programmed to request a connection to that specific port and its associated service.
- **Registered Ports (Numbers 1024 to 49151)** - These port numbers are assigned to user processes or applications. These processes are primarily individual applications that a user has chosen to install rather than common applications that would receive a Well Known Port. When not used for a server resource, these ports may also be used dynamically selected by a client as its source port.
- **Dynamic or Private Ports (Numbers 49152 to 65535)** - Also known as Ephemeral Ports, these are usually assigned dynamically to client applications when initiating a connection. It is not very common for a client to connect to a service using a Dynamic or Private Port (although some peer-to-peer file sharing programs do).
- **Using both TCP and UDP**
  - Some applications may use both TCP and UDP. For example, the low overhead of UDP enables DNS to serve many client requests very quickly. Sometimes, however, sending the requested information may require the reliability of TCP. In this case, the well known port number of 53 is used by both protocols with this service.

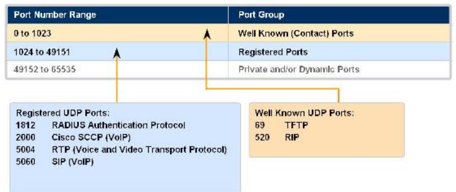
Some Well-Known TCP Ports

Port	Application	Description
9	Discard	Discard all incoming data port
7	Echo	Echo
19	Chargen	Exchange streams of data port
20	FTP-Data	File transfer data port
21	FTP-CMD	File transfer command port
23	Telnet	Telnet remote login port
25	SMTP	Simple Mail Transfer Protocol port
53	DOMAIN	Domain Name Service
79	Finger	Obtains information about active users
80	HTTP	Hypertext Transfer Protocol port
88	Kerberos	Authentication Protocol
110	POP3	PC Mail retrieval service port
119	NNTP	Network news access port
161	SMTP	Network Management
179	BGP	Border Gateway Protocol
513	Rlogin	Remote Login In

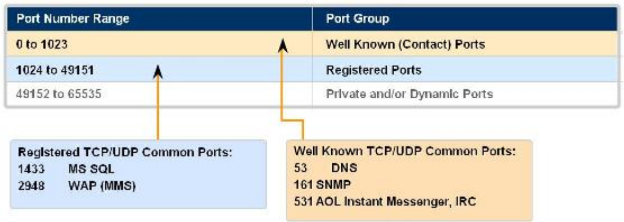
TCP Ports



UDP Ports



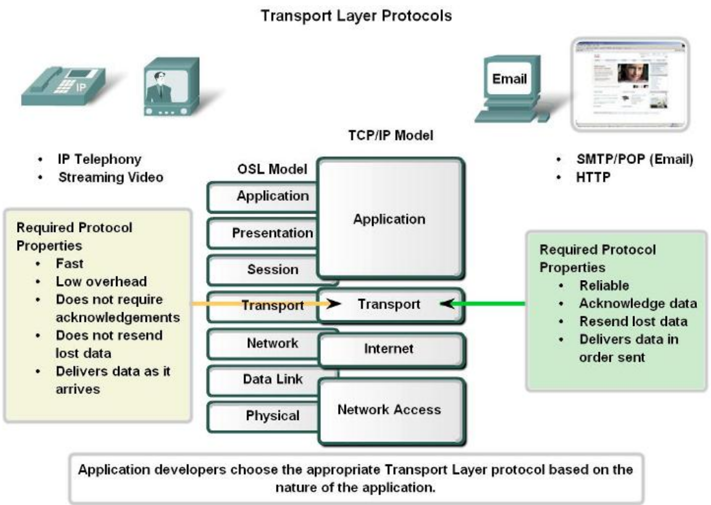
TCP and UDP Ports



- Some applications, such as online games or VoIP, can tolerate some loss of some data. If these applications used TCP, they may experience large delays while TCP detects data loss and retransmits data. These delays would be more detrimental to the application than small data losses. Some applications, such as DNS, will simply retry the request if they do not receive a response, and therefore they do not need TCP to guarantee the message delivery. The low overhead of UDP makes it very desirable for such applications.

