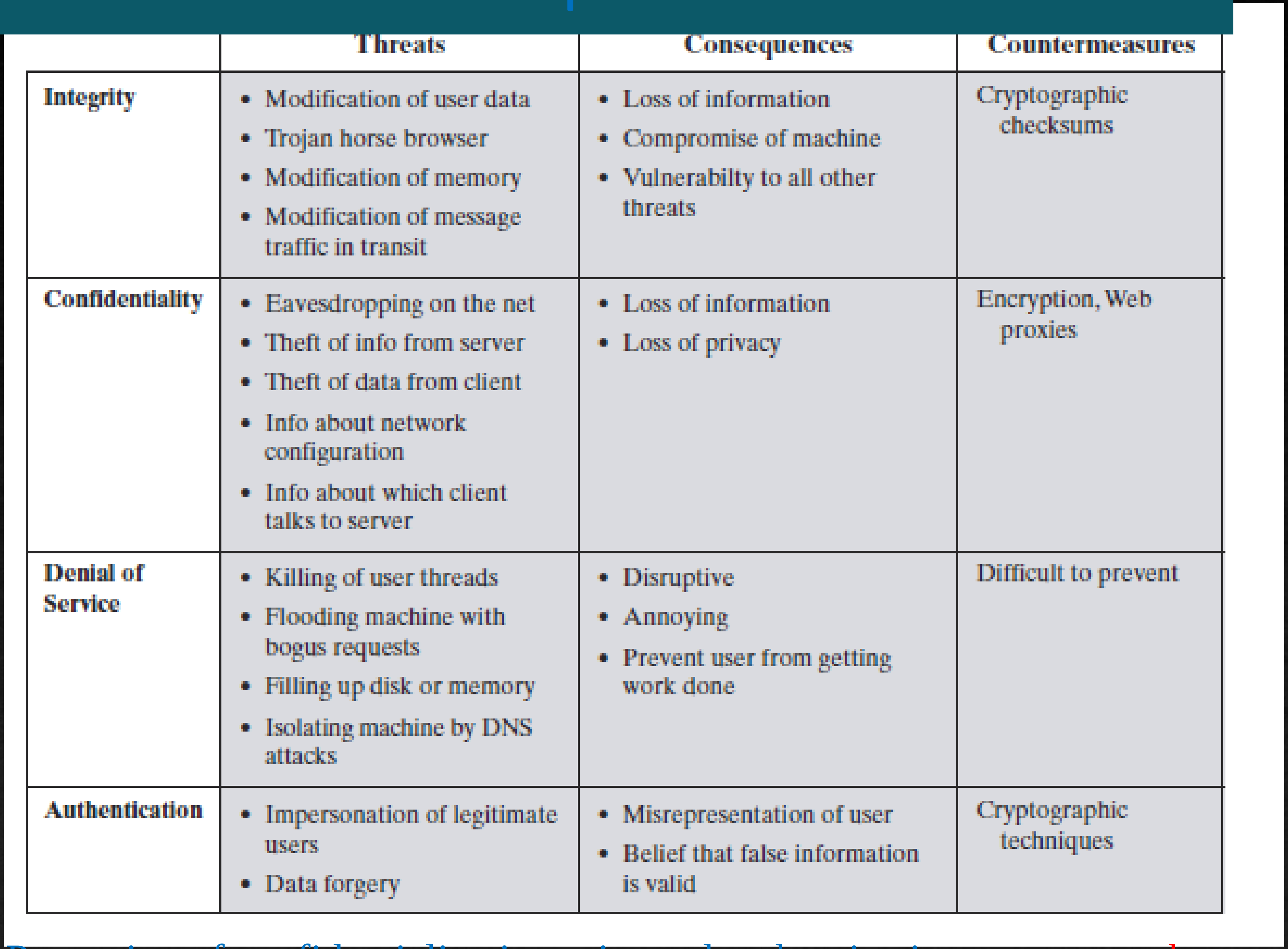
Network Vulnerabities

Double Tap to Add Subtitle

Vulnerabilities in Computer Networks



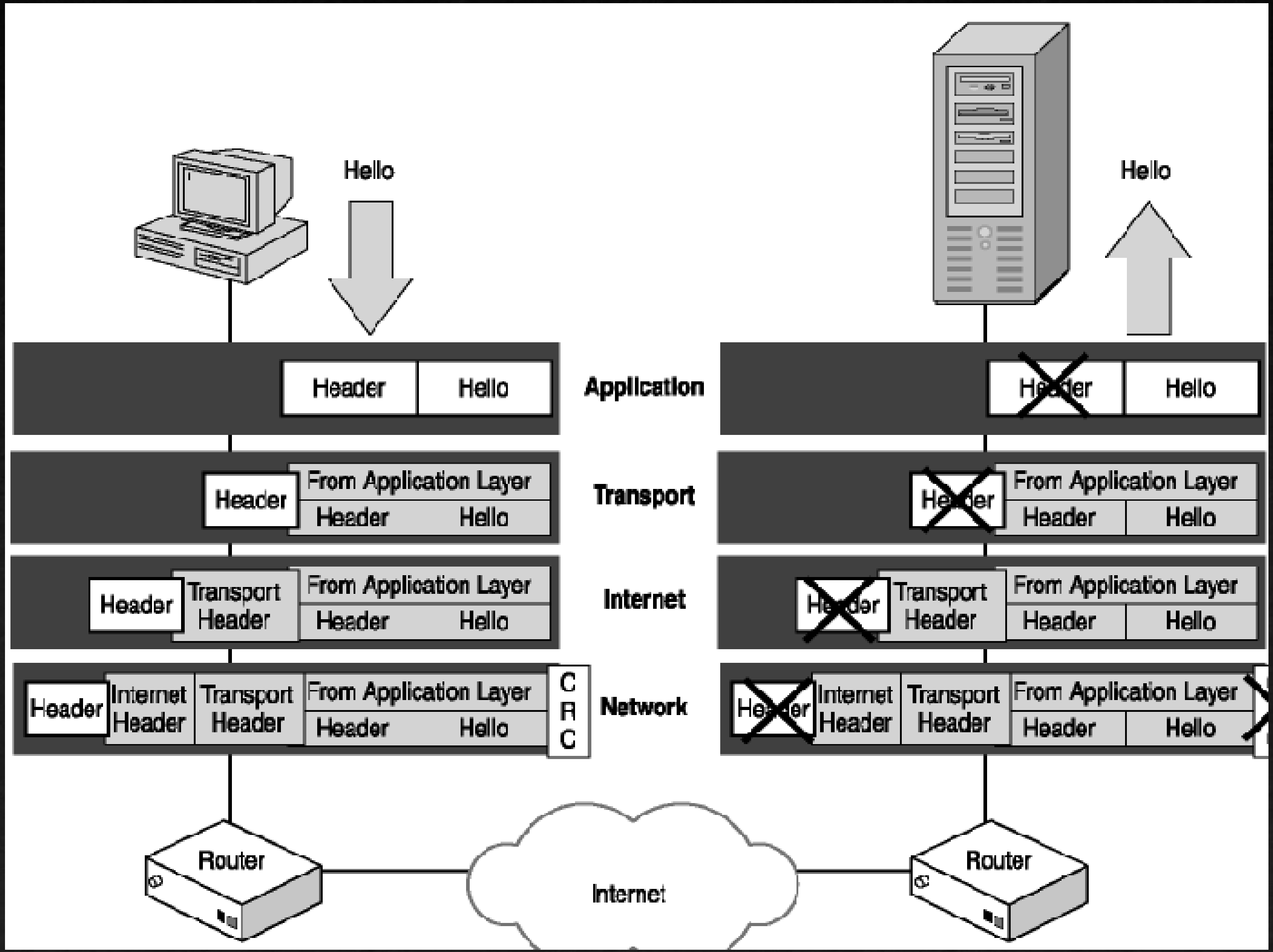
	Threats	Consequences	Countermeasures
Integrity	• Modification of user data • Trojan horse browser • Modification of memory • Modification of message traffic in transit	• Loss of information • Compromise of machine • Vulnerabilty to all other threats	Cryptographic checksums
Confidentiality	• Eavesdropping on the net • Theft of info from server • Theft of data from client • Info about network configuration • Info about which client talks to server	• Loss of information • Loss of privacy	Encryption, Web proxies
Denial of Service	• Killing of user threads • Flooding machine with bogus requests • Filling up disk or memory • Isolating machine by DNS attacks	• Disruptive • Annoying • Prevent user from getting work done	Difficult to prevent
Authentication	• Impersonation of legitimate users • Data forgery	• Misrepresentation of user • Belief that false information is valid	Cryptographic techniques

Protection of confidentiality, integrity and authentication: cryptography (see a seperate module – cryptography and SSL in transport layer)

Outline

- Review of computer network protocols
- Vulnerabilities in network layer and transportation layer
- Hands-on experiments
  - IP packets
  - IP routing
  - IP spoofing (Android studio & emulator)
  - TCP SYN flooding
  - Traffic analysis (Android studio & tablet)

Computer Network Protocols
– Pack and Unpack Data at Each Layer



IP Protocol

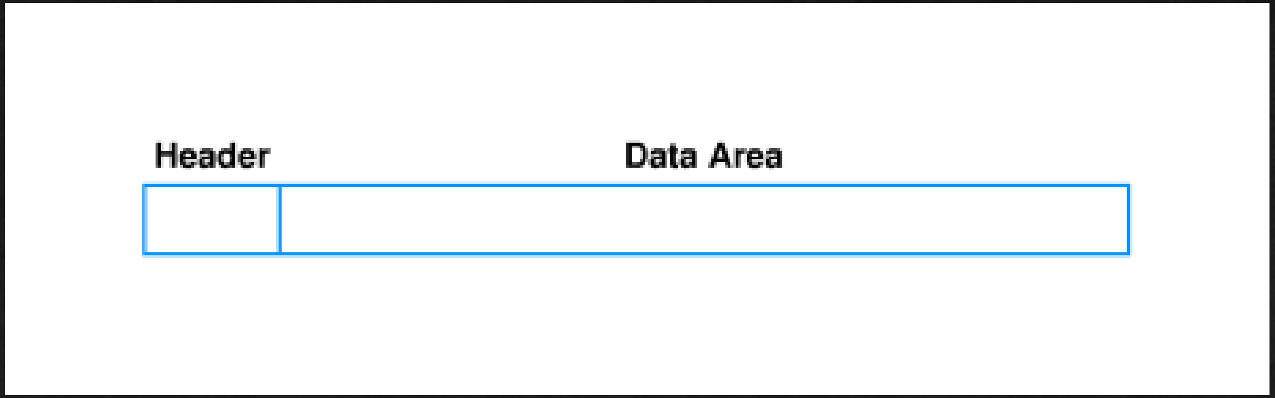
- IP is connectionless in the end-to-end delivery
  - Data delivered in datagrams (packets / frames), each with a header
- Combines collection of physical networks into single, virtual network
- Transport protocols use this connectionless service to provide connectionless data delivery (UDP) and connection-oriented data delivery (TCP)
  - But this is all done on top of IP, which is connectionless, so we'll need to implement quite a bit of extra logic in TCP to get the connection-oriented characteristics out of an underlying connectionless medium

