

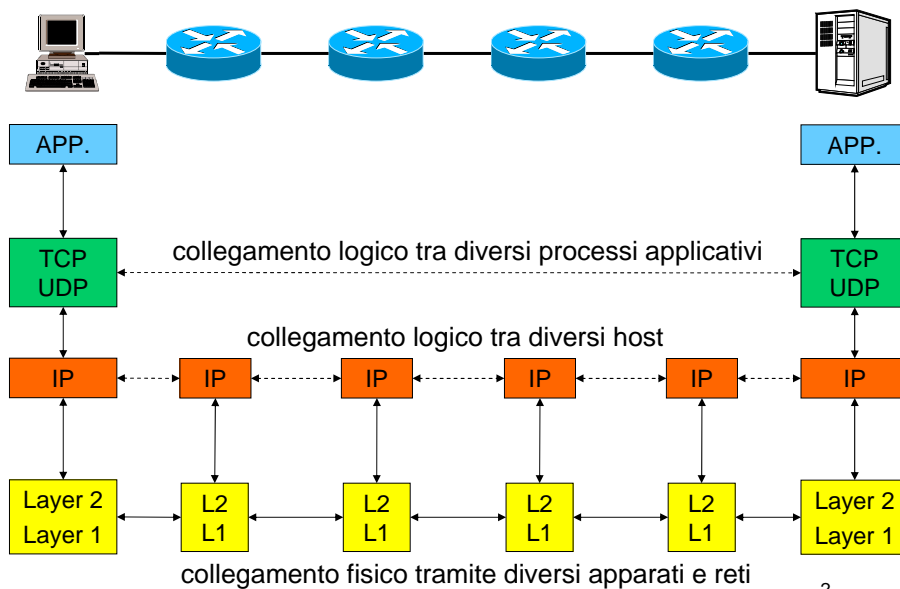


I protocolli UDP e TCP

A.A. 2005/2006

Walter Cerroni

Il livello di trasporto in Internet



Funzioni dello strato di trasporto

- Compito dello strato di trasporto è fornire un servizio di trasporto dati tra i processi applicativi di un host sorgente e quelli di un host destinazione, svincolando gli strati superiori da tutti i problemi di rete
 - realizza una comunicazione **end-to-end**
 - rappresenta l'interfaccia fra gli strati superiori e lo strato di rete
 - l'interazione avviene attraverso un punto di accesso al servizio (**T-SAP**) che negli standard di Internet è chiamato **porta**
 - tipicamente più processi possono utilizzare le funzionalità dello stesso strato di trasporto contemporaneamente (**multiplazione**)
- Analogamente allo strato di rete, può funzionare in modalità **connectionless** e **connection-oriented**
- Protocolli dello strato di trasporto di Internet:
 - **UDP**: modalità **connectionless, non affidabile**
 - **TCP**: modalità **connection-oriented, affidabile**

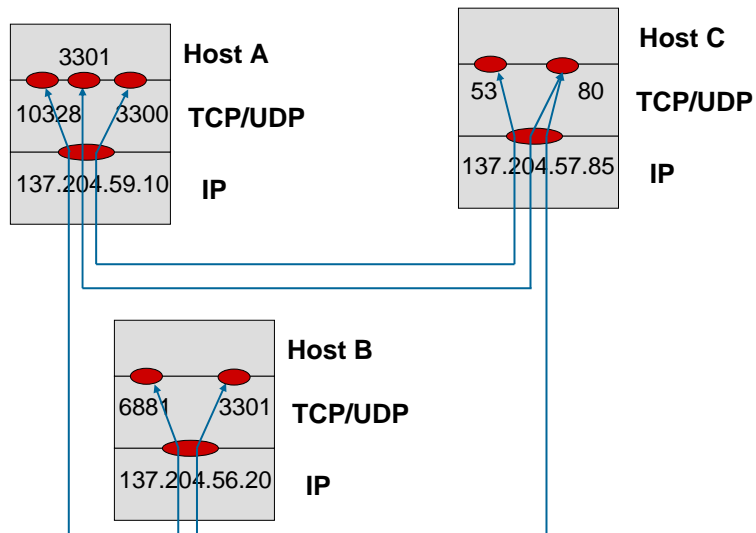
3

Multiplazione/Demultiplazione

- Molteplici processi applicativi in esecuzione sullo stesso host possono aver bisogno contemporaneamente di comunicare tramite lo strato di trasporto
- Il protocollo di trasporto deve poter:
 - raccogliere i dati provenienti da applicazioni diverse e trasmetterli attraverso un unico strato di rete (**multiplexing**)
 - ricevere i dati dallo strato di rete e smistarli correttamente verso le diverse applicazioni (**demultiplexing**)
- **Numero di porta**: è un identificativo che determina univocamente un particolare processo applicativo in esecuzione su un host e che sta utilizzando il protocollo di trasporto
- **Socket**: è l'interfaccia (software) attraverso cui il livello di trasporto scambia dati con le applicazioni

4

Multiplicazione/Demultiplicazione



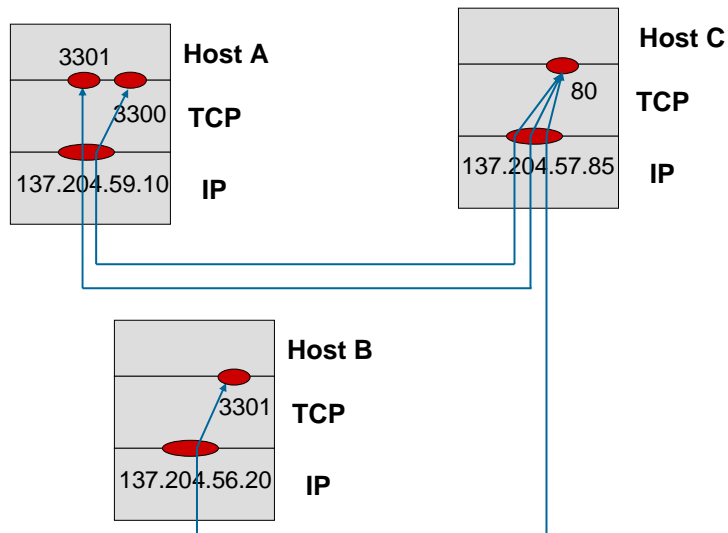
5

L'interfaccia Socket

- Gli standard dei protocolli di trasporto non specificano come questi debbano interagire con gli applicativi
- L'interfaccia fra applicazione e TCP/UDP (**Socket**) dipende dall'implementazione del sistema operativo
 - uso di **primitive di sistema**
- Si dice **Indirizzo della Socket** un numero di porta concatenato ad un indirizzo IP
 - es. 137.204.57.85:80
- Nel trasporto UDP basta conoscere l'indirizzo della socket di destinazione
 - l'indirizzo della socket di origine serve per poter confezionare eventuali risposte
- Nel trasporto TCP sono necessari entrambi
 - la coppia di indirizzi di socket identifica univocamente le connessioni attive