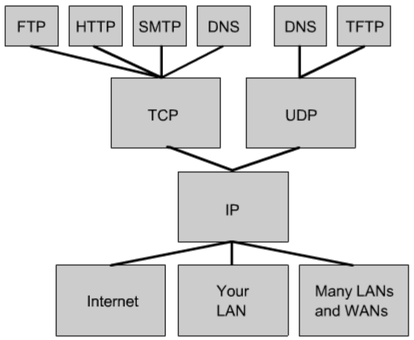
Determining the Need for Reliability

- Applications, such as databases, web pages, and e-mail, require that all of the sent data arrive at the destination in its original condition, in order for the data to be useful. Any missing data could cause a corrupt communication that is either incomplete or unreadable. Therefore, these applications are designed to use a Transport layer protocol that implements reliability. The additional network overhead is considered to be required for these applications.
- Other applications are more tolerant of the loss of small amounts of data. For example, if one or two segments of a video stream fail to arrive, it would only create a momentary disruption in the stream. This may appear as distortion in the image but may not even be noticeable to the user.



Connectionless vs Connection-oriented Protocols

- **Connection-oriented** – Two computers connect before sending any data, sender lets receiver know that data is on the way; recipient acknowledges receipt of data (ACK) or denies receipt (NACK). The ACKing and NACKing is called handshaking. (Type supported by TCP). Reliable, but carries overhead burden.
- **Connectionless** – Computers involved know nothing about each other or the data being sent. Makes no attempt to cause networks senders and receivers to exchange information about their availability or ability to communicate with one another, "best effort" delivery. (Type supported by IP, UDP). Not reliable, but faster and may be good enough. Also upper layer apps may worry about errors and reliability processing, so no need to do it twice.



TCP and UDP Protocols

- The two most common Transport layer protocols of TCP/IP protocol suite are Transmission Control Protocol (TCP) and User Datagram Protocol (UDP). Both protocols manage the communication of multiple applications. The differences between the two are the specific functions that each protocol implements.

User Datagram Protocol (UDP)

- UDP is a simple, connectionless protocol, described in RFC 768. It has the advantage of providing for low overhead data delivery. The pieces of communication in UDP are called datagrams. These datagrams are sent as "best effort" by this Transport layer protocol.

- Applications that use UDP include:
  - Domain Name System (DNS)
  - Video Streaming
  - Voice over IP (VoIP)

Transmission Control Protocol (TCP)

- TCP is a connection-oriented protocol, described in RFC 793. TCP incurs additional overhead to gain functions. Additional functions specified by TCP are the same order delivery, reliable delivery, and flow control. Each TCP segment has 20 bytes of overhead in the header encapsulating the Application layer data, whereas each UDP segment only has 8 bytes of overhead.

- Applications that use TCP are:
  - Web Browsers
  - E-mail
  - File Transfers

Transmission Control Protocol (TCP)

Transmission Control Protocol (TCP)

- The reliability of TCP communication is performed using connection-oriented sessions. Before a host using TCP sends data to another host, the Transport layer initiates a process to create a connection with the destination.
  - Establishes a session between source host and source destination (this ensures that each host is prepared and aware for the connection).
  - The destination host sends acknowledgements to the source for the segments that it receives.
  - As the source receives an acknowledgement, it knows that the data has been successfully delivered and can quit tracking that data.
  - If the source does not receive an acknowledgement within a predetermined amount of time, it retransmits that data to the destination.
  - The establishment of the sessions creates overhead in the form of additional segments being exchanged.
  - There is also additional overhead on the individual hosts created by the necessity to keep track of which segments are awaiting acknowledgement and by the retransmission process.

