Link layer security

CS642: Computer Security

Getting started on network security

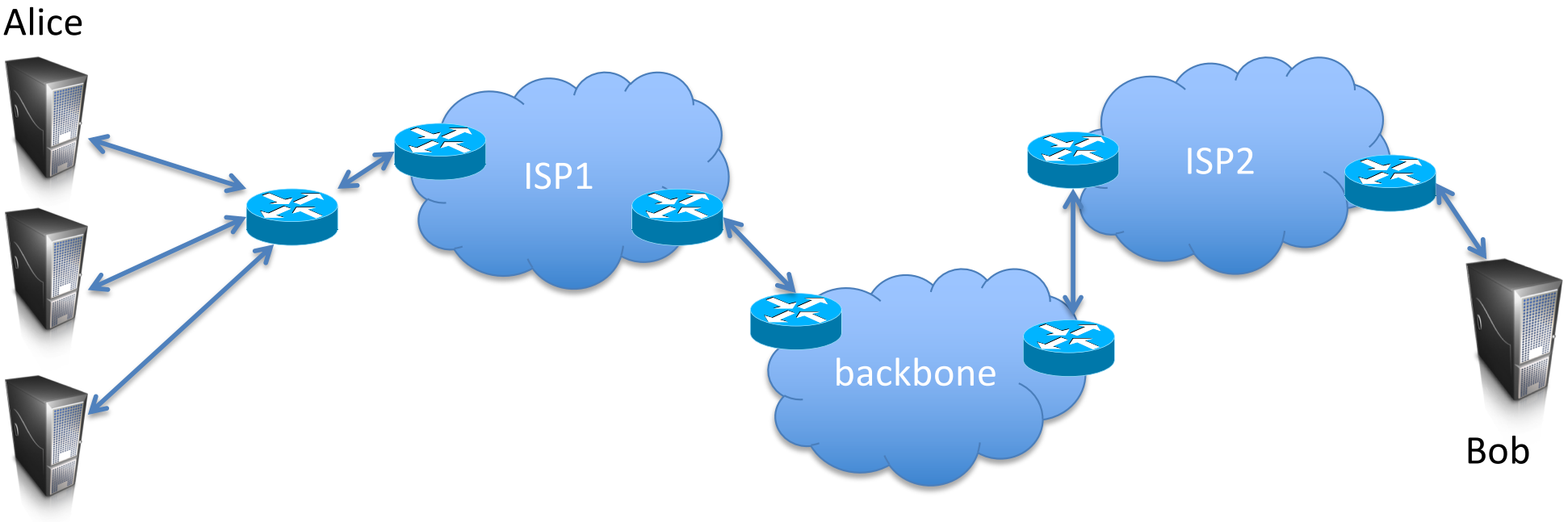
Internet protocol stack

Man-in-the-middle

Address resolution protocol and ARP spoofing

802.11

Internet



Local area network
(LAN)

Ethernet

802.11

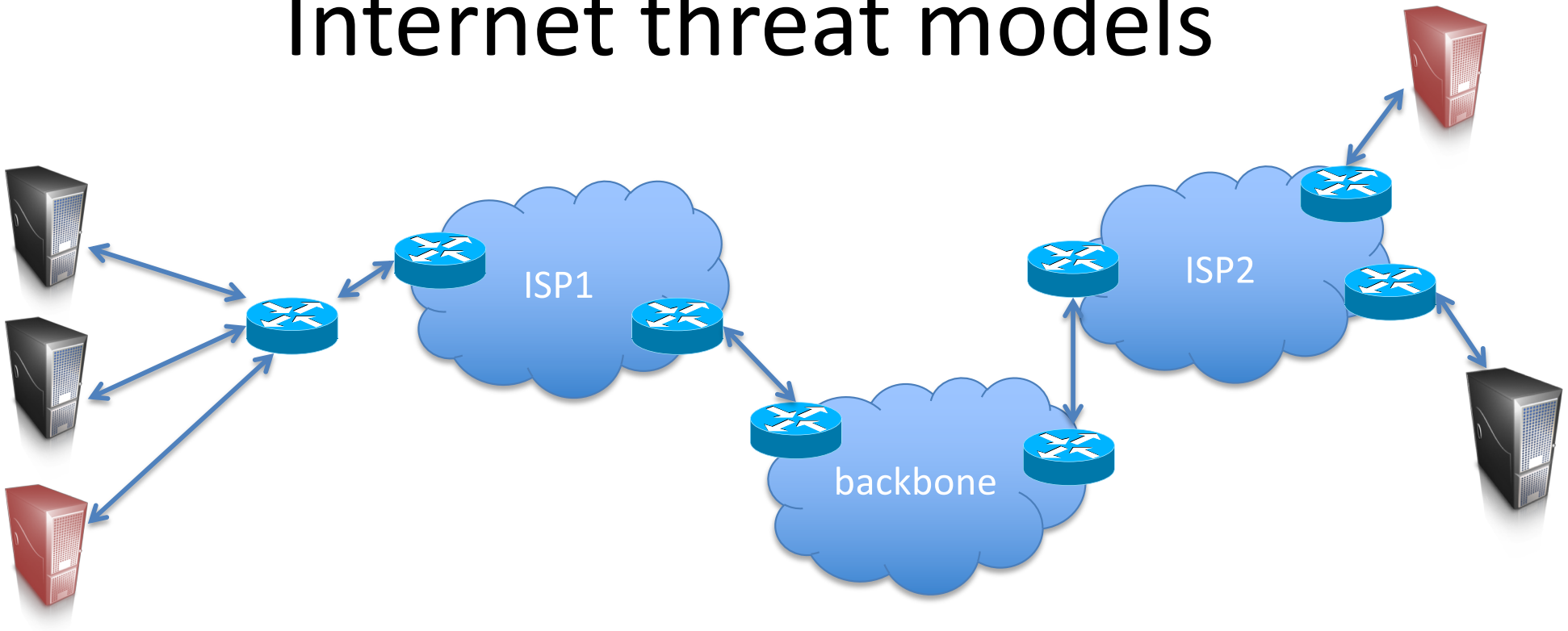
Internet

TCP/IP

BGP (border gateway protocol)

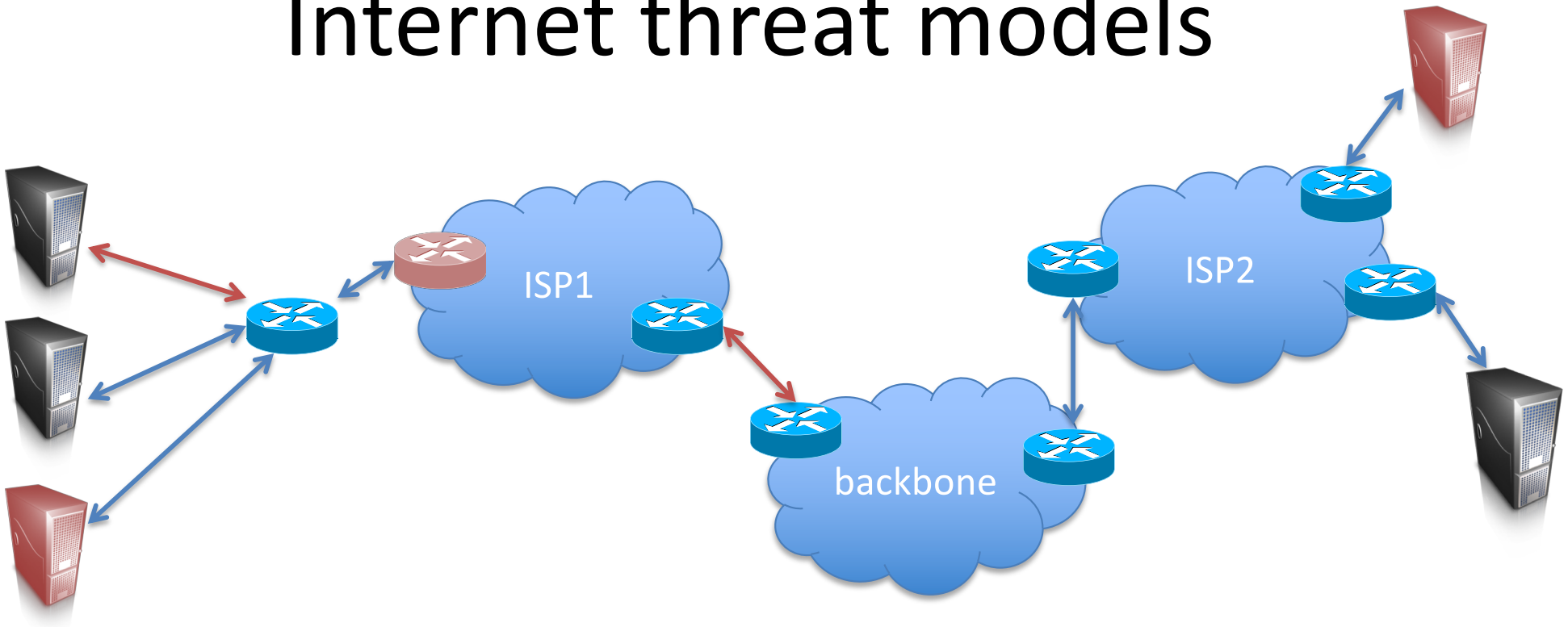
DNS (domain name system)

