(in) Sicurezza in Rete

Parte II

Luigi Vetrano
TechnoLabs S.p.A.

L'Aquila, aprile 2011

Sicurezza: necessità

- Riservatezza:
  - la comunicazione è stata intercettata?
- Autenticazione:
  - l'utente è veramente chi dice di essere?
- Integrità:
  - i dati ricevuti sono proprio quelli spediti?
- Non ripudio:
  - il mio interlocutore può ritrattare quello che ha detto?
- Disponibilità:
  - il mezzo di comunicazione è stato reso inutilizzabile?
- Autorizzazione:
  - ogni utente può accedere solo alle risorse cui ha diritto

Obiettivi

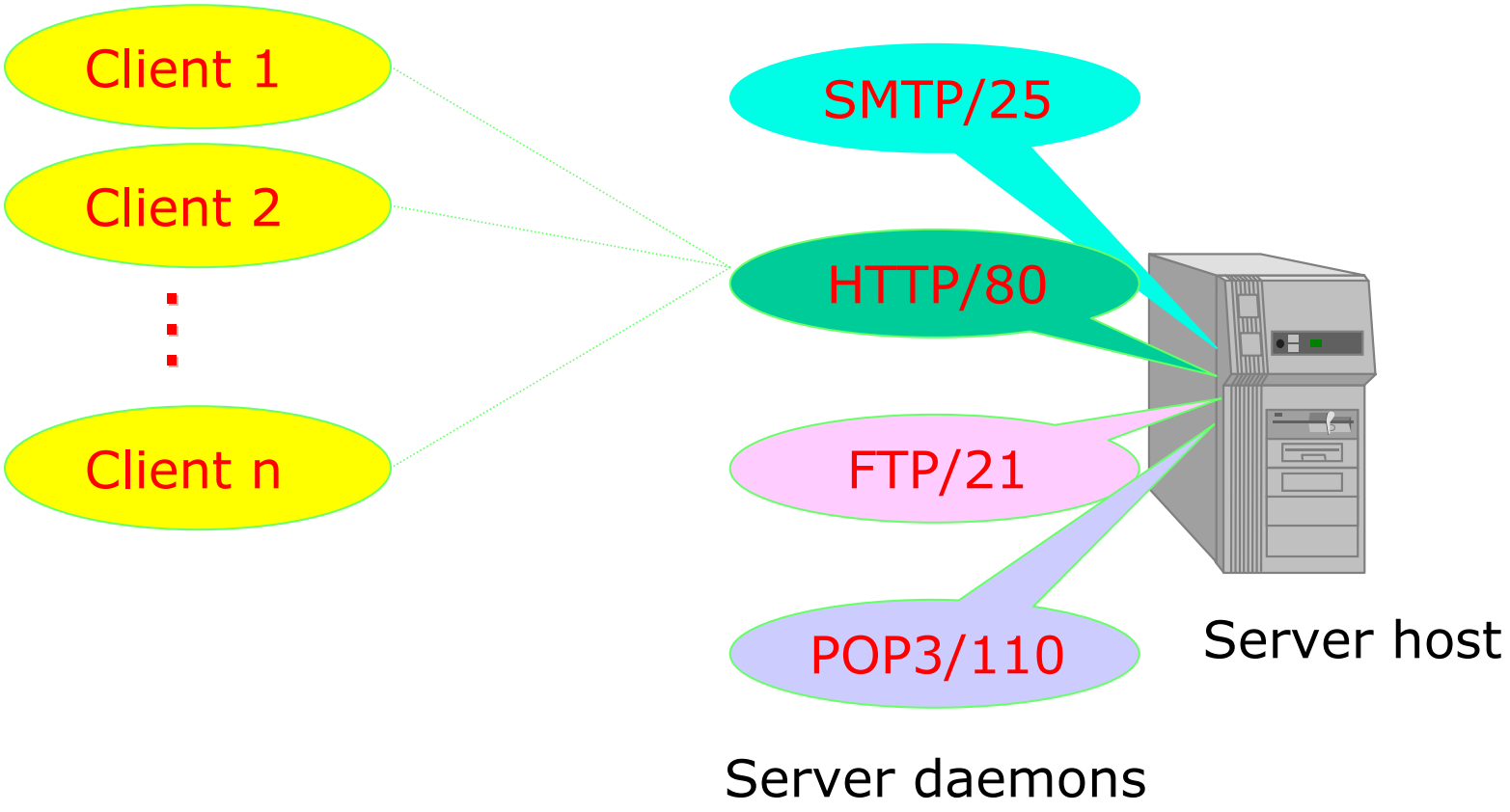
- Reti corporative:
  - funzionalità ed
  - efficienza, ma
  - non sicurezza
- Concentrazione degli attacchi
  - Istituti finanziari e banche - frodi
  - Service Provider - customer database e intercettazioni
  - Pubbliche amministrazioni - sfida, frodi
  - Agenzie governative e della difesa - sfida, spionaggio
  - Aziende farmaceutiche - spionaggio
  - Aziende multinazionali – spionaggio

(fonte: Network Security Solutions ltd, http://www.ns2.co.uk)

Vulnerabilità

- Sono le stesse caratteristiche intrinseche di Internet che rendono possibili la grande maggioranza degli attacchi all'esterno:
  - il modello client-server e
  - la tecnologia di trasmissione del tipo "commutazione di pacchetto".

Modello generale



Sicurezza dei server

- I server ricoprono un ruolo primario nella sicurezza delle reti. Una volta ottenuto un accesso non autorizzato a un server, il resto della rete è facilmente attaccabile.

Servizi di una rete

- Servizi generalmente attivi in una network classica
  - Web aziendale (in casa o in hosting dal proprio provider)
  - E-Mail per le comunicazioni globali
  - Rete locale per l'accesso ad Internet
  - Accesso remoto
  - DNS

Tipologia degli attacchi

- Bisogna distinguere i problemi di insicurezza delle reti in due grandi classi:
  - Debolezze strutturali del protocollo TCP/IP
  - Mancanza di correttezza delle implementazioni dei programmi per la gestione dei servizi
- Nella seconda classe si concentrano la maggioranza degli attacchi

Profilo delle strategie tipiche

- Azioni di scan della rete
  - Per individuare reti o porzioni di reti che possono essere vulnerabili agli attacchi
  - Per individuare i singoli host e le loro caratteristiche:
    - **sistema operativo,**
    - **server running daemon,**
    - **porte TCP aperte**
- Azione intrusiva
  - acquisizione dei diritti di super-utente (root)
  - Installazione di una backdoor
  - Patch al sistema operativo per nascondere le successive intrusioni

Profilo delle strategie tipiche -2

- Azione protettiva
  - patch al sistema operativo per rendere inaccessibile il sistema ad altri hacker. (gli hacker sono i maggiori esperti di sicurezza!)
  - installazione di backdoor
- Azione intercettiva
  - in base all'obiettivo:
    - **Sniffer**
    - **Password cracker**
    - **Back Orifice**
    - **……**

Sicurezza: principi e strategie

- Diversità di difese:
  - non solo più di un meccanismo di sicurezza, ma possibilmente di tipi diversi.
- Semplicità:
  - sistemi semplici sono più facili da capire e mantenere;
  - programmi complessi sono *sicuramente* afflitti da bug, alcuni dei quali possono avere implicazioni di sicurezza.

Sicurezza: principi e strategie

- Sicurezza via mancanza di informazioni (*Security Through Obscurity*)?
  - Sbagliato
    - **una macchina collegata ad Internet con l'unica protezione che nessuno ne è a conoscenza;**
    - **un server in modo che ascolta su di una porta diversa da quella di default;**
  - Corretto
    - **non divulgare i dettagli del proprio NID system;**
    - **informazioni dettagliate sul proprio firewall (protocolli ammessi, modello, ecc. ecc.)**

Tipologia delle vulnerabilità

- Difetti di progettazione nel software o nei protocolli
  - NFS
    - **manca l'autentificazione del cliente**
  - TCP
    - **IP source routing**
    - **Previsione numero di sequenza TCP**
- Difetti nella realizzazione del software o dei protocolli
  - mancanza o errati controlli sul contenuto e la dimensione dei dati (*buffer overflow*);
  - mancanza o errati controlli sul risultato delle operazioni;
  - mancanza o errata gestione esaurimento risorse.
- Errori nella configurazione del sistema e della rete
  - FTP anonimo

Protezioni

- Host
  - strumenti di intrusion detection e controllo
    - **COPS, Tiger, tripwire, swatch**
  - "personal" firewall
    - **ipf, ipchains**
  - port / vulnerability scanner
    - **nmap, nessus**
- Reti locali
  - firewall
  - network intrusion detection
- Protocolli di connessione
  - IPSec
  - SSH
  - SSL/TLS
  - PGP, S/MIME

Network Traffic Sniffing

Sniffer : Qualunque device, software o hardware, capace di ascoltrare (intercettare) i pacchetti che transitano in rete.

Promiscuous Mode : Modalità in cui un computer può intercettare tutto il traffico sul segmento la rete cui appartiene, indipendentemente dalla destinazione.

> Normalmente, un computer intercetta solo il traffico che è destinato a lui o ad un indirizzo di broadcast.

`tcpdump` : sniffer facile da utilizzare con una grande quantità di filtri

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

Sniffer Format ed esempio di Filtro

tcpdump generalmente produce una linea di output per ogni pacchetto che intercetta:

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

time — source IP — direzione — dest IP — protocollo — data bytes — type of service

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

Il byte numero 9 del pacchetto IP (si comincia a contare da 0) trasporta un valore binario uguale a 88 in decimale

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

ip-proto-88 = Cisco EIGRP routing protocol

Filtri su Hosts, Nets, e 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

- indirizzi possono essere scritti sia come hostnames o A.B.C.D
- porte possono essere scritte sia come service names che numeri
- se si specifica un range bisogna usare il numero di bytes

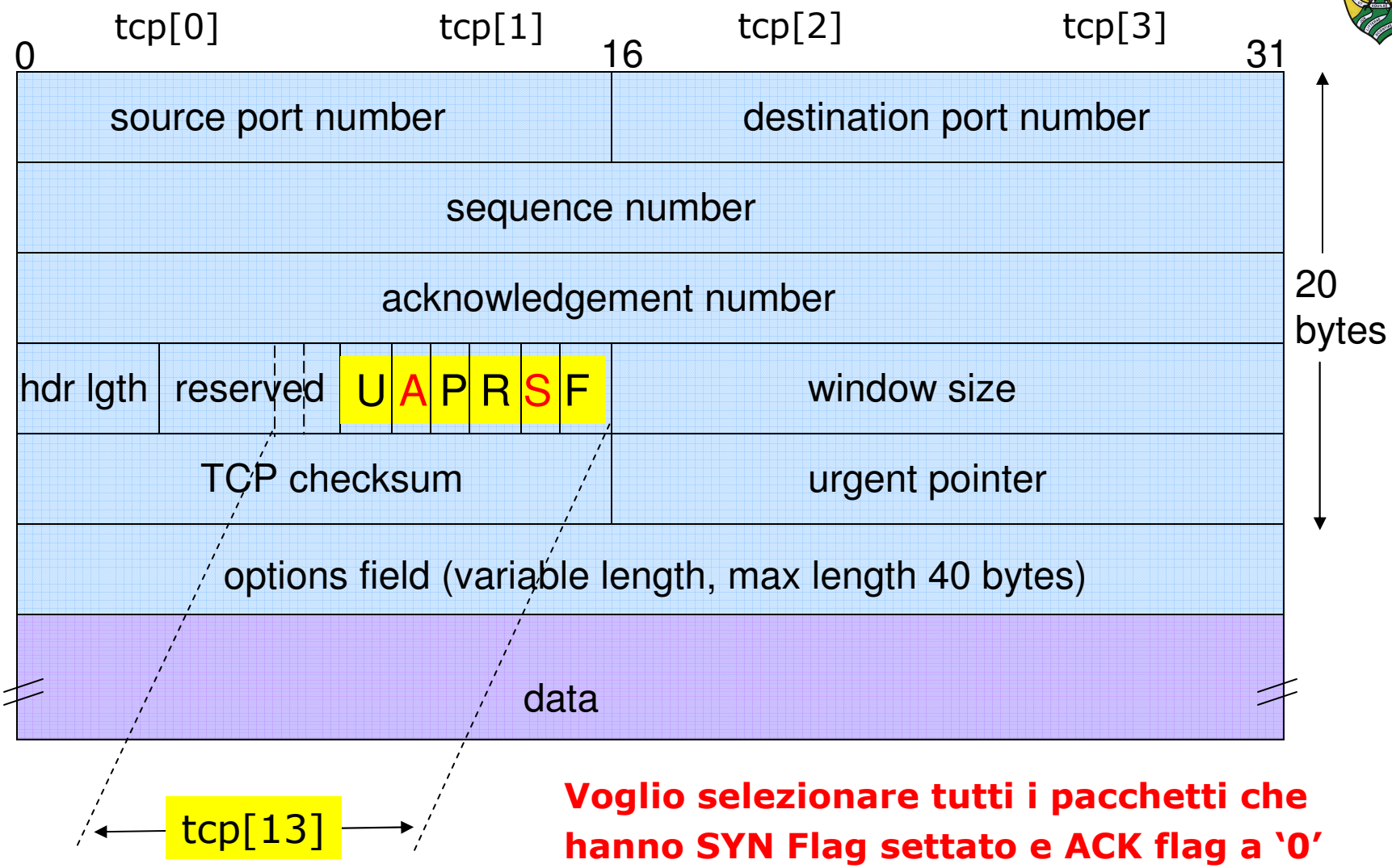
IP	
host is either src or dst:	`host spoofed.target.net`
dst network is 172.31.x.x	`dst net 172.31`
dst network is 172.16 - 172.31	`'dst net 172 and (ip[17]>15) and (ip[17]<32)'`
TCP/UPD	
src port equals 7:	`src port 7`
dst port equals 19:	`dst port chargen`
src ports less than 20:	`udp[0:2] < 20`
dst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

Filtri sofisticati

```
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) )
 )
))
```

Acme > tcpdump -F *filter_filename*

Formato di un pacchetto TCP

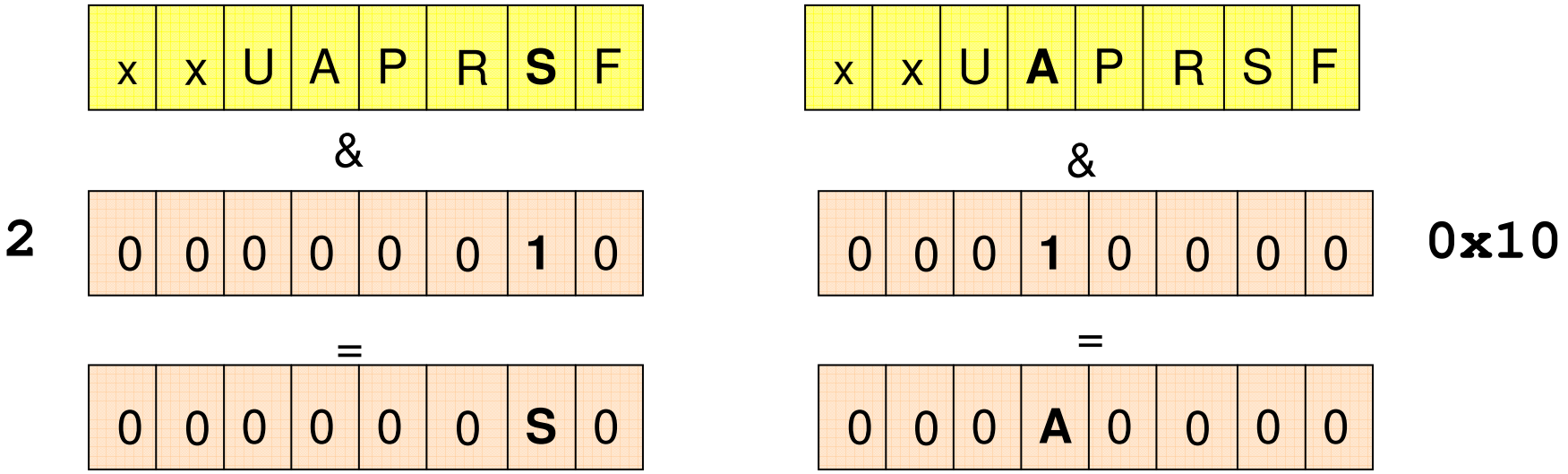


Voglio selezionare tutti i pacchetti che hanno SYN Flag settato e ACK flag a '0'

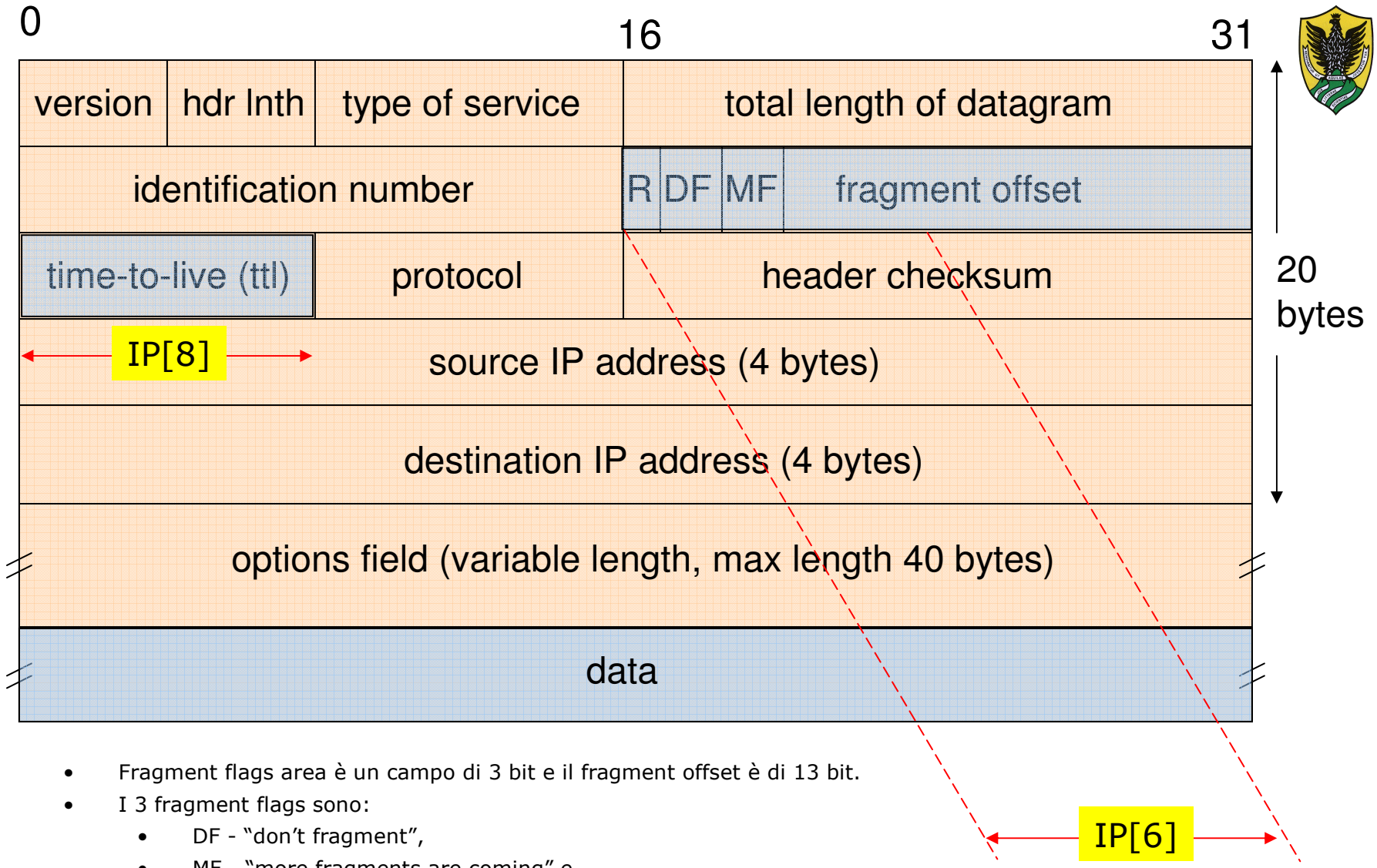
TCP Flag Bit-Masking

SYN flag is set:	`tcp[13] & 2 != 0`
ACK flag is not set:	`tcp[13] & 0x10 = 0`

Mask values may be given as decimal or hexadecimal numbers

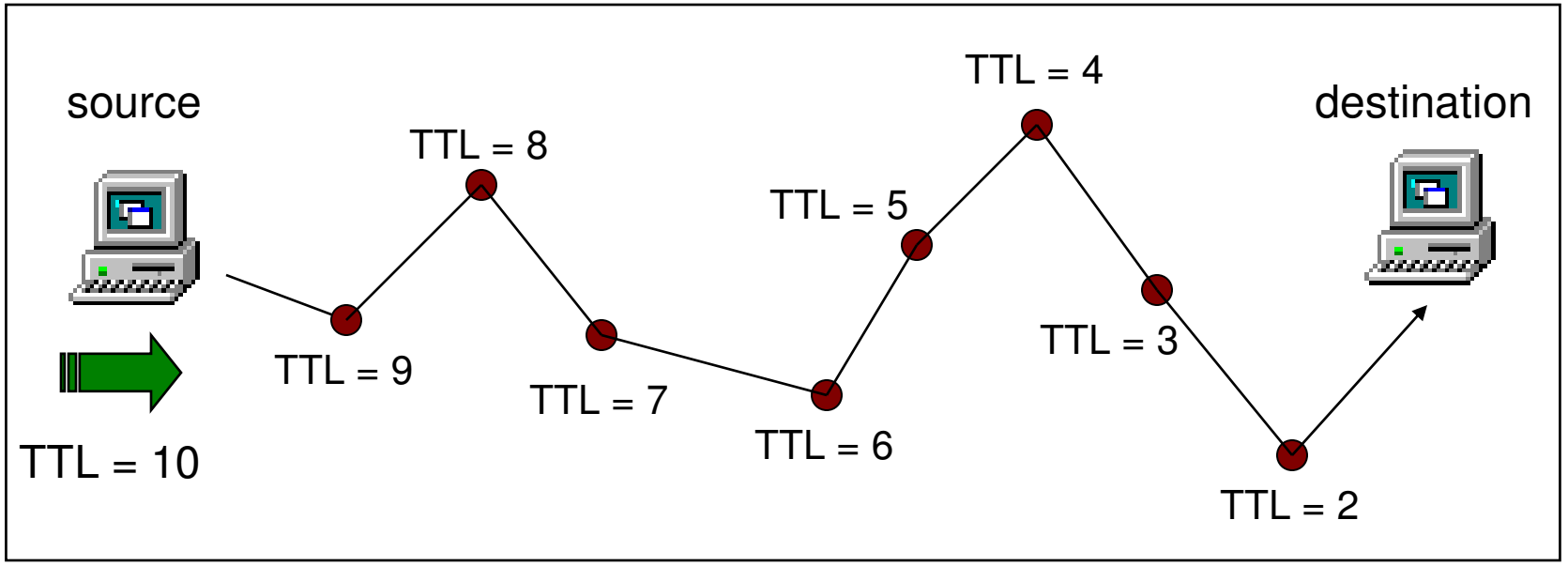


Formato di un datagramma IP



- Fragment flags area è un campo di 3 bit e il fragment offset è di 13 bit.
- I 3 fragment flags sono:
  - DF - "don't fragment",
  - MF - "more fragments are coming" e
  - R - "reserved"

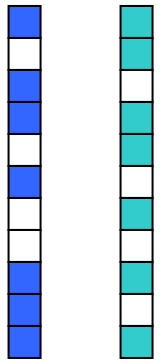
Time To Live (TTL)



- Ogni pacchetto lascia la sorgente con un valore positivo di time to live (TTL)
- Ogni router o gateway lungo il cammino del pacchetto decrementa il TTL prima di instradare il pacchetto
- Il Router che decrementa il TTL di un pacchetto a zero, scarta il pacchetto e invia una segnalazione di errore (ICMP)

Frammentazione e Riassemblaggio di un Datagram

- Un datagram non è riassemblato fino a che non raggiunge la sua destinazione finale (ogni frammento è 'routed' indipendentemente)
- Tutti i frammenti che devono essere riassemblati trasportano lo stesso 'identification number'.
- Il 'fragment offset' ci dice dove piazzare il frammento nella sequenza durante l'operazione di datagram reassembly
- Il flag "more fragments" è settato in tutti i frammenti eccettuato l'ultimo.



receiving computer's fragment reassembly buffer

Esempio: NFS usa 8K o 16K byte per la scrittura mentre Ethernet ha una maximum frame length di soli 1500 bytes (Ethernet's *maximum transmission unit* o MTU).

Fragmented Packets and Byte Offsets

```
12:43:58.43 frag.com > 172.21.164.105:
                    icmp: echo request (frag 4321:1480@0+)
12:43:58.43 frag.com > 172.21.164.105: (frag 4321:1480@1480+)
12:43:58.43 frag.com > 172.21.164.105: (frag 4321:1480@2960+)
12:43:58.43 frag.com > 172.21.164.105: (frag 4321:1480@4440+)
12:43:58.43 frag.com > 172.21.164.105: (frag 4321:1480@5920+)
12:43:58.43 frag.com > 172.21.164.105: (frag 4321:1480@7400+)
12:43:58.43 frag.com > 172.21.164.105: (frag 4321:1480@8880+)
12:43:58.43 frag.com > 172.21.164.105: (frag 4321:1480@10360+)
12:43:58.43 frag.com > 172.21.164.105: (frag 4321:800@11840)
```

Total reassembled packet size is 11840 + 800 = 12640 bytes.