6

Socket TCP



7

Processi client e processi server

- Le comunicazioni fra calcolatori sono originate da processi applicativi che devono scambiare messaggi con altre applicazioni remote
- Perché un calcolatore possa effettivamente utilizzare un messaggio che arriva, occorre che in quel momento vi sia in esecuzione un processo che vada a leggere quel messaggio e sappia cosa farne
- La soluzione più frequente in Internet è l'utilizzo di applicazioni distribuite basate sul **modello client-server**
- Ci sono calcolatori su cui girano **processi Server** che erogano servizi e si aspettano di ricevere richieste di connessione da parte di **processi Client** interessati a tali servizi

8

Modello di servizio

- Il processo Server si predispone a ricevere una richiesta eseguendo una **apertura passiva**
 - UDP: apre una socket e si mette in ascolto sulla relativa porta, in attesa dell'arrivo di una richiesta
 - TCP: apre una socket e si mette in ascolto sulla relativa porta, in attesa dell'arrivo di una richiesta di apertura della connessione
 - il processo server, tipicamente eseguito in background, nel mondo Linux-UNIX è chiamato **demone**
- Quando è interessato ad un determinato servizio, il processo Client esegue una **apertura attiva** tentando di collegarsi al processo server che offre tale servizio
 - UDP: apre una socket inviando direttamente la richiesta al server
 - TCP: apre una socket inviando una richiesta di apertura della connessione
- Il client deve conoscere l'indirizzo IP e il numero di porta usati dal server per potersi collegare alla socket di destinazione

9

Scelta delle porte

- L'indirizzo IP dell'host su cui è in esecuzione il server è:
 - inserito direttamente dall'utente
 - ottenuto tramite traduzione di un nome DNS
- Il numero di porta su cui inviare le richieste al server è scelto secondo una convenzione:
 - tutti i server di un certo tipo devono utilizzare un numero di porta definito a priori tra le cosiddette **well-known port**
 - es.: i server HTTP usano la porta TCP 80
 - l'elenco delle porte note è definito dalla **IANA** ed è reperibile su **www.iana.org** o, su sistemi Linux-UNIX, nel file **/etc/services**
 - server che non utilizzano porte note risultano "nascosti" o raggiungibili solo se se ne pubblica l'URL
 - es.: `http://nascosto.unibo.it:8088`
- Un client, quando apre una socket, non dovrebbe usare una porta nota

10

Well-known port

- Numeri di porta TCP/UDP a 16 bit:

- da **1** a **1023**: porte **well-known**
 - possono essere usati solo dai server in apertura passiva
- da **1024** a **49151**: porte **registrate**
 - sono usati da alcuni servizi ma anche dai client
- da **49152** a **65535**: porte **dinamiche**
 - sono usati dai client

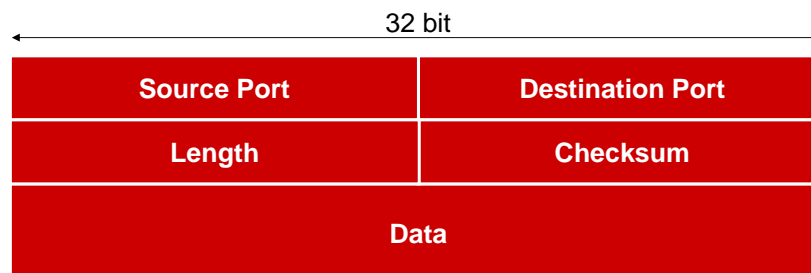
Numero		Nome	Tipo di servizio
21	TCP	FTP	trasferimento file
22	TCP	SSH	terminale virtuale criptato
23	TCP	TELNET	terminale virtuale in chiaro
25	TCP	SMTP	invio posta elettronica
53	UDP	DOMAIN	server DNS
80	TCP	HTTP	server web
110	TCP	POP3	ricezione posta elettronica

11

User Datagram Protocol (UDP)

- Definito in RFC 768

- protocollo di trasporto **non affidabile** di tipo **connectionless**
- concepito per tutte quelle applicazioni per cui una completa gestione delle connessioni non è necessaria
- usato da quelle applicazioni che trasmettono pacchetti singoli, senza necessità di acknowledgment, o richieste brevi
- fa uso di **datagrammi** con un'intestazione di 8 byte



12

Campi dell'intestazione UDP

- **Source/Destination Port:** numero di porta sorgente e destinazione
 - per le operazioni di moltiplicazione/demoltiplicazione
- **Length:** lunghezza in byte del segmento UDP
 - intestazione compresa
- **Checksum:** controllo degli errori su intestazione e dati
 - fa uso di una pseudo-intestazione IP (come il TCP)
- UDP quindi non fa controllo di flusso e di sequenza

- Ad esempio, fanno uso di UDP:
 - DNS (porta 53)
 - RIP (porta 520)
 - diverse applicazioni real-time tramite protocollo RTP

13

Query DNS

The screenshot displays the Wireshark interface with the following data:

No.	Time	Source	Destination	Protocol	Info
3	0.000559	192.168.10.174	137.204.59.1	DNS	Standard query A www.ing2.unibo.it
4	0.023046	137.204.59.1	192.168.10.174	DNS	Standard query response CNAME atrproxy2.unibo.it A 137.204.24.12

Packet 3 Details:

