



Laurea in INFORMATICA

Internet Reti Sicurezza A.A. 2025/2026
 Capitolo 3 – TRANSPORT PROTOCOLS
 Fausto Marcantoni
 fausto.marcantoni@unicam.it

Fausto Marcantoni

1



Dichiarazione di copyright

*L'utilizzo dei contenuti della lezione sono riservati alla fruizione personale degli studenti iscritti ai corsi dell'Università di Camerino. **Sono vietate** la diffusione intera o parziale di video o immagini della lezione, nonché la modifica dei contenuti senza il consenso, espresso per iscritto, del titolare o dei titolari dei diritti d'autore e di immagine.*

Copyright notice

The contents of this lesson are subject to copyright and intended only for personal use by students enrolled in courses offered by the University of Camerino. For this reason, any partial or total reproduction, adaptation, modification and/or transformation of the contents of this lesson, by any means, without the prior written authorization of the copyright owner, is strictly prohibited.



Fausto Marcantoni

Chapter 1 INTERNET e Reti di Calcolatori

1.2

2

Reti di elaboratori

Servizi dello strato di trasporto

Un protocollo dello **strato di trasporto** fornisce una **comunicazione logica** fra i **processi applicativi** che funzionano su host differenti

Fausto Marcantoni

Chapter 3 TRANSPORT PROTOCOL

3.3

3

Reti di elaboratori

servizi a livello di trasporto

Lato mittente

↓

Segmenti

messaggi ricevuti da un processo applicativo
convertiti in pacchetti a livello di trasporto

Il livello di trasporto, quindi, passa il segmento al livello di rete dove viene incapsulato all'interno di un pacchetto a livello di rete (**datagramma**) e inviato a destinazione.

Fausto Marcantoni

Chapter 3 TRANSPORT PROTOCOL

3.4

4

Reti di elaboratori

servizi a livello di trasporto

Lato ricevente

↓

Il livello di rete **estrae il segmento dal datagramma** e lo passa al livello superiore, quello di trasporto.

↓

Il livello di trasporto elabora il segmento ricevuto, rendendo disponibili all'applicazione destinataria i dati del segmento.

Fausto Marcantoni
Chapter 3 TRANSPORT PROTOCOL
3.5

5

Reti di elaboratori

Panoramica dello strato di trasporto

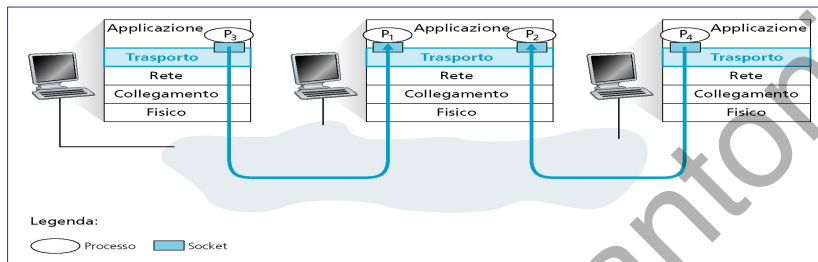
- 4-PDU → **segmento (transport-layer segment) (TCP)**
- 4-PDU → **datagramma (UDP)**
- Internet rende disponibili due protocolli dello strato di trasporto:
 - **UDP (User Datagram Protocol) → servizio inaffidabile senza connessione**
 - **TCP (Transmission Control Protocol) → servizio affidabile orientato alla connessione**
- UDP e TCP
 - Estendono il servizio di spedizione di IP fra due host al servizio di spedizione fra due processi → Multiplexing/Demultiplexing
 - Forniscono il controllo dell'integrità dei dati
- TCP fornisce
 - Trasferimento affidabile
 - Controllo di congestione
 - Controllo di flusso

Fausto Marcantoni
Chapter 3 TRANSPORT PROTOCOL
3.6

6

Multiplexing e Demultiplexing

- Ogni processo ha un **socket**
 - Porta attraverso la quale i dati passano dalla rete al processo e dal processo alla rete
- Ogni socket ha un **identificatore unico**
- Dal lato **ricevente** lo strato di trasporto determina a quale socket consegnare il segmento (**DEMULTIPLEXING**)
- Dal lato **mittente** lo strato di trasporto ottiene i dati dai diversi socket e li completa con le intestazioni (**MULTIPLEXING**)



Fausto Marcantoni

Chapter 3 TRANSPORT PROTOCOL

3.7

7

netstat

[illegible]