IP Protocol: Virtual Packets

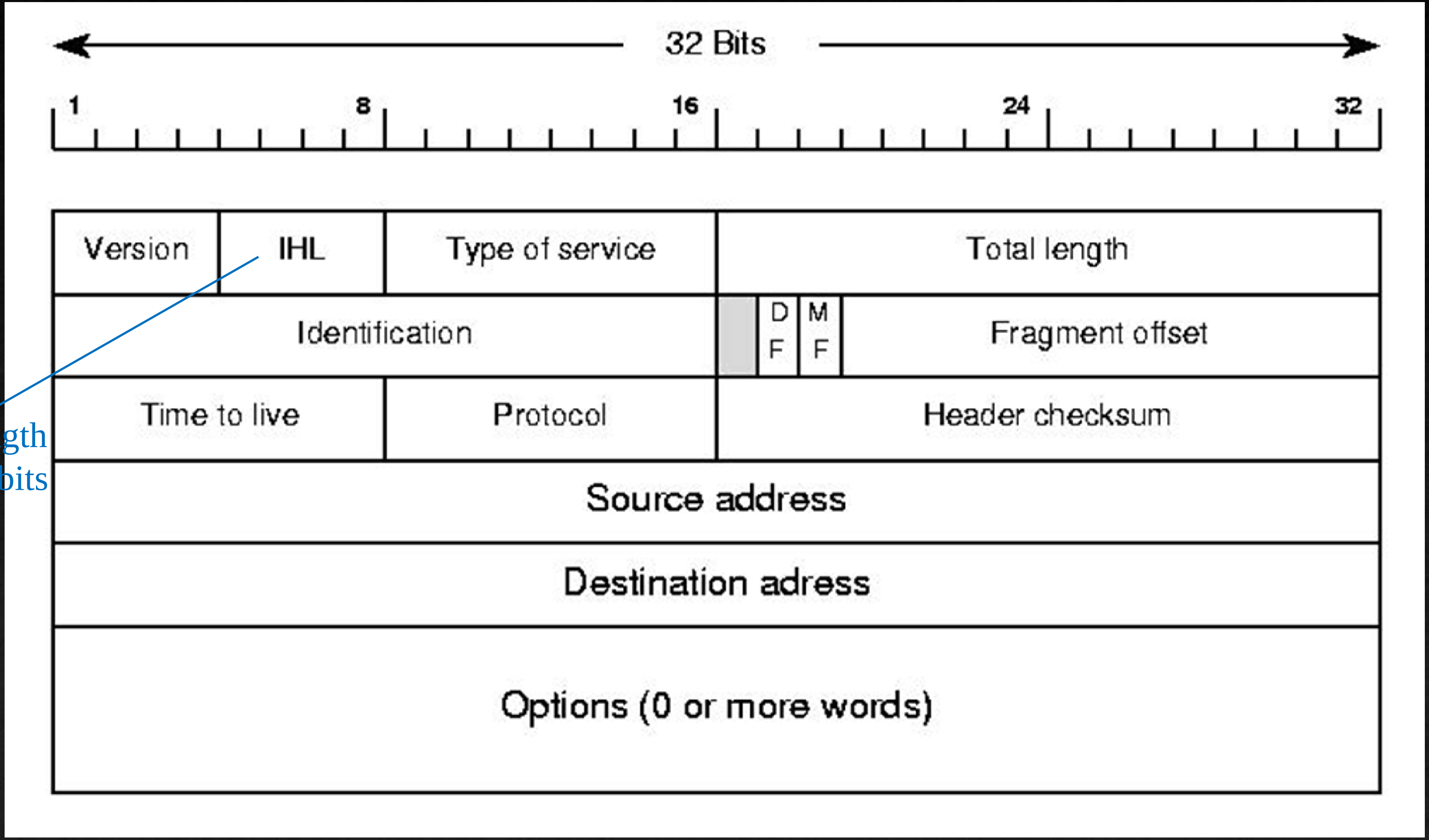
- *Packets* serve same purpose in internet as frames on LAN
- *Routers* (or *gateways*) forward packets between physical networks
- Packets have a uniform, hardware-independent format
  - Includes header and data
  - Why are these "virtual?" Because we would like a packet to be capable of crossing multiple networks, where networks could use different types of technologies (e.g. Token Ring, Ethernet)
- The virtual packet is implemented by encapsulating it in hardware frames for delivery across each physical network
  - Ensures universal format across heterogenous networks

IP Protocol: The IP Datagram

- Formally, the unit of IP data delivery is called a *datagram* which includes header area and data area
- Datagrams can have different sizes
  - Header area usually fixed (20 octets) but can have options
  - Data area can contain between 1 octet and 65,535 octets (2^{16}-1)
  - Usually, data area much larger than header (why?)



IP Protocol: Architecture of IP Header



Header length
= IHL*32 bits

IP Protocol: Architecture of IP address (I)

Historical classful network architecture: Classful network design allowed for a larger number of individual network assignments and fine-grained subnetwork design. Three classes (*A*, *B*, and *C*) were defined for universal unicast addressing. Each class used successively additional octets in the network identifier.

Historical classful network architecture

Class	Leading bits	Size of *network number* bit field	Size of *rest* bit field	Number of networks	Addresses per network	Start address	End address
A	0	8	24	128 (2^{7})	16,777,216 (2^{24})	0.0.0.0	127.255.255.255
B	10	16	16	16,384 (2^{14})	65,536 (2^{16})	128.0.0.0	191.255.255.255
C	110	24	8	2,097,152 (2^{21})	256 (2^{8})	192.0.0.0	223.255.255.255

First 8 bits:

A: 00000000 – 01111111   0 – 127
B: 10000000 – 10111111   128 – 191
C: 11000000 – 11011111   192 – 223

IP Protocol: Architecture of IP address (II)

Private addresses: Computers not connected to the Internet, such as factory machines that communicate only with each other via TCP/IP, need not have globally unique IP addresses. Three ranges of IPv4 addresses for private networks were reserved. These addresses are not routed on the Internet and thus their use need not be coordinated with an IP address registry.

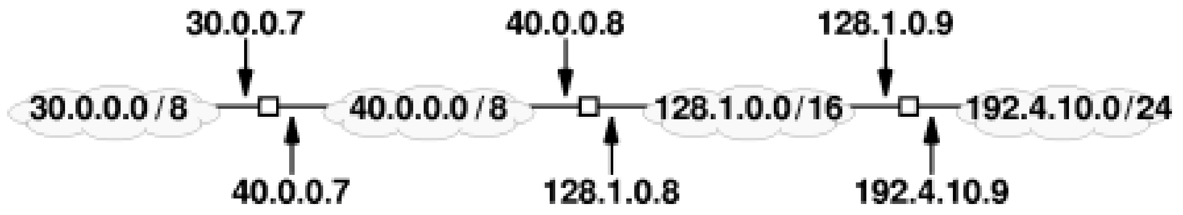
IANA-reserved private IPv4 network ranges			
	Start	End	No. of addresses
24-bit block (/8 prefix, 1 × A)	10.0.0.0	10.255.255.255	16777216
20-bit block (/12 prefix, 16 × B)	172.16.0.0	172.31.255.255	1048576
16-bit block (/16 prefix, 256 × C)	192.168.0.0	192.168.255.255	65536

IP Protocol: Forwarding Datagrams

- The header contains all the information needed to deliver a datagram to a destination *computer*
  - Destination address
  - Source address
  - Identifier
  - Other delivery information
- Routers examine the header of each datagram and forwards the datagram along a path to the destination
  - Use routing table to compute next hop
  - Update routing tables using algorithms
    - Link state, distance vector, Manually

IP Protocol: Routing Tables and Address Masks

- In practice, *destination* stored as network address
- *Next hop* stored as IP address of router
- *Address mask* defines how many bits are used to identify network
  - E.g., class A mask is 255.0.0.0
    class B mask is 255.255.0.0
    class C mask is 255.255.255.0



(a)

Destination	Mask	Next Hop
30.0.0.0	255.0.0.0	40.0.0.7
40.0.0.0	255.0.0.0	deliver direct
128.1.0.0	255.255.0.0	deliver direct
192.4.10.0	255.255.255.0	128.1.0.9

Routing Table for Center Router

(b)

IP Protocol: Address Masks

- To identify destination network, apply *address mask* to *destination address* and compare to *network address* in routing table by using Boolean AND

  if ((Mask[i] & D) == Dest[i]) forward to NextHop[i]

- Consider routing table at 128.1.15.26. Deliver a datagram to D = 192.4.10.9

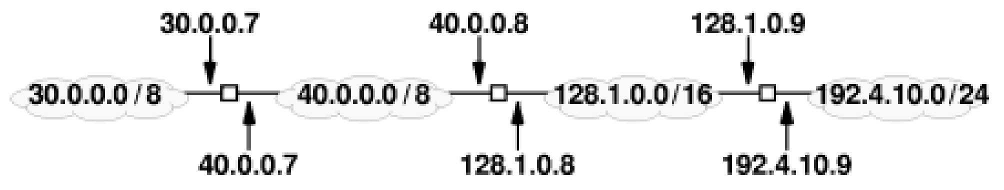
  Mask[1]&D = 255.0.0.0&192.4.10.9 = 192.0.0.0 ≠ Dest[1]

  Mask[2]&D = **255.0.0.0&192.4.10.9** = 192.0.0.0 ≠ Dest[2]

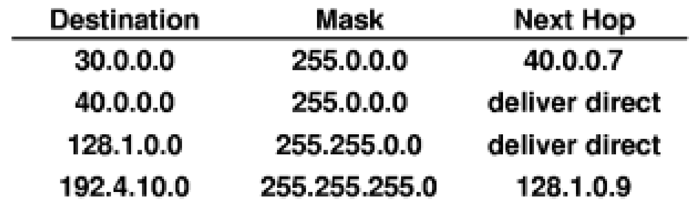
  Mask[3]&D = **255.255.0.0&192.4.10.9** = 192.4.0.0 ≠ Dest[3]

  Mask[4]&D = **255.255.255.0&192.4.10.9** = 192.4.10.0 = Dest[4];

  therefore forward the datagram to NextHop[4] (=128.1.0.9)



(a)



Destination	Mask	Next Hop
30.0.0.0	255.0.0.0	40.0.0.7
40.0.0.0	255.0.0.0	deliver direct
128.1.0.0	255.255.0.0	deliver direct
192.4.10.0	255.255.255.0	128.1.0.9

(b)

