TCP/IP
Hacks and Defenses

2001. 08. 09

KAIST 해킹바이러스연구센타

고려대학교 전산학과

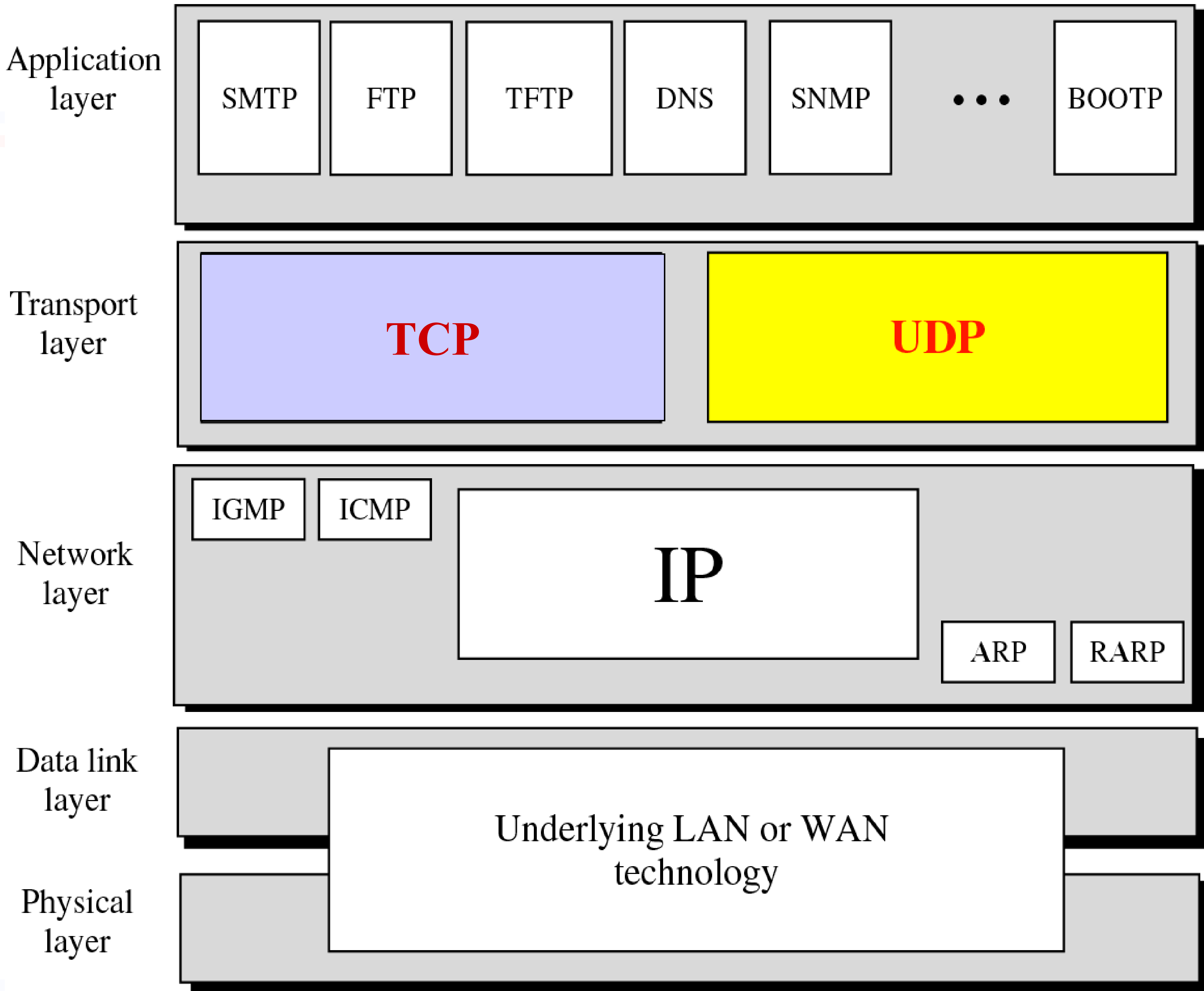
조 충 호

chcho@tiger.korea.ac.kr

Table of contents

Out line

- ❑ Background
  - ✓ Networking Background
  - ✓ Sniffers
  - ✓ Packet Filters
- ❑ Attacks and Defenses
  - ✓ IP, ICMP, ARP, UDP, TCP attacks and probes
  - ✓ Explanation of the attack pattern
  - ✓ Filters to detect attacks in sniffer logs
  - ✓ Cisco router ACLs to block attacks

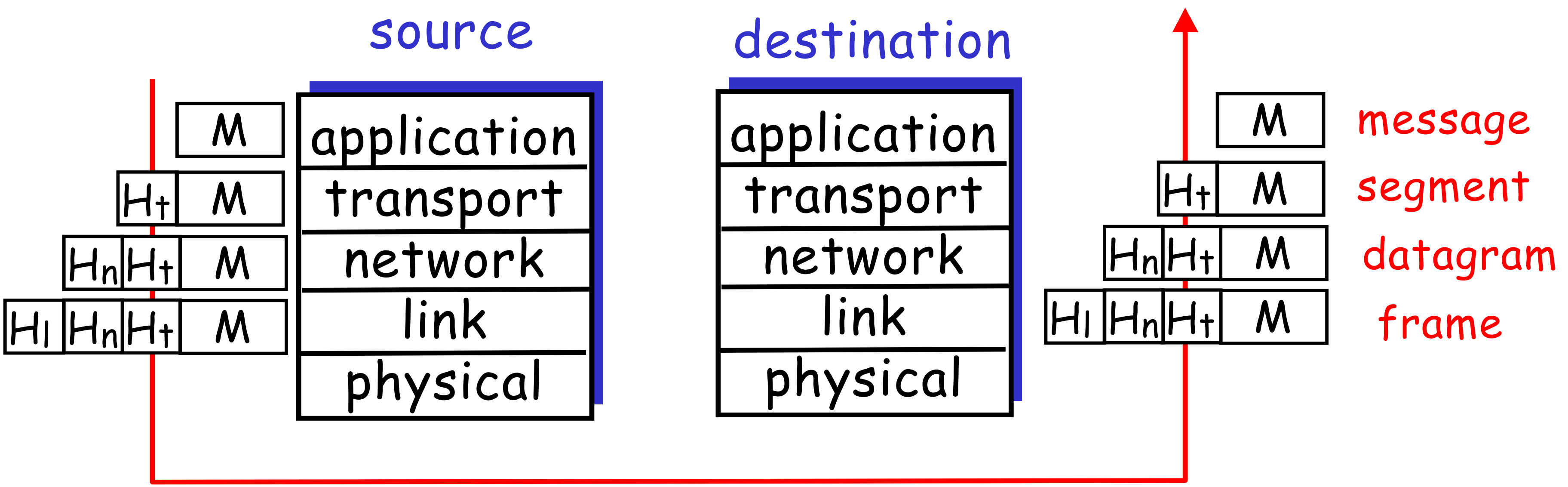

Application layer
SMTP
FTP
TFTP
DNS
SNMP
• • •
BOOTP
Transport layer
TCP
UDP
Network layer
IGMP
ICMP
IP
ARP
RARP
Data link layer
Physical layer
Underlying LAN or WAN technology

Internet Protocol

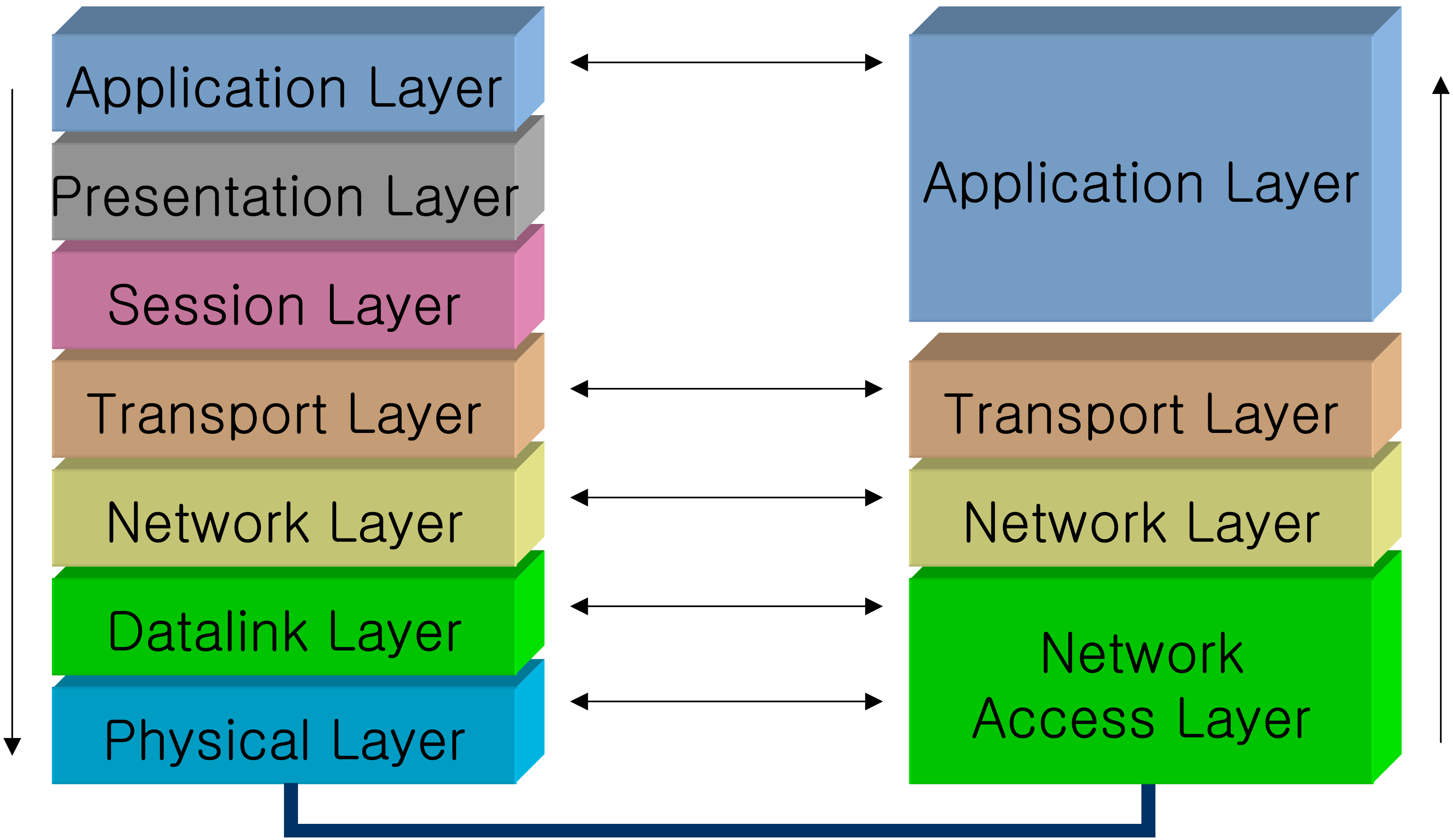
- The Internet Protocol model is a four-layer model (as opposed to the classic seven-layer OSI model):
- The lowest layer is the <u>*network interface layer*</u> where data is put out onto a physical media such as Ethernet or FDDI.
- Above the network interface layer is the <u>*internet layer*</u> where data is transmitted as IP datagrams.
- The <u>*transport layer*</u> is built on top of the internet layer and includes well-known protocols like TCP and UDP. ICMP is actually an adjunct to IP and is an internet layer protocol, but ICMP messages get encapsulated in IP datagrams just like TCP and UDP data.
- Finally there is an <u>*application layer*</u> which includes common applications (`telnet`, NFS, `ping`) which utilize transport layer protocols.

Encapsulation

- Each layer is built on top of the previous layer and data from one layer gets *encapsulated* within the data portion of the hardware frame or datagram format on the layer below it.



OSI RM vs. Internet



Dept. Computer Science Korea Univ.

Information Superhighway

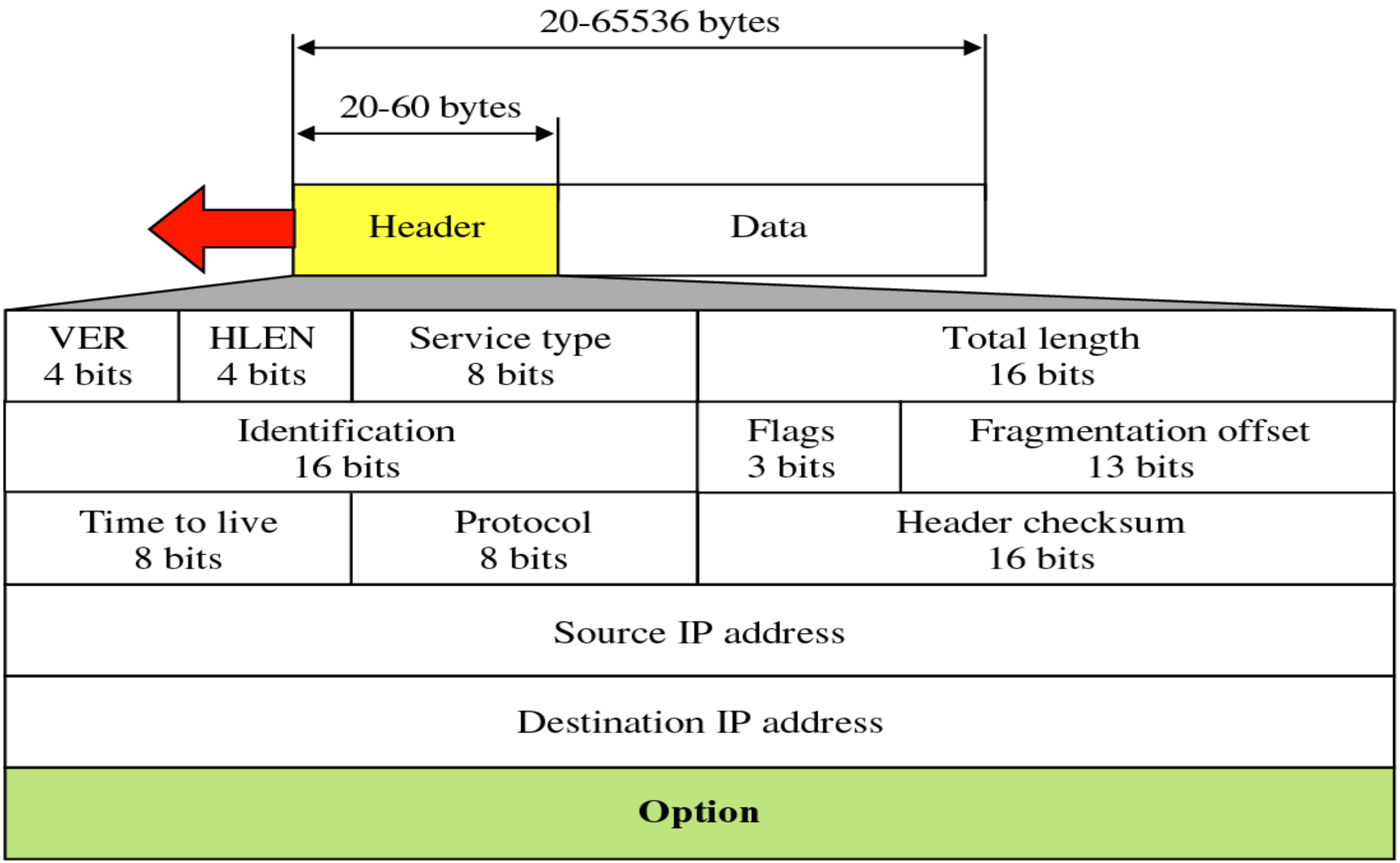
- All TCP, UDP, and ICMP data gets transmitted *encapsulated* within IP datagrams

- IP datagram delivery is *unreliable*: there are no guarantees that a datagram will reach its destination

- IP networking is inherently *connectionless*: IP does not maintain any state information about successive datagrams and each is handled/routed independently

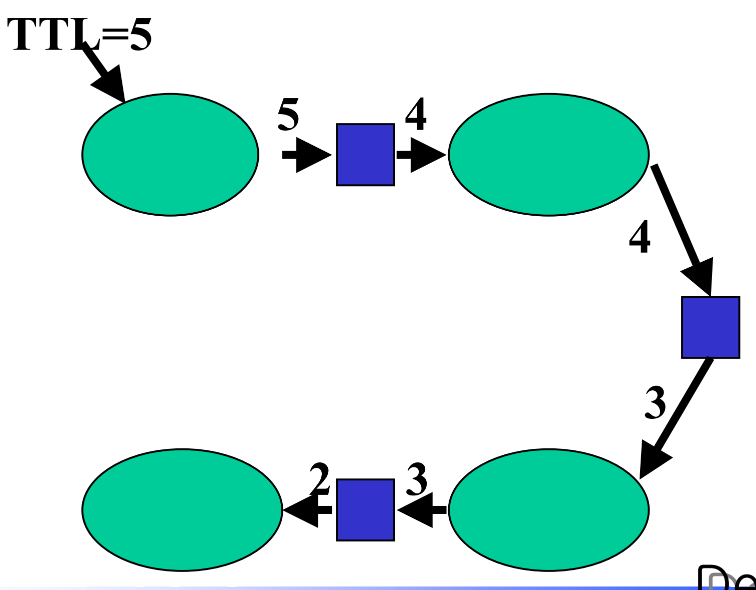
Dept. Computer Science Korea Univ.

IP datagram format



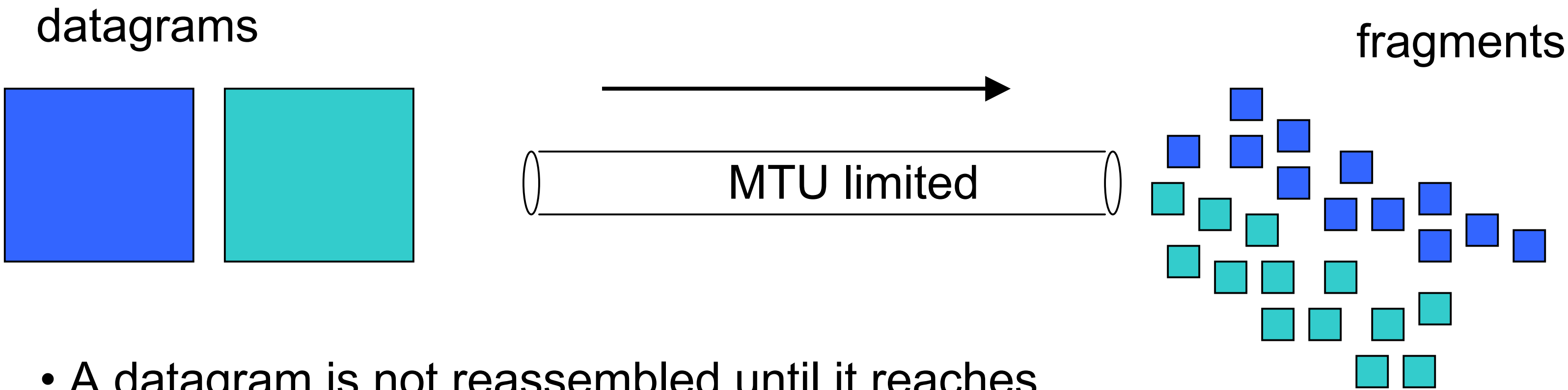
Time To Live (TTL)

- Each packet leaves the source network interface with a positive time to live value
- Each router or gateway along the route of the packet decrements the TTL before forwarding the packet
- Router which decrements a packet's TTL to zero drops the packet and sends back an error
  - ✓ http://www.switch.ch/docs/ttl_default.html



10진수	Protocol
0	예약
1	ICMP
4	IGMP
5	ST(stream protocol)
6	TCP
8	EGP
9	IGP
17	UDP

Fragmentation and Reassembly

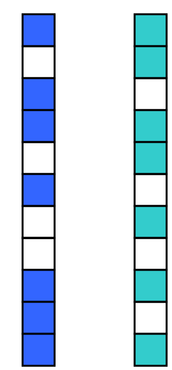


• A datagram is not reassembled until it reaches its final destination (each fragment is routed independently)

• All fragments that should be reassembled together carry the same identification number

• The fragment offset tells where the fragment should be placed in the sequence during datagram reassembly

• The “more fragments” flag is set in all fragments except the last



receiving computer’s fragment reassembly buffer

Dept. Computer Science Korea Univ.