Fausto Marcantoni

Chapter 3 TRANSPORT PROTOCOL

3.8

8

netstat - windows

```

C:\Users\fausto.mfausto>netstat --help

Visualizza le statistiche del protocollo e le connessioni di rete TCP/IP correnti.

NETSTAT [-a] [-b] [-e] [-f] [-n] [-o] [-p proto] [-r] [-s] [-t] [-x] [-y] [interval]

-a Visualizza tutte le connessioni e le porte di ascolto.
-b Visualizza l'eseguibile coinvolto nella creazione di ogni connessione o
  porta di ascolto. In alcuni casi, host di eseguibili noti
  più componenti indipendenti e in questi casi il
  sequenza di componenti coinvolti nella creazione della connessione
  o la porta in ascolto. In questo caso, l'eseguibile
  il nome è in [] nella parte inferiore, in alto è il componente che ha chiamato,
  e così via fino al raggiungimento di TCP/IP. Si noti che questa opzione
  può richiedere molto tempo e avrà esito negativo, a meno che non siano sufficienti
  autorizzazioni.
-e Visualizza le statistiche Ethernet. È possibile combinare
  opzione.
-f Visualizza nomi di dominio completi (FQDN) per stranieri
  indirizzi.
-n Visualizza indirizzi e numeri di porta in formato numerico.
-o Visualizza l'ID del processo proprietario associato a ogni connessione.
-p proto Mostra le connessioni per il protocollo specificato da proto; proto
  può essere qualsiasi: TCP, UDP, TCPv6 o UDPv6. Se usato con -s
  opzione per la visualizzazione delle statistiche per protocollo, Proto può essere qualsiasi:
  IP, IPv6, ICMP, ICMPv6, TCP, TCPv6, UDP o UDPv6.
-q Visualizza tutte le connessioni, le porte di ascolto e i binding
  non in ascolto di porte TCP. Le porte di nonlistening associate possono o meno essere
  essere associate a una connessione attiva.
-r Visualizza la tabella di routing.
-s Visualizza le statistiche per protocollo. Per impostazione predefinita, le statistiche vengono
  visualizzate per IP, IPv6, ICMP, ICMPv6, TCP, TCPv6, UDP e UDPv6;
  l'opzione -p può essere utilizzata per specificare un sottoinsieme del valore predefinito.
-t Visualizza lo stato corrente di offload della connessione.
-x Visualizza connessioni NetworkDirect, listener e condivisi
  endpoint.
-y Visualizza il modello di connessione TCP per tutte le connessioni.
  Non può essere combinato con le altre opzioni.
intervallo Rivisualizza le statistiche selezionate, la sospensione dell'intervallo di secondi
  tra ogni schermo. Premere CTRL+C per interrompere la rivisualizzazione
  Statistiche. Se viene omissso, netstat stamperà il
  informazioni di configurazione una volta.

C:\Users\fausto.mfausto>

```

netstat - linux

```

root@localhost:~# netstat(8)

NAME
netstat - Print network connections, routing tables, interface statis-
tics, masquerade connections, and multicast memberships

SYNOPSIS
netstat [address_family_options] [--tcp|-t] [--udp|-u] [--udplite|-U]
[--sctp|-S] [--raw|-w] [--l2cap|-2] [--rfcomm|-f] [--listening|-l]
[--all|-a] [--numeric|-n] [--numeric-hosts] [--numeric-ports]
[--numeric-users] [--symbolic|-N] [--extend|-e|--extend|-e]
[--timers|-o] [--program|-p] [--verbose|-v] [--continuous|-c]
[--wide|-W] [delay]

netstat [--route|-r] [address_family_options]
[--extend|-e|--extend|-e] [--verbose|-v] [--numeric|-n]
[--numeric-hosts] [--numeric-ports] [--numeric-users] [--continuous|-c]
[delay]

netstat [--interfaces|-I|-i] [--all|-a] [--extend|-e] [--verbose|-v]
[--program|-p] [--numeric|-n] [--numeric-hosts] [--numeric-ports]
[--numeric-users] [--continuous|-c] [delay]

netstat [--groups|-g] [--numeric|-n] [--numeric-hosts]
[--numeric-ports] [--numeric-users] [--continuous|-c] [delay]

netstat [--masquerade|-M] [--extend|-e] [--numeric|-n]
[Manual page netstat(8) line 1 (press h for help or q to quit)]