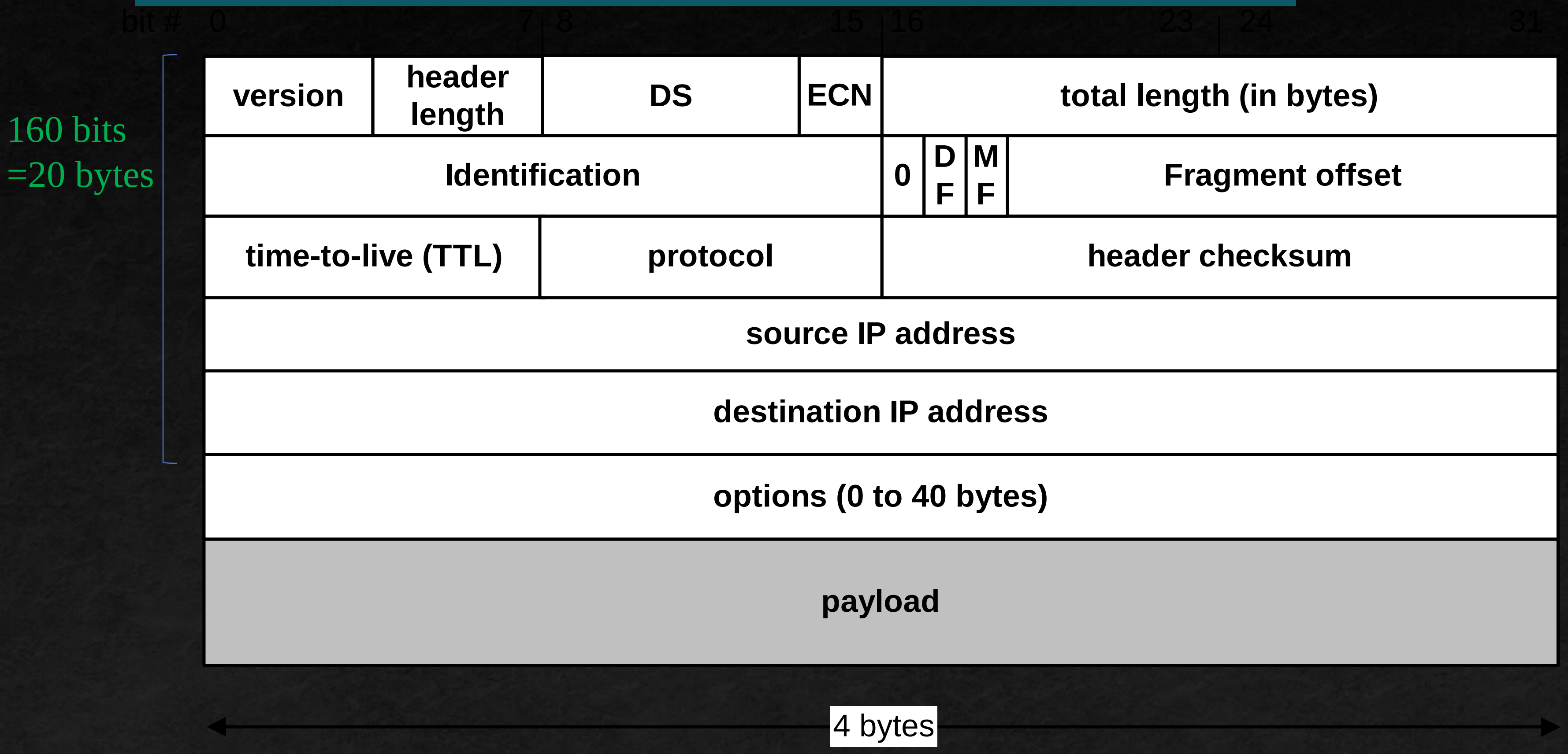
IP Protocol: Forwarding IP Packets

- *Destination address* in IP datagram is always ultimate destination
- Router looks up *next-hop address* and forwards datagram
- *Network interface layer* takes two parameters:
  - IP datagram
  - Next-hop address
- Next-hop address *never* appears in IP datagram

IP Protocol: IP is Best Effort Delivery

- IP provides service equivalent to LAN
- Does *not* guarantee to prevent
  - Duplicate datagrams
  - Delayed or out-of-order delivery
  - Corruption of data
  - Datagram loss
- *Reliable delivery* provided by *transport layer*
- *Network layer* (IP) – can *detect* and *report* errors without actually *fixing* them

IP Protocol: IP Datagram Format



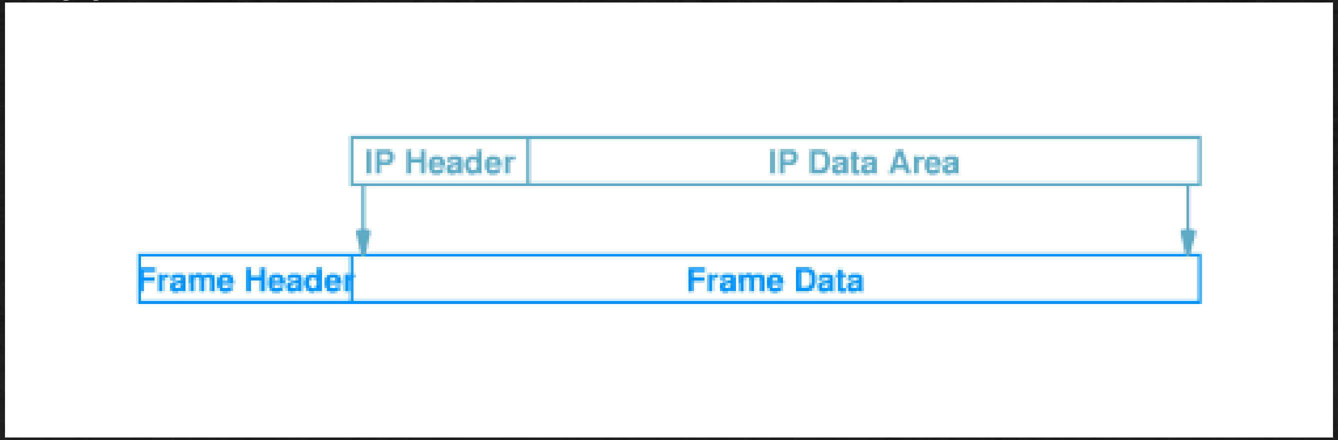
- 20 bytes ≤ Header Size < 2^4 x 4 bytes = 60 bytes
- 20 bytes ≤ Total Length < 2^{16} bytes = 65536 bytes

Datagram Transmission and Frames

- IP internet layer
  - Constructs datagram
  - Determines next hop
  - Hands to network interface layer
- Network interface layer
  - Binds next hop address to hardware address
  - Prepares datagram for transmission
- But ... hardware frame doesn't understand IP; how is datagram transmitted?

Encapsulation

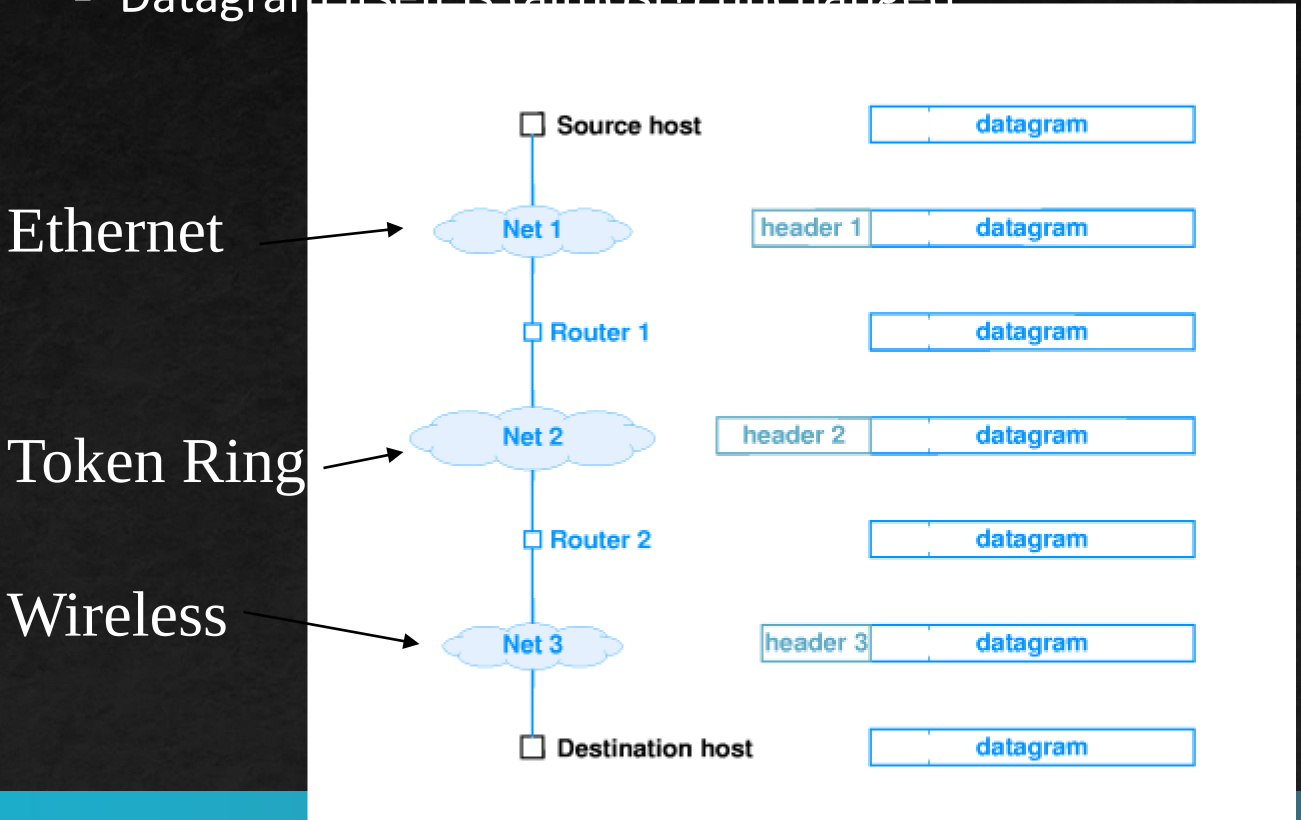
- Network interface layer *encapsulates* IP datagram as data area in hardware frame
  - Hardware ignores IP datagram format
  - Standards for encapsulation describe details
- Standard defines data type for IP datagram, as well as others (e.g., ARP)
- Receiving protocol stack interprets data area based on frame type