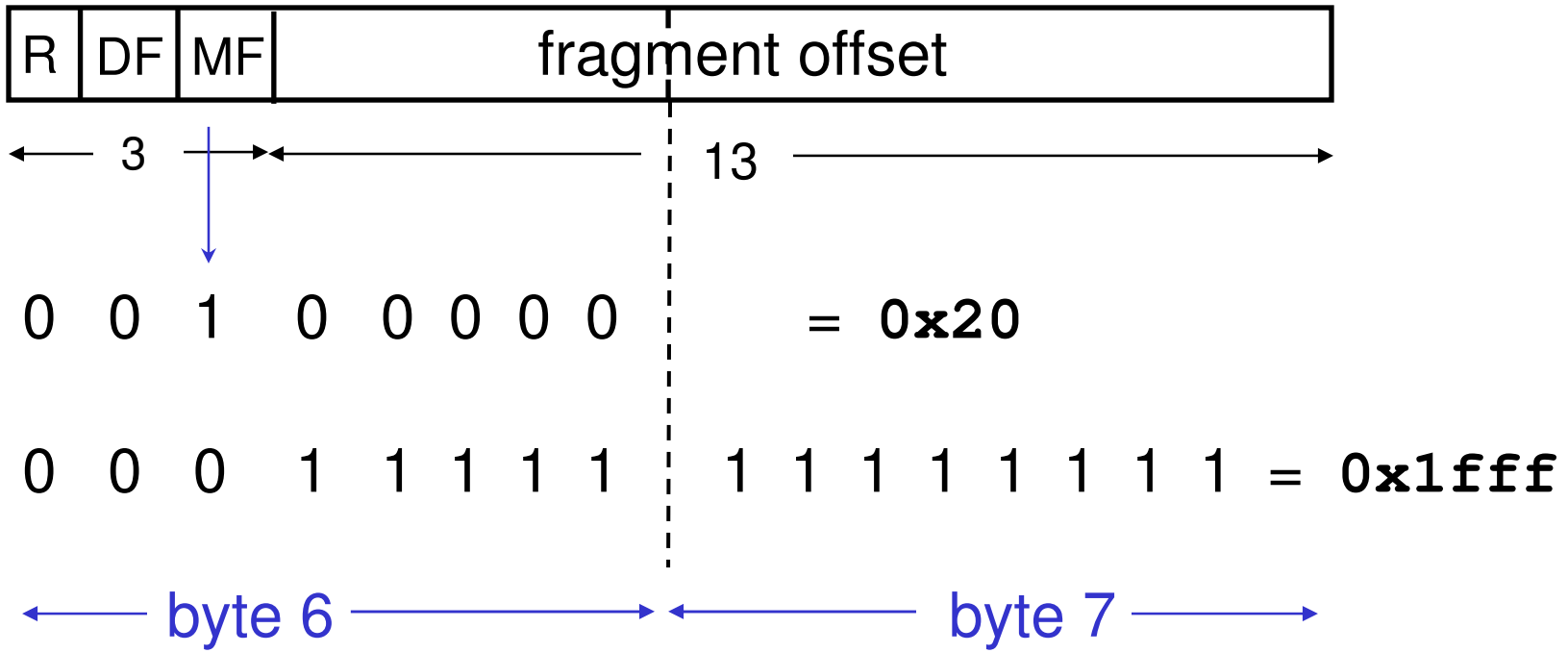
```
fragmented ip packets with more fragments coming
            ip[6:1] & 0x20 != 0

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

Esempio di Bit Masking

```
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
```



Firme (Attack Signatures)

- Ogni attacco può essere identificato da una "signature": proprietà di un pacchetto o di una sequenza di pacchetti considerati come aggregati
- Una signature dovrebbe essere ideata in modo da:
  - Individuare sempre l'attacco
  - Raramente (idealmente mai) dare falsi allarmi
  - Essere di complessità minima
  - Intercettare sottili variazioni dell'attacco
- Questi concetti possono essere estesi alla scrittura di filtri per IDS e firewalls

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 (si parla di 'spoofed source')

Specific Signature

source port = destination port

TCP packet with SYN flag set

port open on target host

Risultato dell' attacco

L'host bersaglio si blocca (DoS)

L'idea di base è inviare un pacchetto ad un host, dove si è modificato il pacchetto in modo da far sembrare che questo sia stato originato dallo stesso ricevente.

Alla ricezione di questo strano pacchetto, un host vulnerabile va in tilt, negando quindi l'accesso agli utenti leciti (DoS: Denyal of Service).

Macchine Windows sono i tipici bersagli di questo tipo di attacco.

Land Attack Signature

`tcpdump` filters

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

Questo filtro seleziona i pacchetti IP dove source e destination *addresses* sono eguali

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

Questo filtro seleziona i pacchetti IP dove source e destination *networks* sono eguali

Volendo monitorare che anche le porte TCP siano uguali, il ns. Filtro diventa:

tcp and (ip[12:4] = ip[16:4]) and (tcp[0:2] = tcp[2:2])

Questo è un tipico esempio di utilizzo di una 'signature' per individuare un attacco

Indirizzi IP Speciali

Indirizzi riservati per reti non connesse ad Internet
(RFC 1918 : *Address Allocation for Private Internets*):

```
10.0.0.0/8
172.16.0.0 - 172.31.255.255
192.168.0.0/16
```

Altri indirizzi riservati dallo IANA:

```
0.0.0.0/8                    58.0.0.0 - 60.255.255.255
1.0.0.0/8                    67.0.0.0 - 126.255.255.255
2.0.0.0/8                    127.0.0.0/8
5.0.0.0/8                    128.0.0.0/16
23.0.0.0/8                   169.254.0.0/16
27.0.0.0/8                   191.255.0.0/16
31.0.0.0/8                   192.0.2.0/24
37.0.0.0/8                   197.0.0.0/8
39.0.0.0/8                   201.0.0.0/8
41.0.0.0/8                   223.255.255.0/24
42.0.0.0/8                   240.0.0.0 - 255.255.255.255
```

Consultare il "Manning Draft" :
http://www.ietf.org/internet-drafts/draft-manning-dsua-03.txt

Indirizzi IP Speciali (cont.)

Questi IP riservati possono apparire nella vs. rete ma solo in circostanze speciali (es: 0.0.0.0 non deve mai apparire come **destination** address e un broadcast mai come **source** address in un pacchetto.

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

ICMP : Internet Control Message Protocol

- Protocollo di servizio per IP
  - diagnostica Internet
  - usato da host e router per segnalare condizioni di errore
- Usa datagrammi IP
  - il payload contiene sempre l'header IP e i primi 8 byte del datagramma che ha provocato il messaggio.
  - vari tipi di messaggio (es. Echo Request/Reply, Host Unreachable, Need to Frag, Time Exceeded, etc..)

Il protocollo ICMP

Ci sono differenti "tipi" di messaggi ICMP, identificati da un ***ICMP message number***

In teoria ci possono essere fino a 256 messaggi ICMP diversi ma quelli evidenziati sono quelli che ci interessano ai fini della sicurezza:

msg#	description
0	***echo reply***
3	***destination unreachable***
4	source quench
5	redirect
8	***echo request***
9	*router advertisement*
10	router solicitation
11	***time exceeded***

msg#	description
12	*parameter problem*
13	*timestamp request*
14	timestamp reply
15	information request
16	information reply
17	address mask request
18	address mask reply

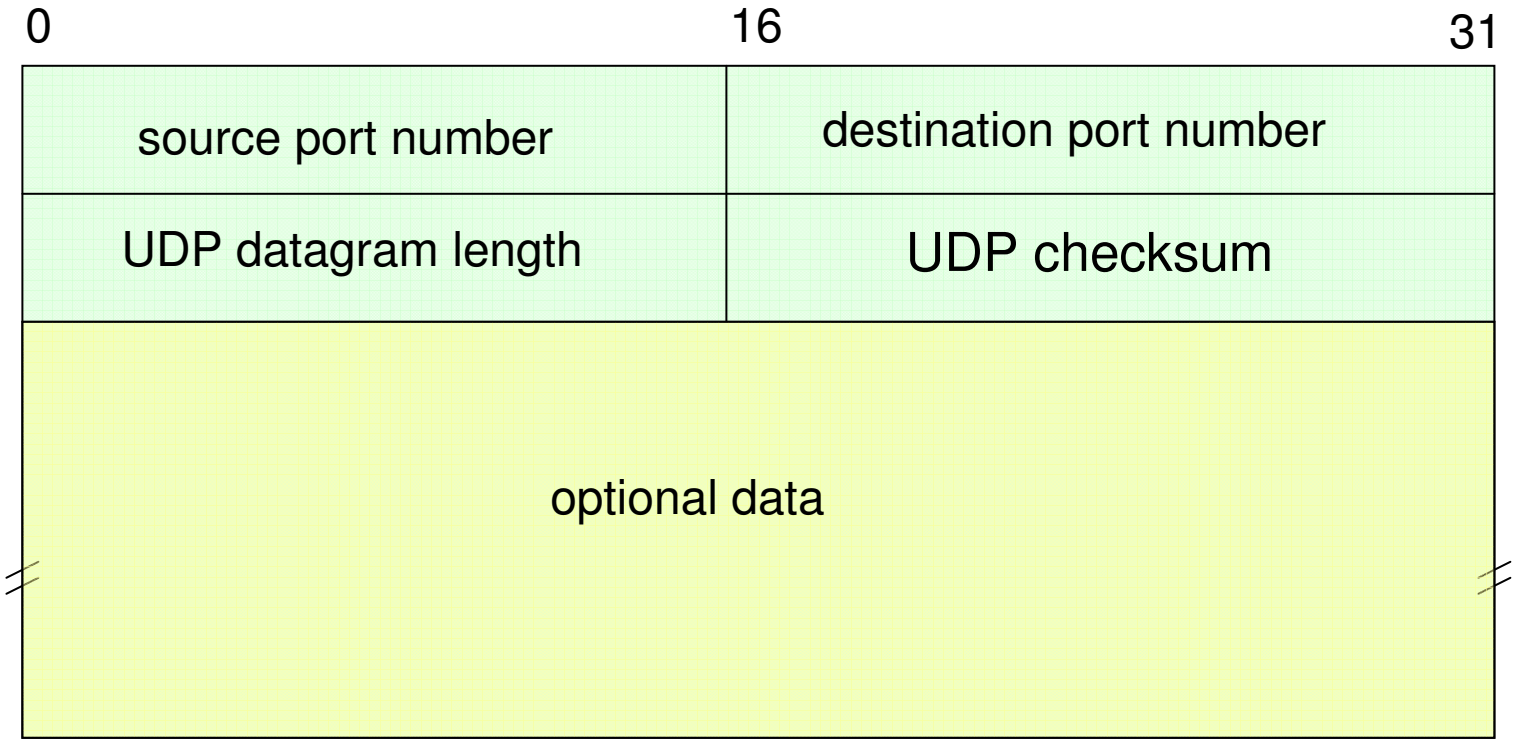
Protocolli di trasporto

- Concetti per protocolli al livello di trasporto:
  - **Porta**
    - **astrazione usata dai protocolli di trasporto per distinguere fra più processi sullo stesso host**
    - **intero a 16 bit (0-65535)**
    - **per servizi standard identificativo assegnato dalla IANA (porte da 0 a 1023 ➔ well-known-ports)**
  - **Socket**
    - **coppia (indirizzo IP, porta)**
    - **generalizzazione del meccanismo di accesso a file**

Note: Ports 0 through 1023 are generally reserved for processes running with superuser privileges

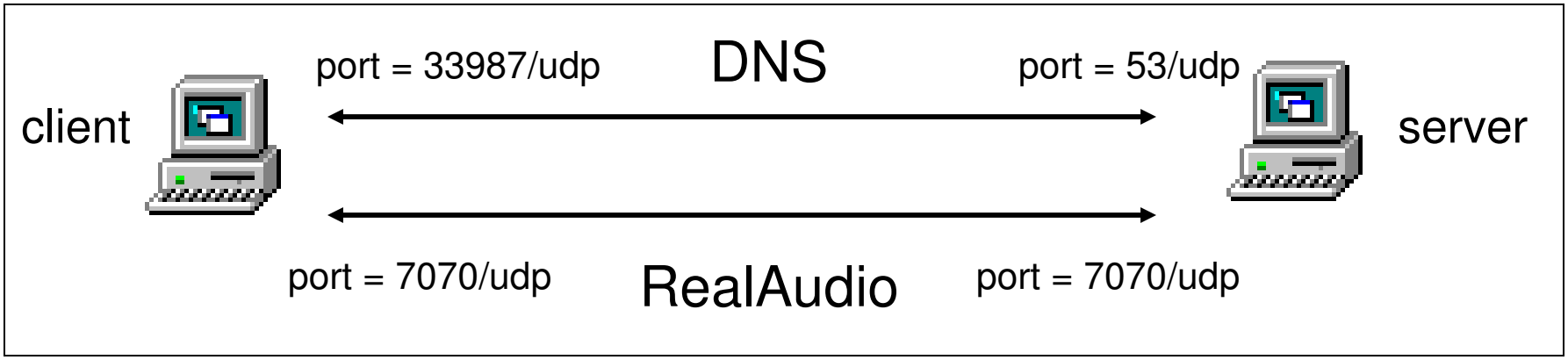
User Datagram Protocol

- Protocollo connectionless (come IP)
  - nessuna garanzia di consegna
  - assenza controllo di flusso
  - assenza di correzione di errore



Il protocollo UDP

- UDP = *no reliability*
  - Non c'è nessuna garanzia che il datagramma raggiunga la destinazione desiderata nè che sia consegnato integro nè nella giusta sequenza temporale
- ***Port numbers*** = identificano i processi Tx e Rx



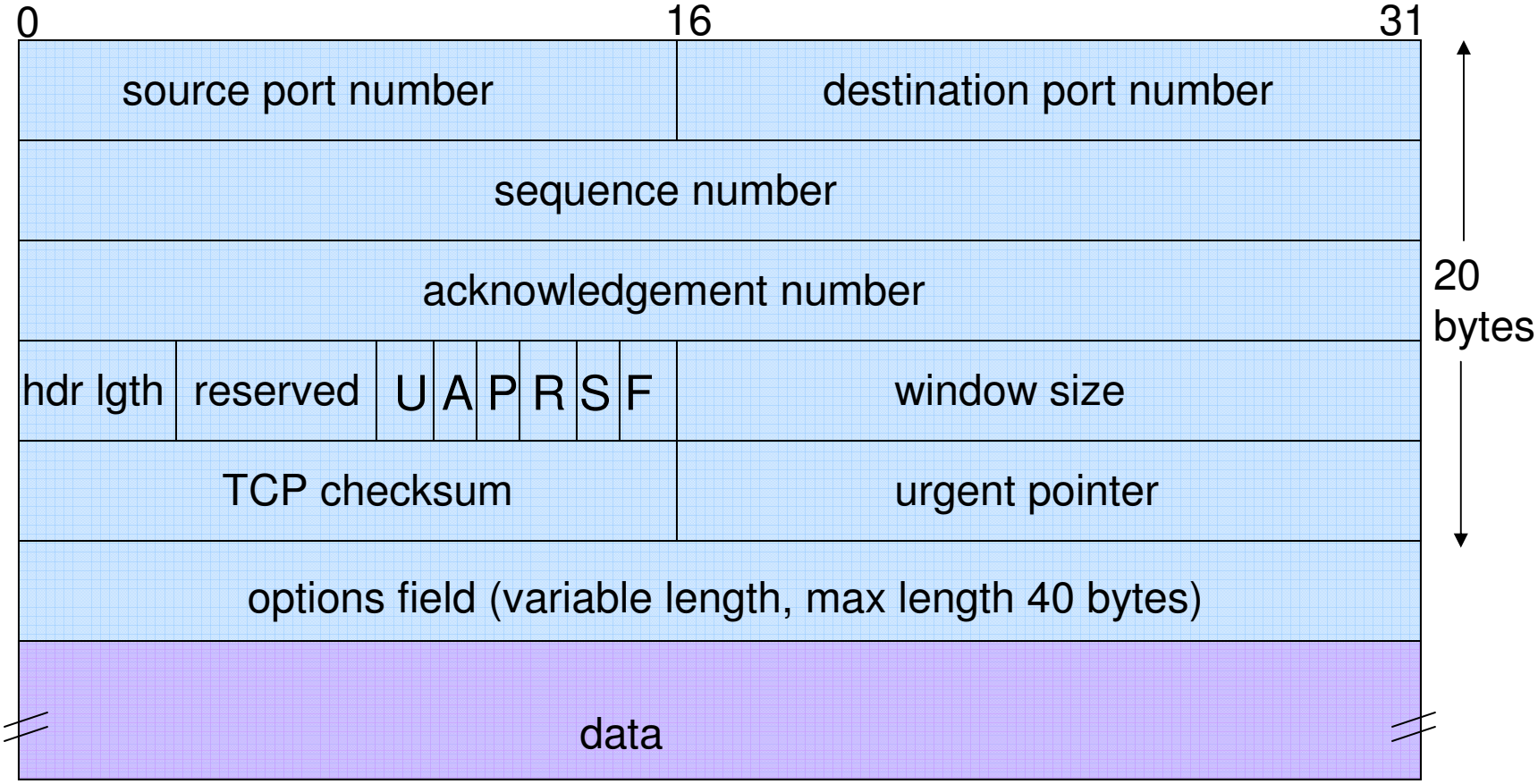
Some UDP streams use the same port number for source and destination

UDP: problemi

- Assenza di correzione di errore
  - Tutti i controlli sul flusso sono a carico del livello applicativo
- Assenza controllo di flusso
  - Implicazione di sicurezza: IP spoofing
  - Semplice per attaccante sostituirsi ad host
    - **Host A effettua query a host B**
    - **Cattivo (C) risponde ad A ed effettua DoS contro B**
- Consigliabile filtrare servizi non essenziali

Trasmission Control Protocol

- **Protocollo connection-oriented**
  - **controllo del flusso**
  - **affidabilità della consegna**
  - **spoofing più difficile (ma non impossibile!)**
- **Connessione virtuale tra due socket**
  - **connessione in chiaro**



TCP Segment Flags

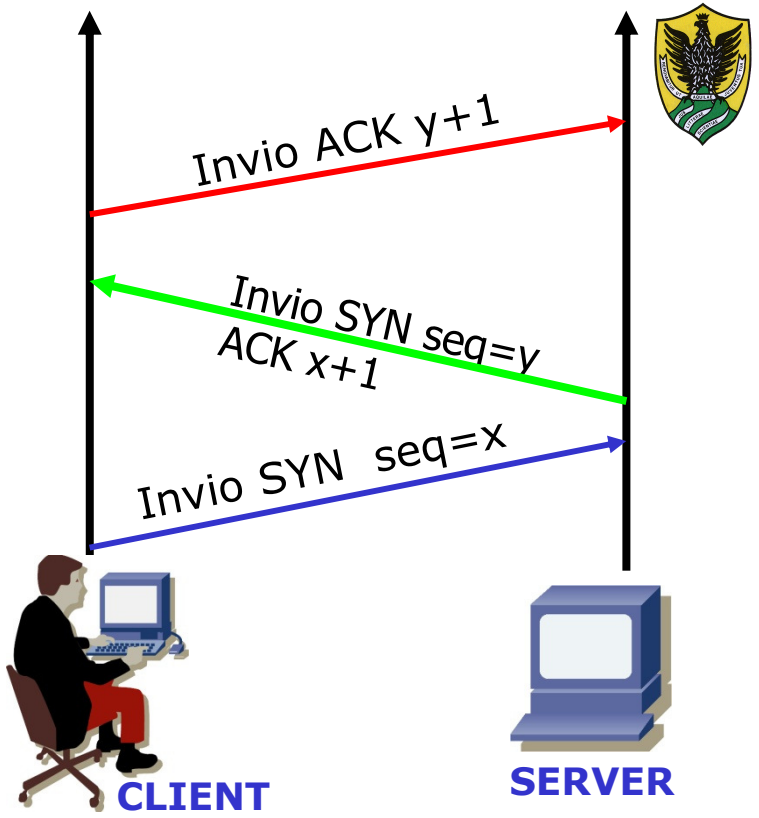
Ci sono 6 flag bits che si possono settare in varie combinazioni nell'header TCP

- **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)

Alcuni attacchi usano combinazioni non-standard dei suddetti flag per crashare o per cercare di identificare il sistema remoto.

TCP: Three-Way Handshake

1. Il client invia un pacchetto TCP al server (linea BLU) con una 'initial TCP sequence number' e il *synchronize* (SYN) flag settato ("active open").
2. Il server risponde con un pacchetto (linea VERDE) che contiene sia il SYN che l' *acknowledgement* (ACK) bit settato ("passive open"). Il pacchetto 'passive open' ha un numero iniziale di sequenza per la comunicazione server-to-client più un ACK per il numero di sequenza iniziale del client.
3. Il client invia indietro un pacchetto di ACKnowledge (linea ROSSA) del numero di sequenza iniziale del server.



A questo punto entrambi i partner sono certi che l'altro li abbia sentiti e la comunicazione può iniziare.

Durante il resto della comunicazione, il bit ACK è sempre on e il flusso dei dati è determinato dall'applicativo che gira sul TCP.

Alla fine della conversazione TCP, uno dei partner inizia la 'closing sequence' inviando un pacchetto FIN.

Il FIN è ACKnowleged dall'altro partner e la connessione si dice "half closed", il che significa che non ci saranno altri dati in quella direzione.Poichè una connessione TCP è full-duplex (i dati fluiscono nelle due direzioni independentemente), ogni canale deve essere shut down separatamente: ovvero c'è bisogno dello scambio di 4 messaggi per terminare una connessione TCP. Il primo FIN inizializza la "active close".

Esempio di Sessione TCP : telnet

client
(port = 33623/tcp)

server
(port = 23/tcp)

SYN

SYN - ACK

ACK

[ACK settato per il resto della sessione]

[continuaz. sessione]

FIN ACK

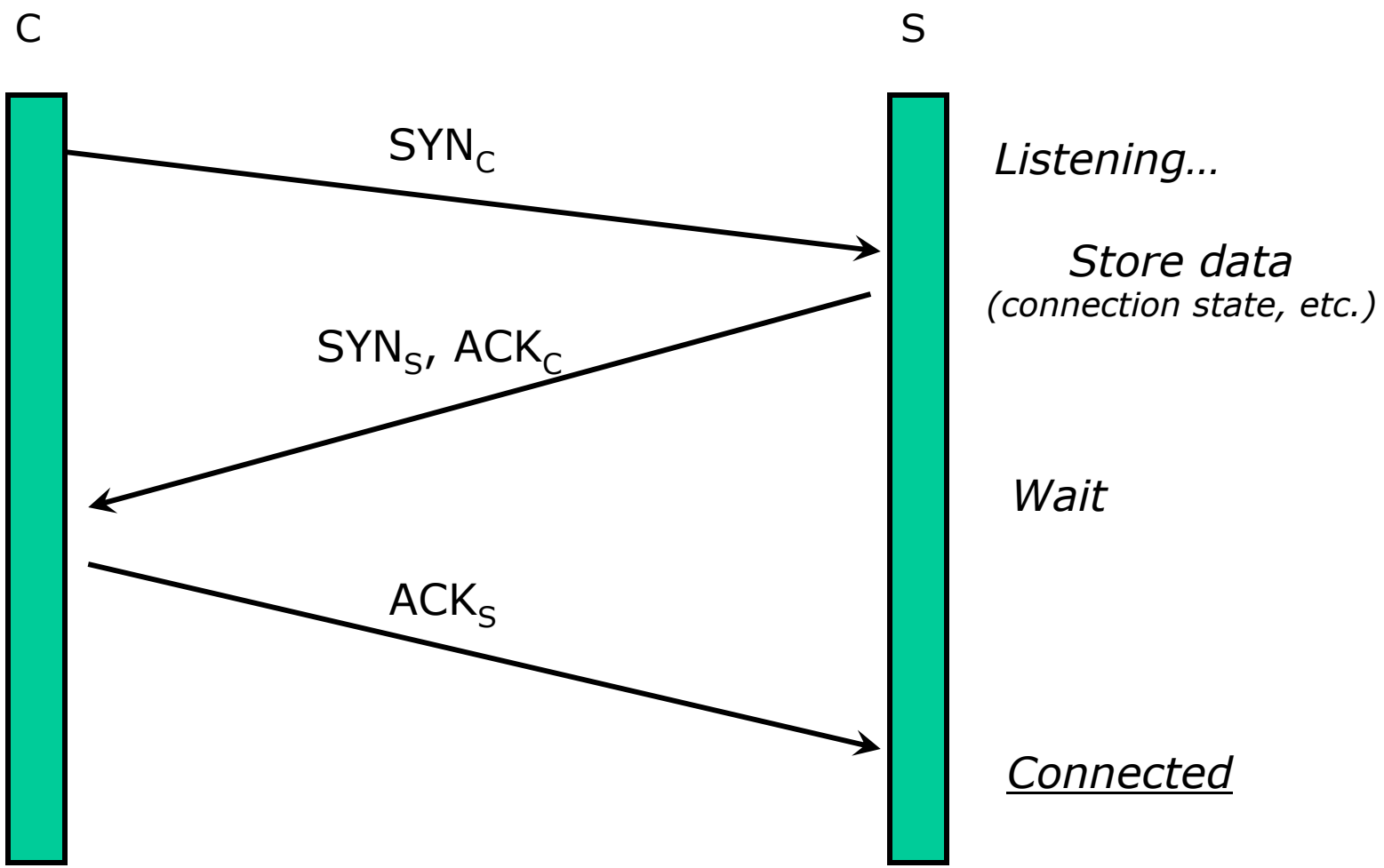
ACK

FIN ACK

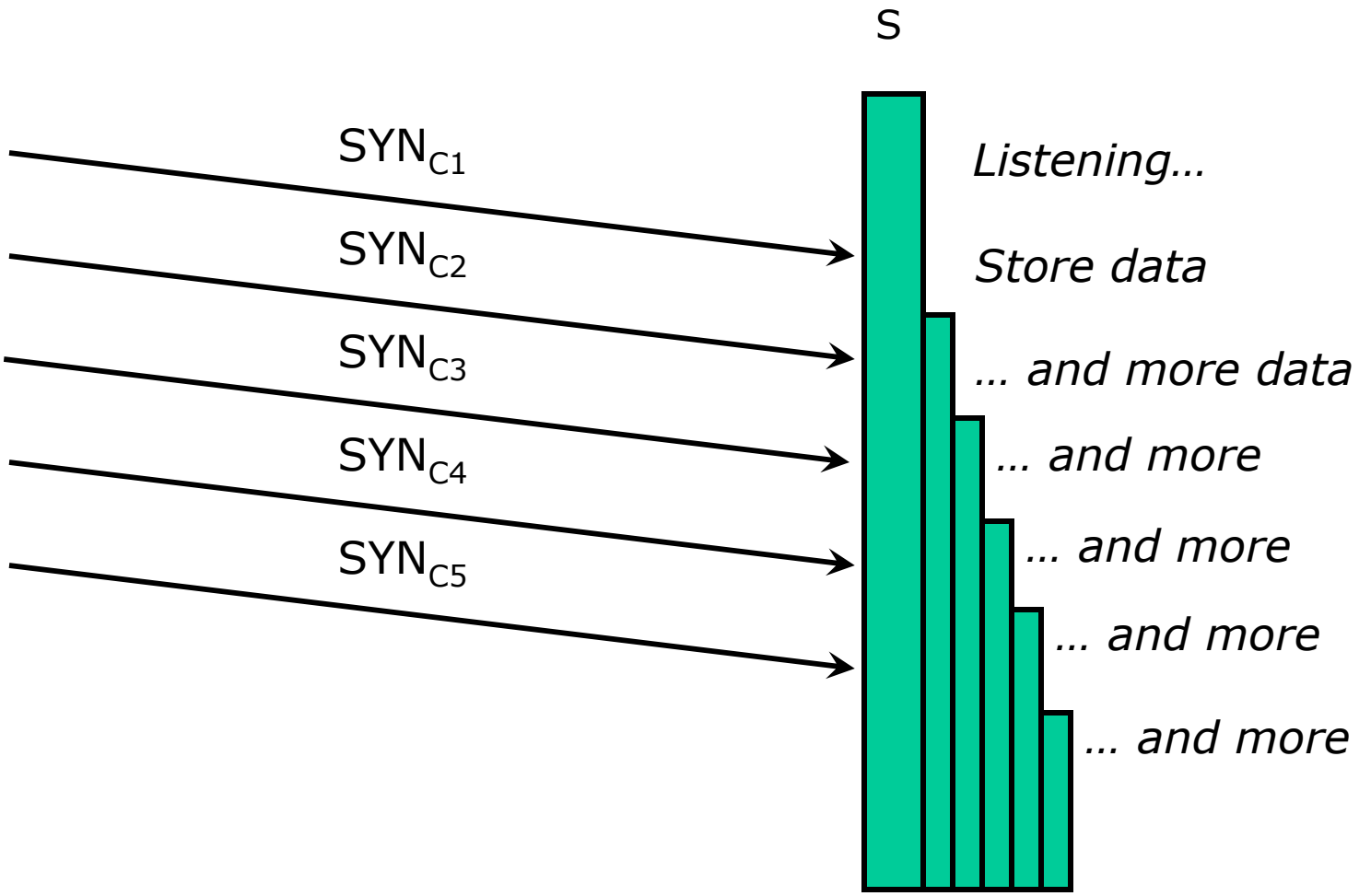
ACK

Sia il client che il server possono avviare la closing sequence

TCP Handshake



SYN Flooding Attack



SYN Flooding Explained

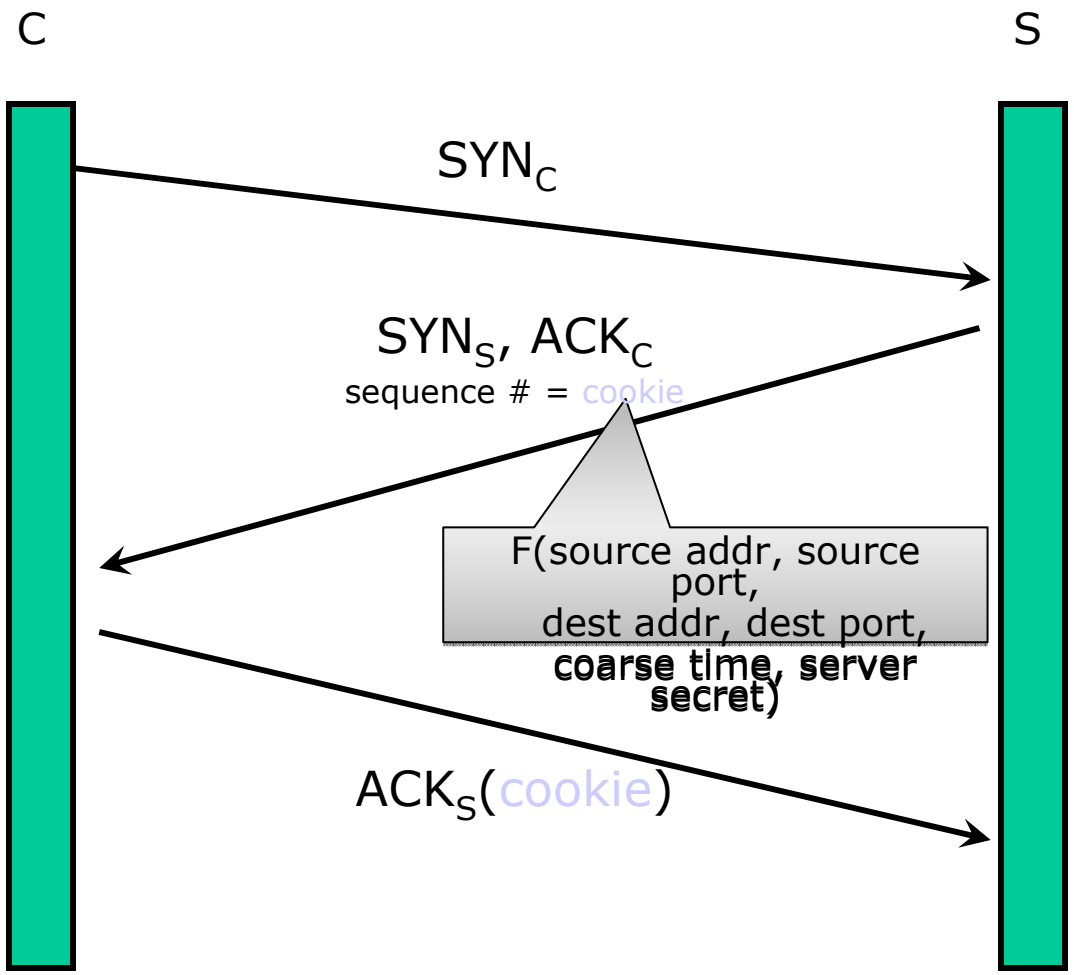
- Attacker sends many connection requests with spoofed source addresses
- Victim allocates resources for each request
  - Connection state maintained until timeout
  - Fixed bound on half-open connections
- Once resources exhausted, requests from legitimate clients are denied
- This is a classic denial of service (DoS) attack
  - Common pattern: it costs nothing to TCP initiator to send a connection request, but TCP responder must allocate state for each request (asymmetry!)

Preventing Denial of Service

- DoS is caused by asymmetric state allocation
  - If responder opens a state for each connection attempt, attacker can initiate thousands of connections from bogus or forged IP addresses
- Cookies ensure that the responder is stateless until initiator produced at least 2 messages
  - Responder's state (IP addresses and ports of the connection) is stored in a cookie and sent to initiator
  - After initiator responds, cookie is regenerated and compared with the cookie returned by the initiator

SYN Cookies

[Bernstein & Schenk]

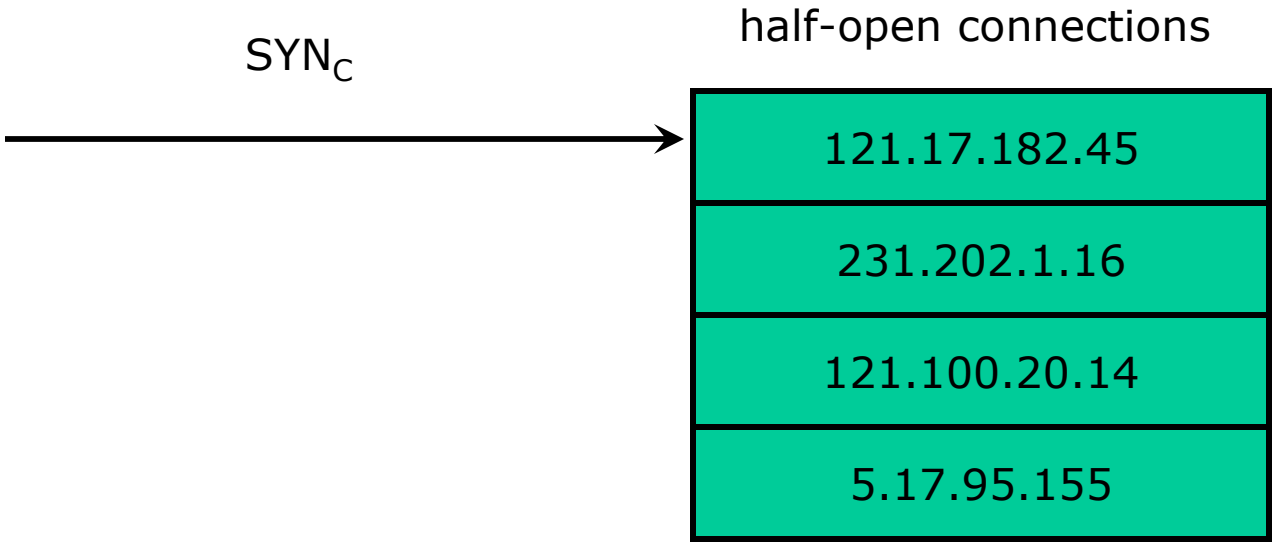


Listening…

Does <u>not</u> store state

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

Another Defense: Random Deletion



- If SYN queue is full, delete random entry
  - Legitimate connections have a chance to complete
  - Fake addresses will be eventually deleted
- Easy to implement

Esempio: TCP Connection Establishment /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)

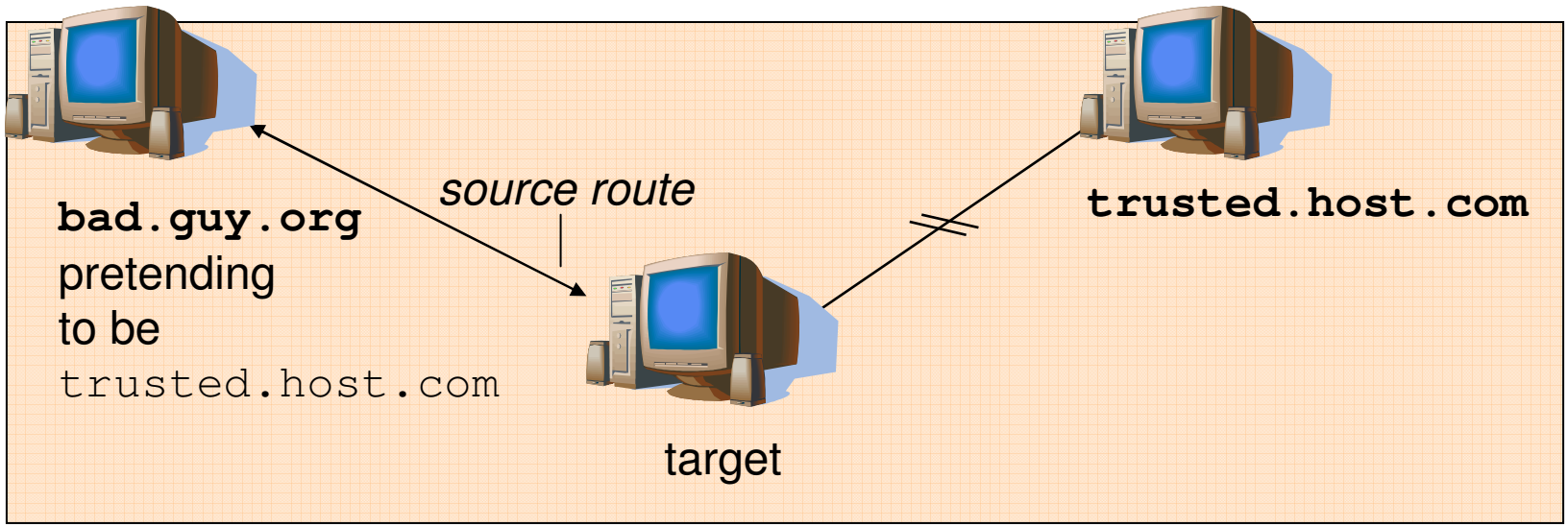
(x) = x data bytes in the packet
win = advertised window size
mss = max segment size announcement
DF = don't fragment flag is set

TCP sequence guessing

- Le connessioni TCP utilizzano un numero di sequenza per riordinare i pacchetti.
- Ad ogni nuova connessione viene utilizzato un numero di sequenza (semi-)casuale.
- Se l'attaccante riesce a predire il numero di sequenza, può generare dei pacchetti con mittente falsificato formalmente corretti (anche se, ovviamente, non riesce a vedere le risposte).
- Perché l'attacco vada a buon fine è necessario che il mittente (vero) non riceva i pacchetti di risposta (o non sia in grado di reagire).
- Come protezione, i router non devono far entrare traffico che risulti proveniente dalla rete interna (*ingress filtering*).

Source-Route Abuse

Una **source route** specifica l'indirizzo IP dei routers che *debbono* gestire il datagramma nel viaggio dalla sorgente alla destinazione

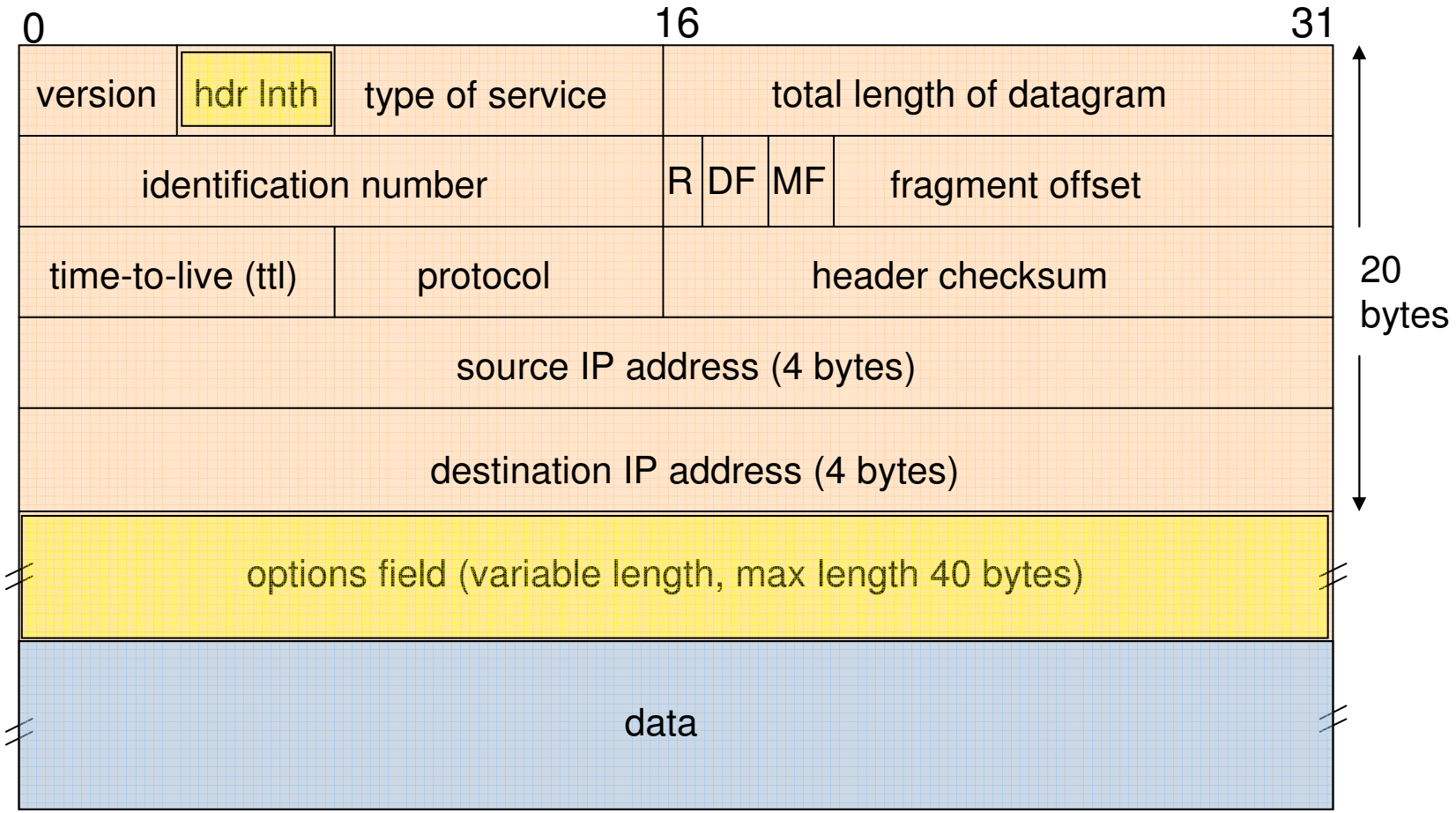


***Pericolo**:* target usa la lista (rovesciata) della source route fornita da **`bad.guy.org`** . Il traffico di ritorno da target, che è indirizzato a **`trusted.host.com`**, viene rigirato a **`bad.guy.org`**

In practice the Source-Route option is very rarely used today.

IP Datagram Format

- Source routing è una opzione di IP
- 'option field' nell'header IP è una lista a lunghezza variabile
- Se non ci sono 'options' la lunghezza della lista è 0



Source-Routed Packets

Signature: cercare quei pacchetti dove

IP Options sono settate

l'unico modo per testare ciò è vedere se IP header size è maggiore di quella no-option size di 20 bytes (5 words da 32-bit)

Se si, l'opzione Loose-Source-Route o la Strict-Source-Route è settata

- **Loose-Source-Route vuol dire che non tutto il cammino è prestabilito ma solo alcuni step (da cui è mandatorio passare)**
- **Strict-Source-Route vuol dire che il cammino è settato completamente**

Loose Source e Record Route

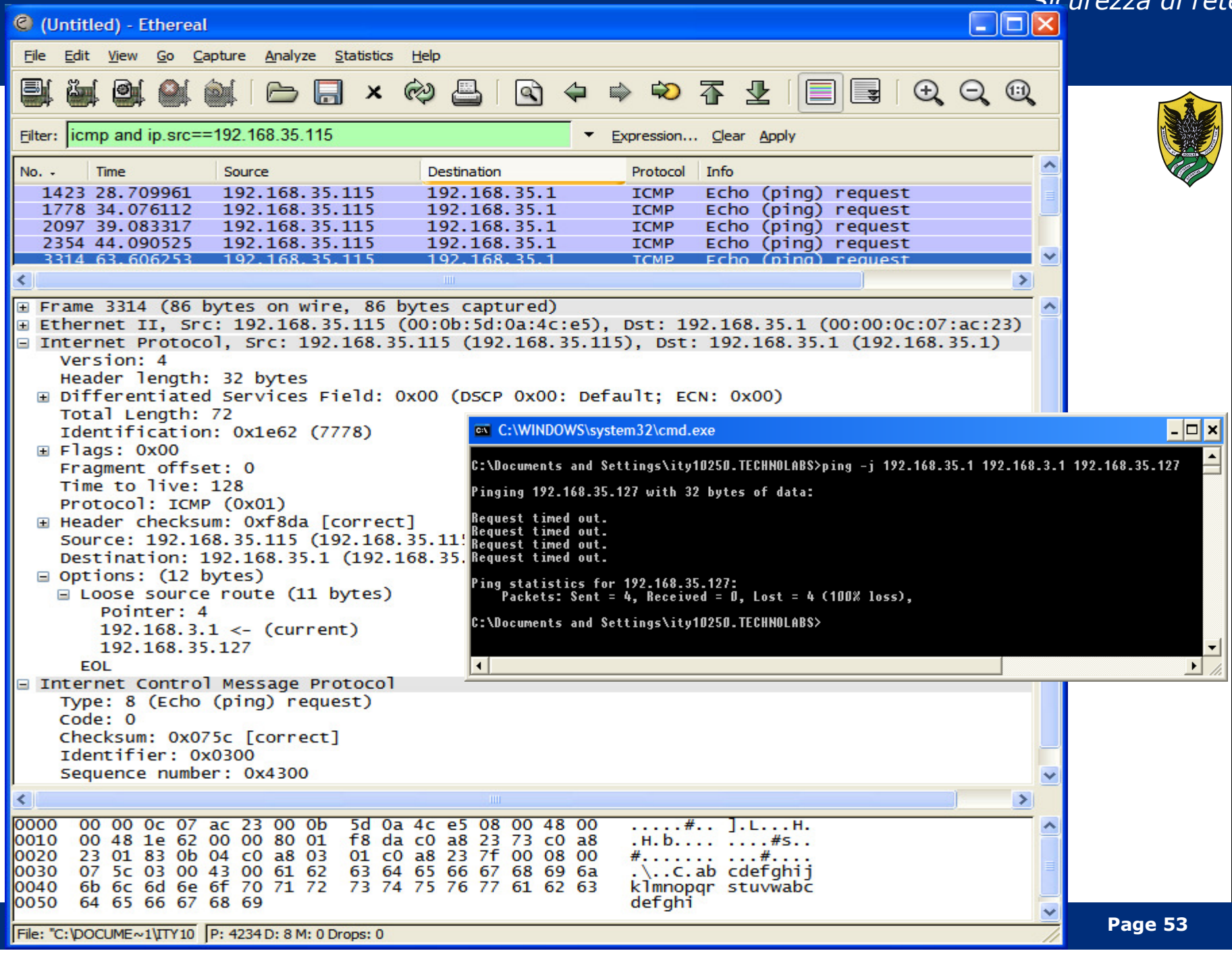
Option field

10000011	length	pointer	route // //	data

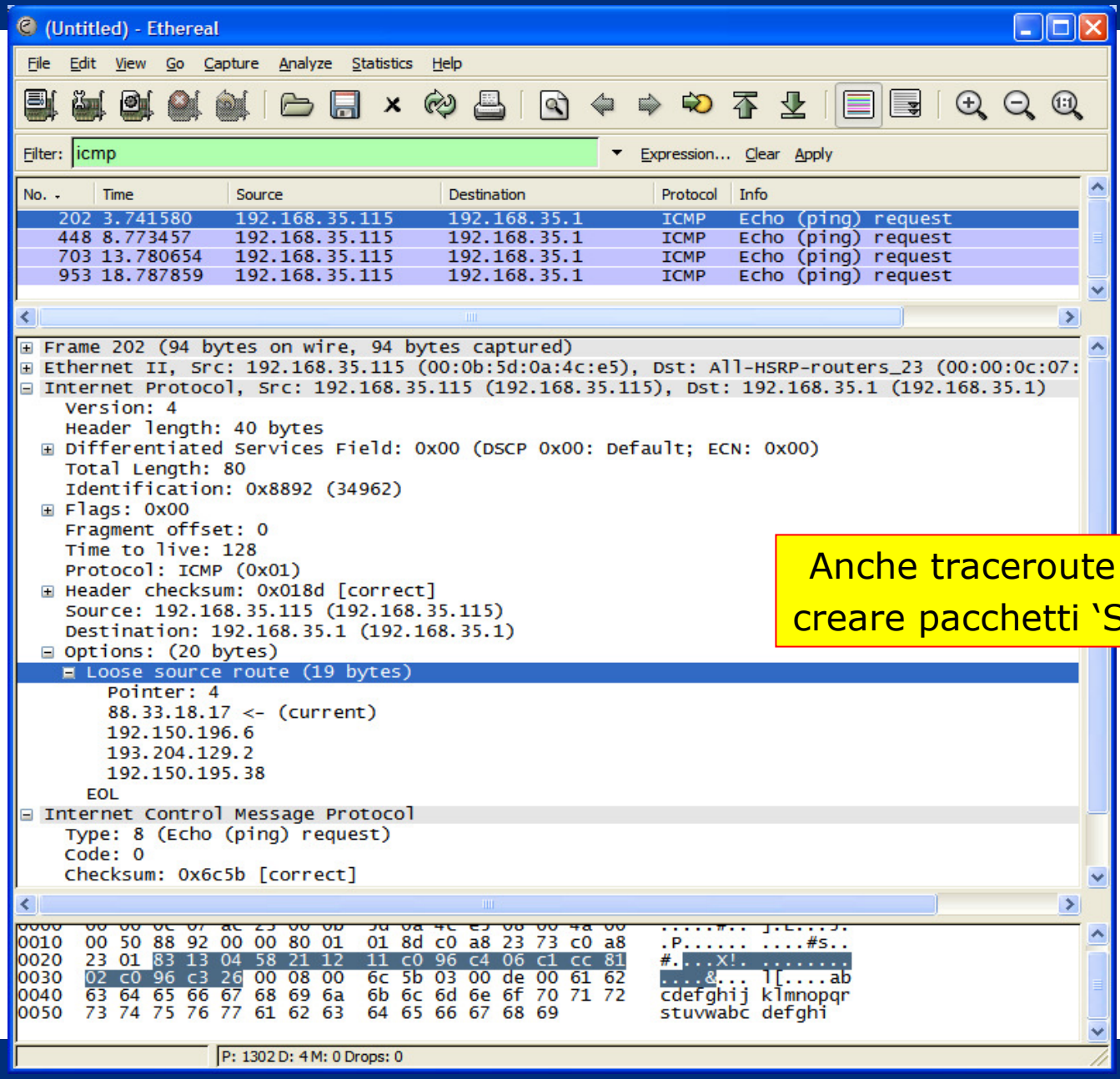
Type=131

Source routed

- Per il source routing, la lista dei routers che devono trattare il pacchetto è trasportata nel campo 'IP option'.
  - IP option field è una lista a lunghezza variabile contenuta nell'header IP.
  - IP header size MAX = 60 bytes
  - Siccome la parte mandatoria consuma 20 di questi bytes, la parte 'options field' è di 40 bytes max: il numero di IP addresses che possono essere trasportati è 9 (perchè i primi 3 bytes di option sono usati: type-length-offset).
  - Questo è uno dei motivi per cui il source routing è poco usato.
- Per determinare se il source routing è stato richiesto da un pacchetto, bisogna vedere il campo *header length*.
  - Se il valore specificato è maggiore di 5 – corrispondente a 5 words da 4-byte – allora l'opzione è stata settata.
- Next step è vedere se il valore del primo byte nel campo "option field" (detto *code* byte) vale 0x83 o 0x89. Questo byte specifica quale particolare tipo di source route è stata richiesta.



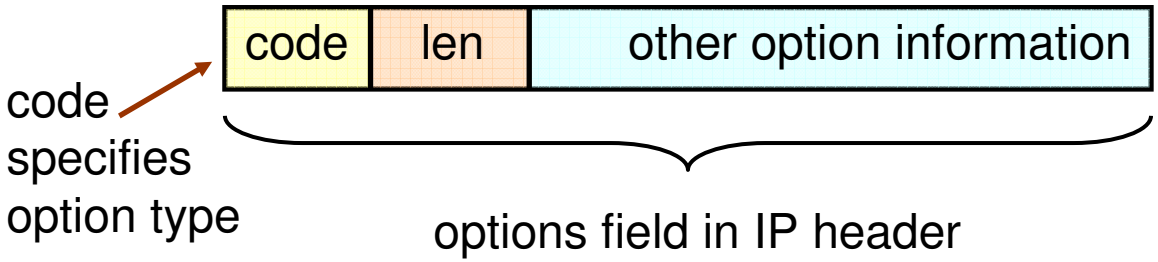
ping -j 192.168.35.1 88.33.18.17 192.150.196.6 193.204.129.2 univaq.it



Anche traceroute permette di creare pacchetti 'Source routed'

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



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 (ovvero allarmarsi se *ip option* è ON).

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.

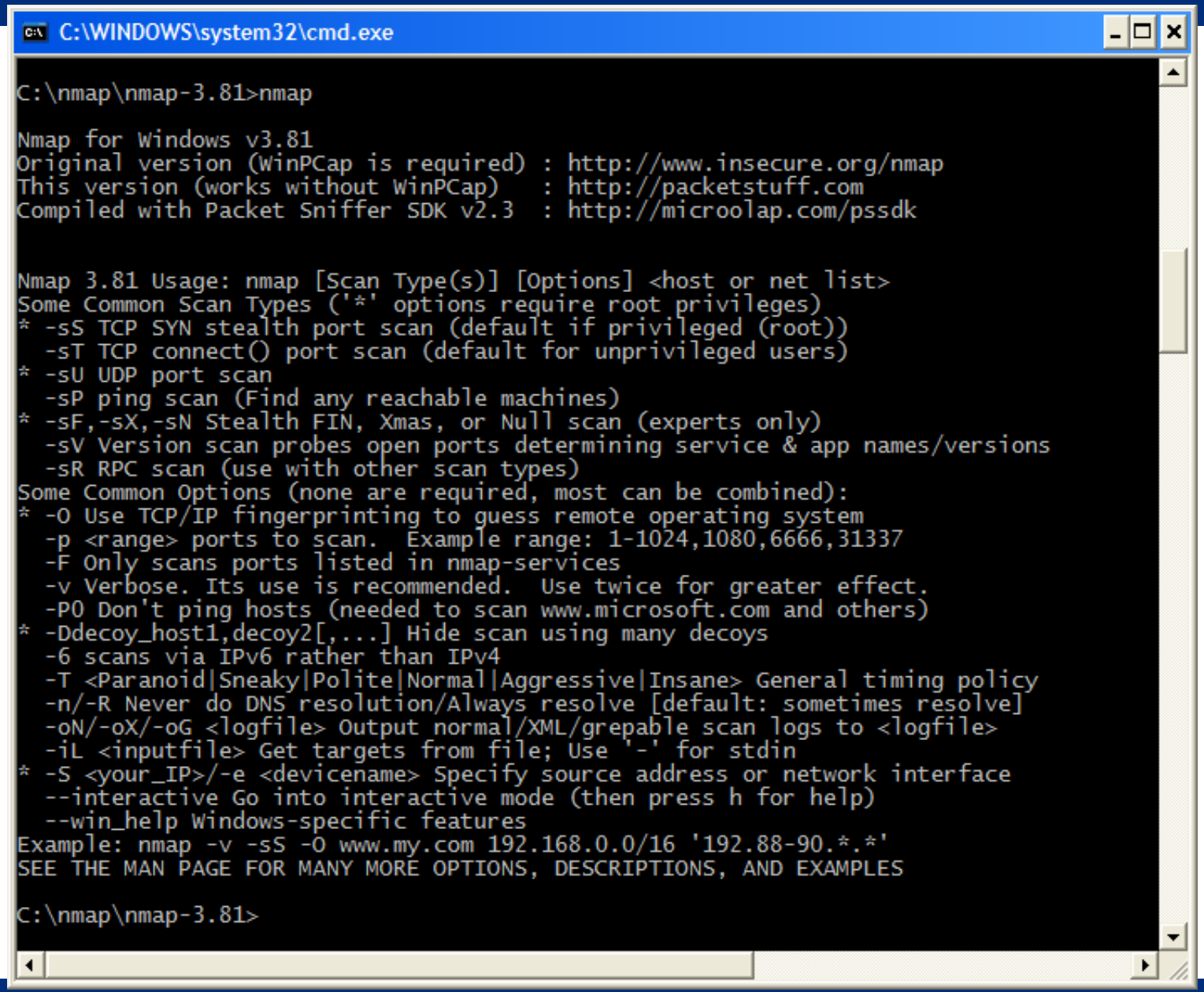
TCP Session Hijacking

- Session Hijacking è l'inserimento in una sessione TCP attiva.
- Spiando una connessione attiva è possibile sostituirsi ad uno dei due interlocutori.
  - **C** spia la connessione tra **A** e **B** e registra i numeri di sequenza dei pacchetti
  - **C** blocca **B** (ad es. via SYN Flood): l'utente in **B** vede interrompersi la sua sessione interattiva
  - **C** invia pacchetti con il corretto numero di sequenza, *con mittente B*, in modo che **A** non si accorga di nulla.
- *hunt* o *dsniff* automatizzano il processo
- I router non devono far entrare traffico che risulti proveniente dalla rete interna (*ingress filtering*).
- Per attaccanti sulla rete interna l'unica difesa è la cifratura dei pacchetti.

Portscan

- L'attaccante effettua da remoto un test dello stato delle porte TCP e/o UDP sulla macchina per determinare le porte attive (aperte), che ammettono connessioni in ingresso, e le porte non utilizzate (chiuse).
- Una porta aperta individua la presenza di un servizio di rete attivo offerto dall'host
- A partire dall'elenco dei servizi offerti l'attaccante può tentare di individuare eventuali vulnerabilità conosciute ed effettuare exploits sulle stesse.
- Vengono impiegate diverse tecniche per cercare di sfuggire all'individuazione
  - Decoy scan
  - SYN scan
  - FIN scan
  - ...

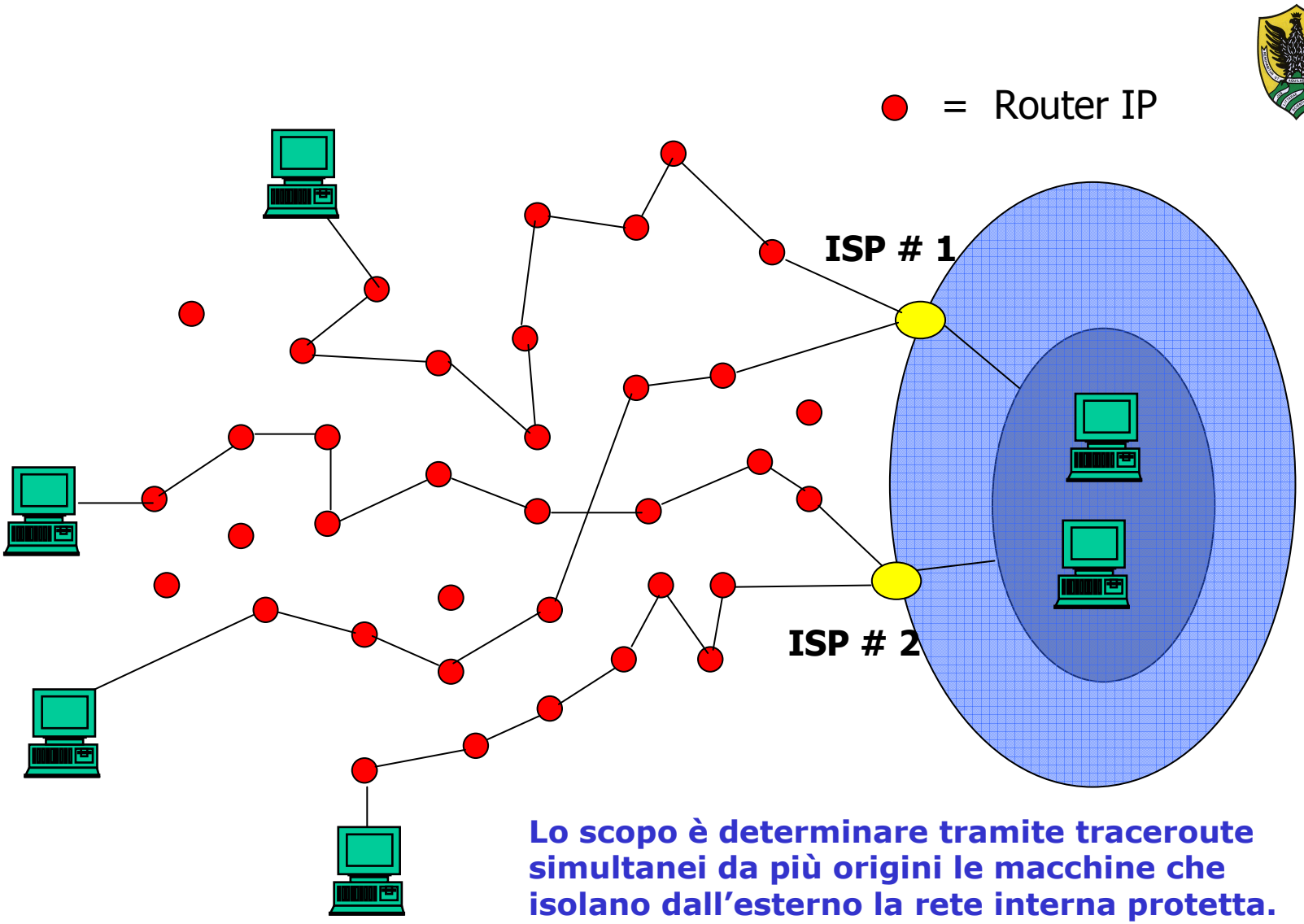
Network scan



Nmap