MTU

- 각 네트워크에서 전달되는 최대 전송 길이를 MTU (Maximum Transfer Unit)
- 네트워크가 사용하는 프로토콜에 따라 프레임 형식과 크기(MTU)가 서로 다름

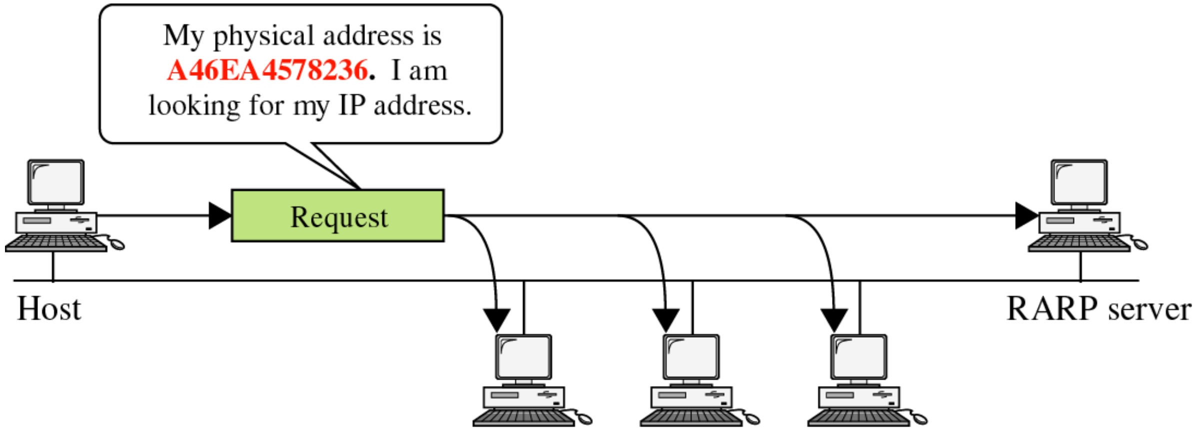
데이터 링크 계층 프레임

Header	MTU : IP 데이터그램	Tailer

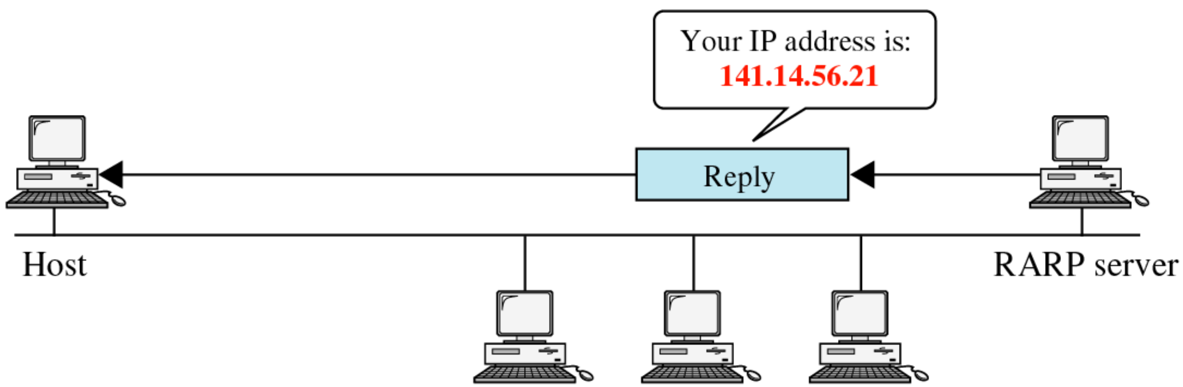
Protocol	MTU
Hyperchannel	65,535
Token Ring (16 Mbps)	17,914
Token Ring (4 Mbps)	4,464
FDD	14,352
Ethernet	1,500
X.25	576
PPP	296

Dept. Computer Science Korea Univ.

ARP(Address Resolution Protocol)



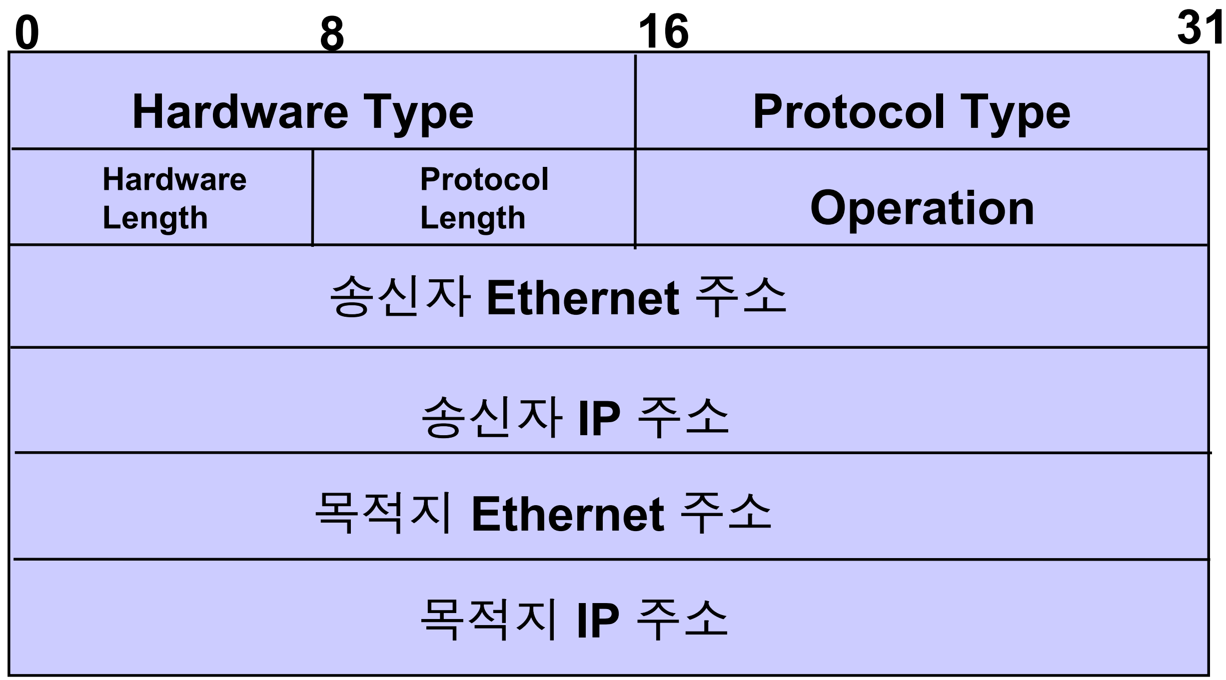
a. RARP request is broadcast

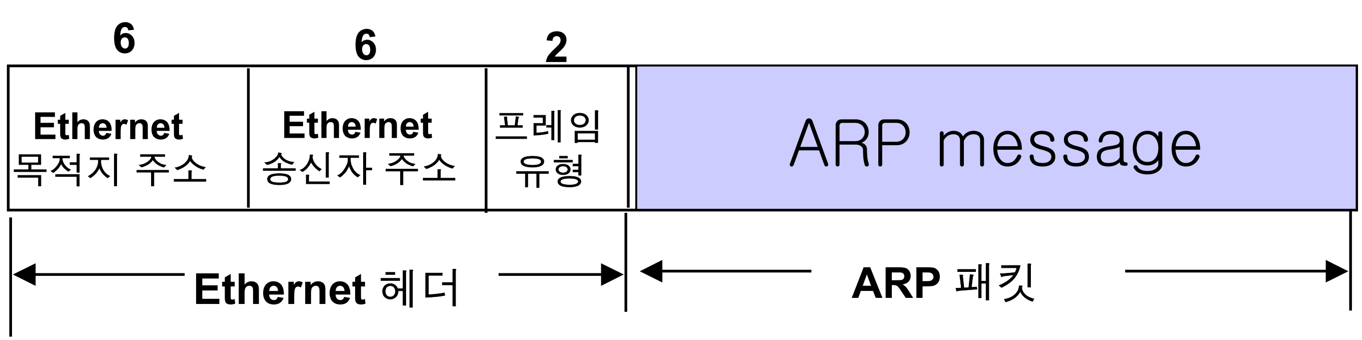


b. RARP reply is unicast

Dept. Computer Science Korea Univ.

ARP Message format



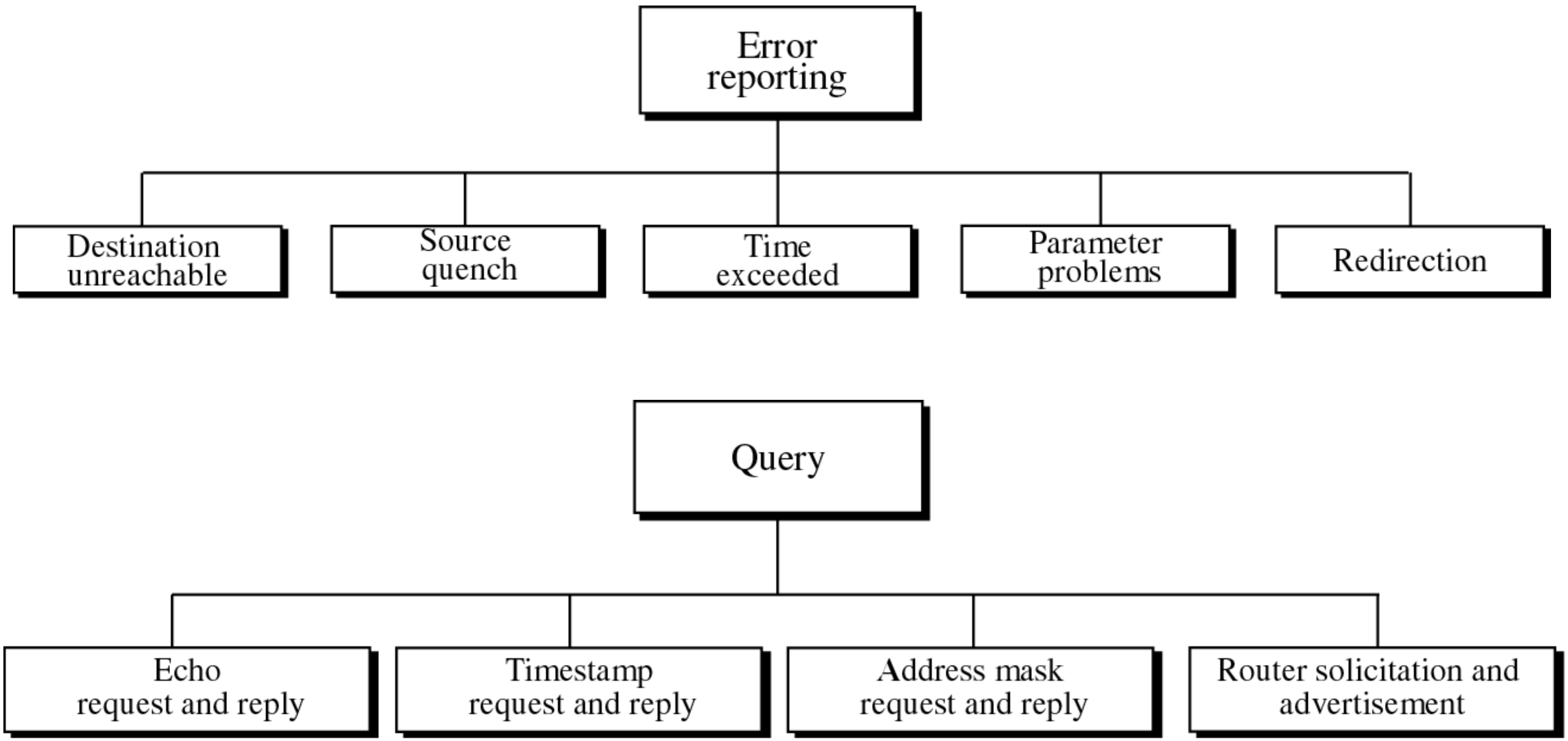


ICMP

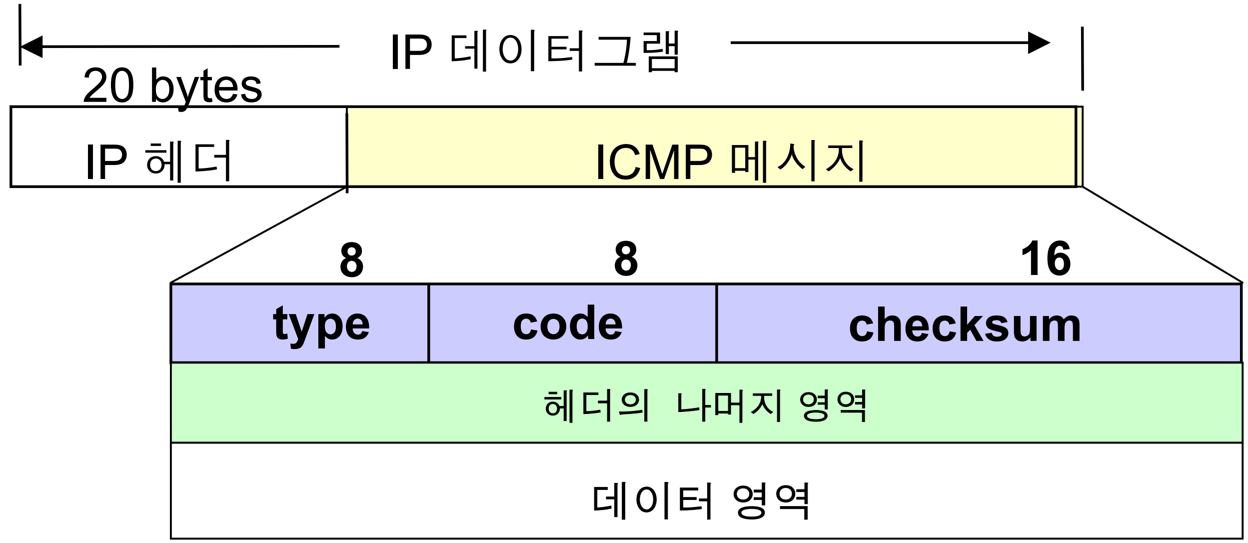
- ICMP is an adjunct of IP
- ICMP is used by various network protocols to transmit informational and administrative/error messages
- There are many different “types” of ICMP messages, identified by *ICMP message number*

msg#	description	msg#	description
0	***echo reply***	12	*parameter problem*
3	***destination unreachable***	13	*timestamp request*
4	source quench	14	timestamp reply
5	redirect	15	information request
8	***echo request***	16	information reply
9	*router advertisement*	17	address mask request
10	router solicitation	18	address mask reply
11	***time exceeded***		

