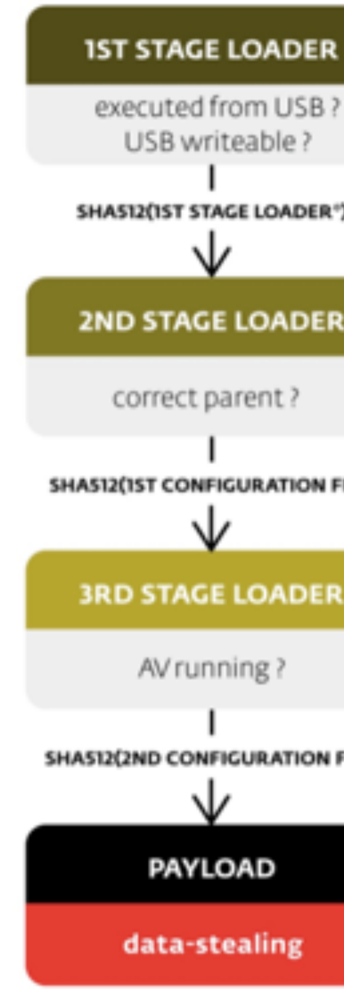
ars technica

Stealthy malware targeting air-gapped PCs leaves no trace of infection

Researchers discover "self-protecting" trojan circulating in the wild.

by Dan Goodin - Mar 24, 2016 5:00pm CDT



Researchers have discovered highly stealthy malware that can infect computers not connected to the Internet and leaves no evidence on the computers it compromises.

USB Thief gets its name because it spreads on USB thumb and hard drives and steals huge volumes of data once it has taken hold. Unlike previously discovered USB-born malware, it uses a series of novel techniques to bind itself to its host drive to ensure it can't easily be copied and analyzed. It uses a multi-staged encryption scheme that derives its key from the device ID of the USB drive. A chain of loader files also contains a list of file names that are unique to every instance of the malware. Some of the file names are based on the precise file content and the time the file was created. As a result, the malware won't execute if the files are moved to a drive other than the one chosen by the original developers.

Gardoň wrote:

> It was quite challenging to analyze this malware because we had no access to any malicious USB device. Moreover, we had no dropper, so we could not create a suitably afflicted USB drive under controlled conditions for further analysis.
>
> Only the submitted files can be analyzed, so the unique device ID had to be brute-forced and combined with common USB disk properties. Moreover, after successful decryption of the malware files, we had to find out the right order of the executables and configuration files, because the file copying process to get the samples to us had changed the file creation timestamp on the samples.

passwords & network security

cs642

computer security

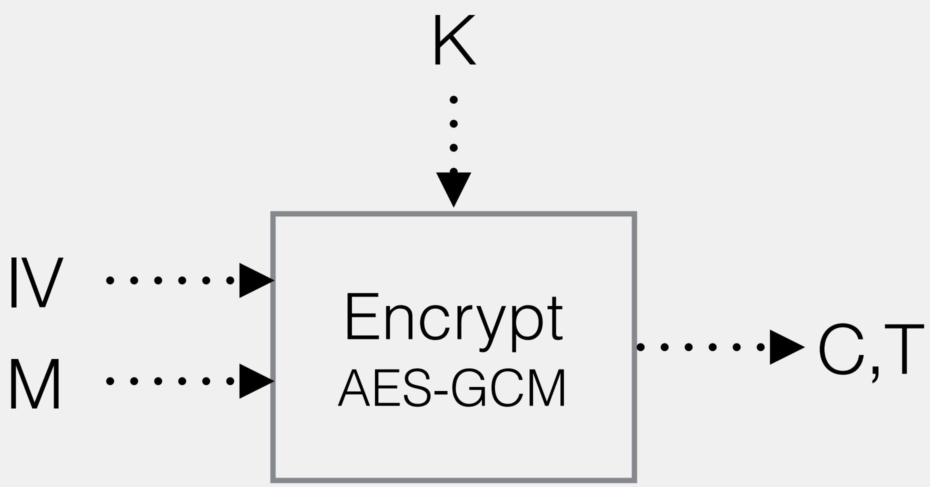
adam everspaugh
ace@cs.wisc.edu

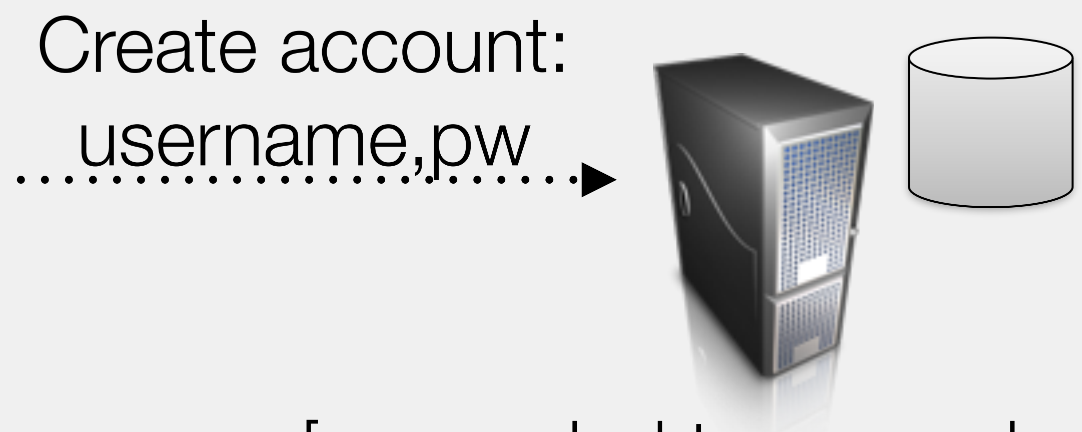
today

* Passwords
* Network security intro
    / Ethernet, MAC, ARP, WiFi

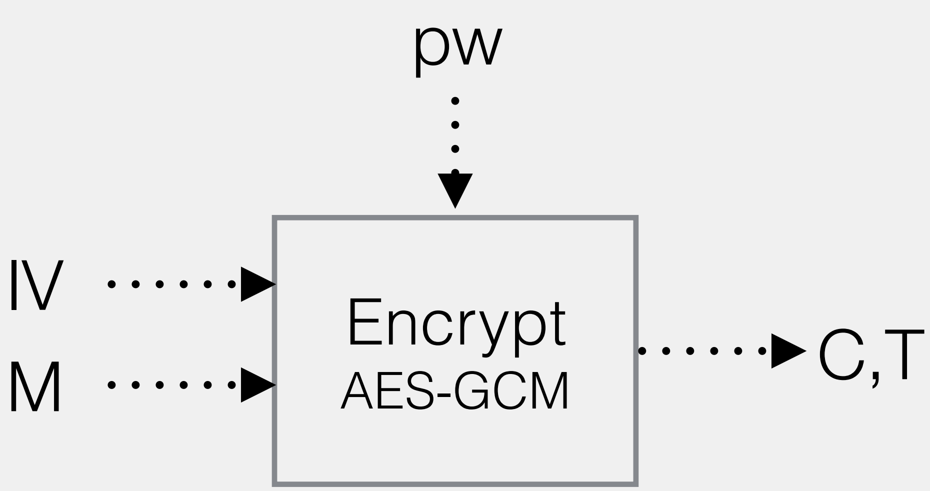
passwords

pw use cases





[server, desktop, or web service]

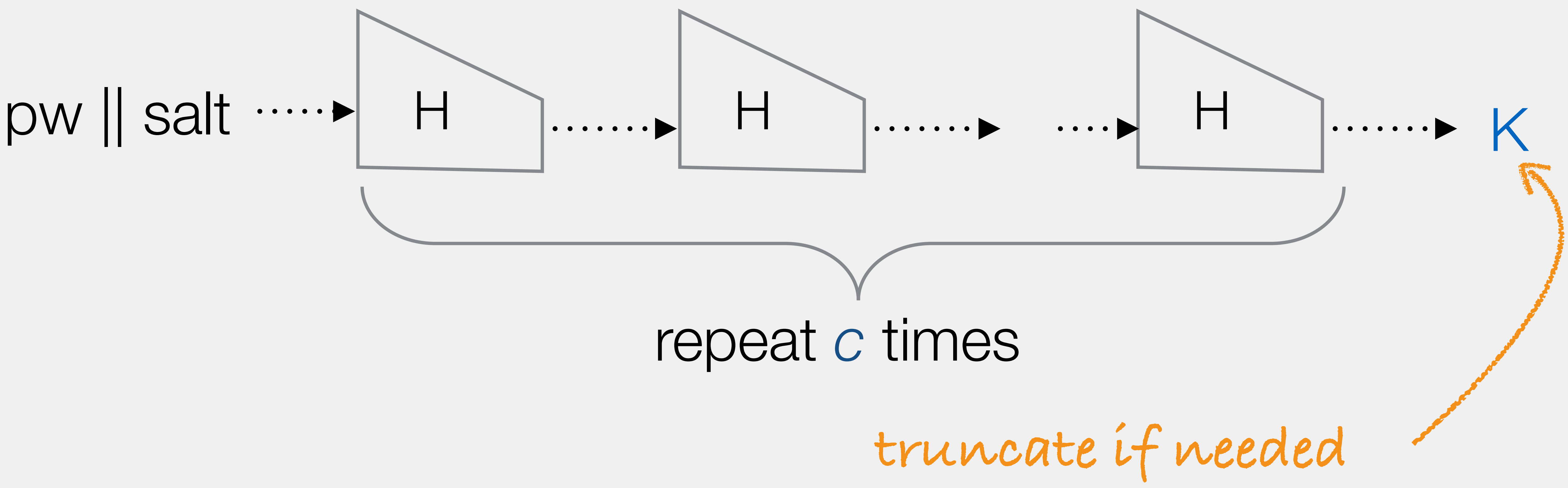


How does the server store the pw?