```
# nmap -sS -O target
  Starting nmap V. 3.81
  Interesting ports on target (192.168.1.1):
  Port        State        Service
  21/tcp      open         ftp
  22/tcp      open         ssh
  23/tcp      open         telnet
  25/tcp      open         smtp
  37/tcp      open         time
  53/tcp      open         domain
  111/tcp     open         sunrpc
  113/tcp     open         auth
  135/tcp     open         loc-srv
  139/tcp     open         netbios-ssn
  515/tcp     open         printer
  849/tcp     open         unknown
  853/tcp     open         unknown
  7000/tcp    open         afs3-fileserver
  TCP Sequence Prediction: Class=64K rule
                           Difficulty=1 (Trivial joke)
  Remote operating system guess: HP-UX B.10.20 A with
  tcp_random_seq = 0
```

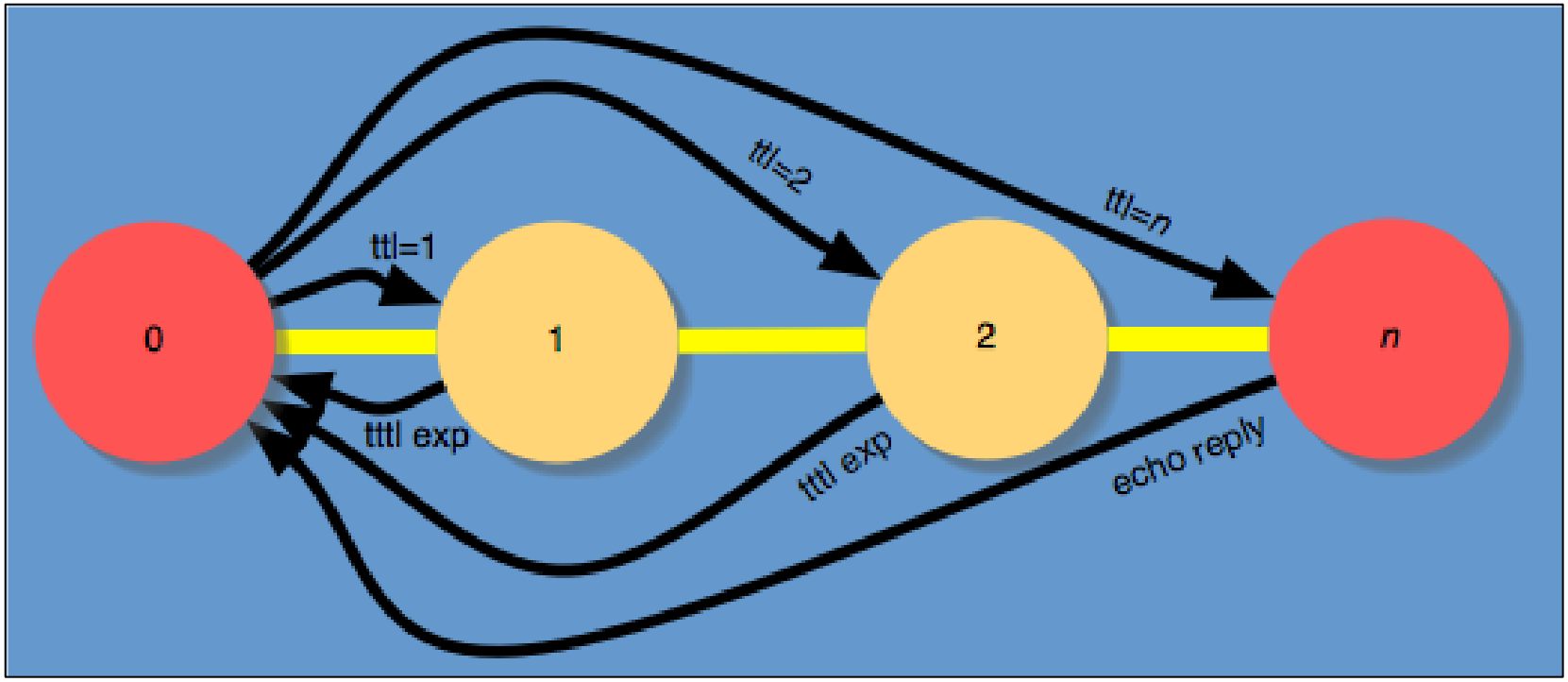
Network mapping con traceroute



Lo scopo è determinare tramite traceroute simultanei da più origini le macchine che isolano dall'esterno la rete interna protetta.

Traceroute

- Usa TTL ed ICMP
- Può aiutare a determinare il 'routing behavior'
  - stabilità
  - topologia



Traceroute in azione

Client session →

```
root@alpha3> traceroute 192.168.1.121
traceroute to 192.168.1.121 (192.168.1.121), 30 hops max, 40 byte packets
1  gw165.testing.net (172.21.165.1)  4.536 ms  3.728 ms  1.789 ms
2  10.1.1.123 (10.1.1.123)  6.377 ms  5.151 ms  6.532 ms
3  192.168.1.121 (192.168.1.121)  6.536 ms  7.892 ms  6.545 ms
```

```
12:58 alpha3.52716 > 192.168.1.121.33435: udp 12
12:58 gw165.testing.net > alpha3: icmp: time exceeded in-transit
12:58 alpha3.52716 > 192.168.1.121.33436: udp 12
12:58 gw165.testing.net > alpha3: icmp: time exceeded in-transit
12:58 alpha3.52716 > 192.168.1.121.33437: udp 12
12:58 gw165.testing.net > alpha3: icmp: time exceeded in-transit
```

3 UDP packets (dst ports 33435, 33436, 33437) are sent to the dst host (ttl=1).

```
12:58 alpha3.52716 > 192.168.1.121.33438: udp 12
12:58 10.1.1.123 > alpha3: icmp: time exceeded in-transit
12:58 alpha3.52716 > 192.168.1.121.33439: udp 12
12:58 10.1.1.123 > alpha3: icmp: time exceeded in-transit
12:58 alpha3.52716 > 192.168.1.121.33440: udp 12
12:58 10.1.1.123 > alpha3: icmp: time exceeded in-transit
```

3 UDP packets (dst ports 33438, 33439, 33440) are sent to the dst host (ttl=2).

```
12:58 alpha3.52716 > 192.168.1.121.33441: udp 12
12:58 192.168.1.121 > alpha3: icmp: 192.168.1.121 udp port 33441 unreachable
12:58 alpha3.52716 > 192.168.1.121.33442: udp 12
12:58 192.168.1.121 > alpha3: icmp: 192.168.1.121 udp port 33442 unreachable
12:58 alpha3.52716 > 192.168.1.121.33443: udp 12
12:58 192.168.1.121 > alpha3: icmp: 192.168.1.121 udp port 33443 unreachable
```

3 UDP packets (dst ports 33441, 33442, 33443) are sent to the dst host (ttl=3).

Notare che la porta sorgente è sempre la stessa per tutta la durata del traceroute

Traceroute in azione

- In questo caso ci sono solo due routers tra sorgente (alpha3) e destinazione (192.168.1.121).
- La sequenza dei pacchetti è la seguente:
  - 3 pacchetti UDP (dst ports 33435, 33436, 33437) sono inviati a dst-host (ttl=1). Ci sono 3 'round trip times' per ogni network hop – ecco perchè vediamo 3 pacchetti inviati da traceroute.
  - I pacchetti muoiono al primo router (gw165) che genera un time-exceeded error message. L'indirizzo IP di gw165 viene visualizzato lato client.
  - 3 pacchetti UDP (dst ports 33438, 33439, 33440) sono inviati a dst-host (ttl=2)
  - 3 pacchetti UDP (dst ports 33441, 33442, 33443) sono inviati a dst-host (ttl=3)
  - I pacchetti arrivano a dest-host (192.168.1.121) ma non c'è nessun server in ascolto sulle porte richieste, pertanto dest-host genera un messaggio di errore "port unreachable".
  - L'IP di dest-host viene visualizzato lato client.
- La utility traceroute distingue tra messaggi "time-exceeded" e "unreachable", e termina quando riceve un "unreachable".

Traceroute univaq.it

C:\WINDOWS\system32\cmd.exe

```
C:\Documents and Settings\ity10250.TECHNOLABS>tracert univaq.it

Tracing route to univaq.it [192.150.195.38]
over a maximum of 30 hops:

  1     *        *        *     Request timed out.
  2    <1 ms    <1 ms    <1 ms  213.82.155.18
  3    29 ms    25 ms     6 ms  host17-18.pool8833.interbusiness.it [88.33.18.17]
  4    81 ms     6 ms     5 ms  r-pe27-vl2.opb.interbusiness.it [217.141.254.141]
  5    67 ms    34 ms    36 ms  host249-8.pool82184.interbusiness.it [82.184.8.249]
  6    41 ms    55 ms    29 ms  r-rm156-vl3.opb.interbusiness.it [151.99.29.144]
  7   167 ms   200 ms   278 ms  host182-8.pool82184.interbusiness.it [82.184.8.182]
  8   123 ms   119 ms    74 ms  garr-nap.namex.it [193.201.28.15]
  9   123 ms   141 ms    59 ms  rt-rm2-rt-rm1-2.rm1.garr.net [193.206.134.117]
 10    74 ms    49 ms    39 ms  rt-rm1-rc-aq-2.rm1.garr.net [193.206.134.102]
 11    27 ms    30 ms    43 ms  uniaq-rc.aq.garr.net [193.206.136.146]
 12   170 ms   117 ms   117 ms  redavpc.cc.univaq.it [192.150.196.6]
 13   457 ms   411 ms   437 ms  193.204.129.2
 14   347 ms   461 ms   398 ms  matematica.univaq.it [192.150.195.38]

Trace complete.

C:\Documents and Settings\ity10250.TECHNOLABS>
```


Network Mapping

Simultaneous traceroutes

```
10:32:24.722 north.mappem.com.38758 > ns.target.net.33476: udp 12
10:32:24.756 north.mappem.com.38758 > ns.target.net.33477: udp 12
10:32:24.801 north.mappem.com.38758 > ns.target.net.33478: udp 12
10:32:24.833 north.mappem.com.38758 > ns.target.net.33479: udp 12
10:32:24.944 north.mappem.com.38758 > ns.target.net.33481: udp 12
10:32:24.975 north.mappem.com.38758 > ns.target.net.33482: udp 12
10:32:25.078 north.mappem.com.38758 > ns.target.net.33484: udp 12

10:32:26.541 south.mappem.com.48412 > ns.target.net.33510: udp 12
10:32:26.745 south.mappem.com.48412 > ns.target.net.33512: udp 12
10:32:26.837 south.mappem.com.48412 > ns.target.net.33513: udp 12
10:32:26.930 south.mappem.com.48412 > ns.target.net.33514: udp 12
10:32:27.033 south.mappem.com.48412 > ns.target.net.33515: udp 12
10:32:27.231 south.mappem.com.48412 > ns.target.net.33517: udp 12
10:32:27.436 south.mappem.com.48412 > ns.target.net.33519: udp 12

10:32:26.425 east.mappem.com.58853 > ns.target.net.33490: udp 12
10:32:26.541 east.mappem.com.58853 > ns.target.net.33491: udp 12
10:32:26.744 east.mappem.com.58853 > ns.target.net.33493: udp 12
10:32:26.836 east.mappem.com.58853 > ns.target.net.33494: udp 12
10:32:26.930 east.mappem.com.58853 > ns.target.net.33495: udp 12
10:32:27.033 east.mappem.com.58853 > ns.target.net.33496: udp 12
10:32:27.232 east.mappem.com.58853 > ns.target.net.33498: udp 12
10:32:27.323 east.mappem.com.58853 > ns.target.net.33499: udp 12
```

tcpdump output relativo a 3 traceroutes simultanei,
Inviati da 3 macchine ubicate in differenti continenti.

Traceroute Filters

`tcpdump` filter :

```
(udp[2:2] > 33434) and (udp[2:2] <= 34999)
```

Cisco ACLs:

Block inbound traceroutes

```
access-list 110 deny udp any 172.16.0.0 0.0.255.255 gt 33434
access-list 111 deny icmp 172.16.0.0 0.0.255.255 any time-exceeded
access-list 111 deny icmp 172.16.0.0 0.0.255.255 any unreachable
```

Allow outbound traceroutes

```
access-list 111 permit udp 172.16.0.0 0.0.255.255 any gt 33434
access-list 110 permit icmp any 172.16.0.0 0.0.255.255 time-exceeded
access-list 110 permit icmp any 172.16.0.0 0.0.255.255 unreachable
```

Prevent router from generating "unreachable" messages (interface command)

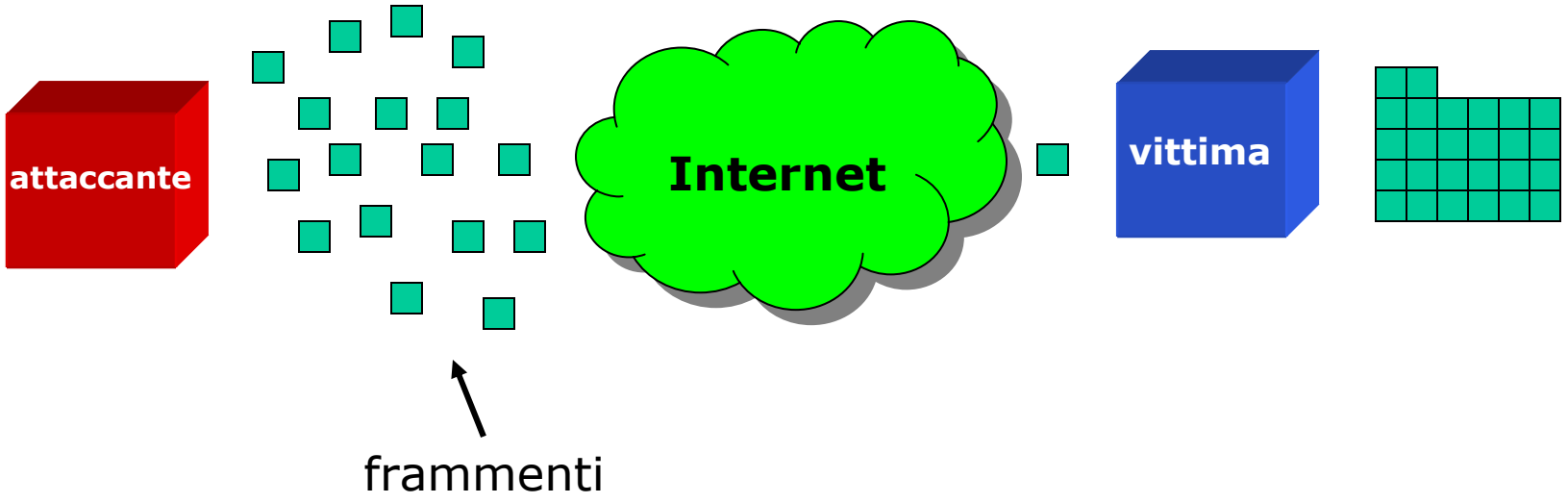
```
no ip unreachables
```

Denial of Service

- Attacchi di tipo Denial of Service (DoS):
  - scopo: rendere inutilizzabile un servizio o una risorsa (eventualmente per sostituirsi ad essa).
  - metodo: inibire la connessione al router o al server, provocarne il crash o comunque il blocco
- Distributed DoS (DDoS)
  - attacco di tipo DoS proveniente da più sorgenti contemporaneamente
  - sfruttano molti host compromessi su reti diverse per lanciare attacchi DoS su vittima
- Sempre usato IP spoofing
  - molto difficile rintracciare l'origine dell'attacco

DoS: ping of death

- Invio pacchetti ICMP ping di dimensione maggiore della dimensione massima consentita (65535 bytes)
  - Attaccante frammenta pacchetti
  - Frammenti riassemblati da vittima
  - Ultimo frammento causa overflow



Ping o' Death Attack

Denial of Service

```
12:43:58.431 big.pinger.org > www.mynetwork.net:
                              icmp: echo request (frag 4321:380@0+)
12:43:58.431 big.pinger.org > www.mynetwork.net: (frag 4321:380@2656+)
12:43:58.431 big.pinger.org > www.mynetwork.net: (frag 4321:380@3040+)
12:43:58.431 big.pinger.org > www.mynetwork.net: (frag 4321:380@3416+)
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+)
12:43:58.431 big.pinger.org > www.mynetwork.net: (frag 4321:380@760+)
...
12:43:58.491 big.pinger.org > www.mynetwork.net: (frag 4321:380@63080+)
12:43:58.491 big.pinger.org > www.mynetwork.net: (frag 4321:380@63456+)
12:43:58.491 big.pinger.org > www.mynetwork.net: (frag 4321:380@63840+)
12:43:58.491 big.pinger.org > www.mynetwork.net: (frag 4321:380@64216+)
12:43:58.491 big.pinger.org > www.mynetwork.net: (frag 4321:380@64600+)
12:43:58.491 big.pinger.org > www.mynetwork.net: (frag 4321:380@64976+)
12:43:58.491 big.pinger.org > www.mynetwork.net: (frag 4321:380@65360)
```

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):

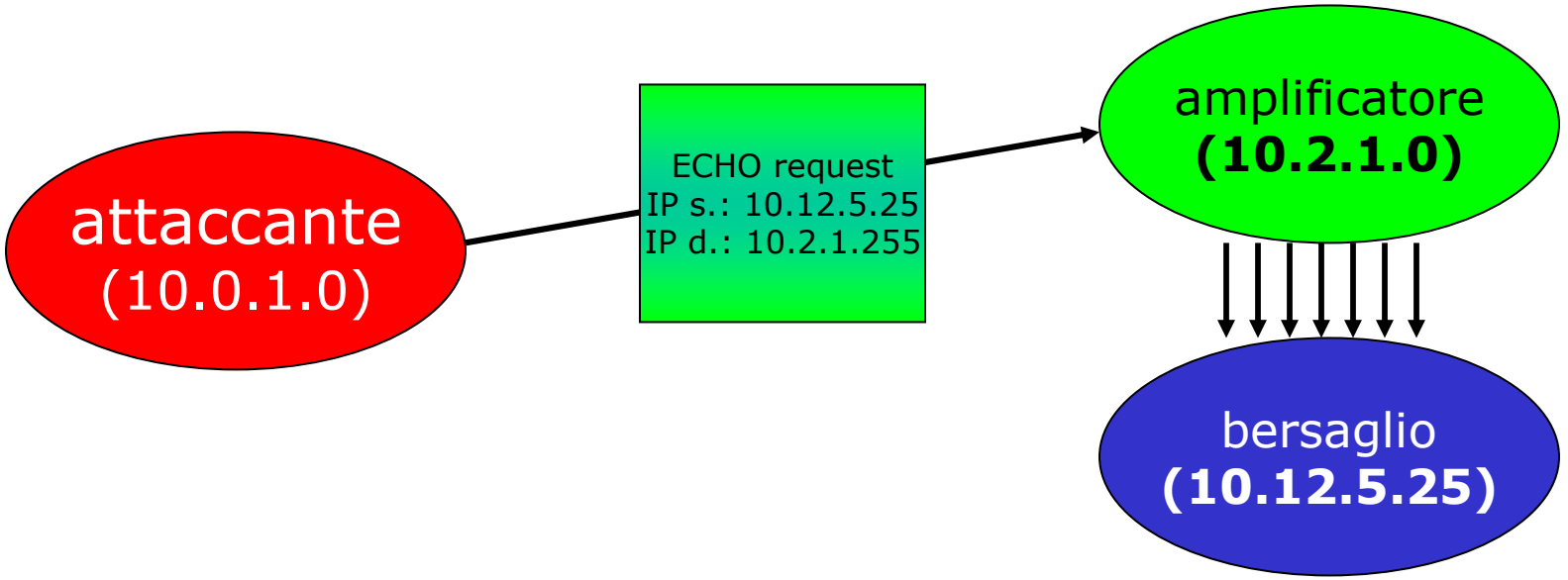
```
(ip[6:1] & 0x20 = 0) and (ip[6:2] & 0x1fff != 0)
```

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
```


DoS: smurf / fraggle

- Broadcast storm
  - Invio di pacchetti ICMP echo all'indirizzo di broadcast di una rete (fraggle usa pacchetti UDP)
- Coinvolti solitamente almeno tre siti:
  - sito origine dell'attacco ("attaccante")
  - 2 siti vittime, uno intermedio ("amplificatore") ed uno "bersaglio"

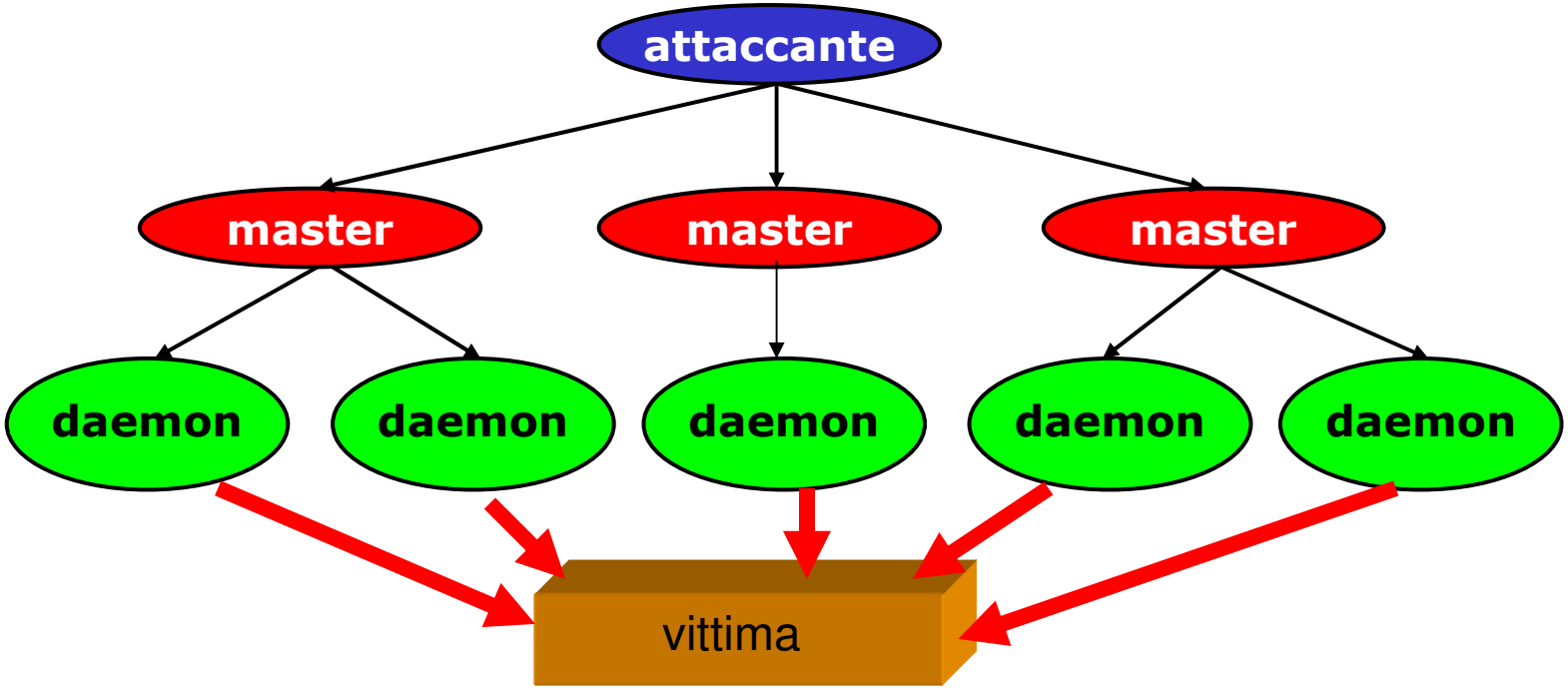


DoS: SYN Flood

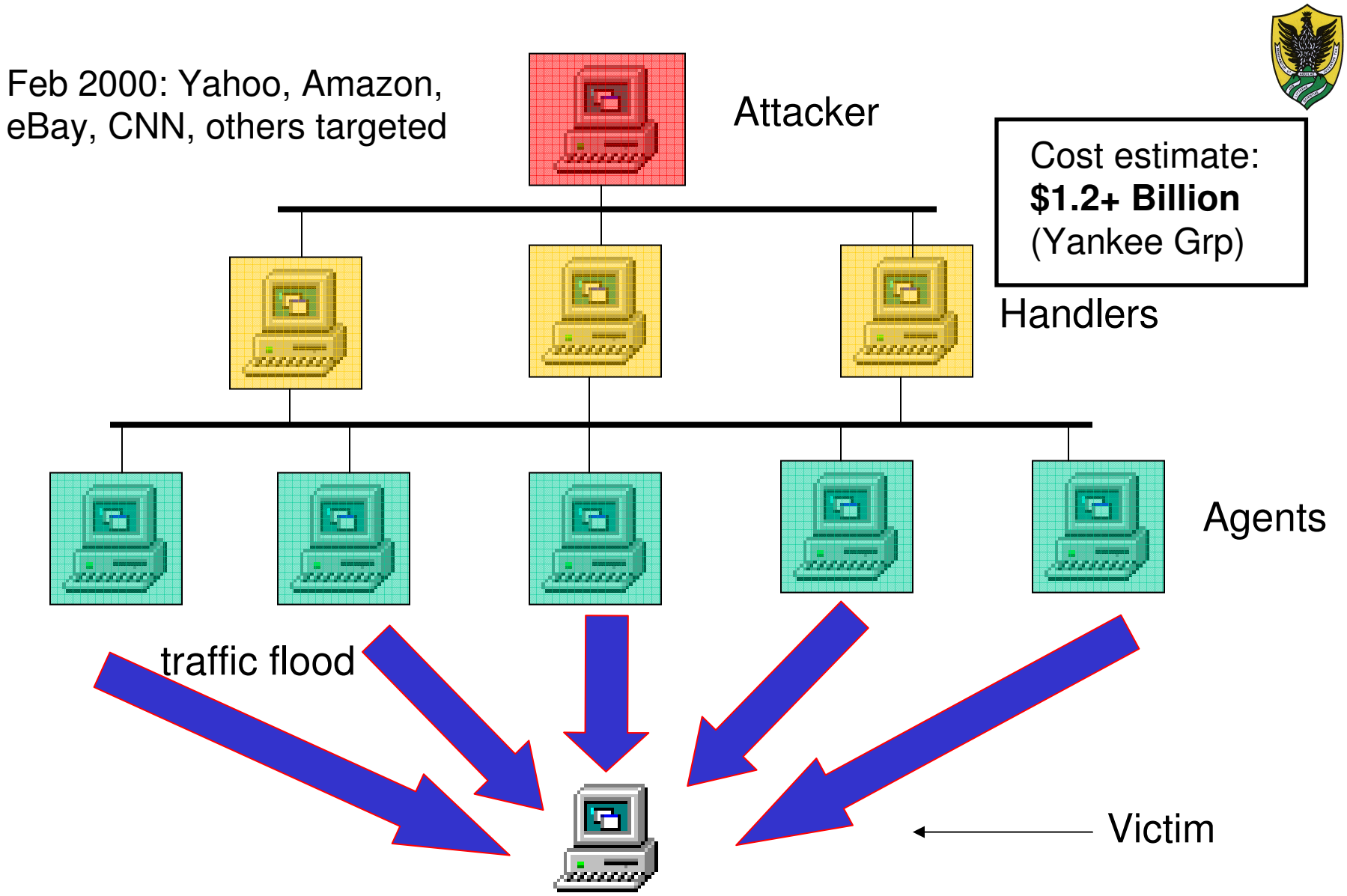
- Richiesta grande numero di connessioni da host diversi (spoofing), tramite invio pacchetti TCP SYN, senza mai inviare pacchetto di chiusura del “three-way handshake”
  - causa riempimento coda di connessione (può bloccare un router)
  - non è possibile rintracciare origine attacco (mittente falsificato)
  - non è possibile usare access-list (ip sorgente varia in modo casuale)
  - Contromisure: aumentare la dimensione della coda di connessione (SYN ACK queue) e diminuire il tempo di time-out per il "three-way handshake".
  - Filtri contro l’IP spoofing diminuiscono probabilità che la propria LAN sia utilizzata come base per questo tipo di attacchi

Distributed DoS (DDoS)

- Attacchi DoS provenienti contemporaneamente da più sorgenti (anche ~ 1000)
- Struttura multi-livello
  attaccante → master → demoni → vittima



Distributed Denial of Service Attacks



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!