```

ss - another utility to investigate sockets

```

Shell No.1
File Actions Edit View Help
SS(8) System Manager's Manual SS(8)
NAME
  ss - another utility to investigate sockets
SYNOPSIS
  ss [options] [FILTER]
DESCRIPTION
  ss is used to dump socket statistics. It can display more TCP and
  UDP statistics than netstat.
OPTIONS
  When no option is used ss displays the following information:
  -h, --help          Show summary of options.
  -V, --version       Output version information.
  -H, --no-header     Suppress header line.
  -O, --oneline       Output one line per socket.

root@kali:~# ss -ant
State  Recv-Q  Send-Q  Local Address:Port  Peer Address:Port  Process
ESTAB   0        0      193.205.92.191:56630  93.184.220.29:80    root@kali:~#
ESTAB   0        0      193.205.92.191:34100  13.35.43.91:443     root@kali:~#
ESTAB   0        0      193.205.92.191:37348  99.86.154.2:443     root@kali:~#
ESTAB   0        0      193.205.92.191:40834  34.216.82.69:443    root@kali:~#

```

Sysinternals Suite

Microsoft | TechNet

Windows Sysinternals

Search TechN

[Home](#) [Learn](#) [Downloads](#) [Community](#)

Windows Sysinternals > Downloads > Sysinternals Suite

Utilities

- Sysinternals Suite
- Utilities Index

- File and Disk Utilities
- Networking Utilities
- Process Utilities
- Security Utilities
- System Information Utilities
- Miscellaneous Utilities

Sysinternals Suite

By Mark Russinovich

Updated: August 29, 2016

[Download Sysinternals Suite \(20.2 MB\)](#)
[Download Sysinternals Suite for Nano Server \(4.4 MB\)](#)

Rate: ☆☆☆☆☆

Share this content

Introduction

<https://technet.microsoft.com/it-it/bb842062>

Reti di elaboratori

TCPView - sysinternals

Fausto Marcantoni

Chapter 3 TRANSPORT PROTOCOL

3.13

13

Reti di elaboratori

Il segmento TCP

Segmento TCP

Fausto Marcantoni

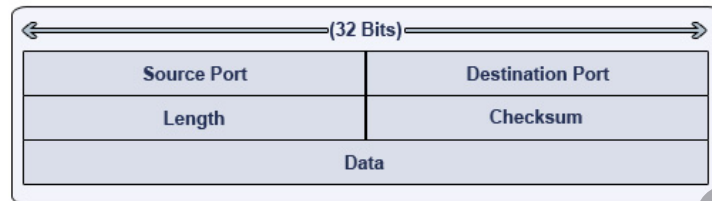
Chapter 3 TRANSPORT PROTOCOL

3.14

14

Datagramma UDP

Il pacchetto UDP è chiamato **user datagram**.



Numeri di porta

- I **campi speciali dell'intestazione** del segmento che indicano il socket a cui consegnare il processo si chiamano **PORTE**
- CAMPO NUMERO DI PORTA SORGENTE
- CAMPO NUMERO DI PORTA DESTINAZIONE
- 16 bit per il numero di porta → 0 a 65535
- Numeri di porta ben conosciuti (riservati) da 0 a 1023

<http://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.txt>

/etc/services

```

1 # Network services, Internet style
2 #
3 # Updated from https://www.iana.org/assignments/service-names-port-numbers/service
4 #
5 # New ports will be added on request if they have been officially assigned
6 # by IANA and used in the real-world or are needed by a debian package.
7 # If you need a huge list of used numbers please install the nmap package.
8
9 tcpmux      1/tcp          # TCP port service multiplexer
10 echo       7/tcp
11 echo       7/udp
12 discard    9/tcp          sink null
13 discard    9/udp          sink null
14 systat    11/tcp          users
15 daytime    13/tcp
16 daytime    13/udp
17 netstat    15/tcp
18 qotd       17/tcp          quote
19 chargen    19/tcp          ttytst source
20 chargen    19/udp          ttytst source
21 ftp-data   20/tcp
22 ftp        21/tcp
23 fsp        21/udp          fspd
24 ssh        22/tcp          # SSH Remote Login Protocol
25 telnet     23/tcp
26 smtp       25/tcp          mail
27 time       37/tcp          timeserver
28 time       37/udp          timeserver
29 whois      43/tcp          nicname
30 tacacs     49/tcp          # Login Host Protocol (TACACS)
31 tacacs     49/udp
32 domain     53/tcp          # Domain Name Server

```

Numeri di porta

■ Operazioni Demultiplexing

1. Ad ogni socket è assegnato dal terminale un numero di porta
2. All'arrivo, lo strato di trasporto esamina il numero di porta di destinazione
3. Dirige il segmento verso il socket corrispondente
4. Dal socket si arriva al processo di destinazione