Encapsulation Across Multiple Hops

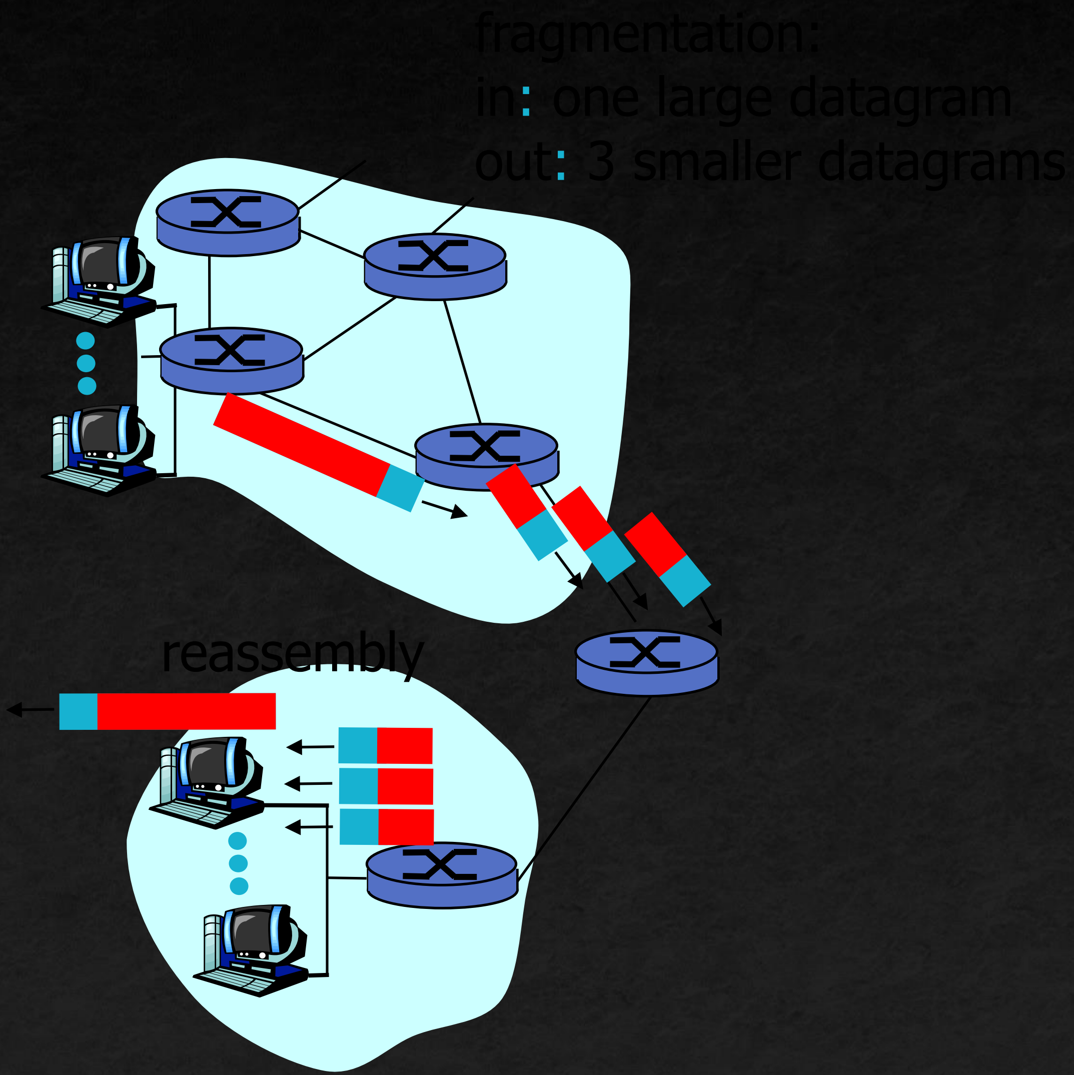
Each router in the path from the source to the destination:

- *Unencapsulates* incoming datagram from frame
- Processes datagram - determines next hop
- *Encapsulates* datagram in outgoing frame
- Datagram may be encapsulated in different hardware format at each hop
- Datagram itself is (almost!) unchanged

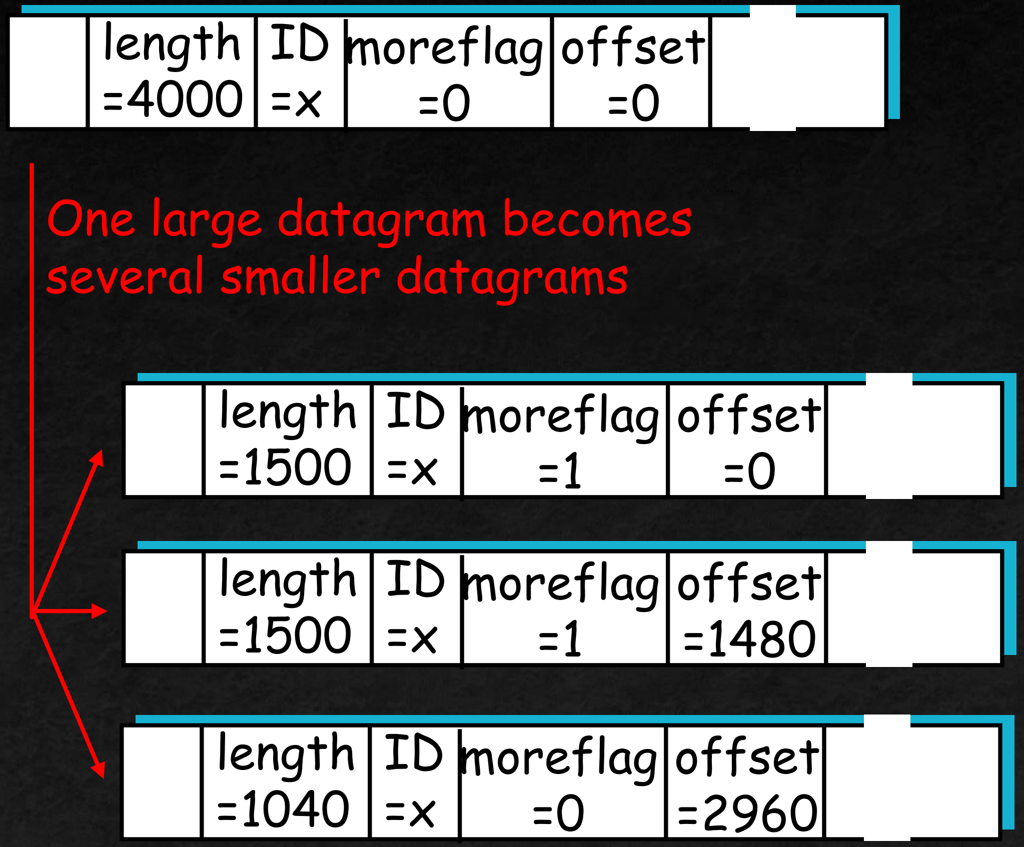


IP Fragmentation & Reassembly

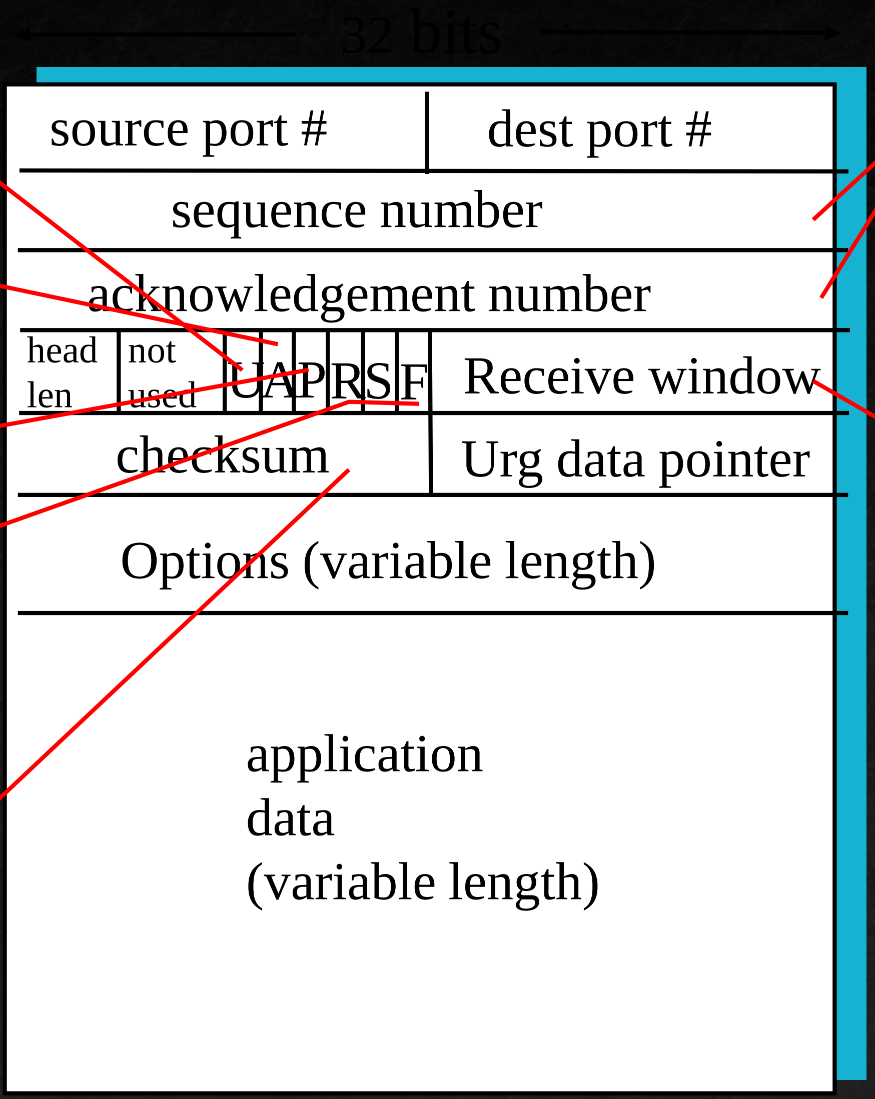
- Network links have MTU (max.transfer size) - largest possible link-level frame.
  - different link types, different MTUs
- large IP datagram divided ("fragmented") within net
  - one datagram becomes several datagrams
  - "reassembled" only at final destination
  - IP header bits used to identify, order related fragments



IP Fragmentation and Reassembly



TCP Protocol: TCP segment structure



URG: urgent data (generally not used)

ACK: ACK # valid

PSH: push data now (generally not used)

RST, SYN, FIN: connection estab (setup, teardown commands)

Internet checksum (as in UDP)

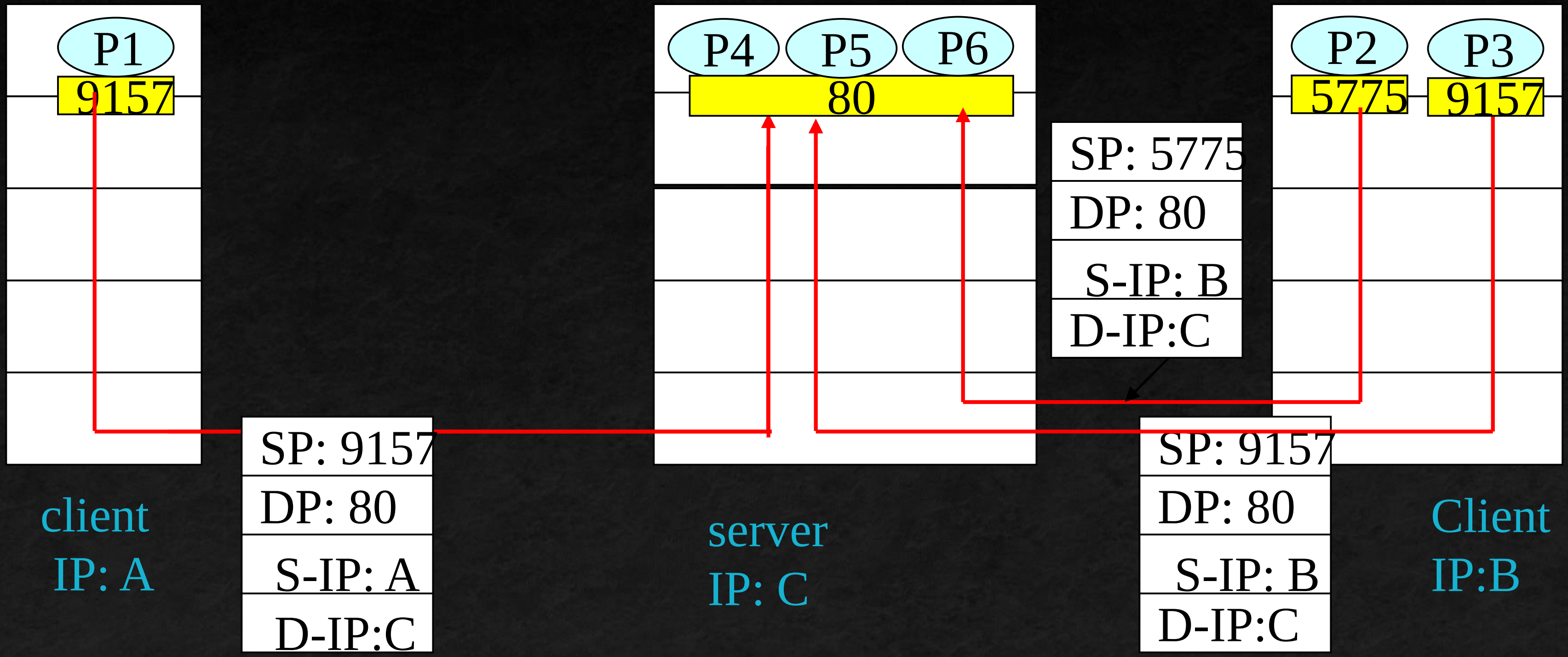
counting by bytes of data (not segments!)