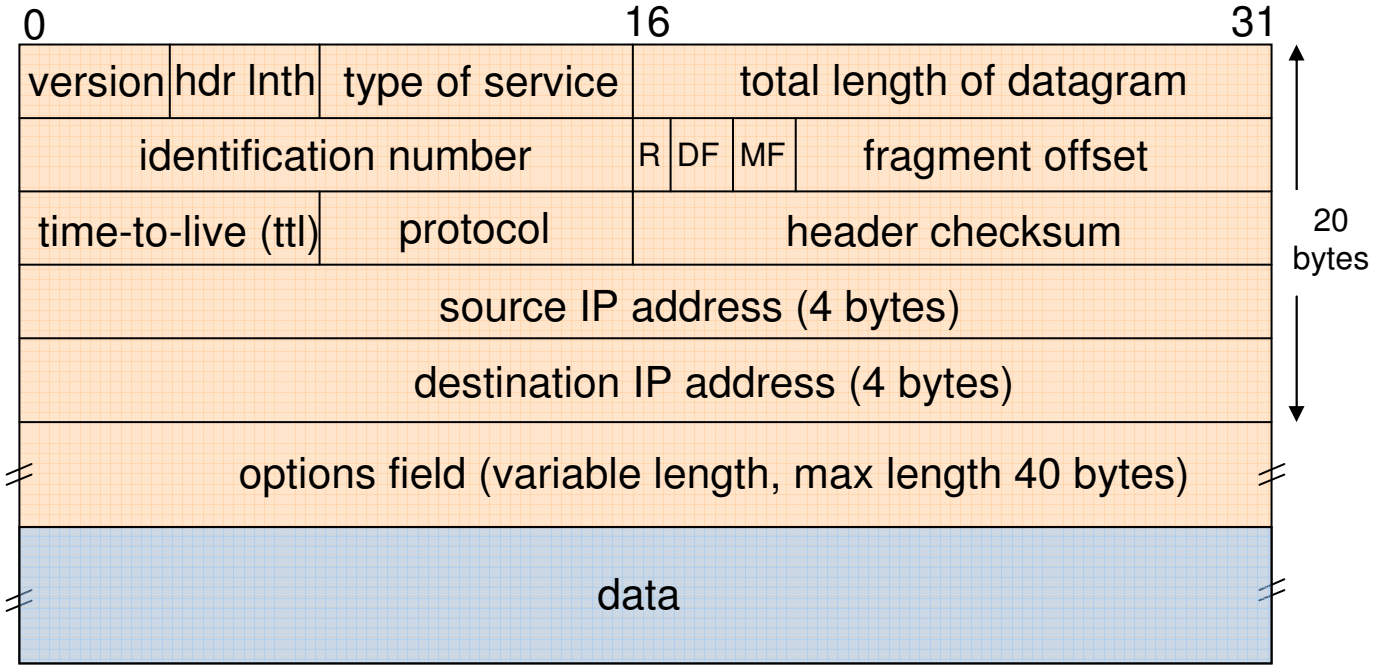
Detect impossible fragments:

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

Impossible Fragmentation

- The fact that the header offset value has to be multiplied by 8 to get the number of bytes implies that offsets can only be on 8-byte boundaries. In other words, this means that all non-final fragments must be multiples of 8 bytes in length (in order to re-assemble correctly).
- Along this line, an ID system designer might write a filter to detect non-final fragments that carry data that is not a multiple of 8 bytes in length. Any such fragments are non-natural and have almost certainly been crafted using a special tool. An example filter to detect these fragments is given in the slide above (the filter works because all multiples of 8 have the last 3 bits = 0).

Impossible Fragmentation



Detect impossible fragments:

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

the filter works because all multiples of 8 have the last 3 bits = 0

"Small" Services (spesso usati per diagnostica e da hackers)

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

Diagnostic Port Attack

Denial of Service

`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)
```

Scanning with SF Packets

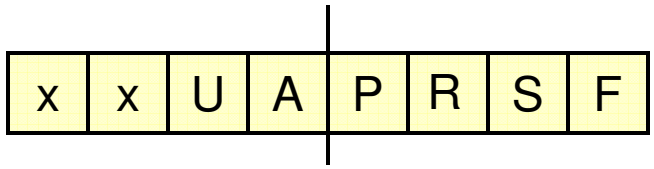
```
16:36:06.54 pop3.net.0 > 192.168.26.203.110: SF 1681588224:1681588224
16:36:06.54 pop3.net.0 > 192.168.18.84.110:  SF 1681588224:1681588224
16:36:06.57 pop3.net.0 > 192.168.43.254.110: SF 1681588224:1681588224
16:36:06.58 pop3.net.0 > 192.168.24.209.110: SF 1681588224:1681588224
16:36:06.60 pop3.net.0 > 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

Filtrare i Flags TCP

tcpdump permette il filtering di ognuno dei tcp flags (i routers Cisco NO)



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`

Pertanto, volendo intercettare connessioni attive aperte vs la porta IMAP si usa:

(SYN flag set, ACK flag not set)

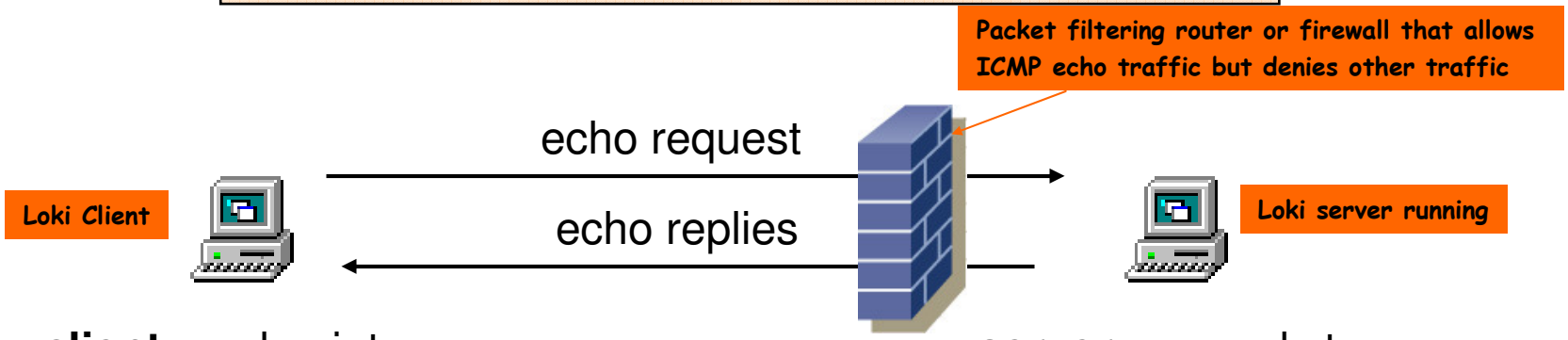
```
tcp and (dst port 143)
    and (tcp[13] & 0x02 != 0) and (tcp[13] & 0x10 = 0)
```

Information Tunneling (Loki)

Note: Tunneling can be accomplished using any type of packet – but Loki uses ICMP echo packets by default

ICMP **echo request** or **echo reply** packet

Loki data contained in echo packet payload



client can log into server and issue commands

server responds to commands from client

Information Tunneling (Loki)

Loki client session – client receives server's `/etc/passwd` file

```
[root@client virwin]# loki -d server.test.net
LOKI2   route [(c) 1997 guild corporation worldwide]
loki> ls /etc/passwd
/etc/passwd
loki> more /etc/passwd
::::::::::::::
/etc/passwd
::::::::::::::
root:3QZC8SBkLivns:0:0:root:/root:/bin/bash
daemon:*:1:1:daemon:/usr/sbin:/bin/sh
bin:*:2:2:bin:/bin:/bin/sh
sys:*:3:3:sys:/dev:/bin/sh
man:*:6:100:man:/var/catman:/bin/sh
lp:*:7:7:lp:/var/spool/lpd:/bin/sh
mail:*:8:8:mail:/var/spool/mail:/bin/sh
news:*:9:9:news:/var/spool/news:/bin/sh
uucp:*:10:10:uucp:/var/spool/uucp:/bin/sh
proxy:*:13:13:proxy:/bin:/bin/sh
loki>
```

ftp://ftp.2600.com/pub/publications/phrack/PHRACK51.gz

Information Tunneling (Loki)

Notare che ad ogni echo request (ping)
Corrispondono diverse echo reply.
Questo comportamento è indicativo di una
Applicazione client-server nascosta

Loki sniffer trace -- only ping packets are recorded by the sniffer

```
12:58:22.225 client.test.net > server.test.net: icmp: echo request
12:58:22.225 server.test.net > client.test.net: icmp: echo reply
12:58:22.275 server.test.net > client.test.net: icmp: echo reply
12:58:22.285 server.test.net > client.test.net: icmp: echo reply
12:58:28.985 client.test.net > server.test.net: icmp: echo request
12:58:28.985 server.test.net > client.test.net: icmp: echo reply
12:58:29.035 server.test.net > client.test.net: icmp: echo reply
12:58:29.055 server.test.net > client.test.net: icmp: echo reply
12:58:29.075 server.test.net > client.test.net: icmp: echo reply
12:58:29.095 server.test.net > client.test.net: icmp: echo reply
12:58:29.115 server.test.net > client.test.net: icmp: echo reply
12:58:29.135 server.test.net > client.test.net: icmp: echo reply
12:58:29.155 server.test.net > client.test.net: icmp: echo reply
12:58:29.175 server.test.net > client.test.net: icmp: echo reply
12:58:29.195 server.test.net > client.test.net: icmp: echo reply
12:58:29.215 server.test.net > client.test.net: icmp: echo reply
12:58:29.235 server.test.net > client.test.net: icmp: echo reply
12:58:29.255 server.test.net > client.test.net: icmp: echo reply
12:58:29.275 server.test.net > client.test.net: icmp: echo reply
```

Information Tunneling (Loki)

Look for **ICMP echo request** or **echo reply** packets :

```
((icmp[0] = 8) or (icmp[0] = 0))
```

where the packet **sequence number** is set to **`f001`**
(for original implementation released by **`www.2600.com/phrack`**):

```
and ((icmp[6:2] = 0xf001) or (icmp[6:2] = 0x01f0))
```

Note: In this case we need to look for both the “big-endian” and “little-endian” versions of the sequence number.

Recall that packet header values are always transmitted in “network byte order” or big-endian format. ***So why look for the little-endian version****?*

Regular FTP

FTP data transfer – normal

```
21:50:35  192.168.16.3.45472 > ftp.server.org.21:  S
21:51:33   ftp.server.org.20 > 192.168.16.3.45473: S
21:52:14   ftp.server.org.20 > 192.168.16.3.45474: S
21:52:31   ftp.server.org.20 > 192.168.16.3.45475: S
21:53:18   ftp.server.org.20 > 192.168.16.3.45476: S
21:53:38   ftp.server.org.20 > 192.168.16.3.45477: S
21:54:15   ftp.server.org.20 > 192.168.16.3.45478: S
21:54:57   ftp.server.org.20 > 192.168.16.3.45479: S
21:55:13   ftp.server.org.20 > 192.168.16.3.45480: S
```

- The *established* ACL will prevent an external FTP server from opening FTP data connections to machines on your network
- The inbound SYN connections to high-numbered ports may trip portscan alarms
- Since we do not want to allow inbound active open connections in general, we must come up with a better plan …
- We might allow only active opens that use a source port of 20, but that is dangerous (example: **`nmap`** has a command line switch to set the source port of the scan)

Passive FTP

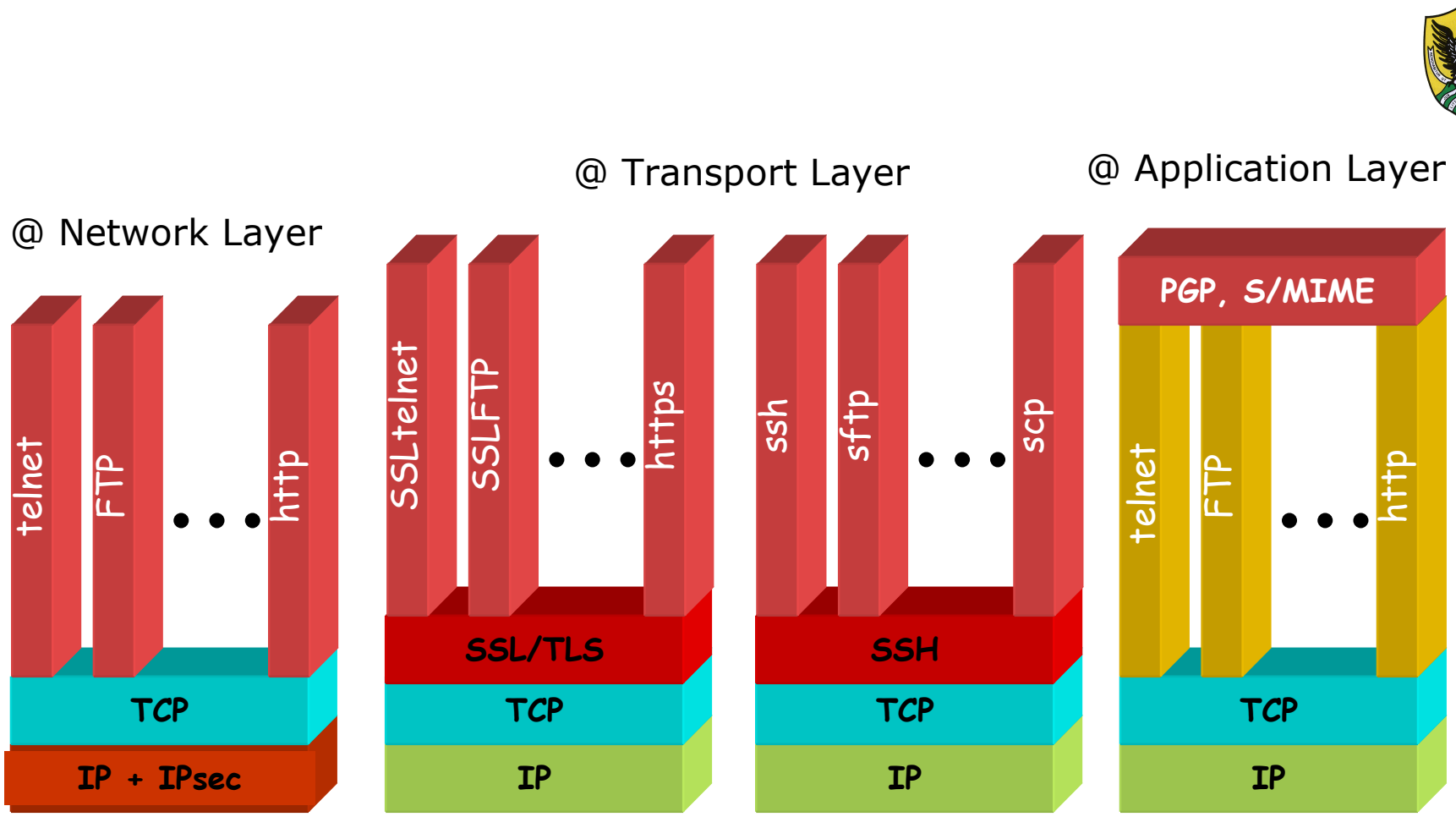
FTP data transfer - passive

```
06:11:21  192.168.16.3.1986 > ftp.server.org.21: S
06:11:25  192.168.16.3.1987 > ftp.server.org.47370: S
06:14:25  192.168.16.3.1988 > ftp.server.org.47377: S
06:17:10  192.168.16.3.1989 > ftp.server.org.47384: S
06:18:22  192.168.16.3.1990 > ftp.server.org.47387: S
06:20:52  192.168.16.3.1991 > ftp.server.org.47391: S
```

- If passive FTP is used, all active open requests are made outbound from the client to the server
- However, the external FTP server must be capable of operating in PASV mode (most are nowadays)
- Under this scheme, the *established* ACL will not block the data transfer, and this is the preferable way to do things
- If this is not possible for your site, a stateful firewall may be the only solution

Protocolli di protezione

Protocolli di Protezione



@ Datalink Layer (VLAN) ???

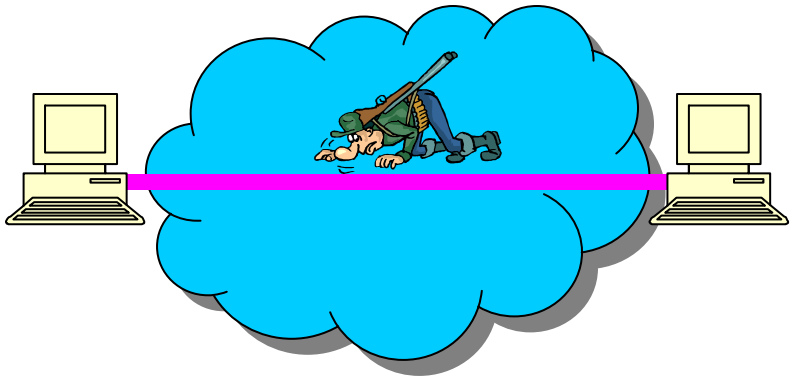
Protocolli

- **Network level**
  - trasparenti per le applicazioni, che non devono essere modificate
  - il network layer deve essere modificato
- **Transport level**
  - viene fornita una libreria di funzioni che può essere utilizzata dai programmi applicativi
  - necessaria la modifica dei programmi applicativi
- **Application level**
  - trasparenti per la rete
  - i servizi di sicurezza devono essere individualmente inclusi in ogni applicazione
  - necessaria la modifica dei programmi applicativi

IPsec

- Autenticazione e cifratura al livello Network
  - Authentication Header (AH)
    - **autentifica ogni pacchetto;**
  - Encapsulating Security Payload (ESP)
    - **cifra i dati in ogni pacchetto;**
  - Internet Key Exchange (IKE)
    - **protocollo di negoziazione**
      - metodi di autentificazione
      - metodi di cifratura
    - **consente scambio sicuro di chiavi**

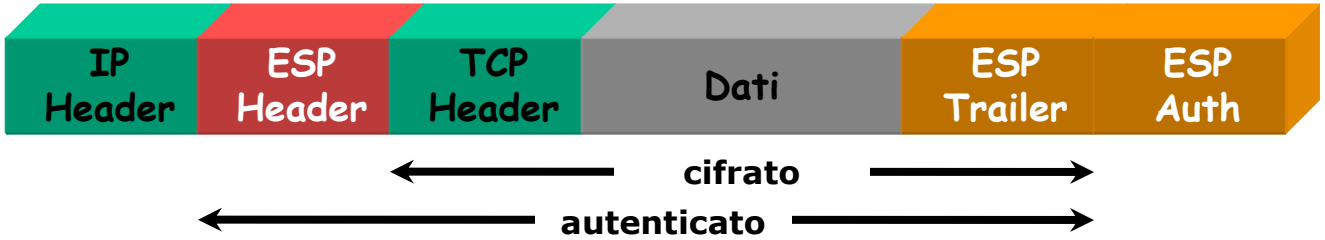
IPsec: end-to-end



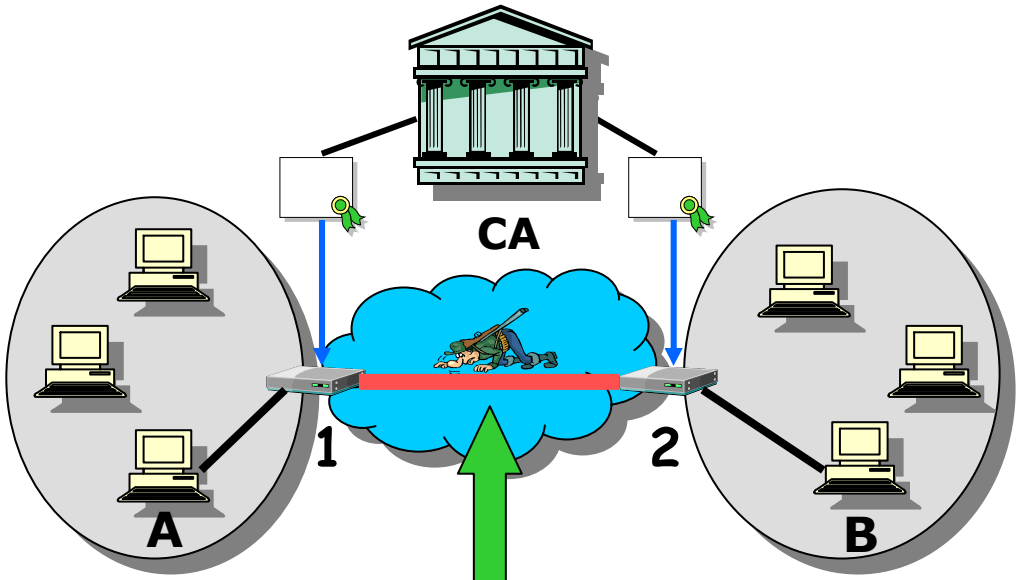
- **AH**
  - campi variabili: TOS, Frg offset, TTL, …
- **ESP**
  - autentificazione facoltativa
  - l'IP header non è protetto

Authentication Header

Encapsulated Security Payload

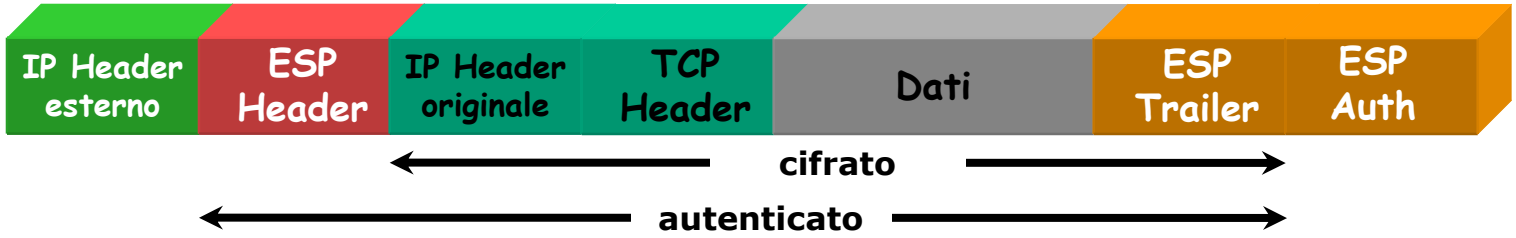


IPsec: tunnel



Gli indirizzi A e B non sono visibili all'esterno dei gateway 1 e 2

Encapsulated Security Payload



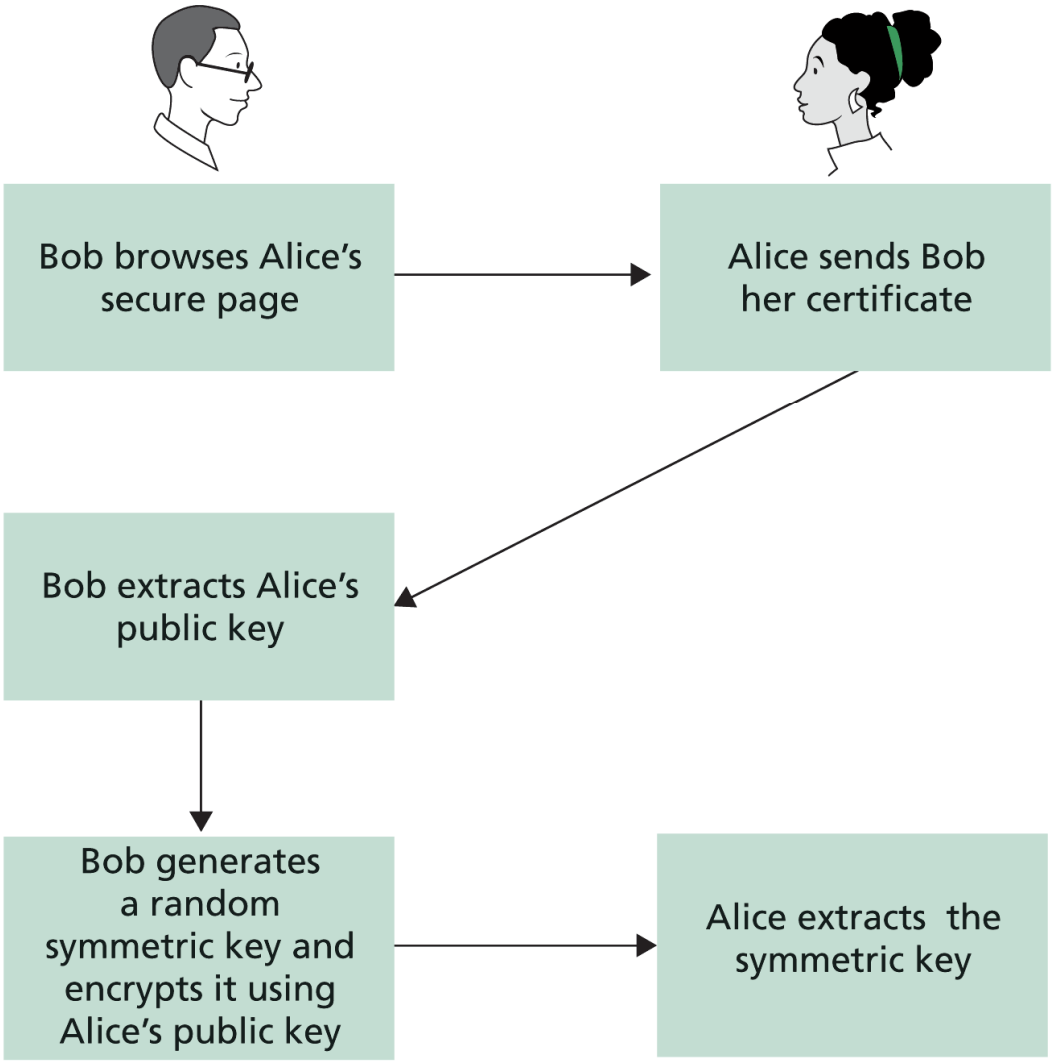
SSH

- Rimpiazza telnet e i comandi "r" (rlogin, rshell)
  - Versione 1 e Versione 2 (IETF SECSH Working Group);
    - **nella versione 1 richiede una distribuzione "manuale" delle chiavi di host e utenti;**
  - connessione crittografata: protegge da:
    - **IP spoofing;**
    - **IP source routing;**
    - **DNS spoofing;**
    - **sniffing;**
    - **man-in-the-middle;**
  - tunneling traffico tcp e X11;
  - compressione dei dati (facoltativa, utile per connessioni lente).
- Molto apprezzato anche dagli hacker, che spesso lo installano sulle macchine compromesse, perché
  - impedisce agli amministratori di capire quali dati vengano trasferiti e
  - per il meccanismo di port forwarding.

