Introduction to Information Security

—— IP Security

Dr. Tianlei HU

Associate Professor

College of Computer Science, Zhejiang Univ.

htl@zju.edu.cn

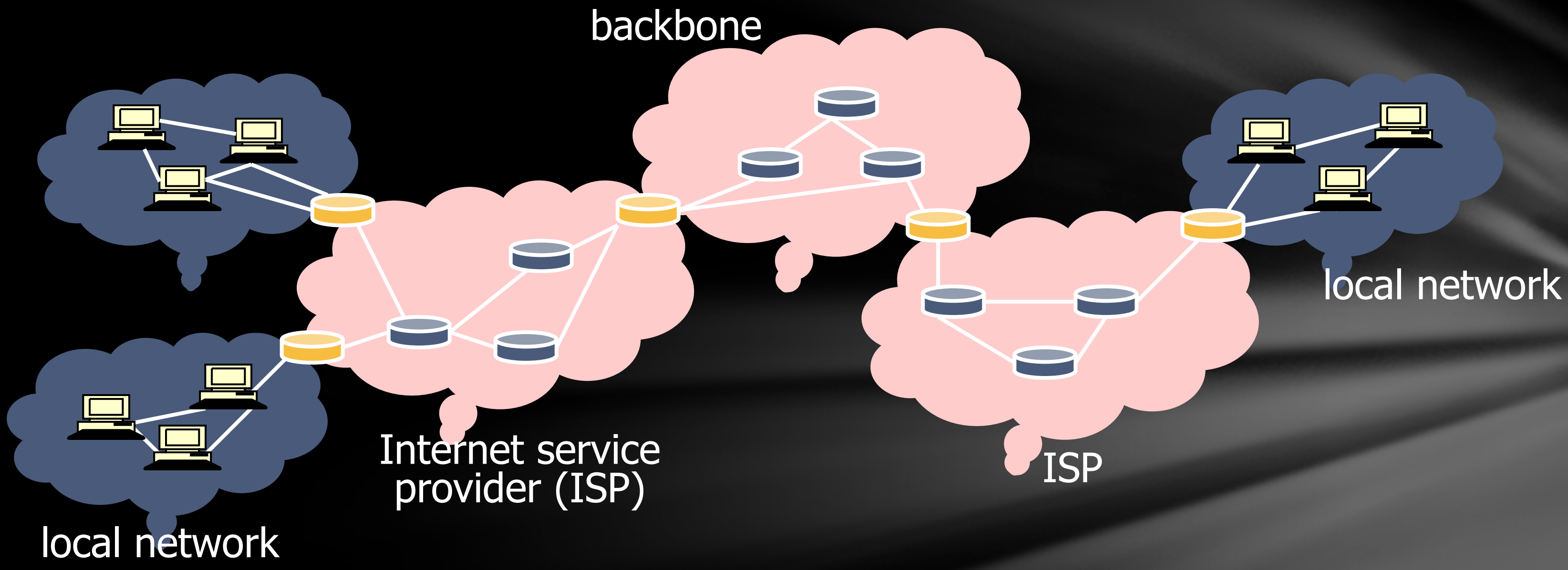
Agenda

- TCP/IP Protocol Stack
- TCP/IP Security Issues
- Internet infrastructure security protection:
  - IP level —— IPSEC
  - TCP level —— SSL/TLS

TCP/IP Protocol Stack

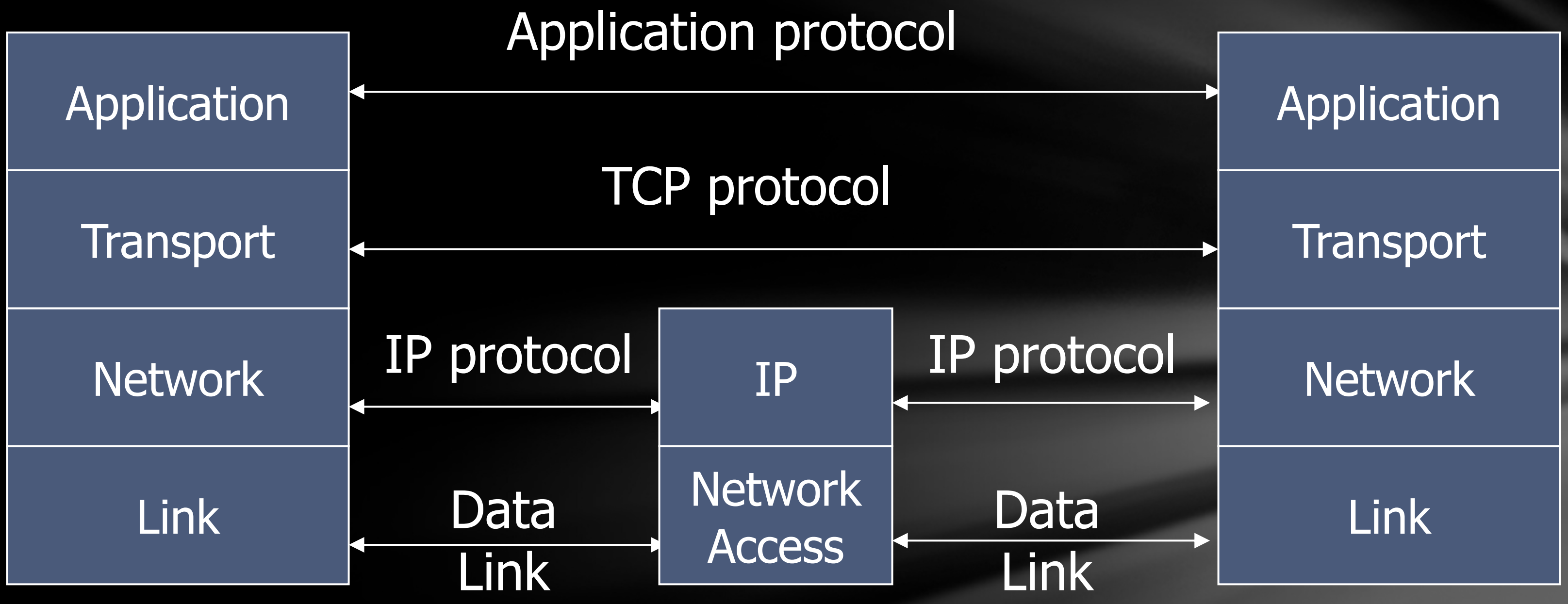
The Internet Infrastructure

Internet Structure



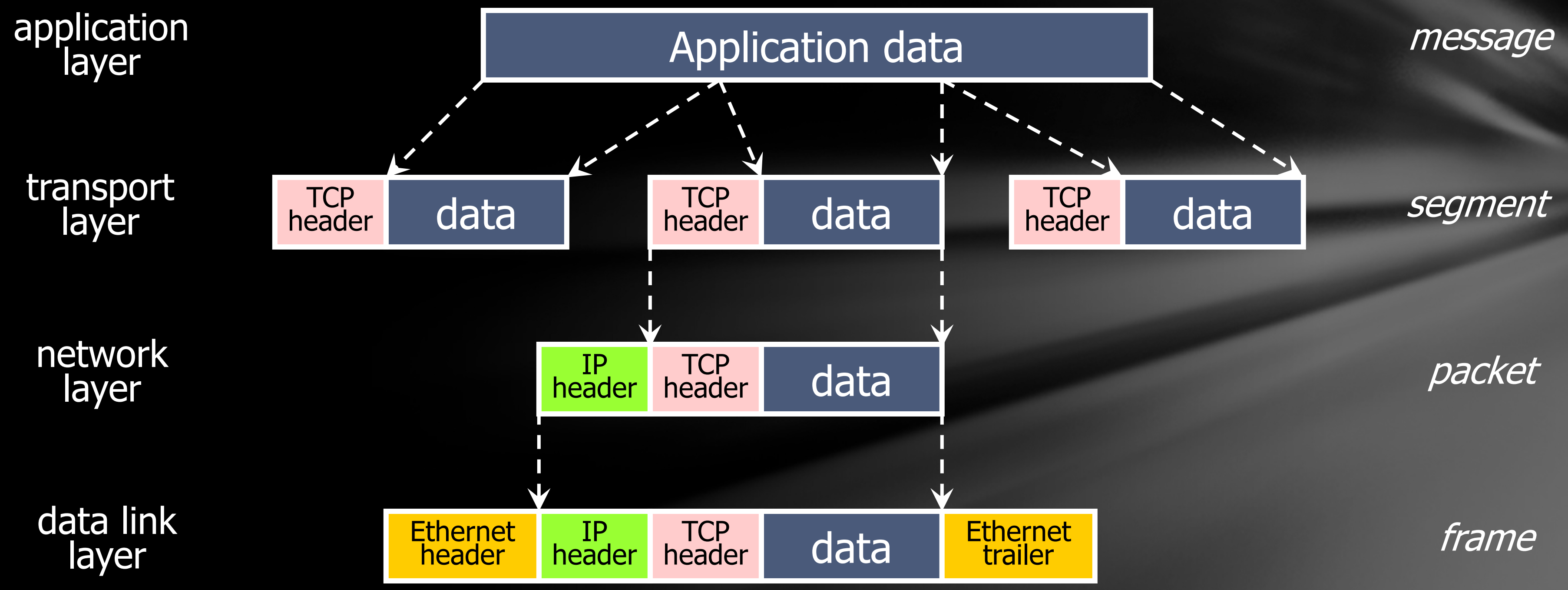
- Use TCP/IP protocol stack to route and connect
- Use BGP(Border Gateway Protocol) for router discovery
- Use DNS(Domain Name System) to find the IP address

TCP Protocol Stack



- Application layer —— HTTP, SMTP, FTP, TELNET, DNS, …
- Transport layer—— TCP, UDP
- Network layer —— IP, ICMP, BGP, OSPF, IGMP
- Data link layer —— ARP, RARP, Ethernet, HDLC, PPP

Data Format


application layer
Application data
message
transport layer
TCP header
data
TCP header
data
TCP header
data
segment
network layer
IP header
TCP header
data
packet
data link layer
Ethernet header
IP header
TCP header
data
Ethernet trailer
frame

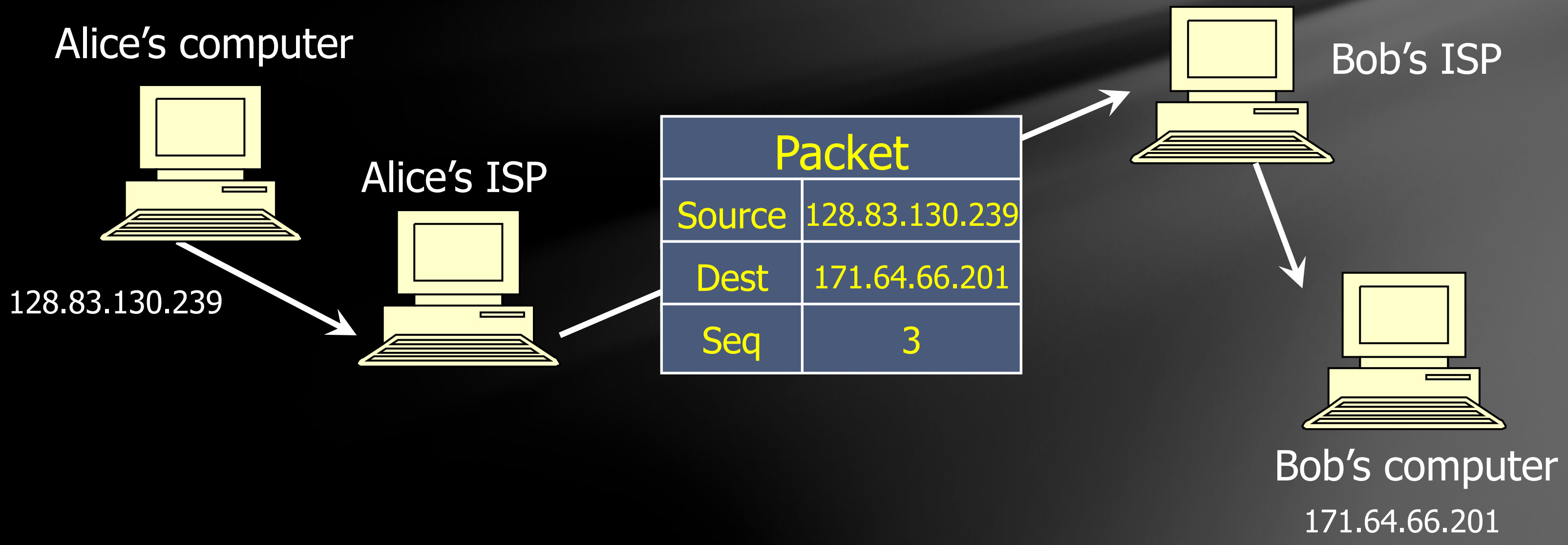
IP (Internet Protocol)

Connectionless

- Unreliable, “best-effort” protocol

It uses numeric addresses for routing

- Typically several hops in the route



Packet	
Source	128.83.130.239
Dest	171.64.66.201
Seq	3

ICMP (Control Message Protocol)