Internet threat models



(1) Malicious hosts

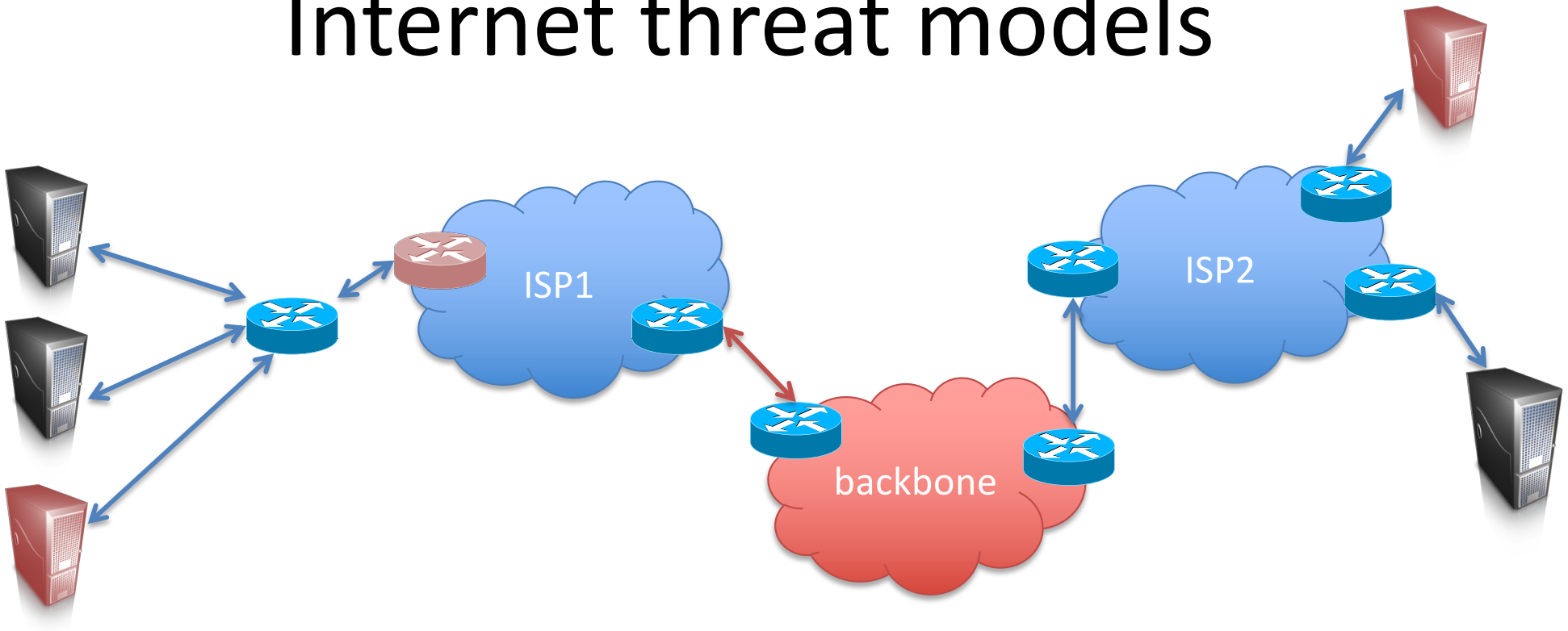
Internet threat models



(1) Malicious hosts

(2) Subverted routers or links

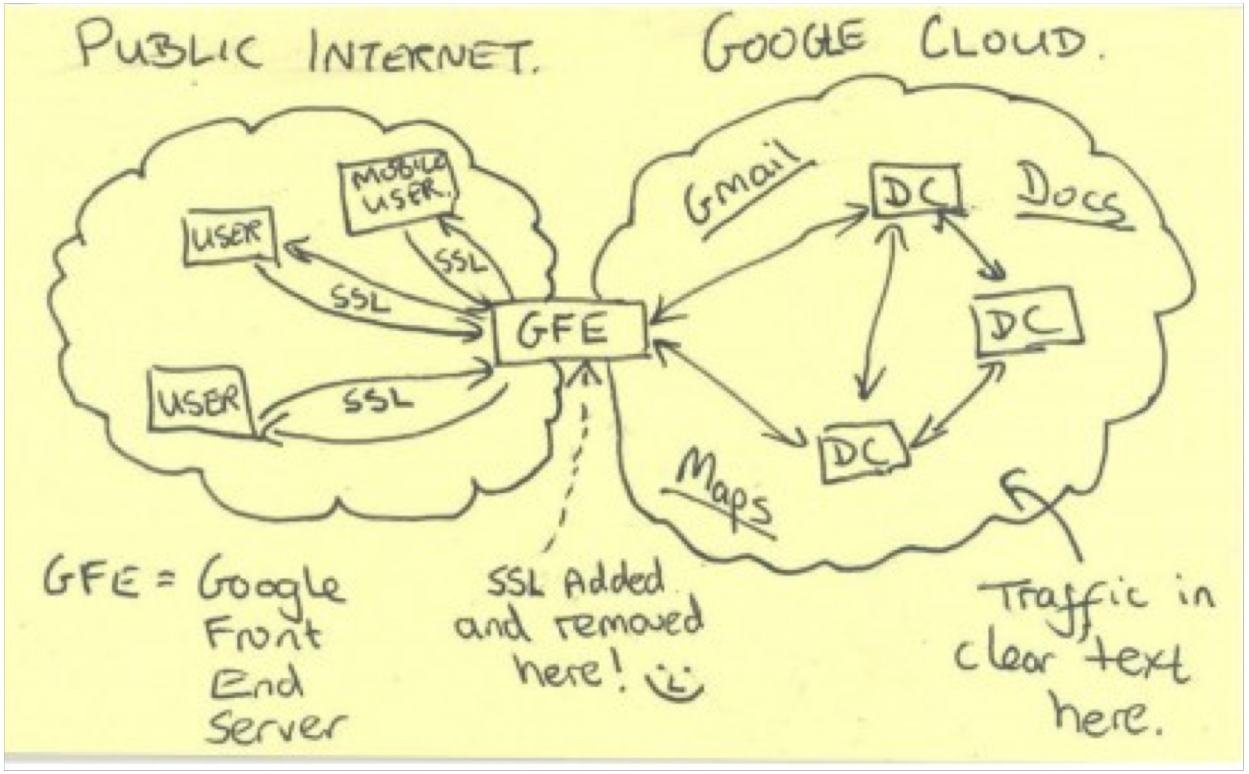
Internet threat models



(1) Malicious hosts

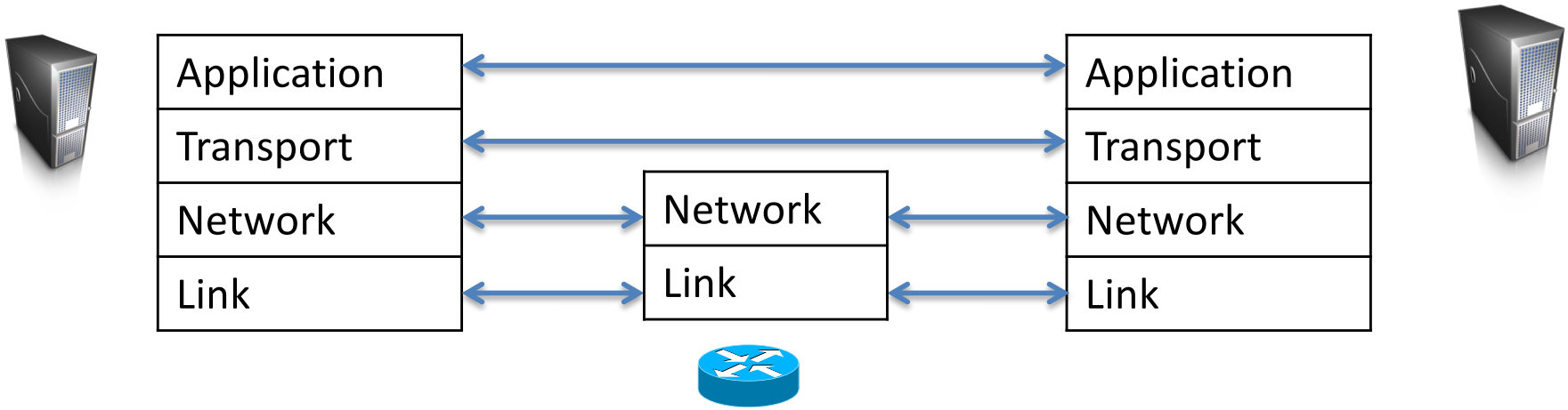
(2) Subverted routers or links

(3) Malicious ISPs or backbone

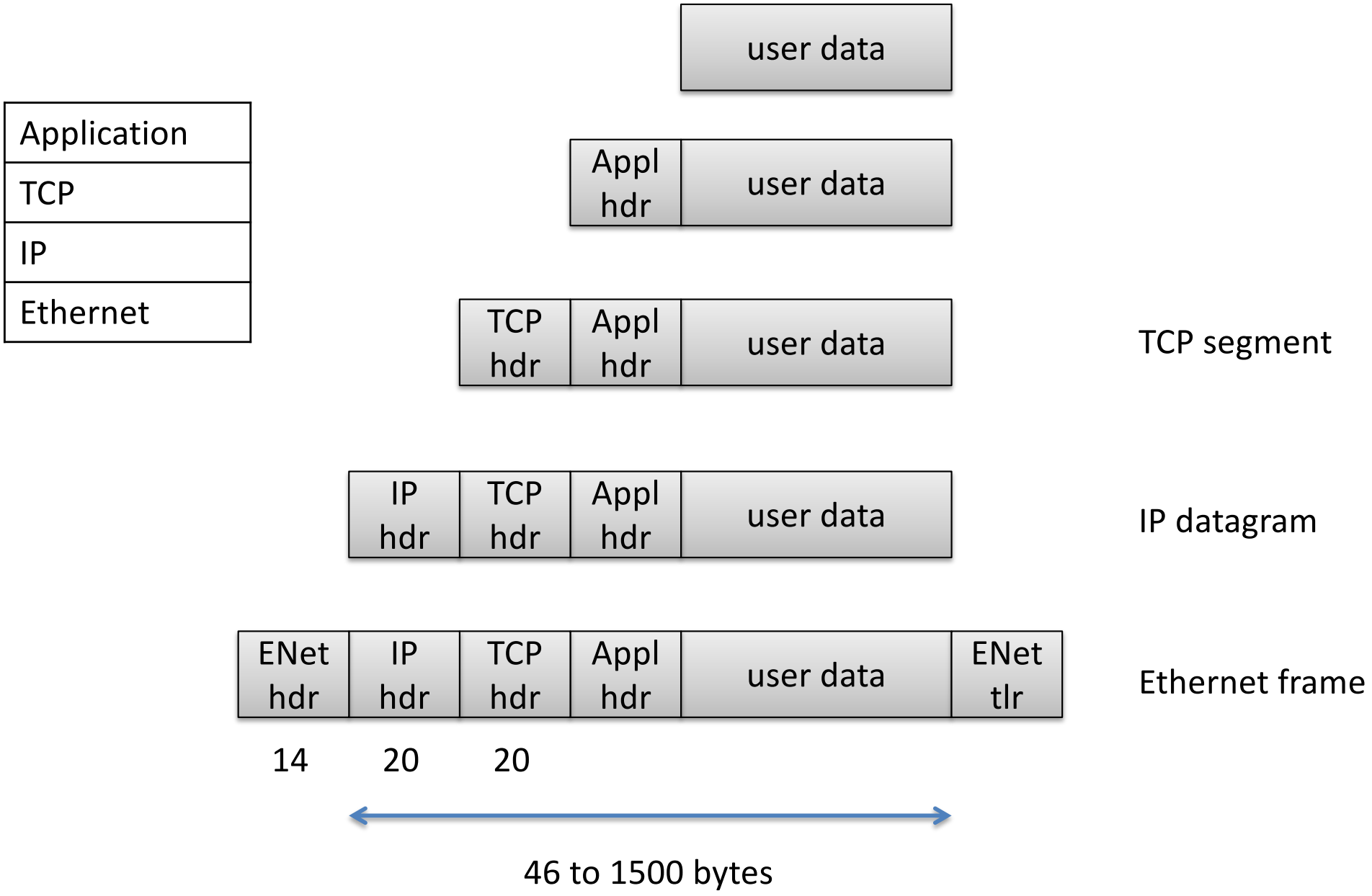

PUBLIC INTERNET.
GOOGLE CLOUD.
MOBILE USER.
USER
USER
SSL
SSL
SSL
GFE
Gmail
DC
Docs
DC
DC
Maps
GFE = Google Front End Server
SSL Added and removed here!
Traffic in clear text here.