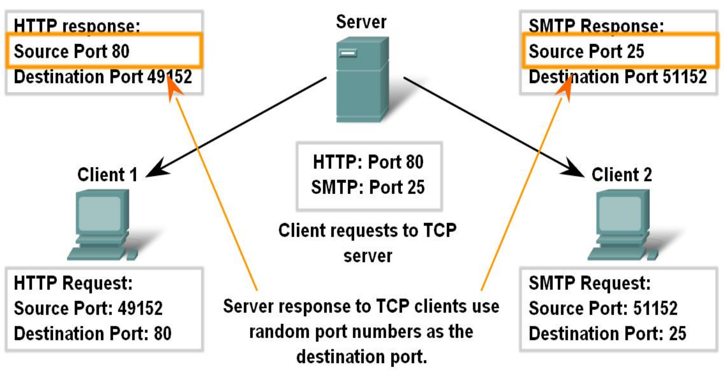
TCP Segment Structure

TCP Header

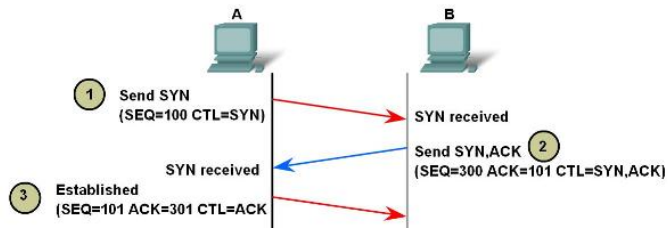
Bit offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	Source port																Destination port															
32	Sequence number																															
64	Acknowledgment number																															
96	Data offset				Reserved				CWR	ECE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

- Source port (16 bits) – identifies the sending port
- Destination port (16 bits) – identifies the receiving port
- Sequence number (32 bits) – has a dual role
- If the SYN flag is set, then this is the initial sequence number. The sequence number of the actual first data byte (and the acknowledged number in the corresponding ACK) will then be this sequence number plus 1.
- If the SYN flag is clear, then this is the sequence number of the first data byte
- Acknowledgment number (32 bits)
- Data offset (4 bits) – specifies the size of the TCP header in 32-bit words
- Reserved (4 bits) – for future use and should be set to zero
- Flags (8 bits) (aka Control bits) – contains 8 1-bit flags
- Window (16 bits) – the size of the receive window, which specifies the number of bytes that the receiver is currently willing to receive.
- Checksum (16 bits) – The 16-bit checksum field is used for error-checking of the header and data
- Urgent pointer (16 bits) – if the URG flag is set, then this 16-bit field is an offset from the sequence number indicating the last urgent data byte.

Clients Sending TCP Requests

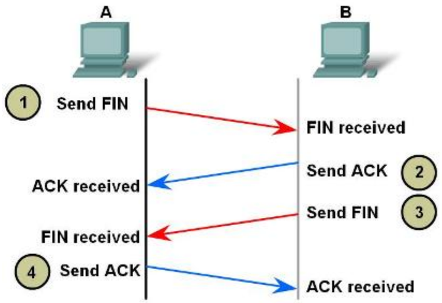


TCP Connection Establishment and Termination



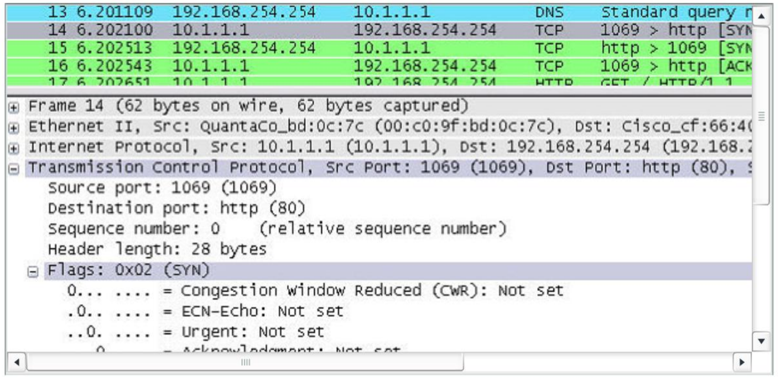
A sends SYN request to B
B sends ACK response and SYN request to A
A sends ACK response to B

TCP Connection Establishment and Termination



A sends FIN request to B
B sends ACK response to A
B sends FIN request to A
A sends ACK response to B