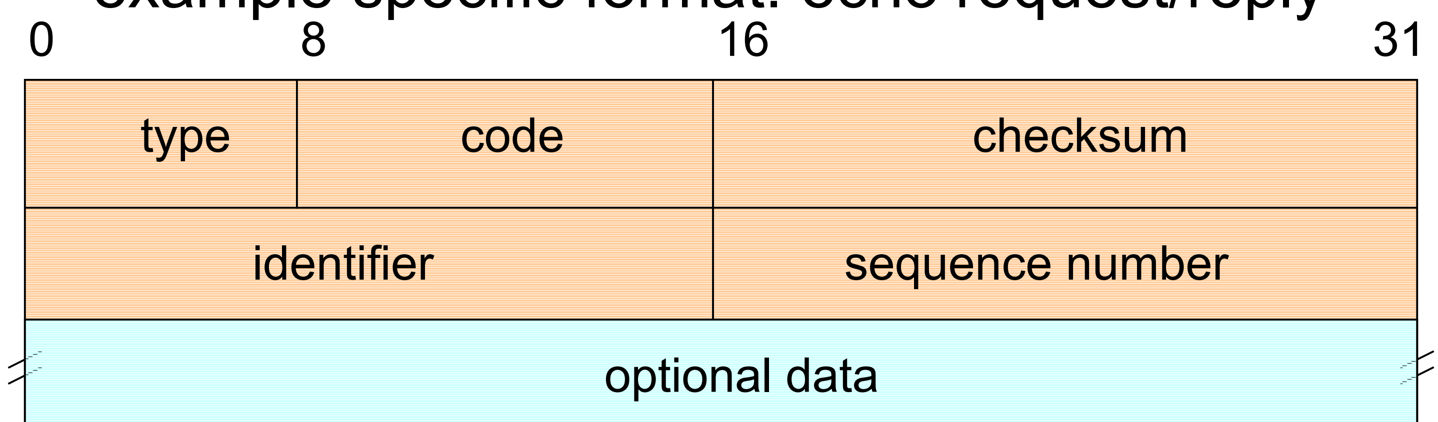
ICMP message type



Encapsulation of ICMP Packet

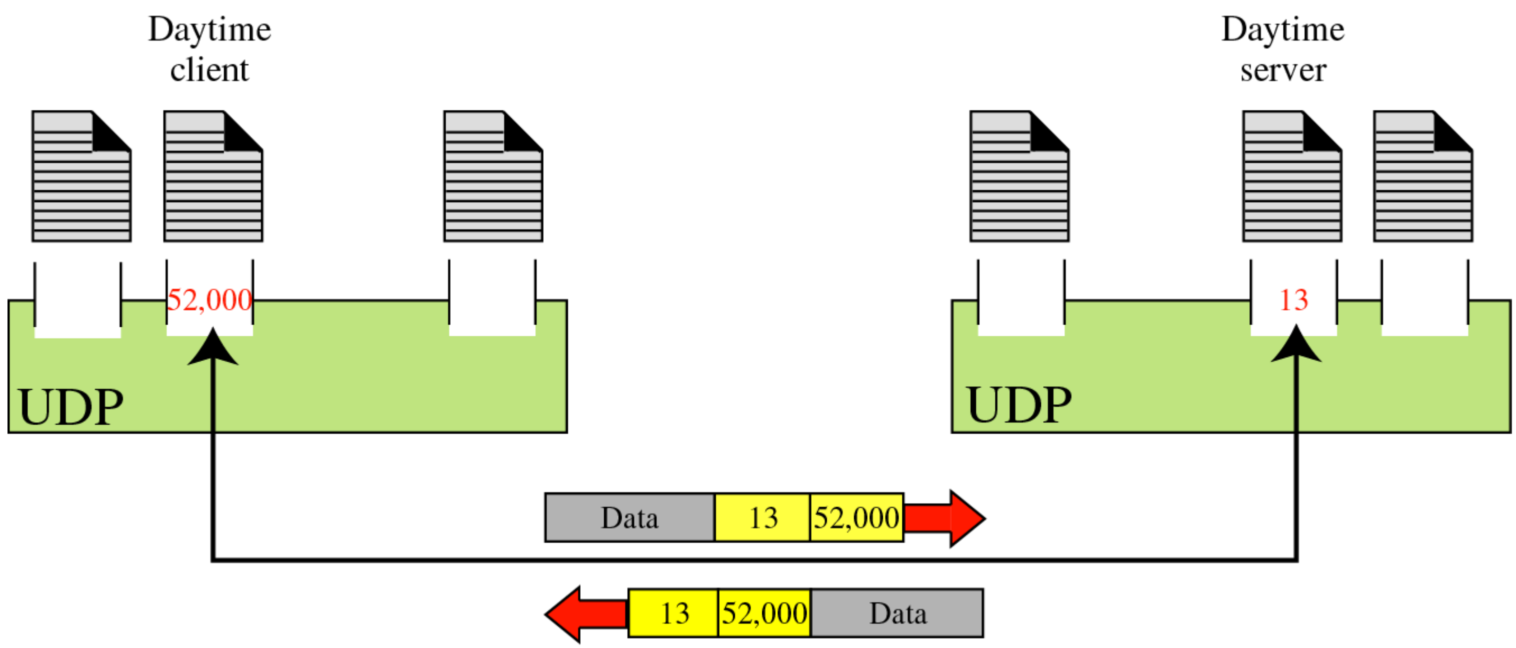


example specific format: echo request/reply

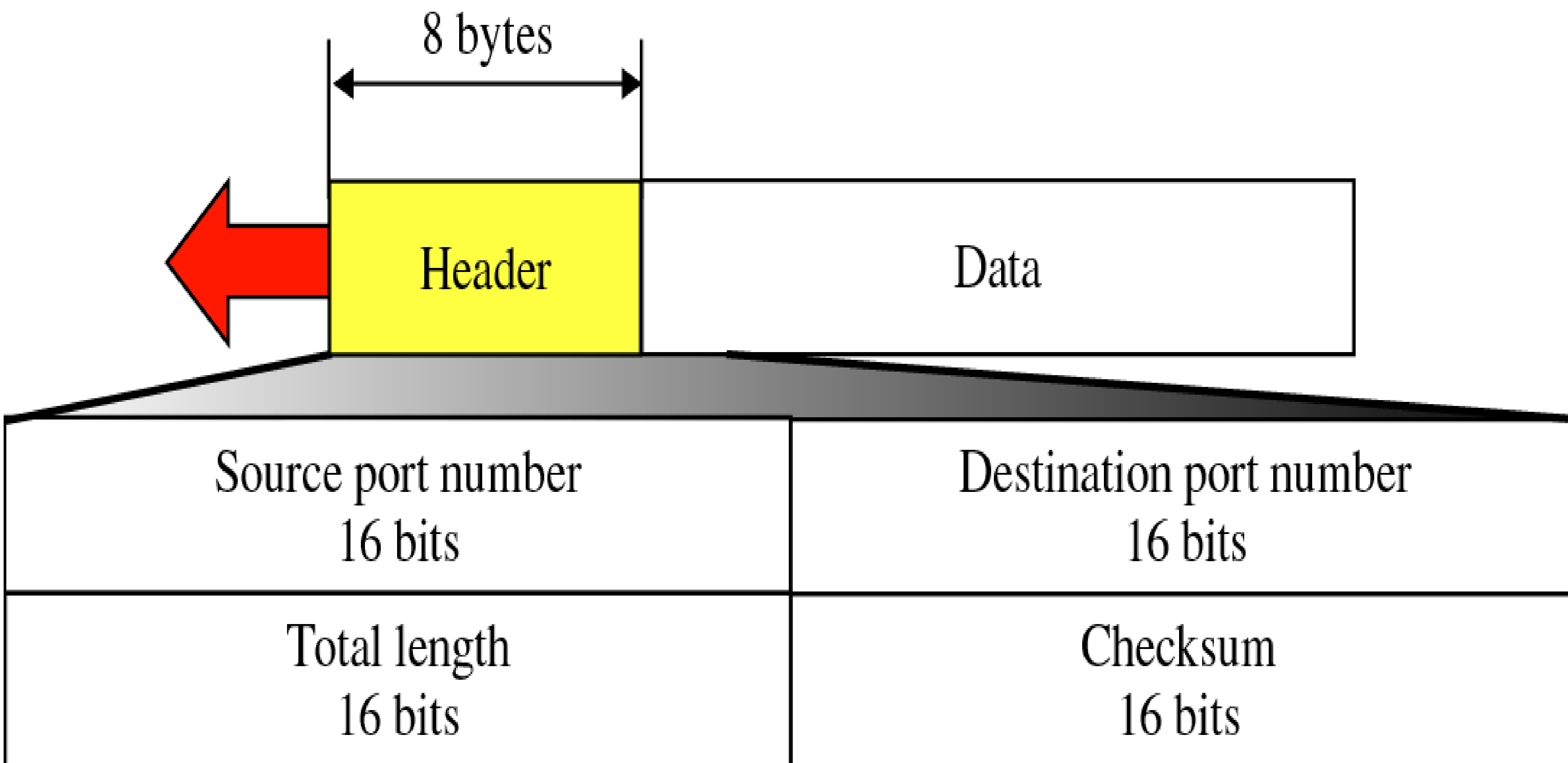


The UDP Protocol

- UDP provides *no reliability*: there is no guarantee that the datagrams ever reach the desired destination
- *Port numbers* identify the sending process and the receiving process



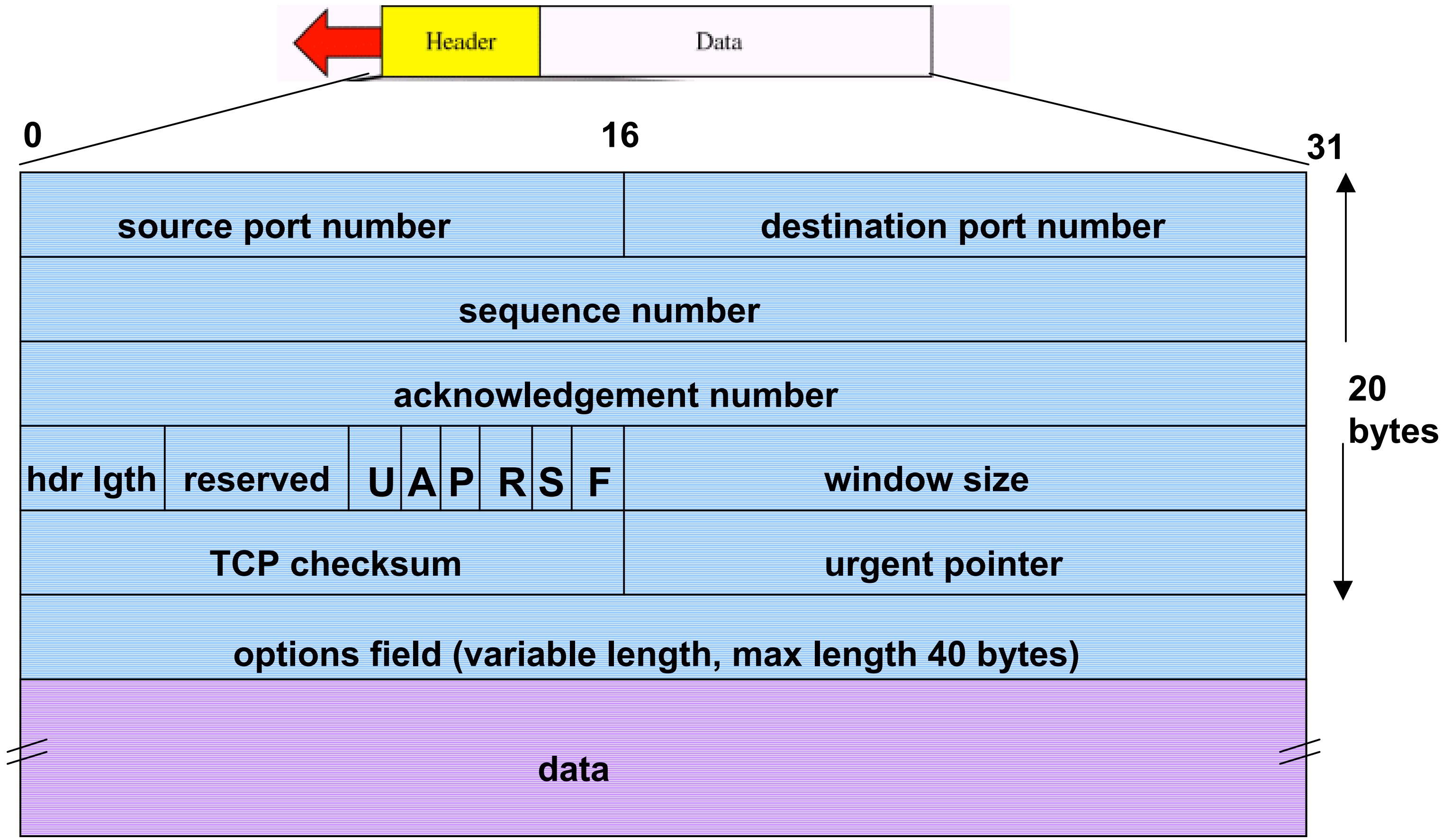
UDP datagram format



The TCP Protocol

- TCP is a *reliable*, *connection-oriented* service:
  - TCP will *retransmit* a packet if the destination does not acknowledge receipt of that packet within a specified time
  - Each TCP packet has a unique *sequence number* associated with it for ordering and retransmission purposes
- As with UDP, *port numbers* identify the sending and the receiving process
- The function of the TCP packet is designated by a combination of *flags*

TCP Segment Format



TCP Segment Flags

- SYN : synchronize the sequence numbers to establish a connection
- ACK : acknowledgement number is valid
- RST : reset (abort) the connection
- FIN : sender is finished sending data -- initiate a half close
- PSH : tells receiver not to buffer the data before passing it to the application (interactive applications use this)
- URG : urgent pointer is valid (often results from an interrupt)

Dept. Computer Science Korea Univ.

3-way handshaking for connection establishment

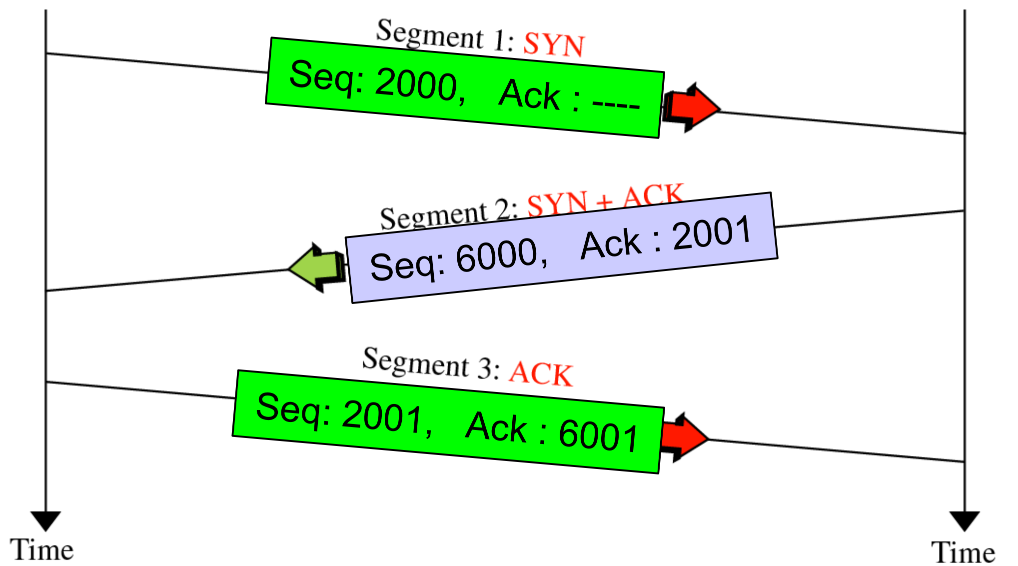
active open by client

passive open by server

client
(port = 33623/tcp)

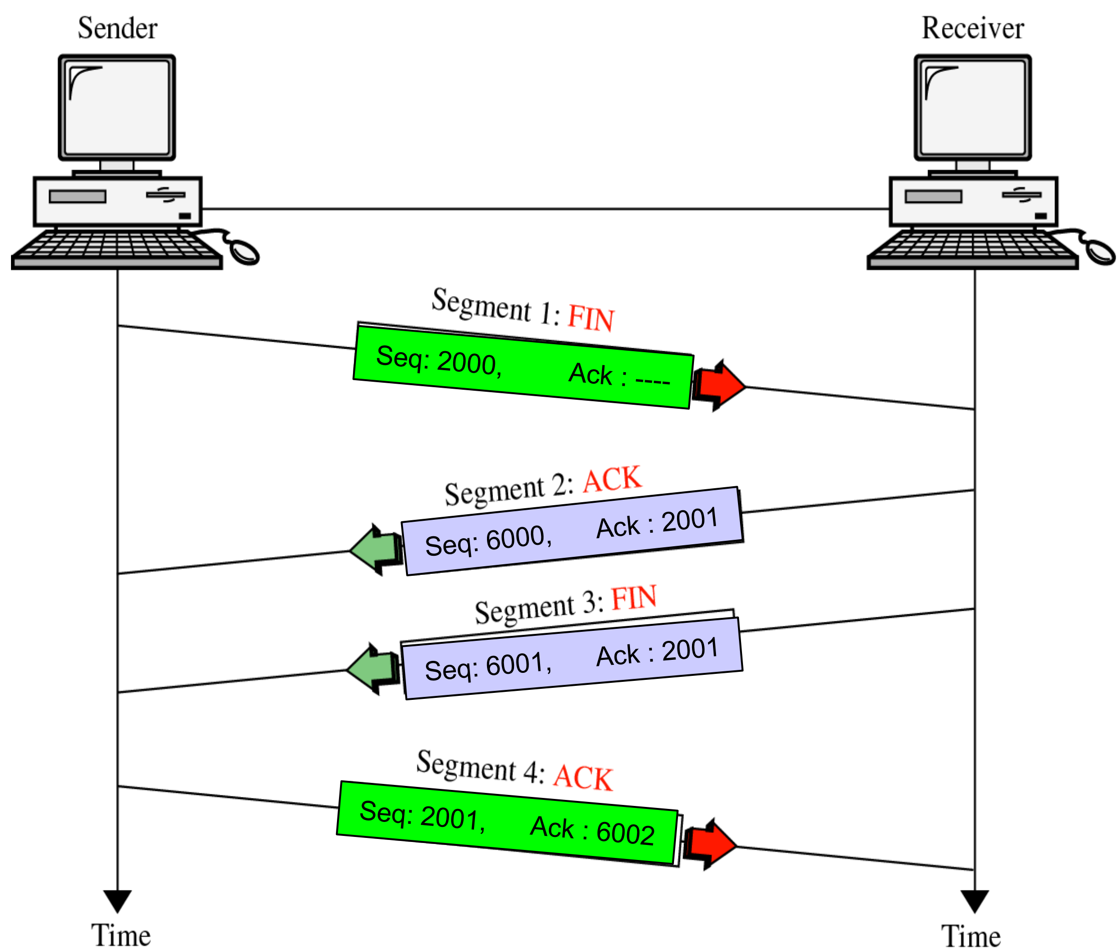
[ACK set for remainder of session]
[session proceeds]

server
(port = 23/tcp)

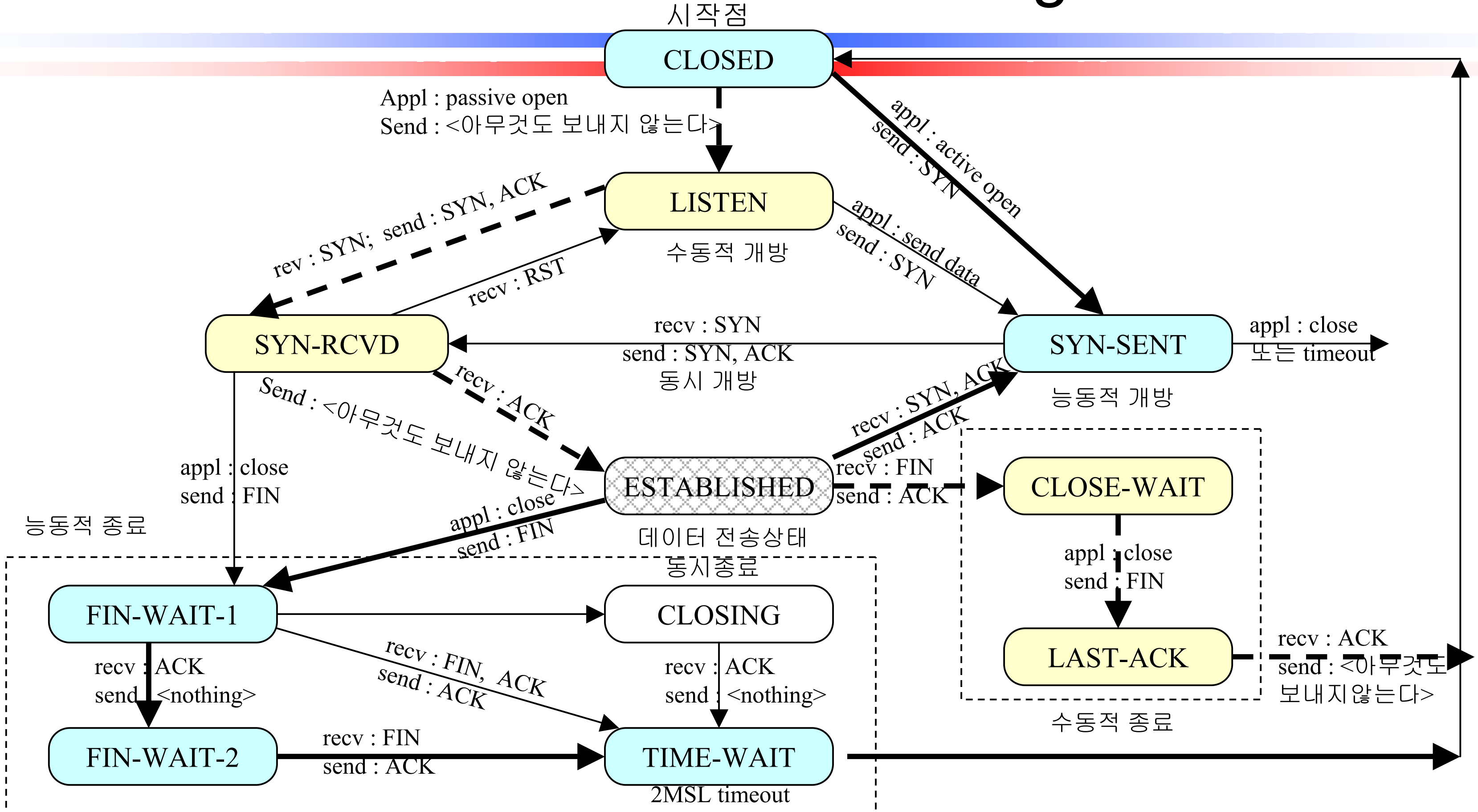


4-way handshaking for connection release

Either the client or the server may initiate the closing sequence



TCP state transition diagram



실선화살표 - 클라이언트에 대한 정상적인 천이를 가리킴

점선화살표 - 서버의 대한 정상적인 천이를 가리킴

appl : 응용이 여기에 나타난 동작을 발행할 땡 일어나는 상태 천이를 가리킴

recv : 세그먼트를 수신할 때 취해지는 상태천이를 가리킴

send : 이 천이를 대해 보내지는 것

Network Traffic Sniffing

Sniffer : Any device, software or hardware, that listens to all packets traveling along a network.

Promiscuous Mode : Allows the computer to listen to all traffic on the network, regardless of a packet's destination IP address.

Normally, a computer listens only to the traffic that is destined for its own address or a broadcast address.

`tcpdump` : This is the sniffer we will use for this class.