Internet protocol stack

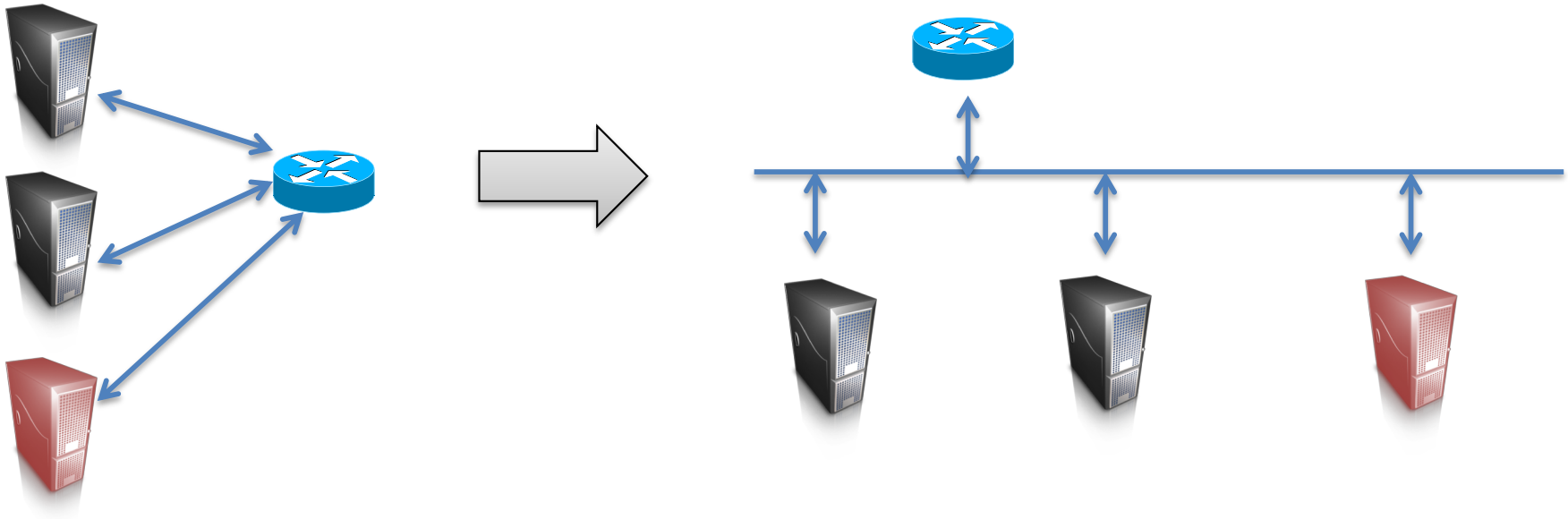
Application	HTTP, FTP, SMTP, SSH, etc.
Transport	TCP, UDP
Network	IP, ICMP, IGMP
Link	802x (802.11, Ethernet)



Internet protocol stack



Ethernet



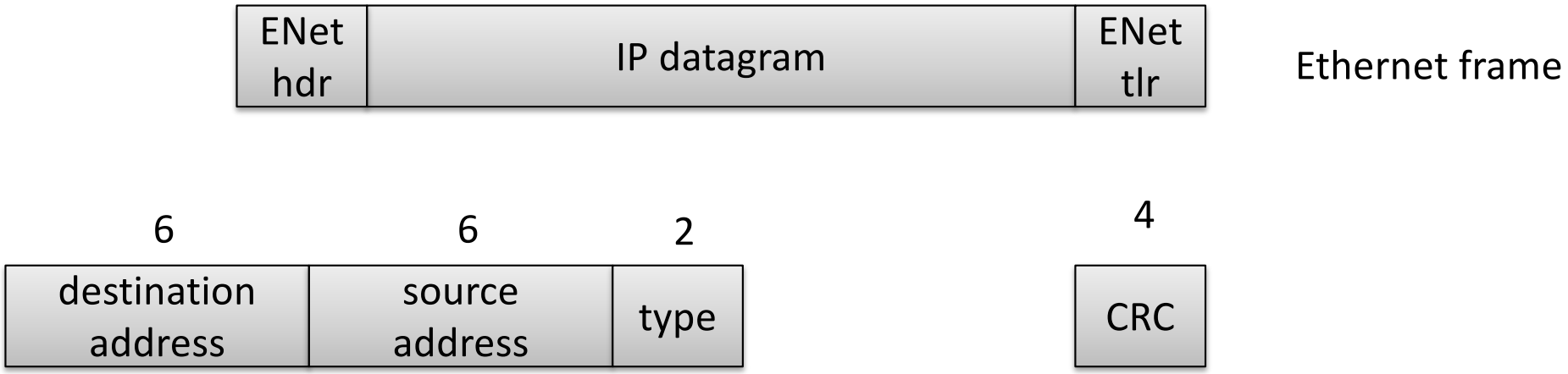
Carrier Sense, Multiple Access with Collision Detection (CSMA/CD)

Take turns using broadcast channel (the wire)

Detect collisions, jam, and random backoff

Security issues?

Ethernet



Media access control (MAC) addresses 48 bits

Type = what is data payload (0x0800 = IPv4, 0x0806 = ARP, 0x86DD = IPv6)

32 bit Cyclic Redundancy Check (CRC) checksum

802.2 LLC frame format slightly different, but similar ideas

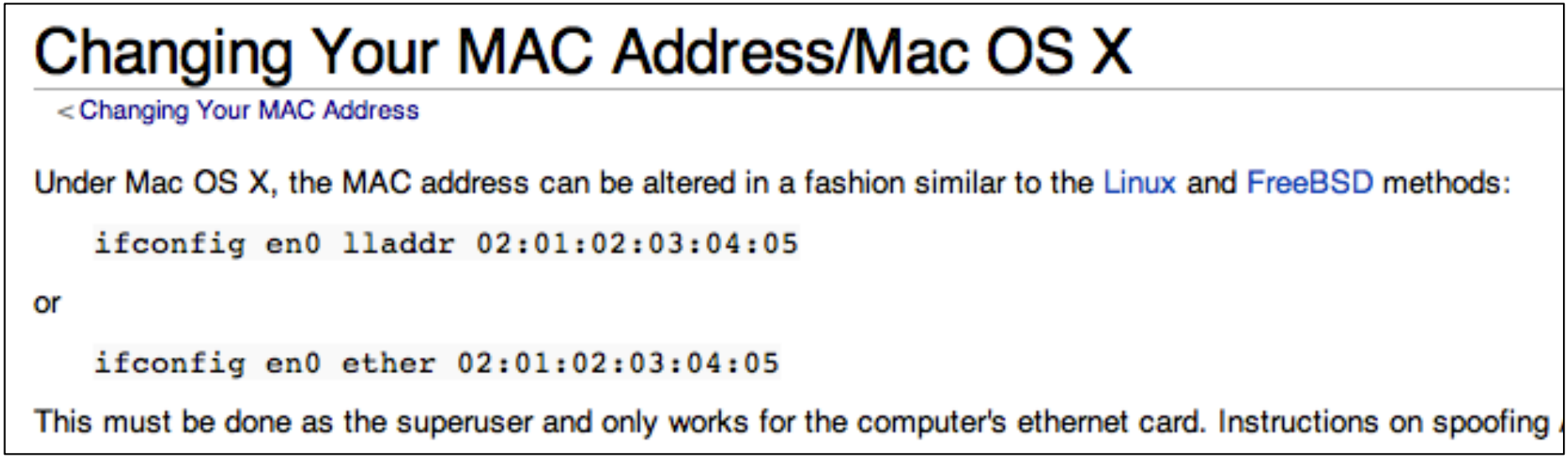
MAC addresses

- Two types: universally or locally administered

  - 2 LSBs of first byte are control bits:
    - 1st LSB: multicast/unicast
    - 2nd LSB: universal/local flag
- Hardware (ethernet card/WiFi card) initialized with MAC address
- But:
  - Most ethernet cards allow one to change address

MAC spoofing

- Many LANs, WiFis use MAC-based access controls – e.g. CS LAN



Changing Your MAC Address/Mac OS X

< Changing Your MAC Address

Under Mac OS X, the MAC address can be altered in a fashion similar to the Linux and FreeBSD methods:

```
ifconfig en0 lladdr 02:01:02:03:04:05
```

or

```
ifconfig en0 ether 02:01:02:03:04:05
```

This must be done as the superuser and only works for the computer's ethernet card. Instructions on spoofing

Courtesy of wikibooks
http://en.wikibooks.org/wiki/Changing_Your_MAC_Address/Mac_OS_X