SSL/TLS

- Secure Sockets Layer (SSL) sviluppato da Netscape Communications
  - L'ultima versione (V3.0) è del Marzo 1996;
  - Netscape Communicator, Internet Explorer.
- Transport Layer Security (TSL) Working Group (IETF)
  - versione 1.0 del Gennaio 1999 (RFC 2246).
- Utilizza certificati X.509
- Può essere usato per ogni applicazione TCP (ad es. HTTP, Telnet, FTP, POP3, IMAP)
  - usato da praticamente tutti i server web "sicuri": https://…..
  - le vecchie applicazioni "insicure" possono essere usate in modalità tunnel (ad es. stunnel: http://www.stunnel.org)\
- Non interagisce bene con firewall/proxy (man-in-the-middle)

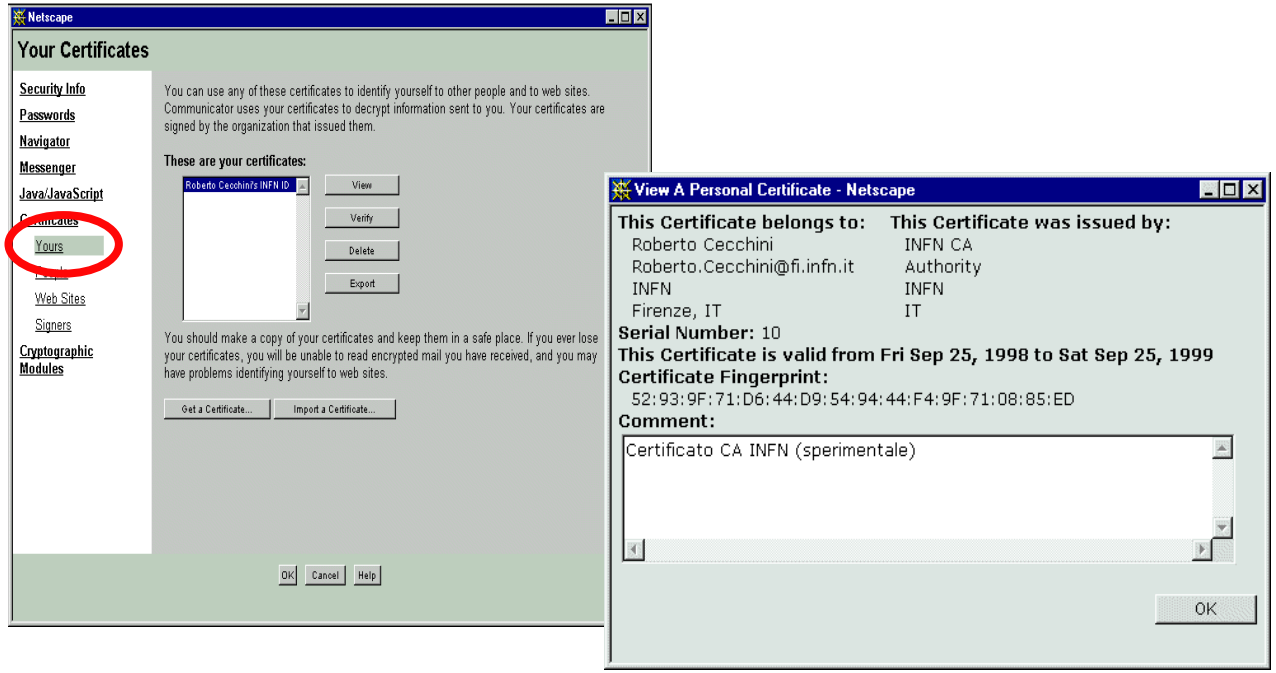
SSL : handshake



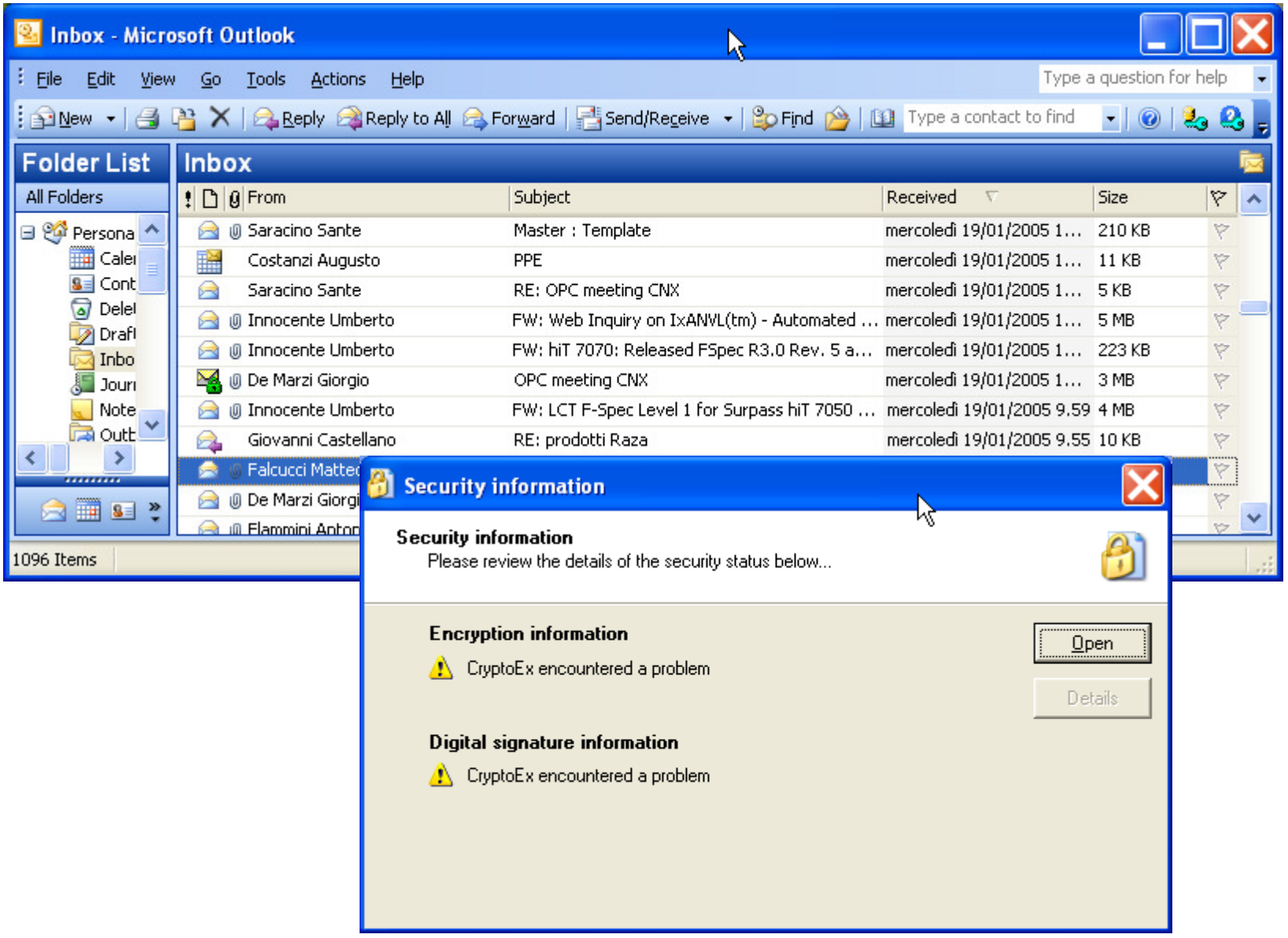
- High-level overview of the handshake phase of SSL

S/MIME 1/2

- Permette di inviare e-mail MIME firmati e crittografati
- Supportato (tra gli altri) da *Communicator* e *Outlook*
- Utilizza certificati X.509



S/MIME 2/2



Router di frontiera

- Responsabile dell'inoltro del traffico tra LAN ed Internet.
- E' il primo sbarramento della propria rete
  - difficile l'aggiramento da parte dell'end-user.
- Permette di centralizzare un buon numero di controlli di sicurezza.
- Fondamentale la sua protezione
  - una compromissione può aprire l'accesso alla LAN;
  - una inadeguata politica di filtraggio può esporre la LAN ad attacchi;
  - la corruzione delle tabelle di routing può provocare disservizi e accesso non autorizzato a dati.
- Un router correttamente configurato può minimizzare effetti derivanti da sito compromesso in attacchi DDoS.

Network Intrusion Detection System

- Forniscono informazioni utili su traffico ostile.
- Facilitano l'identificazione dell'origine di scansioni e/o attacchi.
- Permettono di lanciare allarmi "in tempo reale".
- Non sono una panacea per tutti i problemi di sicurezza. In particolare *non* sostituiscono:
  - firewall ben configurati;
  - security audit regolari;
  - una politica di sicurezza "seria".
- Producono falsi positivi.
- Possono essere "accecati" da attacchi DoS.
- Sono in difficoltà con reti veloci
  - rete di sensori (uno per macchina?).

Firewall

- Network device con almeno 2 interfacce di rete
  - Router o hardware dedicato
- Separa zone amministrativamente diverse
  - LAN esterna (insicura)
  - DMZ
  - Intranet
- Effettua routing fra le diverse zone
- Può effettuare mascheramento indirizzi (NAT)
- Filtra traffico fra le diverse zone tramite regole predefinite
  - Router con filtri è un firewall!
- Può mediare accessi a specifiche applicazioni
  - Proxy server

Packet filter

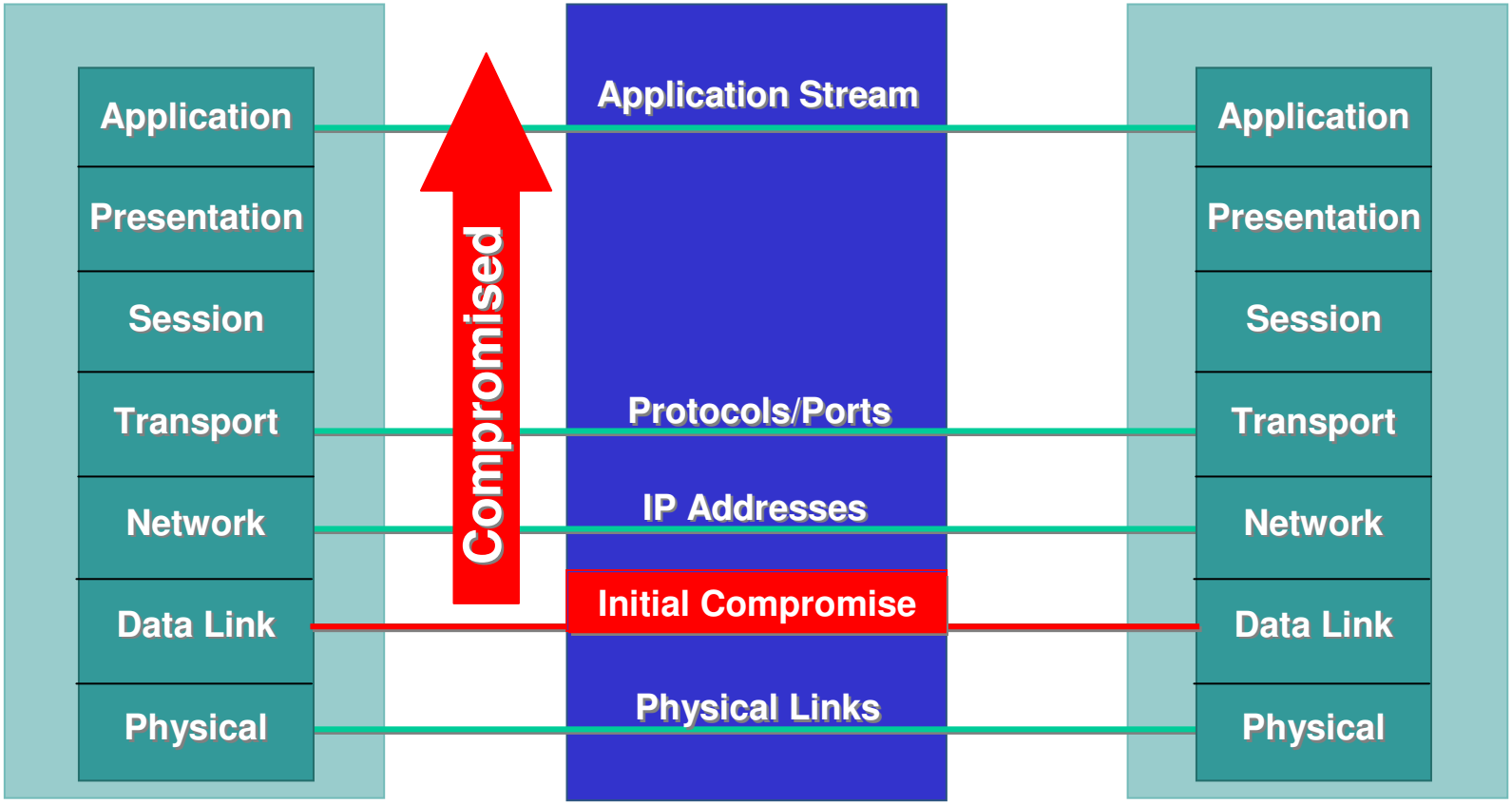
- Un device (ad es. un router) che controlla ogni pacchetto IP in arrivo per decidere se inoltrarlo o no.
- I filtri si basano sulle seguenti informazioni
  - semplici
    - **protocollo;**
    - **porte;**
    - **indirizzi sorgente e destinazione;**
    - **flag e opzioni tcp;**
    - **non prendono in considerazione la parte dati;**
    - **prendono le decisioni pacchetto per pacchetto;**
  - *stateful* (dinamici)
    - **tengono traccia delle connessioni in corso e possono avere conoscenza dei protocolli più comuni.**
    - **la necessità di tenere traccia delle connessioni aperte, ovviamente, aumenta il lavoro del filtro**

TechnoLabs

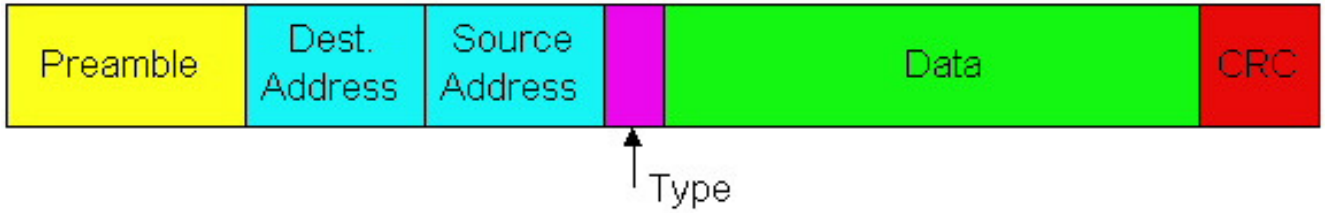
Ethernet

The Domino Effect

- Unfortunately this means if one layer is hacked, communications are compromised without the other layers being aware of the problem
- Security is only as strong as your weakest link
- When it comes to networking, layer 2 can be a VERY weak link



Ethernet Format: Framing

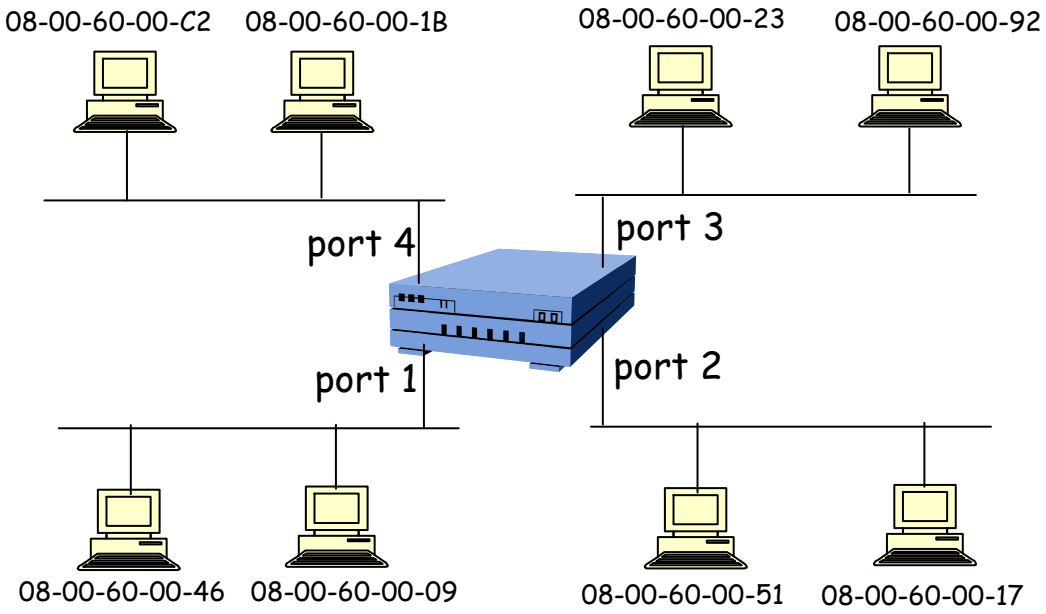


- Preamble: (clearing your throat)
  - 8 bytes, allows sender/receiver clocks to synchronize
- Destination/Source Address: (hey Paul, Tom here)
  - 6 bytes each
- Type:
  - 2 bytes, indicates higher layer protocol
  - 0x0800 is IP, 0x0806 is ARP
- Data: 46-1500 bytes
- FCS (CRC):
  - catches most transmission errors - errored frames dropped

Switching (same as Bridging)

- Goals
  - traffic isolation
  - "transparent" operation
  - plug-and-play
- Operation
  - store and forward Ethernet frames
  - examine frame header and selectively forward frame based on MAC destination address
  - when frame is to be forwarded on segment, uses CSMA/CD to access segment

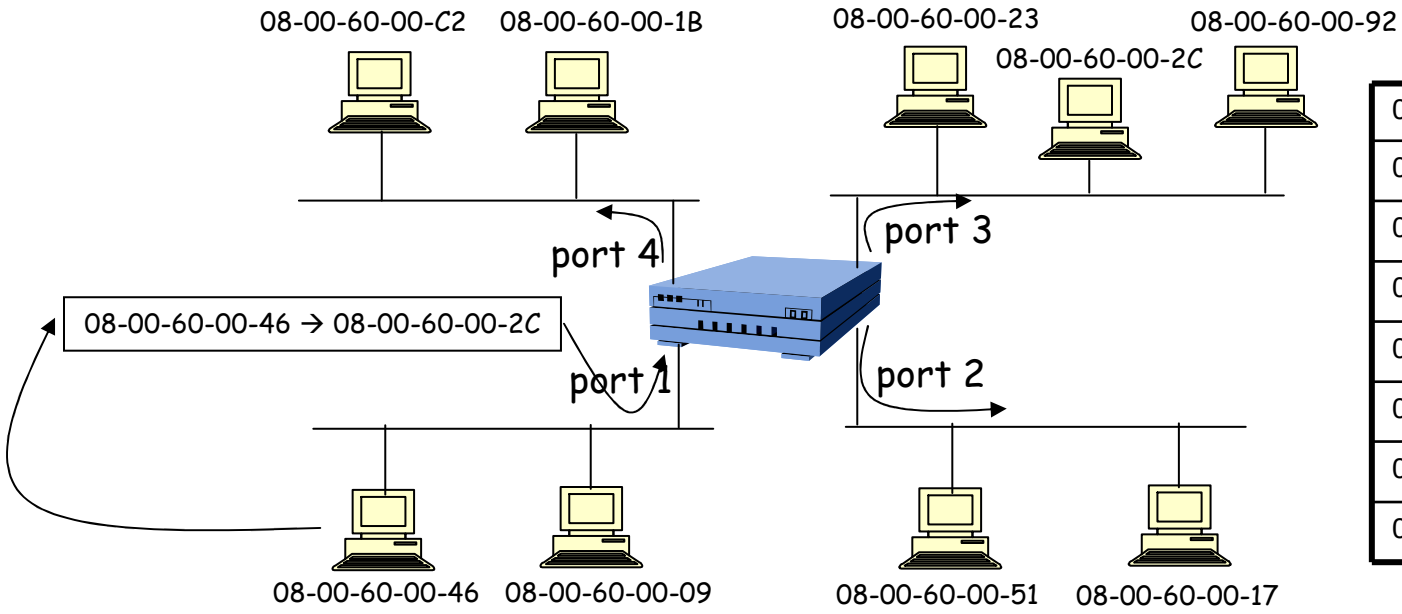
Bridged LAN – MAC learning



Learnt address table

08-00-60-00-09	1
08-00-60-00-C2	4
08-00-60-00-23	3
08-00-60-00-46	1
08-00-60-00-51	2
08-00-60-00-92	3
08-00-60-00-17	2
08-00-60-00-1B	4

Multicast bridging / flooding



Address table

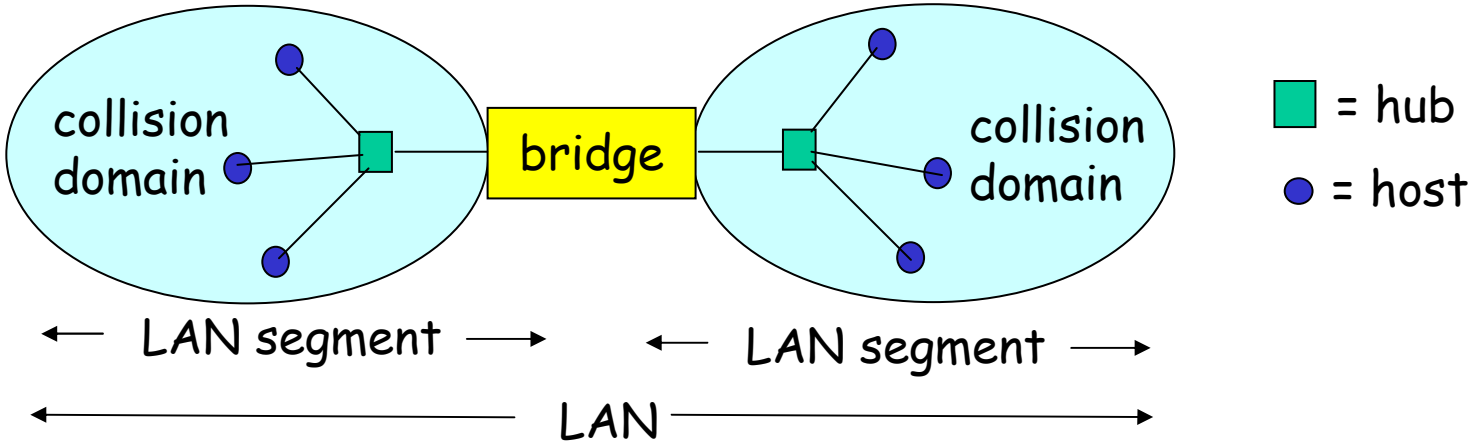
08-00-60-00-09	1
08-00-60-00-C2	4
08-00-60-00-23	3
08-00-60-00-46	1
08-00-60-00-51	2
08-00-60-00-92	3
08-00-60-00-17	2
08-00-60-00-1B	4

Implementing the MAC table

- Operations:
  - Dmac lookup
  - Smac learning
  - Entry aging
- Constraints:
  - Time budget: 672ns at 1Gbps
  - Space budget: cost-dependent, a few MBytes ?
  - Processing budget: polling required for aging
- Options:
  - Hash-tables
  - Binary search
  - CAM (content addressable memories)

Bridging: Advantages

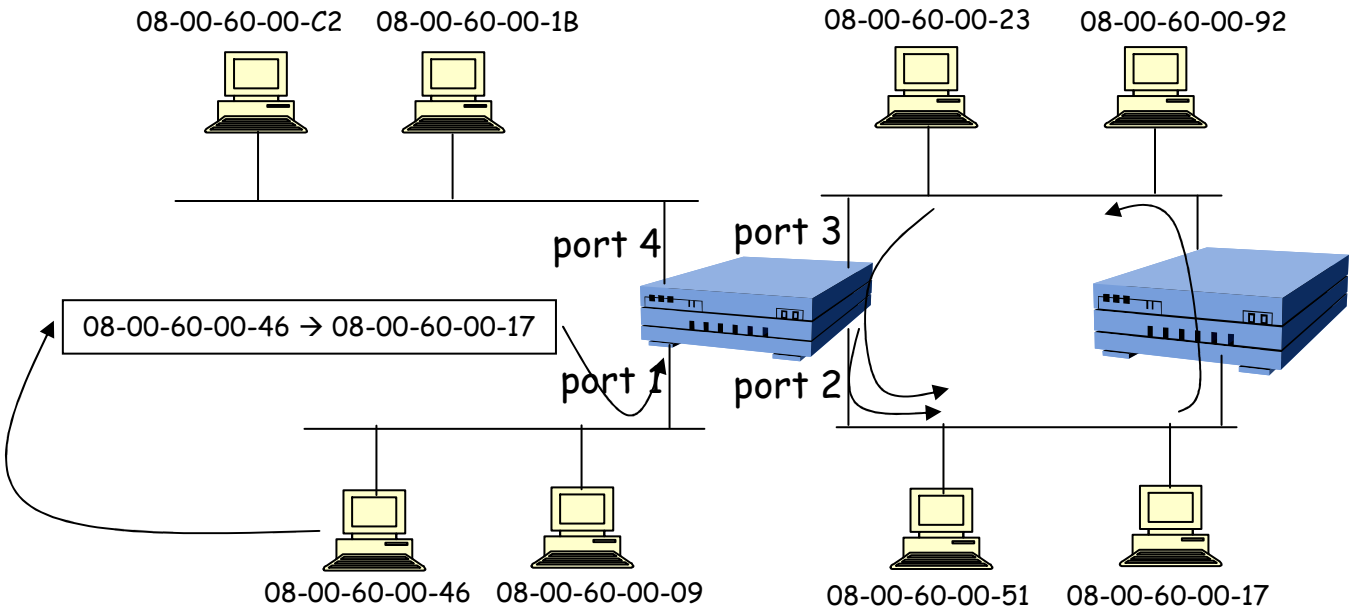
- Breaks up a "broadcast domain" (LAN) into multiple "collision domains" (LAN segments)



- Increased total max throughput
- limitless nodes and geographical coverage
- Can connect different Ethernet types
- Transparent "plug-and-play"

Loops: disaster!

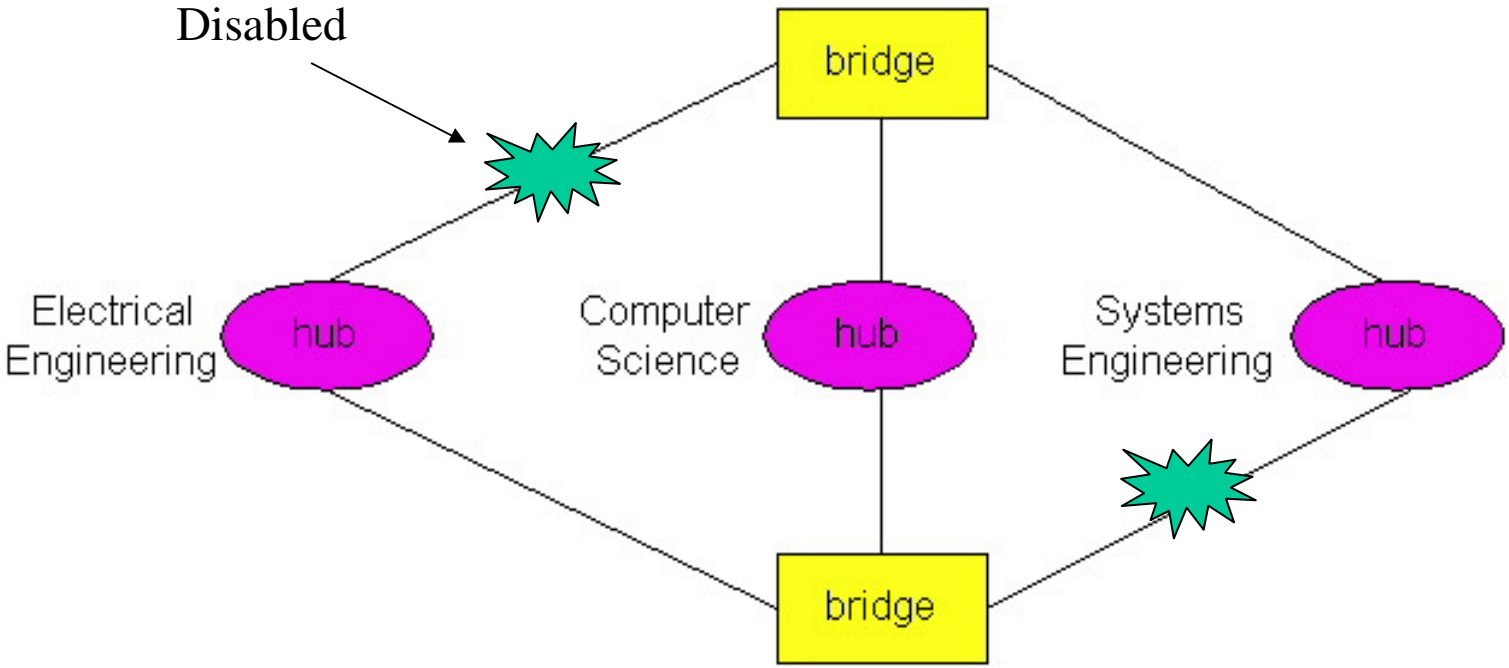
- Inadvertent
- Intentional: for redundancy



- Frame loops forever!
- Learning messed up!