- Frame 3 (77 bytes on wire, 77 bytes captured)
- Ethernet II, Src: 00:0d:56:0a:a0:cb, Dst: 00:b0:d0:ec:46:62
- Internet Protocol, Src Addr: 192.168.10.174 (192.168.10.174), Dst Addr: 137.204.59.1 (137.204.59.1)
- User Datagram Protocol, Src Port: 1031 (1031), Dst Port: 53 (53)
 - Source port: 1031 (1031)
 - Destination port: 53 (53)
 - Length: 43
 - Checksum: 0x9127 (correct)
- Domain Name System (query)

Packet 4 Details:

- Frame 4 (248 bytes on wire, 248 bytes captured)
- Ethernet II, Src: 00:b0:d0:ec:46:62, Dst: 00:0d:56:0a:a0:cb
- Internet Protocol, Src Addr: 137.204.59.1 (137.204.59.1), Dst Addr: 192.168.10.174 (192.168.10.174)
- User Datagram Protocol, Src Port: 53 (53), Dst Port: 1031 (1031)
 - Source port: 53 (53)
 - Destination port: 1031 (1031)
 - Length: 214
 - Checksum: 0x80ee (correct)
- Domain Name System (response)

14

Voice over IP (VoIP)

The screenshot shows the Wireshark interface with a capture of VoIP traffic. The main packet list shows several RTP packets from source 10.1.3.143 to destination 10.1.6.18. Two detail windows are open:

- Packet 35:** Frame 35 (294 bytes on wire, 294 bytes captured). Ethernet II, Src: 00:04:76:22:20:17, Dst: 00:d0:50:10:01:66. Internet Protocol, Src Addr: 10.1.3.143 (10.1.3.143), Dst Addr: 10.1.6.18 (10.1.6.18). User Datagram Protocol, Src Port: 5000 (5000), Dst Port: 2006 (2006). Real-Time Transport Protocol, Length: 260, Checksum: 0x5251 (correct).
- Packet 40:** Frame 40 (294 bytes on wire, 294 bytes captured). Ethernet II, Src: 00:08:21:91:64:60, Dst: 00:04:76:22:20:17. Internet Protocol, Src Addr: 10.1.6.18 (10.1.6.18), Dst Addr: 10.1.3.143 (10.1.3.143). User Datagram Protocol, Src Port: 2006 (2006), Dst Port: 5000 (5000). Real-Time Transport Protocol, Length: 260, Checksum: 0x748f (correct).

15

Transmission Control Protocol (TCP)

- Definito in RFC 793, estensioni in RFC 1323
- Ha lo scopo di realizzare una comunicazione **affidabile** di tipo **full-duplex** fra processi applicativi di due host, in modalità **connection-oriented** e con **controllo di flusso**
- E' progettato assumendo che il livello inferiore sia in grado di fornire solamente un semplice ed inaffidabile servizio di trasferimento dei pacchetti di tipo connectionless (esattamente quello che fa IP)
- Usa la modalità connection-oriented: deve prevedere tutte le procedure per
 - instaurare una connessione
 - controllarne il corretto andamento
 - terminare la connessione

16

Funzioni del TCP

- **Affidabilità del collegamento:** il TCP garantisce la completa correttezza nella consegna dei dati, finché esiste connettività a livello di rete, tramite:
 - **numerazione sequenziale** dei dati, prendendo come unità di riferimento il byte
 - conferma esplicita della ricezione di ogni blocco di byte da parte del ricevitore (**acknowledgment**)
 - ritrasmissione dei dati di cui non viene confermata la ricezione
- **Controllo di flusso:** prevede un meccanismo a finestra che permette al ricevitore di regolare il flusso dei dati inviati dal trasmettitore
- **Controllo dell'errore:** per il riconoscimento degli errori di trasmissione viene effettuato un controllo mediante checksum a 16 bit su tutto il contenuto informativo

17

Il segmento TCP

- TCP incapsula i dati delle applicazioni in unità informative chiamati **segmenti**
- Il segmento TCP prevede
 - un'intestazione standard di 20 byte
 - un'eventuale aggiunta all'intestazione di dimensione variabile per negoziare delle opzioni
 - un payload di dimensione variabile (anche nulla) contenente i dati di applicazione
- Il segmento TCP ha una dimensione massima detta **Maximum Segment Size (MSS)**
 - MSS corrisponde alla massima dimensione del blocco dati di applicazione che può essere contenuto nel segmento

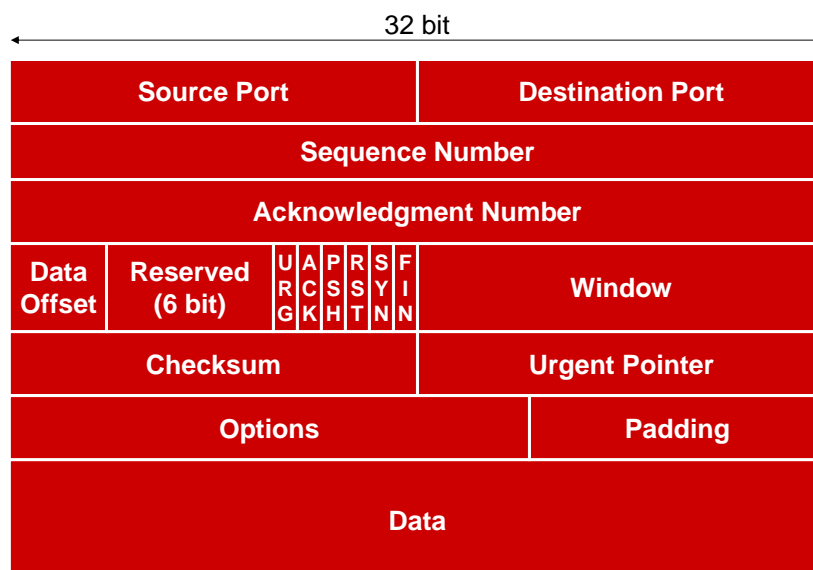
18

Dimensioni del segmento TCP

- MSS deve:
 - essere non superiore alla massima dimensione del payload IP meno un header TCP
 - $65535 - 20 - 20 = 65495$ byte
 - rispettare i limiti imposti alle dimensioni dei pacchetti dalle reti che bisogna attraversare (**Maximum Transmission Unit** – MTU)
 - un tipico valore per MTU sono i 1500 byte imposti da Ethernet
- MSS dipende dall'implementazione
 - normalmente **MSS = MTU – 20 – 20** (parametro configurabile)
- In generale non è possibile conoscere la MTU di ogni rete intermedia che verrà attraversata dai segmenti
 - la rete adotta un meccanismo di frammentazione dei pacchetti
 - questo può condurre a inefficienza
 - è stato definito un algoritmo detto **Path MTU discovery** (RFC 1191) basato sul bit DF e sul messaggio ICMP relativo