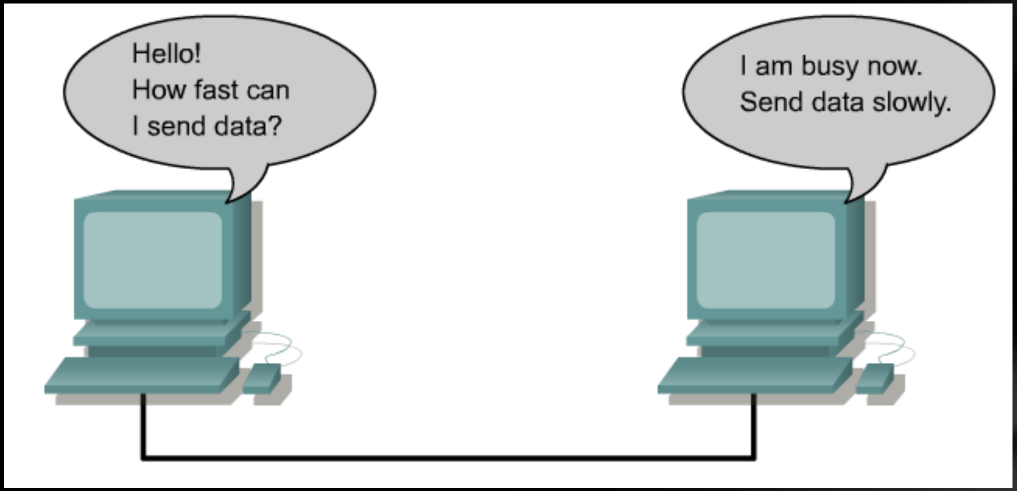
Provides feedback about network operation

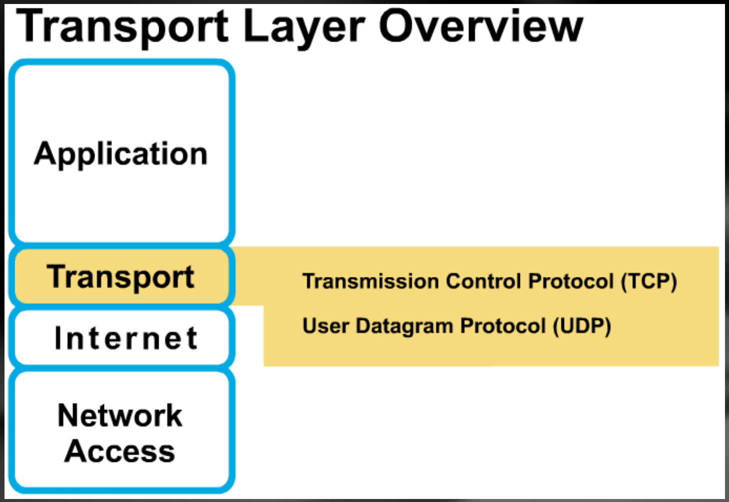
- “Out-of-band” messages carried in IP packets
- Error reporting, congestion control, reachability, etc.

Example messages:

- Destination unreachable
- Time exceeded
- Parameter problem
- Redirect to a better gateway
- Reachability test (echo/echo reply)
- Message transit delay (timestamp request/reply)

IP & TCP/UDP





IP is best-effort delivery.

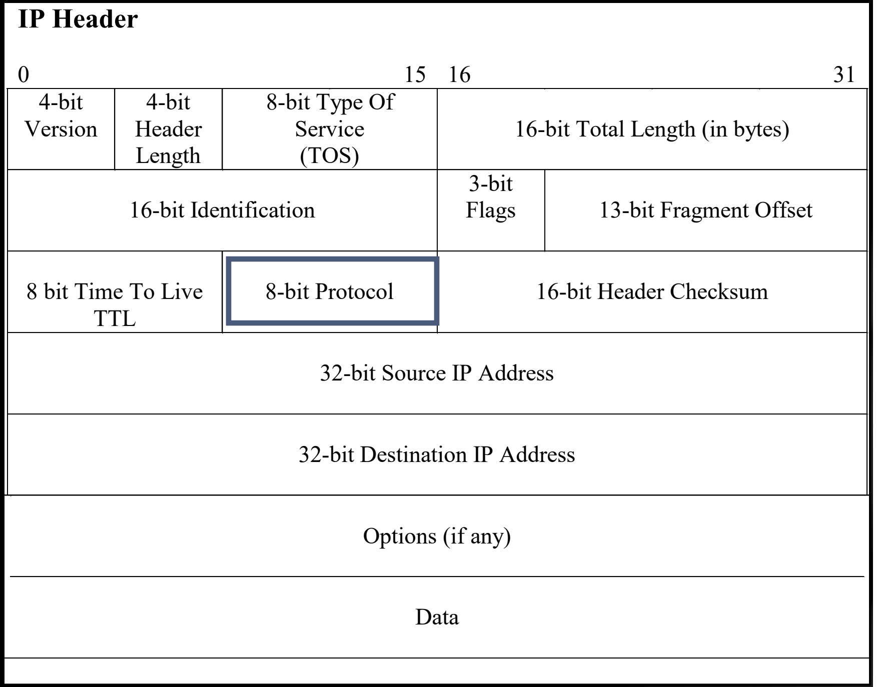
The transport layer (**TCP**) is responsible for the **reliability** and **flow control** from source to destination.

This is accomplished using the following:

- sliding windows (flow control)
- sequencing numbers and acknowledgments (reliability)
- synchronization (establish a virtual circuit)

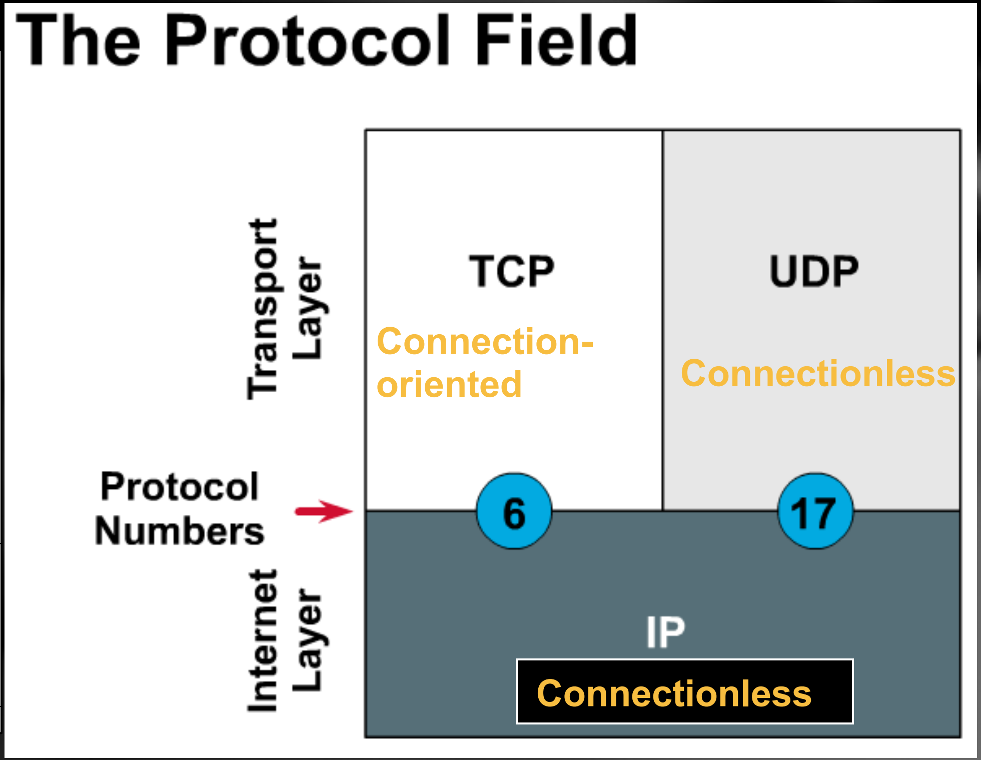
Note: *Although straightforward in its operation, TCP can be a very complicated protocol in its operation. Most of the details regarding TCP are beyond the scope of this module and presentation.*

IP & TCP/UDP



IP Header

0		15	16	31
4-bit Version	4-bit Header Length	8-bit Type Of Service (TOS)	16-bit Total Length (in bytes)	
16-bit Identification			3-bit Flags	13-bit Fragment Offset
8 bit Time To Live TTL		8-bit Protocol	16-bit Header Checksum	
32-bit Source IP Address				
32-bit Destination IP Address				
Options (if any)				
Data				



IP Packet has a Protocol field that specifies whether the segment is TCP or UDP.

User Datagram Protocol

IP provides routing

- IP address gets datagram to a specific machine

UDP separates traffic by port

- The destination port number gets the UDP datagram to the particular application process, e.g., 128.3.23.3, 53
- The source port number provides the returning destination

Minimal guarantees (… mice and elephants)