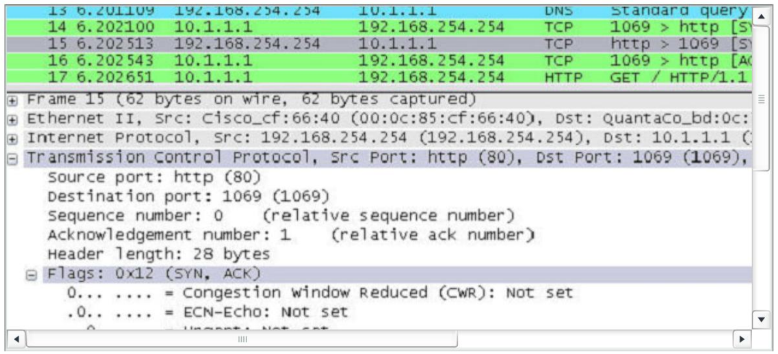
TCP 3-way Handshake (SYN)



Protocol Analyzer shows initial client request for session in frame 14

TCP segment in this frame shows:
- SYN flag set to validate an initial Sequence number
- Randomized sequence number valid (relative value is 0)
- Random source port 1069
- Well known destination port is 80 (HTTP port) indicates web server (httpd)

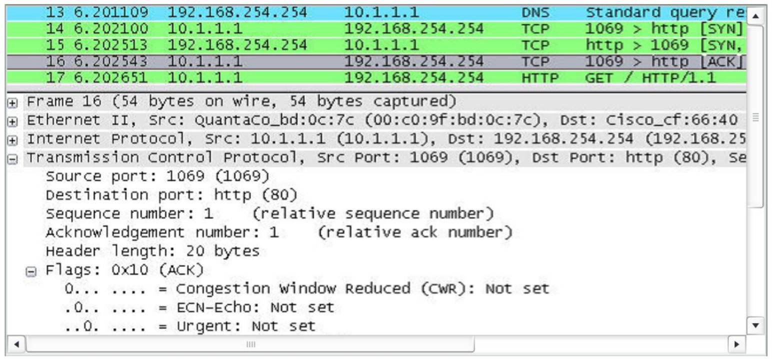
TCP 3-way Handshake (SYN, ACK)



A protocol analyzer shows server response in frame 15

- ACK flag set to indicate a valid Acknowledgement number
- Acknowledgement number response to initial sequence number as relative value of 1
- SYN flag set to indicate the Initial sequence number for the server to client session
- Destination port number of 1069 to corresponding to the clients source port
- Source port number of 80 (HTTP) indicating the web server service (httpd)

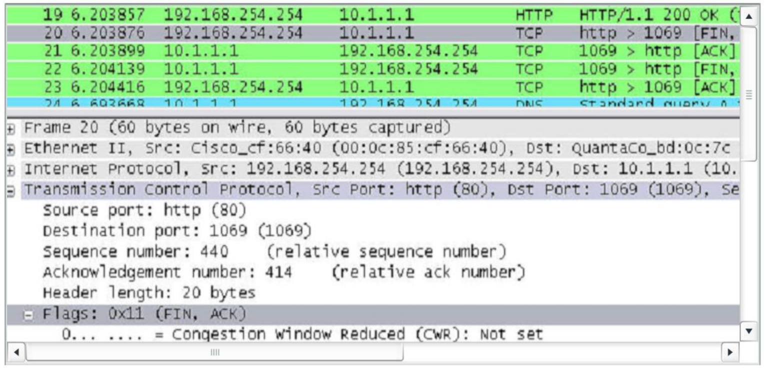
TCP 3-way Handshake (ACK)



Protocol Analyzer shows client response to session in frame 16

The TCP segment in this frame shows:
- ACK flag set to indicate a valid Acknowledgement number
- Acknowledgement number response to initial sequence number as relative value of 1
- Source port number of 1069 to corresponding
- Destination port number of 80 (HTTP) indicating the web server service (httpd)

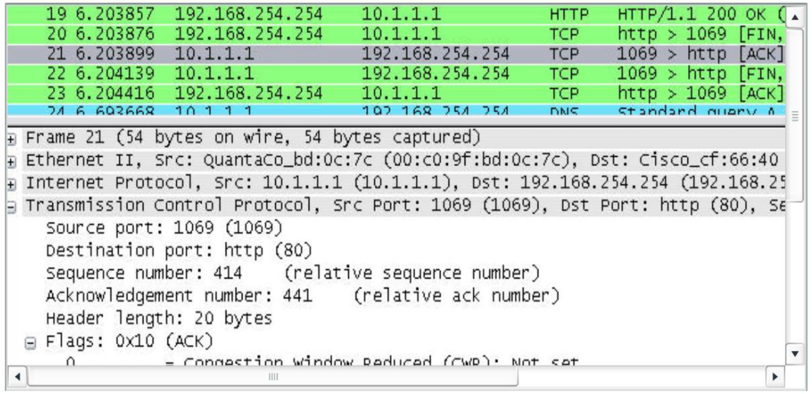
TCP Session Termination (FIN)



- A protocol analyzer shows details of frame 20, TCP FIN request.