Password-based symmetric encryption

[password-based key derivation function]

PBKDF(pw, salt):



pbkdf

pw-based encryption

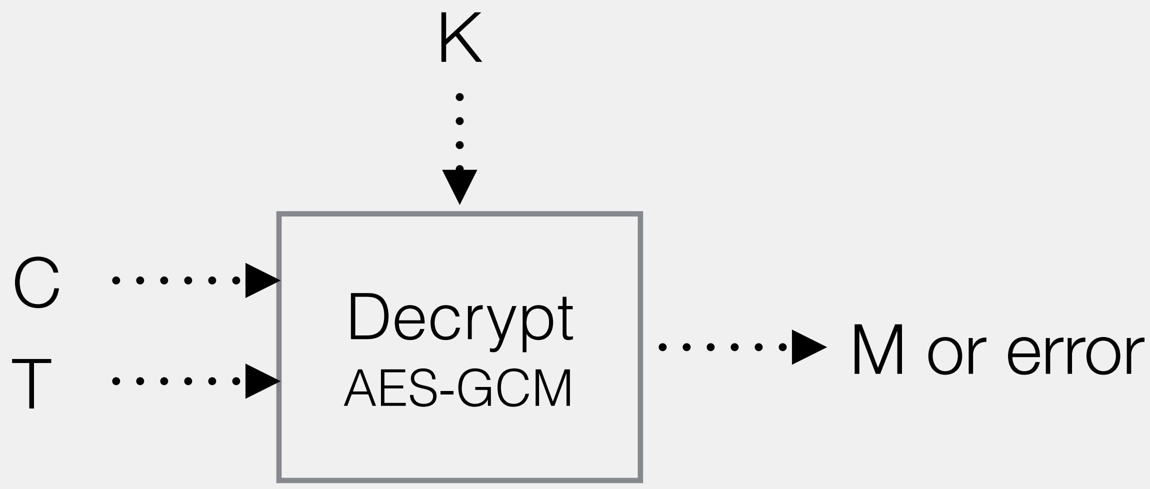
Enc(pw,M,R):	Dec(pw,C):
salt || R' = R	salt || C' = C
K = PBKDF(pw,salt)	K = PBKDF(pw,salt)
C = Enc'(K,M,R')	M = Dec'(K,C')
Return (salt,C)	Return M

Enc'/Dec' is some authenticated encryption scheme,
like AES-GCM

PBKDF + symmetric encryption → pw-based encryption

Attacks?

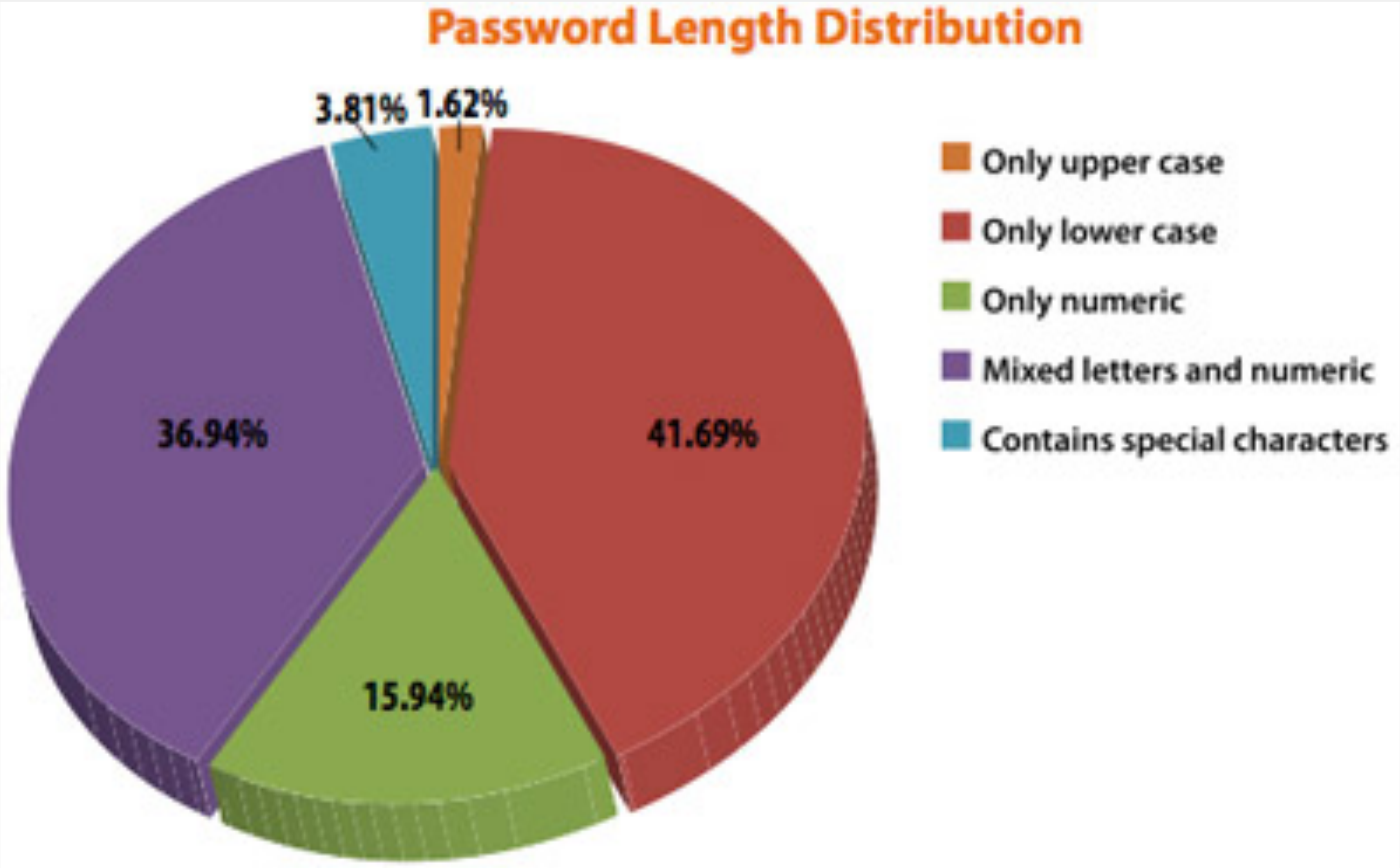
dictionary attack




```
DictionaryAttack(D,C,T):
for pw* in D:
   M* = Dec(pw*,C,T)
   if M* ≠ error:
     return pw,M*
```

- Given an authenticated encryption output (C,T), dictionary *D* of possible password
- Enumerate *D* in order of likelihood
- Test each candidate password

pw distribution



From an Imperva study of released RockMe.com password database (2010)

password storage

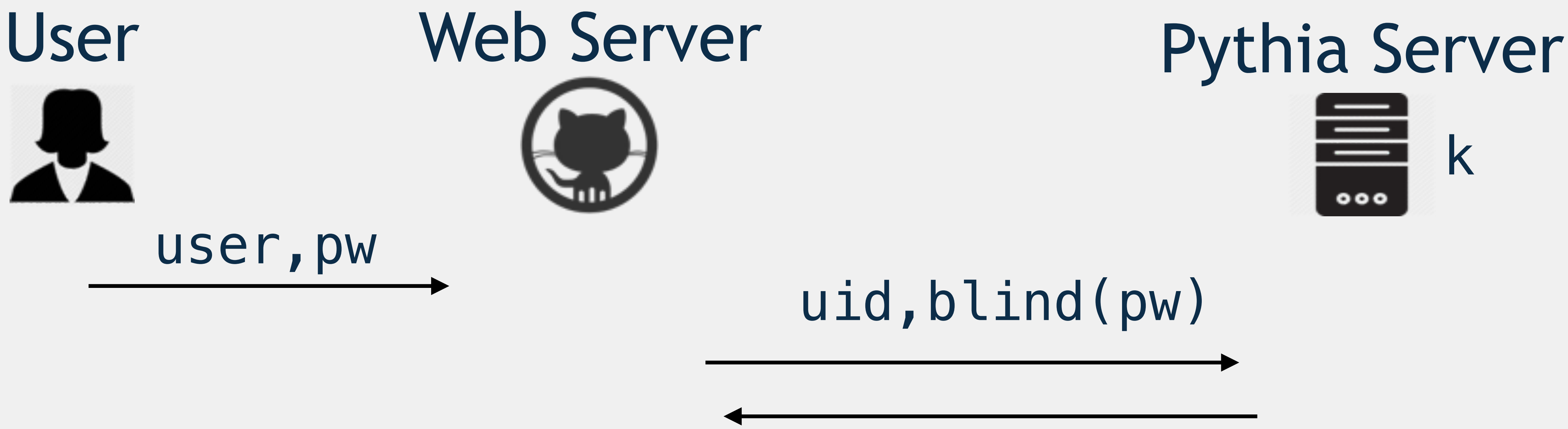
- Password storage + PBKDF
- Increase number of iterations: H^c(salt || pw)
- Use a slower computation
  - scrypt, bcrypt
  - Slower than SHA2, use lots of memory, hard to parallelize
- Costs? Benefits?

Facebook's Password Onion

```
$cur  = 'password'
$cur  = md5($cur)
$salt = randbytes(20)
$cur  = hmac_sha1($cur, $salt)
$cur  = remote_hmac_sha256($cur, $secret)
$cur  = scrypt($cur, $salt)
$cur  = hmac_sha256($cur, $salt)
```

[A. Muffet, https://video.adm.ntnu.no/pres/54b660049af94]

Protecting passwords



$z = F_k(uid, pw)$

Separates password database and key
Permits key rotation without changing passwords

[The Pythia PRF Service, 2015, Everspaugh, et. al]

INTERMISSION

Getting started on network security

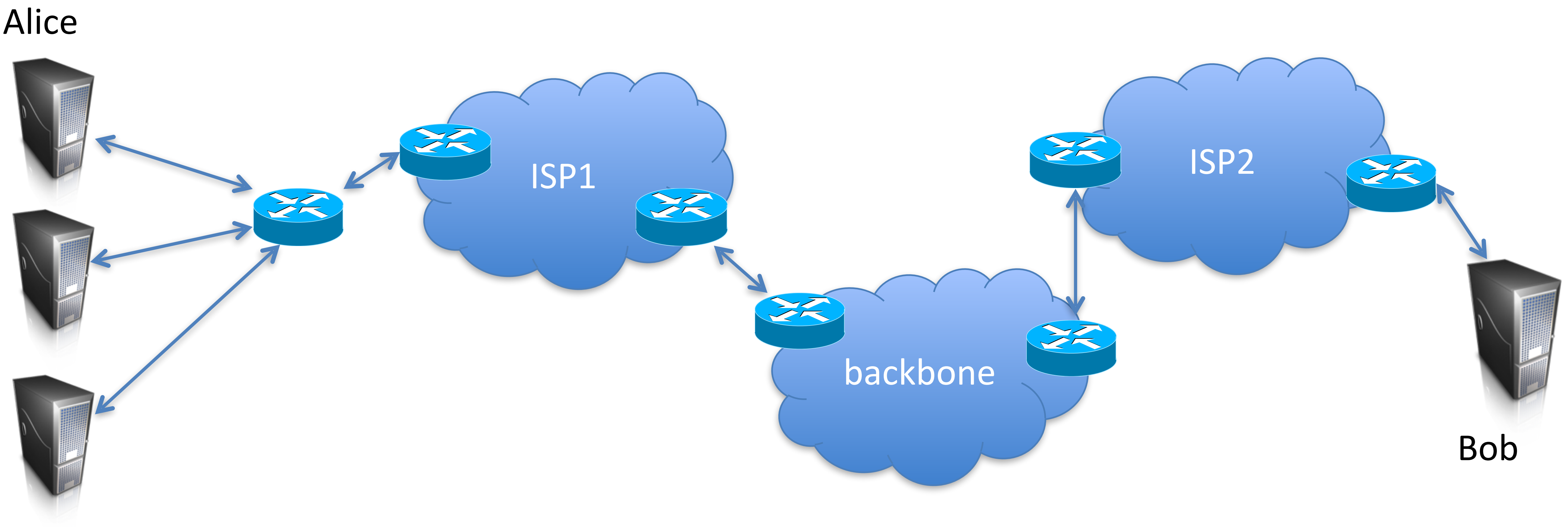
Internet protocol stack

Man-in-the-middle

Address resolution protocol and ARP spoofing

802.11

Internet



Local area network (LAN)