- No acknowledgment
- No flow control
- No message continuation

Transmission Control Protocol

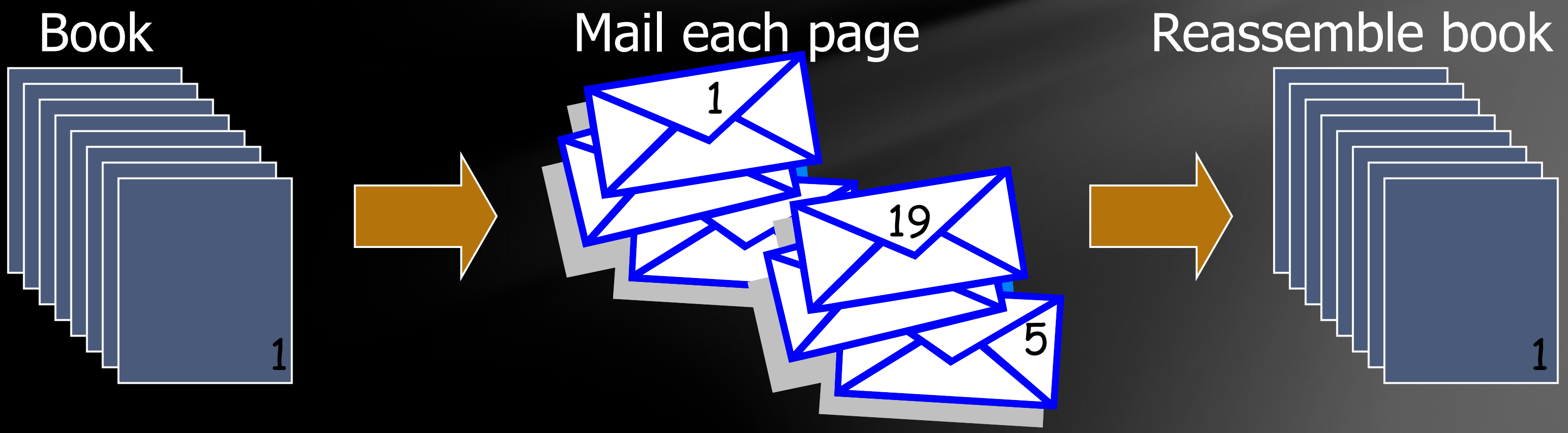
Sender: break data into segments

- The sequence number is attached to every packet

Receiver: reassemble segments in the correct order

- Acknowledge receipt; lost packets are re-sent

Connection state maintained on both sides

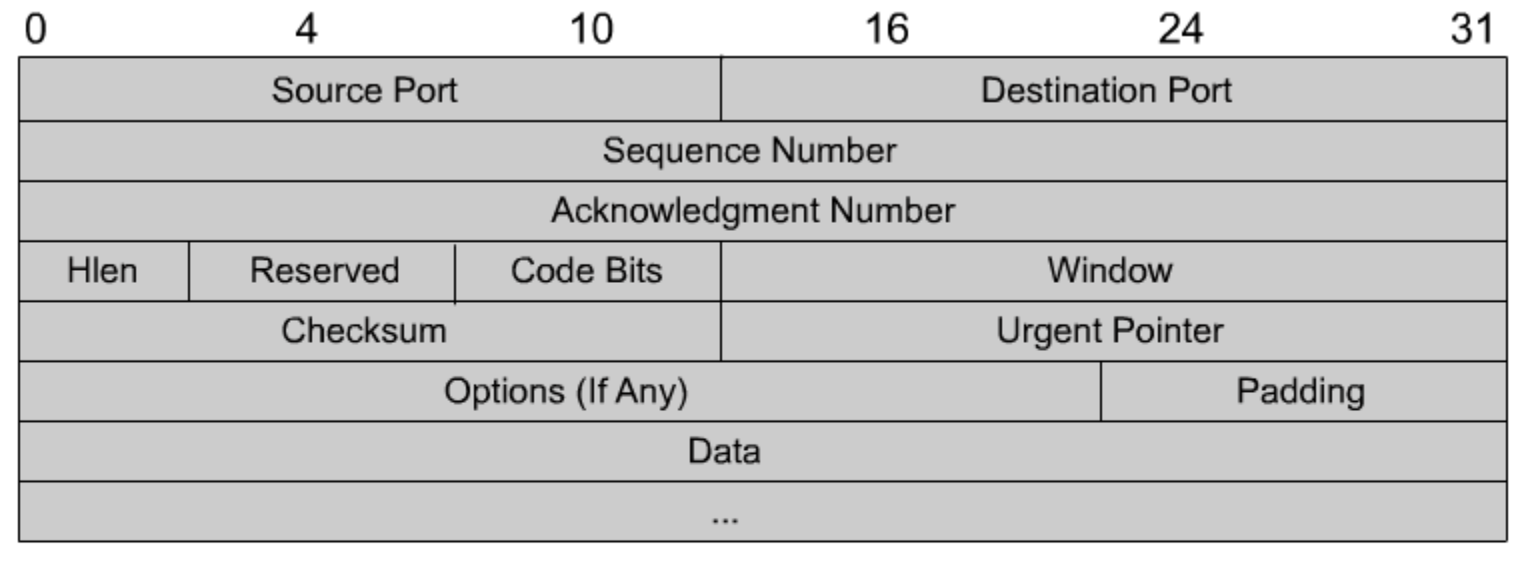


Book

Mail each page

Reassemble book

TCP



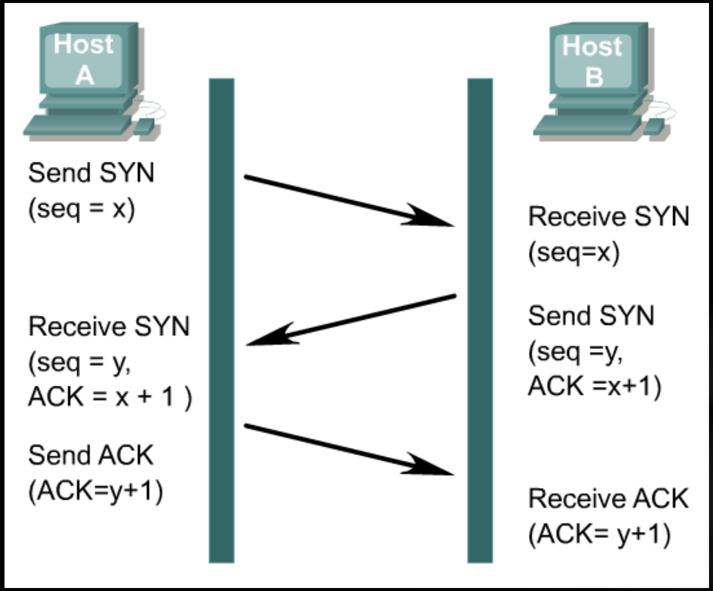
<table>
<tr><th>0</th><th>4</th><th>10</th><th>16</th><th>24</th><th>31</th></tr>
<tr><td colspan="3">Source Port</td><td colspan="3">Destination Port</td></tr>
<tr><td colspan="6">Sequence Number</td></tr>
<tr><td colspan="6">Acknowledgment Number</td></tr>
<tr><td>Hlen</td><td>Reserved</td><td>Code Bits</td><td colspan="3">Window</td></tr>
<tr><td colspan="3">Checksum</td><td colspan="3">Urgent Pointer</td></tr>
<tr><td colspan="4">Options (If Any)</td><td colspan="2">Padding</td></tr>
<tr><td colspan="6">Data</td></tr>
<tr><td colspan="6">...</td></tr>
</table>

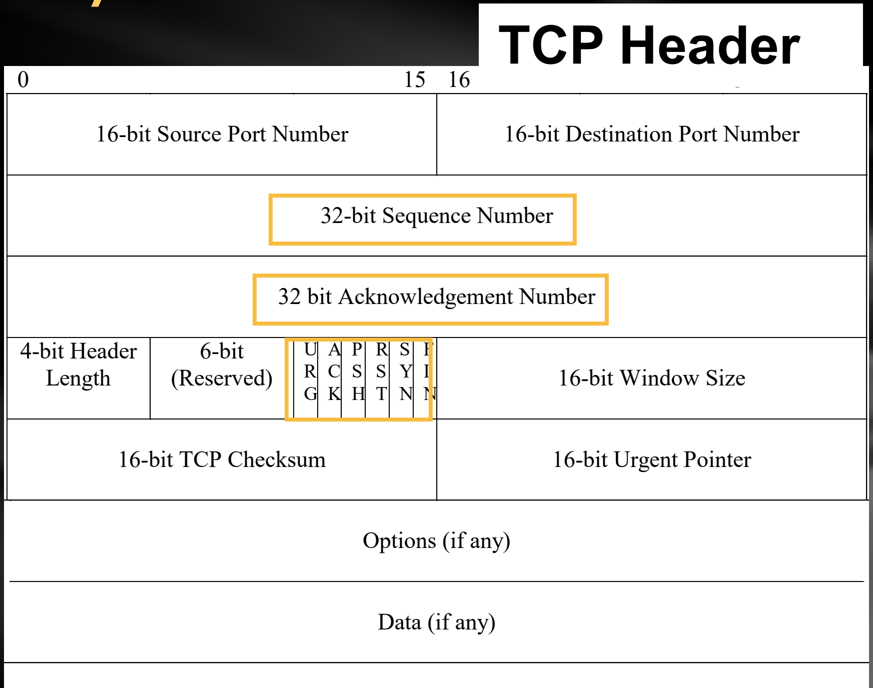
TCP -- a connection-oriented, reliable protocol; that provides **flow control** by providing sliding windows and **reliability** by providing sequence numbers and acknowledgments.

TCP re-sends anything not received and supplies a **virtual circuit** between end-user applications.

The advantage of TCP is that it provides guaranteed delivery of the segments.

Synchronization or 3-way handshake





TCP Header

0		15	16
16-bit Source Port Number			16-bit Destination Port Number
32-bit Sequence Number			
32 bit Acknowledgement Number			
4-bit Header Length	6-bit (Reserved)	URG ACK PSH RST SYN FIN	16-bit Window Size
16-bit TCP Checksum			16-bit Urgent Pointer
Options (if any)			
Data (if any)			

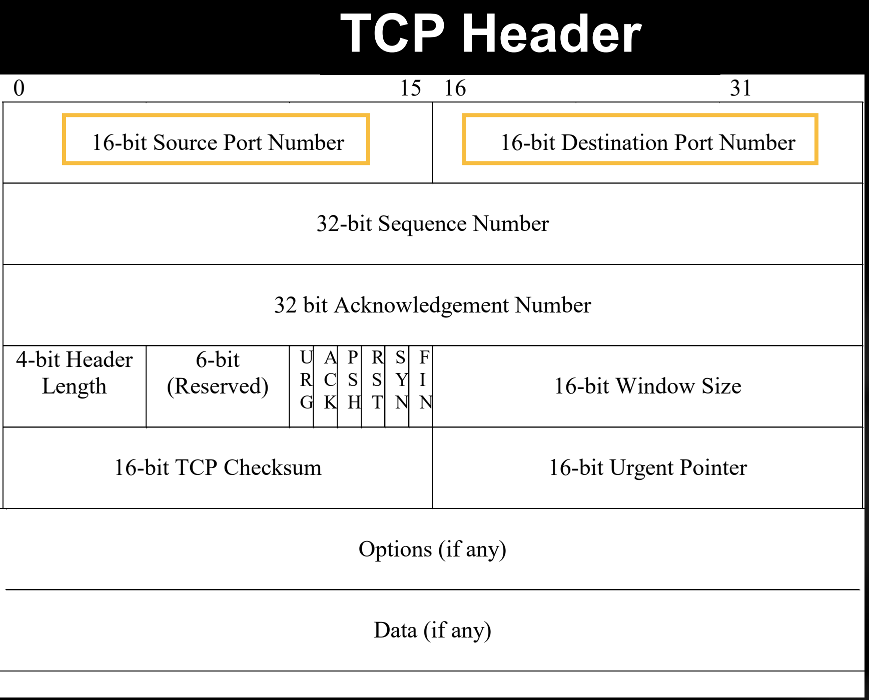
The two ends must synchronize on each other’s initial TCP sequence numbers (ISNs) to establish a connection.

Sequence numbers are used to track packets' order and ensure that no packets are lost in transmission.

The initial sequence number is the starting number used when a TCP connection is established.

Exchanging beginning sequence numbers during the connection sequence ensures that lost data can be recovered.

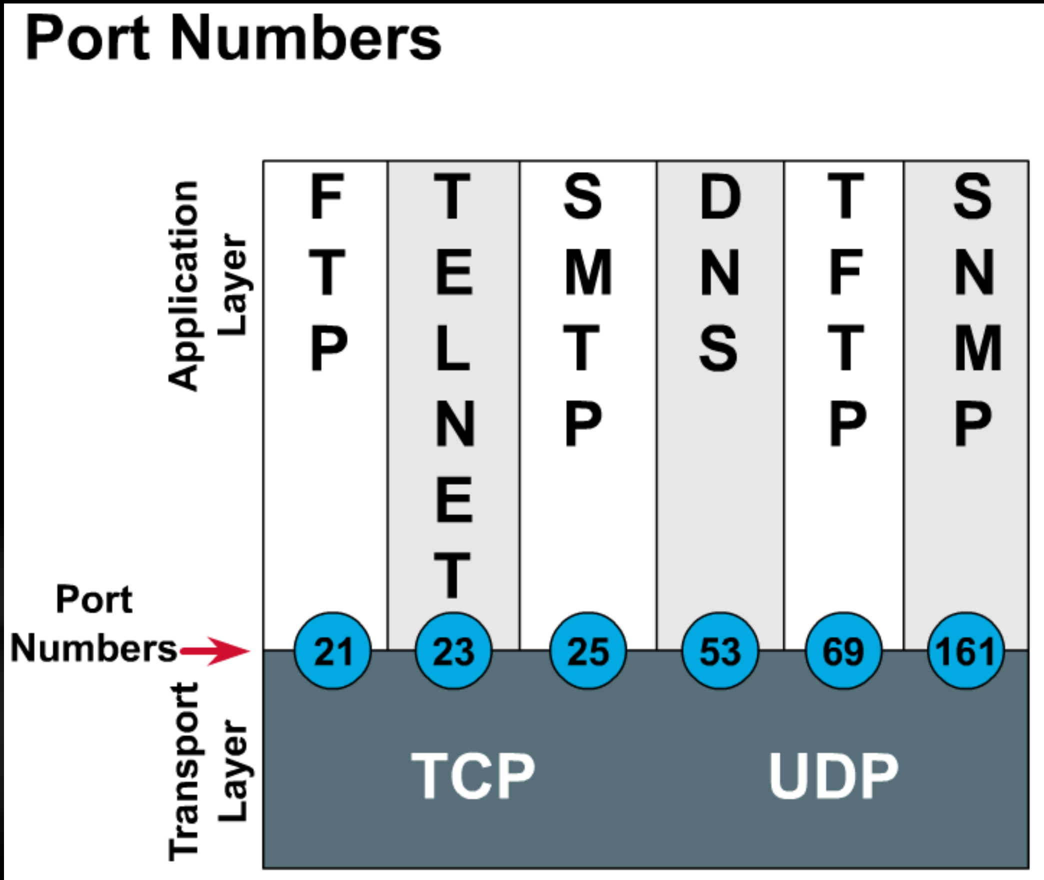
Port Numbers



TCP Header

0 15 16 31

16-bit Source Port Number			16-bit Destination Port Number	
32-bit Sequence Number				
32 bit Acknowledgement Number				
4-bit Header Length	6-bit (Reserved)	U R G A C K P S H R S T S Y N F I N	16-bit Window Size	
16-bit TCP Checksum			16-bit Urgent Pointer	
Options (if any)				
Data (if any)				



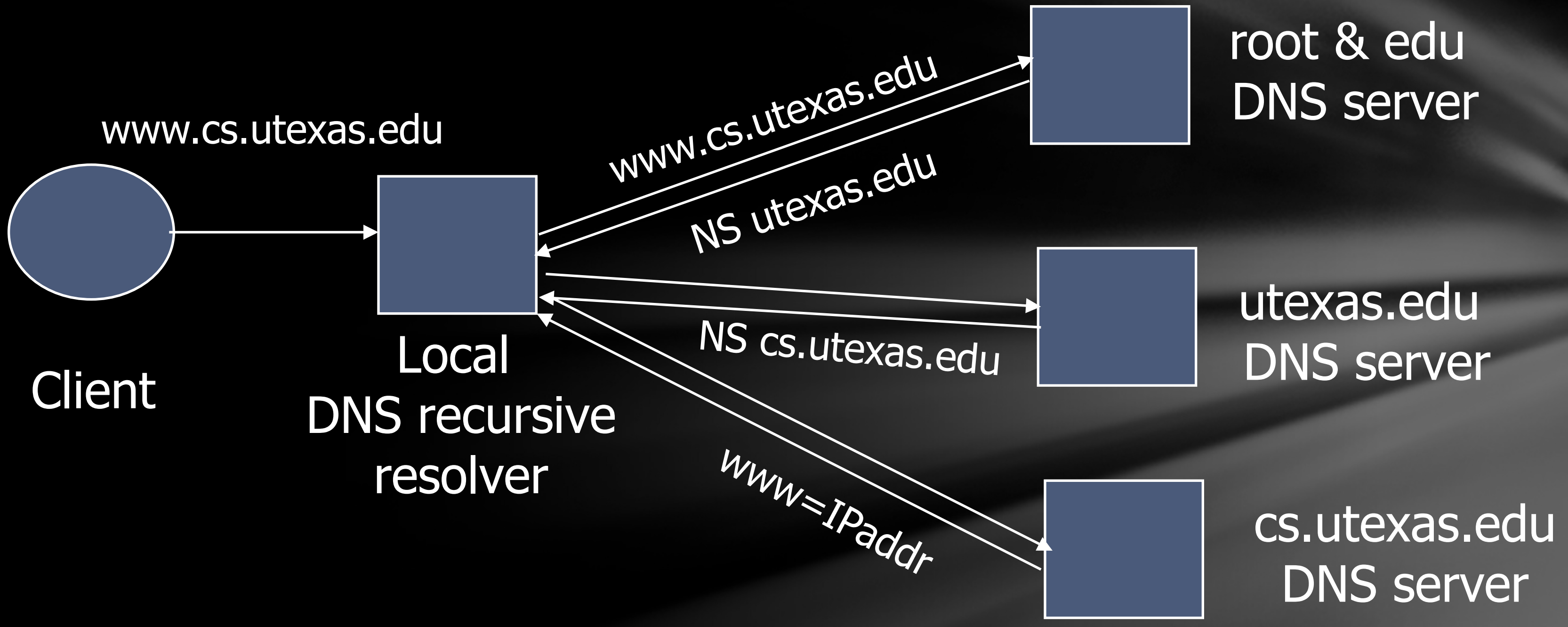
Port Numbers

Application software developers have agreed to use the **well-known port numbers** defined in RFC 1700.