- The destination and source ports
- The header field contents and values

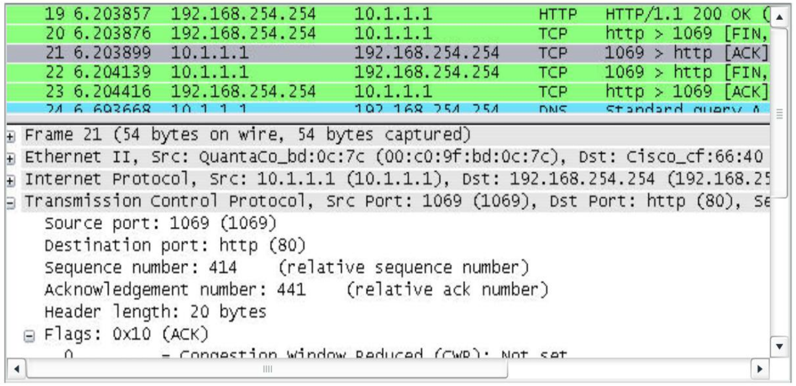
TCP Session Termination (ACK)



- A protocol analyzer shows details of frame 21, TCP ACK response.

- The destination and source ports
- The header field contents and values

TCP Session Termination (ACK)



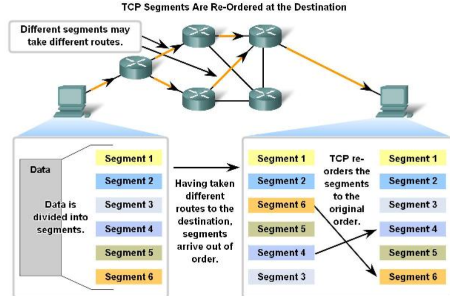
- A protocol analyzer shows details of frame 21, TCP ACK response.

- The destination and source ports
- The header field contents and values

TCP Segment Reassembly

- When services send data using TCP, segments may arrive at their destination out of order. For the original message to be understood by the recipient, the data in these segments is reassembled into the original order. Sequence numbers are assigned in the header of each packet to achieve this goal.

TCP Segments Are Re-Ordered at the Destination



User Datagram Protocol (UDP)

User Datagram Protocol

- UDP is a simple protocol that provides the basic Transport layer functions. It much lower overhead than TCP, since it is not connection-oriented and does not provide the sophisticated retransmission, sequencing, and flow control mechanisms.
- This does not mean that applications that use UDP are always unreliable. It simply means that these functions are not provided by the Transport layer protocol and must be implemented elsewhere if required.
- Although the total amount of UDP traffic found on a typical network is often relatively low, key Application layer protocols that use UDP include:
  - Domain Name System (DNS)
  - Simple Network Management Protocol (SNMP)
  - Dynamic Host Configuration Protocol (DHCP)
  - Routing Information Protocol (RIP)
  - Trivial File Transfer Protocol (TFTP)
  - Online games

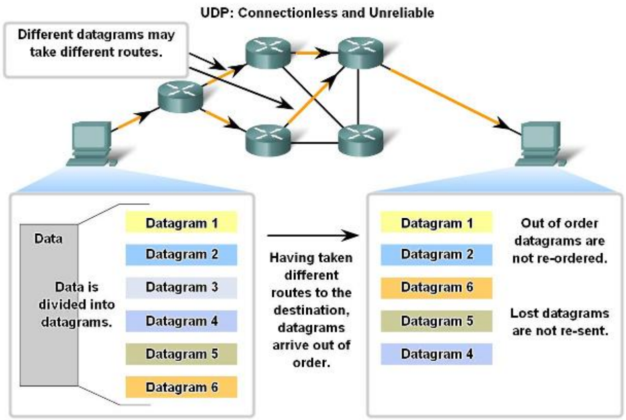
- Some applications, such as online games or VoIP, can tolerate some loss of some data. If these applications used TCP, they may experience large delays while TCP detects data loss and retransmits data. These delays would be more detrimental to the application than small data losses. Some applications, such as DNS, will simply retry the request if they do not receive a response, and therefore they do not need TCP to guarantee the message delivery. The low overhead of UDP makes it very desirable for such applications.