Breaking the loop: Spanning Tree

- Disable set of switch interfaces so that resulting topology is a "tree" that "spans" all LAN segments



Spanning tree protocol (IEEE 802.1d)

- Every bridge has bridge-id
  - bridge-id = 2-byte priority + 6-byte MAC addr
    - **Question: MAC address of bridge??**
- Every port of bridge has
  - port-id = 1-byte priority + 1-byte port-number
  - port-cost = inversely proportional to link speed
- Bridge with lowest bridge-id is root bridge
- On each LAN segment, bridge with lowest path cost to root is designated bridge (use bridge-id and port-id to break ties)
- A bridge forwards frames through a port only if it is a designated bridge for that LAN segment

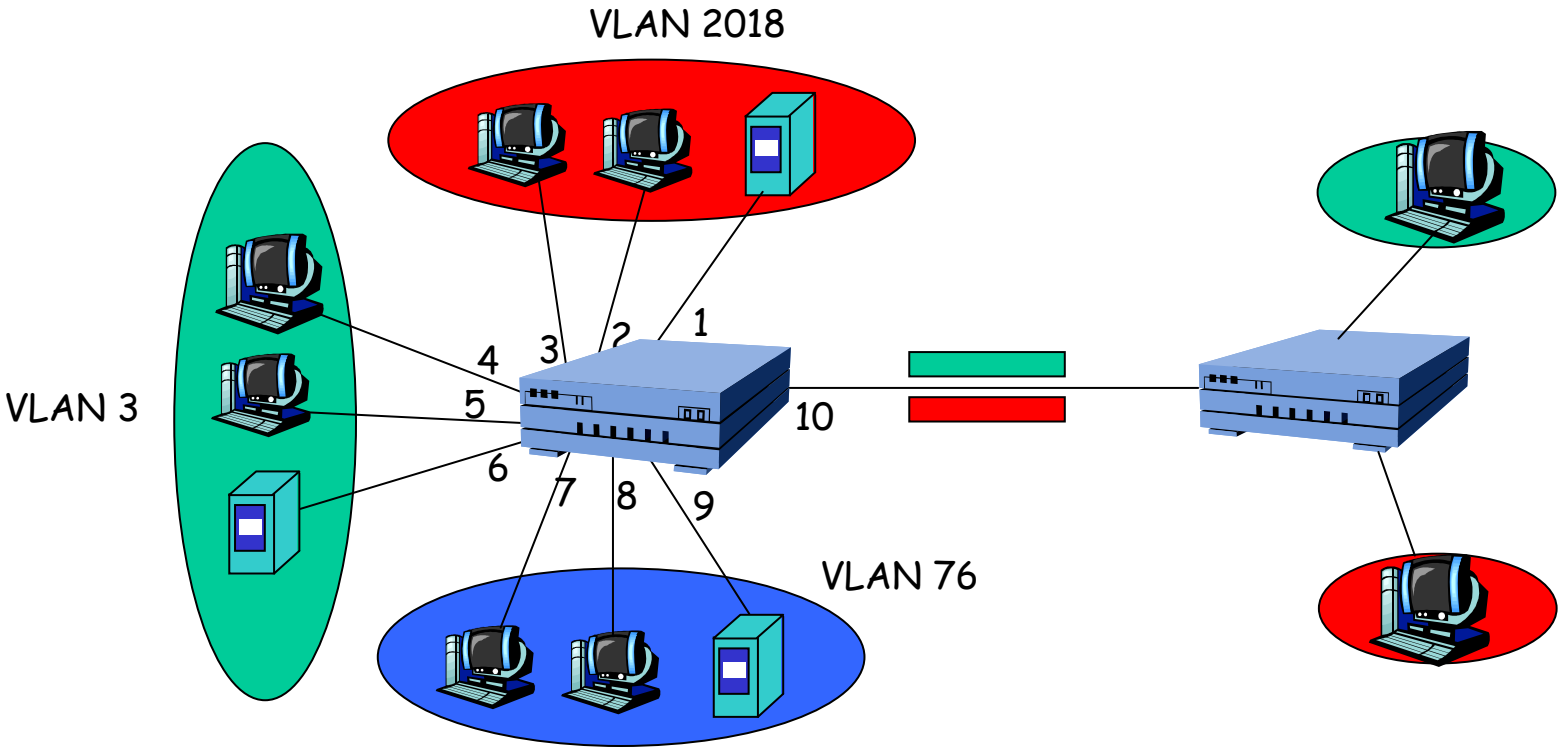
STP terminology

- Port roles:
  - Root port (switch port leading to root)
  - Designated port (LAN port leading to root)
  - Alternate / backup port (anything else)
- Port states:
  - Blocking (no send/rcv, except STP bpdus)
  - Listening (prepare for learning/forwarding)
  - Learning (learn MAC addr but no forwarding)
  - Forwarding (send/rcv frames)
- Can disable STP on port or switch
  - All frames are forwarded
  - BPDUs?

Virtual LANs

- LAN (broadcast domain) grows large
- "departments" or "workgroups" not happy with big broadcast domain
  - Security (eavesdropping)
  - Bandwidth consumed by flooding/multicasting
- Split LAN into multiple broadcast domains
  - Multiple physical LANs?
    - **Too expensive!**
    - **People move all the time!**
- VLAN: logical partition of LAN

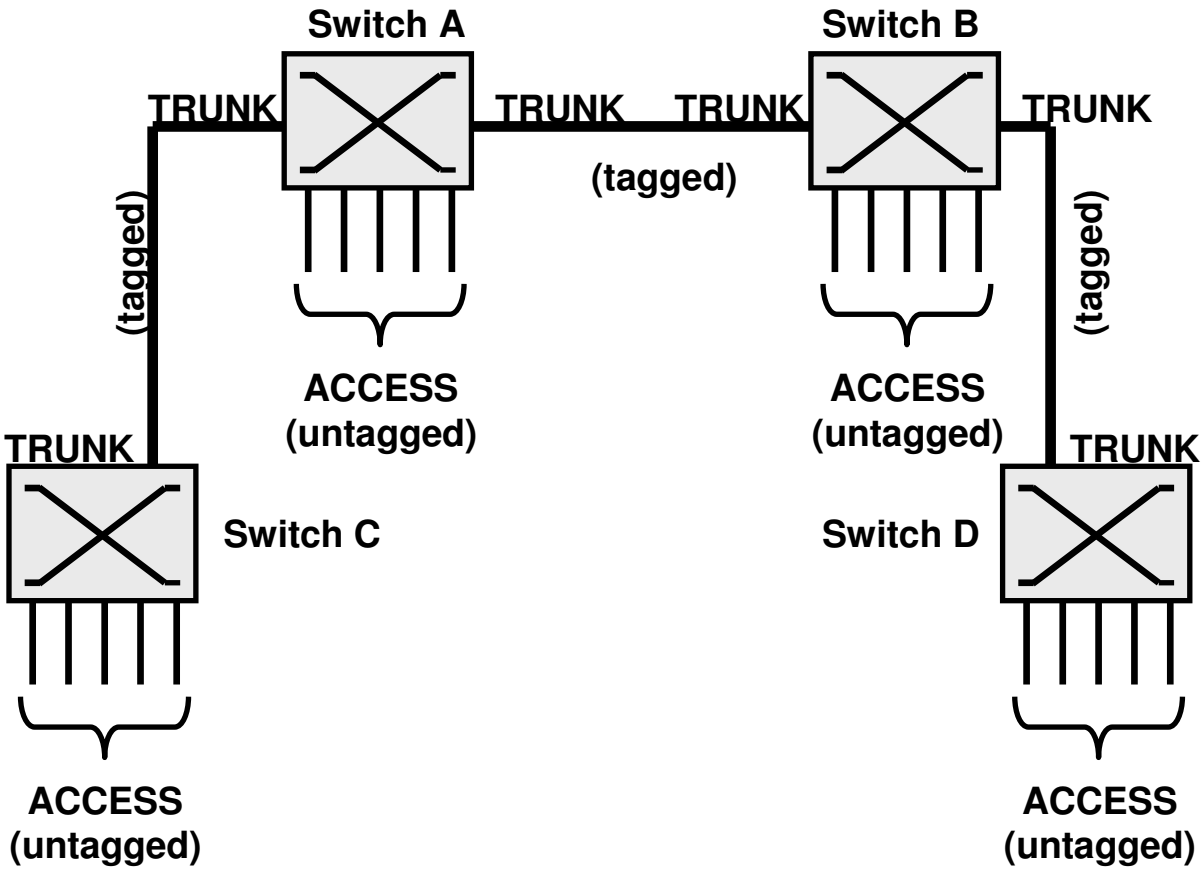
VLAN scenario



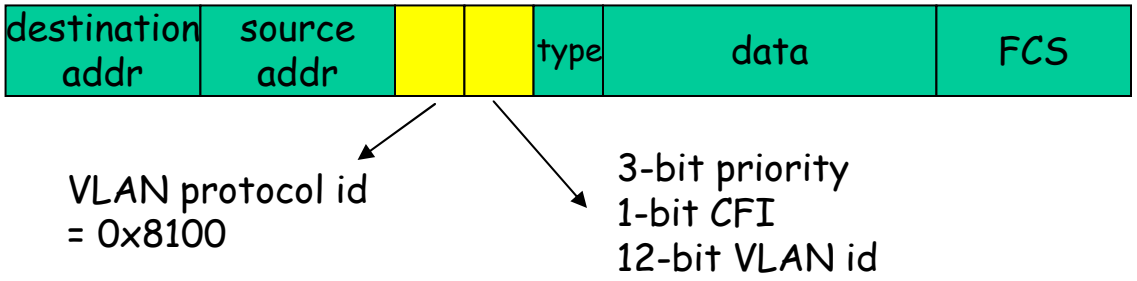
- Assign each switch port to a VLAN
  - Logically split switch into multiple switches
- Assign port to multiple VLANs – "trunk port"

Definizione delle porte

- Access: riceve e trasmette trame *untagged*
- Trunk: riceve e trasmette trame *tagged*



VLANs: IEEE 802.1q



- "Tagged" Ethernet frames contain VLAN-id
- Switch adds/removes tag when forwarding frames between trunk and non-trunk ports
- Complications:
  - Hosts and legacy switches do not understand VLAN tags
  - Tag insertion/removal requires FCS recomputation
  - Frame length increases beyond legacy MTU

Implementation: MAC table

Example MAC adress table

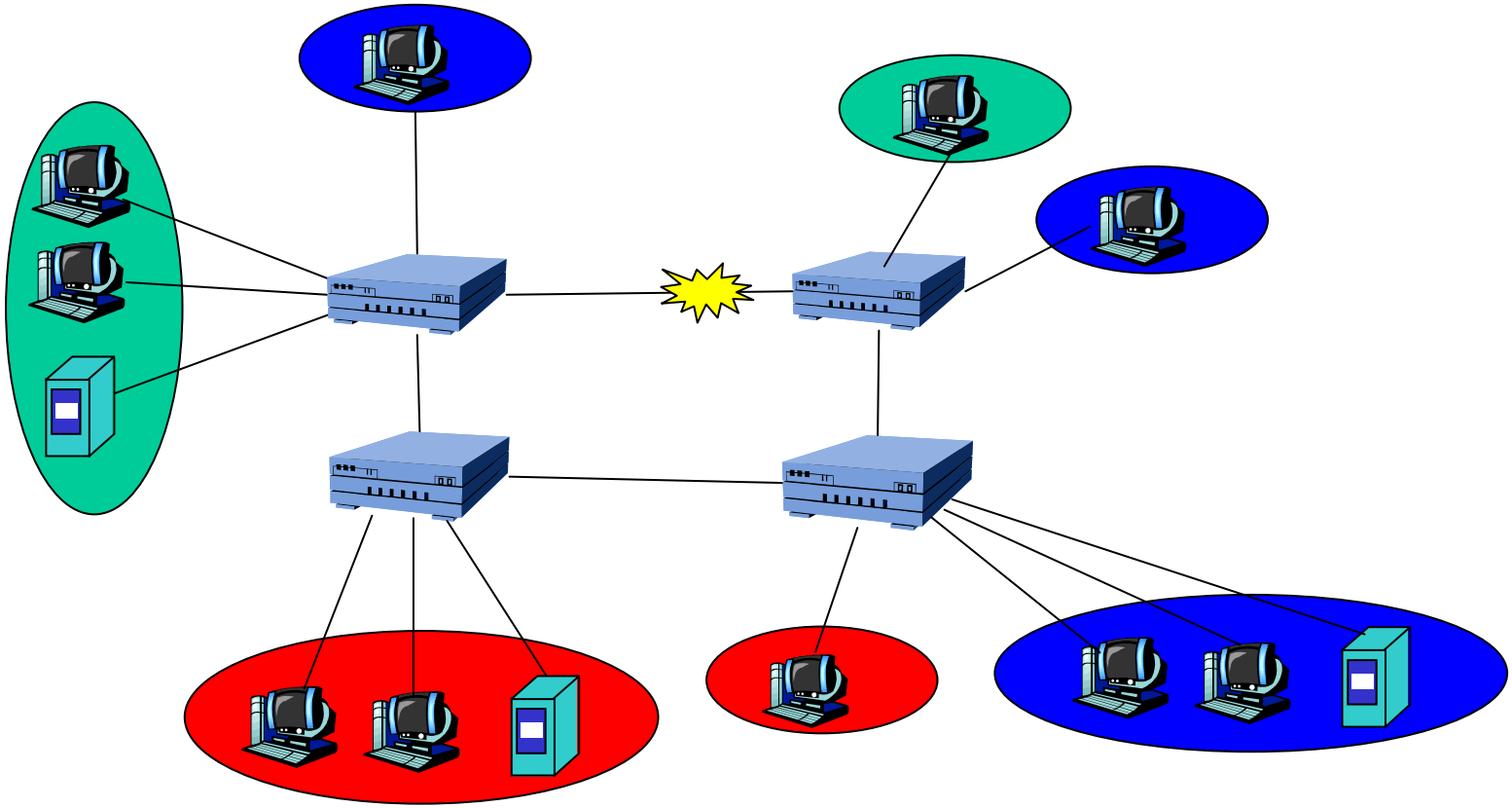
VLAN	MAC address	port
3	08-00-60-00-09	10
3	08-00-60-00-51	4
3	08-00-60-00-17	6
76	08-00-60-00-A3	7
76	08-00-60-00-46	8
76	08-00-60-00-1B	10
2018	08-00-60-00-51	3
2018	08-00-60-00-92	10

port 10: trunk port

same MAC address in multiple VLANs

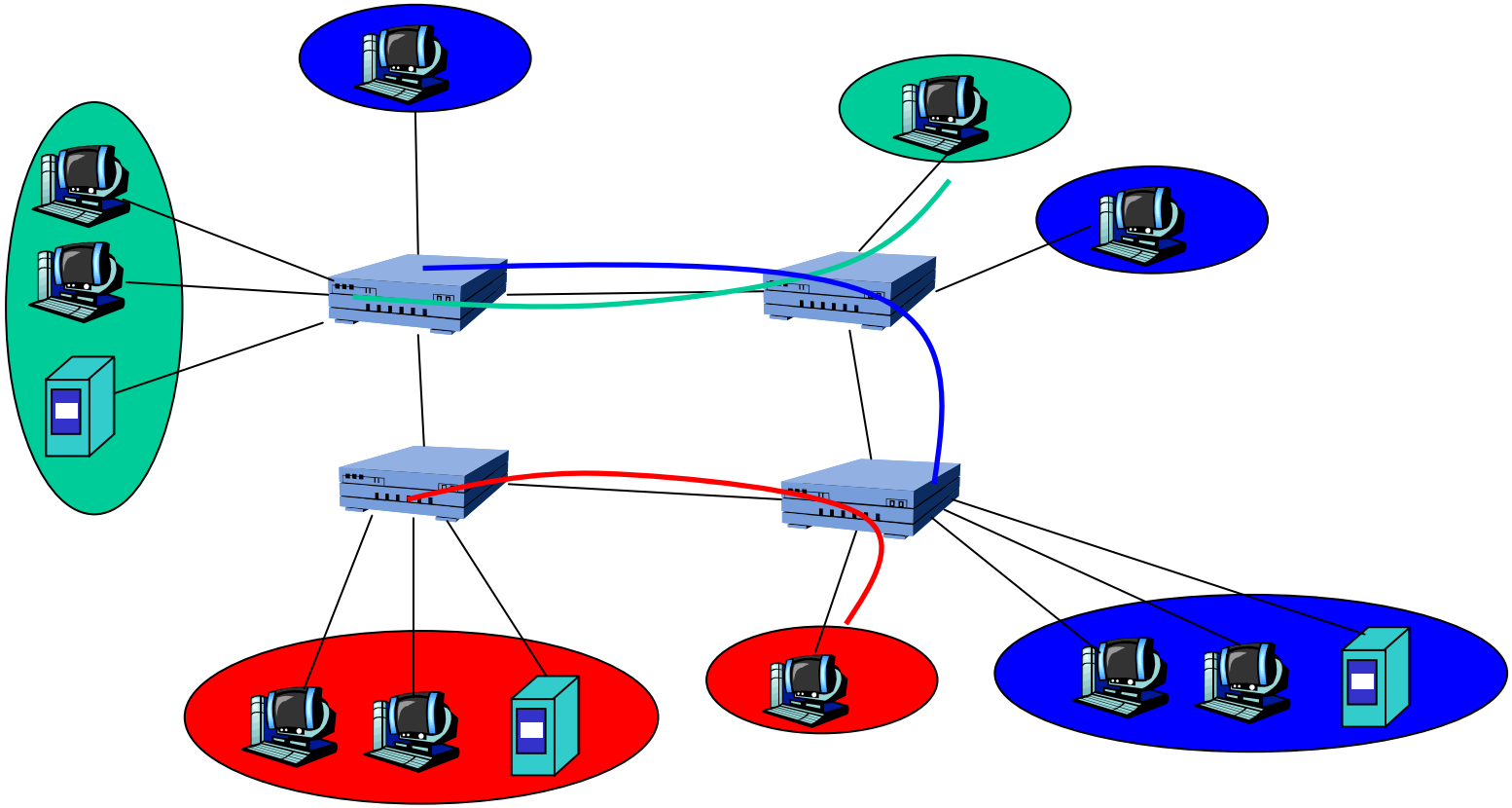
- MAC lookup qualified by VLAN-id
- MAC address can repeat across VLANs

Spanning tree



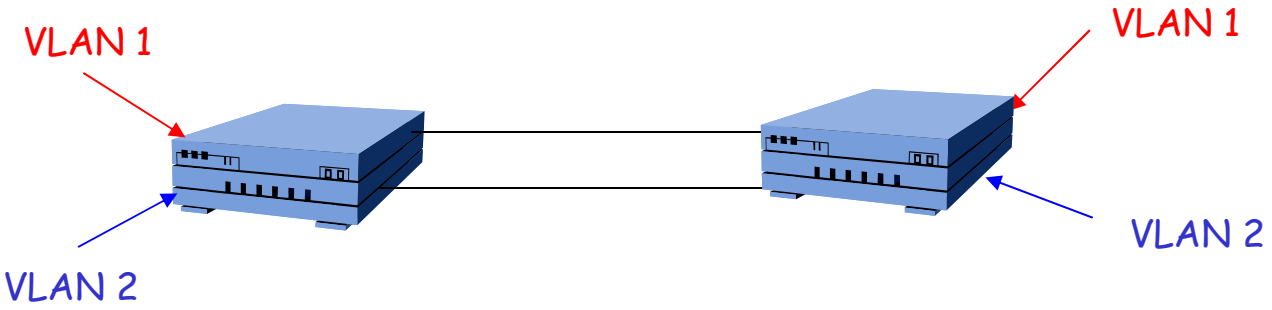
- VLANs share one spanning tree
  - unnecessary wastage of link capacity

Spanning forest (IEEE 802.1s)



- Can create spanning tree “groups”
- Can map one or more VLANs to each stp group

Load balancing



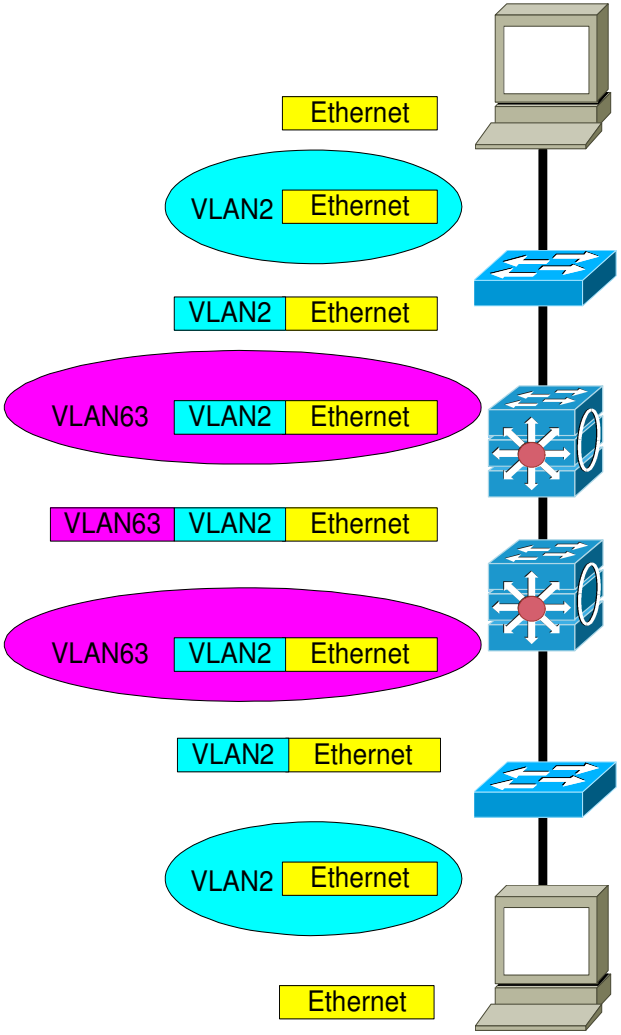
- How to make use of both links (load balance)?
- How to load balance while still maintaining redundancy?

Further discussion topics

- LAN Features:
  - Link aggregation (IEEE 802.3ad)
  - Rapid spanning tree (IEEE 802.1w)
  - VLAN-stacking
- LAN design:
  - Communication between VLANs?
  - How many VLANs in a LAN?
  - Ethernet in MANs?
  - Extending Ethernet across WANs?
- Switch design

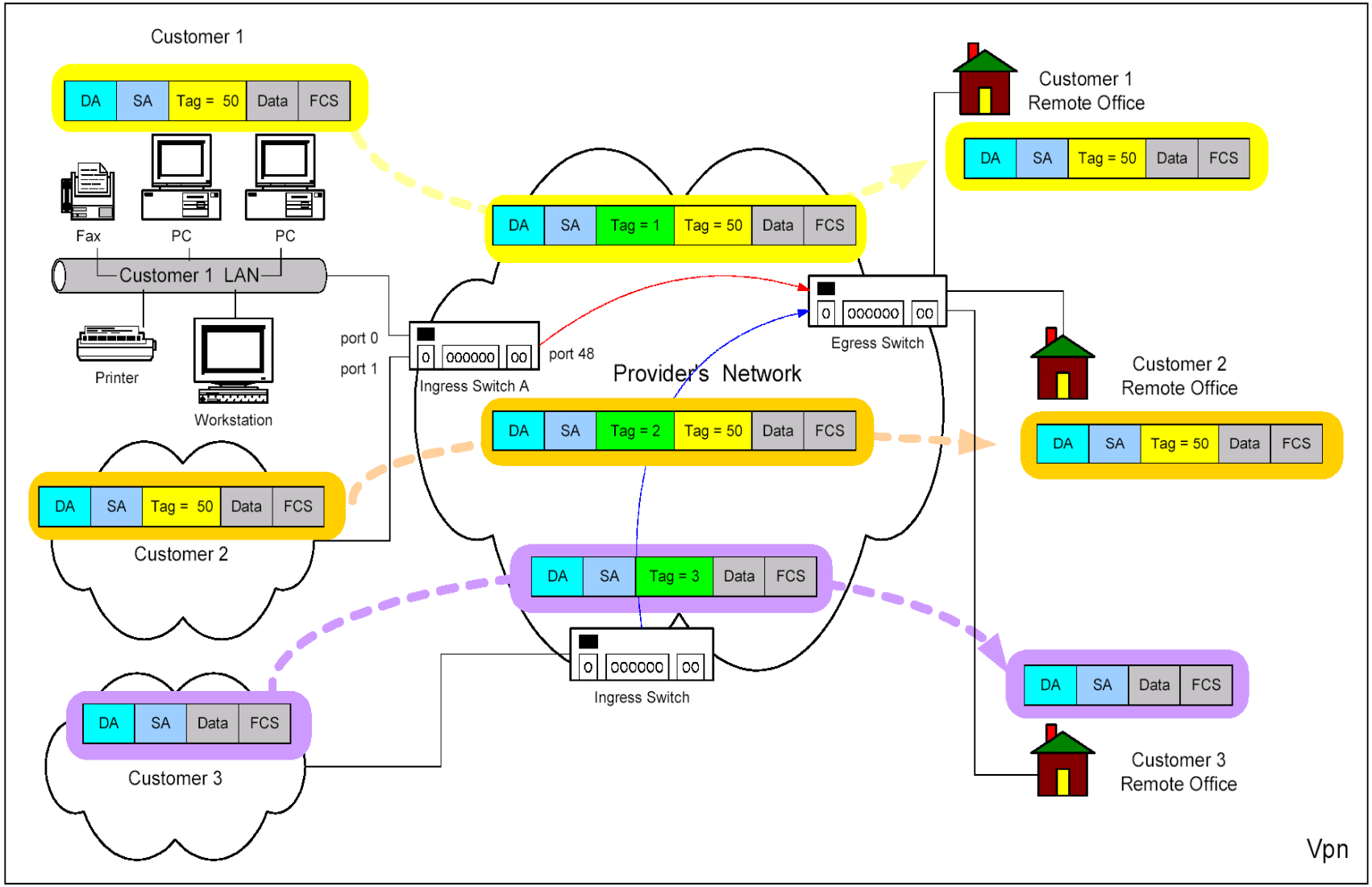
Dot1q-in-Dot1q Stacked VLAN

DA 6 bytes	SA 6 bytes	TPID 2 bytes	802.1Q/p TCI 2 bytes	TPID 2 bytes	802.1Q/p TCI 2 bytes	Len/ Etype 2 bytes	DATA 0-1500 bytes	FCS 4 bytes



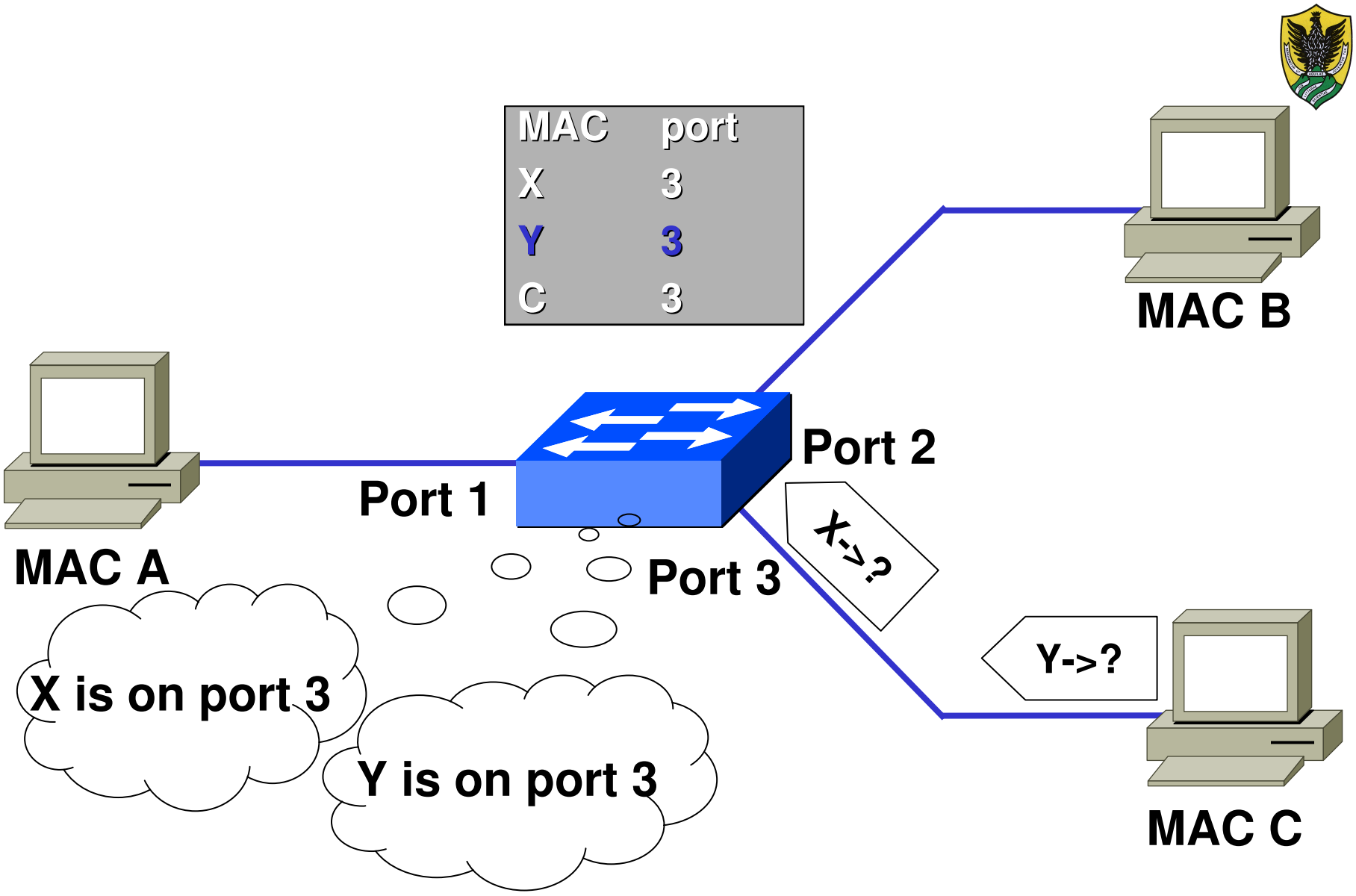
Dot1q-in-Dot1q Stacked VLAN

Stacked VLANs can be seen as VLAN tunneling in which the VLAN identity is tunneled through the Provider Network.