For example, any conversation bound for a **Telnet** application uses the standard port number **23**.

DNS

- DNS(Domain Name Service) maps domain names to numeric IP addresses



- DNS root server, responsible for the highest level domain name resolution.
  - When the local domain name server does not know how to resolve the domain name, it will ask the authoritative server.
  - Then cascade asked up until the root server.

Security Issues of TCP/IP

Sniffing

Many transmitted data aren’t encrypted.

- FTP and Telnet send passwords in clear text
- Many web applications use HTTP protocols without encryption.

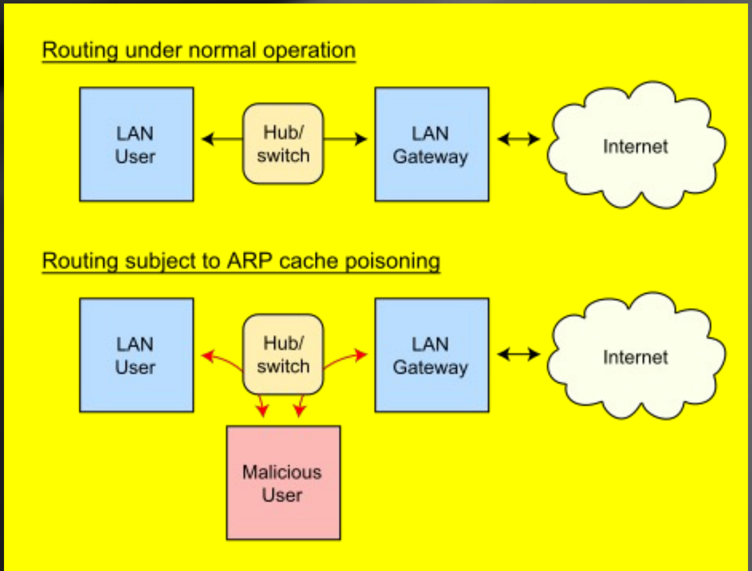
Promiscuous mode network interface card can read all data.

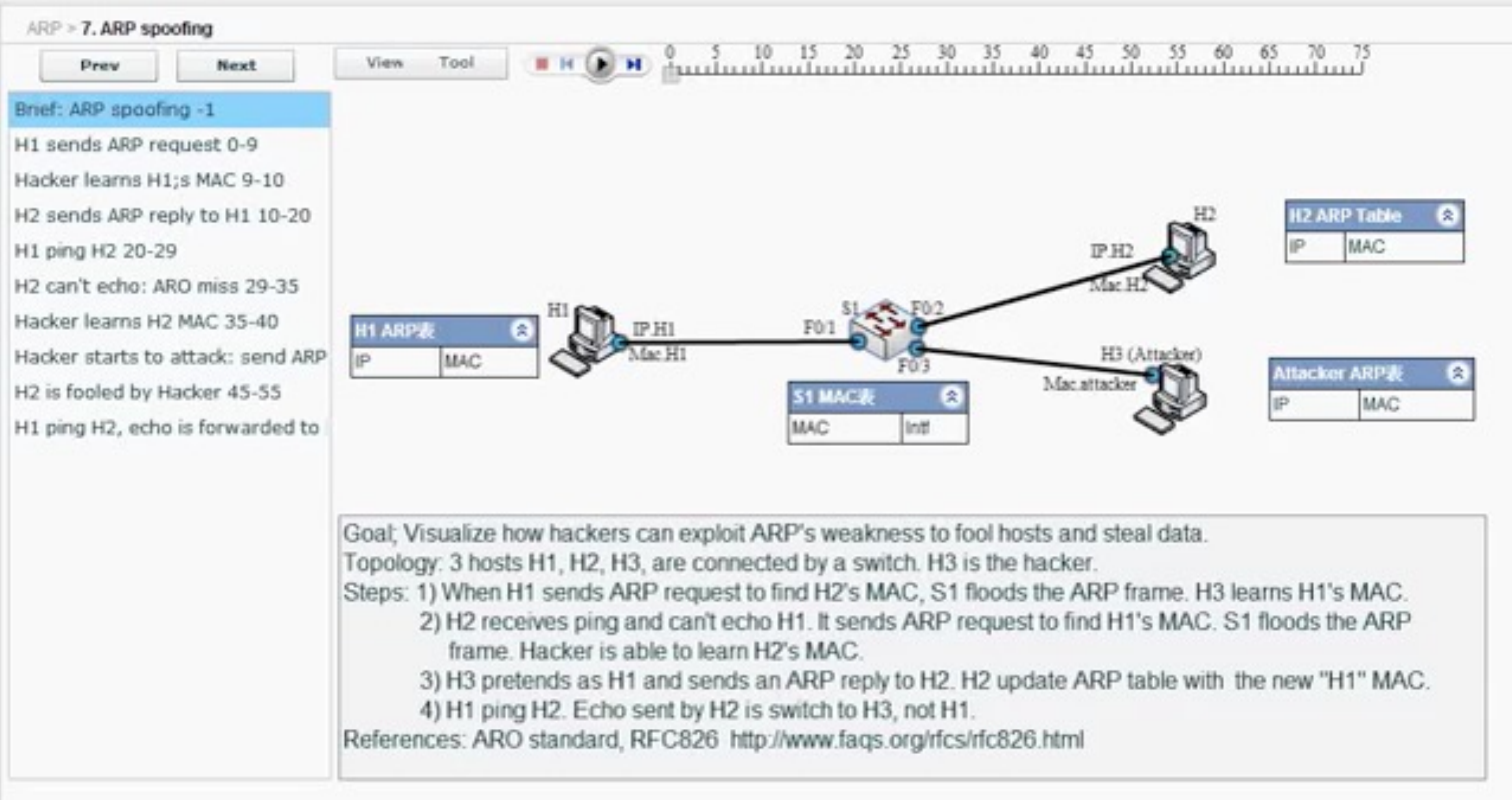
Sniffing tools:

- Tcpdump / libpcap, http: //www.tcpdump.org/
  - Open source
  - Included in most Unix/Linux distributions
- Wireshark, http: //www.wireshark.org/
  - Open-source network protocol analysis tool
  - The successor of the famous “ethereal.”
- Commview, http: //www.tamos.com/products/commview/
  - Commercial products of network security and monitoring

ARP Spoofing

- Also called ARP Poisoning
- ARP is stateless: ARP: IP -> MAC, RARP: MAC -> IP
- OS implements ARP with an ARP Cache, but the update strategies of ARP Cache are different. Some OS(Solaris, etc.) only accept the first response package.
- Forge an ICMP packet to let the victim machine initiate an ARP request. In the immediate aftermath, send a forged ARP response packet to the victim machine; the ARP cache will be poisoned.
- You can use ARP Spoofing to initiate:
  - Intercepted attack
  - Man-in-the-middle attack
  - Denial of service attacks
- Reference:
  - [http: //en.wikipedia.org/wiki/ARP_spoofing](http://en.wikipedia.org/wiki/ARP_spoofing)
  - [http: //hakipedia.com/index.php/ARP_Poisoning](http://hakipedia.com/index.php/ARP_Poisoning)





Brief: ARP spoofing -1

H1 sends ARP request 0-9

Hacker learns H1;s MAC 9-10

H2 sends ARP reply to H1 10-20

H1 ping H2 20-29

H2 can't echo: ARO miss 29-35

Hacker learns H2 MAC 35-40

Hacker starts to attack: send ARP

H2 is fooled by Hacker 45-55

H1 ping H2, echo is forwarded to

Goal; Visualize how hackers can exploit ARP's weakness to fool hosts and steal data.

Topology: 3 hosts H1, H2, H3, are connected by a switch. H3 is the hacker.

Steps: 1) When H1 sends ARP request to find H2's MAC, S1 floods the ARP frame. H3 learns H1's MAC.

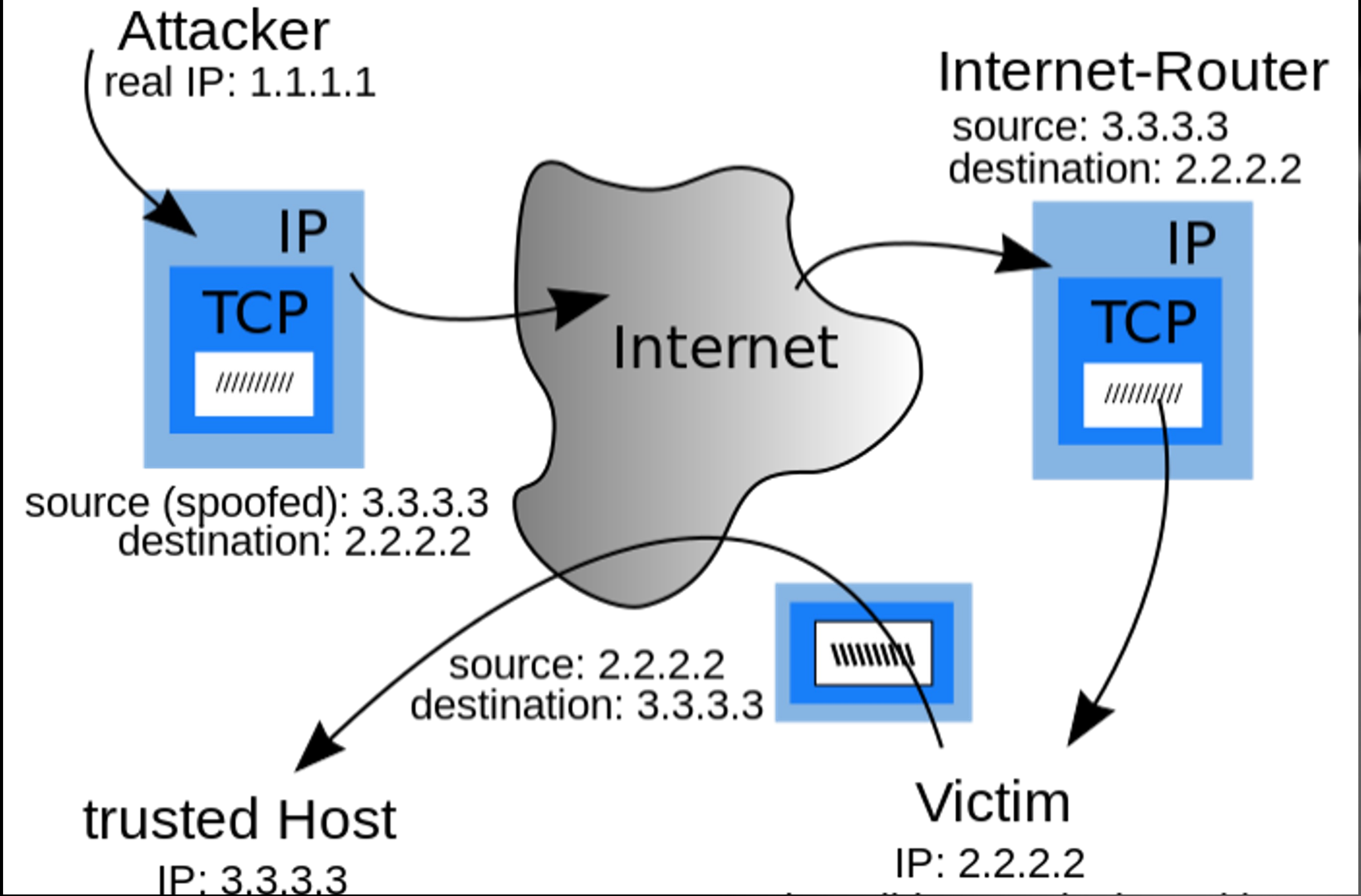
2) H2 receives ping and can't echo H1. It sends ARP request to find H1's MAC. S1 floods the ARP frame. Hacker is able to learn H2's MAC.

3) H3 pretends as H1 and sends an ARP reply to H2. H2 update ARP table with the new "H1" MAC.

4) H1 ping H2. Echo sent by H2 is switch to H3, not H1.