Ethernet

802.11

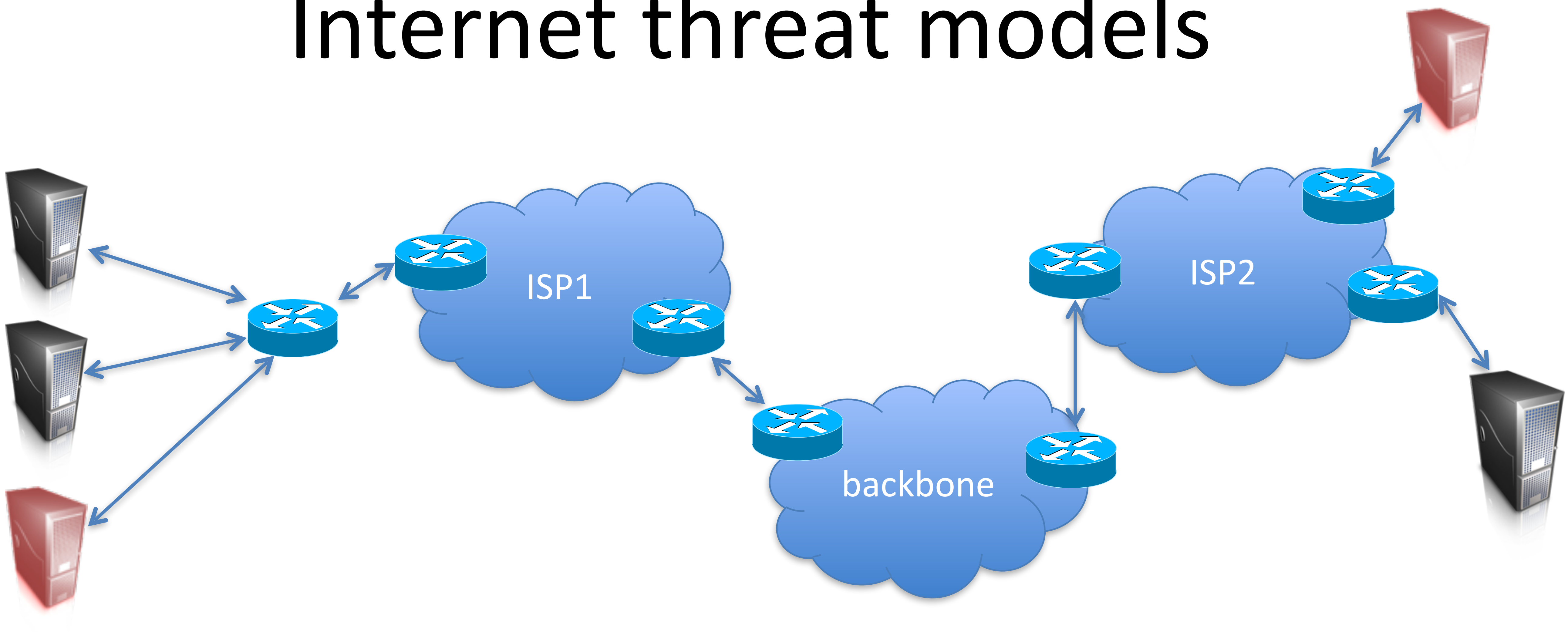
Internet

TCP/IP

BGP (border gateway protocol)

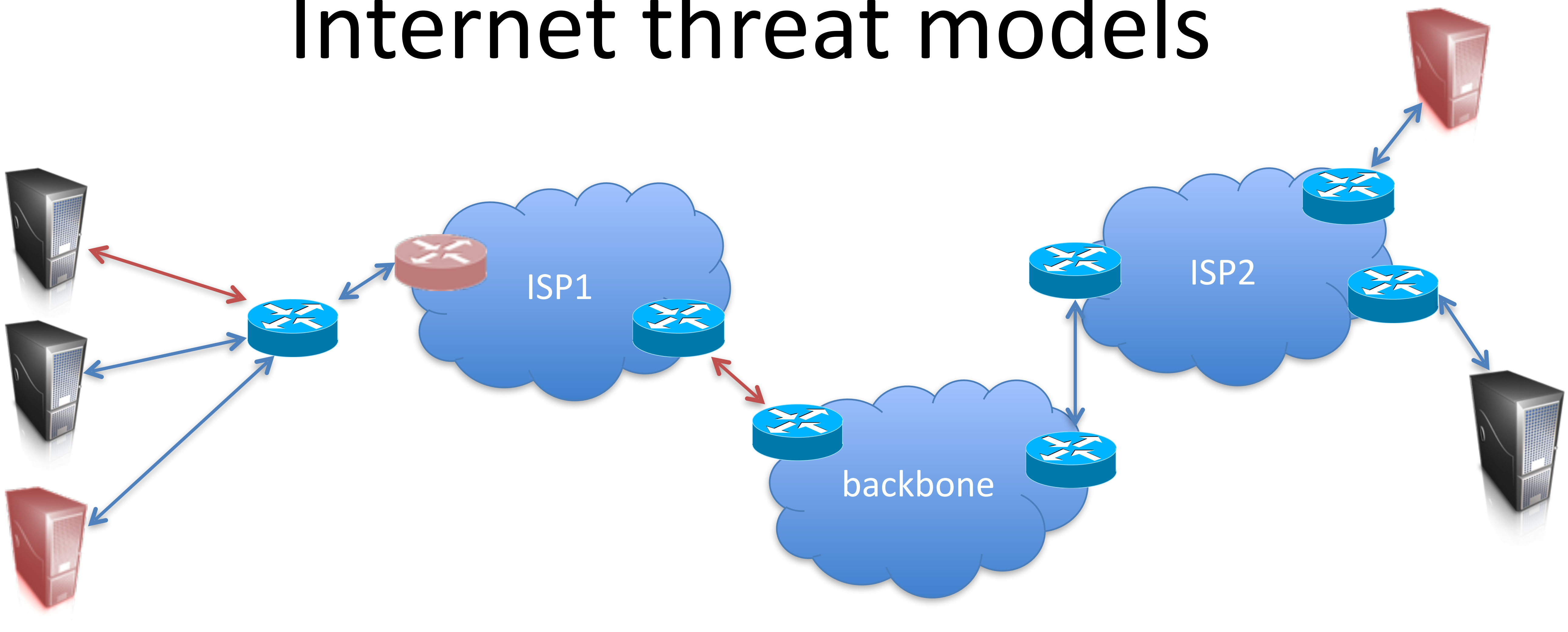
DNS (domain name system)