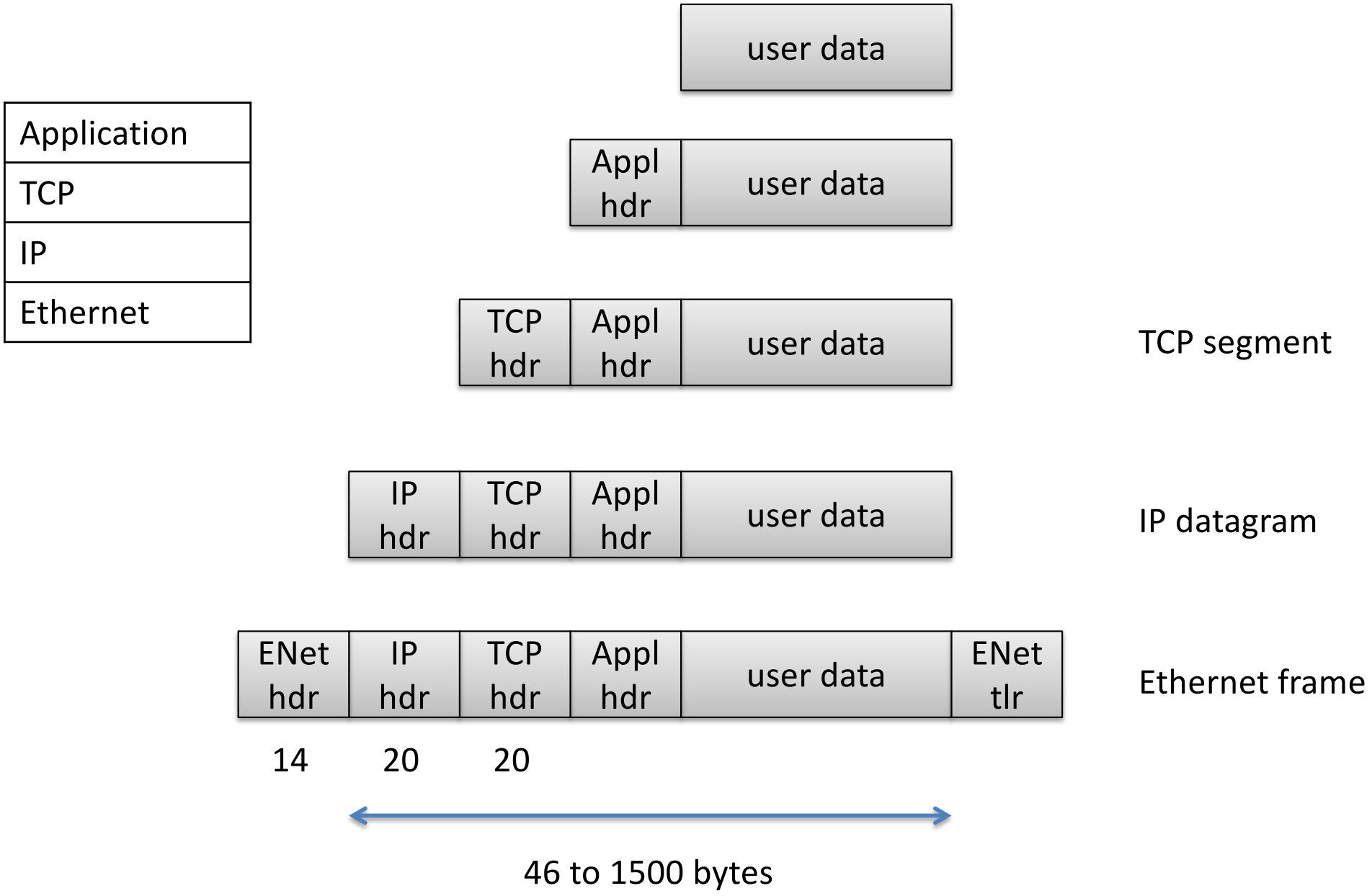
MAC spoofing

Aaron Swartz, a fellow at Harvard University's Center for Ethics and an open source programmer involved with creating the RSS 1.0 specification and more generally in the open culture movement, has been arrested and charged with wire fraud, computer fraud, unlawfully obtaining information from a protected computer, and recklessly damaging a protected computer after he entered a computer lab at MIT in Cambridge, Massachusetts and downloaded two-thirds of the material on JSTOR, an academic journal repository.

http://en.wikinews.org/wiki/Aaron_Swartz_arrested_and_charged_for_downloading_JSTOR_articles

Supposedly used MAC spoofing to get onto MIT network

Internet protocol stack

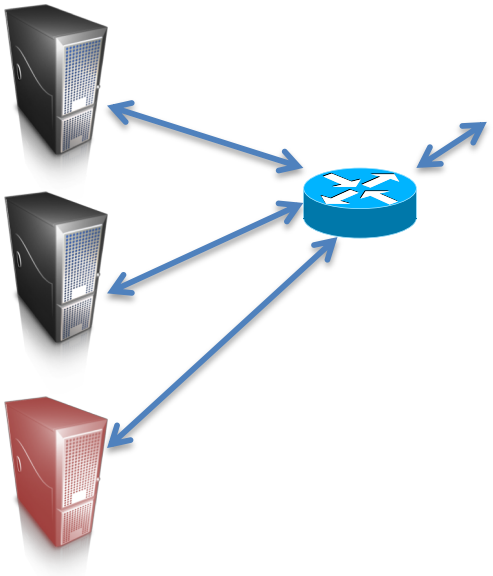


IPv4

Ethernet frame containing IP datagram

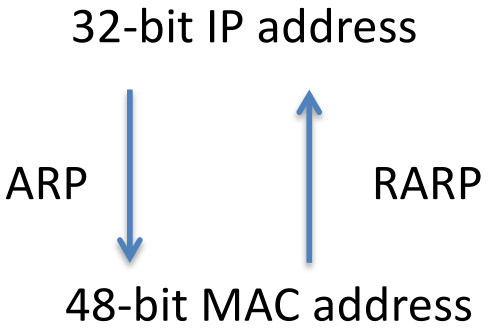
4-bit version	4-bit hdr len	8-bit type of service	16-bit total length (in bytes)	
16-bit identification			3-bit flags	13-bit fragmentation offset
8-bit time to live (TTL)		8-bit protocol	16-bit header checksum	
32-bit source IP address				
32-bit destination IP address				
options (optional)				

Address resolution protocol



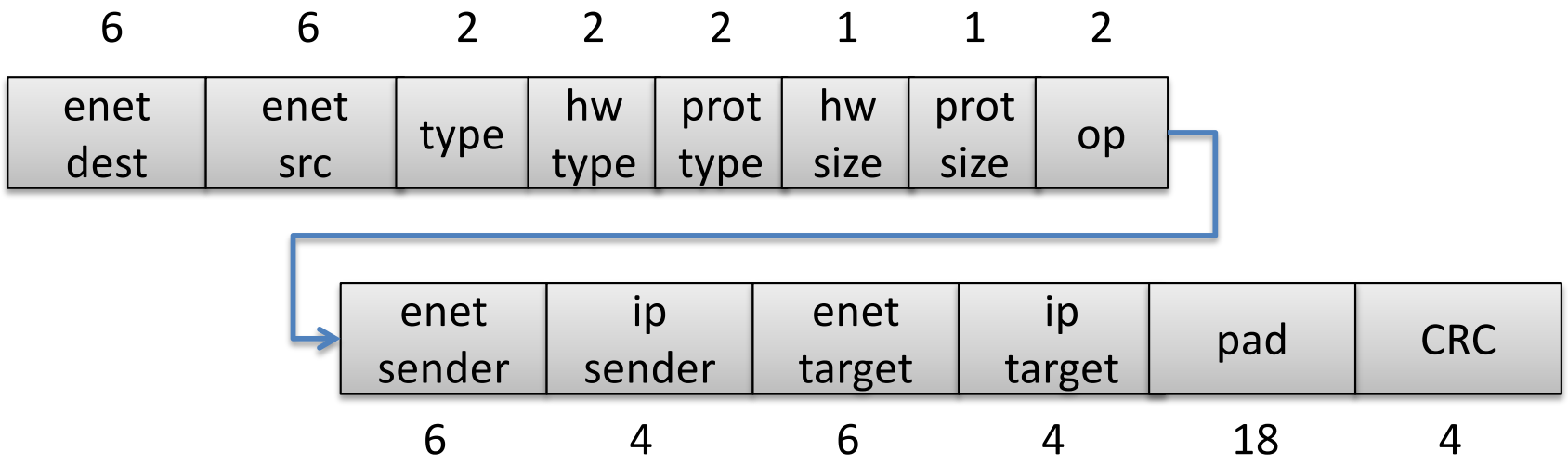
IP routing:
Figure out where to send
an IP packet based on destination address.

Link layer and IP must cooperate to get things sent



ARP/RARP enables this cooperation by mapping IPs to MACs

Address resolution protocol



frame type = 0x0806 (ARP) or 0x8035 (RARP)

enet dest is all 1's, 0xFFFFFFFFFFFF for broadcast

hw type, prot(ocol) type specify what types of addresses we're looking up

op specifies whether this is an ARP request, ARP reply, RARP request, RARP reply

ARP caches

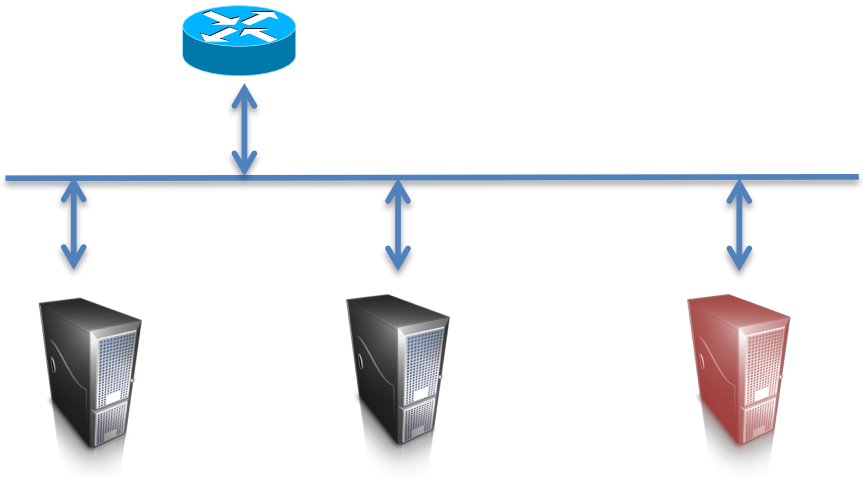
- Hosts maintain cache of ARP data
  - just a table mapping between IPs and MACs

```
usage: arp [-n] [-i interface] hostname
       arp [-n] [-i interface] [-l] -a
       arp -d hostname [pub] [ifscope interface]
       arp -d [-i interface] -a
       arp -s hostname ether_addr [temp] [reject] [blackhole] [pub [only]] [ifsc
ope interface]
       arp -S hostname ether_addr [temp] [reject] [blackhole] [pub [only]] [ifsc
ope interface]
       arp -f filename
[swift:642/background] arp -a
? (192.168.0.1) at f4:f2:6d:2d:57:c6 on en0 ifscope [ethernet]
? (192.168.0.108) at c8:3a:6b:ab:29:2c on en0 ifscope [ethernet]
? (192.168.0.114) at 0:e:58:8d:24:38 on en0 ifscope [ethernet]
? (192.168.0.117) at 74:75:48:5c:a:b1 on en0 ifscope [ethernet]
? (192.168.0.123) at 60:6d:c7:68:80:23 on en0 ifscope [ethernet]
```