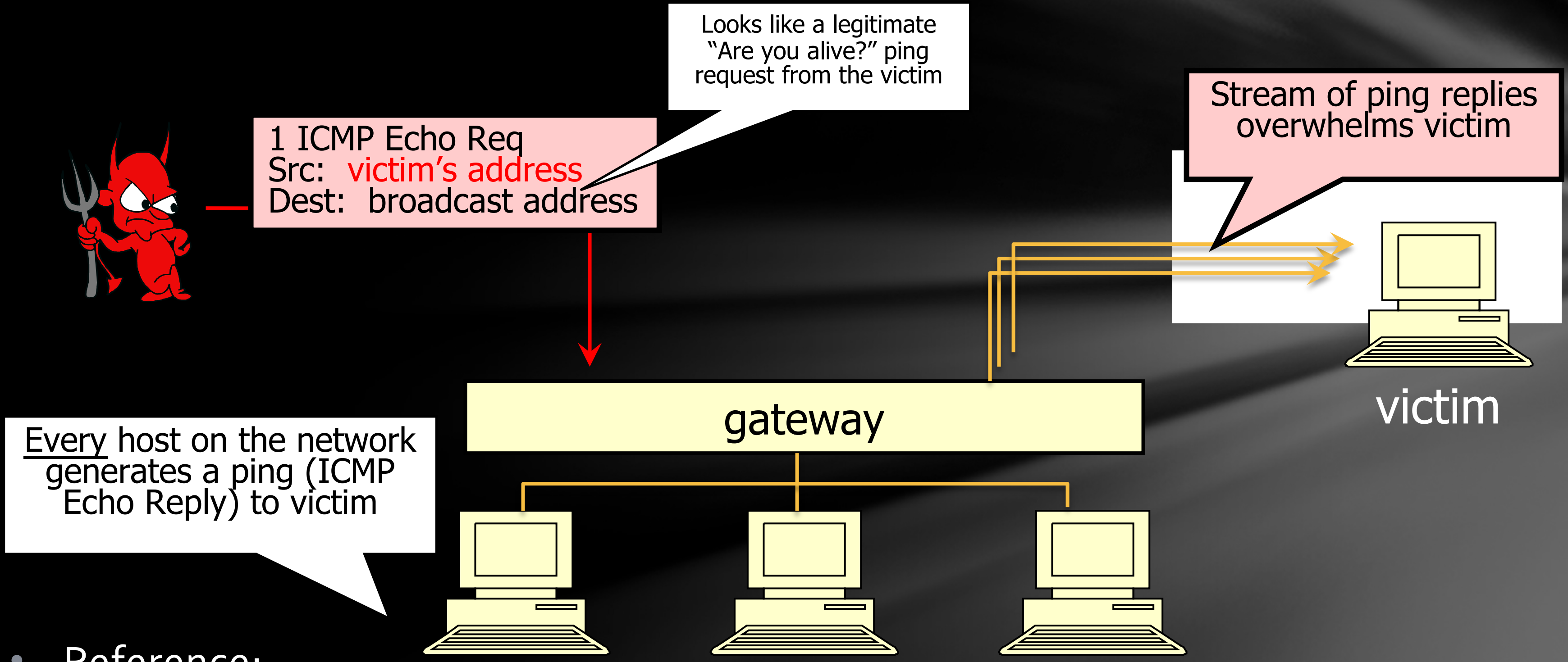
References: ARO standard, RFC826 http://www.faqs.org/rfcs/rfc826.html

IP Spoofing


Attacker
real IP: 1.1.1.1
IP
TCP
//////////
source (spoofed): 3.3.3.3
destination: 2.2.2.2
Internet
Internet-Router
source: 3.3.3.3
destination: 2.2.2.2
IP
TCP
//////////
source: 2.2.2.2
destination: 3.3.3.3
\\\\\\\\\\
trusted Host
IP: 3.3.3.3
Victim
IP: 2.2.2.2

IP Spoofing

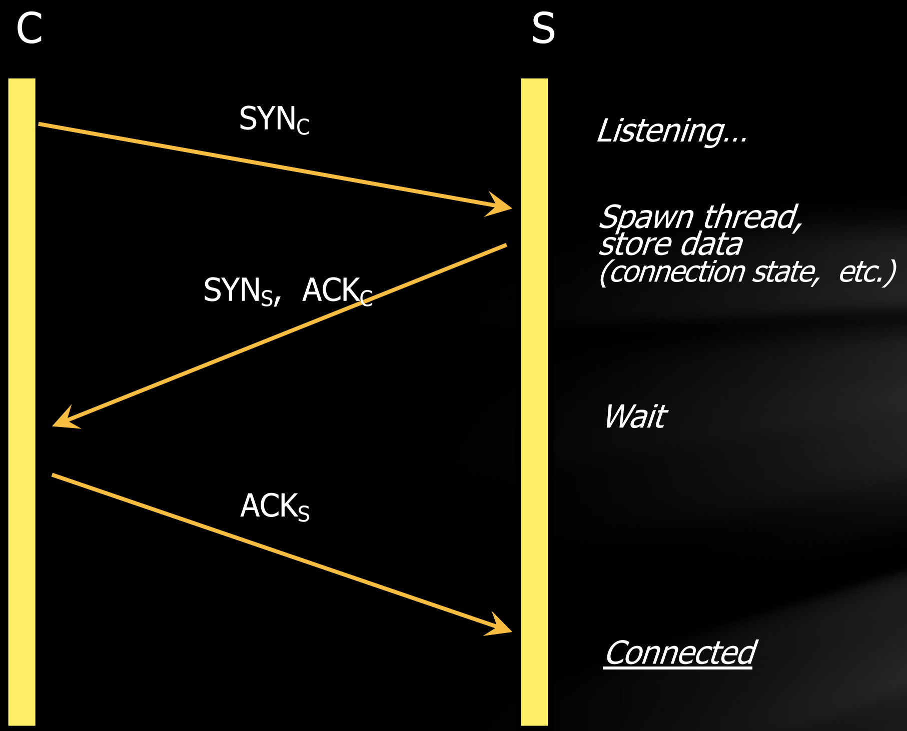
- Also called IP Smurf （Named after the program smurf who first do this attack）


Looks like a legitimate "Are you alive?" ping request from the victim
1 ICMP Echo Req
Src: victim's address
Dest: broadcast address
Stream of ping replies overwhelms victim
gateway
victim
Every host on the network generates a ping (ICMP Echo Reply) to victim

- Reference:
  - http: //en.wikipedia.org/wiki/IP_address_spoofing
  - http: //www.sans.org/reading_room/whitepapers/threats/introduction-ip-spoofing_959

TCP SYN Flooding

TCP three-way handshake



C

S

SYN$_C$

Listening…

Spawn thread,
store data
(connection state, etc.)

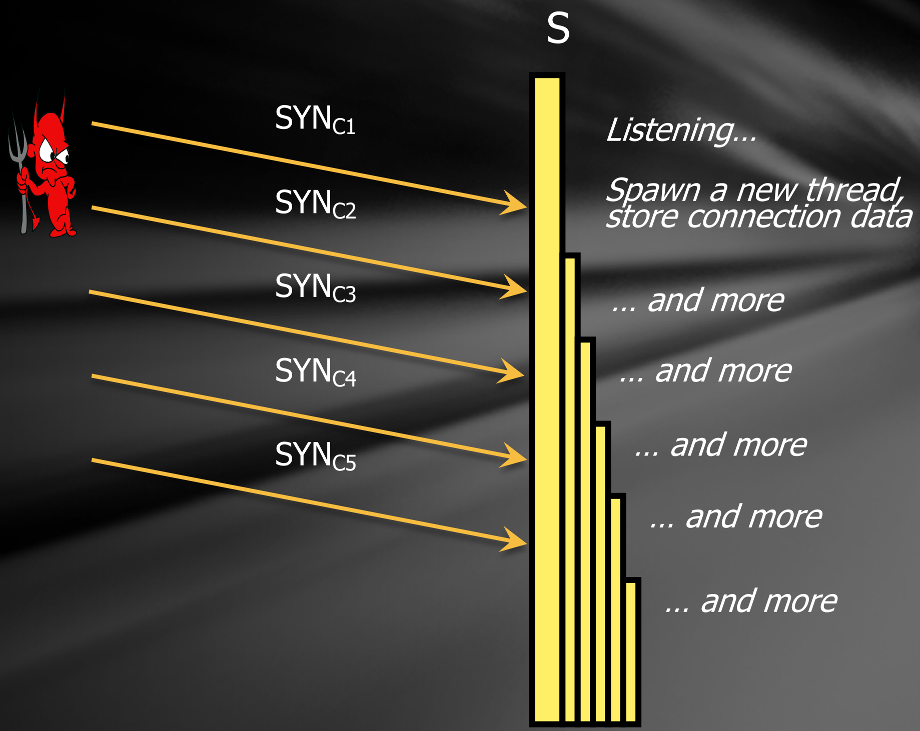
SYN$_S$, ACK$_C$

Wait

ACK$_S$

__Connected__

What if?



S

SYN$_{C1}$

SYN$_{C2}$

SYN$_{C3}$

SYN$_{C4}$

SYN$_{C5}$

Listening…

Spawn a new thread,
store connection data

… and more

… and more

… and more

… and more

… and more

TCP SYN Flooding

Principle of SYN Flooding:

- Attackers send numerous requests while forging IP address
- Attacked host allocate resource for each request
  - New threads, new memory for connection state until timeout
- Once the resource is exhausted, the client can not properly connect

Most classic DOS attacks:

- The initiator does not consume resources; however, the recipient must create a thread for each request.
- **Asymmetry!**

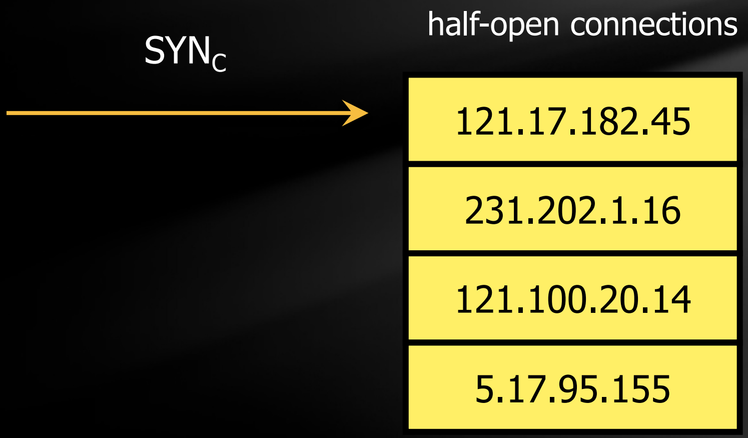
History of SYN Flooding

- TCP SYN flooding was discovered by Bill Cheswick and Steve Bellovin in 1994, and an implementation of it was published in their book "Firewalls and Internet Security: Repelling the Wily Hacker." Unfortunately, there was no countermeasure in the next two years.
- In 1996, a famous online security magazine "Phrack Magazine" published an article about the detail of SYN flooding with an attack tool. It was widely spread ……
- Until Sep. 1996, all ICPs in the U.S. were flooded repeatedly and again and again …

TCP SYN Flooding

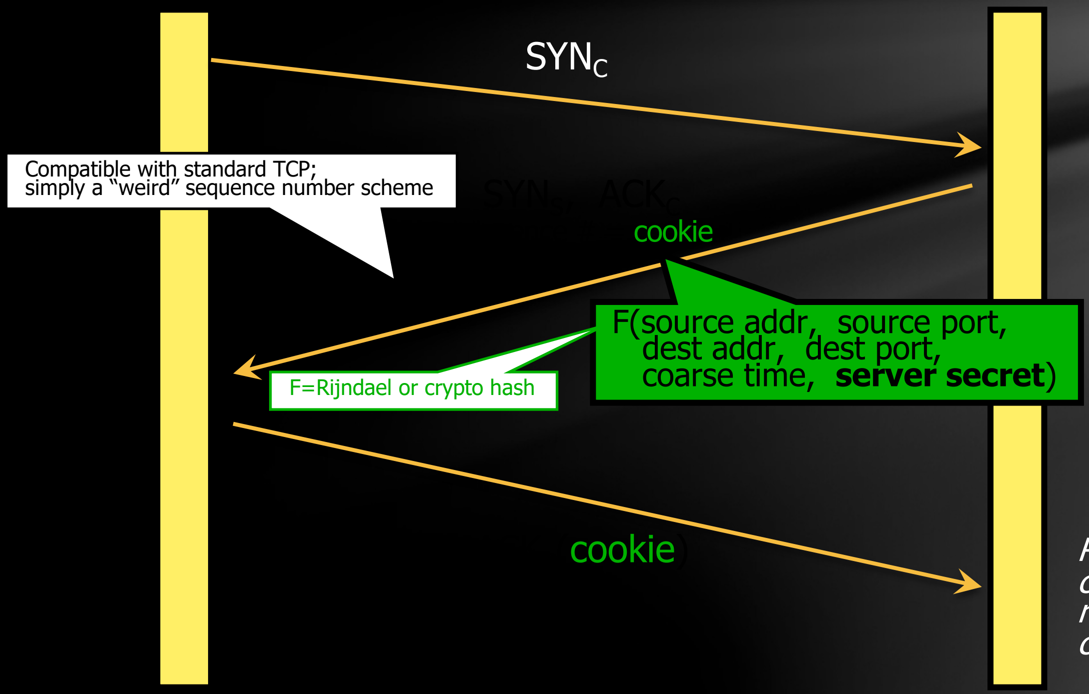
How to prevent it??

- **Ways 1: random delete** —— If the SYN queue is full, randomly delete one.
  - Normal connections can be completed, and flooding connections will eventually be deleted.
  - Easy to implement!!



TCP SYN Flooding

- **Ways 2: SYN Cookies**
  - Why can SYN Flooding succeed？ **Asymmetric resource allocation**！
  - SYN Cookies ensure that the server will not store the states unless it receives at least two messages from the client.
    - The server will store the socket information (IP and port of the server and the client) in a cookie and send the cookie to the client.
    - The client must send the cookie along with the 2nd request, and the server will re-compute the cookie and compares it with the cookie sent by the client.



Compatible with standard TCP;
simply a "weird" sequence number scheme

F(source addr, source port,
dest addr, dest port,
coarse time, **server secret**)

F=Rijndael or crypto hash

Listening…

Does <u>not</u> store state

Cookie must be unforgeable
and tamper-proof (why?)
Client should not be able
to invert a cookie (why?)