Internet threat models



(1) Malicious hosts

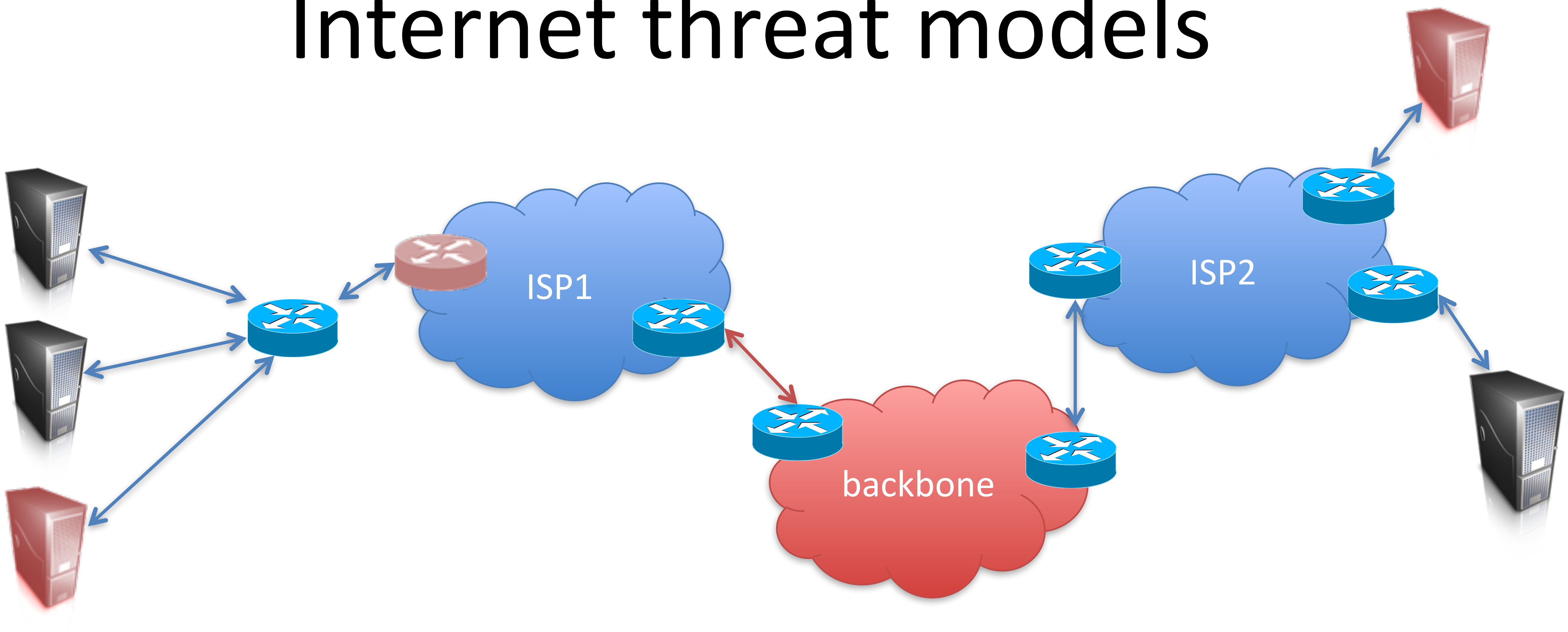
Internet threat models



(1) Malicious hosts

(2) Subverted routers or links

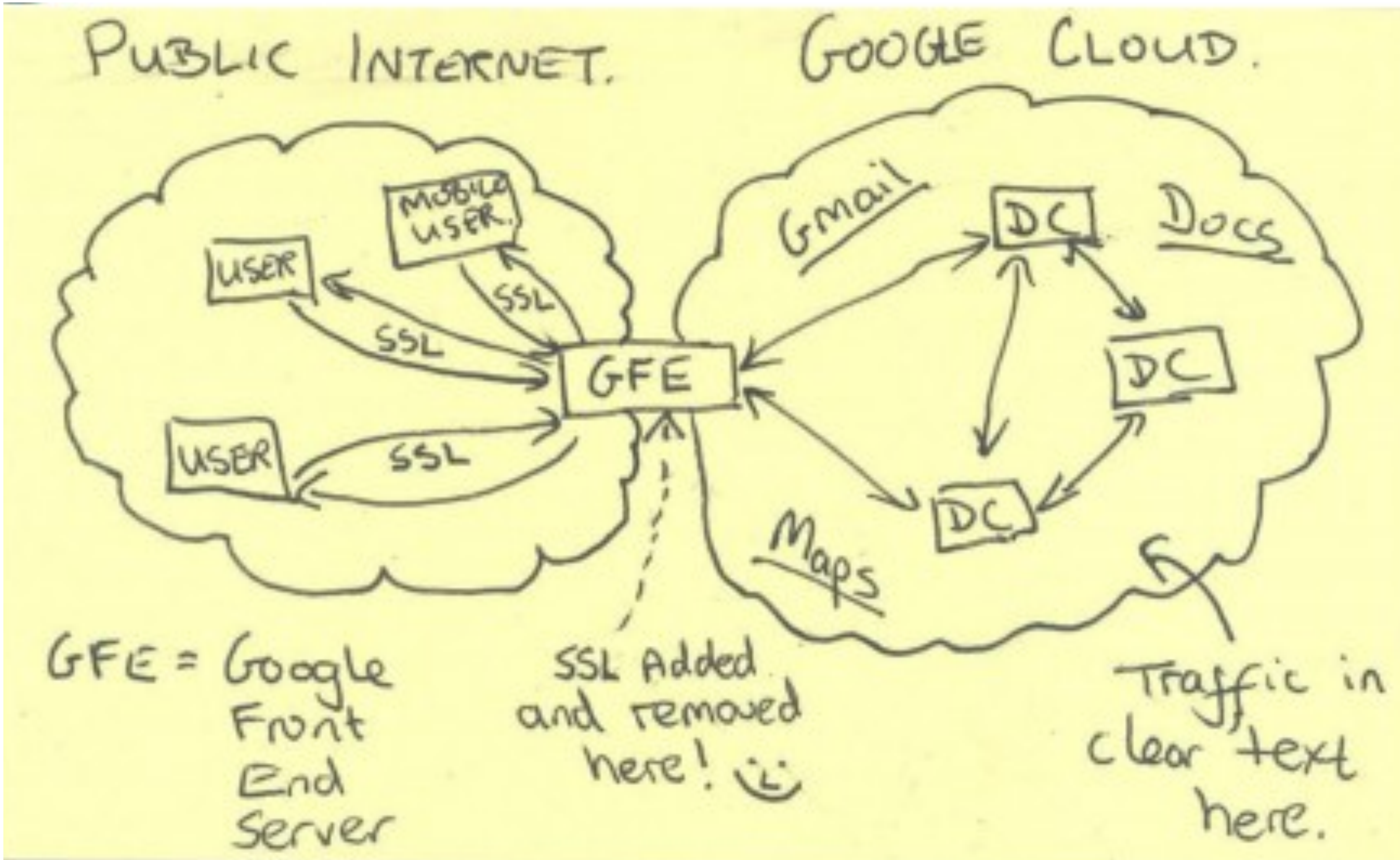
Internet threat models



(1) Malicious hosts

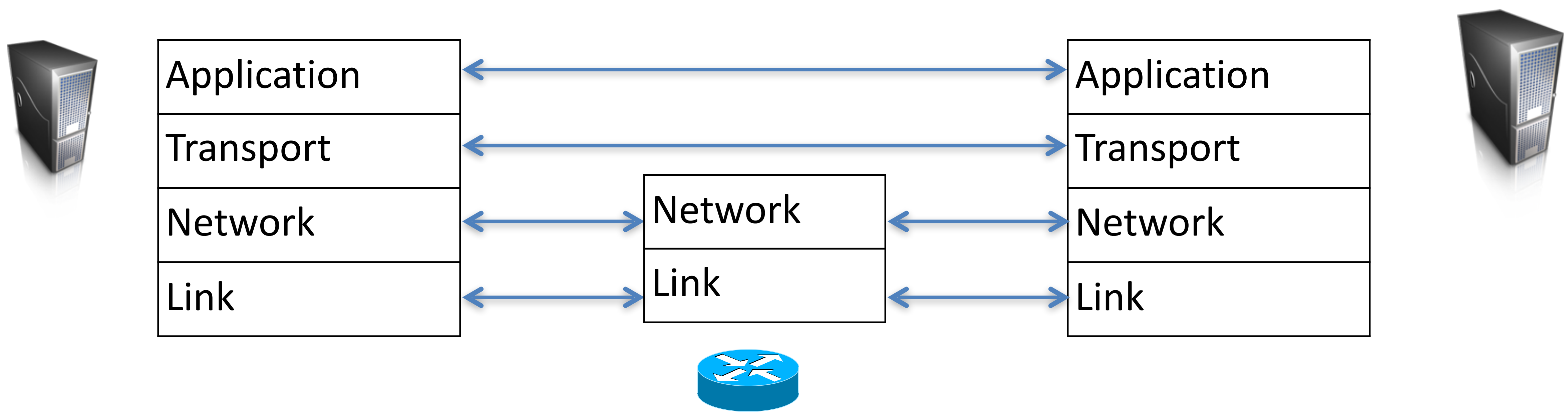
(2) Subverted routers or links

(3) Malicious ISPs or backbone

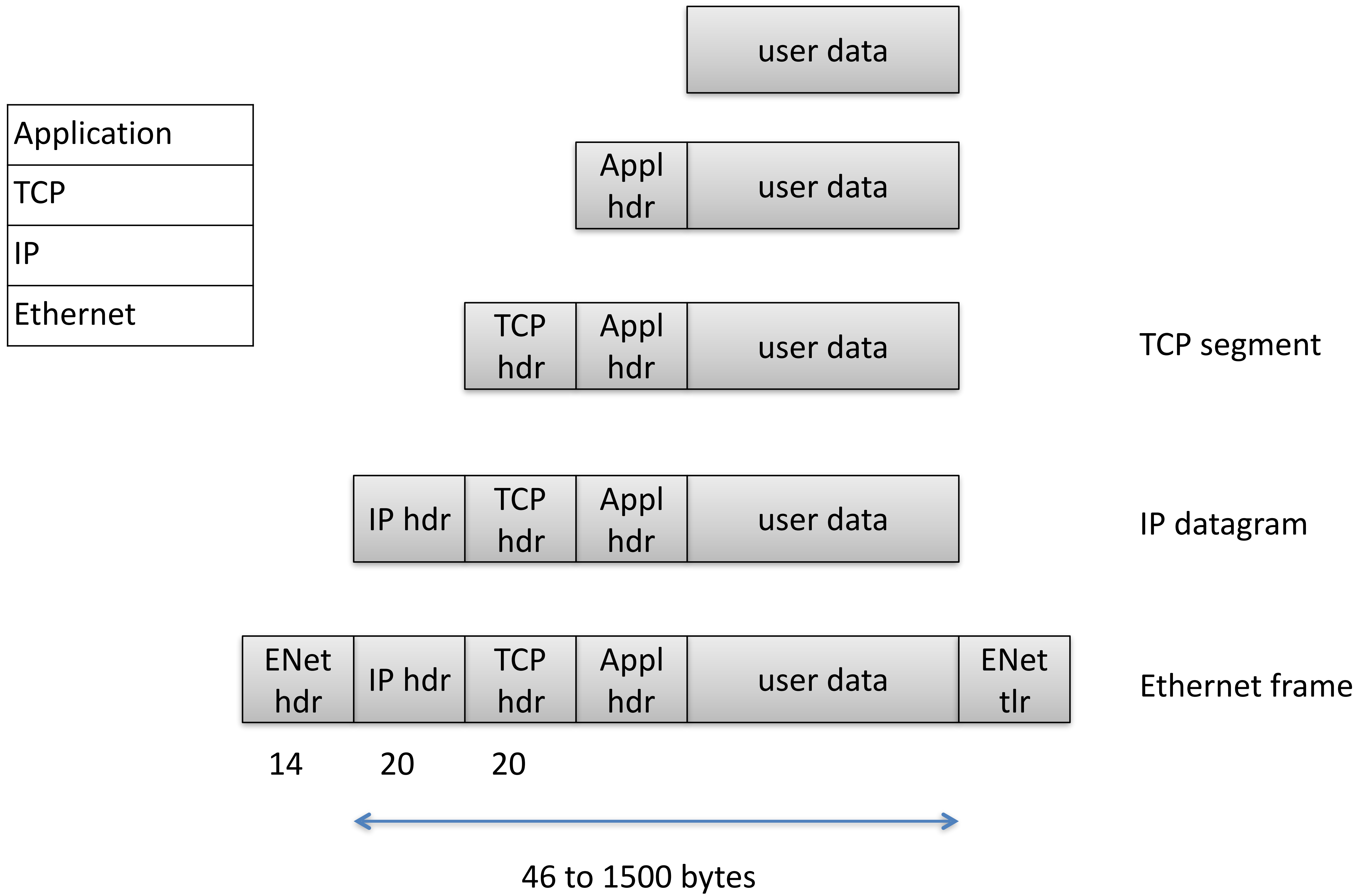

PUBLIC INTERNET.
GOOGLE CLOUD.
MOBILE USER.
USER
USER
SSL
SSL
SSL
GFE
Gmail
Docs
Maps
DC
DC
DC
GFE = Google Front End Server
SSL Added and removed here!
Traffic in clear text here.