19

Formato del segmento TCP



20

Campi dell'intestazione TCP

- **Source/Destination Port:** numero di porta sorgente e destinazione
- **Sequence Number:** numero di sequenza del primo byte contenuto nel segmento; se è presente il bit SYN, questo è il numero di sequenza iniziale su cui sincronizzarsi
- **Acknowledgment Number:** se il bit ACK è a 1, questo è il numero di sequenza del blocco di dati che ci si aspetta di ricevere
- **Data offset:** numero di parole di 32 bit dell'intestazione TCP; indica dove iniziano i dati
- **Reserved:** sei bit riservati per uso futuro; devono essere posti a zero

21

Campi dell'intestazione TCP

- **Control bit:** sono 6 flag
 - **URG** posto a 1 se si deve considerare il campo Urgent Pointer
 - **ACK** posto a 1 se si deve considerare il campo Acknowledgment Number
 - **PSH** posto a 1 indica la funzione di push, per la consegna immediata delle informazioni
 - **RST** posto a 1 per resettare la connessione e rifiutare un segmento o un tentativo di connessione non validi
 - **SYN** posto a 1 per stabilire la connessione e per sincronizzare i numeri di sequenza
 - **FIN** posto a 1 per indicare la fine dei dati da trasmettere e chiudere la connessione in una direzione
- **Window:** dimensione della finestra in ricezione per il controllo di flusso, cioè il numero di byte che il ricevitore è disposto a ricevere a partire dal numero di sequenza contenuto nel campo Acknowledgment Number

22

Campi dell'intestazione TCP

- **Checksum**: controllo d'errore su intestazione e dati
 - effettuata su blocchi da 16 bit
 - si include una pseudo-intestazione IP

Source Address		
Destination Address		
0 0 0 0 0 0 0	Protocol = 6	TCP Length

- **Urgent Pointer**: contiene un puntatore a dati urgenti eventualmente presenti nel pacchetto (es. per abortire un programma remoto in esecuzione); ha senso se il bit URG è posto ad 1
- **Options**: contiene eventuali opzioni per la connessione
- **Padding**: bit aggiuntivi per fare in modo che l'intestazione sia multipla di 32 bit

23

Controllo di flusso

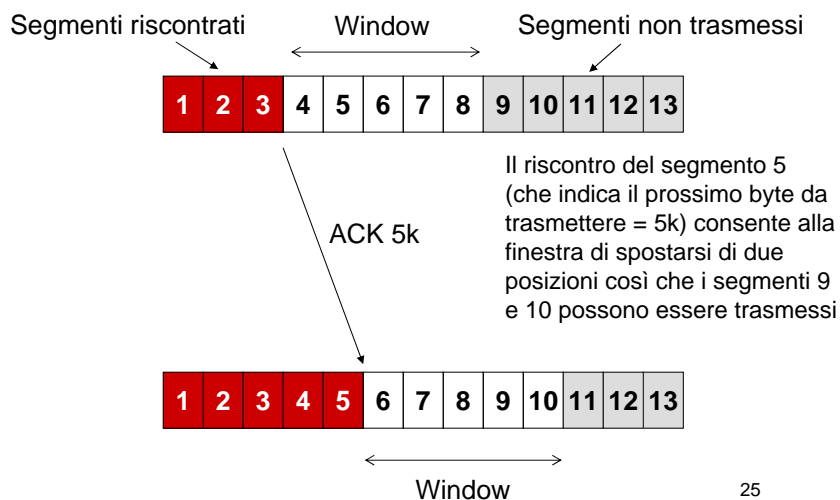
- Le velocità di elaborazione di mittente e destinatario possono essere molto diverse
 - il mittente non deve saturare il destinatario
- Si utilizza un meccanismo a **finestra scorrevole**
 - la finestra indica la quantità di informazioni che si possono trasmettere senza avere riscontro da parte del destinatario
 - deve essere dimensionata in modo congruente con le memorie di trasmissione e ricezione
 - il mittente conosce le dimensioni della propria memoria ma non conosce quelle della memoria di ricezione del destinatario
- Il destinatario deve comunicare al mittente le dimensioni della sua memoria di ricezione
 - nell'intestazione del pacchetto TCP è contenuto il campo Window

24

Meccanismo a finestra scorrevole

Dimensione della finestra = 5 kB

Dimensione del segmento = 1 kB



25

Numerazione dei segmenti TCP

- Per avere la massima flessibilità si sceglie di assegnare un numero non ai segmenti ma ai singoli byte trasportati nei segmenti
 - i dati trasportati sono pensati come un unico flusso (**stream**) di byte
 - si comincia a numerare da un numero N scelto all'atto dell'apertura della connessione
 - il campo Sequence Number individua il primo byte del segmento
- La conferma di corretta ricezione viene data mettendo nel campo Acknowledgment Number il numero del byte successivo all'ultimo ricevuto
 - prossimo byte che ci si aspetta di ricevere

26

Numerazione dei segmenti TCP

- La rete presente tra mittente e destinatario non è un canale sequenziale
 - possono facilmente esserci segmenti ritardati o duplicati che possono compromettere l'integrità dei dati trasmessi
- I numeri di sequenza (codificati con 32 bit) possono essere riutilizzati solo se si è sicuri che non esistano più in rete vecchi segmenti con gli stessi numeri
 - massimo tempo di vita dei segmenti (**Maximum Segment Lifetime** – MSL) legato al TTL di IP
- All'apertura della connessione si deve scegliere il numero di sequenza iniziale (**Initial Sequence Number** – ISN)
 - numero variabile legato al valore di un contatore
 - ISN deve essere concordato fra i due host che aprono la connessione (sincronizzazione)