Recompute cookie,
compare with with the one
received, only establish
connection if they match

TCP SYN Flooding

- Implementation of SYN Cookies
  - SYN Cookies are included in the standard TCP/IP protocol stack implementation of Linux and FreeBSD.
  - The syn-ack cookies option is not enabled by default in legacy Linux versions. System administrators must add the following line in the boot script :
    - echo 1 > /proc/sys/net/ipv4/tcp_syncookies
- Reference:
  - http: //en.wikipedia.org/wiki/SYN_flood
  - In 1996, CERT Advisory on TCP SYN Flooding“CERT® Advisory CA-1996-21 TCP SYN Flooding and IP Spoofing Attacks”
    - http: //www.cert.org/advisories/CA-1996-21.html
  - In 2007, Request For Comment on SYN Flooding and solutions, RFC 4987: “TCP SYN Flooding Attacks and Common Mitigations.”
    - http://tools.ietf.org/html/rfc4987
  - SYN Cookies: http: //cr.yp.to/syncookies.html

TCP SYN Prediction Attack

- **TCP sequence prediction attack**
  - Every packet in the TCP protocol has a sequence number SYN; the receiver will sort and reorganize packets according to the SYN.
  - Once the attacker can predict the SYN, he can send “forged” packets to the receiver and make it reorganize packets to serve the attacker’s purpose.
  - Morris, R., "A Weakness in the 4.2BSD UNIX TCP/IP Software", CSTR 117, AT&T Bell Laboratories, Murray Hill, NJ, 1985.
- TCP SYN prediction attack is the source of many other attacks, including:
  - TCP spoofing
  - TCP connection hijacking
  - TCP reset

TCP SYN Prediction Attack

- The method against TCP SYN prediction attack is "to select a random initial SYN(ISN) to make prediction impossible."
  - In TCP original protocol(RFC0793), a global 32-bit ISN generator is recommended, and it increases 1 per every 4 microseconds.
    - Easy to be predicted & Easy to be attacked.
  - RFC6528 suggests: the ISN should be chosen under the rules below:
    - ISN = M + F(localip, localport, remoteip, remoteport, secretkey)
    - M is a 4 microseconds timer, F is a pseudo-random numbers function, and a one-way hash function is suggested for the implementation.
- Reference:
  - http: //en.wikipedia.org/wiki/TCP_sequence_prediction_attack
  - http: //tools.ietf.org/html/rfc6528

TCP Congestion Control

- **TCP Congestion Control Protocol**
  - On packet loss, which means the network is congested, TCP protocol requires the sender:
    - Half down the speed, and continue halving until there is no packet loss or the speed is 0
    - If the packet loss stops, the transmission speed will increase slowly
- Attack scenario: Alice is a good user, while Bob is a malicious user
  - Alice and Bob suffer packet loss at the same time
  - Alice lowers its speed; however, Bob violates the protocol and achieves better speed
- Solution: Add ack nonces, and return nonce at ack to prove it is not a cheat.
- Reference: Stefan Savage et al. TCP Congestion Control with a Misbehaving Receiver. http://cseweb.ucsd.edu/~savage/papers/CCR99.pdf

DNS Spoofing

- DNS Spoofing:
  - Modify the DNS server or the local DNS service (often modify the DNS cache database to make it a DNS cache poisoning attack), redirect the page required to a wrong IP, which sends the traffic to another server (often the attacker's machine).
- Pharming:
  - Basis of Pharming (pharming phishing)

Reference:

http: //www.checkpoint.com/defense/advisories/public/dnsvideo/index.html

- http: //www.unixwiz.net/techtips/iguide-kaminsky-dns-vuln.html
- http: //compsec101.antibozo.net/papers/dnssec/dnssec.html

DNS Cache Poisoning Attack

Check Point®
SOFTWARE TECHNOLOGIES LTD.

Against DNS Spoofing – DNSSEC

- Why does DNS cache poisoning happen?
  - DNS requests and responses are not authenticated!
    - The attacker provides faked DNS information.
- DNSSEC （Domain Name System Security Extensions） was designed to confront the DNS Spoofing
  - All response of DNSSEC is authenticated !
  - DNSSEC neither provide encryption service nor is used in confronting DoS attack.
  - DNSSEC was described in RFC4033, RFC4034, and RFC4035.
- Reference:
  - http: //en.wikipedia.org/wiki/DNSSEC
  - http: //en.wikipedia.org/wiki/TSIG

IPSEC

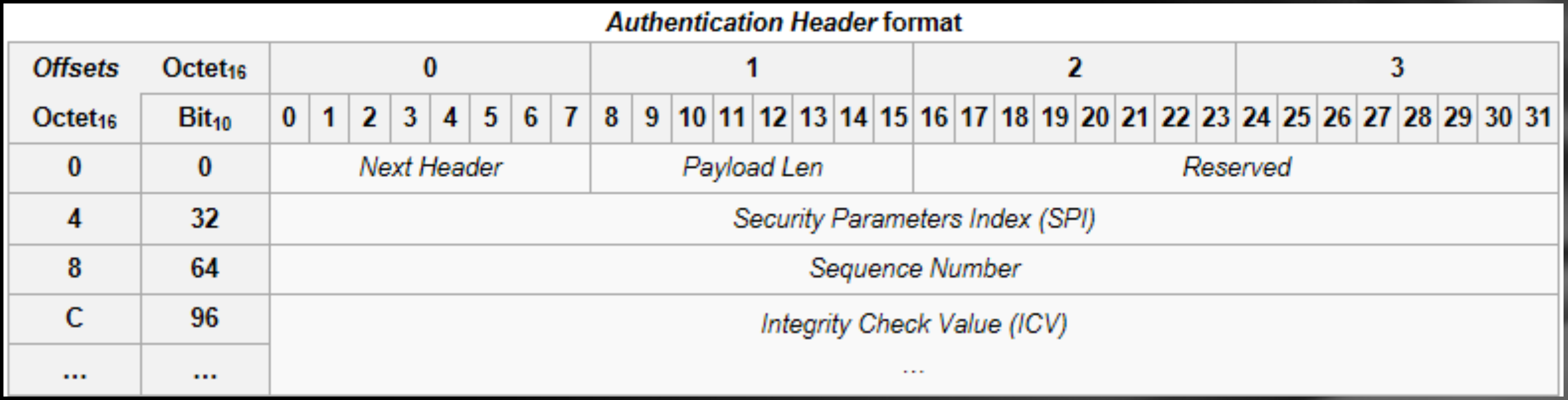
IP layer security mechanisms

IPSec Overview

- IPSec is to support the encryption and authentication of all network traffic in the IP layer. There are four most important standard documents:
  - RFC 2401: Security Architecture for IP
  - RFC 2402: IP Authentication Header （IPv4 and IPv6）
  - RFC 2406: IP Encapsulating Security Payload （IPv4 and IPv6）
  - RFC 2408: Internet Security Association and Key Management Protocol (ISAKMP)
  - IPv6 must support IPSEC, while IPv4 optionally supports it.
- Three core components of IPSEC:
  - Authentication Headers, AH / 验证头
  - Encapsulating Security Payloads, ESP / 载荷安全性封装
  - Security Associations, SA / 安全关联

IPSec Concepts —— AH

- provide data integrity and authentication service for IP packet
  - Optionally provide anti-replay



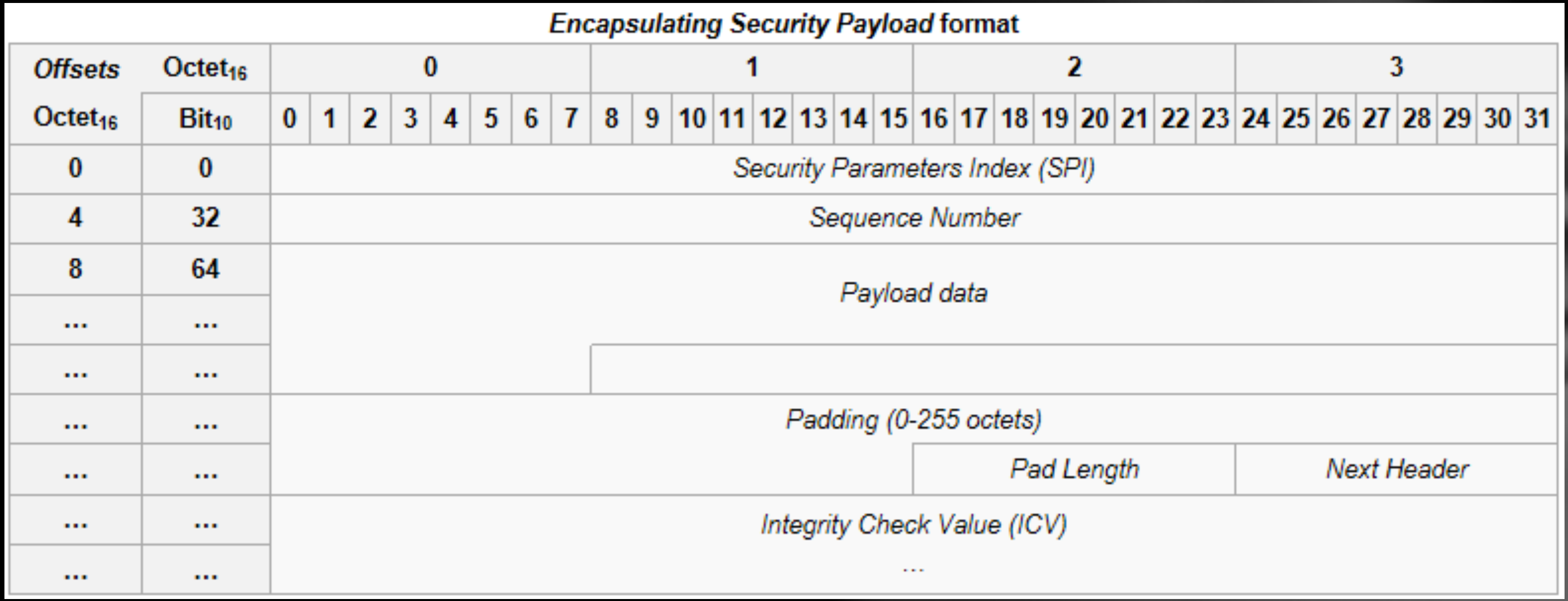
Authentication Header format

Offsets	Octet$_{16}$	0								1								2								3							
Octet$_{16}$	Bit$_{10}$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	*Next Header*								*Payload Len*								*Reserved*															
4	32	*Security Parameters Index (SPI)*																															
8	64	*Sequence Number*																															
C	96	*Integrity Check Value (ICV)*																															
…	…	…																															

- Next Header(8bit): next header type following the AH
- Payload Len (8bit): length of AH in 32-bit minus 2.
  - For an AH data length = 96 bits (3 words), plus 3 words fixed head.
  - , the payload length field will be 4.
- Reserved (16bit): backup.
- Security Parameters Index (32bit): index of the SA related to these IP packets
- Sequence Number (32bit): a monotonically increasing counter to prevent replay attacks.
- Integrity Check Value, ICV: Contains the MAC or integrity check value (ICV).

IPSec Concepts —— ESP

- Provide security, confidentiality, and authentication services (optional)



Encapsulating Security Payload format

<table>
<tr><th>Offsets</th><th>Octet$_{16}$</th><th colspan="8">0</th><th colspan="8">1</th><th colspan="8">2</th><th colspan="8">3</th></tr>
<tr><th>Octet$_{16}$</th><th>Bit$_{10}$</th><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>10</th><th>11</th><th>12</th><th>13</th><th>14</th><th>15</th><th>16</th><th>17</th><th>18</th><th>19</th><th>20</th><th>21</th><th>22</th><th>23</th><th>24</th><th>25</th><th>26</th><th>27</th><th>28</th><th>29</th><th>30</th><th>31</th></tr>
<tr><td>0</td><td>0</td><td colspan="32"><i>Security Parameters Index (SPI)</i></td></tr>
<tr><td>4</td><td>32</td><td colspan="32"><i>Sequence Number</i></td></tr>
<tr><td>8</td><td>64</td><td colspan="32" rowspan="2"><i>Payload data</i></td></tr>
<tr><td>...</td><td>...</td></tr>
<tr><td>...</td><td>...</td><td colspan="8"></td><td colspan="24"></td></tr>
<tr><td>...</td><td>...</td><td colspan="32"><i>Padding (0-255 octets)</i></td></tr>
<tr><td>...</td><td>...</td><td colspan="16"></td><td colspan="8"><i>Pad Length</i></td><td colspan="8"><i>Next Header</i></td></tr>
<tr><td>...</td><td>...</td><td colspan="32" rowspan="2"><i>Integrity Check Value (ICV)</i>
...</td></tr>
<tr><td>...</td><td>...</td></tr>
</table>

- Security Parameters Index, SPI (32bit): index of the SA related to the packets
- Sequence Number(32bit): a monotonically increasing counter to prevent replay attacks.
- Payload data: original IP package
- Padding(0-255bit) / Pad Length(8bit): Information about padding
- Next Header (8bit): next header type description following the ESP
- Integrity Check Value, ICV: Contains the integrity check value (ICV)

IPSec Concepts —— SA

- IPSec uses Security Association(SA) to integrate security services
- SA defines a series of algorithms and parameters(key, etc.) for a one-way sender-recipient flow for encryption and authentication.
  - If you need secure two-way communication, you need to create two SA.
- Three parameters uniquely determine an SA:
  - Security parameter index (SPI): a bit string associated with the SA
  - IP destination address: SA's destination address
  - Security protocol identifier: Specify the AH or ESP

In a nutshell, an SA is a logical group of security parameters enabling information sharing with another entity.