Unix version: **`ftp://ftp.ee.lbl.gov/tcpdump.tar.Z`**
Win95/98/NT port: **`http://netgroup-serv.polito.it/windump`**

tcpdump Filters

- **tcpdump** allows the user to select certain types of traffic from the packets seen by the interface
  - ✓ The basic syntax for a `tcpdump` filter is

<header>[<offset>:<length>] <relation> <value>

  - ✓ In plain terms, use this filter to detect `telnet` traffic:

```
tcpdump 'tcp[2:2] = 23'
```

- Any field in the IP, ICMP, UDP, and/or TCP header can be selected and filtered on using byte offsets
- Macros are also implemented for some header fields to make filtering easier, e.g.
  - ✓ Source/destination address and port
  - ✓ Network address
  - ✓ Protocol type
  - ✓ … and many others

Dept. Computer Science Korea Univ.

ARP packet – tcpdump

```
[root@consult /root]# tcpdump -e arp
tcpdump: listening on eth0
07:44:23.898915 79:94:74:11:d7:dc bc:47:d8:7b:31:51 arp 42: arp reply
                82.195.6.82 is-at 79:94:74:11:d7:dc
07:44:23.898954 b8:29:3:9c:9e:5c 3f:cf:9b:70:fa:14 arp 42: arp reply
                204.227.135.56 is-at b8:29:3:9c:9e:5c
07:44:23.898991 5:6f:25:db:4b:76 97:a0:d6:c7:f1:8f arp 42: arp reply
                158.81.199.91 is-at 5:6f:25:db:4b:76
07:44:23.899027 f0:f4:2c:8f:50:f7 a6:ca:21:a1:dd:26 arp 42: arp reply
                114.215.48.176 is-at f0:f4:2c:8f:50:f7
07:44:23.899063 10:3:1:5b:78:9f de:d0:b:d0:60:fa arp 42: arp reply
                171.63.250.67 is-at 10:3:1:5b:78:9f
```

Sniffer Format and Filter Introduction

`23:06:37 10.1.101.1 > 224.0.0.10: ip-proto-88 40 [tos 0xc0]`

time	source IP	dest IP	protocol	data bytes	type of service

`ip[9] = ip[9:1] = 88`

Byte number 9 of the IP header (counting begins at 0) carries a binary value that is equal to 88 in decimal

ftp://ftp.isi.edu/in-notes/iana/assignments/protocol-numbers =>

ip-proto-88 = EIGRP

Dept. Computer Science Korea Univ.

Filtering on Hosts, Nets, and Ports

`08:08:16.155 spoofed.target.net.7 > 172.31.203.17.chargen: udp`

timestamp	src IP	src port	dst IP	dst port	protocol
08:08:16.155	spoofed.target.net	7	172.31.203.17	chargen	udp

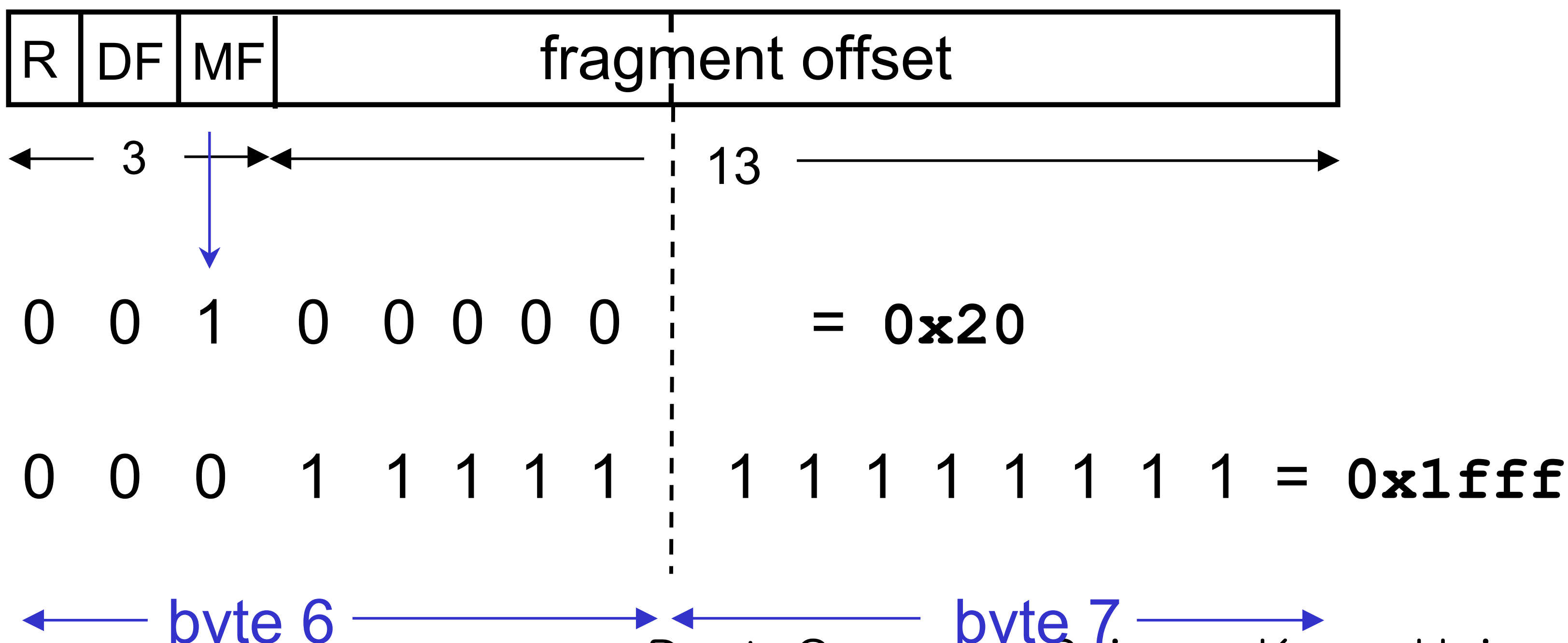
- addresses may be written as either hostnames or x.x.x.x
- ports may be written as either service names or numbers
- if a range is specified the byte numbers must be used

host is either src or dst:	**`host spoofed.target.net`**
dst network is 172.31.x.x	**`dst net 172.31`**
dst network is 172.16 - 172.31	**`‘dst net 172 and (ip[17]>15) and (ip[17]<32)’`**
src port equals 7:	**`src port 7`**
dst port equals 19:	**`dst port chargen`**
src ports less than 20:	**`udp[0:2] < 20`**
dst ports less than 20:	**`udp[2:2] < 20`**

Bit Masking Operations

```
fragmented ip packets with more fragments coming
              ip[6:1] & 0x20 != 0

fragmented ip packets with zero offset (first in sequence)
              ip[6:2] & 0x1fff = 0
```



TCP Connection Establishment and Termination

Establishment

```
client.4247 > server.514: S 3073470005:3073470005(0) win 512 <mss 1460>
server.514  > client.4247: S 1932608000:1932608000(0)
                                 ack 3073470006 win 61320 <mss 1460> (DF)
client.4247 > server.514: . ack 1932608001 win 32120 (DF)
```

Termination

```
client.4247 > server.514: F 3073470006:3073470006(0)
                                           ack 1932608001 win 32120
server.514  > client.4247: . ack 3073470007 win 61320 (DF)
server.514  > client.4247: F 1932608001:1932608001(0)
                                           ack 3073470007 win 61320 (DF)
client.4247 > server.514: . ack 1932608002 win 32120 (DF)
```

S = SYN flag is set
F = FIN flag is set
. = none of the SFRP flags are set
(ack and urg are displayed differently)

(x) = x data bytes in the packet
win = advertised window size
mss = max segment size announcement
DF = don’t fragment flag is set

Dept. Computer Science Korea Univ.

TCP Connection Establishment and Termination

Relative Sequence Number Format

```
Establishment
client.4247 > server.514: S 3073470005:3073470005(0) win 512 <mss 1460>
server.514  > client.4247: S 1932608000:1932608000(0)
                                 ack 3073470006 win 61320 <mss 1460> (DF)
client.4247 > server.514: . ack 1 win 32120 (DF)
```

```
Termination
client.4247 > server.514: F 1:1(0) ack 1 win 32120
server.514  > client.4247: . ack 2 win 61320 (DF)
server.514  > client.4247: F 1:1(0) ack 2 win 61320 (DF)
client.4247 > server.514: . ack 2 win 32120 (DF)
```

- Sequence numbers are given as (if the S,F,or R flags are set):
  `initial : expected`
  where **`expected = data + initial`**
- By default **`tcpdump`** will print relative sequence numbers for all packets except the initial SYN connections
- These relative sequence numbers give the difference between the current packet's sequence number and the initial packet's sequence number

More Complicated Filters

```
ip and
(
(ip[12:4] = ip[16:4])
or
((not src net 192.168) and
(
    (ip[19] = 0xff) or
    (ip[19] = 0x00) or
    ((ip[6:1] & 0x20 != 0) and (ip[6:2] & 0x1fff = 0)) or
    (net 0 or net 127 or net 1) or
    (ip[12] > 239)      or
    ( ((ip[0:1] & 0x0f) > 5) )
 )
))
```

- `tcpdump` allows such filters to be saved in a file and then read in by `tcpdump` at runtime with the `-F` flag:

  `tcpdump -F` *filterfile*

What Is a Packet Filter?

- A router or other gateway device needs to be able to look into packet headers to route packets correctly

- Most routers can also be configured to permit or reject packets based on header information

- Each packet is checked individually as it passes through the router-- there is no "state"

Dept. Computer Science Korea Univ.

Why Use Packet Filters?

- Inexpensive
- One less device to maintain
- Can be quickly deployed in places where a full firewall installation isn't warranted

Dept. Computer Science Korea Univ.

Well-Known Ports

- Recall that TCP and UDP-based applications are identified by port number:
  - ✓ Common servers assigned to well-known ports:

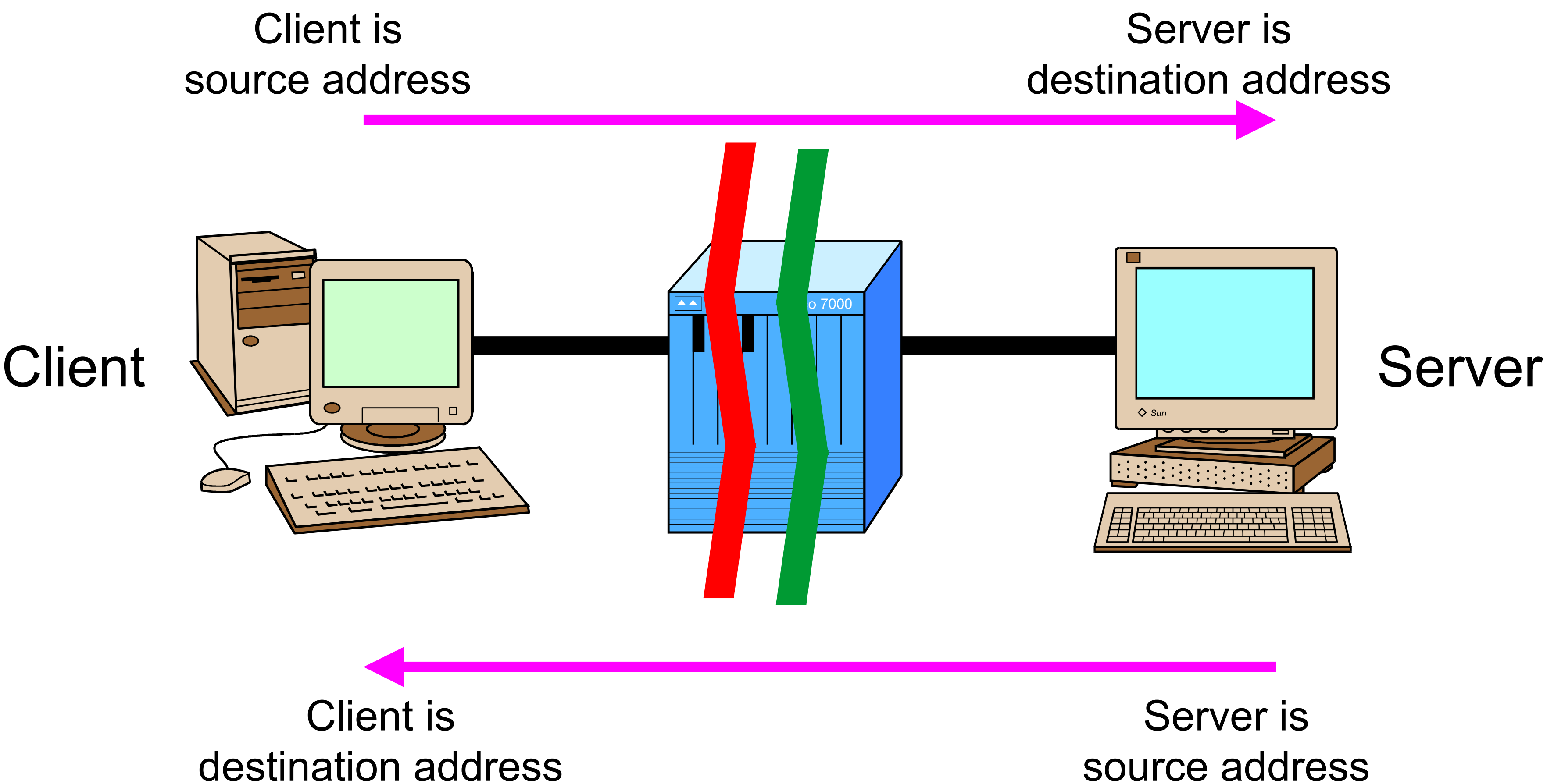
Port	Protocol	Port	Protocol
-----	-----------	-----	-----------
7	Echo	23	TELNET
9	Discard	25	SMTP
11	Users	53	DNS
13	Daytime	67	BOOTP
17	Quote	79	Finger
19	Chargen	80	HTTP
20	FTP, Data	111	RPC
21	FTP, Control		

  - ✓ Client side generally chooses a random port number
- Block certain types of network traffic by blocking access to the appropriate server port
- `ftp://ftp.isi.edu/in-notes/iana/assignments/port-numbers`

Connection Status

- The TCP ACK flag identifies the start of new sessions
  - ✓ The first packet in a session has this bit off
  - ✓ The first return packet and all other packets have this bit turned on
- Stop new connections by dropping packets that do not have the ACK bit turned on

Direction Is Important!



Cisco Access Control Lists

- ACLs are made up of individual rules:

list identifier　　protocol　　destination address/port

```
access-list 101 permit tcp any gt 1024 172.16.0.0 0.0.255.255 eq 23
```

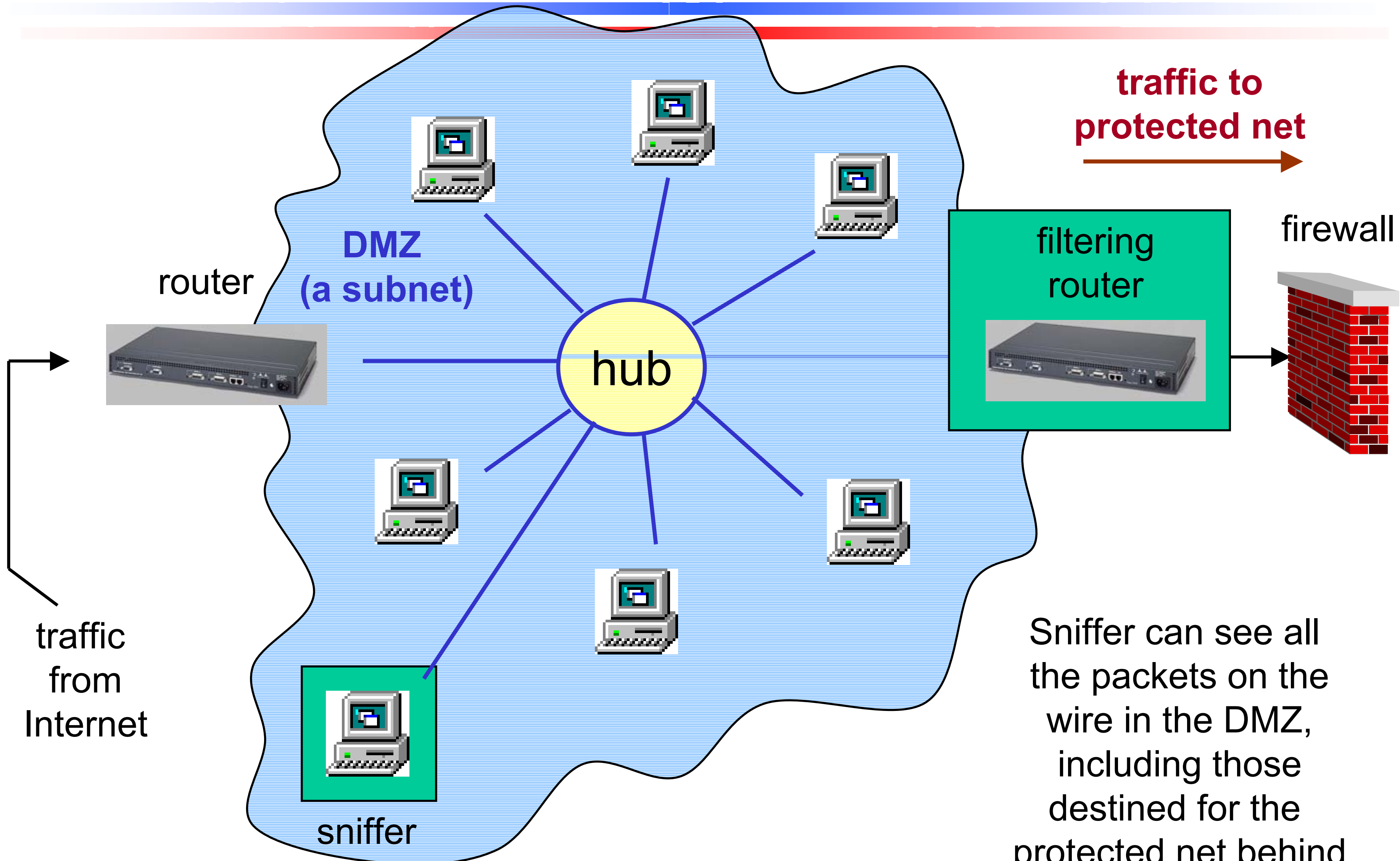
action　　src addr/port

- Use "established" at the end of any rule to match ACK flag:

```
access-list 102 permit tcp any any established
```

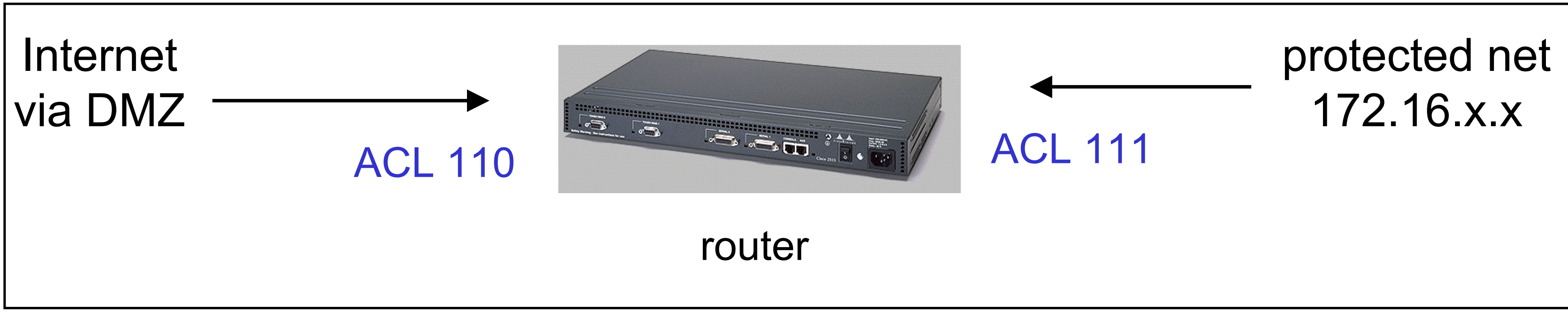

Dept. Computer Science Korea Univ.