ARP has no authentication

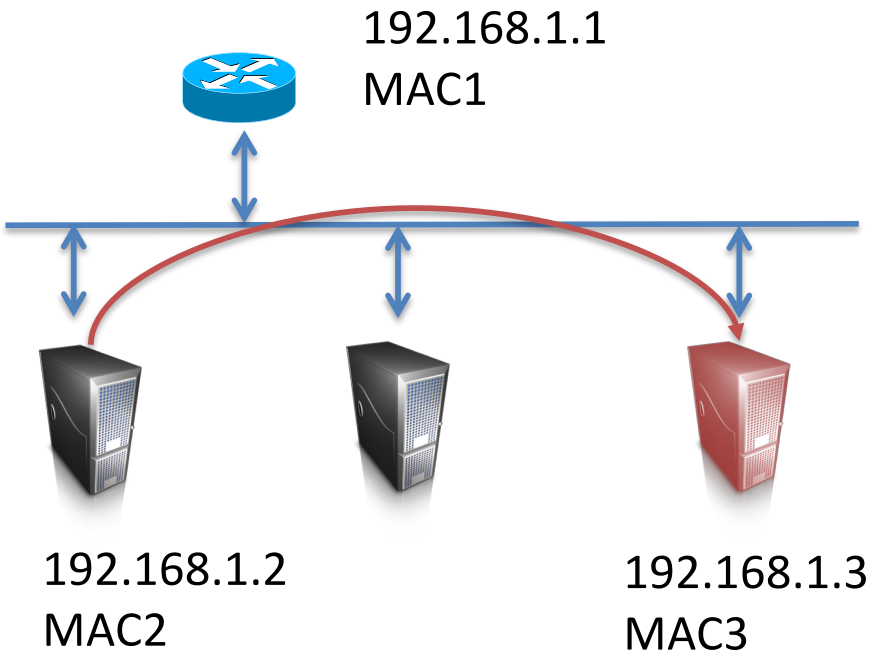
- Easy to sniff packets on (non-switched) ethernet
- What else can we do?

<u>Easy Denial of Service (DoS):</u>
Send ARP reply associating **gateway** 192.168.1.1 with a non-used MAC address



ARP has no authentication

- Easy to sniff packets on (non-switched) ethernet
- What else can we do?



Active Man-in-the-Middle:

ARP reply to MAC2
192.168.1.1 -> MAC3

ARP reply to MAC1
192.168.1.2 -> MAC3

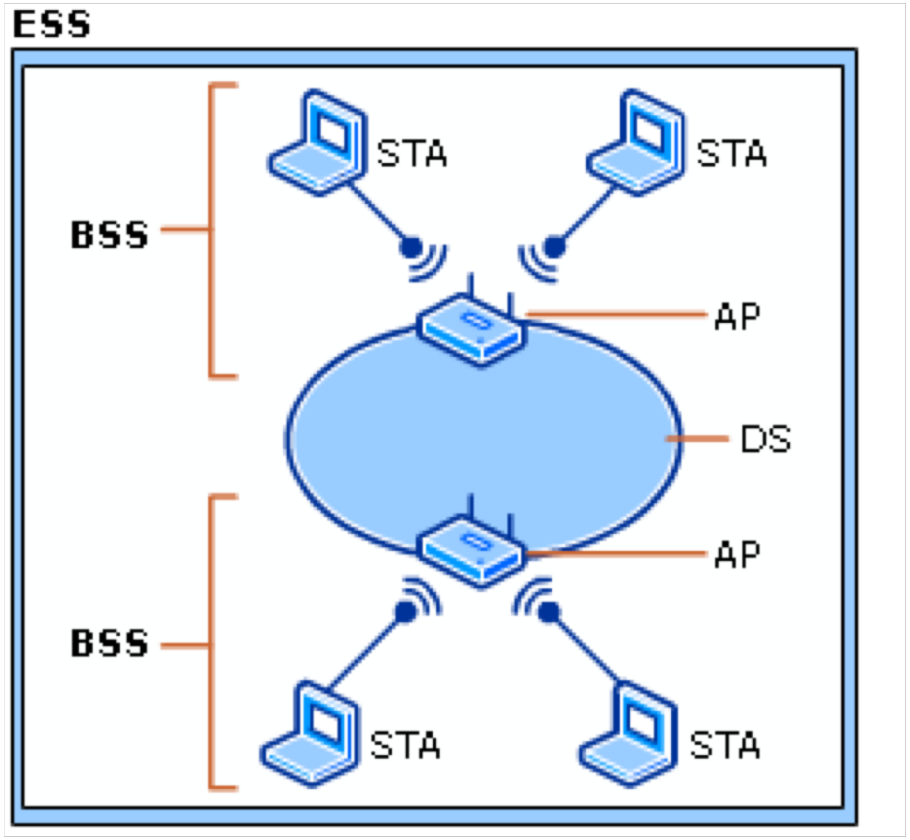
Now traffic “routed” through malicious box

802.11 (wifi)

STA = station
AP = access point

BSS = basic service set
DS = distribution service
ESS = extended service set

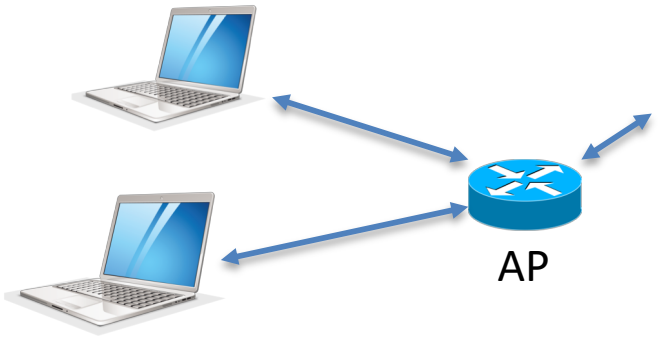
SSID (service set identifier) identifies the 802.11 network



http://technet.microsoft.com/en-us/library/cc757419(WS.10).aspx

Typical WiFi modes:
Unsecured
Wireless Protected Access (WPA2) - password authenticated, encrypted

802.11 association



802.11 association

AP

Probe request

SSID: “linksys”, BSSID: MAC1

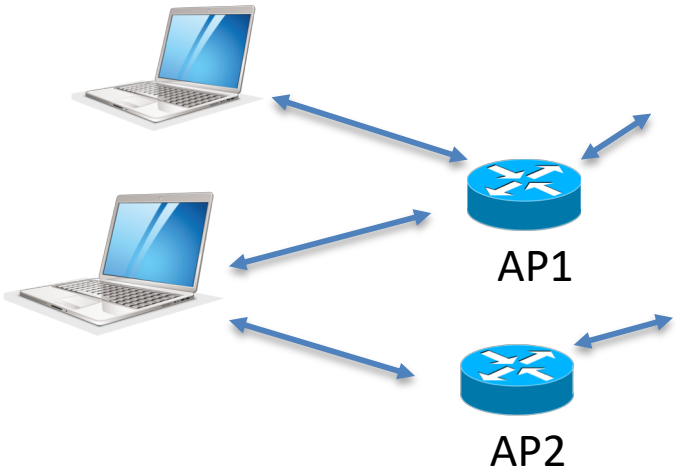
Auth request MAC1

Auth response

Associate request MAC1

Associate response

802.11 association



Two APs for same network


MAC1


MAC2

Probe request

SSID: “linksys”, BSSID: MAC1

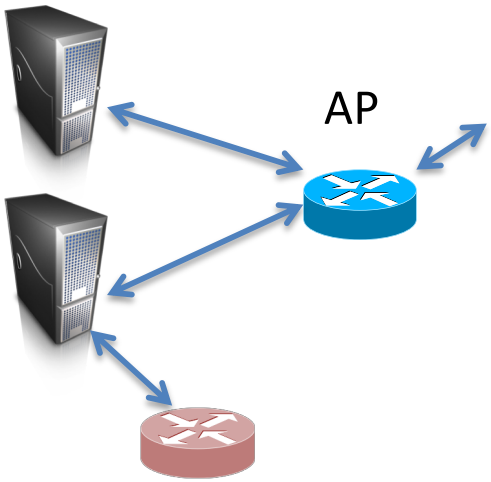
SSID: “linksys”, BSSID: MAC2

Choose one
of MAC1, MAC2

Auth request MAC2

...

802.11 evil twins



Basic idea:

- Attacker pretends to be an AP to intercept traffic or collect data

802.11 association

Probe request

SSID: “linksys”, BSSID: MAC1

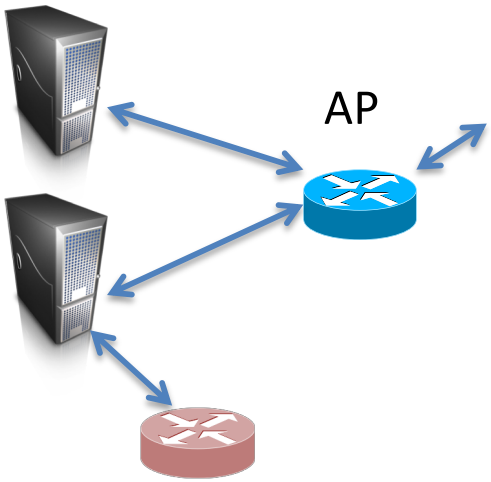
Auth request MAC1

Auth response

Associate request MAC1

Associate response

802.11 evil twins



Basic idea:

- Attacker pretends to be an AP to intercept traffic or collect data

Two APs for same network

MAC1

Probe request

MAC2

SSID: “linksys”, BSSID: MAC1

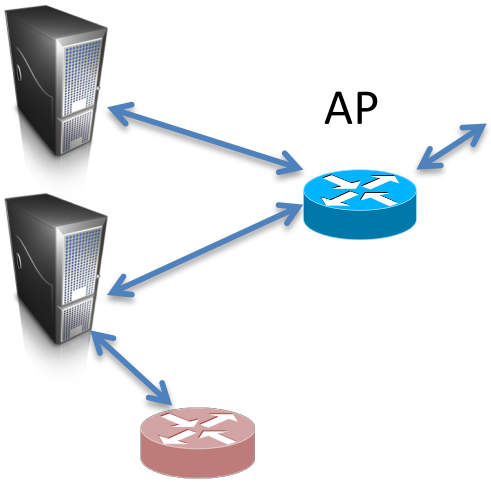
SSID: “linksys”, BSSID: MAC2

Choose one of MAC1, MAC2

Auth request MAC2

...