IPSec Concepts —— SA

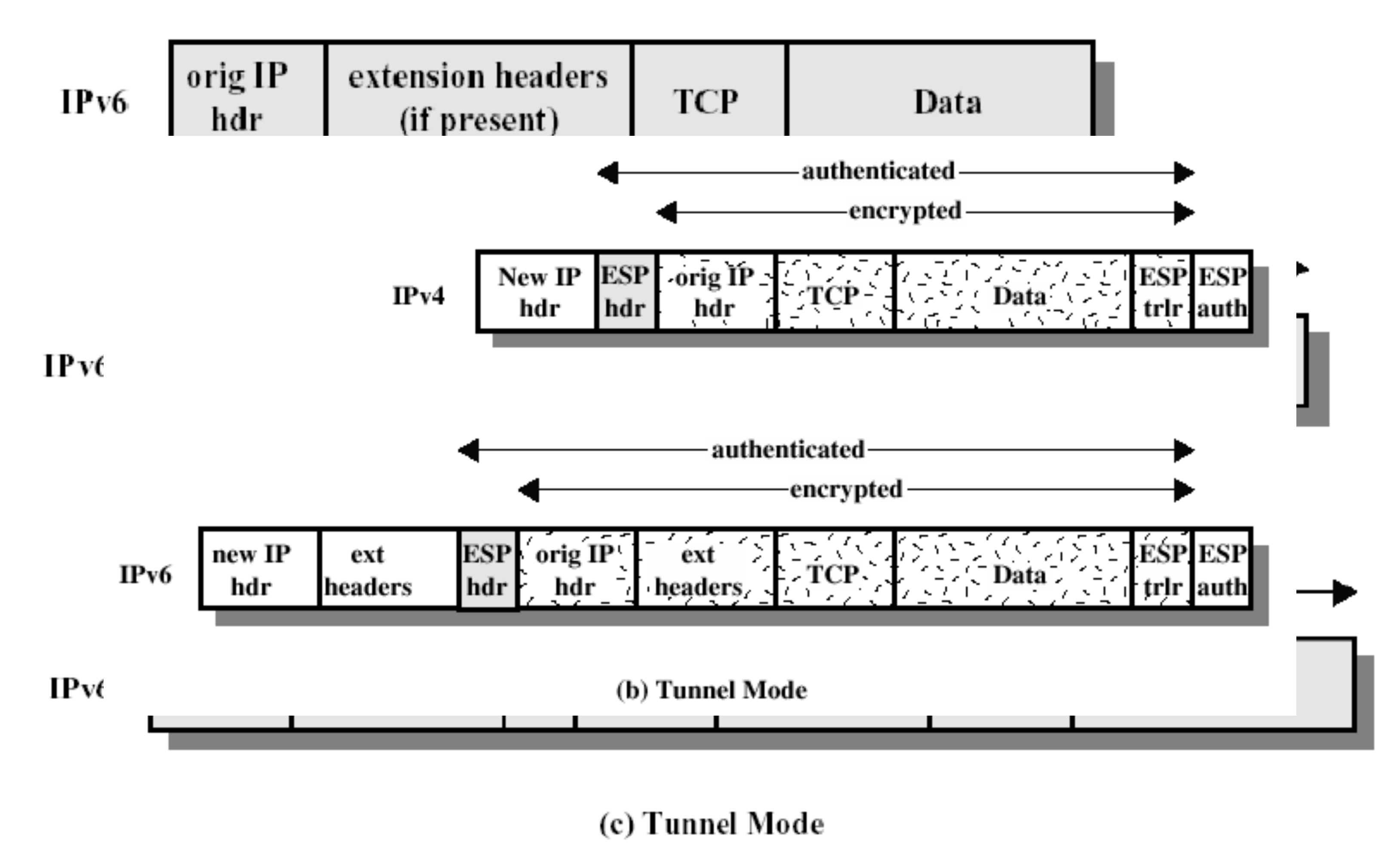
- SA uses Internet Security Association and Key Management Protocol, ISAKMP, defined in RFC2408, to exchange keys.
    - Provide the mechanism for building the Security Association, and define the key exchange frame, which is not included in the key exchange protocol.
    - The protocols for key exchange:
        - Internet Key Exchange （http: //en.wikipedia.org/wiki/Internet_key_exchange）
            - RFC2409 is the most widely used.
            - Oakley key determination protocol: based on optimized Diffie-Hellman algorithm
        - Kerberized Internet Negotiation of Keys
            - RFC4430
            - Realize with Kerberos Protocol, based on the Symmetric key technology.

IPSec Concepts —— Mode of Operation

IPSec can be used in peer-to-peer or network tunnel transport.

- **Transport Mode**
  - Transport Mode IPSec only protects the IP packet content, not the IP header.
  - Since the IP header is not modified, the routing process will not be affected. The data of the transport layer and the application layer are both protected.
  - Typically used in peer-to-peer communication between two hosts
- **Tunnel Mode**
  - Tunnel Mode IPSec will encrypt or authenticate the whole IP packet. The original IP packet will be concealed into a new IP packet, and a new IP header will be attached.
  - Typically used to protect the VPN between network and network, the host-to-network communication and the peer-to-peer communication

IPSec —— ESP/AH & Mode of Operation



(c) Tunnel Mode

Security-related algorithms in IPSec

- ESP: encryption + authentication(optional)；
- AH: authentication(without encryption)
- Algorithms（RFC4835）：
  - ESP Encryption:

AES-CBC with 128-bit keys	MUST / 所有实现必须支持, 最推荐
TripleDES-CBC	MUST- / 所有实现必须支持, 未来可能不需要
AES-CTR	SHOULD / 推荐支持
DES-CBC	SHOULD NOT / 不推荐, 不应再使用

  - ESP Authentication / AH Authentication:

HMAC-SHA1-96	MUST /所有实现必须支持, 最推荐
AES-XCBC-MAC-96	SHOULD+ / 推荐支持, 未来可能成为必须支持
HMAC-MD5-96	MAY / 可选支持

- Provide three models for Encryption +Authentication:
  - ESP with Authentication
  - Transport connection: inner transport mode ESP ＋outer transport mode AH （authentication without encryption）
  - Transport tunnel: inner transport mode AH＋ outer tunnel mode ESP （authentication with encryption ）

Advantages of IPSEC

IPSEC can be implemented and enforced in the firewall/router

- All packets passing the border will be security-enhanced
- The hosts protected by the firewall do not need to deal with the security issues

IPSEC is transparent to the end-user

- Applications built on an IPSEC network do not need to do anything special
- Confidentiality and integrity are ensured automatically

SSL/TLS

TCP layer security

SSL&TLS - History

- Transport Layer Security(TLS) and its predecessor Secure Socket Layer(SSL) are designed to provide reliable end-to-end security services for TCP to provide confidentiality, integrity, and authentication services.
  - Netscape proposed secure Sockets Layer (SSL) v3 as an Internet draft document in 1996
  - TLS (Transport Layer Security) working group is formed to develop the common standards
    - TLS v1.0 / SSL v3.1: RFC 2246, in Jan. 1999: As an upgrade of SSLv3
    - TLS v1.1 / SSL v3.2: RFC 4346, in Apr. 2006
    - TLS v1.2 / SSL v3.3: RFC 5246, in Aug. 2008

SSL&TLS - Concepts

SSL Connection

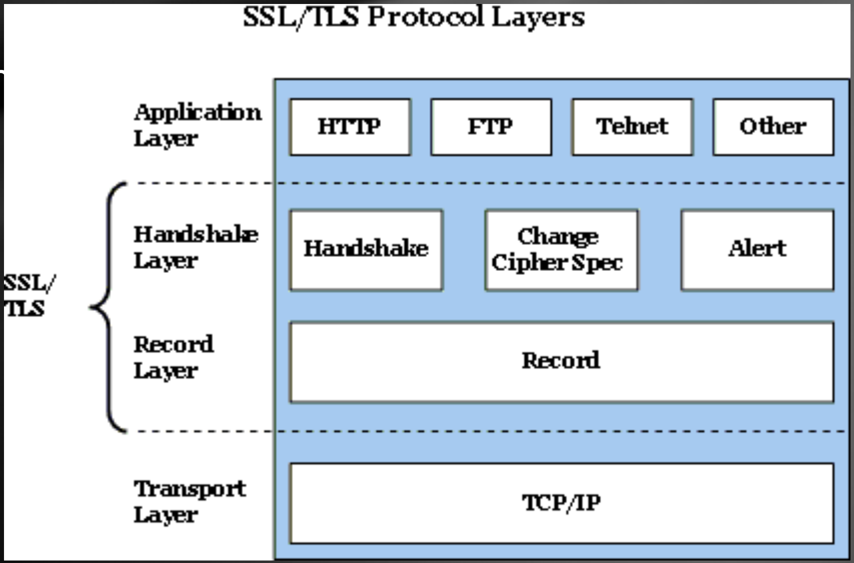
- A connection is a transmission providing a suitable type of service (OSI layer definition).
- An SSL connection is a point-to-point relationship. Connection is temporary, and each connection is associated with a session.

SSL Session

- An SSL session is an association between the client and the server. The handshake protocol creates a session. Sessions define the password security parameters shared by a set of connections.
- Avoid costly negotiation prices for providing each connection security parameter.

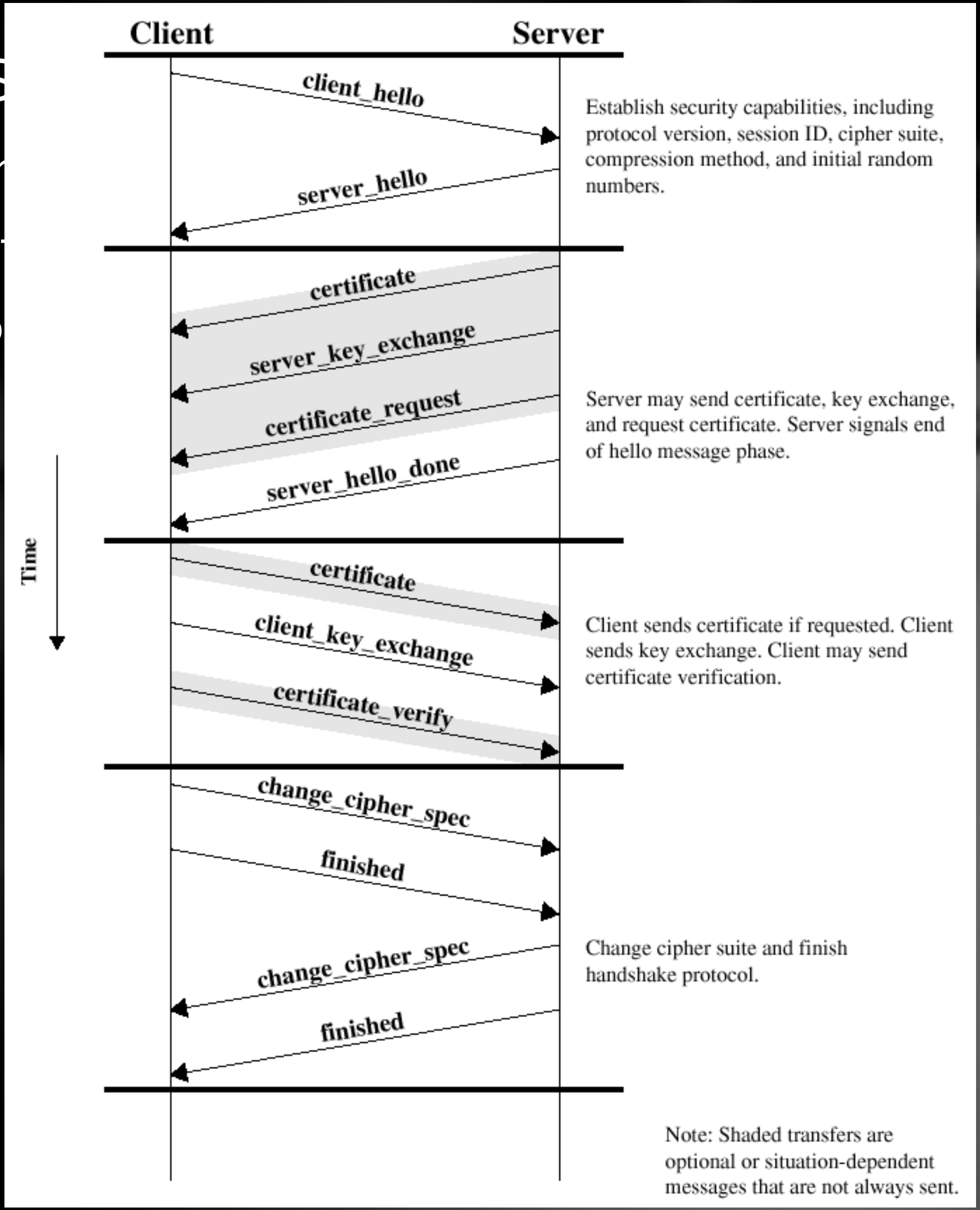
SSL/TLS Protocol Stack

- SSL/TLS protocol is between the transport layer and the application layer, and it is divided into two layers:
  - **Handshake Layer** defined three sub-protocols:
    - Handshake sub protocol
    - Change Cipher Spec sub protocol
    - Alert sub protocol
  - **Record Layer,** Receive and encrypt inform send it to the transport layer.
    - Receive message
    - Block
    - Compress /decompress(optional)
    - Calculate MAC/HMAC
    - Encryption