# bytes rcvr willing to accept

Connection-oriented TCP Multiplexing and Demutiplexing

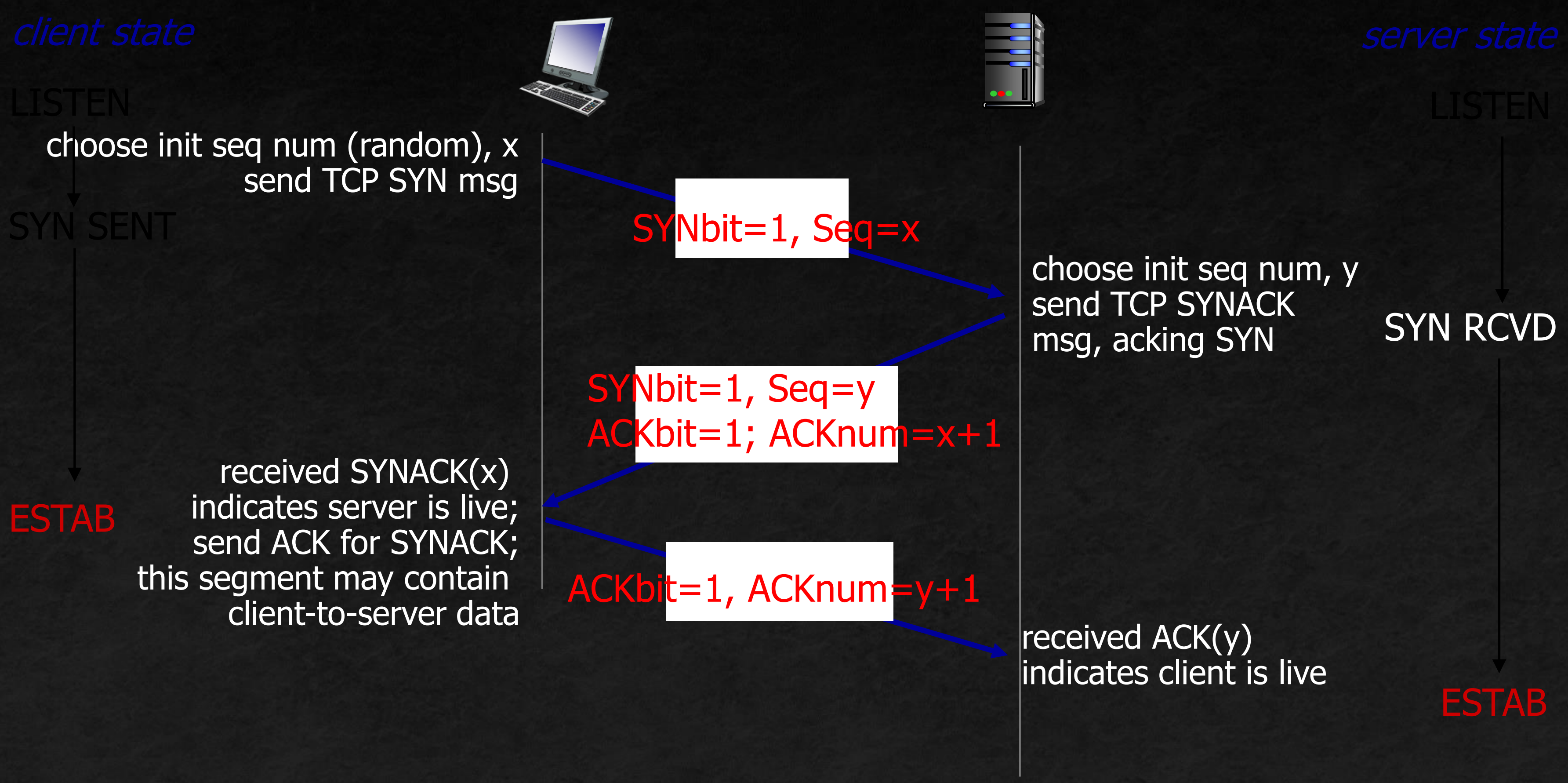
TCP socket identified by 4-tuple: source IP address, source port number, dest IP address, dest port number

Receiving host uses all four values to direct segment to appropriate socket



- A port is a unit door of an apartment building.
- A socket is a path to a port.
- A process is a person in the unit.
- An IP address is the street address of the building.

TCP: Establishing Connection with 3-way handshake



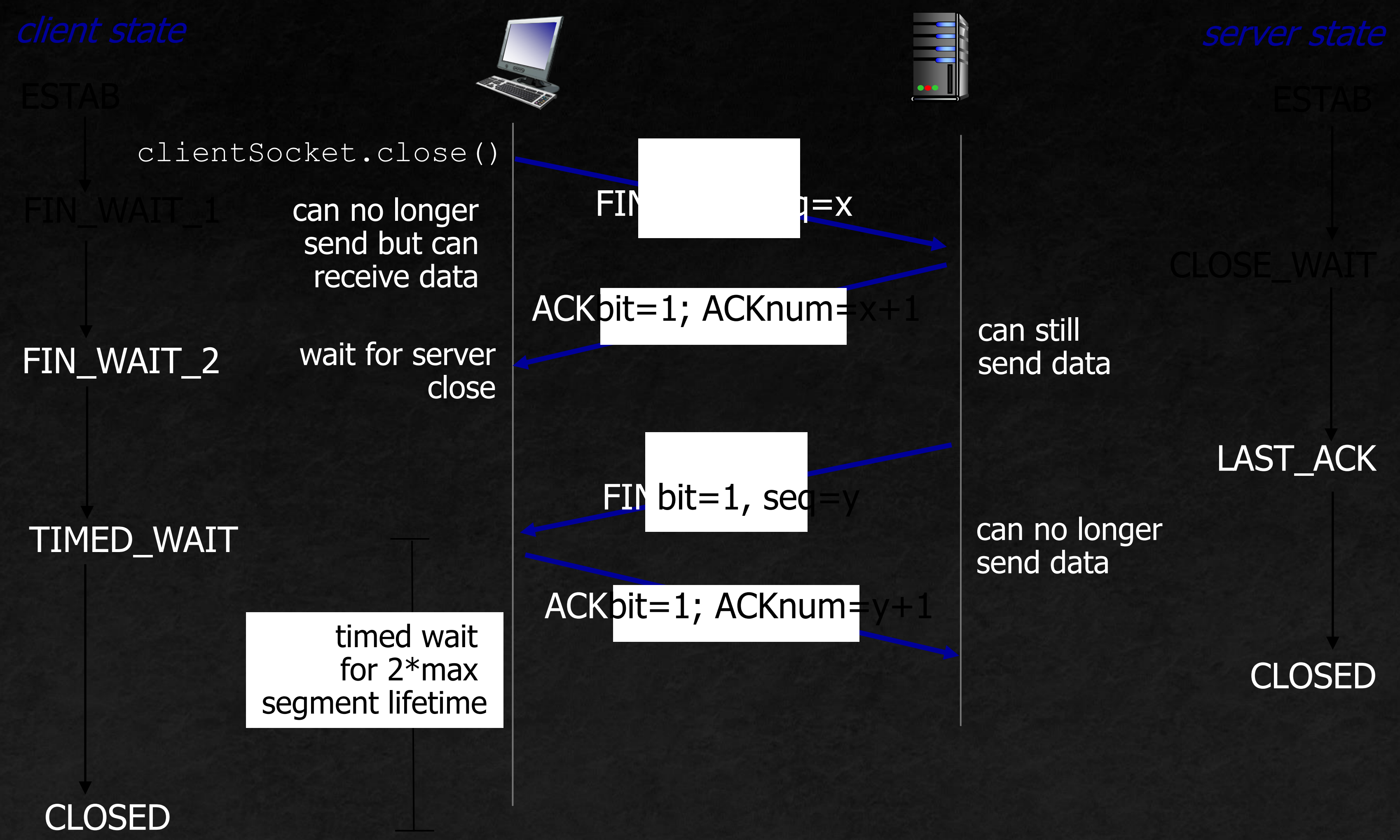
choose init seq num (random), x
send TCP SYN msg

choose init seq num, y
send TCP SYNACK
msg, acking SYN

received SYNACK(x)
indicates server is live;
send ACK for SYNACK;
this segment may contain
client-to-server data