802.11 evil twins



Basic idea:

- Attacker pretends to be an AP to intercept traffic or collect data

Basic attack: rogue AP

MAC1

Probe request

MAC2

SSID: “linksys”, BSSID: MAC1

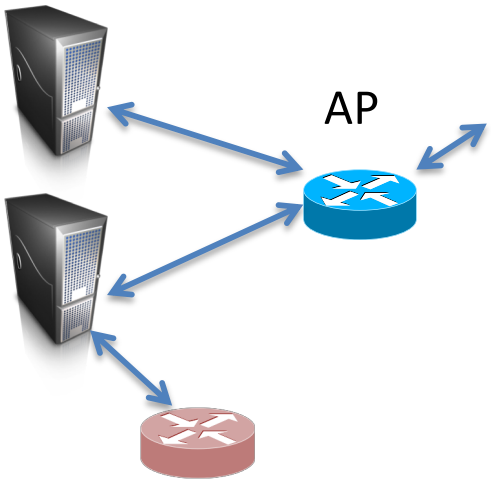
SSID: “linksys”, BSSID: MAC2

Choose one of MAC1, MAC2

Auth request MAC2

...

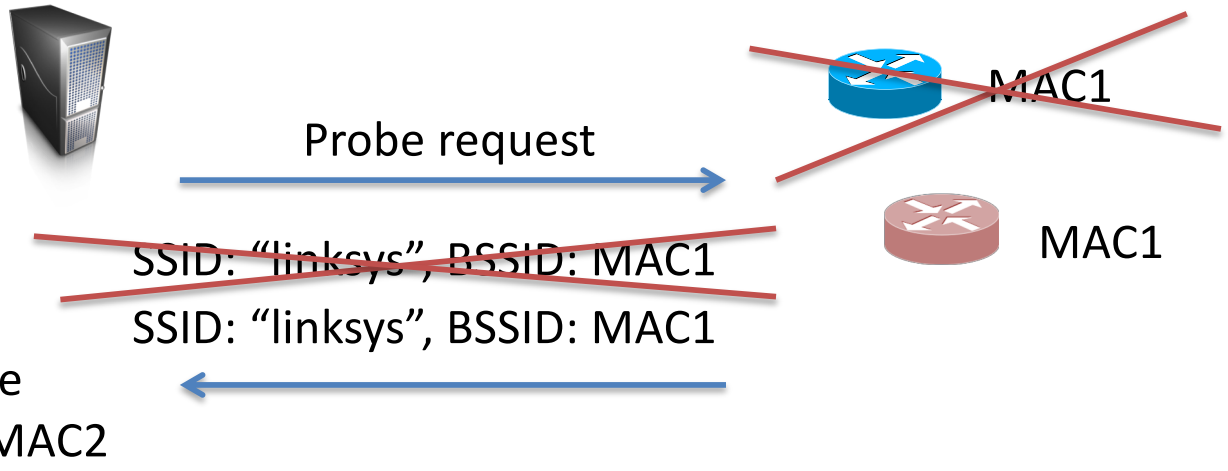
802.11 evil twins



Basic idea:

- Attacker pretends to be an AP to intercept traffic or collect data

Evil twin: spoof MAC1



Attacker can send forged disassociate message to victim to get it to look for new connection

Victim might send out probe requests for particular SSIDs, giving attacker info

...

Conceptually similar to ARP poisoning

Parrot ARdrone

Drone is a WiFi access point
Uses unsecured 802.11 connection (WiFi)
Controlled from iPad or iPhone with an app
Uses MAC address for security

IP protocol (IPv4)

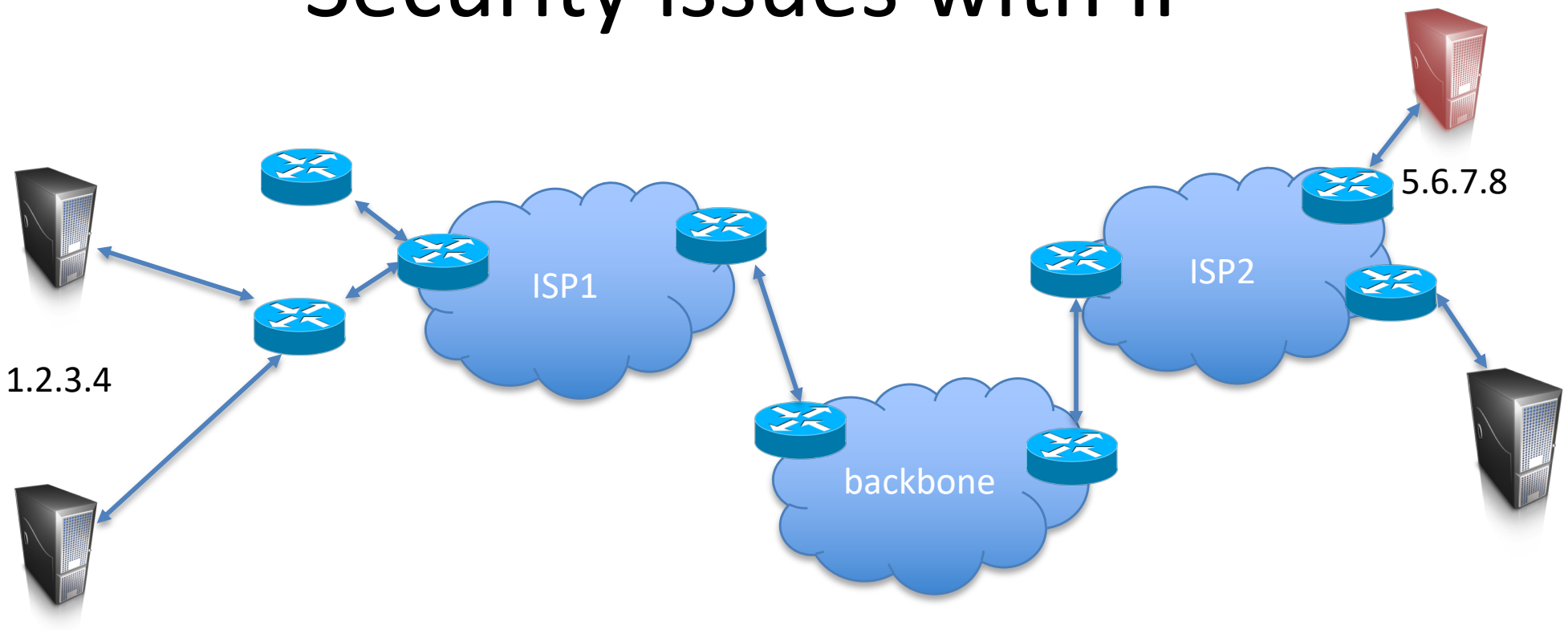
- Connectionless
  - no state
- Unreliable
  - no guarantees
- ICMP (Internet Control Message Protocol)
  - error messages, etc.
  - often used by tools such as ping, traceroute

IPv4

Ethernet frame containing IP datagram

4-bit version	4-bit hdr len	8-bit type of service	16-bit total length (in bytes)	
16-bit identification			3-bit flags	13-bit fragmentation 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 (optional)				

Security issues with IP

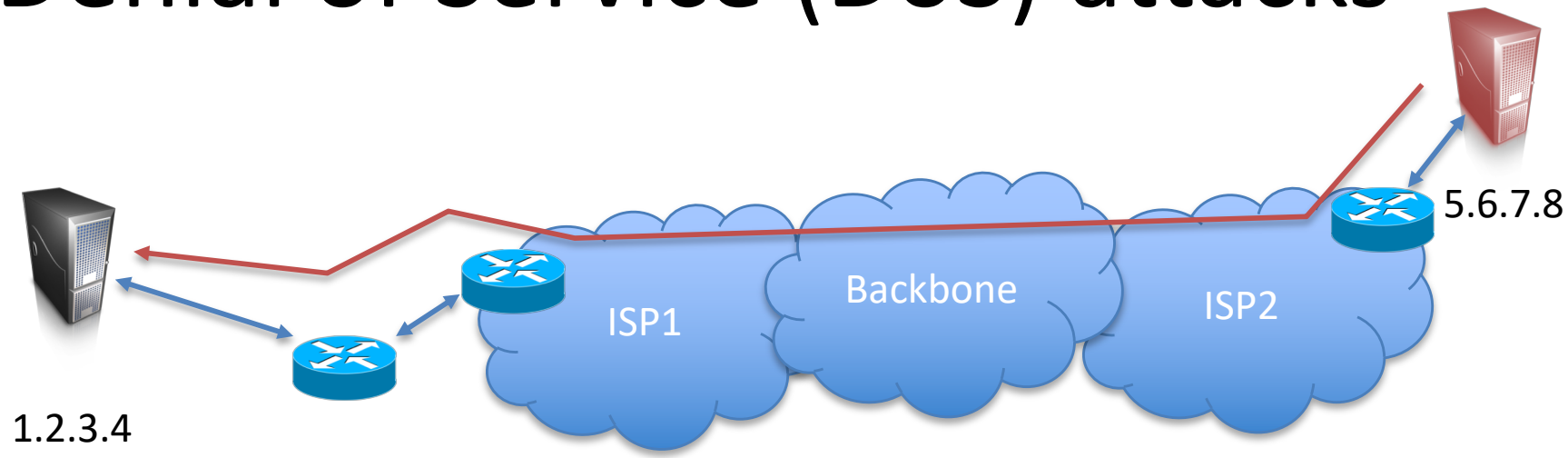


Routing has issues, we'll get to that later

What else?