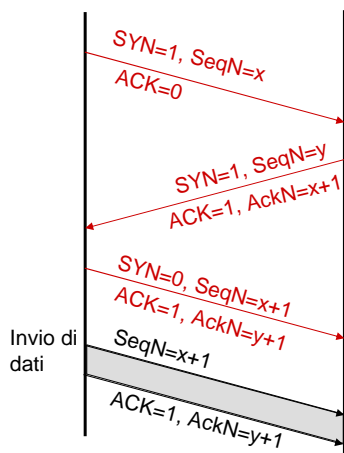
27

Porte e connessioni TCP

- Per essere affidabile ed eseguire il controllo di flusso, TCP è orientato alla connessione
- Il numero di porta concatenato con l'indirizzo IP dell'host costituisce un **end-point** di una connessione
 - es.: 137.204.57.85:80
- Una **connessione** è univocamente determinata dall'associazione di due end-point
 - es.: 137.204.57.85:80 \leftrightarrow 192.168.10.99:10364
- Le connessioni TCP sono
 - full-duplex
 - punto-punto (non viene gestito il multicast)
 - end-to-end
- Un singolo end-point può essere condiviso tra più connessioni sulla stessa macchina (multiplexing)

28

Apertura della connessione TCP



- L'apertura della connessione non è banale perché la rete può perdere, duplicare o ritardare i pacchetti
- TCP usa uno schema detto **Three Way Handshake** che risulta essere molto robusto
 - sincronizzazione di entrambi i numeri di sequenza
- Il primo pacchetto dati ha numero di sequenza uguale all'ACK precedente
 - ACK non occupa spazio di numerazione

29

Apertura della connessione TCP

Source	Destination	Protocol	Info
192.168.10.174	137.204.24.12	TCP	1043 > 80 [SYN] Seq=27982673 Ack=0 win=64240 Len=0
137.204.24.12	192.168.10.174	TCP	80 > 1043 [SYN, ACK] Seq=1251642841 Ack=27982674 win=5840 Len=0
192.168.10.174	137.204.24.12	TCP	1043 > 80 [ACK] Seq=27982674 Ack=1251642842 win=64240 Len=0
192.168.10.174	137.204.24.12	HTTP	GET / HTTP/1.1

5 0.023889 192.168.10.174 137.204.24.12 TCP...

- Frame 5 (62 bytes on wire, 62 bytes captured)
- Ethernet II, Src: 00:0d:56:0a:a0:cb, Dst: 02:00:c0:00:00:00
- Internet Protocol, Src Addr: 192.168.10.174, Dst Addr: 137.204.24.12
- Transmission Control Protocol, Src Port: 1043 (1043), Dst Port: 80 (80)
 - sequence number: 27982673
 - Header length: 28 bytes
 - Flags: 0x0002 (SYN)
 - window size: 64240
 - Checksum: 0x19a1 (correct)
 - Options: (8 bytes)
 - Maximum segment size: 1460 bytes
 - NOP
 - NOP
 - SACK permitted

8 0.027329 192.168.10.174 137.204.24.12 HTTP GET / HTTP/1.1

- Frame 8 (668 bytes on wire, 668 bytes captured)
- Ethernet II, Src: 00:0d:56:0a:a0:cb, Dst: 00:50:ba:c6:fa:e6
- Internet Protocol, Src Addr: 192.168.10.174 (192.168.10.17), Dst Addr: 137.204.24.12
- Transmission Control Protocol, Src Port: 1043 (1043), Dst Port: 80 (80)
 - sequence number: 27982674
 - Next sequence number: 27983288
 - Acknowledgement number: 1251642842
 - Header length: 20 bytes
 - Flags: 0x0018 (PSH, ACK)
 - window size: 64240
 - Checksum: 0x6faf (incorrect, should be 0xf9ca)
- Hypertext Transfer Protocol

30

Rifiuto di apertura della connessione TCP

```

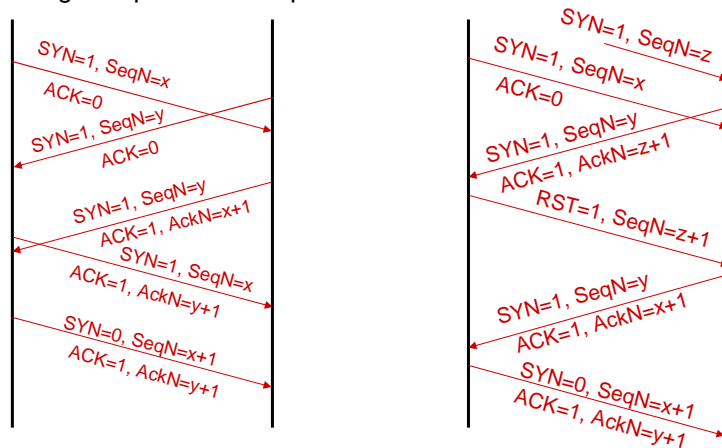
1 0.000000 192.168.10.85 192.168.10.179 TCP 56996 > 22 [SYN] Seq=3009988816 Ack=0 Win=5840 Len=0 MSS=1460...
  Frame 1 (74 bytes on wire, 74 bytes captured)
  Ethernet II, Src: 00:a0:c9:ac:ff:a6, Dst: 00:08:74:1a:3b:f7
  Internet Protocol, Src Addr: 192.168.10.85 (192.168.10.85), Dst Addr: 192.168.10.179 (192.168.10.179)
  Transmission Control Protocol, Src Port: 56996 (56996), Dst Port: 22 (22), Seq: 3009988816
    Source port: 56996 (56996)
    Destination port: 22 (22)
    Sequence number: 3009988816
    Header length: 40 bytes
    Flags: 0x0002 (SYN)
    Window size: 5840
    Checksum: 0xd009 (correct)
    Options: (20 bytes)

2 0.000357 192.168.10.179 192.168.10.85 TCP 22 > 56996 [RST, ACK] Seq=0 Ack=3009988817 Win=0 Len=0
  Frame 2 (60 bytes on wire, 60 bytes captured)
  Ethernet II, Src: 00:08:74:1a:3b:f7, Dst: 00:a0:c9:ac:ff:a6
  Internet Protocol, Src Addr: 192.168.10.179 (192.168.10.179), Dst Addr: 192.168.10.85 (192.168.10.85)
  Transmission Control Protocol, Src Port: 22 (22), Dst Port: 56996 (56996), Seq: 0, Ack: 3009988817
    Source port: 22 (22)
    Destination port: 56996 (56996)
    Sequence number: 0
    Acknowledgement number: 3009988817
    Header length: 20 bytes
    Flags: 0x0014 (RST, ACK)
    Window size: 0
    Checksum: 0xbe82 (correct)
  