UDP Datagram Structure

bits	0 - 15	16 - 31
0	Source Port	Destination Port
32	Length	Checksum
64	Data	

- **Source port:** This field identifies the sending port when meaningful and should be assumed to be the port to reply to if needed. If not used, then it should be zero.
- **Destination port**: This field identifies the destination port and is required.
- **Length:** A 16-bit field that specifies the length in bytes of the entire datagram: header and data. The minimum length is 8 bytes since that's the length of the header. The field size sets a theoretical limit of 65,535 bytes (8 byte header + 65527 bytes of data) for a UDP datagram. The practical limit for the data length which is imposed by the underlying IPv4 protocol is 65,507 bytes.
- **Checksum**: The 16-bit checksum field is used for error-checking of the header and data. The algorithm for computing the checksum is different for transport over IPv4 and IPv6. If the checksum is omitted in IPv4, the field uses the value all-zeros. This field is not optional for IPv6.

UDP Datagram Reassembly



UDP/TCP Operation Comparison

- There are two protocols at Layer 4 – TCP and UDP. Both TCP and UDP use IP as their underlying protocol.
- TCP must be used when applications need to guarantee the delivery of a packet. When applications do not need a guarantee, UDP is used.
- UDP is often used for applications and services such as real-time audio and video. These applications require less overhead. They also do not need to be re-sequenced since packets that arrive late or out of order have no value.

TCP	UDP
Connection-oriented delivery	Connectionless delivery, faster
Uses windows and ACKs	No windows or ACKs
Full header	Smaller header, less overhead
Sequencing	No sequencing
Provides reliability	Relies on app layer protocols for reliability
FTP, HTTP, SMTP, and DNS	DNS, TFTP, SNMP, and DHCP

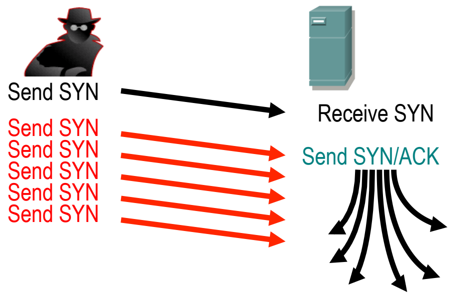
UDP segment format

← 0 – 15 →	← 16 - 31 →	← 31 - 47 →	← 48 – 63 →	64 →
Source Port	Destination Port	Length	Checksum	Data…

Denial of Service Attacks

DoS attacks are designed to deny services to legitimate users.
DoS attacks are used by hackers to overwhelm and crash systems.
SYN flooding is a DoS attack that exploits the three way handshake.

1. Hacker initiates a SYN but spoofs the source IP address.
2. Target replies to the unreachable IP address and waits for final ACK.
3. Hackers floods target with false SYN requests tying up its connection resources, preventing it from responding to legitimate connection requests.



To defend against these attacks, **decrease** the connection **timeout period** and **increase** the connection **queue size**. Software also exists that can detect these types of attacks and initiate defensive measures.

TUGAS

1. Buat resume tentang TCP dan UDP serta SCTP!
2. The following is a dump of a UDP header in hexadecimal format

06 32 00 0D 00 1C E2 17

a. What is the source port number?
b. What is the destination port number?
c. What is the total length of the user datagram?
d. What is the length of the data?
e. Is the packet directed from a client to a server or vise versa?
f. What is the client process?

3. The following is a dump of a UDP header in hexadecimal format

05320017 00000001 00000000 500207FF 00000000

a. What is the source port number?
b. What is the destination port number?
c. What the sequence number?
d. What is the acknowledgment number?
e. What is the length of the header?
f. What is the type of the segment?
g. What is the window size?