- No source address authentication in general

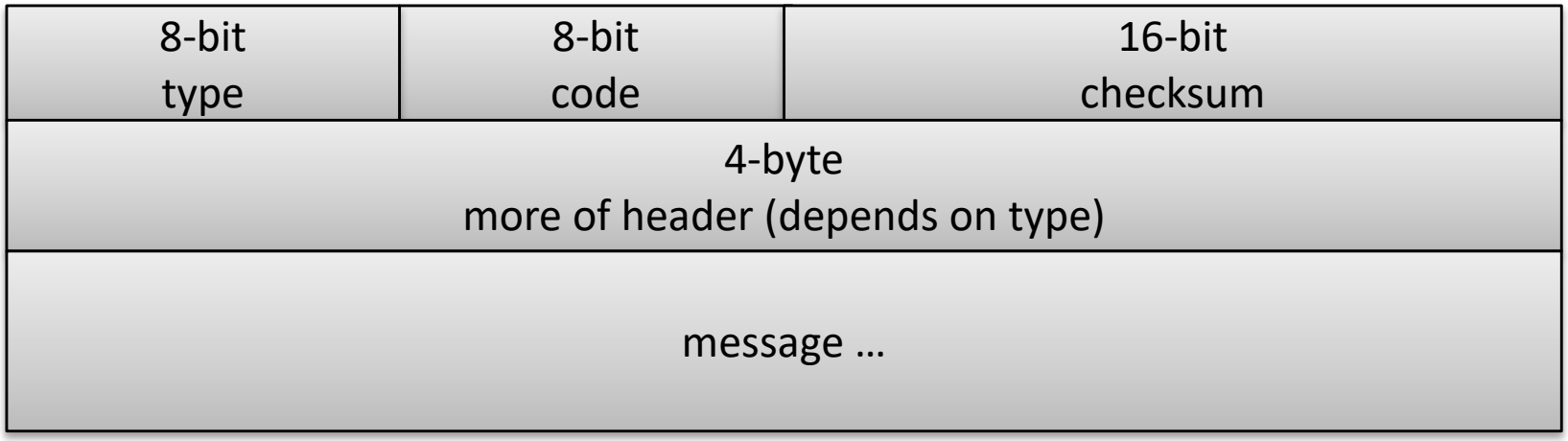
Denial of Service (DoS) attacks



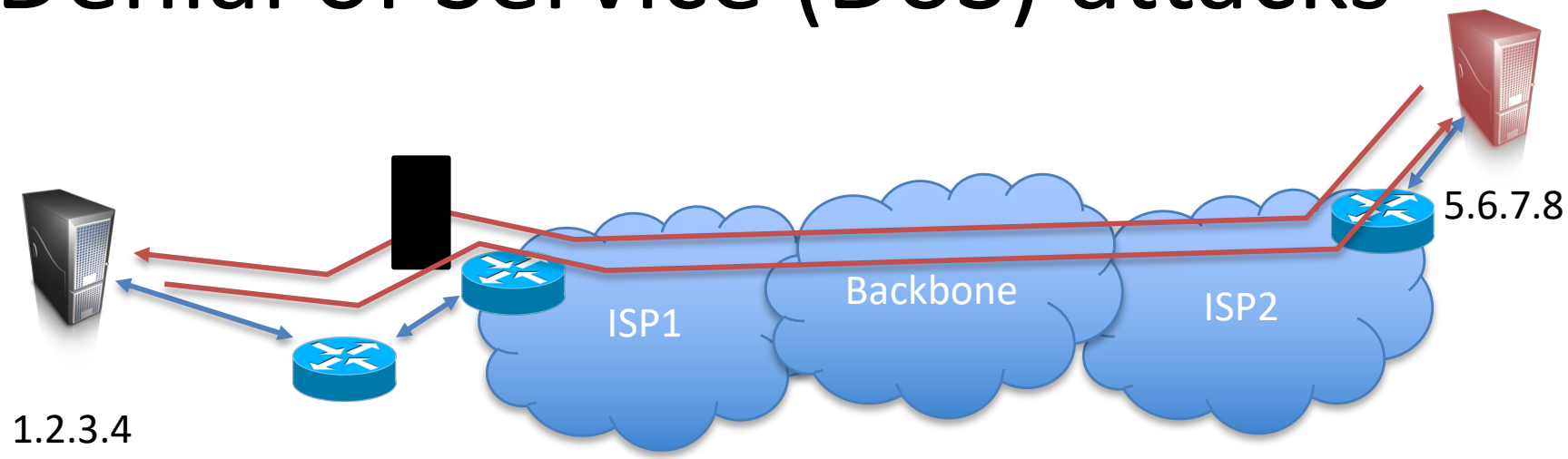
Goal: prevent legitimate users from accessing victim (1.2.3.4)

ICMP ping flood

ICMP
(Internet Control Message Protocol)



Denial of Service (DoS) attacks

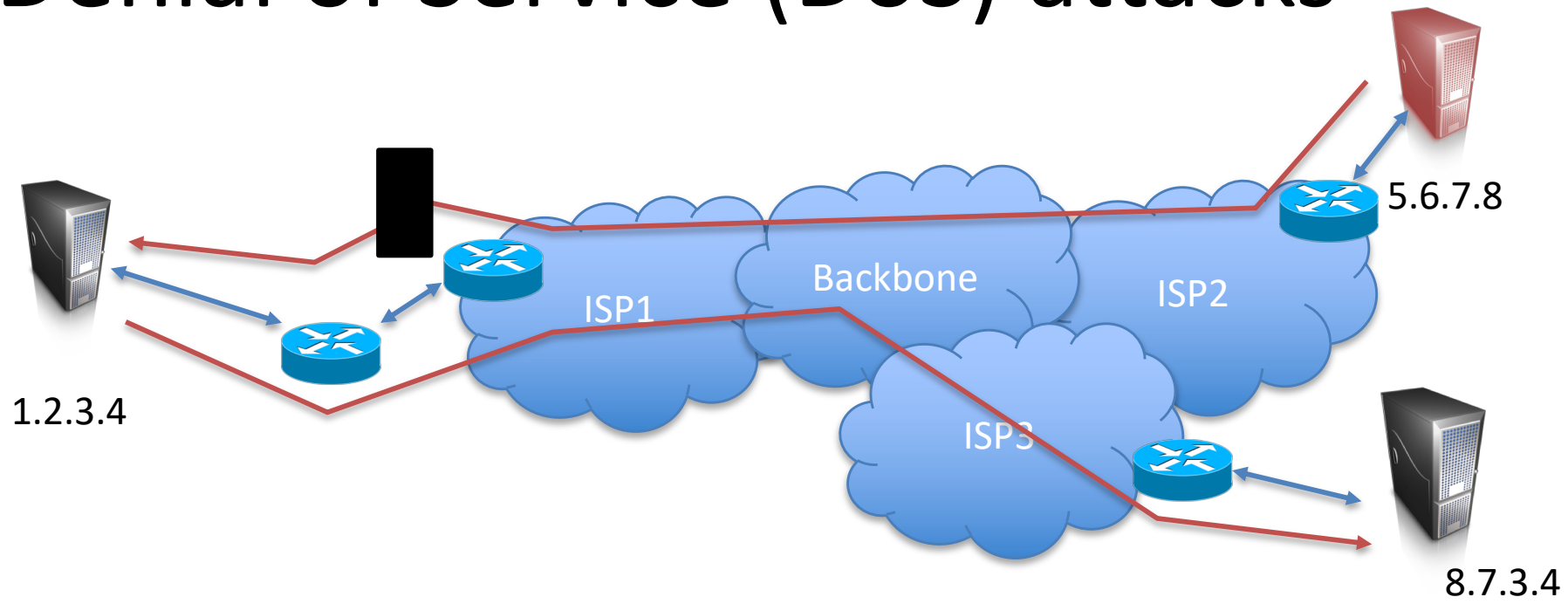


Goal is to prevent legitimate users from accessing victim (1.2.3.4)

ICMP ping flood

- Attacker sends ICMP pings as fast as possible to victim
- When will this work as a DoS? Attacker resources > victim's
- How can this be prevented? Ingress filtering near victim

Denial of Service (DoS) attacks



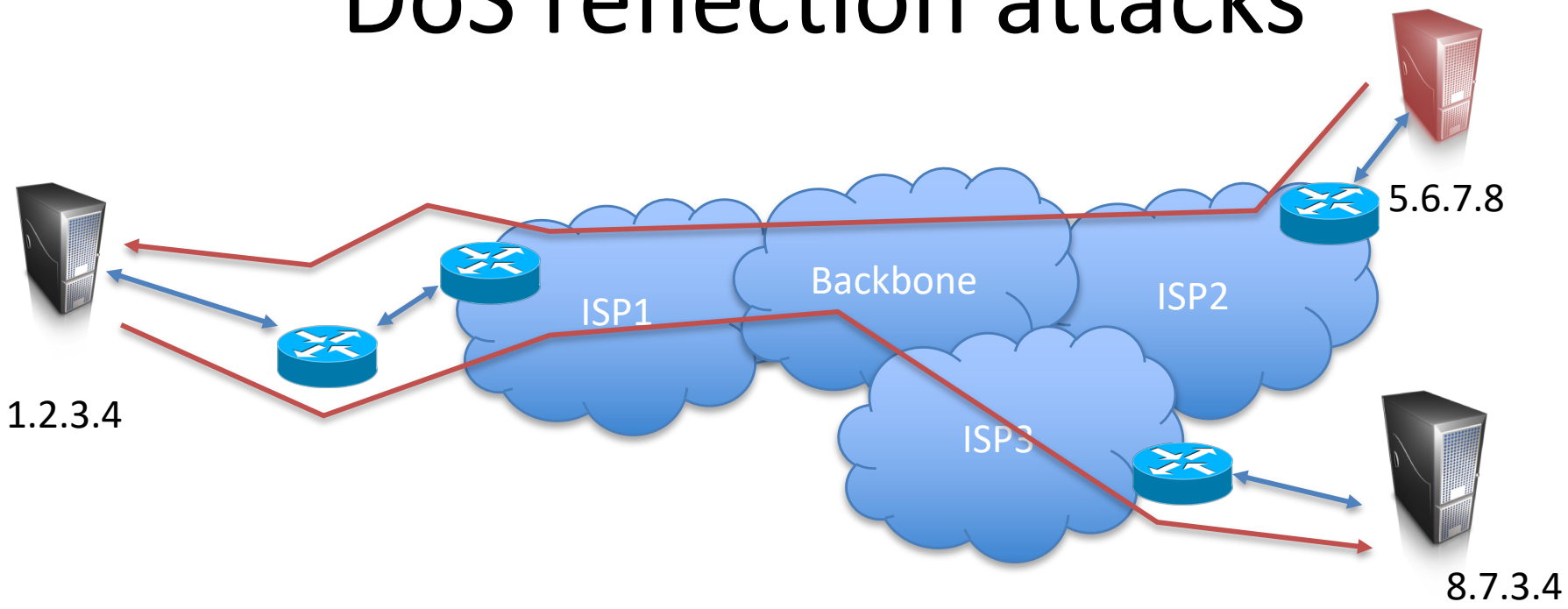
How can attacker avoid ingress filtering?