Reti di elaboratori
Numeri di porte (TCP)

```

> Frame 109: 438 bytes on wire (3504 bits), 438 bytes captured (3504 bits) on interface 0
> Ethernet II, Src: Cisco_f6:85:00 (08:96:ad:f6:85:00), Dst: AsustekC_0a:e6:1d (d8:50:e6:0a:e6:1d)
> Internet Protocol Version 4, Src: a1089.dscd.akamai.net (193.206.135.170), Dst: mfausto.amministrazione.unicam (193.205.92.108)
< Transmission Control Protocol, Src Port: http (80), Dst Port: ndtp (2882), Seq: 1, Ack: 320, Len: 384
  Source Port: http (80)
  Destination Port: ndtp (2882)
  [Stream index: 2]
  [TCP Segment Len: 384]
  Sequence number: 1 (relative sequence number)

```

NUMERO DI PORTA SORGENTE

NUMERO DI PORTA DESTINAZIONE

Fausto Marcantoni
Chapter 3 TRANSPORT PROTOCOL
3.19

19

Reti di elaboratori
Numeri di porta

I campi del numero di porta di origine e di destinazione nei segmenti a livello di trasporto.

```

Transmission Control Protocol, Src Port: http (80), Dst Port: ndtp (2882), Seq: 1, Ack
Source Port: http (80)
Destination Port: ndtp (2882)
[Stream index: 2]

```

```

20  01011100 01101100 00000000 01010000 00001011 01000010 11010101 01111010  \l P B z
28  11011110 10001110 10010111 00001010 11000000 00001101 01010000 00011000  .....P.
30  00000000 11101101 10101000 01111100 00000000 00000000 01001000 01010100  ...|..HT

```

32 bit

00000000 01010000=80

Numero di porta di origine

Numero di porta di destinazione

Altri campi dell'intestazione

Dati dell'applicazione (messaggio)

00001011 01000010 =2882

Fausto Marcantoni
Chapter 3 TRANSPORT PROTOCOL
3.20

20

Reti di elaboratori

Numeri di porte (UDP)

```

> Frame 222: 1285 bytes on wire (10280 bits), 1285 bytes captured (10280 bits) on interface \Device\
> Ethernet II, Src: ASUSTekC_0a:e6:1d (d8:50:e6:0a:e6:1d), Dst: Cisco_e2:b4:5f (10:b3:d5:e2:b4:5f)
> Internet Protocol Version 4, Src: mfausto.amministrazione.unicam (193.205.92.201), Dst: mil04s50-i
  User Datagram Protocol, Src Port: 57863, Dst Port: 443
    Source Port: 57863
    Destination Port: 443
    Length: 1251
    Checksum: 0x8395 [unverified]
    [Checksum Status: Unverified]
    [Stream index: 7]
    [Timestamps]
    UDP payload (1243 bytes)
  
```

NUMERO DI PORTA SORGENTE

NUMERO DI PORTA DESTINAZIONE

```

0020  d1 0e e2 07 01 bb 04 e3 83 95 42 6e eb c6 21 49  ....!I
0030  d4 85 a3 d3 cd 68 34 3f 1b dd 36 d0 80 7c 30 a6  ....h4? ..6..|
  