Internet protocol stack

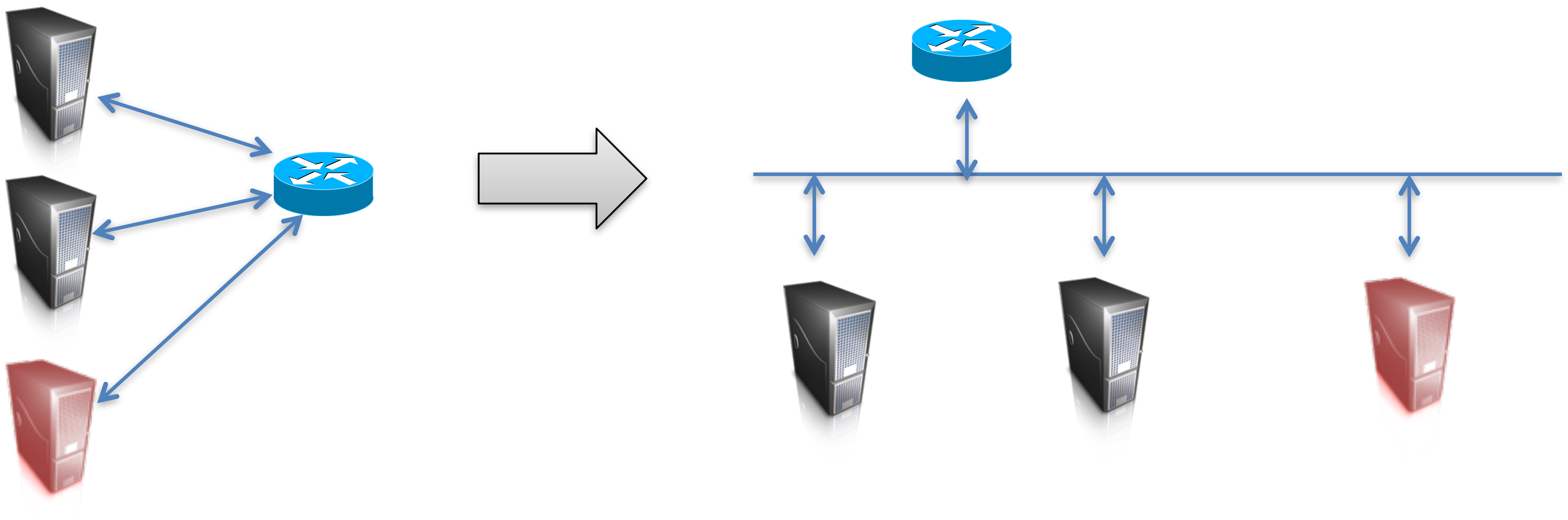
Application	HTTP, FTP, SMTP, SSH, etc.
Transport	TCP, UDP
Network	IP, ICMP, IGMP
Link	802x (802.11, Ethernet)



Internet protocol stack



Ethernet



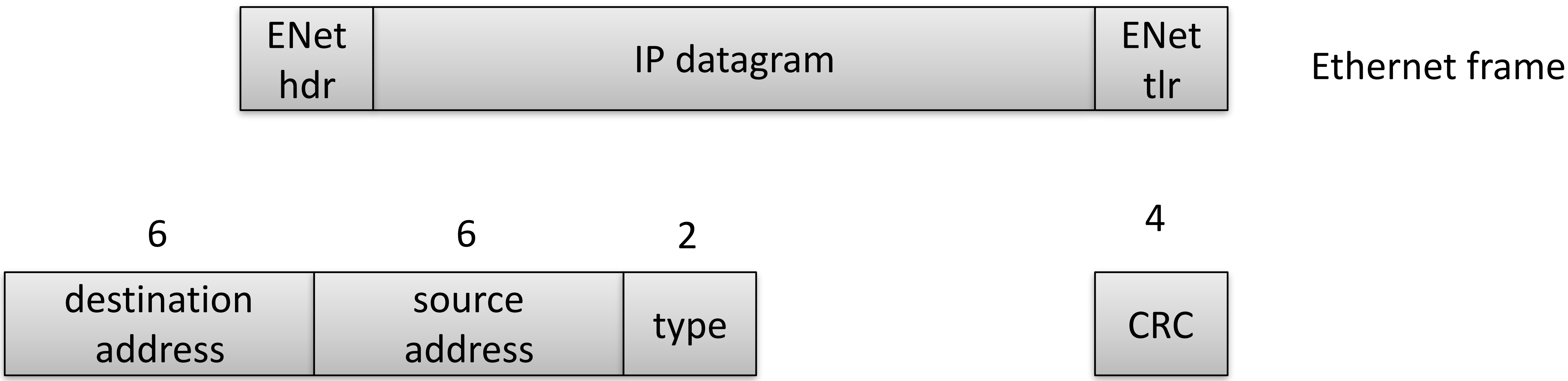
Carrier Sense, Multiple Access with Collision Detection (CSMA/CD)

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

3 byte 2 control bits & OID	3 byte NIC identifier

- Hardware (ethernet card/WiFi card) initialized with MAC address
- But: most network cards permit changing MAC address

MAC spoofing

- Many LANs, WiFis use MAC-based access controls

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
```

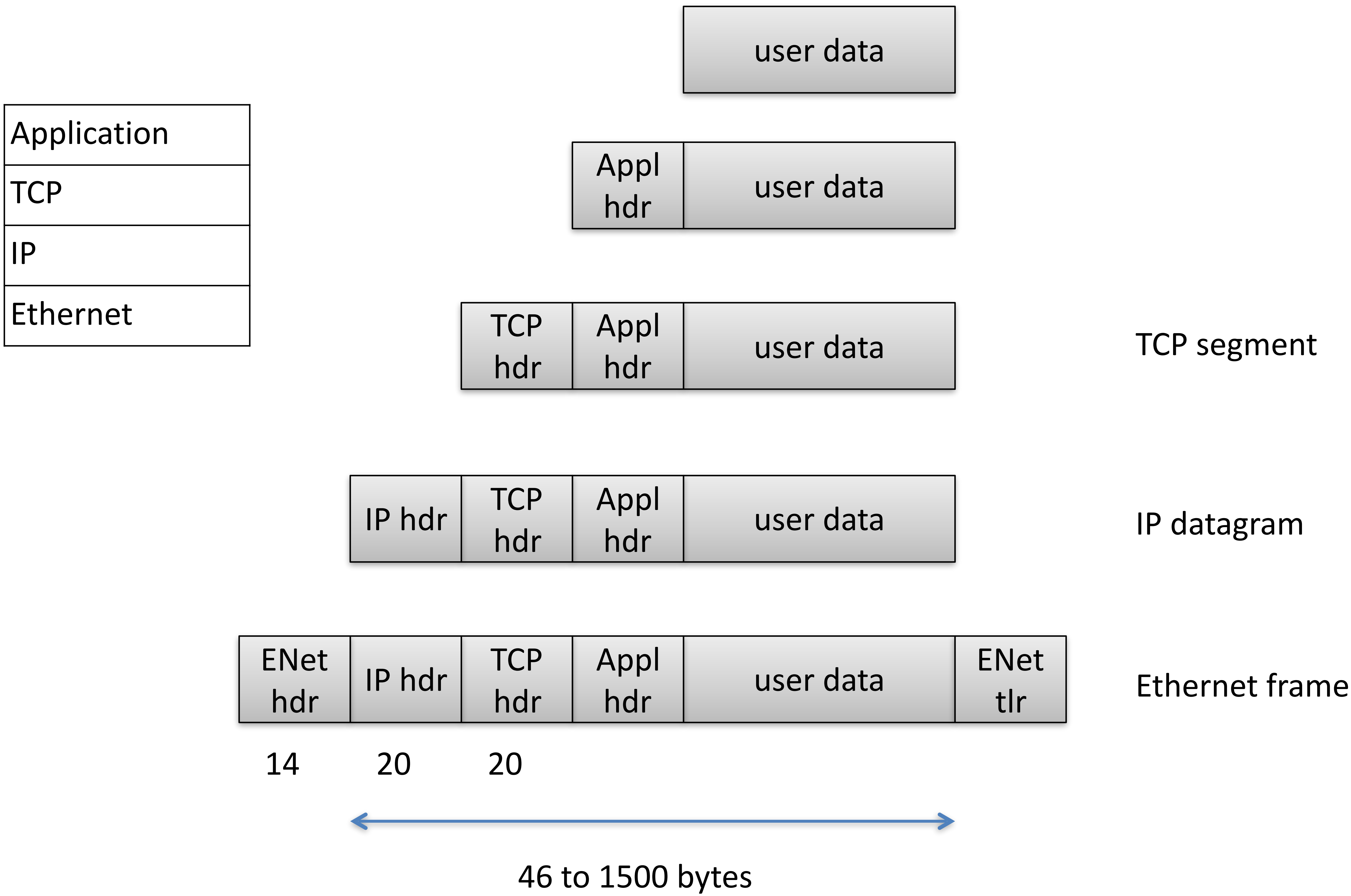
or