Attacker can send packet with fake source IP "spoofed" packet
Packet will get routed correctly
Replies will not

Send IP packet with source: 8.7.3.4 dest: 1.2.3.4 from 5.6.7.8

Filter based on source may be incorrect

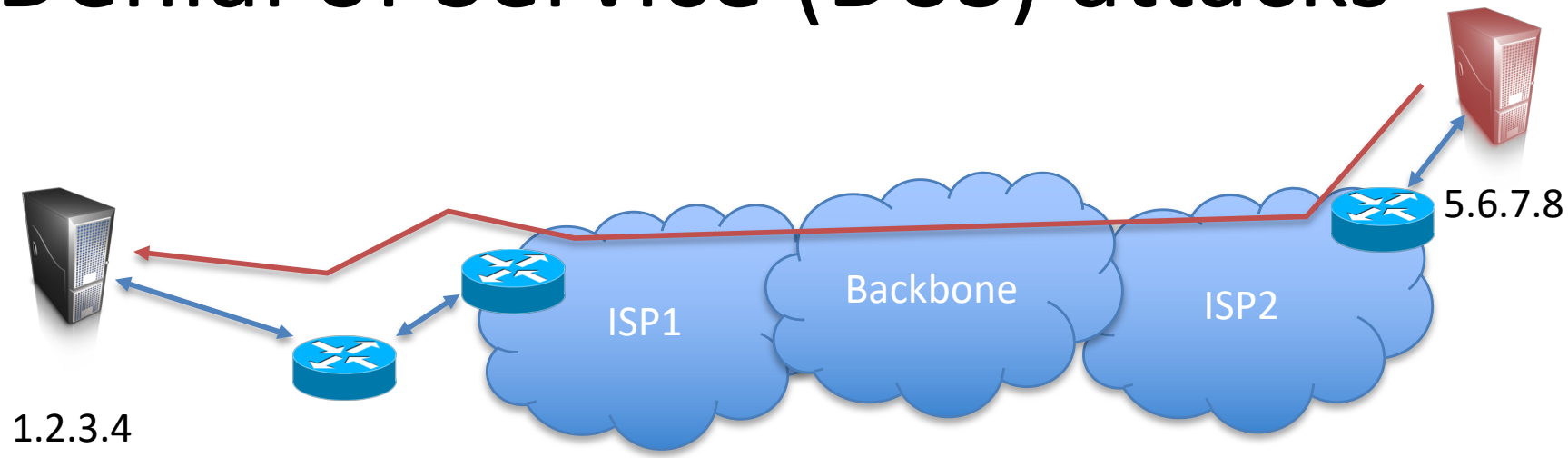
DoS reflection attacks



Note a valid packet sends a reply to 8.7.3.4

- Attacker can bounce an attack against 8.7.3.4 off 1.2.3.4
- “Frame” 1.2.3.4
- Single-packet exploit (1.2.3.4 in foreign country)

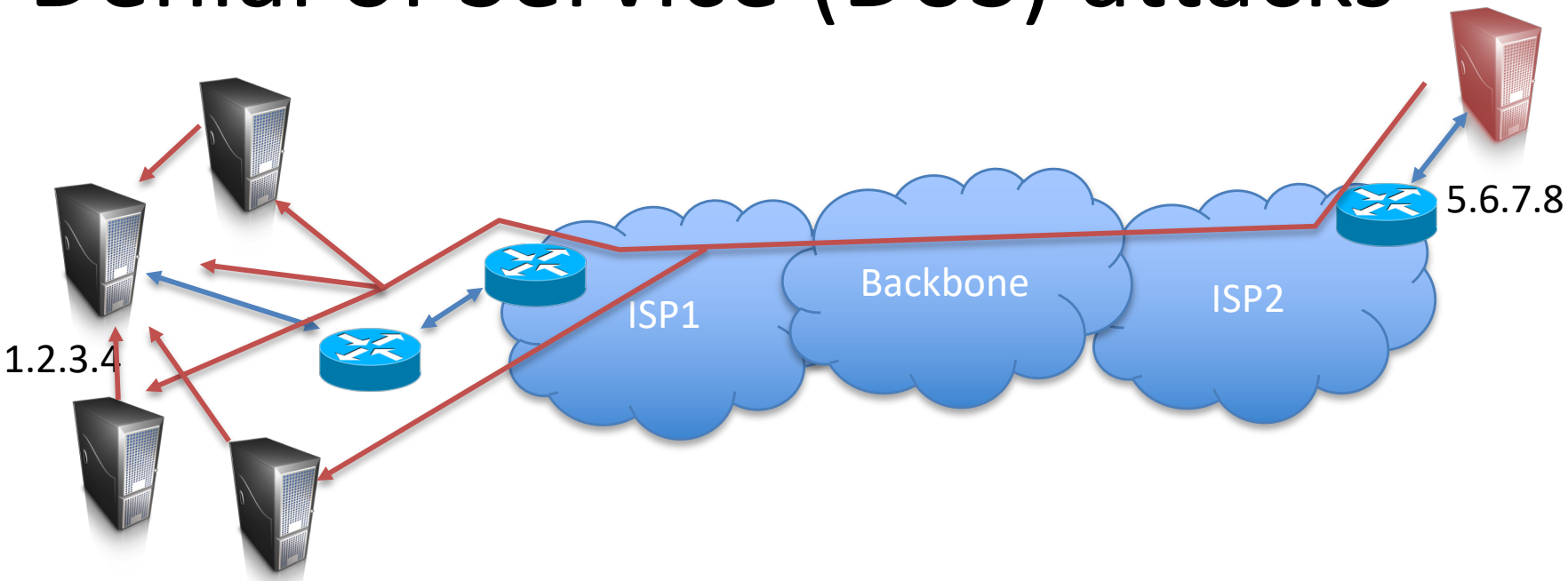
Denial of Service (DoS) attacks



DoS works better when there is ***asymmetry*** between victim and attacker

- Attacker uses few resources to cause victim to consume lots of resources

Denial of Service (DoS) attacks



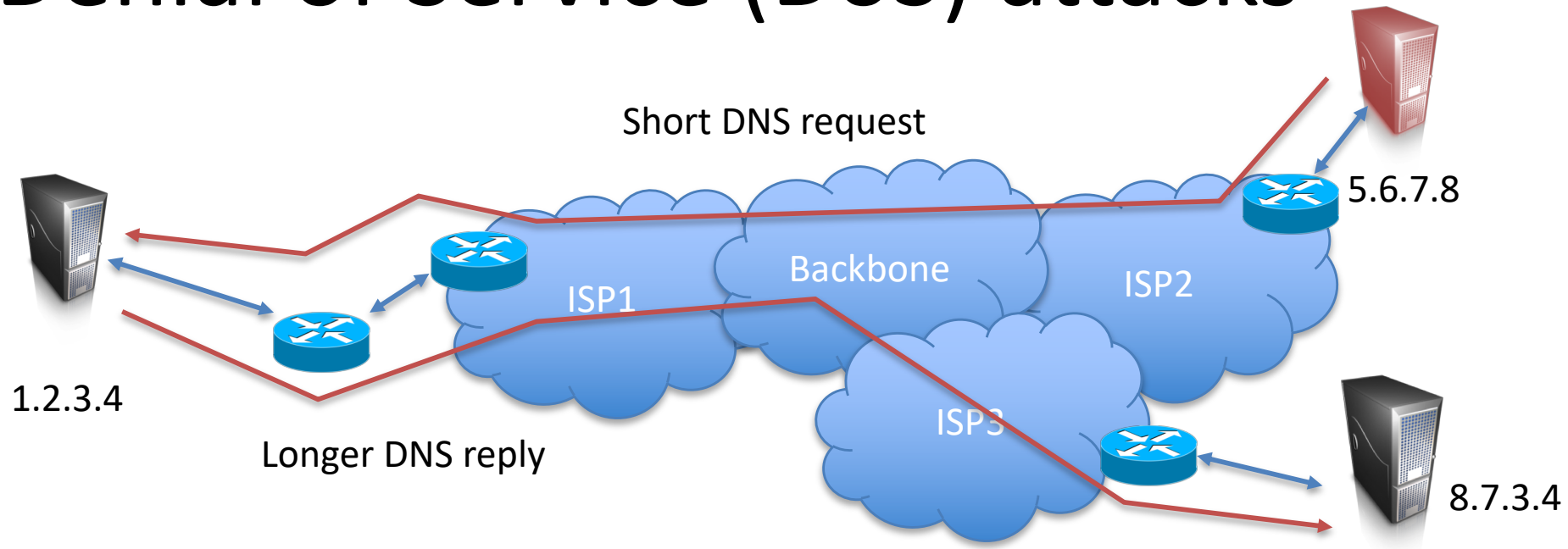
DoS works better when there is ***asymmetry*** between victim and attacker

- Attacker uses few resources to cause victim to consume lots of resources

Old example: Smurf attack

Router allows attacker to send broadcast ICMP ping on network. Attacker spoofs SRC address to be 1.2.3.4

Denial of Service (DoS) attacks



DoS works better when there is ***asymmetry*** between victim and attacker

- Attacker uses few resources to cause victim to consume lots of resources

More recent: DNS reflection attacks

Send DNS request w/ spoofed target IP (~65 byte request)

DNS replies sent to target (~512 byte response)