Network Architecture



Sniffer can see all the packets on the wire in the DMZ, including those destined for the protected net behind the firewall

ACL Examples



- In this course, ACL 110 will be used to block dangerous traffic coming from the Internet towards our protected, internal network
- ACL 111 will be used to stop some traffic from escaping our internal network for the outside world
- The IP address range for the protected network will change as needed for the examples, and will not remain fixed at 172.16.x.x

Dept. Computer Science Korea Univ.

IP
ATTACKS AND DEFENSES

Land Attack

Sniffer data trace

```
10:56:32.395383 gamma.victim.net.139 > gamma.victim.net.139: S
10:56:35.145383 gamma.victim.net.139 > gamma.victim.net.139: S
10:56:36.265383 gamma.victim.net.139 > gamma.victim.net.139: S
```

General Signature

source IP = destination IP (spoofed source)

Specific Signature

source port = destination port

- TCP packet with SYN flag set
- port open on target host

Result of successful attack

target machine locks up

Land Attack

`tcpdump` filters

```
ip[12:4] = ip[16:4]
```

This filter will detect IP packets where the source and destination *addresses* are equal

```
ip[12:2] = ip[16:2]
```

This filter will detect IP packets where the source and destination *networks* are equal

Cisco ACL

If the protected internal network has addresses 10.x.x.x then
an appropriate anti-spoofing filter would be:

```
access-list 110 deny ip 10.0.0.0 0.255.255.255 any
```

Anti-Spoofing ACLs

Note: Some attacks depend on spoofing the source IP address

Block inbound traffic sourced from your own address space:

```
access-list 110 deny ip 192.200.0.0 0.0.255.255 any
```

Block outbound traffic *not* sourced from your own address space:

```
access-list 111 permit ip 192.200.0.0 0.0.255.255 any
access-list 111 deny any any
```

Block inbound traffic sourced from invalid IP addresses:

```
access-list 110 deny ip host 0.0.0.0 any
access-list 110 deny ip 127.0.0.0 0.255.255.255 any
access-list 110 deny ip 10.0.0.0 0.255.255.255 any
access-list 110 deny ip 172.16.0.0 0.15.255.255 any
access-list 110 deny ip 192.168.0.0 0.0.255.255 any
access-list 110 deny ip 192.0.2.0 0.0.0.255 any
access-list 110 deny ip 169.254.0.0 0.0.255.255 any
access-list 110 deny ip 240.0.0.0 15.255.255.255 any
```

More on Special IP Addresses

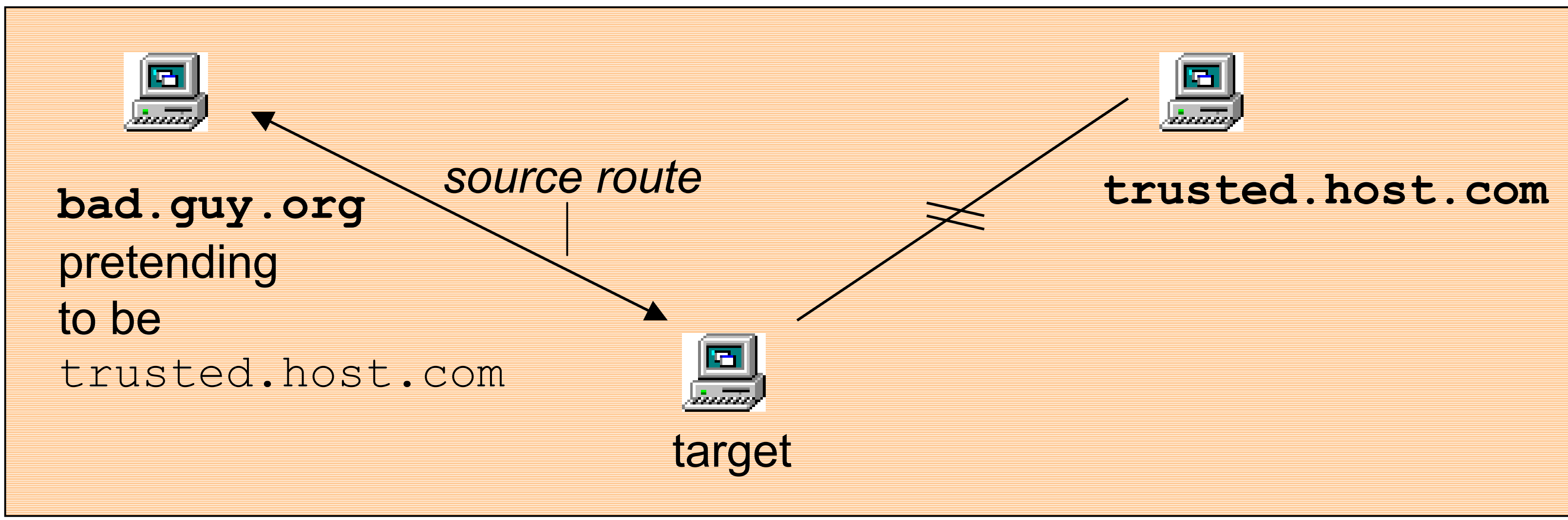
Some reserved IP addresses may appear on your network, but only under special circumstances . . .

IP address			appears as		circumstances
net	subnet	host	src	dst	
0	n/a	0	ok	never	system boot - local
0	n/a	any	ok	never	system boot - local
127	n/a	any	never	never	loopback address
-1	n/a	-1	never	ok	limited broadcast
netid	n/a	-1	never	ok	net-directed bc
netid	subid	-1	never	ok	subnet-directed bc
netid	-1	-1	never	ok	all-subnet directed bc

-1 : field is all 1 bits   **0** : field is all 0 bits

Source-Route Abuse

A **source route** specifies the IP addresses of the routers that *must* handle the datagram as it travels from source to destination



Danger*:* The target host uses the reverse of the source route provided by **bad.guy.org** in a TCP active open request. Return traffic from the target that is meant for **trusted.host.com** gets routed to **bad.guy.org**

Source-Routed Packets

- Source routing is an IP option
- The option field in the IP header is a variable length list
- If no options are set the length of the list is 0

Signature: look for packets where

IP Options are set

the only way to test for this is to see if the IP header size is larger than the no-option size of 20 bytes or 5 32-bit words

Either the Loose-Source-Route option is set or the Strict-Source-Route option is set

IP Options

- loose source routing **`(code 0x83)`**
  - ✓ specify a list of IP addresses that must be traversed by the datagram
- strict source routing **`(code 0x89)`**
  - ✓ only the IP addresses in a given list may be traversed by the datagram
- security and handling restrictions **`(see RFC 1108)`**
  - ✓ for military applications
- record route **`(code 0x07)`**
  - ✓ have each router record its IP address
- timestamp **`(code 0x44)`**
  - ✓ have each router record its IP address and time

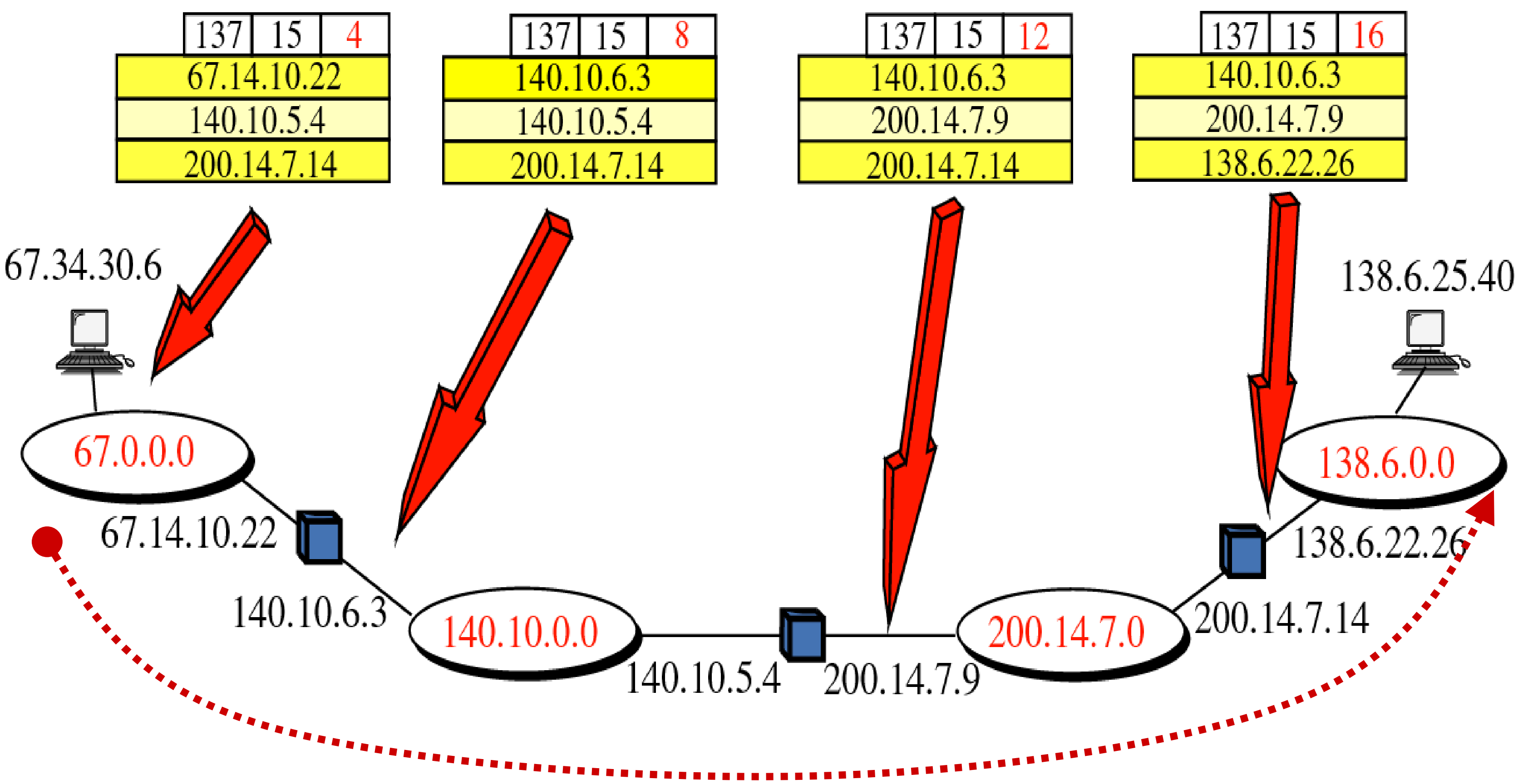
code	len	other option information

code specifies option type

options field in IP header

Dept. Computer Science Korea Univ.

Example of Source Routing



Source-Routed Packets

`tcpdump` filter

Check to see if IP options are set:

```
(ip[0:1] & 0x0f > 5)
```

and check to see if the option is loose or strict source routing:

```
and ((ip[20:1] = 0x83) or (ip[20:1] = 0x89))
```

Note: It is usually sufficient to simply check to see if any IP options have been set.

Cisco Configuration

```
no ip source-route
```

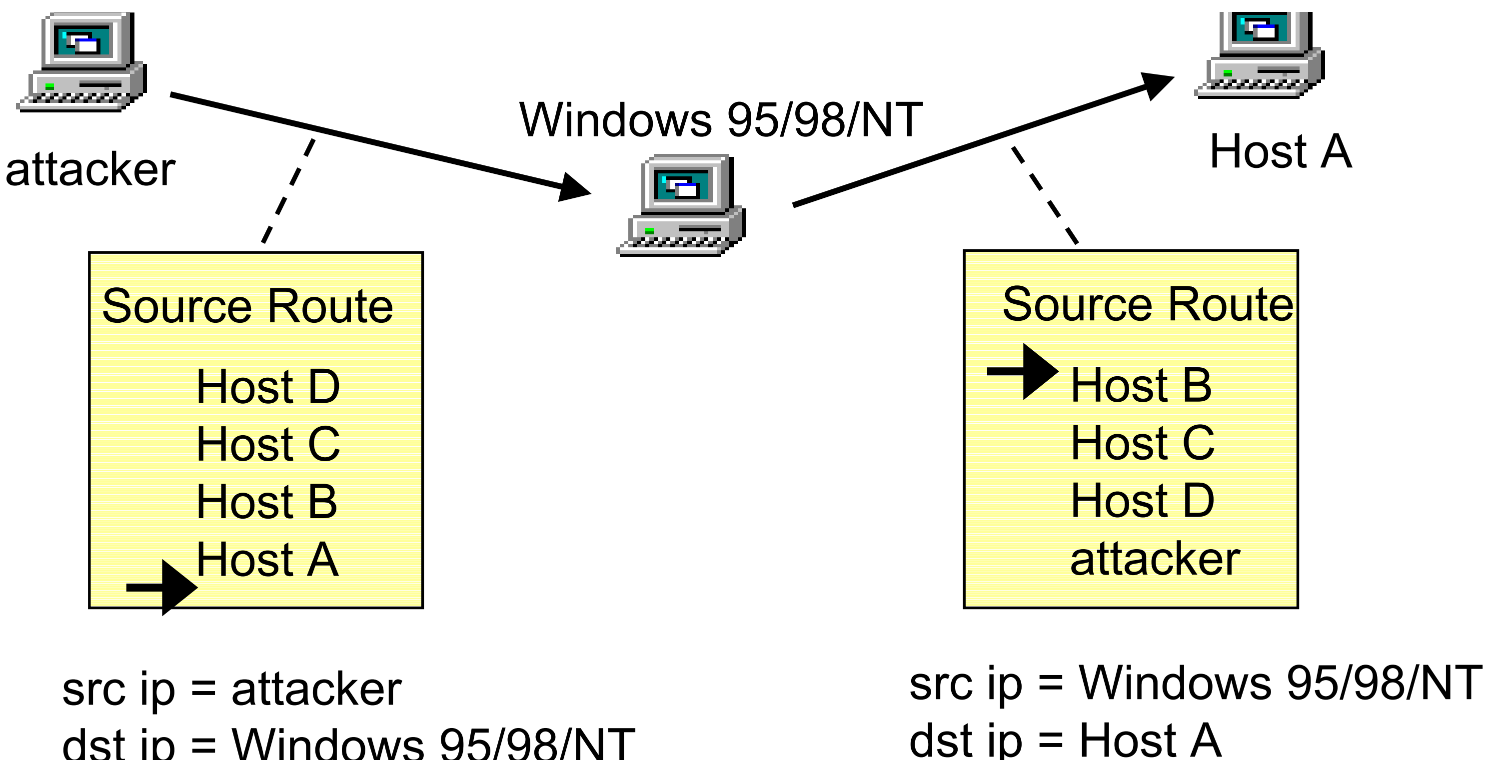
This command causes the router to drop all source-routed packets it receives.

It is a global configuration command.

Windows IP Source Routing Vulnerability

A scenario of particular concern is where an attacker manipulates the source route to gain access to the first host in the reversed route (Host A above), where this host is only accessible via the Windows machine's second (internal) interface

If an attacker sends a specially crafted source routed packet, such that the offset pointer (indicates the next IP address to be processed) points beyond the last field (indicates that the packet has reached its destination), the receiving TCP/IP stack will accept the datagram and pass it to the application layer for processing. The response to this datagram will then be delivered to the first host in the reversed source route.



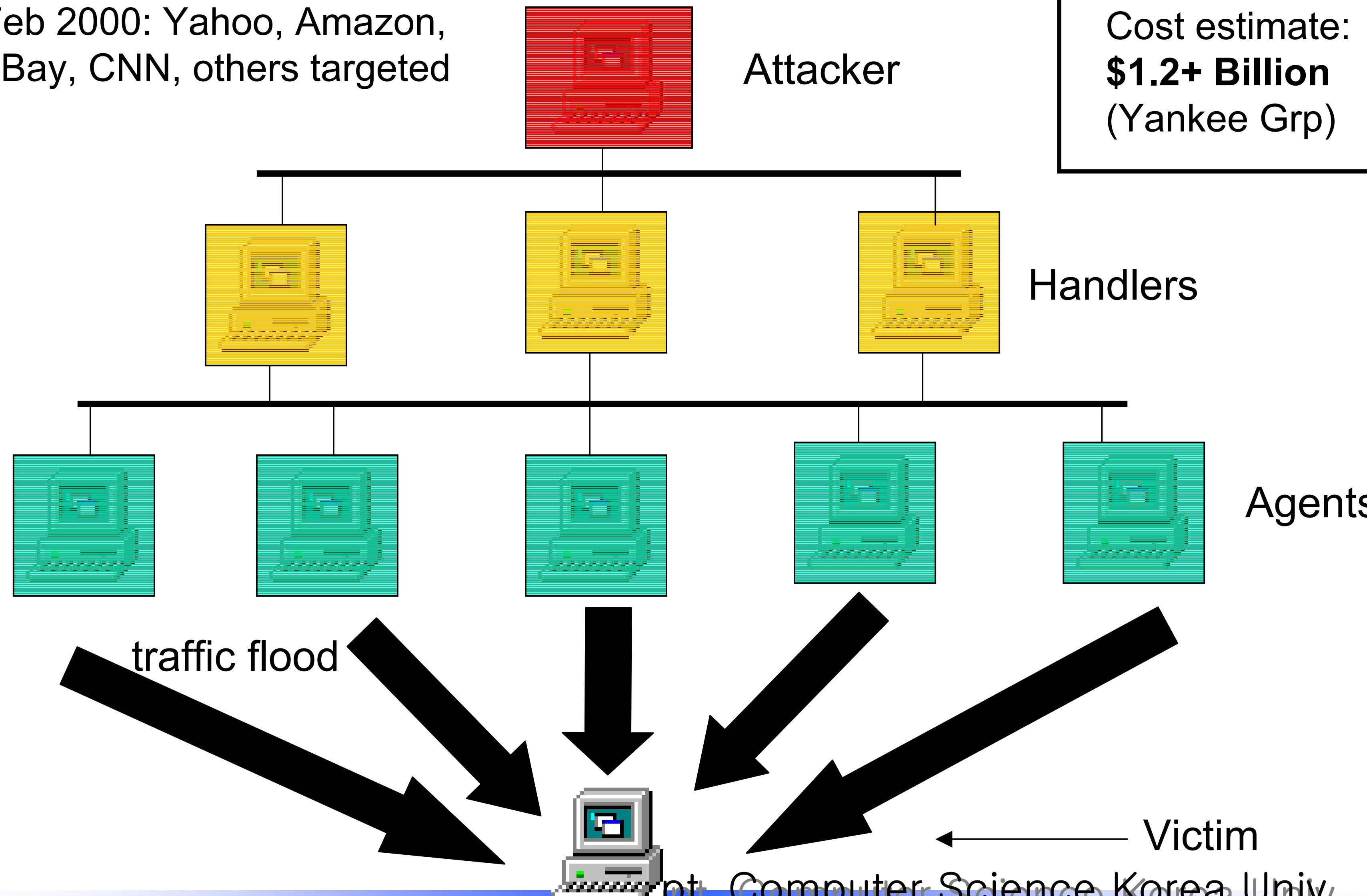
ICMP
ATTACKS AND DEFENSES

Dept. Computer Science Korea Univ.