received ACK(y)
indicates client is live

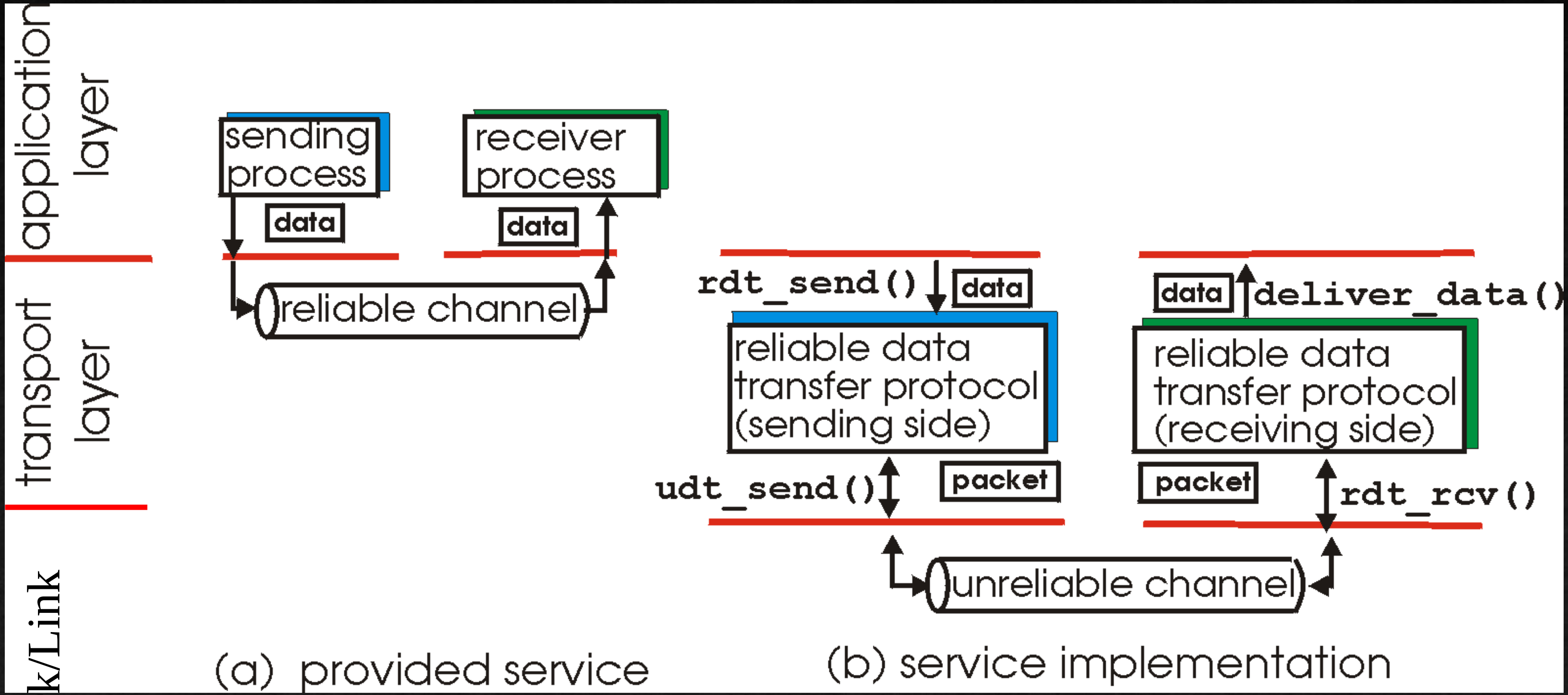
TCP: closing a connection



- ❖client, server each close their side of connection: send TCP segment with FIN bit = 1
- ❖respond to received FIN with ACK: on receiving FIN, ACK can be combined with own FIN
- ❖simultaneous FIN exchanges can be handled

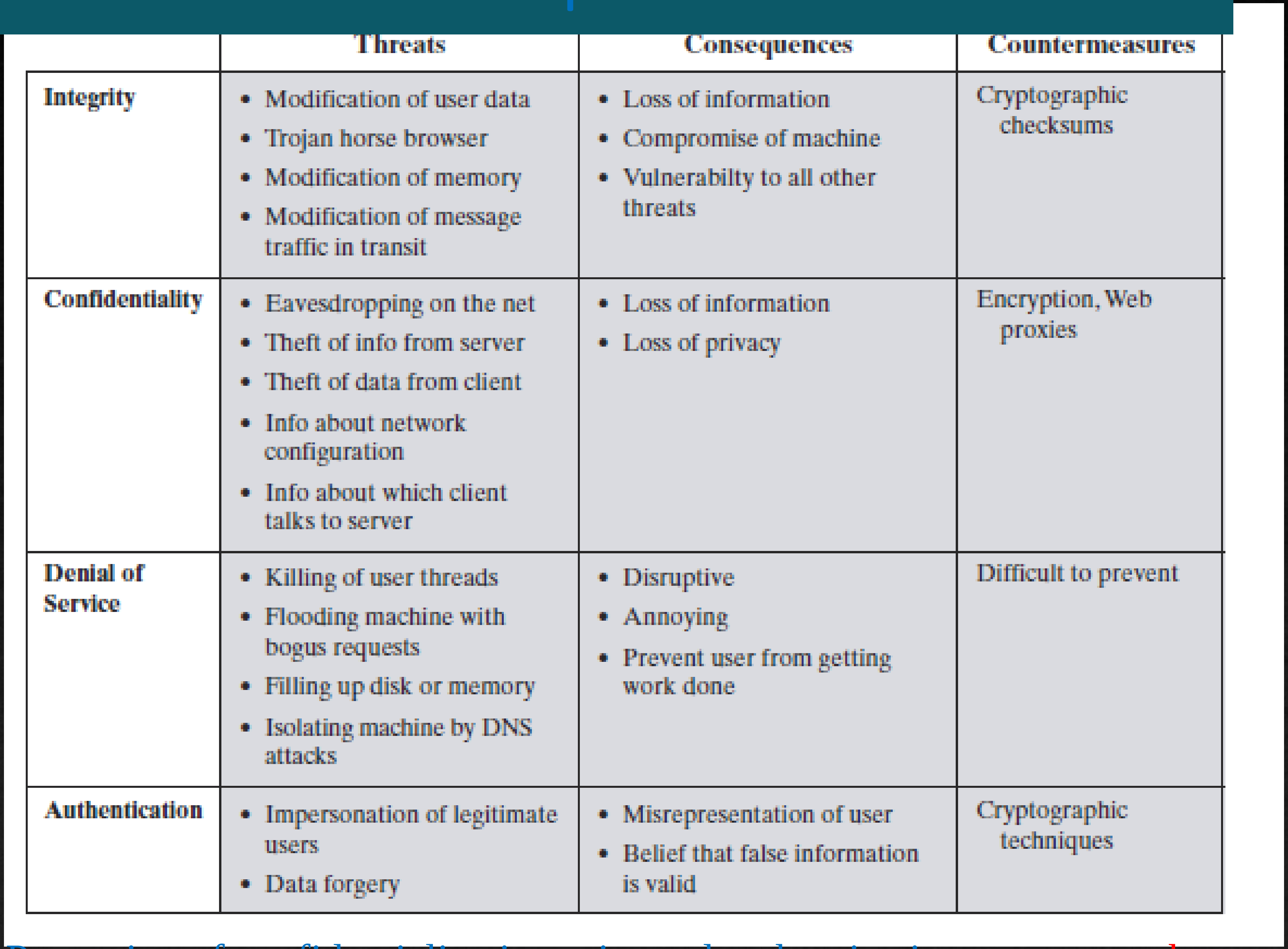
General Principles of Reliable data transfer

Important in app., transport, link layers



Network/Link Layer

Vulnerabilities in Computer Networks



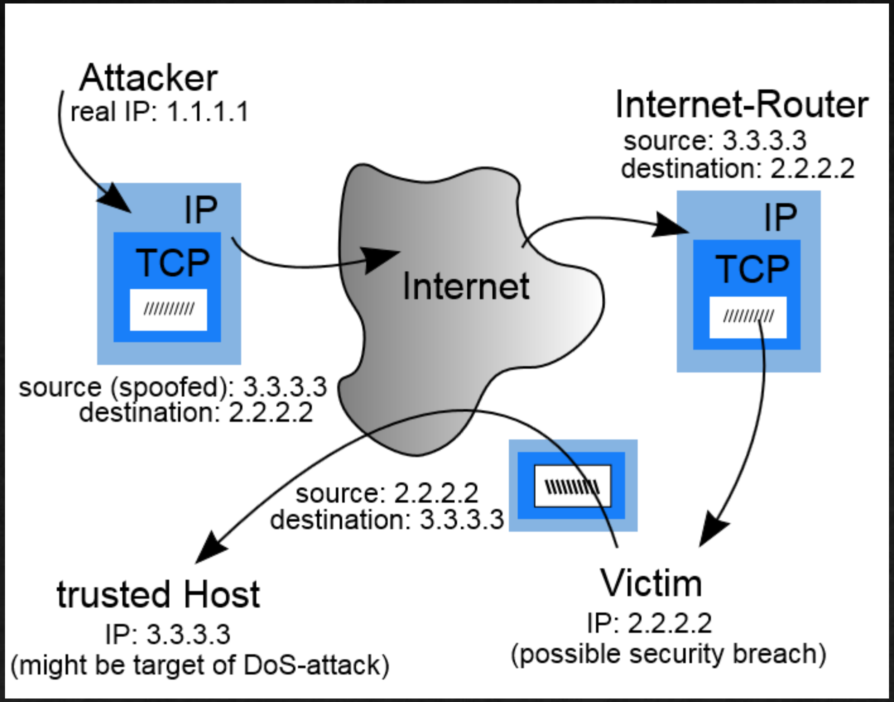
	Threats	Consequences	Countermeasures
Integrity	• Modification of user data • Trojan horse browser • Modification of memory • Modification of message traffic in transit	• Loss of information • Compromise of machine • Vulnerabilty to all other threats	Cryptographic checksums
Confidentiality	• Eavesdropping on the net • Theft of info from server • Theft of data from client • Info about network configuration • Info about which client talks to server	• Loss of information • Loss of privacy	Encryption, Web proxies
Denial of Service	• Killing of user threads • Flooding machine with bogus requests • Filling up disk or memory • Isolating machine by DNS attacks	• Disruptive • Annoying • Prevent user from getting work done	Difficult to prevent
Authentication	• Impersonation of legitimate users • Data forgery	• Misrepresentation of user • Belief that false information is valid	Cryptographic techniques

Protection of confidentiality, integrity and authentication: cryptography (see a seperate module – cryptography and SSL in transport layer)

Attacks in network layer

IP Spoofing

IP spoofing is sending IP packets with a buggy source address with intent to conceal the sender's identity. IP spoofing may be used in denial-of-service (DoS) attacks.



IP Spoofing - Continue

- IP spoofing is most frequently used in denial-of-service attacks. In such attacks, the goal is to flood the victim with overwhelming amounts of traffic, and the attacker does not care about receiving responses to the attack packets. Packets with spoofed addresses are thus suitable for such attacks. They are more difficult to filter if each spoofed packet appears to come from a different address, and they hide the true source of the attack.
- Denial of service attacks that use spoofing typically randomly choose addresses from the entire IP address space, though more sophisticated spoofing mechanisms might avoid unroutable addresses or unused portions of the IP address space. Attackers typically have a spoofing available tool, so defenses against denial-of-service attacks that rely on the validity of the source IP address in attack packets might have trouble with spoofed packets.

IP Spoofing - Continue

- IP spoofing can also be a method of attack used by network intruders to defeat network security measures, such as authentication based on IP addresses. This method of attack on a remote system can be extremely difficult, as it involves modifying thousands of packets at a time. This type of attack is most effective where trust relationships exist between machines.

  For example, it is common on some corporate networks to have internal systems trust each other, so that users can log in without a username or password provided they are connecting from another machine on the internal network (and so must already be logged in). By spoofing a connection from a trusted machine, an attacker may be able to access the target machine without authentication.

How to prevent IP Spoofing

1. Use an access control list to deny private IP addresses on your downstream interface.
2. Implement filtering of both inbound and outbound traffic.
3. Configure your routers and switches if they support such configuration, to reject packets originating from outside your local network that claim to originate from within.
4. Use authentication based on key exchange between the machines on your network; something like IPsec will significantly cut down on the risk of spoofing.
5. Enable encryption sessions on your router so that trusted hosts that are outside your network can securely communicate with your local hosts.