```

31

Caratteristiche del 3WH

- Il three-way handshake
 - resiste alla instaurazione contemporanea di due connessioni
 - ignora pacchetti di apertura ritardatari



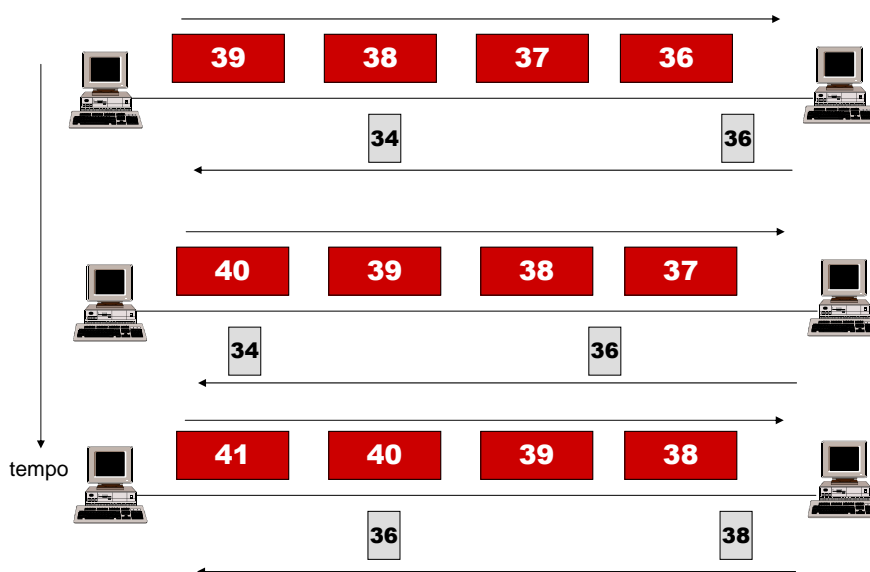
32

Conferma della trasmissione

- Se non si riceve un ACK entro un certo time-out, si ritrasmettono i segmenti non riscontrati
- Gli ACK possono essere trasmessi
 - in piggybacking utilizzando segmenti dati, quando c'è da trasmettere in direzione opposta
 - utilizzando dei segmenti ACK-only che contengono solamente l'header con ACK=1
- Gli ACK sono **cumulativi** e per default la procedura è **go-back-n**
 - si può negoziare la **selective repeat** con le Opzioni (SACK)
- Al momento della ricezione corretta di un segmento il ricevitore può inviare subito un ACK oppure ritardarlo (**Delayed ACK**)
 - l'obiettivo è quello di minimizzare il numero di ACK
 - se si ritarda troppo possono scattare i time-out → si usa un timer

33

ACK ritardati



ACK duplicati

- Quando il destinatario riceve un segmento fuori sequenza, cioè con un numero superiore a quello atteso:
 - uno o più segmenti sono andati persi
 - un segmento trasmesso dopo un altro lo ha superato a causa dei diversi percorsi possibili e dei ritardi variabili in rete
- Il TCP ricevente ritrasmette l'ACK per l'ultimo segmento ricevuto correttamente in sequenza, generando un ACK duplicato (**duplicate ACK**)
 - le implementazioni classiche di TCP ignorano gli ACK duplicati
 - le implementazioni recenti prevedono specifiche azioni se ricevono dei duplicate ACK

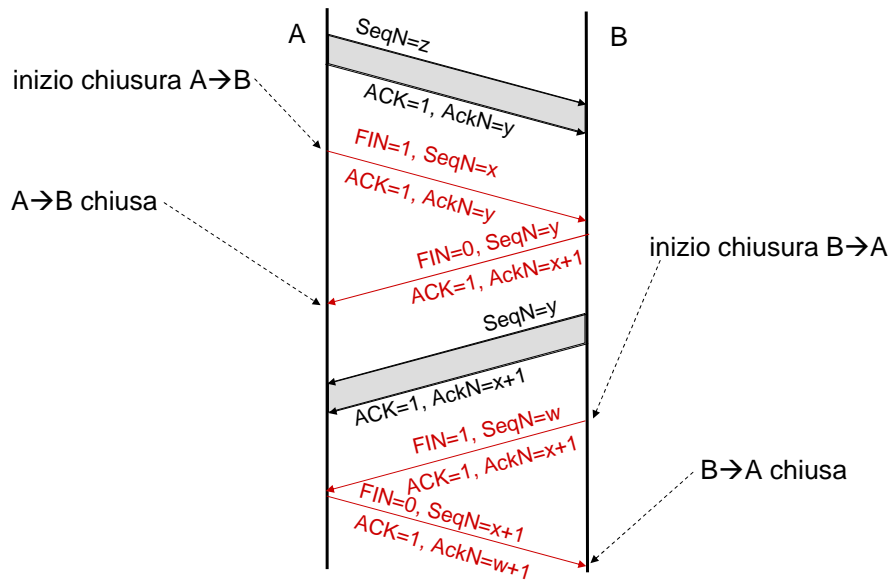
35

Chiusura della connessione TCP

- Il TCP cerca di realizzare la chiusura ordinata (**soft release**) della connessione, garantendo che non vadano persi dati
 - questo problema non può essere risolto in modo rigoroso su una rete inaffidabile in modalità full-duplex
- TCP sceglie di realizzare la chiusura in modalità simplex
 - le due direzioni vengono rilasciate in modo indipendente
 - il TCP che intende terminare la trasmissione emette un segmento con FIN=1
 - quando questo segmento riceve l'ACK la direzione si considera chiusa
 - se dopo un certo tempo non arriva l'ACK il mittente del FIN rilascia comunque la connessione
 - l'altra direzione può continuare a trasmettere dati finché non decide di chiudere