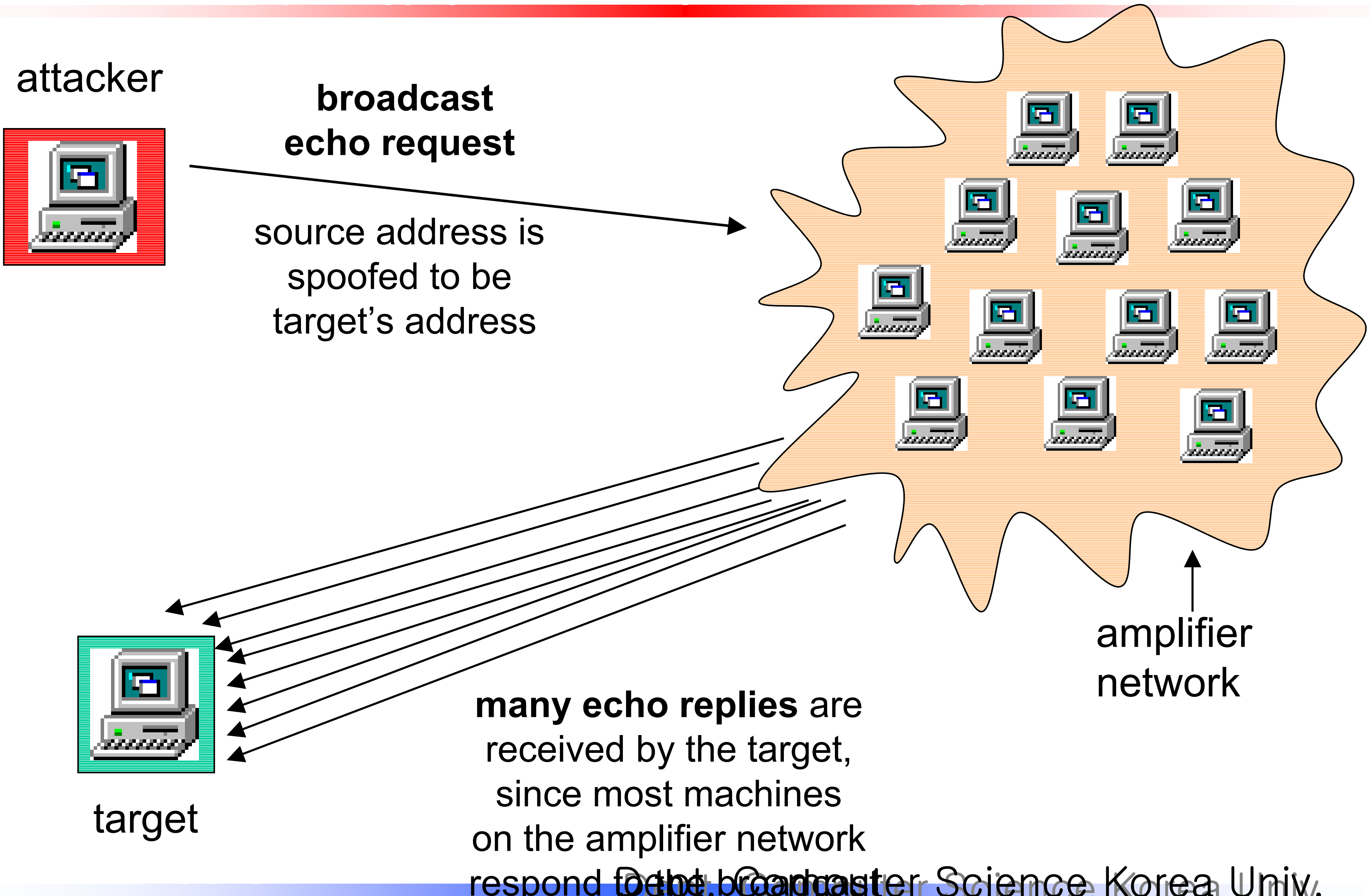
Distributed Denial of Service Attacks

Feb 2000: Yahoo, Amazon, eBay, CNN, others targeted

Cost estimate:
$1.2+ Billion
(Yankee Grp)



Smurf Attack



Smurf Attack

Denial of Service

Sniffer data trace excerpts:

```
00:00:05.327 spoofed.target.com > 192.168.15.255: icmp: echo request
00:00:05.342 spoofed.target.com > 192.168.1.255:  icmp: echo request
00:00:14.154 spoofed.target.com > 192.168.15.255: icmp: echo request
00:00:14.171 spoofed.target.com > 192.168.1.255:  icmp: echo request
00:00:19.055 spoofed.target.com > 192.168.15.255: icmp: echo request
00:00:19.073 spoofed.target.com > 192.168.1.255:  icmp: echo request
00:00:23.873 spoofed.target.com > 192.168.15.255: icmp: echo request ...

05:20:48.261 spoofed.target.com > 192.168.0.0:        icmp: echo request
05:20:48.263 spoofed.target.com > 255.255.255.255: icmp: echo request
05:21:35.792 spoofed.target.com > 192.168.0.0:        icmp: echo request
05:21:35.819 spoofed.target.com > 255.255.255.255: icmp: echo request
05:22:16.909 spoofed.target.com > 192.168.0.0:        icmp: echo request
05:22:16.927 spoofed.target.com > 255.255.255.255: icmp: echo request
05:22:58.046 spoofed.target.com > 192.168.0.0:        icmp: echo request
05:22:58.061 spoofed.target.com > 255.255.255.255: icmp: echo request ...
```

These spoofed packets are logged by a sniffing device on the “amplifier network”. Note the frequency of requests (timestamps close together) and the broadcast destination addresses.

Smurf Attack

- These filters will prevent your site from being used as the **amplifier** network.
- You will need to specify your subnet scheme via bit-masking if your network does not use the simple subnet scheme illustrated here.

Cisco Configuration (interface command):

```
no ip directed-broadcast
```

will disable the translation of directed broadcasts to physical broadcasts

Cisco ACLs:

```
access-list 110 deny ip any host 192.168.255.255
access-list 110 deny ip any host 192.168.0.0
```

explicitly denies traffic destined for broadcast addresses behind the router

`tcpdump` filters: Monitor all broadcasts except expected types

general 255 broadcasts: `ip[19] = 0xff`

general 0 broadcasts: `ip[19] = 0x00`

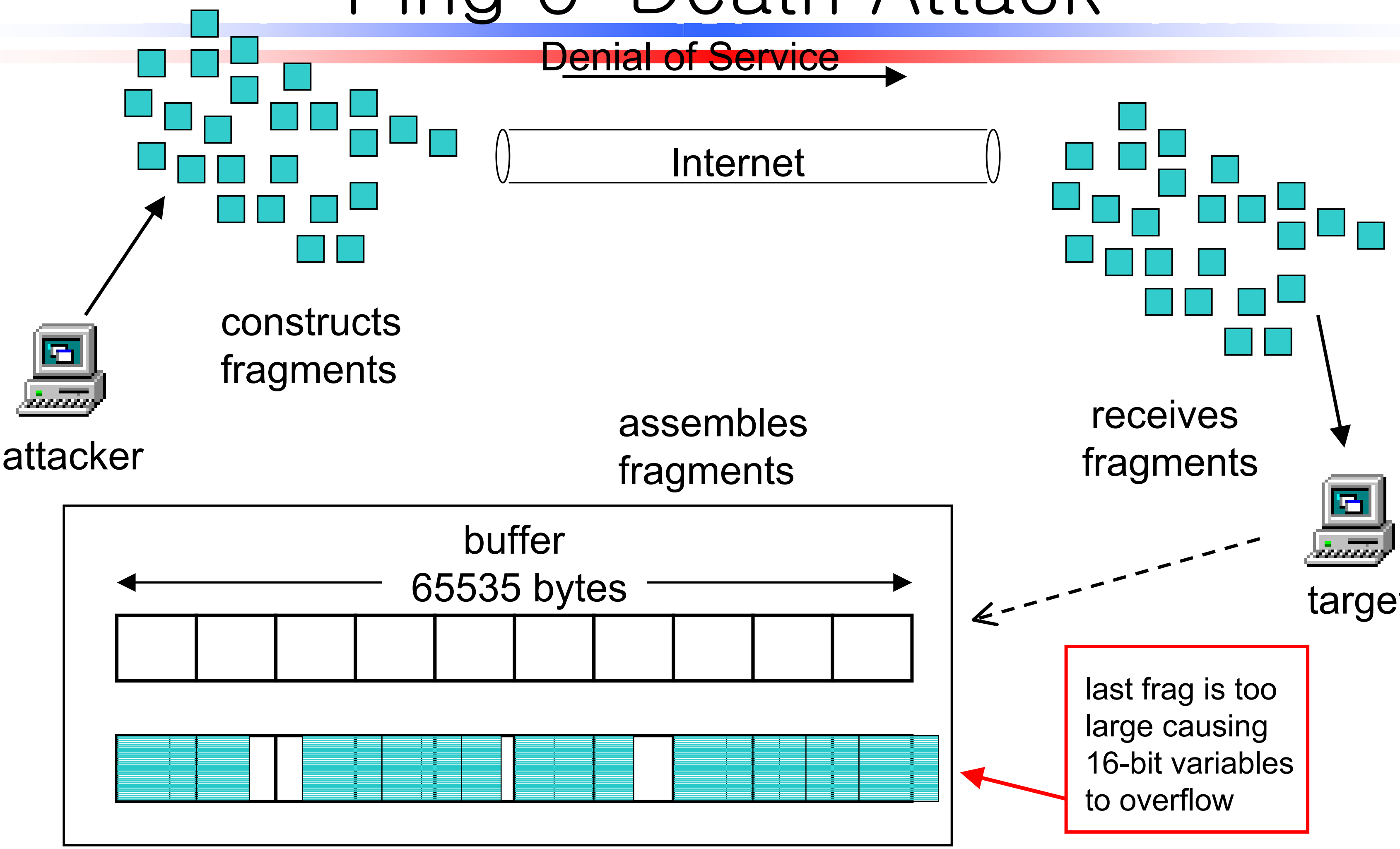
Ping o' Death Attack

- The ping-of-death causes a buffer to overflow on the target host by sending an echo request packet that is larger than the maximum IP packet size of 65535 bytes. Theoretically, any IP packet that is larger than the maximum packet size could be used, but the attack has been popularized in the form of an ICMP echo request.

- In order to generate such an "impossible packet", the attacker uses special tools to craft fragments and send them to the target.

- When the target host receives these fragments and tries to reassemble them or process the reassembled datagram, its operating system may crash or hang

Ping o' Death Attack



Attacker sends a ping packet that is larger than the maximum IP packet size of 65535 bytes (380 + 65360 = 65740).

Ping o' Death Attack

Denial of Service

All fragment filter (MF=1 or offset>0):

```
(ip[6:1] & 0x20 != 0) or (ip[6:2] & 0x1fff != 0)
```

Final fragment filter (MF=0 and offset>0):

```
(ip[6:1] & 0x20 = 0) and (ip[6:2] & 0x1fff != 0)
```

Ping o' Death filter: will the fragments assemble so that the total datagram size is greater than 65535 bytes?

```
(  ip[2:2] -                    #IP total length (bytes)
   ((ip[0:1]&0x0f)*4) +         #IP header length (bytes)
   ((ip[6:2]&0x1fff)*8)         #fragment offset (bytes)
) > 65535
```

Impossible Fragmentation

Recall: The offset field is only 13 bits long (instead of 16 bits long).

- The offset value must be multiplied by 8 to compute byte counts.
- Offsets can only be on 8 byte boundaries.
- **Naturally occurring non-final fragments will always carry data payloads that are multiples of 8 bytes in length .**

```
12:43:58.431 big.pinger.org > www.mynetwork.net: (frag 4321:380@376+)
12:43:58.431 big.pinger.org > www.mynetwork.net: (frag 4321:380@3800+)
12:43:58.431 big.pinger.org > www.mynetwork.net: (frag 4321:380@4176+)
```

Not natural!

Detect impossible fragments:

```
(ip[6:1]&0x20 != 0) and
( (ip[2:2] - ((ip[0:1]&0x0f)*4)) & 0x7 != 0)
```

ICMP Redirect Attack

- ICMP Redirect 메시지는 하나의 네트워크에 여러 개의 라우터가 있을경우, 호스트가 패킷을 올바른 라우터에게 보내도록 알려주는 역할을 한다.
- 공격자는 이를 이용하여 다른 세그먼트에 있는 호스트에게 위조된 ICMP Redirect 메시지를 보내 공격자의 호스트로 패킷을 보내도록 하여 스니핑하는 방법이다.

ARP
ATTACKS AND DEFENSES

Dept. Computer Science Korea Univ.

ARP Redirection Attack

- "ARP Redirect" 공격은 위조된 arp reply를 보내는 방법을 사용
- 공격자 호스트가 "나의 MAC 주소가 라우터의 MAC 주소이다" 라는 위조된 arp reply를 브로드캐스트로 네트워크에 주기적으로 보내어, 스위칭 네트워크상의 다른 모든 호스트들이 공격자 호스트를 라우터로 믿게 끔 함
- 결국 외부 네트워크와의 모든 트래픽은 공격자 호스트를 통하여 지나가게 되고 공격자는 스니퍼를 통해여 필요한 정보를 도청할 수 있게 됨
- 이때 공격호스트는 IP Forwarding기능을 설정해야 공격 호스트로 오는 모든 트래픽을 원래의 게이트 웨이로 포워딩 해줄수 있음.

ARP Spoofing Attack

- ARP redirect와 비슷한 공격 방법으로 다른 세그먼트에 존재하는 호스트간의 트래픽을 스니핑하고자 할 때 사용
- 공격자는 자신의 MAC 주소를 스니핑하고자 하는 두 호스트의 MAC주소로 위장하는 arp reply 패킷을 네트워크에 흘림.
- 이런한 arp reply를 받은 두 호스트는 자신의 arp cache를 갱신하게 되고, 두 호스트간의 연결이 일어날때 공격자의 호스트 MAC주소를 사용하게 된다.
- 결국 두 호스트간의 모든 트래픽은 공격자가 위치한 세그먼트로 들어오게 된다.
- 이러한 경우 arp redirect 공격과 마찬가지로 공격자 호스트로 넘어오는 트래픽을 본래의 호스트로 reply 해주어야 한다.

UDP
ATTACKS AND DEFENSES

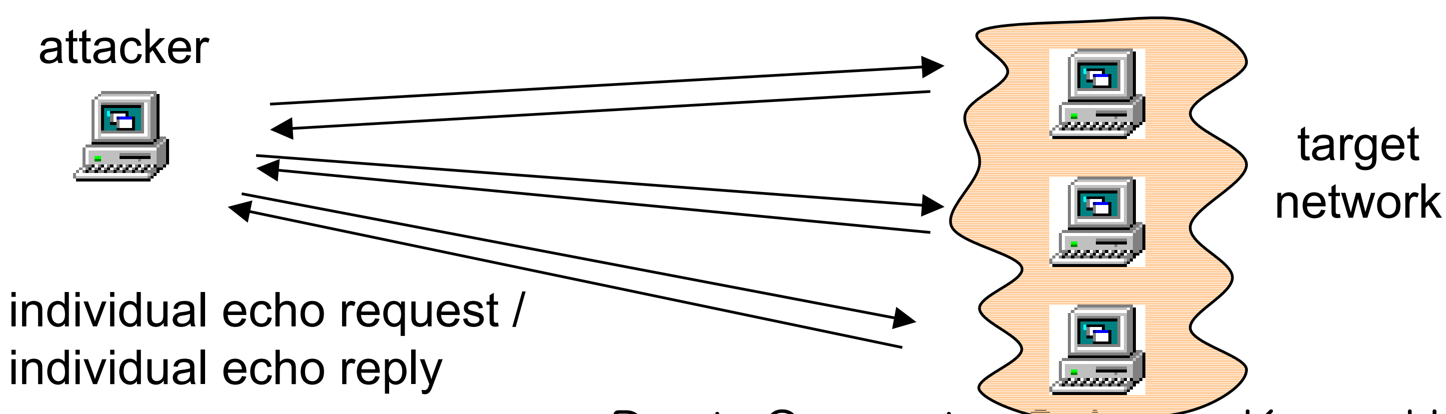
Dept. Computer Science Korea Univ.

"Small" Services

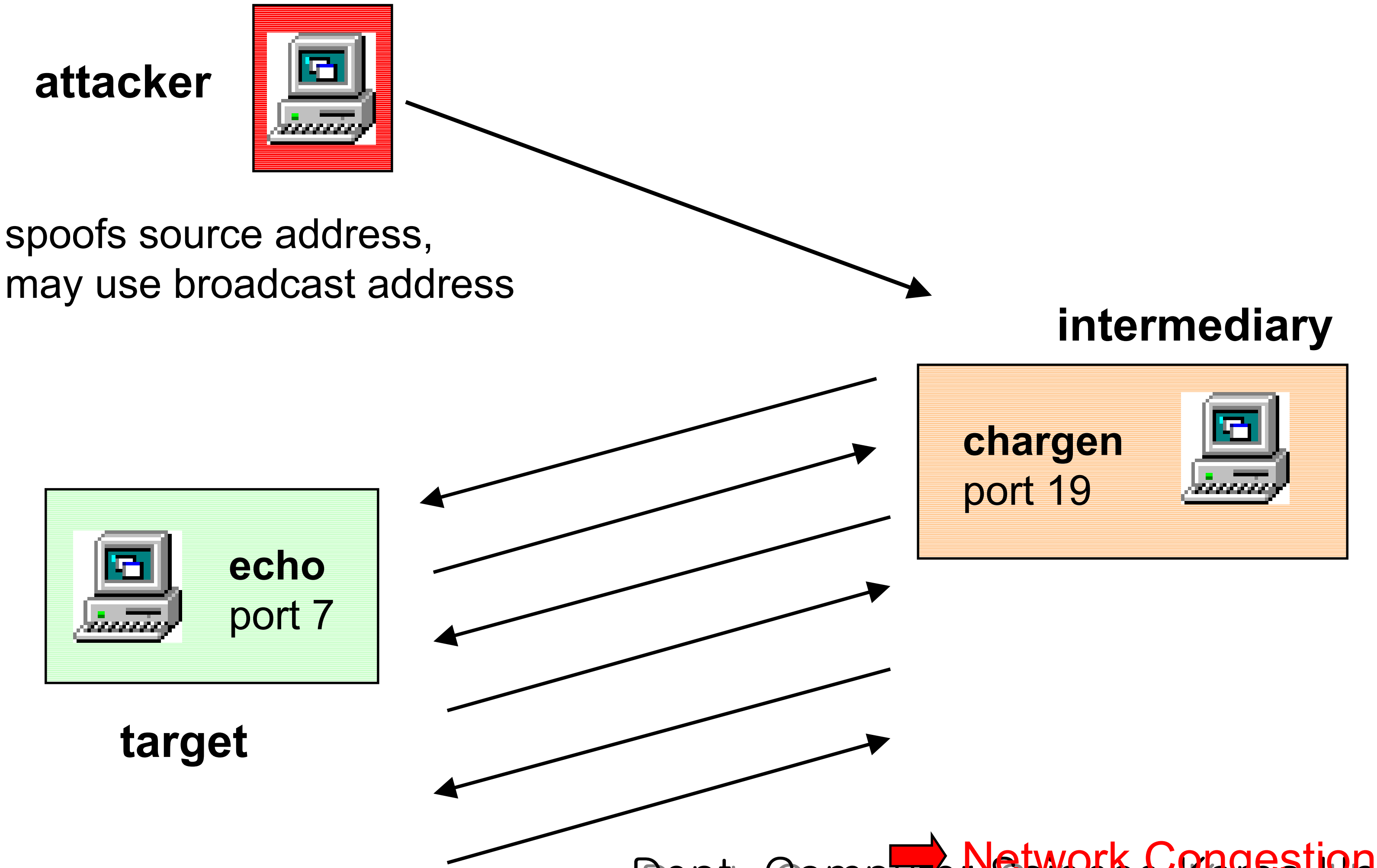
service name	port number	service description
echo	7/udp 7/tcp	server echoes the data that the client sends
discard	9/udp 9/tcp	server silently discards whatever data the client sends
daytime	13/udp 13/tcp	server returns the time and date in a human readable format
chargen	19/udp 19/tcp	server responds with a datagram containing a string of ascii characters server sends a continual stream of characters until the connection is terminated by the client
time	37/udp 37/tcp	server returns the time as a 32-bit binary number

Network Mapping

- Similar to ICMP pings, UDP echo requests may be used to map out networks. A UDP packet sent to the echo port on a target host causes the target host to echo back whatever data was contained in the payload of the probe packet.
- Queries directed to the chargen, time, or daytime service ports may function just as well to generate a network map.
- Each of these services, echo included, will automatically respond to a probe by generating a packet, which is the basic property of a "ping".