```
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

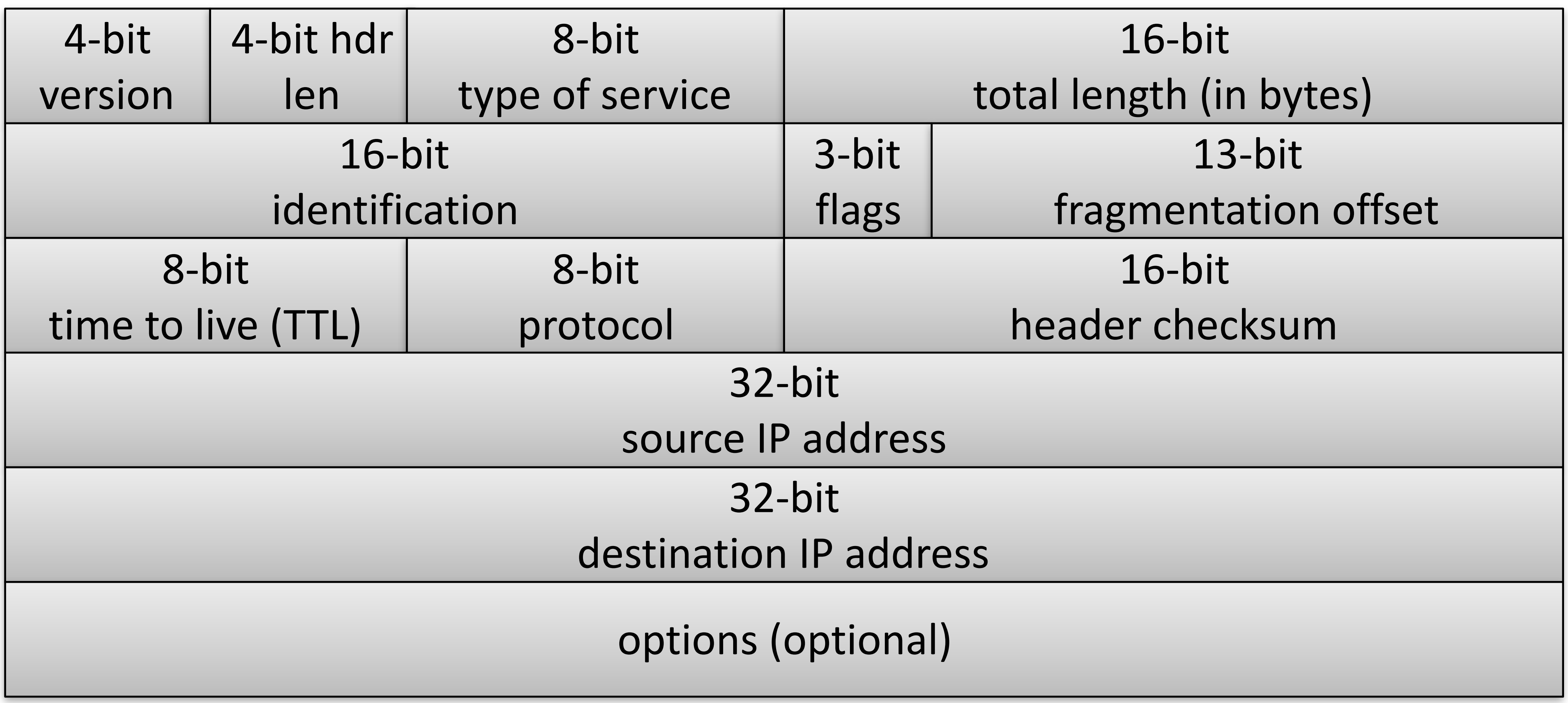
Courtesy of wikibooks
http://en.wikibooks.org/wiki/Changing_Your_MAC_Address/Mac_OS_X

Internet protocol stack

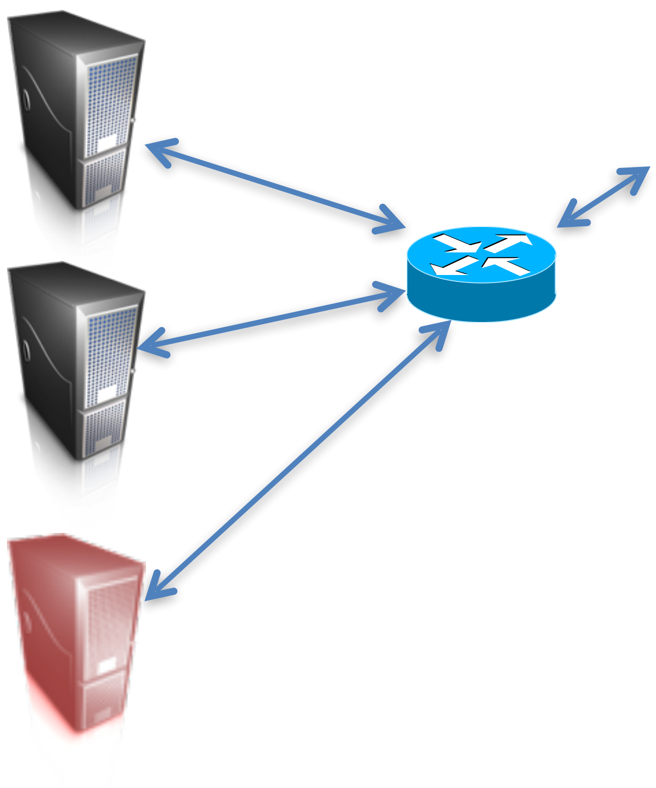


IPv4

Ethernet frame containing IP datagram



Address resolution protocol



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

Link layer and IP must cooperate to
route packets

32-bit IP address

ARP ↓

48-bit MAC address

ARP enables this cooperation by
mapping IPs to MACs

Address resolution protocol

6	6	2	2	2	1	1	2
enet dest	enet src	type	hw type	prot type	hw size	prot size	op

enet sender	ip sender	enet target	ip target	pad	CRC
6	4	6	4	18	4

frame type = 0x0806 (ARP)

enet dest is 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

ARP caches

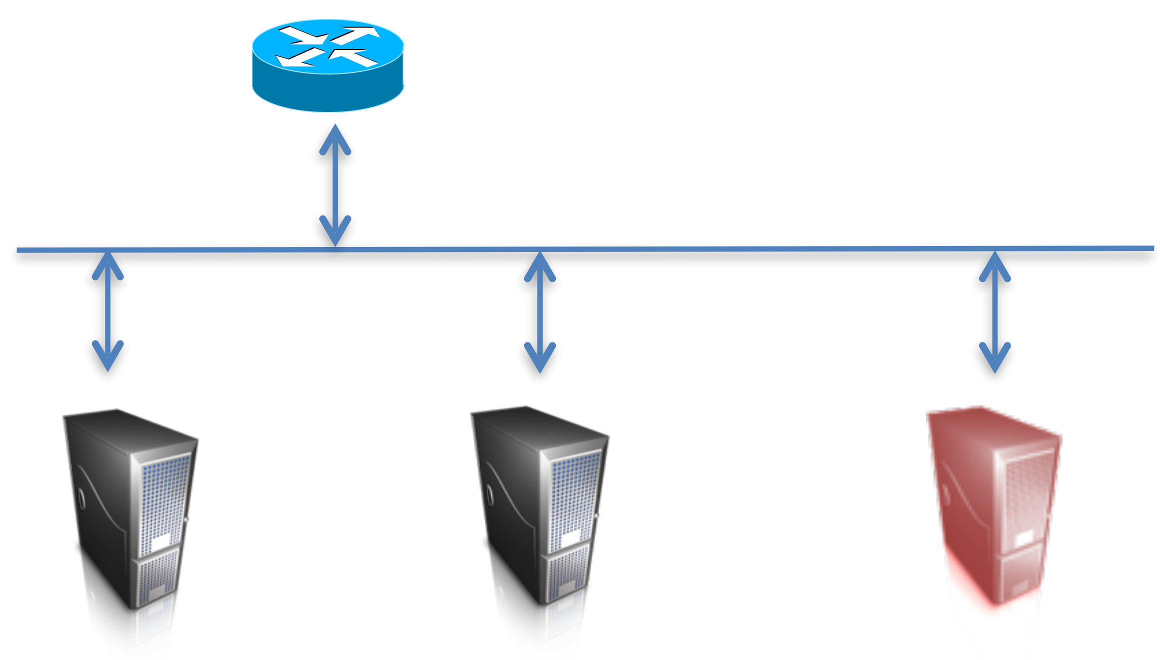
- Hosts maintain cache of ARP data
  – just a table mapping between IPs and MACs

```
rist@wifi-212:~/work/teaching/642-fall-2011/slides$ arp -a
? (172.16.219.1) at 0:50:56:c0:0:1 on vmnet1 ifscope permanent [ethernet]
? (172.16.219.255) at (incomplete) on vmnet1 ifscope [ethernet]
? (192.168.1.1) at 98:fc:11:91:73:92 on en1 ifscope [ethernet]
? (192.168.1.255) at (incomplete) on en1 ifscope [ethernet]
? (192.168.38.255) at (incomplete) on vmnet8 ifscope [ethernet]
rist@wifi-212:~/work/teaching/642-fall-2011/slides$
```

ARP has no authentication

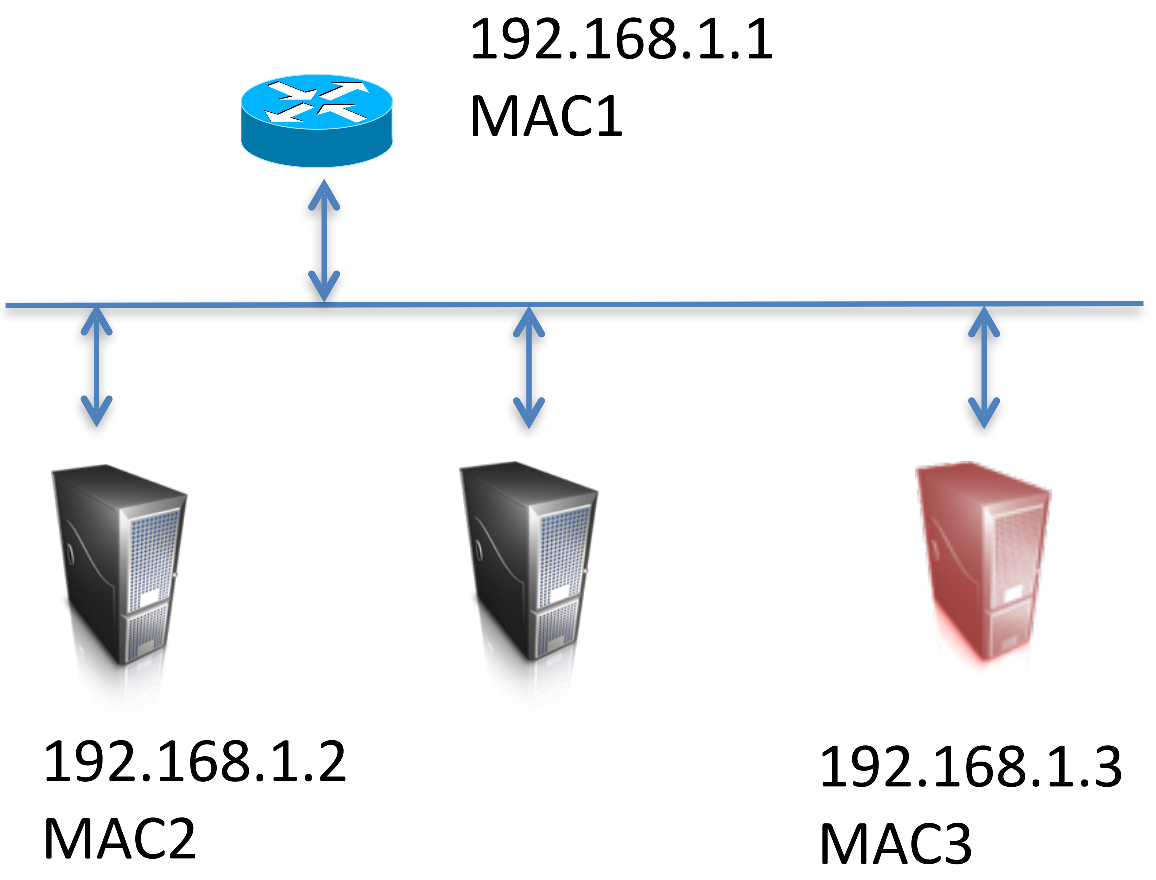
- Easy to sniff packets on (non-switched) ethernet
- What else can we do?