Other Attacks in Network Layer

Routing (RIP) Attack:

Routing Information Protocol (RIP) is used to distribute routing information within networks, such as shortest-paths, and advertising routes out from the local network. The original version of RIP has no built in authentication, and the information provided in a RIP packet is often used without verifying it.

- An attacker could forge a RIP packet, claiming his host "X" has the fastest path out of the network. All packets sent out from that network would then be routed through X, where they could be modified or examined.
- An attacker could also use RIP to effectively impersonate any host, by causing all traffic sent to that host to be sent to the attacker's machine instead.

Mitigations:

- New version of RIP was enhanced with a simple password authentication algorithm, which makes RIP attack harder to happen.
- Internet Protocol Security (Ipsec) provides a protocol suite for secure Internet Protocol (IP) communications by authenticating and encrypting each IP packet of a communication session.

ICMP Attack:

The **Internet Control Message Protocol** (**ICMP**) is used by network devices, like routers, to send error messages indicating, for example, that a requested service is not available or that a host or router could not be reached. There is no authentication in ICMP, which leads to attacks using ICMP that can result in a denial of service, or allowing the attacker to intercept packets.

An attacker sends forged ICMP echo packets to vulnerable networks' broadcast addresses. All the systems on those networks send ICMP echo replies to the victim, consuming the target system's available bandwidth and creating a denial of service (DoS) to legitimate traffic.

Mitigations:

- Most ICMP attacks can be effectively reduced by deploying Firewalls at critical locations of a network to filter un-wanted traffic and from any destinations.
- Configure your ICMP parameters in your network devices as follows:
  - Allow ping ICMP Echo-Request outbound and Echo-Reply messages inbound.
  - Allow traceroute TTL(Time to Live)-Exceeded and Port-Unreachable messages inbound.
  - Blocking other types of ICMP traffic.

Ping Flood (ICMP Flood)

PING is one of the most common uses of ICMP which sends an ICMP "Echo Request" to a host, and waits for that host to send back an ICMP "Echo Reply" message. Attacker simply sends a huge number of "ICMP Echo Requests" typically overloading its victim that it expends all its resources responding until it can no longer process valid network traffic.

Mitigations:

- Block ICMP altogether at perimeter of your network via firewall filters.
- Limit the rate at which a single source can send ICMP Packets.

Packet Sniffing:

Most network applications distribute network packets in clear/plain text. A packet sniffing tool can exploit information passed in clear text providing the hacker with sensitive information such as user account names and passwords.

Mitigations:

- ➢Authentication - Using strong authentication, such as one-time passwords.
- ➢Cryptography - The most effective method for countering packet sniffers does renders them irrelevant.
- ➢Anti-sniffer tools - Use these tools to employ software and hardware designed to detect the use of sniffers on a network.

Attacks in Transport Layer

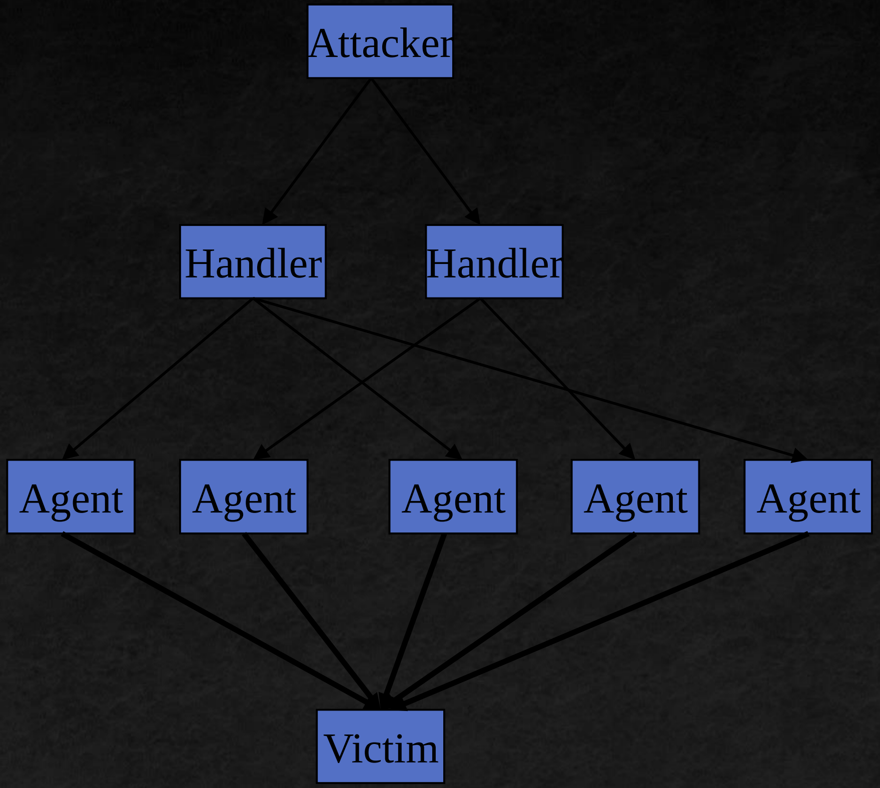
Denial of Service in Transport Layer

Make a service unusable, usually by overloading the server or network

- Consume host resources
  - TCP SYN floods
- Consume bandwidth
  - UDP floods
- Crashing the victim
  - TCP options (unused, or used incorrectly)
- Forcing more computation
  - Taking long path in processing of packets

Distributed DoS

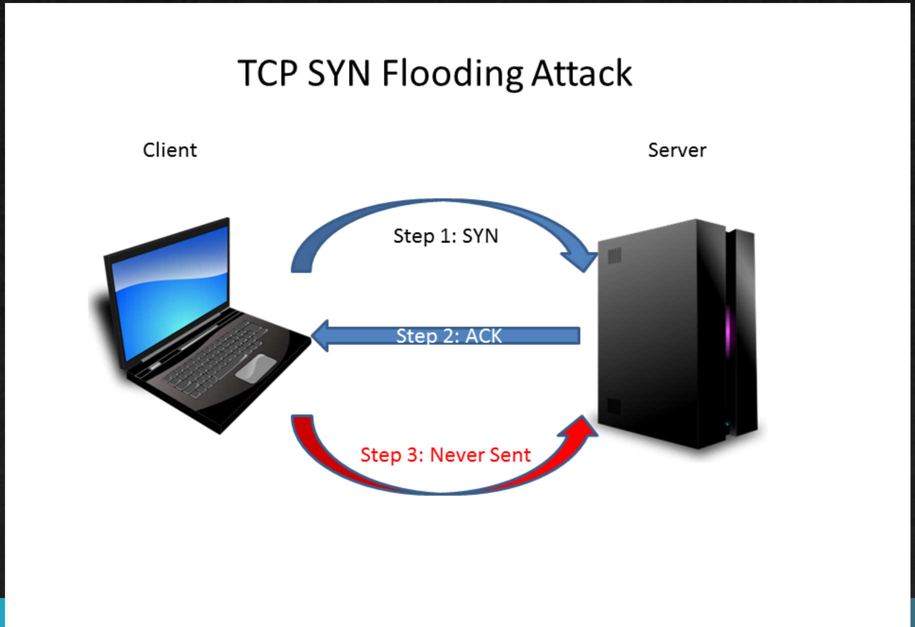
- The handlers are usually very high volume servers
  - Easy to hide the attack packets
- The agents are usually home users with DSL/Cable
  - Already infected and the agent installed
- Very difficult to track down the attacker
- How to differentiate between DDoS and Flash Crowd?
  - Flash Crowd: Many clients using a service legitimately
    - Slashdot Effect
    - Victoria Secret Webcast
  - Generally the flash crowd disappears when the network is flooded
  - Sources in flash crowd are clustered



TCP SYN Flooding

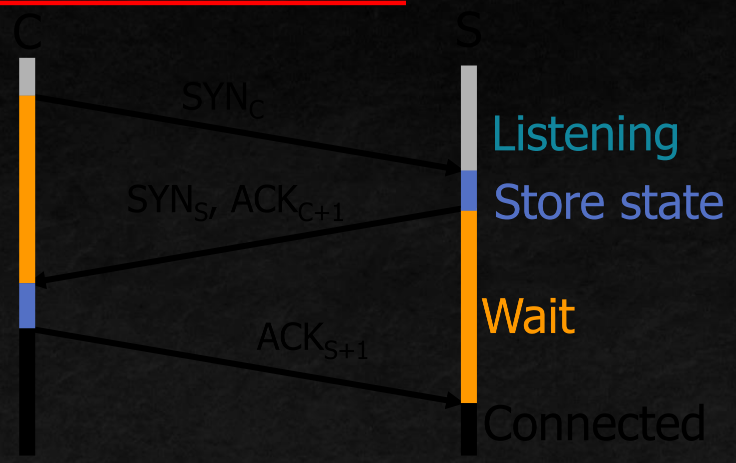
In a server that provides TCP connections for services such as Telnet, Web, Email, etc., lots of half-open TCP connections will cause a problem known as TCP SYN Flooding attack. This problem is due to the TCP 3-way hand-shaking protocol:

A client initiates a TCP connection by sending a TCP SYN packet to the server in Step 1. The server upon receiving the TCP SYN packet replies with an ACK packet in Step 2. However, the client may not send an ACK packet to the server to complete the TCP 3-way hand-shaking protocol. If the client keeps sending the SYN packets, the server will eventually run out of resource to other TCP connection requests.

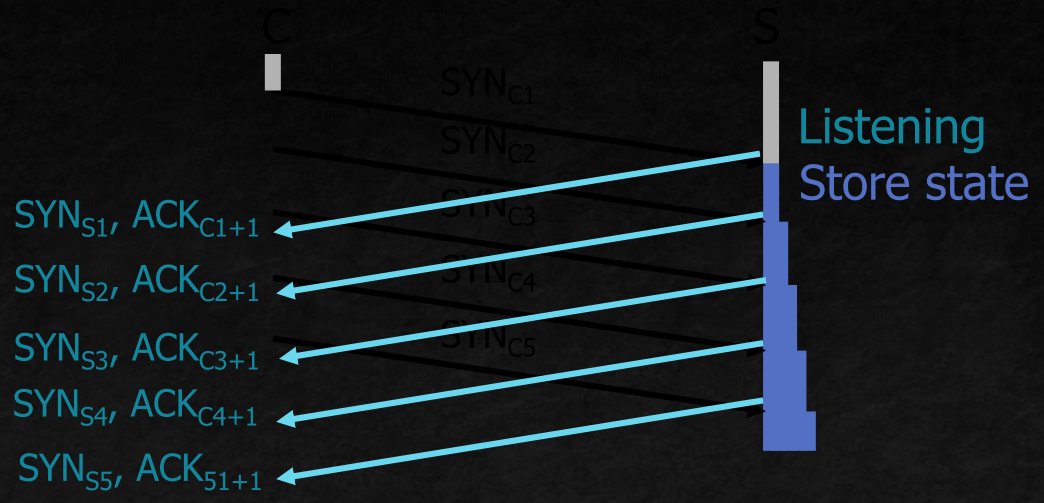


TCP SYN Flooding - Continue

TCP Three Way Handshake for establish connection



SYN Flooding



- The backlog queue is a large memory structure used to handle incoming packets with the SYN flag set until the moment the three-way handshake process is completed.
- An operating system allocates part of the system memory for every incoming connection. Every TCP port can handle a defined number of incoming requests. The backlog queue controls how many half-open connections can be handled by the operating system at the same time.
- When a maximum number of incoming connections is reached, subsequent requests are silently dropped by the operating system.