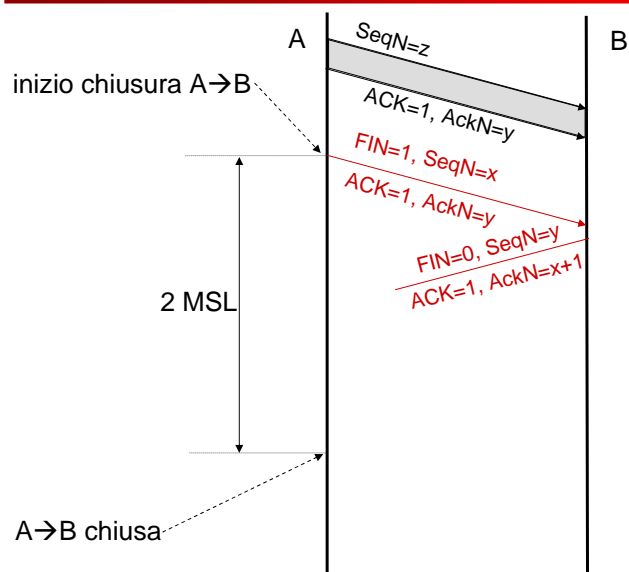
36

Esempio di chiusura normale



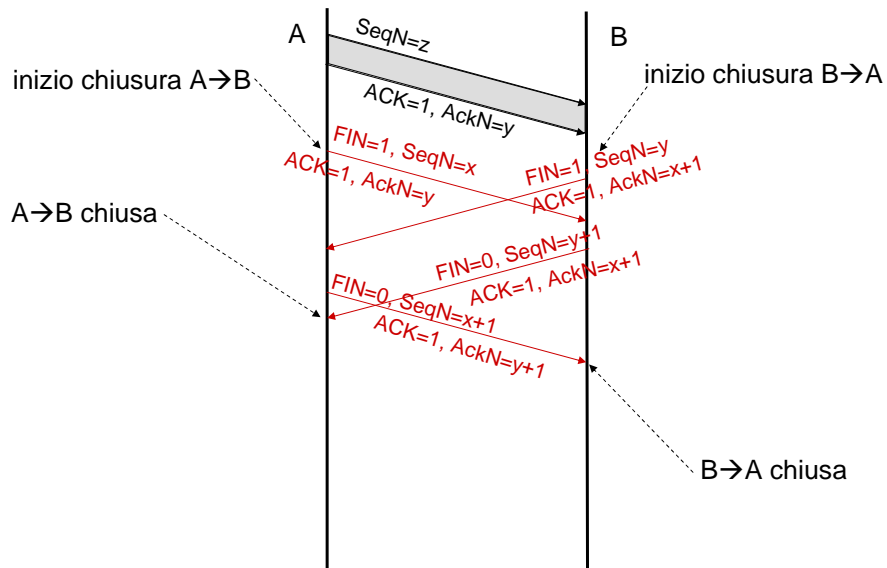
37

Esempio di chiusura con ACK perduto



38

Chiusura contemporanea



39

Comando NETSTAT

netstat -n

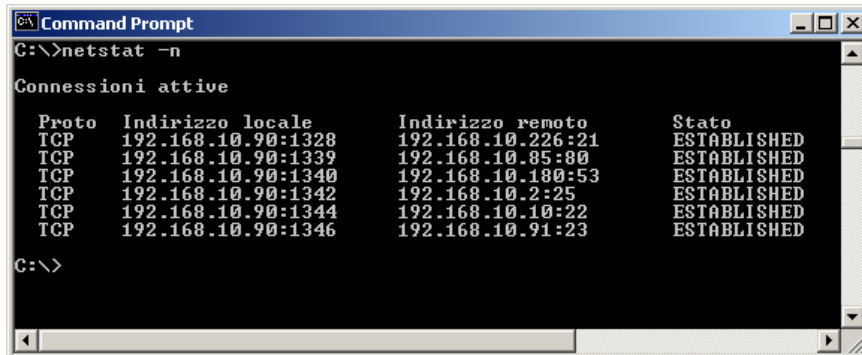
visualizza le informazioni relative alle connessioni TCP attive in un host

netstat -s

visualizza una serie di dati di tipo statistico sul traffico relativo ai diversi protocolli

40

Comando NETSTAT – Esempio 1



```
C:\>netstat -n

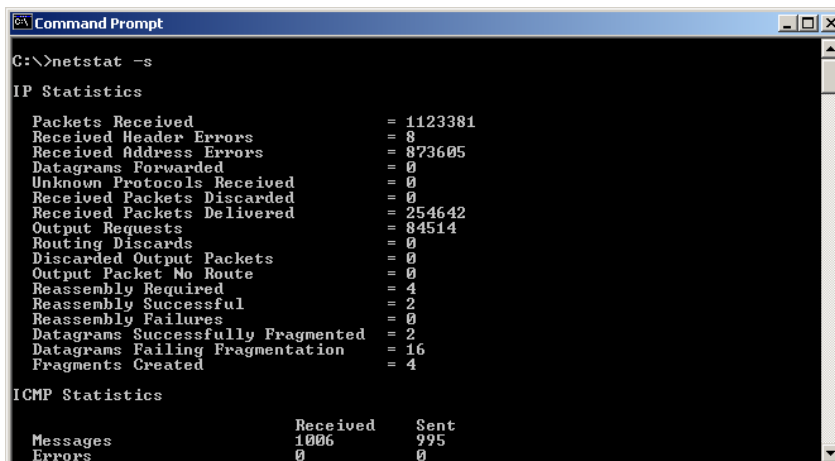
Connessioni attive

Proto  Indirizzo locale      Indirizzo remoto      Stato
TCP    192.168.10.90:1328    192.168.10.226:21    ESTABLISHED
TCP    192.168.10.90:1339    192.168.10.85:80     ESTABLISHED
TCP    192.168.10.90:1340    192.168.10.180:53    ESTABLISHED
TCP    192.168.10.90:1342    192.168.10.2:25     ESTABLISHED
TCP    192.168.10.90:1344    192.168.10.10:22     ESTABLISHED
TCP    192.168.10.90:1346    192.168.10.91:23    ESTABLISHED

C:\>
```