<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?



<u>Active Man-in-the-Middle:</u>

ARP reply to MAC2
192.168.1.1 -> MAC3

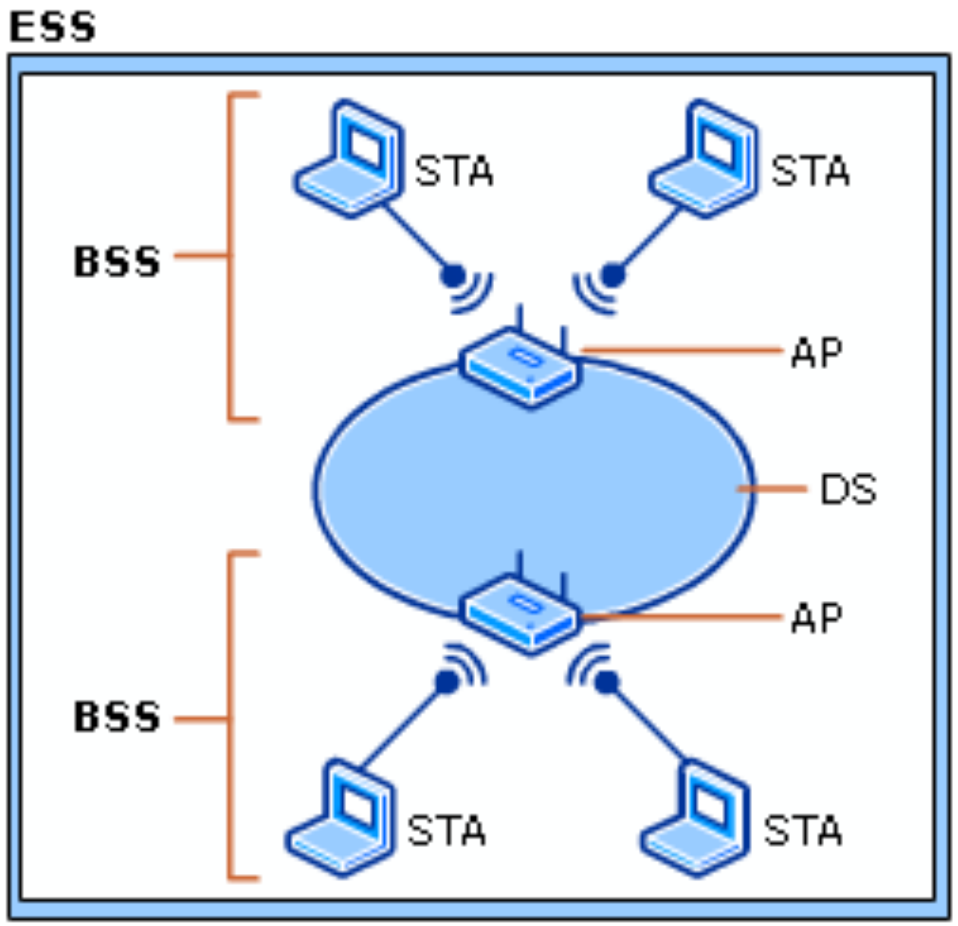
ARP reply to MAC1
192.168.1.2 -> MAC3

Now traffic "routed" through malicious box

802.11 (wifi)

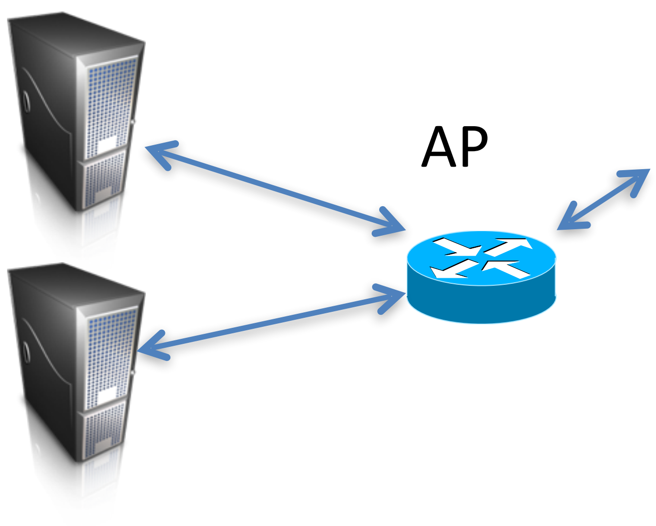
STA = station
BSS = basic service set
DS = distribution service
ESS = extended service set

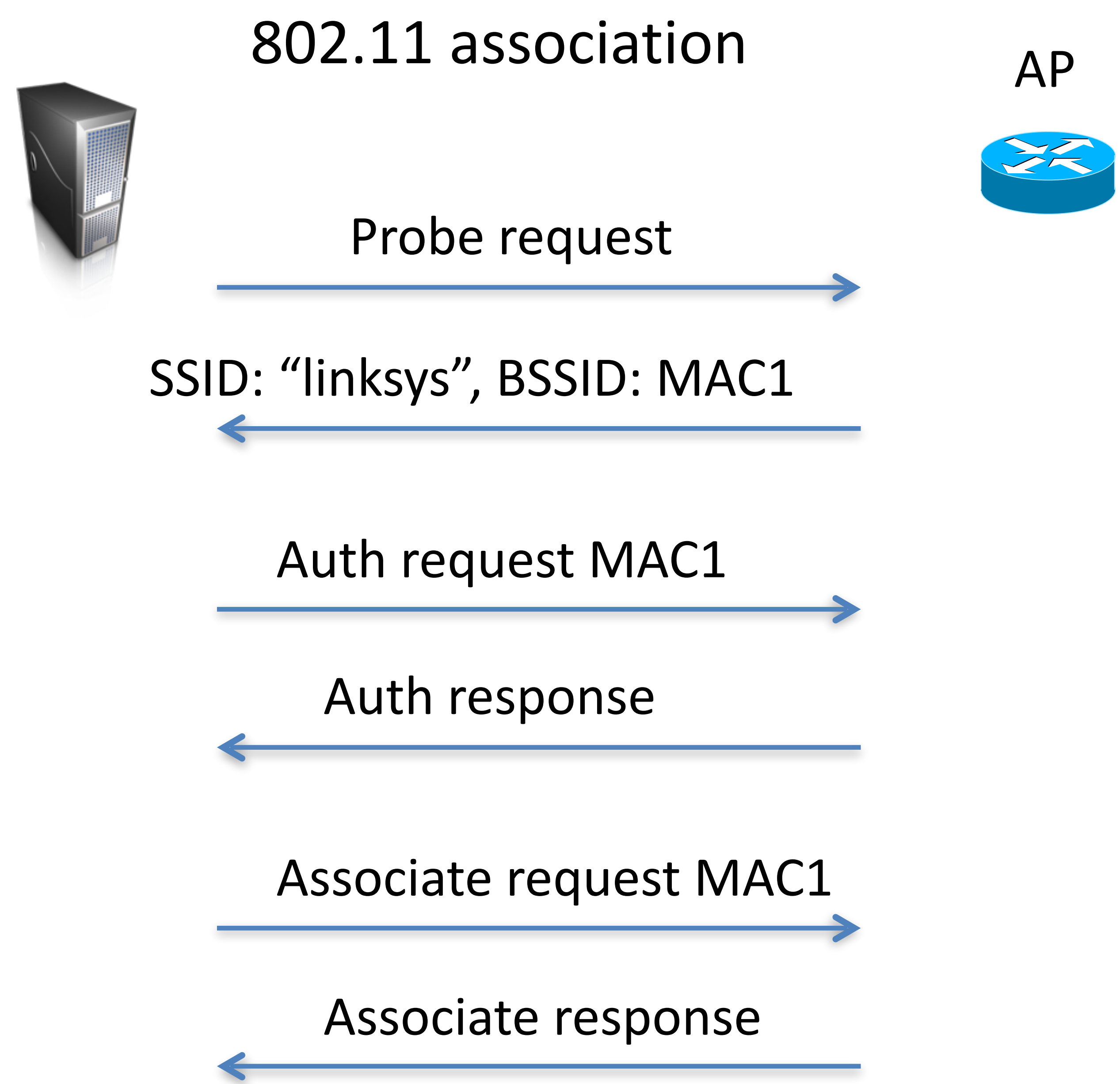
SSID (service set identifier)
identifies the 802.11 network



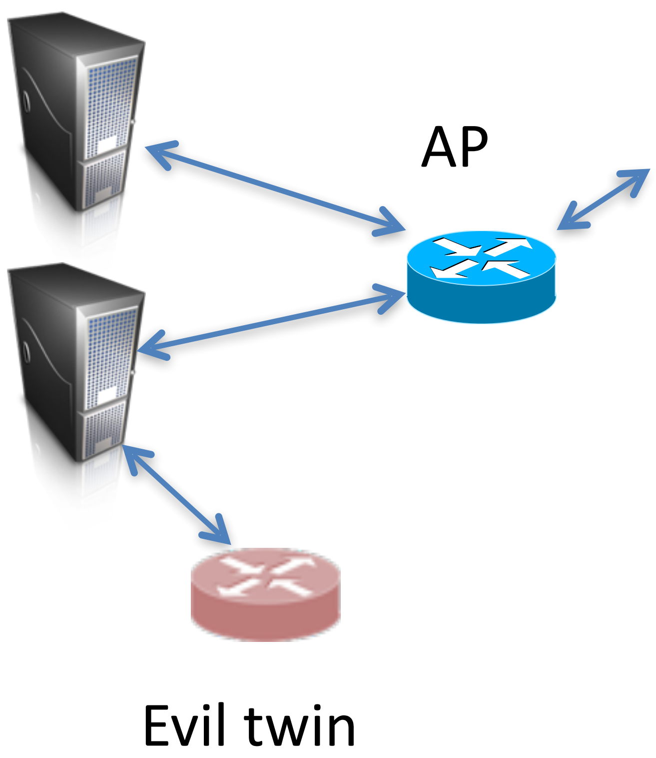
http://technet.microsoft.com/en-us/library/cc757419(WS.10).aspx