UDP Diagnostic Port Attack



UDP Diagnostic Port Attack

`tcpdump` **filter:**

```
udp and ( ((port 7) and (port 13)) or
          ((port 7) and (port 19)) or
          ((port 7) and (port 37)) or
          ((port 13) and (port 19)) or
          ((port 13) and (port 37)) or
          ((port 19) and (port 37)) or
          ((src port 7) and (dst port 7)) or
          ((src port 13) and (dst port 13)) or
          ((src port 19) and (dst port 19)) or
          ((src port 37) and (dst port 37)) )
```

Or, if you are not allowing the outside world to access these services (a good idea), a simpler filter will do the job:

```
udp and (port 7 or port 13 or port 19 or port 37)
```

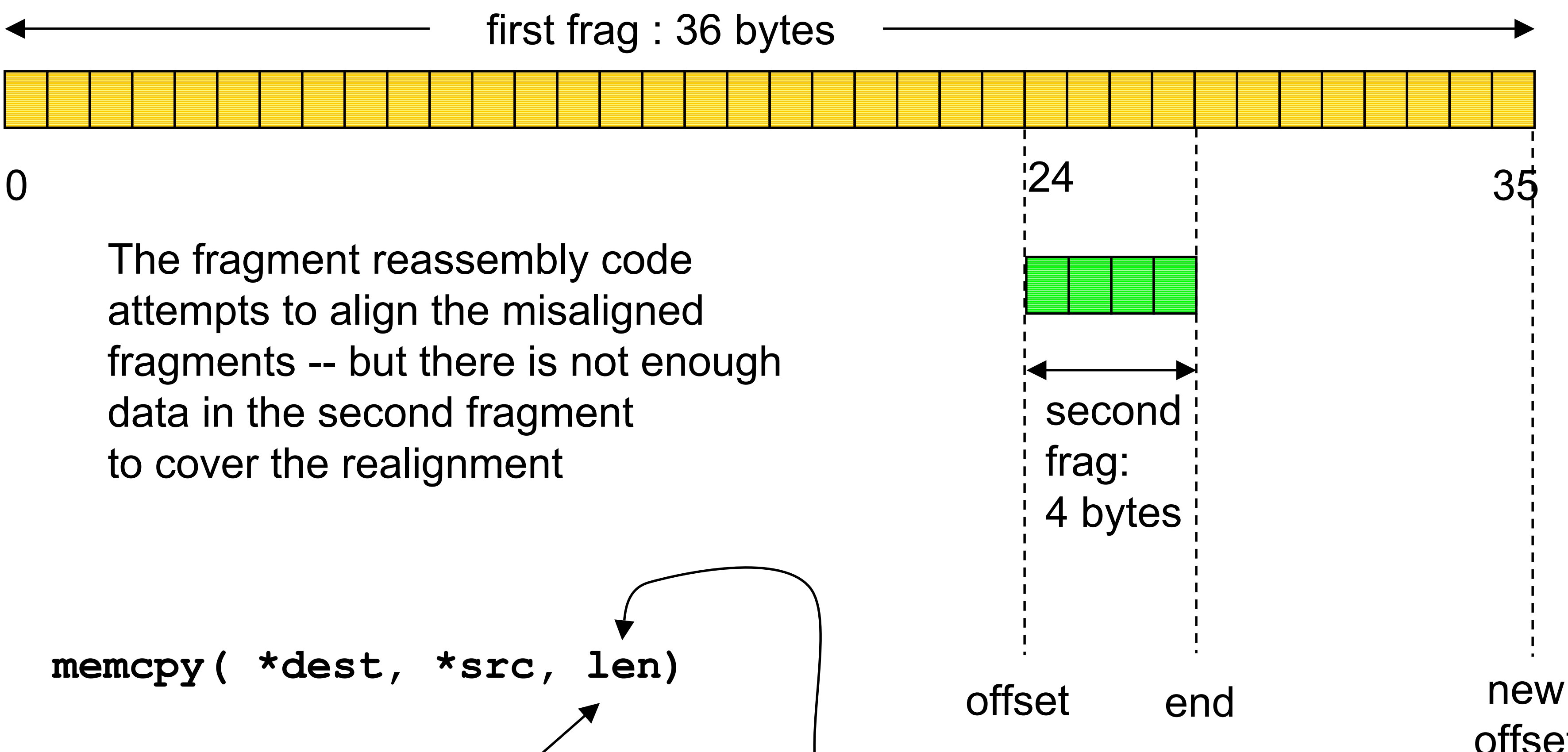
Teardrop Attack

Denial of Service

- This attack exploits a weakness in the fragment reassembly code of some operating systems. Specifically, teardrop sends two fragments that do not overlap properly, causing some machines to crash when they try to reassemble them
- When the kernel receives the second fragment it compares the offset of the second fragment with the end of the first fragment. Since the offset of the second fragment is less than the end of the first fragment, a fragment alignment routine is run.
- The problem comes when the program passes this negative number to a `memcpy` operation that is expecting an unsigned value. The negative number gets interpreted as a very large positive number and the kernel winds up trying to copy pages and pages of data. This results in a reboot or a halt, depending on the amount of physical memory on the victim.

Teardrop Attack

Denial of Service



The fragment reassembly code attempts to align the misaligned fragments -- but there is not enough data in the second fragment to cover the realignment

```
memcpy( *dest, *src, len)
```

unsigned int or unsigned long

len = end - newoffset < 0

Teardrop Attack

Denial of Service

```
10:25:48 attacker.org.45959 > target.net.53: udp 28 (frag 242:36@0+)
10:25:48 attacker.org > target.net: (frag 242:4@24)
```

Teardrop Signature
2 UDP fragments
-first frag: 0+ fragment with payload of size N
-second frag: final fragment with offset < N and payload size < (N-offset)

General Properties
-Fragments do not overlap correctly
-Non-final fragment is carrying data that is not multiple of 8 bytes in length

Result of a successful attack
target machine reboots or halts - depending on the amount of physical memory

`tcpdump` filter: (finds all fragmented UDP)

```
udp and ((ip[6:1] & 0x20 != 0) or (ip[6:2] & 0x1fff != 0))
```

UDP Portscanning

- Portscanners attempt to connect to many ports on a target host, in order to determine which ports are open (have listeners running)
- Ports are queried very quickly in a seemingly random order
- The UDP packets carry no data (only IP and UDP headers are sent)

Closed ports respond with an ICMP "port unreachable" error message

```
11:40:36.445995 attacker.org.53160 > target.com.516: udp 0
11:40:36.455995 target.com > attacker.org:
                 icmp: 172.21.165.150 udp port 516 unreachable
```

open port do not respond at all

```
11:40:36.855995 attacker.org.53160 > target.com.514: udp 0
11:40:37.005995 attacker.org.53161 > target.com.514: udp 0
11:40:37.165995 attacker.org.53160 > target.com.514: udp 0
11:40:37.255995 attacker.org.53161 > target.com.514: udp 0
```

- The open ports are taken to be those that do not respond to probing
- Recall that UDP is an unreliable protocol (as is ICMP)
- This unreliability makes UDP portscanning difficult when the attacker and the target are far apart (packets are more likely to get lost along the way)

TCP
ATTACKS AND DEFENSES

Dept. Computer Science Korea Univ.

Scanning for a Specific TCP Service

- An attacker probes many machines on the target network in order to find a host running a particular vulnerable server

- Some services that are often targeted are
  - ✓ HTTP : port 80
  - ✓ SOCKS : port 1080
  - ✓ IMAP : port 143
  - ✓ DNS :port 53
  - ✓ POP3 : port 110

Scanning with SF Packets

```
16:36:06.54 pop3.net.0 > 192.168.26.203.110: SF 1681588224:1681588224
16:36:06.54 pop3.net.0 > 192.168.18.84.110: SF 1681588224:1681588224
16:36:06.57 pop3.net.0 > 192.168.43.254.110: SF 1681588224:1681588224
16:36:06.58 pop3.net.0 > 192.168.24.209.110: SF 1681588224:1681588224
16:36:06.60 pop3.net.0 > 192.168.17.197.110: SF 1681588224:1681588224
16:36:06.62 pop3.net.0 > 192.168.16.181.110: SF 1681588224:1681588224
16:36:06.64 pop3.net.0 > 192.168.20.65.110: SF 1681588224:1681588224 ...
```

```
01:56:58.62 dns.edu.0 > 192.168.93.0.53: SF 2216558592:2216558592
01:56:58.63 dns.edu.0 > 192.168.93.1.53: SF 2216558592:2216558592
01:56:58.65 dns.edu.0 > 192.168.93.2.53: SF 2216558592:2216558592
01:56:58.67 dns.edu.0 > 192.168.93.3.53: SF 2216558592:2216558592
01:56:58.69 dns.edu.0 > 192.168.93.4.53: SF 2216558592:2216558592
01:56:58.71 dns.edu.0 > 192.168.93.5.53: SF 2216558592:2216558592
01:56:58.73 dns.edu.0 > 192.168.93.6.53: SF 2216558592:2216558592 ...
```

- These packets are specially crafted, as evidenced by the impossible flag combination: SYN and FIN flags set simultaneously
- Note the hardwired source ports and sequence numbers
- The destination IP addresses may be orderly or randomized
- The impossible flag setting may elude some firewalls and ID systems

Detecting Network Scans

If you know that no hosts on your network are running a particular service, it is a good idea to monitor any traffic directed to that well-known service port

```
tcp and (dst port 143 or dst port 1080 or dst port 110)
```

If only a few hosts are running a particular service, monitor any traffic involving the service but not the identified hosts

```
tcp and
(
   ((dst port 80) and (not host www.mynet.com))
   or
   ((dst port 53) and (not host ns.mynet.com))
)
```

Also you should try to detect packets with impossible flag settings such as SYN-FIN, but more on this later ….

Filtering on TCP Flags

`tcpdump` allows filtering on each of the individual tcp flags (Cisco routers do not)

x	x	U	A	P	R	S	F

← byte 13 of the tcp header

SYN flag set:	`tcp[13] & 0x02 != 0`
ACK flag set:	`tcp[13] & 0x10 != 0`
RST flag set:	`tcp[13] & 0x04 != 0`
FIN flag set:	`tcp[13] & 0x01 != 0`
PSH flag set:	`tcp[13] & 0x08 != 0`
URG flag set:	`tcp[13] & 0x20 != 0`
no flags set:	`tcp[13] & 0x3f = 0`

Therefore, if we want to detect active open connections to the IMAP port we use:

```
tcp and (dst port 143)
    and (tcp[13] & 0x02 != 0) and (tcp[13] & 0x10 = 0)
```

Blocking Network Scans

If it is sufficient to allow your users to initiate connections "outbound", but not have connections initiated "inbound" the *established* ACL should be used :

```
access-list 110 permit tcp any 192.168.0.0 0.0.255.255 established
```

This rule requires that the "ACK" bit be set on any inbound TCP packets. The ACK bit will be set on all but the "active open" connections

Otherwise you may block all traffic destined to a particular port

```
access-list 110 deny tcp any 192.168.0.0 0.0.255.255 eq 1999
```

Or you may specify the traffic that is allowed to particular hosts and deny all else (remember order is important)

```
access-list 110 permit tcp any host 192.168.2.2 eq 53
access-list 110 permit tcp any host 192.168.1.1 eq 53 established
access-list 110 deny tcp any 192.168.0.0 0.0.255.255 eq 53
```

What are attackers hoping to accomplish?

Let us diverge for a moment and consider what an attacker might do to a vulnerable server ….

Vulnerabilities exist to allow an attacker to do the following remotely on many servers:

- execute arbitrary commands on the server
- gain unauthorized access to server files or directories
- gain shell access at the privilege level of the server process (often root)
- deny service to regular server clients by consuming server resources
- corrupt information that the server needs (e.g. a nameserver cache)

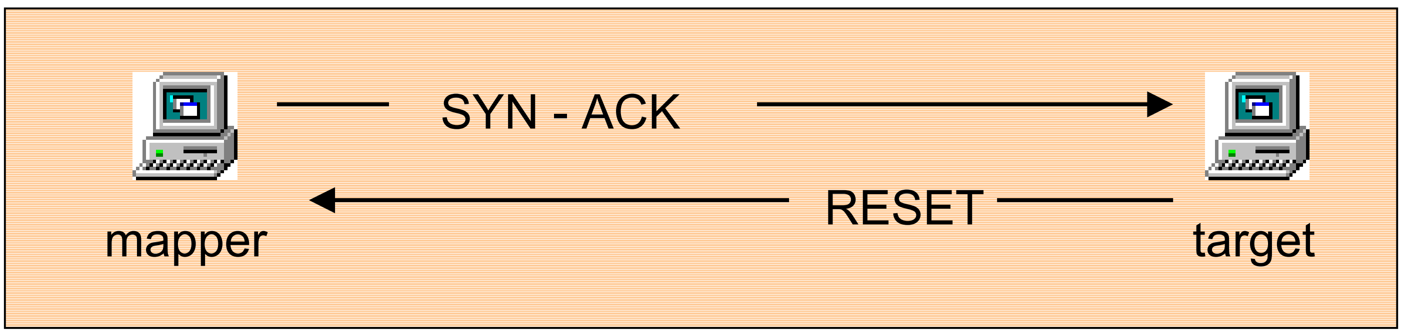
Bottom line: *protect your servers!*

Most DDoS zombies were compromised because they were running known vulnerable software -- patches had been available for years...

Stealthy Host Discovery using TCP

The first step in finding a vulnerable host is discovering which hosts are alive. We have seen host discovery using ICMP and UDP echo requests, similarly TCP echo requests could be employed.

But consider the following stealthy technique:



If a SYN-ACK packet is sent to a host, the host will return a RESET, regardless of whether the target port is open or not.

Stealthy Host Discovery using TCP

a la **nmap** – Detecting

Recall

- The ACK number is valid on the following packets: SA, A, FA, RA
- A SYN consumes one sequence number =>
  - the ACK number on a SA packet can never be < 1
  - the ACK number on an A packet can never be < 1
  - the ACK number on a FA packet can never be < 1
- **nmap** uses ACK-only packets with the ACK number set to zero

Signature

- ACK flag is set, RST flag is not set
- ACK number is set equal to zero

```
(tcp[13] & 0x10 != 0) and (tcp[13] & 0x04 = 0)
    and (tcp[8:4] = 0)
```

detect pattern

```
20:32:46.07 attacker.37827 > 172.21.165.101.80: . ack 0
20:32:46.35 attacker.37827 > 172.21.165.108.80: . ack 0
20:32:46.48 attacker.37827 > 172.21.165.180.80: . ack 0
20:32:46.62 attacker.37827 > 172.21.165.181.80: . ack 0
20:32:46.75 attacker.37827 > 172.21.165.186.80: . ack 0
```

TCP Portscanning

- An attacker queries many ports on a target machine to determine which ports are open (live) and which ports are closed (dead)

- An open port indicates that a service is offered

- If an attacker knows what services are offered, he/she may be able to guess what security vulnerabilities are available to exploit

nmap

- a freely available scanner that runs on UNIX operating systems.
- This excellent scanner allows us to demonstrate several portscanning techniques and methods for detecting the scans.
- Try it yourself for free:
  `http://www.insecure.org/nmap`

TCP Portscanning

connect requests

open port

```
scanner.8831 > target.514: S 3209086149:3209086149(0)
target.514 > scanner.8831: S 1346112000:1346112000(0) ack 3209086150
scanner.8831 > target.514: . ack 1346112001
scanner.8831 > target.514: F 3209086150:3209086150(0) ack 1346112001
target.514 > scanner.8831: . ack 3209086151
target.514 > scanner.8831: F 1346112001:1346112001(0) ack 3209086151
scanner.8831 > target.514: . ack 1346112002
```

closed port

```
scanner.12441 > target.516: S 1573861375:1573861375(0)
target.516 > scanner.12441: R 0:0(0) ack 1573861376
```

- If a port is open, the regular three-way handshake is completed, the scanner then gracefully closes the connection by sending an active close request
- If a port is closed, the target responds to the connection request with a RESET
- Most target hosts will log these connection attempts

TCP Portscanning

SYN packets

open port

```
scanner.52894 > target.514: S 3900690976:3900690976(0)
target.514 > scanner.52894: S 1379776000:1379776000(0) ack 3900690977
scanner.52894 > target.514: R 3900690977:3900690977(0)
```

closed port

```
scanner.52894 > target.516: S 3900690976:3900690976(0)
target.516 > scanner.52894: R 0:0(0) ack 3900690977
```

- In this technique, the three-way handshake is not completed. For this reason, it is often called *half-open* scanning
- A SYN is sent, and the scanner waits for a response
- If an open target port returns a SYN-ACK, the OS on the scanning host will immediately tear down the connection, since it did not issue the connect request
- Target hosts may not log these probes since the handshake is never completed

SYN and Connect Scans

Detecting and Blocking

Cisco ACL

The *established* filter will effectively thwart both of these scanning techniques

```
access-list 110 permit any 172.21.0.0 0.0.255.255 established
```

If you cannot implement this across the board, implement it on as many ports as you can (or even better - deny *all* access to some ports)

`tcpdump` filter

Monitor any inbound SYN connections to ports you are not expecting traffic on …. the scans will drop out

```
tcp and
(tcp[13] & 0x02 != 0) and (tcp[13] & 0x10 = 0)
and (not dst port 53) and (not dst port 80)
and (not dst port 25) and (not dst port 21)
```

RFC 793

TCP Functional Specification

Order in which checks are performed

- If a port is closed
  - ✓ if the incoming segment contains a RESET: it is discarded
  - ✓ if the incoming segment does *not* contain a RESET:
    - a RESET is sent in response

- If a port is open and in the *listen* state
  - ✓ if the incoming segment contains a RESET: it is discarded
  - ✓ if the incoming segment contains an ACK:
    - a RESET is sent in response
  - ✓ if the incoming segment contains a SYN
    - a SYN ACK is set in response
  - ✓ if none of the above are true:
    - the segment is discarded (*this should never happen*)

TCP Portscanning

- Segments containing a RESET are always discarded (not useful)

- Segments containing an ACK always generate a RESET (useful as a "ping")

- ***Answer to question:***
Non-SYN packets that do not contain a RESET or an ACK could be used to portscan
  - ports that do not respond to probing with a RESET are assumed to be open

that is, assuming that the target OS follows the RFC specifications ….