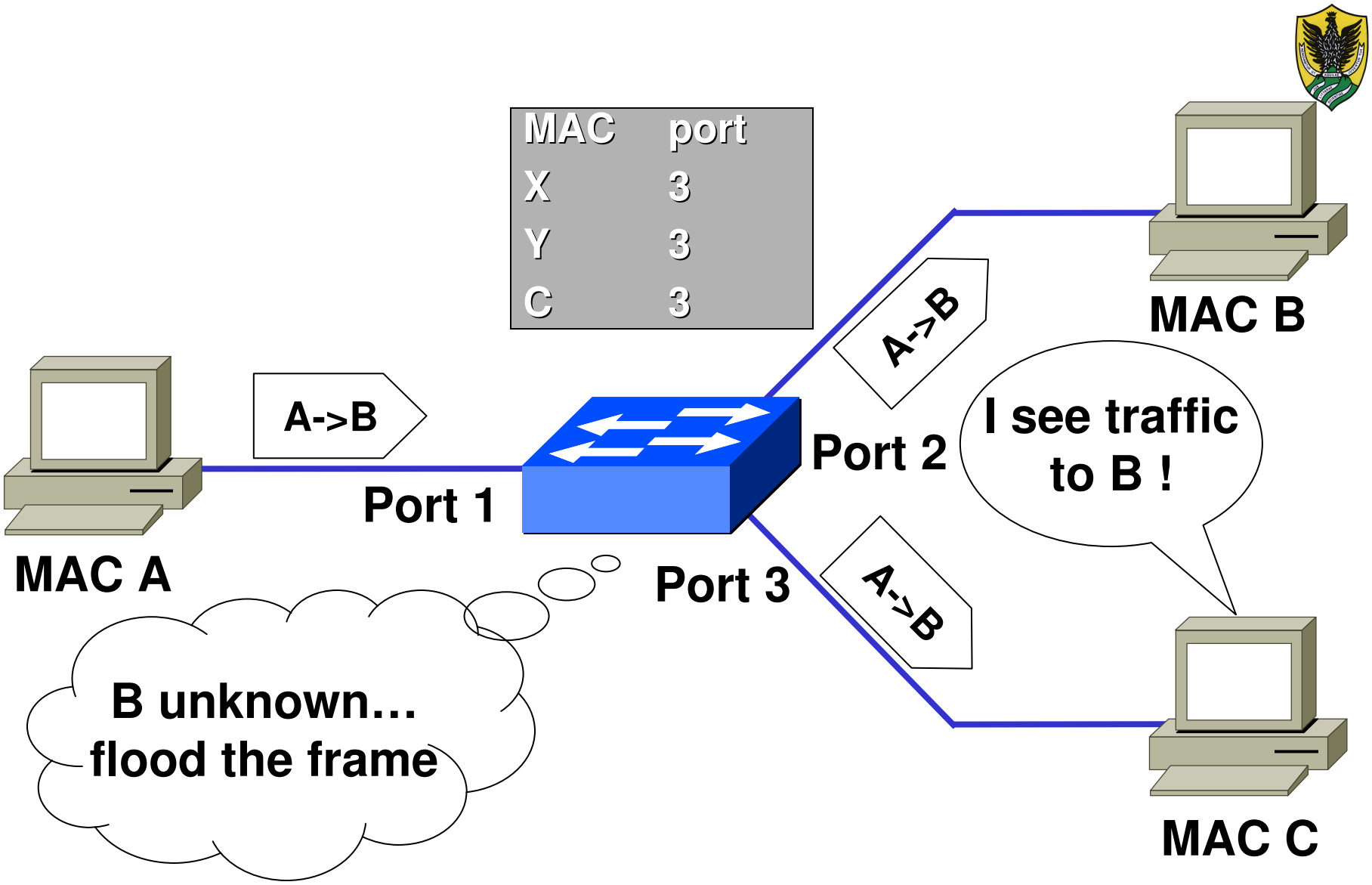


MAC Attacks

CAM Overflow 1/2



CAM Overflow 2/2

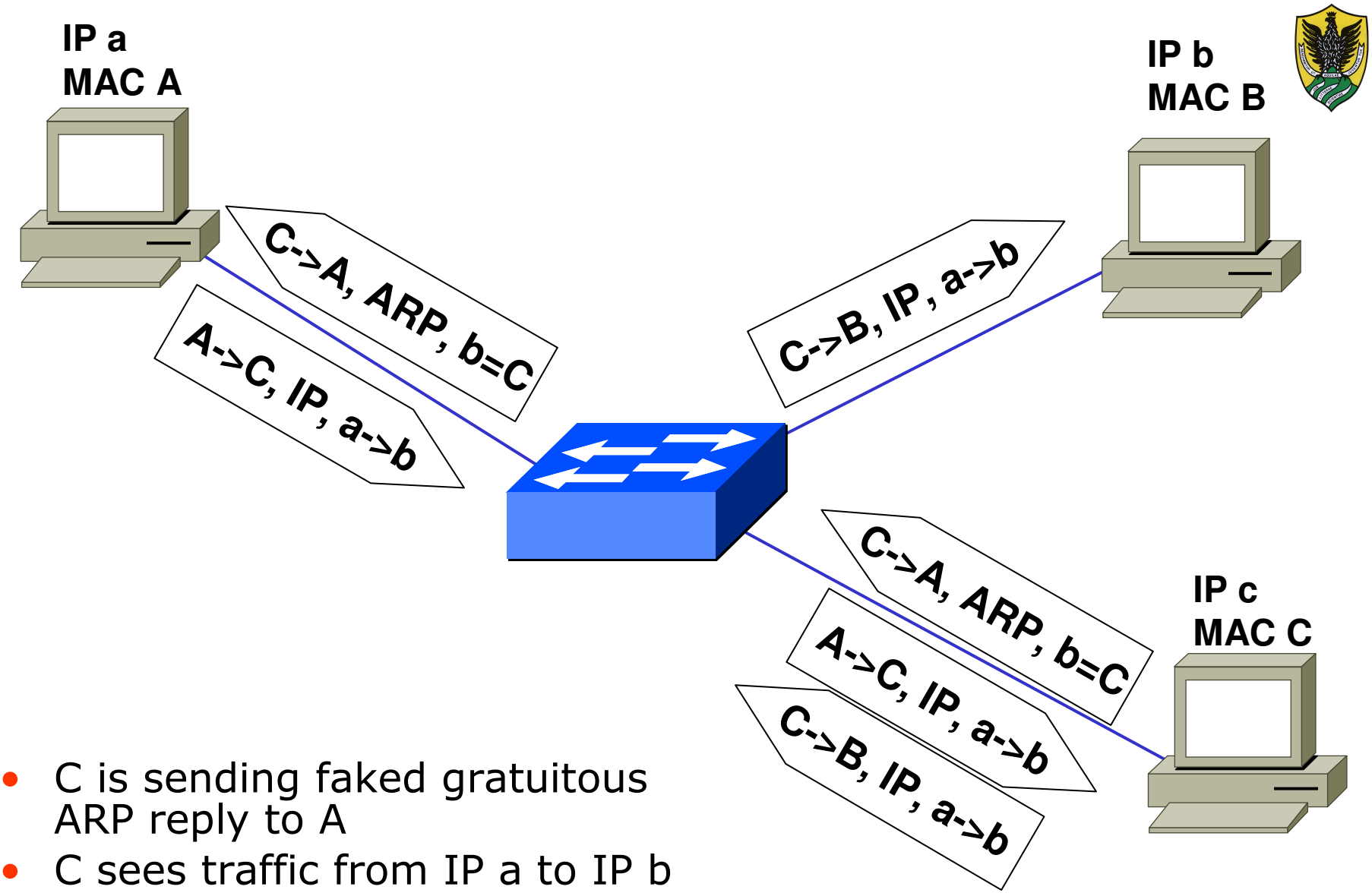


MAC Flooding Attack Mitigation

- Port Security
  - Allows you to specify MAC addresses for each port, or to learn a certain number of MAC addresses per port
  - Upon detection of an invalid MAC block only the offending MAC or just shut down the port
- Smart CAM table
  - Never overwrite existing entries
  - Only time-out inactive entries
  - Active hosts will never be overwritten
- Speak first
  - Deviation from learning bridge: never flood
  - Requires a hosts to send traffic first before receiving

ARP Attacks

ARP Spoofing



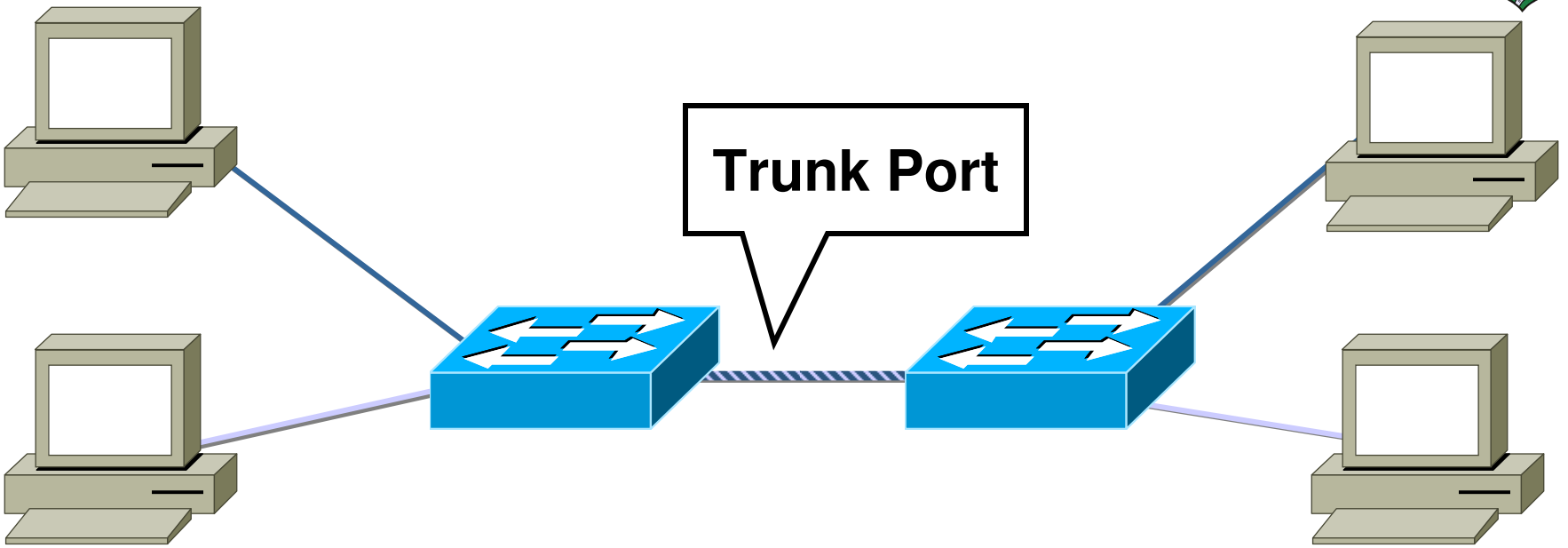
- C is sending faked gratuitous ARP reply to A
- C sees traffic from IP a to IP b

Mitigating ARP Spoofing

- ARP spoofing works only within one VLAN
- static ARP table on critical stations (but dynamic ARP override static ARP on most hosts!)
- ARP ACL: checking ARP packets within a VLAN
  - Either by static definition
  - Or by snooping DHCP for dynamic leases
- No direct communication among a VLAN: private VLAN
  - Spoofed ARP packet cannot reach other hosts

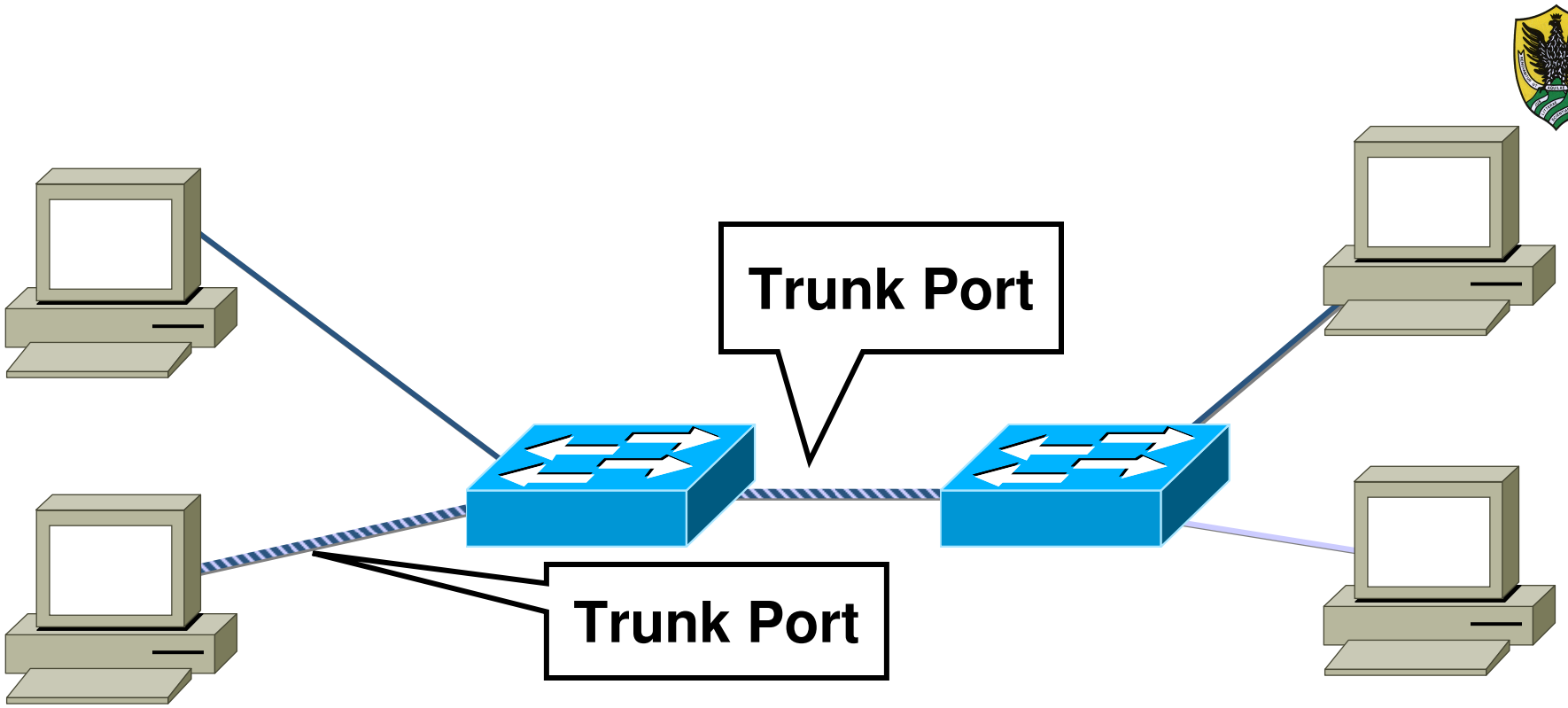
VLAN “Hopping” Attacks

Trunk Port Refresher



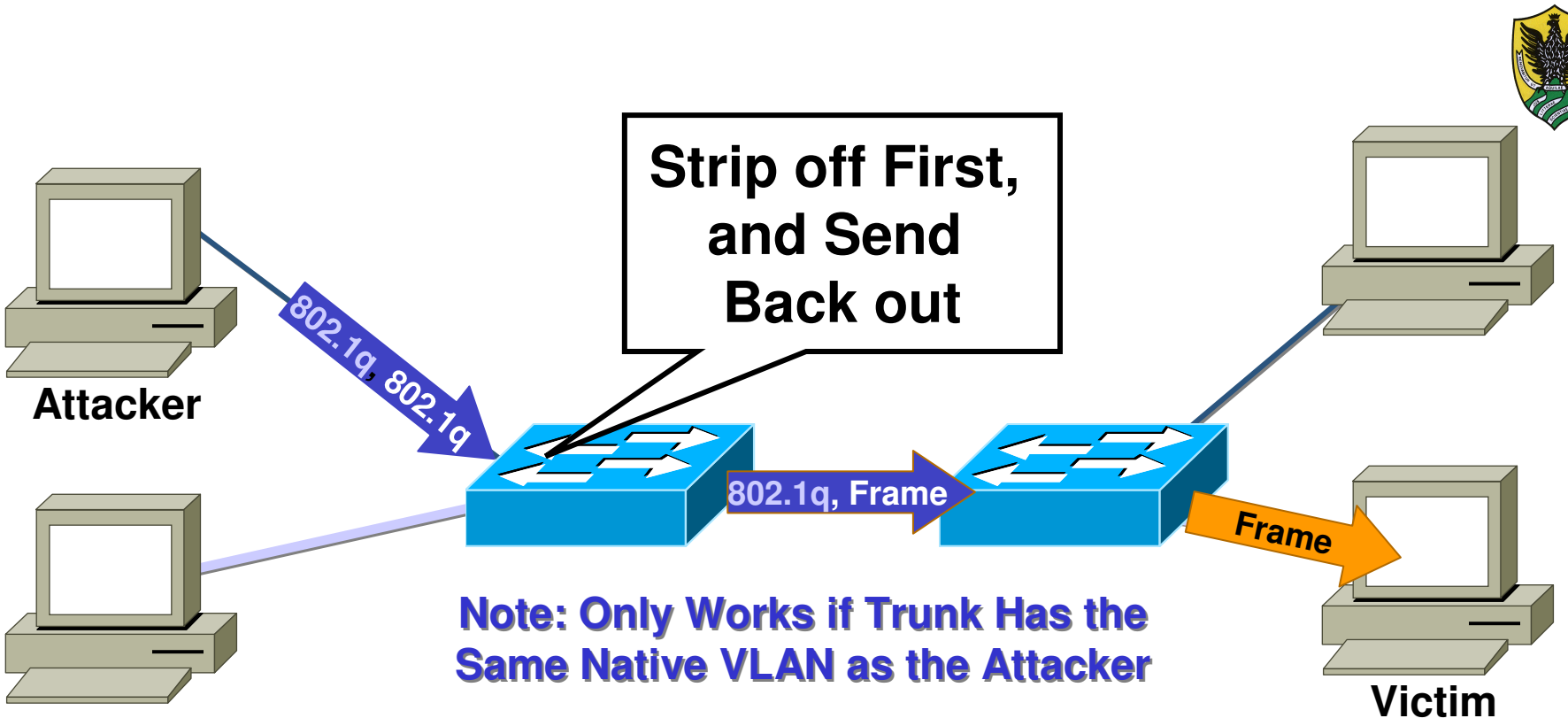
- Trunk ports have access to all VLANs by default
- Used to route traffic for multiple VLANs across the same physical link (generally used between switches)

Basic VLAN Hopping Attack



- A station can spoof as a switch with 802.1Q signaling
- The station is then member of all VLANs
- Requires a trunking favorable setting on the port (the SANS paper is three years old)
  - http://www.sans.org/newlook/resources/IDFAQ/vlan.htm

Double Encapsulated 802.1Q VLAN Hopping Attack



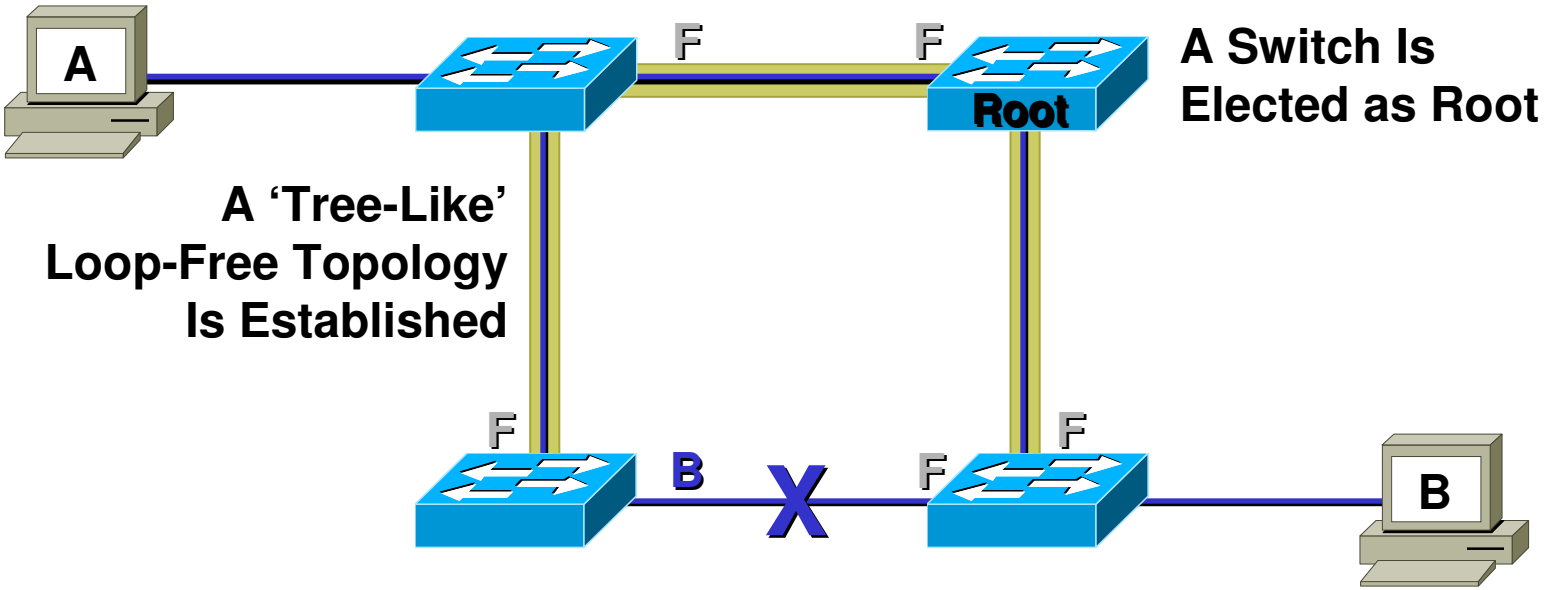
- Send double encapsulated 802.1Q frames
- Switch performs only one level of decapsulation
- Unidirectional traffic only
- Works even if trunk ports are set to off

Mitigation

- Use recent switches
- Disable auto-trunking
- Never put host in the trunk native VLAN
- Put unused ports in an unused VLAN

Spanning Tree Attacks

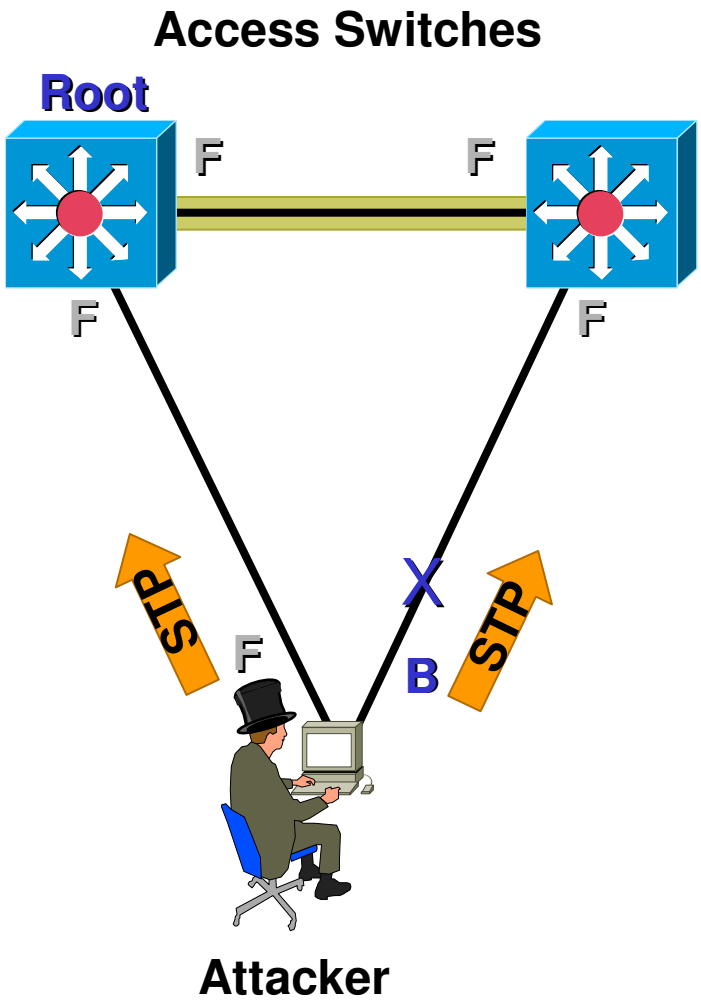
Spanning Tree Basics



Loop-Free Connectivity

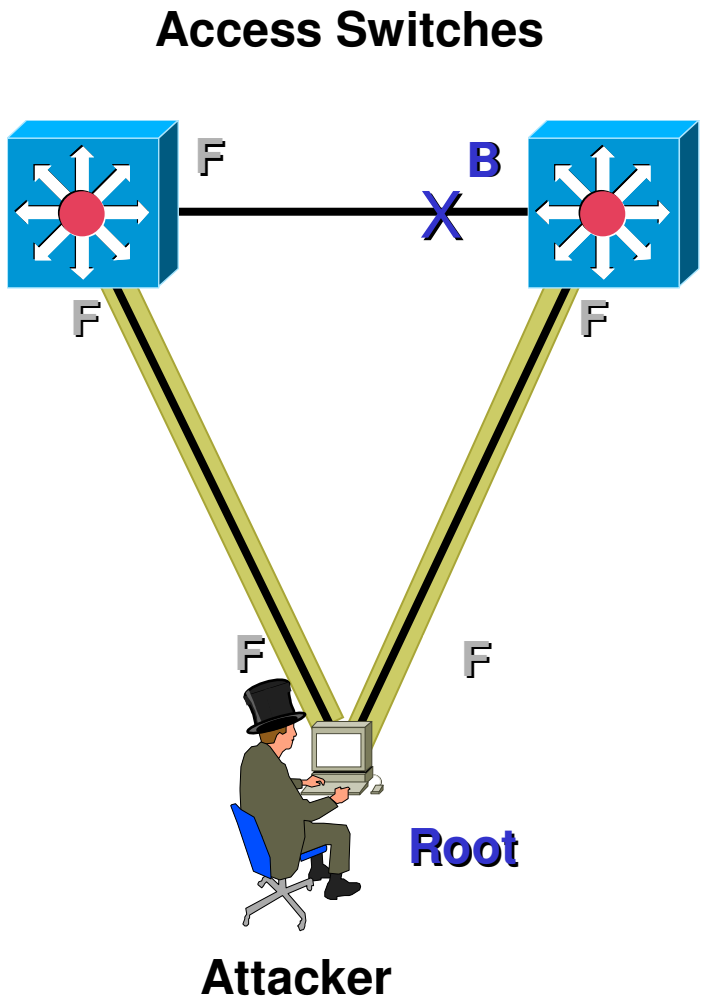
Spanning Tree Attack Example 1/2

- Send BPDU messages from attacker to force spanning tree recalculations
  - Impact likely to be DoS
- Send BPDU messages to become root bridge



Spanning Tree Attack Example 2/2

- Send BPDU messages from attacker to force spanning tree recalculations
  - Impact likely to be DoS
- Send BPDU messages to become root bridge
  - The hacker then sees frames he shouldn't
    - **MITM, DoS, etc. all possible**
    - **Any attack is very sensitive to the original topology, trunking, PVST, etc.**
    - **Requires attacker to be dual homed to two different switches**



STP Attack Mitigation

- Disable STP
  (It is not needed in loop free topologies)
- BPDU Guard
  - Disables ports upon detection of a BPDU message on the port
- Root Guard
  - Disables ports who would become the root bridge due to their BPDU advertisement