802.11 association





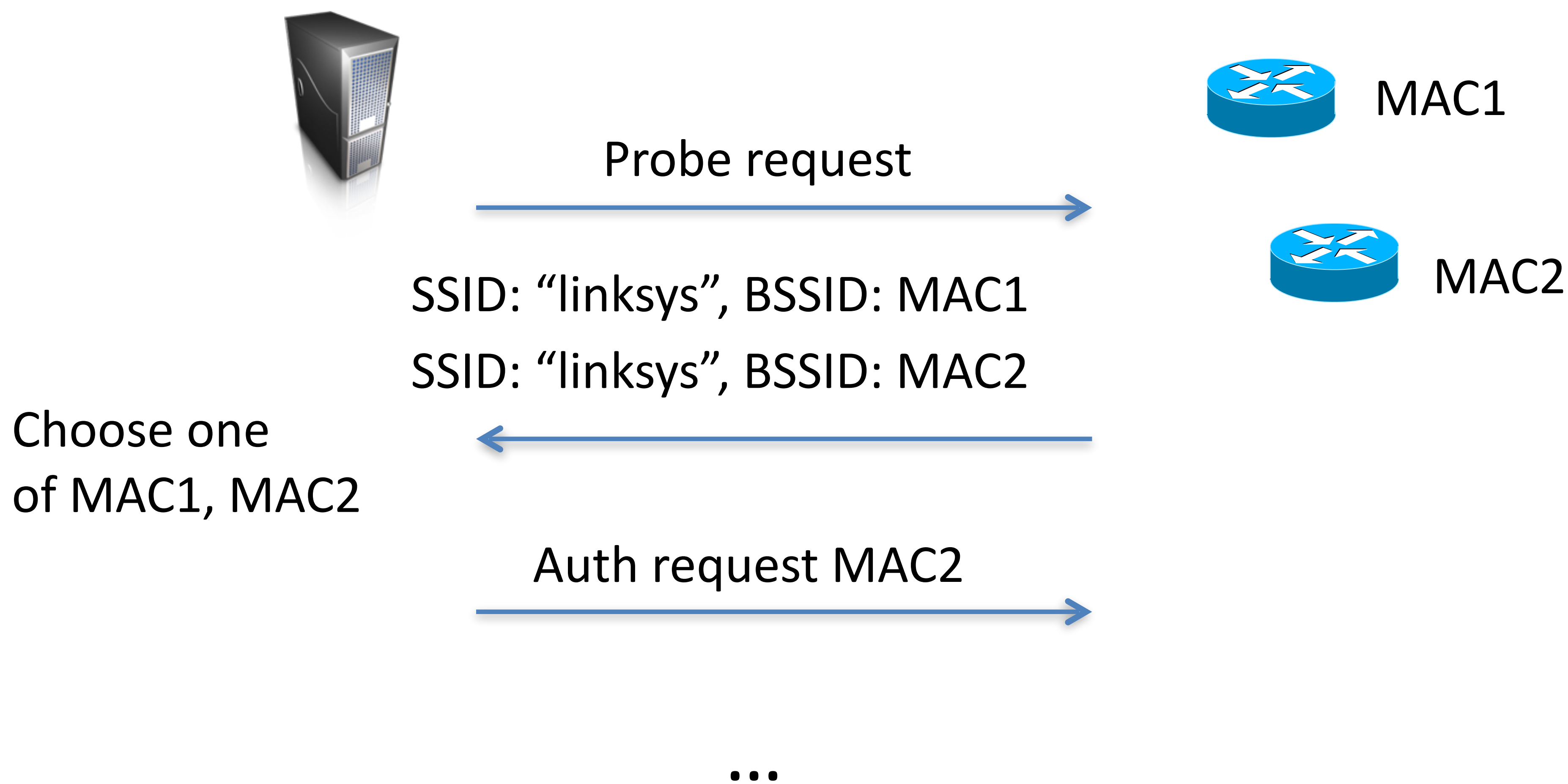
802.11 evil twins



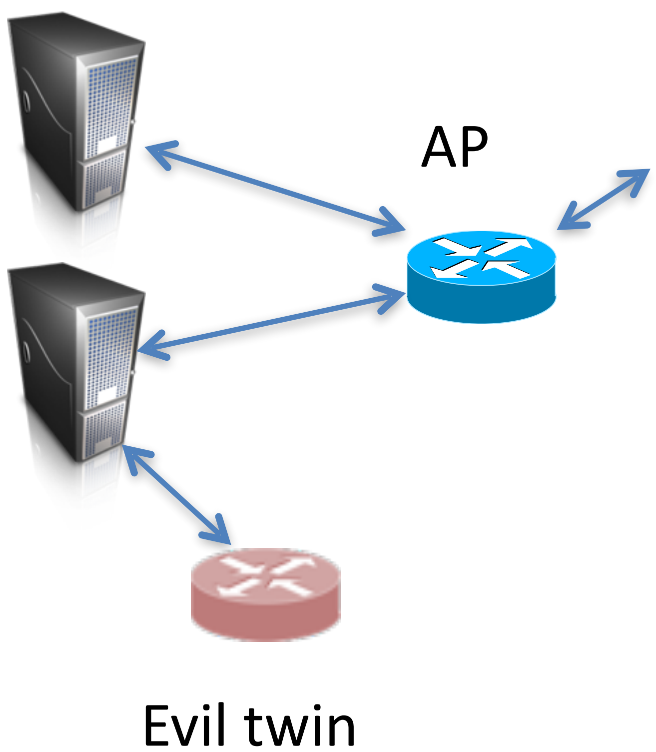
Basic idea:

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

Two APs for same network



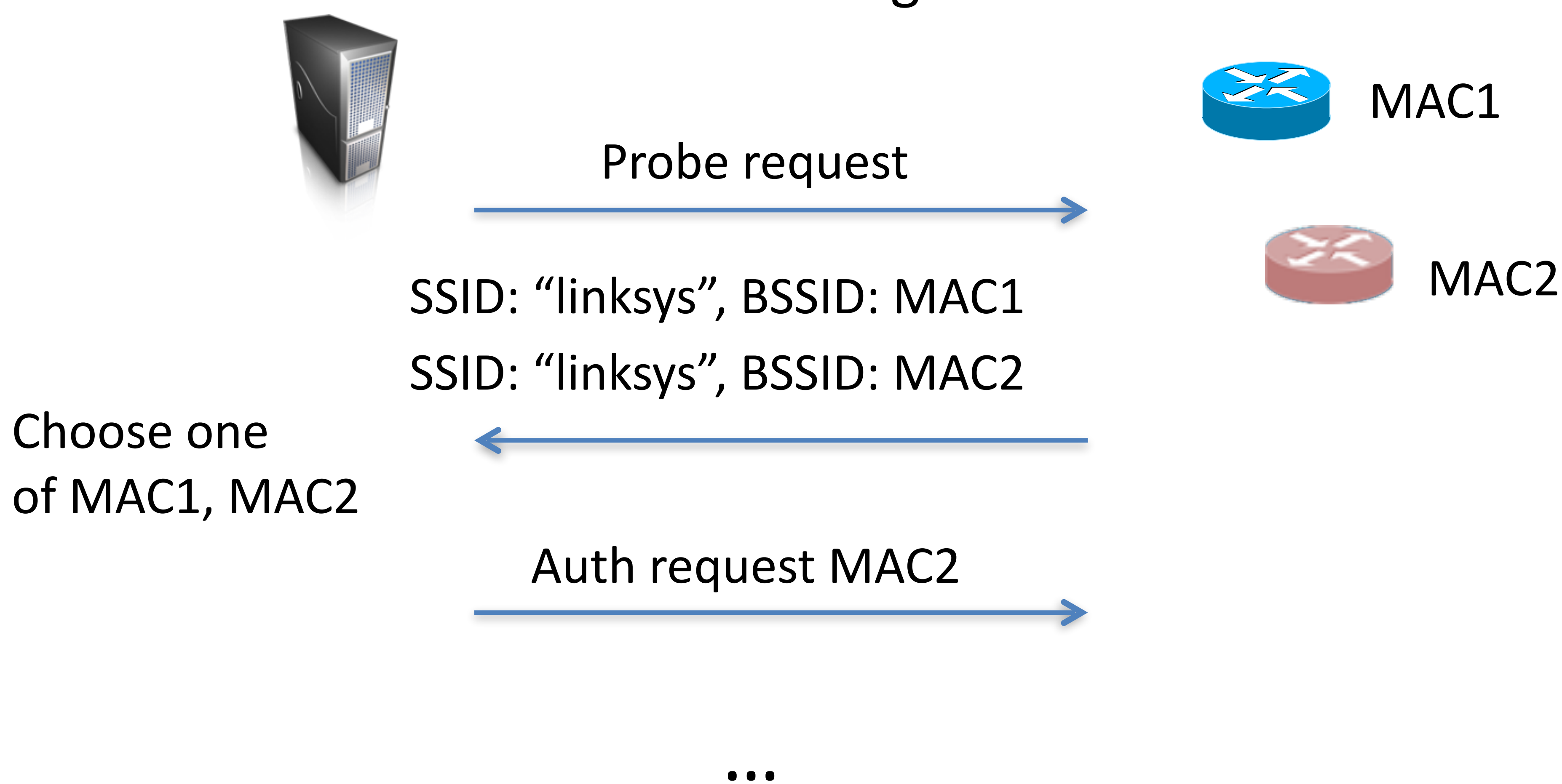
802.11 evil twins



Basic idea:

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

Basic attack: rogue AP



* Password based key derivation protocol (PBKDF)
  / Dictionary attacks
  / bcrypt, scrypt

* Network Security
  / Ethernet sniffing
  / ARP cache poisoning, MitM, DoS
  / WiFi Evil Twins

* Exit slips
  / 1 thing you learned
  / 1 thing you didn't understand

recap