41

Comando NETSTAT – Esempio 2 (I)



```
C:\>netstat -s

IP Statistics

Packets Received           = 1123381
Received Header Errors     = 8
Received Address Errors    = 873605
Datagrams Forwarded        = 0
Unknown Protocols Received = 0
Received Packets Discarded = 0
Received Packets Delivered = 254642
Output Requests            = 84514
Routing Discards           = 0
Discarded Output Packets   = 0
Output Packet No Route     = 0
Reassembly Required        = 4
Reassembly Successful      = 2
Reassembly Failures        = 0
Datagrams Successfully Fragmented = 2
Datagrams Failing Fragmentation = 16
Fragments Created          = 4

ICMP Statistics

Messages      Received    Sent
Errors        0           0
```

42

Comando NETSTAT – Esempio 2 (II)

```
Command Prompt
ICMP Statistics
      Received      Sent
Messages      1006      995
Errors         0         0
Destination Unreachable 181       16
Time Exceeded  158       0
Parameter Problems 0         0
Source Quenches 0         0
Redirects      0         0
Echoes        18        961
Echo Replies   649       18
Timestamps    0         0
Timestamp Replies 0         0
Address Masks  0         0
Address Mask Replies 0         0

TCP Statistics
Active Opens          = 5591
Passive Opens        = 905
Failed Connection Attempts = 251
Reset Connections    = 1443
Current Connections  = 1
Segments Received    = 67791
Segments Sent        = 61190
Segments Retransmitted = 166

UDP Statistics
Datagrams Received   = 135828
No Ports             = 94830
Receive Errors       = 12
Datagrams Sent       = 22135

C:\>
```

43

Programma NMAP

nmap HOST

effettua un port scanning sulla macchina HOST e visualizza le porte TCP corrispondenti a processi in fase di ascolto (SERVER)

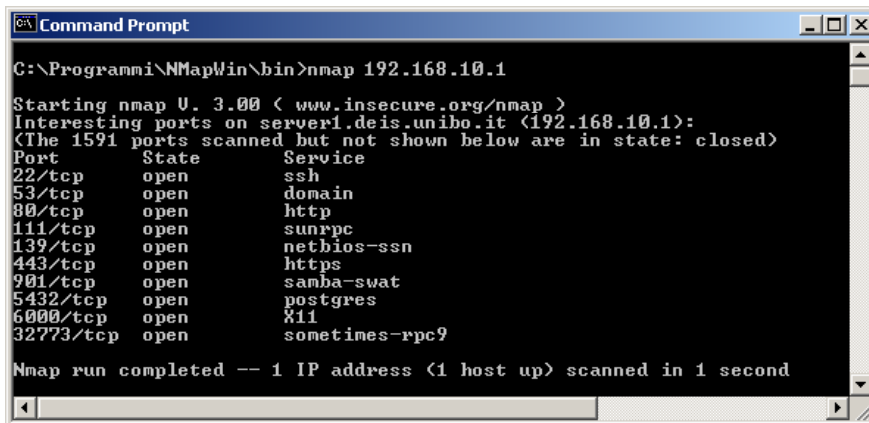
- Port Scanning:** si invia un SYN verso ogni porta
- se si riceve un SYN+ACK la porta è attiva
 - se si riceve un RST+ACK la porta è chiusa

Disponibile su:

<http://www.insecure.org/nmap>

44

Programma NMAP – Esempio 1

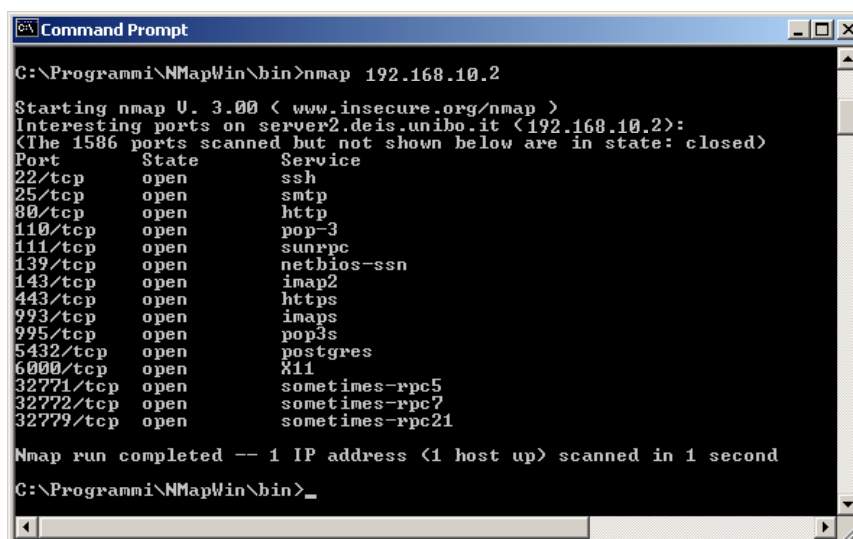


```
C:\Programmi\NMapWin\bin>nmap 192.168.10.1
Starting nmap U. 3.00 ( www.insecure.org/nmap )
Interesting ports on server1.deis.unibo.it (192.168.10.1):
(The 1591 ports scanned but not shown below are in state: closed)
Port      State      Service
22/tcp    open      ssh
53/tcp    open      domain
80/tcp    open      http
111/tcp   open      sunrpc
139/tcp   open      netbios-ssn
443/tcp   open      https
901/tcp   open      samba-swat
5432/tcp  open      postgres
6000/tcp  open      X11
32773/tcp open      sometimes-rpc9

Nmap run completed -- 1 IP address (1 host up) scanned in 1 second
```

45

Programma NMAP – Esempio 2



```
C:\Programmi\NMapWin\bin>nmap 192.168.10.2
Starting nmap U. 3.00 ( www.insecure.org/nmap )
Interesting ports on server2.deis.unibo.it (192.168.10.2):
(The 1586 ports scanned but not shown below are in state: closed)
Port      State      Service
22/tcp    open      ssh
25/tcp    open      smtp
80/tcp    open      http
110/tcp   open      pop-3
111/tcp   open      sunrpc
139/tcp   open      netbios-ssn
143/tcp   open      imap2
443/tcp   open      https
993/tcp   open      imaps
995/tcp   open      pop3s
5432/tcp  open      postgres
6000/tcp  open      X11
32771/tcp open      sometimes-rpc5
32772/tcp open      sometimes-rpc7
32779/tcp open      sometimes-rpc21

Nmap run completed -- 1 IP address (1 host up) scanned in 1 second
C:\Programmi\NMapWin\bin>_
```

46