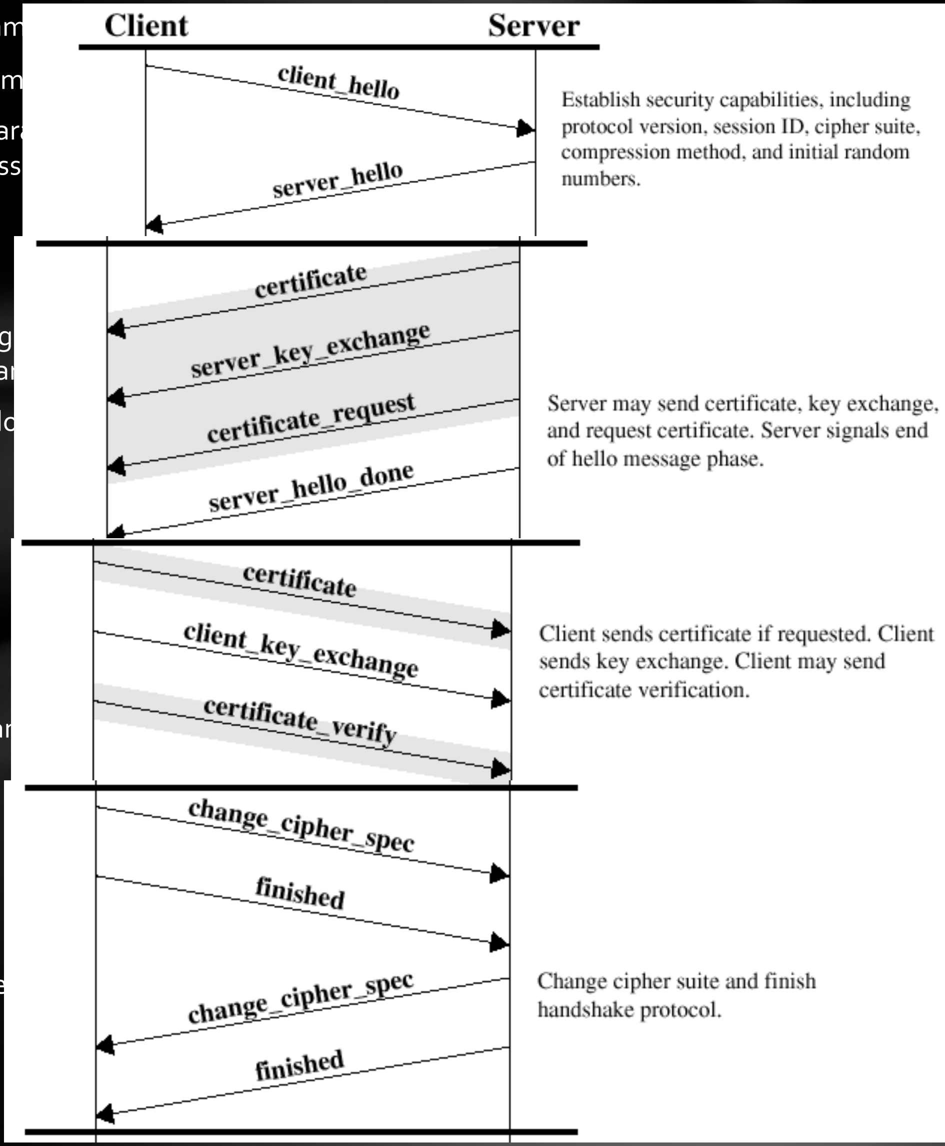
SSL Handshake Protocol

- The most complex part of S
  - Make the server and clients auth
  - Negotiate the encryption algori
  - Execute the handshake protoco


Client
Server
client_hello
server_hello
Establish security capabilities, including protocol version, session ID, cipher suite, compression method, and initial random numbers.
certificate
server_key_exchange
certificate_request
server_hello_done
Server may send certificate, key exchange, and request certificate. Server signals end of hello message phase.
Time
certificate
client_key_exchange
certificate_verify
Client sends certificate if requested. Client sends key exchange. Client may send certificate verification.
change_cipher_spec
finished
change_cipher_spec
finished
Change cipher suite and finish handshake protocol.
Note: Shaded transfers are optional or situation-dependent messages that are not always sent.

SSL Handshake Protocol

- **Phase 1: Establish safety negotiation**
  - The client sends a client_hello message having the following param
    - Version, random numbers, security ID, Ciphertext family, Com
  - The server sends back a hello_server message having the same par out from a proposed set of cryptographic algorithms and compress
- **Phase 2: Server authentication and key exchange**
  - If the server needs to be authenticated, send a certificate
  - Send server_key_exchange packets (depending on the type of alg if the Diffie-Hellman algorithm, exchange agreement of global pa
  - Send the signature information to authenticate (ClientHello.rand | ServerParams)
  - The server requests a certificate_request to the client
  - The server sends a hello-ending packet server_hello_done
- **Phase 3: Client authentication and key exchange**
  - Send your own CA certificate certificate
  - Send the customer password exchange packets client_key_exchar the type of algorithm used.
  - Send a certificate to verify the message certificate_verify
- **Phase 4: end**
  - The client sends the message change_cipher_spec
  - Send the finished message under the new algorithm and a new ke password exchange and authentication process.
  - The server sends change_cipher_spec messages.
  - The server sends the finished message.



Establish security capabilities, including protocol version, session ID, cipher suite, compression method, and initial random numbers.

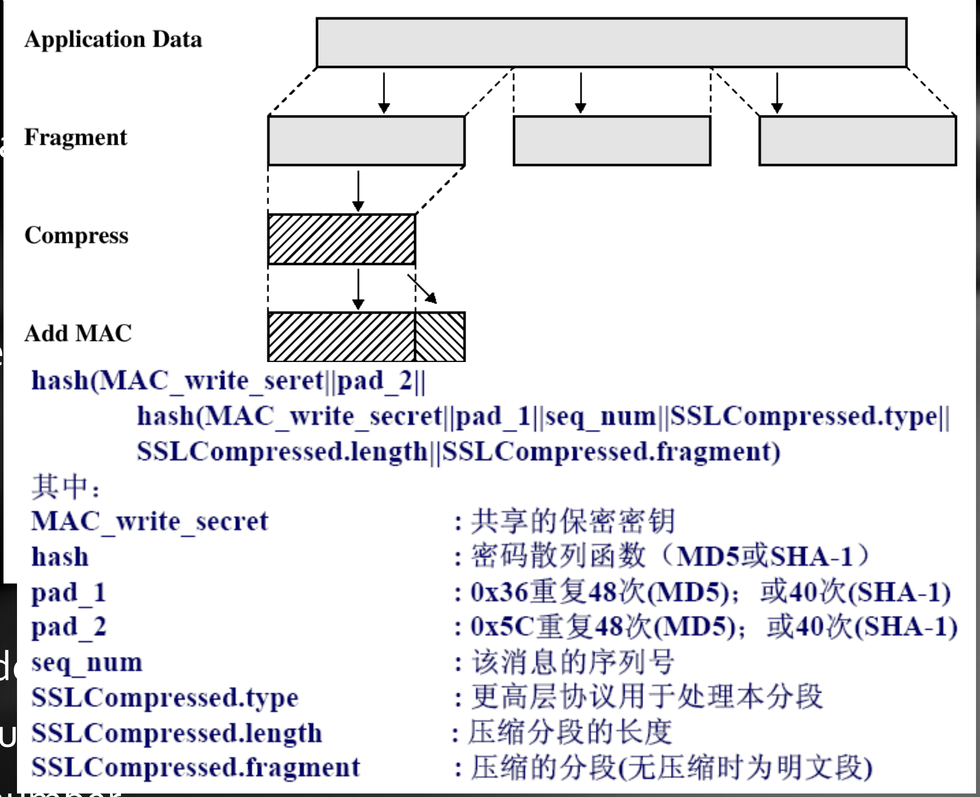
Server may send certificate, key exchange, and request certificate. Server signals end of hello message phase.

Client sends certificate if requested. Client sends key exchange. Client may send certificate verification.

Change cipher suite and finish handshake protocol.

SSL Record Protocol

- **Step 1: fragmentation**
  - Message data is fragmented into 214 (16384) byte block sizes or smaller.
- **Step 2: compression (optional)**
  - It must be lossless compression; if the dat
    1024 bytes.
- **Step 3: MAC calculation**
  - Using shared secret key MAC_write_secre
- **Step 4: Encryption**
  - Encryption with chosen algorithms.
- **Step 5: add an SSL record header**
  - Content type (8bits): high-level protocol d
  - Major version (8bits): the major version nu
  - Minor version (8bits): The minor version number
  - Compression length (16bits): plaintext data piece (or compressed piece) length



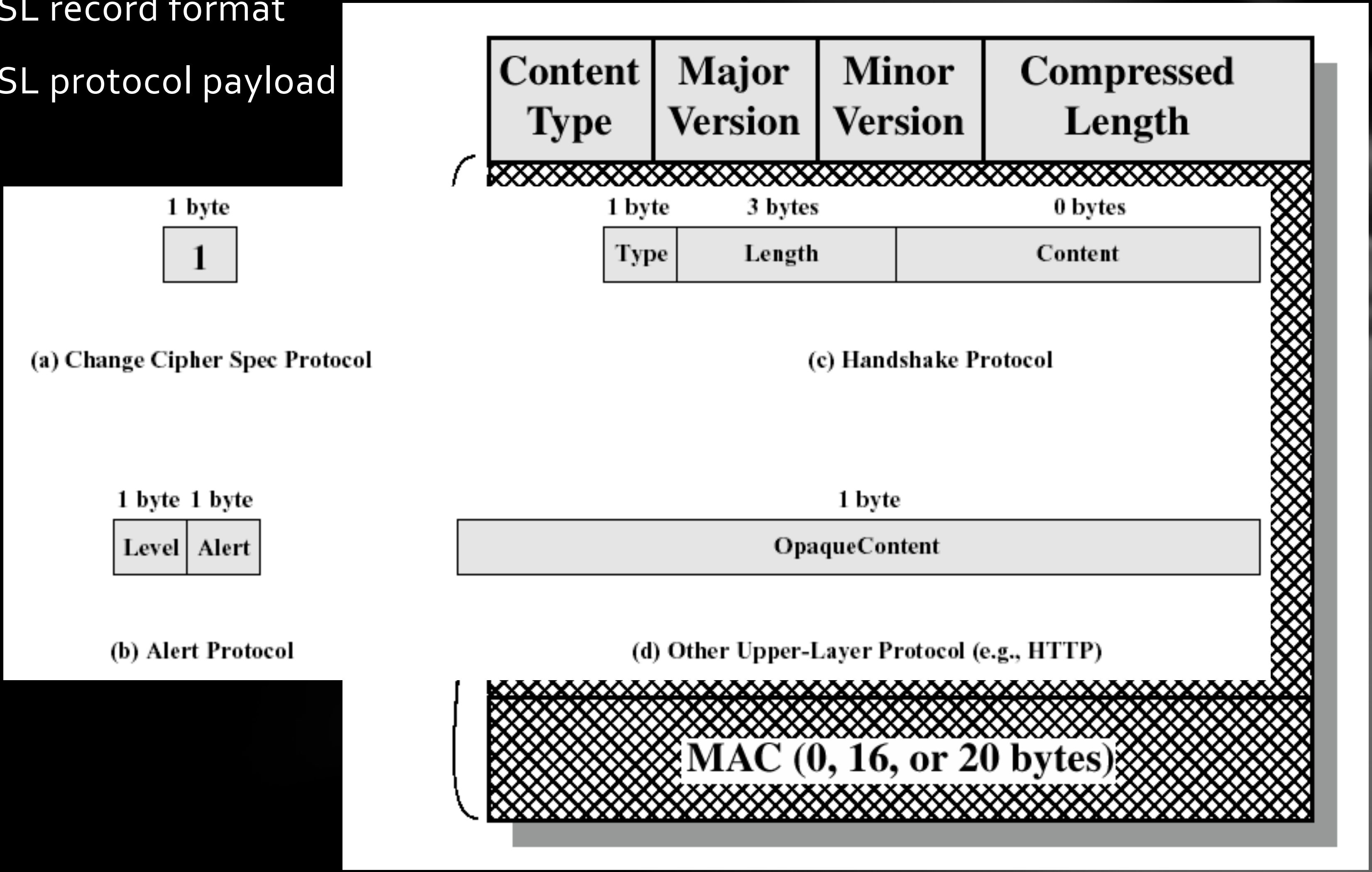
hash(MAC_write_seret||pad_2||
hash(MAC_write_secret||pad_1||seq_num||SSLCompressed.type||
SSLCompressed.length||SSLCompressed.fragment)

其中:

MAC_write_secret	: 共享的保密密钥
hash	: 密码散列函数（MD5或SHA-1）
pad_1	: 0x36重复48次(MD5)；或40次(SHA-1)
pad_2	: 0x5C重复48次(MD5)；或40次(SHA-1)
seq_num	: 该消息的序列号
SSLCompressed.type	: 更高层协议用于处理本分段
SSLCompressed.length	: 压缩分段的长度
SSLCompressed.fragment	: 压缩的分段(无压缩时为明文段)

SSL Record Format and Payload

- SSL record format
- SSL protocol payload



Review

- Security Issues in TCP/IP
  - Sniffing
  - ARP Spoofing
  - IP Spoofing
  - TCP SYN Flooding
  - TCP SYN Prediction
  - TCP Congestion Control
  - DNS Spoofing
- Security mechanism in IP / TCP
  - IPSec:
    - Security Association、AH、ESP
    - Transport Model and Tunnel Model
  - SSL/TLS:
    - Concepts, Record Protocol and Handshake Protocol