For a Linux operating system:

Flags	**live port**	**dead port**
None	**0**	**RA**
F	**0**	**RA**
S	SA	RA
SF	SFA	RA
R	0	0
RF	0	0
SR	0	0
SRF	0	0
A	R	R
FA	R	R
SA	R	R
SFA	R	R
RA	0	0
RFA	0	0
SRA	0	0
SFRA	0	0

TCP Portscanning

Non-SYN-ACK-RST packets

The response is as we expected for most targets ….
(only the responses for the FIN scan are shown, the response to null and xmastree packets is identical)

open port

```
11:24:52.545 scanner.org.57298 > target.com.514: F 0:0(0)
11:24:52.655 scanner.org.57299 > target.com.514: F 0:0(0)
11:24:53.445 scanner.org.57298 > target.com.514: F 0:0(0)
11:24:53.535 scanner.org.57299 > target.com.514: F 0:0(0)
```

closed port

```
11:24:52.495 scanner.org.57298 > target.com.516: F 0:0(0)
11:24:52.495 target.com.516 > scanner.org.57298: R 0:0(0) ack 0
```

- Some operating systems do not follow the RFC and send RESETs from the open ports as well as from the closed ports
- Systems not susceptible to this type of portscanning (do not follow RFC):
  MS Windows, Cisco, BSDI, HP/UX, MVS, IRIX

TCP Portscanning

Non-SYN-ACK-RST packets – Detecting and Blocking

Flags	live port	dead port
None	**0**	**RA**
F	**0**	**RA**
S	SA	RA
SF	**SFA**	**RA**
R	0	0
RF	**0**	**0**
SR	**0**	**0**
SRF	**0**	**0**
A	R	R
FA	R	R
SA	R	R
SFA	**R**	**R**
RA	0	0
RFA	**0**	**0**
SRA	**0**	**0**
SFRA	**0**	**0**

- We should detect any "impossible" packet, these are packets where the following flag combinations are present:
  - no flags
  - FIN only
  - SYN FIN
  - RST FIN
  - SYN RST
- The allowed packets are then
  - S only
  - SYN ACK
  - ACK only
  - FIN ACK
  - RST only
  - RST ACK

TCP Portscanning

Non-SYN-ACK-RST packets – Detecting and Blocking

SYN flag set:	`tcp[13] & 0x02 != 0`
ACK flag set:	`tcp[13] & 0x10 != 0`
RST flag set:	`tcp[13] & 0x04 != 0`
FIN flag set:	`tcp[13] & 0x01 != 0`
no flags set:	`tcp[13] & 0x3f = 0`

No flags are set

```
tcp and (tcp[13] & 0x3f = 0)
```

FIN flag is set and ACK flag is not

```
tcp and (tcp[13] & 0x01 != 0) and (tcp[13] & 0x10 = 0)
```

SYN flag and FIN flag are set simultaneously

```
tcp and (tcp[13] & 0x02 != 0) and (tcp[13] & 0x01 != 0)
```

RST flag and FIN flag are set simultaneously

```
tcp and (tcp[13] & 0x04 != 0) and (tcp[13] & 0x01 != 0)
```

SYN flag and RST flag are set simultaneously

```
tcp and (tcp[13] & 0x02 != 0) and (tcp[13] & 0x04 != 0)
```

Linux Blind TCP Spoofing

Another reason to filter for impossible flag combinations

- Allows remote attackers to pass data to the application layer without completing the three-way handshake
- Attacker can spoof the packets so that they appear to be sourced from a host that the target trusts
- No need for the attacker to guess the sequence numbers sent by the server

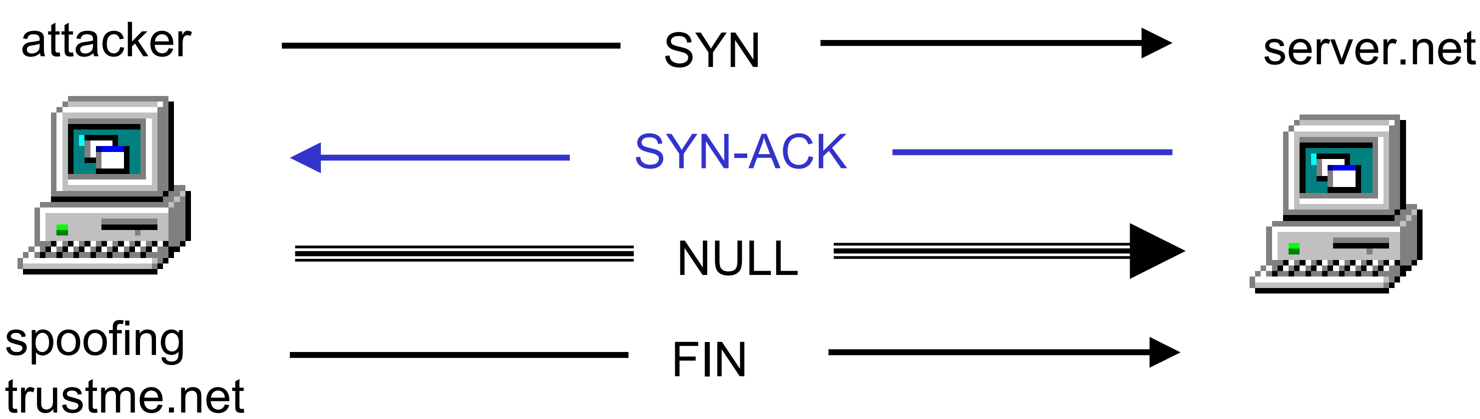
Properties of the vulnerable Linux operating system:

- The ACK number is only verified on incoming packets that have the ACK flag set
- Linux queues data from TCP segments without ACK information prior to the completion of the three way handshake
- Linux passes data to the application layer upon receipt of a packet containing the FIN flag regardless of whether the connection has been established

Linux Blind TCP Spoofing

Another reason to filter for impossible flag combinations

```
21:49:25 trustme.net.9999 > server.net.2222: S 20985:20985(0)
21:49:25 server.net.2222 > trustme.net.9999: S (0) ack 20986
21:49:25 trustme.net.9999 > server.net.2222: . 20986:20994(8)
21:49:25 trustme.net.9999 > server.net.2222: F 20994:20994(0)
```



Data contained in the NULL packet is passed to the application layer

TCP Portscanning

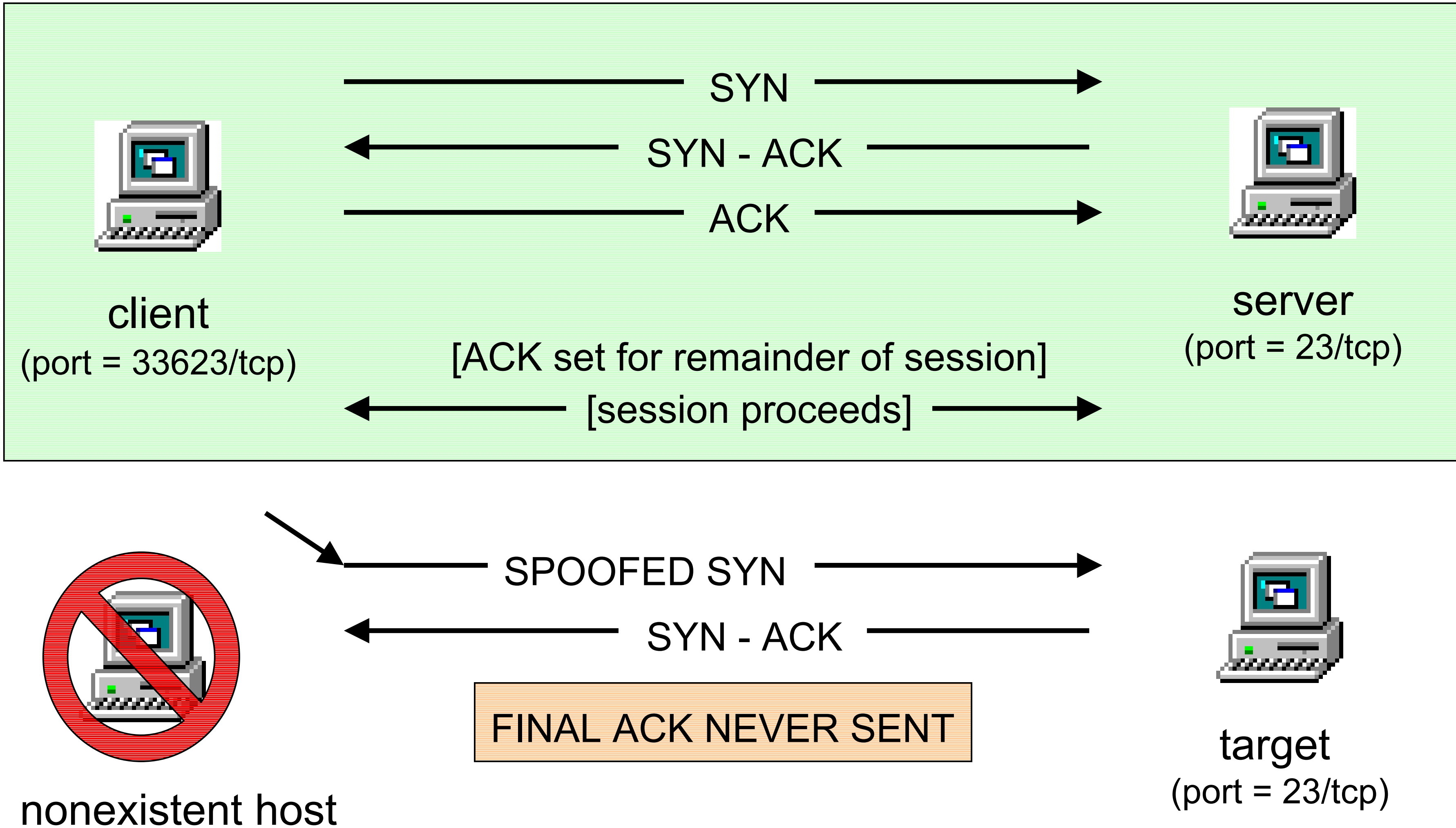
Scanning with Decoy Addresses

```
06:43:55 10.2.2.2.57536 > target.328: S 1496167267:1496167267(0)
06:43:55 10.3.3.3.57536 > target.328: S 1496167267:1496167267(0)
06:43:55 scanner.57536 > target.328: S 1496167267:1496167267(0)
06:43:55 10.4.4.4.57536 > target.328: S 1496167267:1496167267(0)
06:43:55 10.5.5.5.57536 > target.328: S 1496167267:1496167267(0)
06:43:55 10.2.2.2.57536 > target.994: S 1496167267:1496167267(0)
06:43:55 10.3.3.3.57536 > target.994: S 1496167267:1496167267(0)
06:43:55 scanner.57536 > target.994: S 1496167267:1496167267(0)
06:43:55 10.4.4.4.57536 > target.994: S 1496167267:1496167267(0)
06:43:55 10.5.5.5.57536 > target.994: S 1496167267:1496167267(0)
06:43:55 10.2.2.2.57536 > target.280: S 1496167267:1496167267(0)
06:43:55 10.3.3.3.57536 > target.280: S 1496167267:1496167267(0)
06:43:55 scanner.57536 > target.280: S 1496167267:1496167267(0)
06:43:55 10.4.4.4.57536 > target.280: S 1496167267:1496167267(0)
06:43:55 10.5.5.5.57536 > target.280: S 1496167267:1496167267(0)
...
```

Do you *really* know who is scanning you?

TCP SYN Flooding

Denial of Service Attack

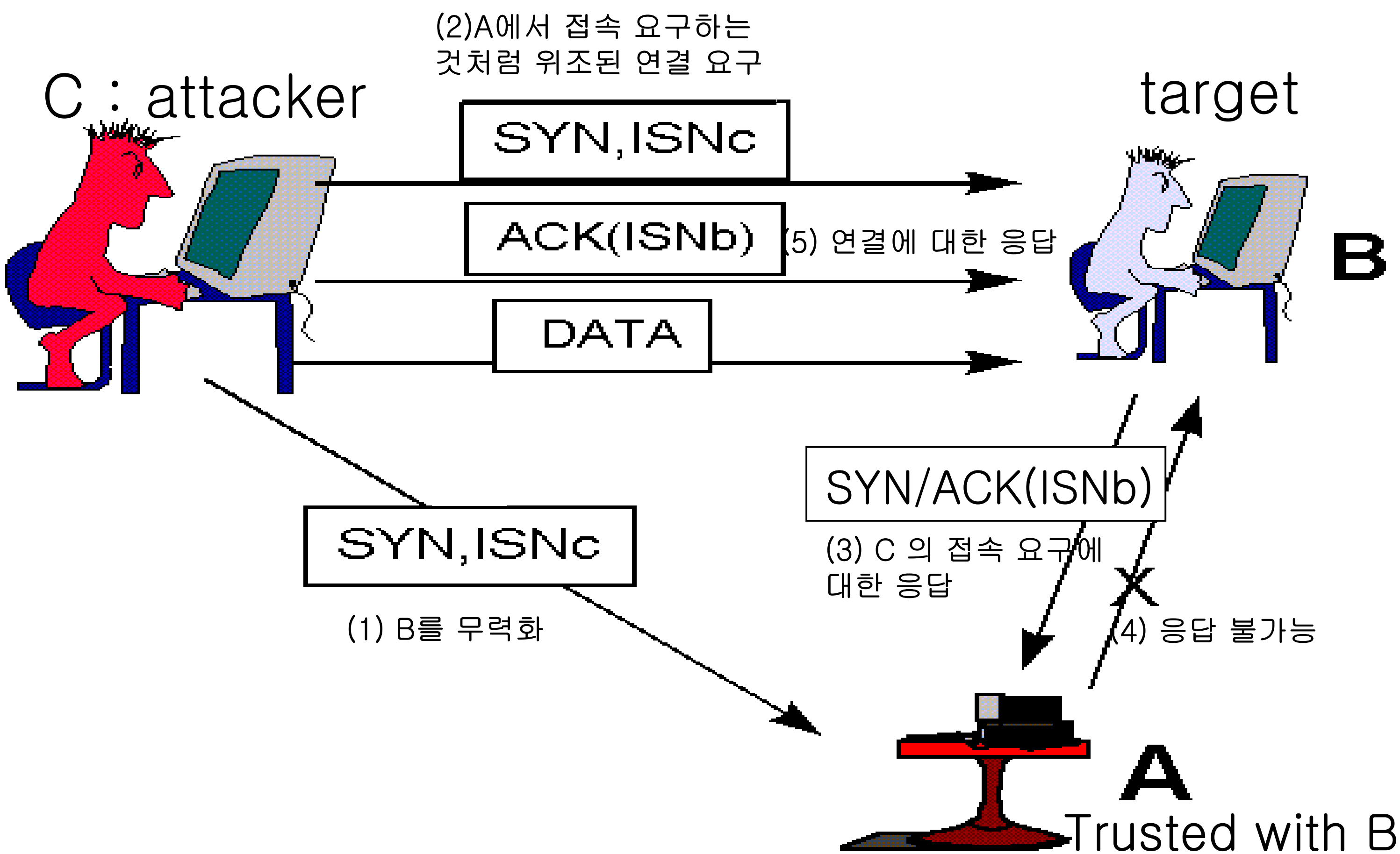


TCP SYN Flooding

Denial of Service

- In a three-way handshake the server responds to a client's initial SYN packet by sending a SYN-ACK.
- The server then waits for another ACK from the client before the connection becomes "established".
- Given this behavior consider what would happen if the initial SYN connection is spoofed so that it appears to have been generated by a nonexistent host? In that case the server will respond to the spoofed packet by sending a SYN-ACK to the nonexistent IP address and will then patiently wait for the final ACK which it will never receive.
- Defense
  - ✓ Firewall에서 SYN packet 지나간 후 일정시간 지나도 ACK 이루어지지 않으면 Server에 RST 전송
  - ✓ 라우터에서 외부로 나가는 패킷의 Source Address주소가 자신의 서브넷이 아니면 폐기

IP spoofing



(2)A에서 접속 요구하는
것처럼 위조된 연결 요구
C : attacker
target
SYN,ISNc
ACK(ISNb)
(5) 연결에 대한 응답
DATA
B
SYN/ACK(ISNb)
(3) C 의 접속 요구에
대한 응답
SYN,ISNc
(1) B를 무력화
(4) 응답 불가능
A
Trusted with B

스니핑 방지 대책

- 네트워크 설정을 통하여 스니핑을 어렵게 하는 많은 방법이 있으니 가장 좋은 방법은 데이터를 암호화하는 것이다.
- 특히, 웹호스팅, 인터넷데이터센터(IDS) 등과 같이 여러 업체가 같은 네트워크를 공유하는 환경에서는 스니핑으로부터 보안 대책이 마련 되어야 한다.
- 스니핑 방지를 위한 대책으로 먼저 네트워크를 스니핑하는 호스트를 주기적으로 점검하는 방법이 있다.
- 또한 스위칭 환경의 네트워크를 구성하여(비론 스니핑이 가능하기는 하지만) 되도록 스니핑이 어렵도록 구성해야 한다.

Dept. Computer Science Korea Univ.

스니핑 방지 대책

❑ 암호화

- ✓ SSL (Secure Socket Layer)
- ✓ PGP and S/MIME
- ✓ SSH (Secure Shell)
- ✓ VPN (Virtual Private Networks)

Dept. Computer Science Korea Univ.

스니핑 방지 대책

- 스위치 환경의 네트워크 구성
  - 스위치는 트래픽을 전달할 때 모든 세그먼트로 브로드캐스트 하지 않고 해당 세그먼트에만 전달하므로 일반 허브를 사용하는 것보다 안전하다.
  - 그러나 스위치 환경에서의 스니핑 기법과 같은 공격을 할 수 있다.
  - 같은 세그먼트 내에서의 스니핑을 막을 수 없다.
  - 스위치의 각 포트에 대하여 MAC 주소를 static(permanent)하게 대응시키면 ARP spoofing, ARP redirect 등의 공격을 막을수 있다.

Dept. Computer Science Korea Univ.

스니핑 방지 대책

❑ 스니퍼 탐지

- ✓ Ping을 이용하는 방법
  - 의심이 가는 시스템에게 ping을 보내는데 MAC 주소를 위장하여 보내는 방법이다.
  - 만약 ping reply(ICMP Echo Reply)를 받게되면, 해당 호스트가 스니핑을 하고 있는 것이다. 왜냐하면 존재하지 않는 MAC 주소를 사용했기 때문에 스니핑을 하지 않는 호스트는 누구도 ping request를 볼 수 없게되며, reply를 하지 않게 된다.
- ✓ ARP를 이용하는 방법
  - Ping 방법과 유사한 방법으로 non-broadcast로 위조된 ARP request를 보냈을 때 ARP reponse가 오면 상대방 호스트가 ‘promiscuous mode’로 설정되어 있는 것이다

Dept. Computer Science Korea Univ.

스니핑 방지 대책

- ✓ DNS 방법
  - 일반적으로 스니핑 프로그램은 사용자의 편의를 위하여 스니핑한 시스템의 IP 주소를 보여주지 않고 도메인 네임을 보여주기 위하여 Inverse-DNS lookup을 수행하기 땜에 DNS 트래픽을 감시하여 스니퍼를 탐지할 수 있다.
- ✓ 유인(decoy)방법
  - IDS를 사용하여 미리 설정된 계정이나 패스워드를 탐지 시키도록 유도하여 스니핑을 탐지할 수 있다.
- ✓ Host method
  - 호스트 단위에서 'promiscuous mode'를 확인하는 방법으로 'ifconfig –a' 명령을 이용하여 확인할 수 있다.