TCP SYN Flooding - Continue

Identifying the Attacker

The IP address of an attacking system is hidden because the source addresses in the SYN packets are often falsified. When the packet arrives at the server, there is no way to determine its true source IP address. Since the network forwards packets based on destination address, the only way to validate the source of a packet is to use input source filtering in the client side.

Attack Detection

Most of the operating systems provide a command line tool “netstat” to display protocol statistics and current TCP/IP network connections. The following command running on a Window 7 machine lists network connections. Pay attention to the state column. If there are lots of “SYN_RECEIVED” connections, the system is under attack. The SYN_RECEIVE state indicates that a connection request has been received from the network.

How to detect a TCP SYN attack

- The netstat command shows how many connections are currently in the half-open state. The half-open state is described as SYN_RECEIVED in Windows and as SYN_RECV in Unix systems.

```
 # netstat -n -p TCP
- tcp     0    0 10.100.0.200:21        237.177.154.8:25882     SYN_RECV
- tcp     0    0 10.100.0.200:21        236.15.133.204:2577     SYN_RECV
- tcp     0    0 10.100.0.200:21        127.160.6.129:51748     SYN_RECV
- tcp     0    0 10.100.0.200:21        230.220.13.25:47393     SYN_RECV
- tcp     0    0 10.100.0.200:21        227.200.204.182:60427   SYN_RECV
- tcp     0    0 10.100.0.200:21        232.115.18.38:278       SYN_RECV
- tcp     0    0 10.100.0.200:21        229.116.95.96:5122      SYN_RECV
- tcp     0    0 10.100.0.200:21        236.219.139.207:49162   SYN_RECV
- tcp     0    0 10.100.0.200:21        238.100.72.228:37899    SYN_RECV    - ...
```

- How many half-open connections are in the backlog queue at the moment can be counted. In the example below, 769 connections (for TELNET) in the SYN RECEIVED state are kept in the backlog queue.

```
# netstat -n -p TCP | grep SYN_RECV | grep :23 | wc -l 769
```

- The other method for detecting SYN attacks is to print TCP statistics and look at the TCP parameters which count dropped connection requests.
- TcpHalfOpenDrop parameter on a Sun Solaris machine.

```
# netstat -s -P tcp | grep tcpHalfOpenDrop        tcpHalfOpenDrop     =   473
```

- It is important to note that every TCP port has its own backlog queue, but only one variable of the TCP/IP stack controls the size of backlog queues for all ports.

Built-in Protection for SYN Flooding

The most important parameter in Windows 2000 and also in Windows Server 2003 is SynAttackProtect. Enabling this parameter allows the operating system to handle incoming connections more efficiently. The protection can be set by adding a SynAttackProtect DWORD value to the following registry key:
HKLM\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters

- When a SYN attack is detected the SynAttackProtect parameter changes the behavior of the TCP/IP stack. This allows the operating system to handle more SYN requests. It works by disabling some socket options, adding additional delays to connection indications and changing the timeout for connection requests. When the value of SynAttackProtect is set to 1, the number of retransmissions is reduced. The recommended value of SynAttackProtect is 2, which additionally delays the indication of a connection to the Windows Socket until the three-way handshake is completed.
- By enabling the SynAttackProtect parameter we don't change the TCP/IP stack behavior until under a SYN attack. But even then, when SynAttackProtect starts to operate, the operating system can handle legitimate incoming connections.

Built-in Protection for SYN Flooding - Continue

The operating system enables protection against SYN attacks automatically when it detects that values of the following three parameters are exceeded. These parameters are TcpMaxHalfOpen, TcpMaxHalfOpenRetried and TcpMaxPortsExhausted. To change the values of these parameters, first we have to add them to the same registry key as we made for SynAttackProtect.

- TcpMaxHalfOpen registry entry defines the maximum number of SYN RECEIVED states which can be handled concurrently before SYN protection starts working. The recommended value of this parameter is 100 for Windows 2000 Server and 500 for Windows 2000 Advanced Server.
- TcpMaxHalfOpenRetried defines the maximum number of half-open connections, for which the operating system has performed at least one retransmission, before SYN protection begins to operate. The recommended value is 80 for Windows 2000 Server, and 400 for Advanced Server.
- TcpMaxPortsExhausted registry entry defines the number of dropped SYN requests, after which the protection against

Other Attacks in Transport Layer

1. Session hijacking

This kind of attack occurs after a source and destination computer have established a communications link. A third computer disables the ability of one the computers to communicate, and then imitates that computer. Because the connection has already been established, the third computer can disrupt the C-I-A (confidentiality integrity and availability) triad.

Protection against session hijacking

- Use SSL/HTTPS encryption for the entire web site, and you have the best guarantee that no man in the middle attacks will be able to sniff an existing client session cookie
- use some sort of encryption on the session value itself that is stored in your session cookie

Protection against session hijacking Using HTTPS/SSL

HTTPS is a combination of the standard HTTP protocol and the cryptographic security of the SSL protocol. The HTTPS protocol contains mechanisms for secure identification of the server and encryption of the client-server communication.

- Obtain an SSL certificate from a Certificate Authority (CA). A CA is third party that the client trusts to verify that the site using the certificate is indeed the owner of the certificate. There are many CAs to choose from, Google provides a list of popular CAs. Then configure Internet Information Service (IIS) so that the site uses the certificate.
- Make Sure That the Session Cookie is Sent Over an Encrypted Connection
- In order to fully safeguard against session hijacking, make sure that all communication where the session cookie is sent is encrypted. There are two options to achieve this:

  Option 1 - Force SSL at All Times

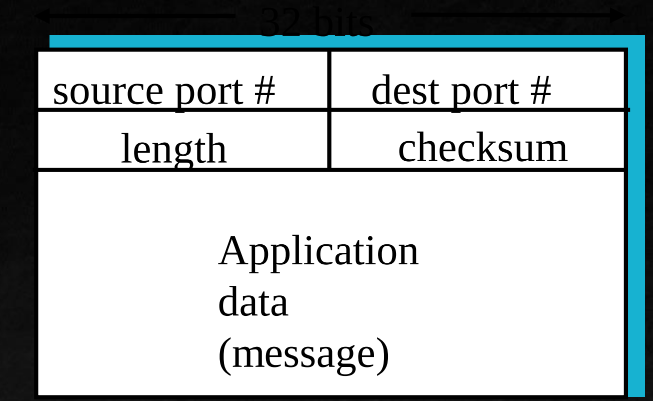
  Option 2 – Only Send Session Cookies Over SSL

  By setting requireSSL="true" on the forms-element in web.config, specify that the session cookie should only be sent when using the HTTPS protocol. This approach enables to use SSL only on parts of the site (edit/admin for example) and allow non-encrypted communication when browsing the public parts of the site.

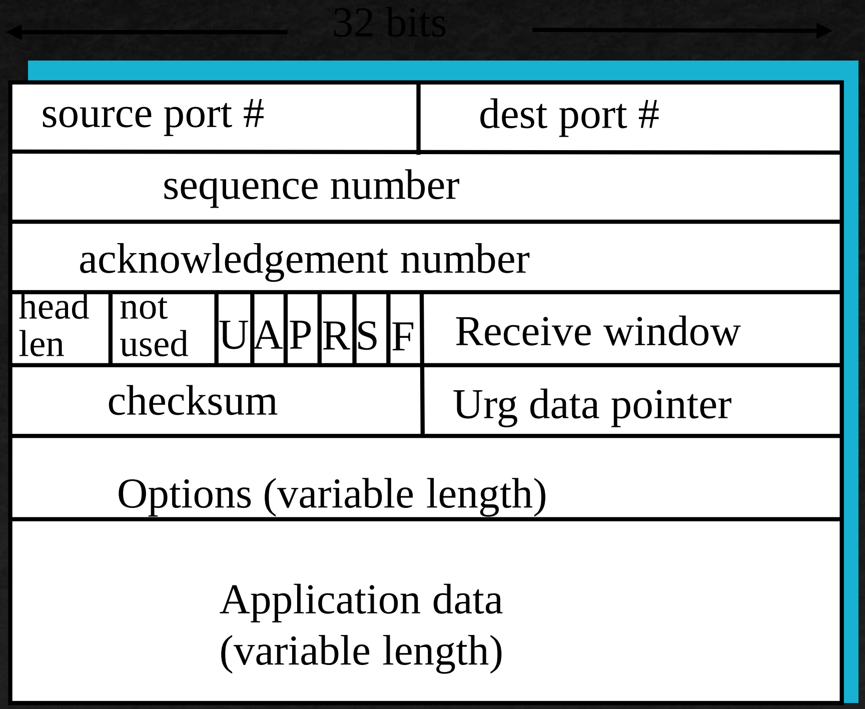
2. TCP Connection Spoofing

At the Transport layer, either a UDP or TCP header is added to the message. By knowing the UDP or TCP header fields and lengths, the ports that are used for communications between a source and destination computer can be identified, and that information can be corrupted or exploited.

- If attacker knows initial seq # and amount of traffic sent, it can estimate likely current values
- Send a flood of packets with likely seq numbers
- Attacker can inject packets into existing connection



UDP segment format



TCP segment format

How TCP connection spoofing works

Injecting IP packets which seems to originate from another host is insufficient to impersonate that host during a TCP connection, because every TCP segment has a 32bit sequence number. A segment with a sequence number which is out of line will be ignored. That means in order to successfully insert a TCP segment into an existing transmission it needs to guess the next sequence number, otherwise the segment will be discarded.

- This isn't so hard when the attacker can eavesdrop at least on the client; otherwise, it can only brute-force the sequence number.
- With more simple transport protocols, like UDP for example, the attacker doesn't have problem. UDP has no sequence numbers, so unless an upper protocol layer replicates the functionality similar to sequence numbers, it can just insert additional segments which will then be treated as if they were coming from the real host.
- If the attacker doesn’t know existing connection and instead want to establish a new TCP connection which appears to originate from another host, a 3-way handshake is required (client sends SYN, server sends ACK, client sends SYNACK). The ACK by the server includes a random number which the attack needs for an acceptable SYNACK. So when it can only send IP packets but not receive any of the packets intended for the spoofed host, attacler will have to guess this random number.

Transport Layer Solution

The transport layer represents the last place that segments can be authenticated before they affect connection management. TCP has a variety of current and proposed mechanisms to increase the authentication of segments, protecting against both off-path and on-path third-party spoofing attacks.Other transport protocols, such as SCTP and DCCP, also have limited antispoofing mechanisms.

1. TCP MD5 Authentication
2. TCH RST window attenuation
3. TCP timestamp authentication