```

Fausto Marcantoni

Chapter 3 TRANSPORT PROTOCOL

3.21

21

Reti di elaboratori

Comunicazione processo - processo

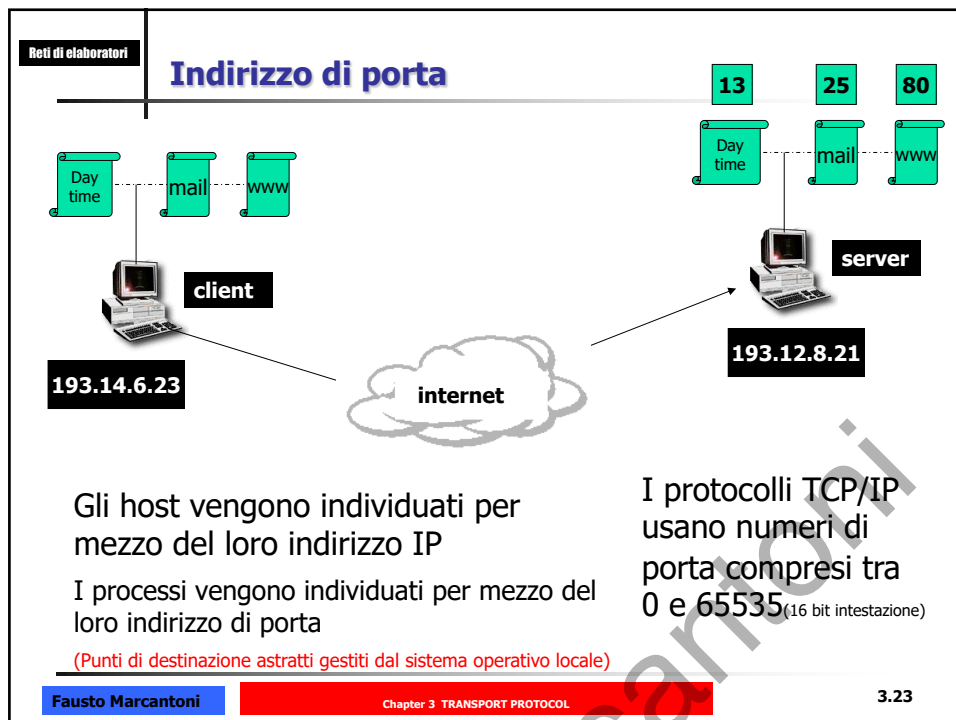
- IP si occupa della comunicazione tra dispositivi
- Il messaggio deve giungere al processo → UDP e/o TCP

Fausto Marcantoni

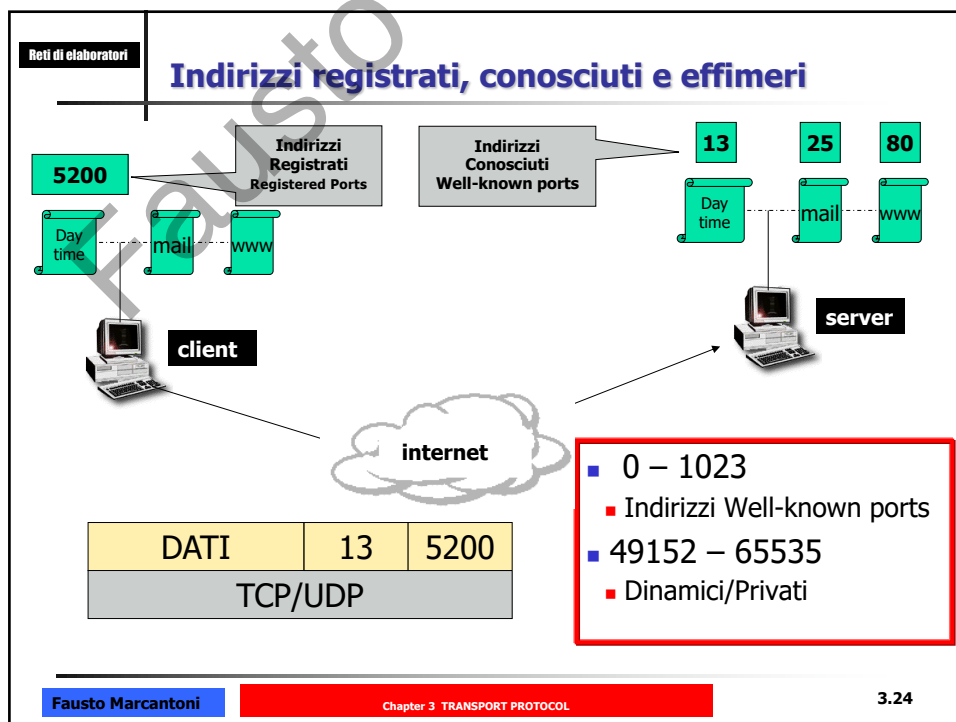
Chapter 3 TRANSPORT PROTOCOL

3.22

22



23



24

Reti di elaboratori				
Indirizzi conosciuti, dinamici, effimeri				
Port	TCP	UDP	IANA status ^[1]	Description
0	Reserved	Reserved	Official	In programming APIs (not in communication between hosts), requests a system-allocated (dynamic) port ^[5]
	N/A	N/A	Unofficial	
1	Yes	Assigned	Official	TCP Port Service Multiplexer (TCPMUX). Historic. Both TCP and UDP have been assigned to TCPMUX by IANA, ^[1] but by design only TCP is specified. ^[6]
5	Assigned	Assigned	Official	Remote Job Entry ^[7] was historically using socket 5 in its <i>old socket form</i> , while MIB PIM has identified it as TCP/5 ^[8] and IANA has assigned both TCP and UDP 5 to it.
7	Yes	Yes	Official	Echo Protocol ^{[9][10]}
9	Yes, and SCTP ^[11]	Yes	Official	Discard Protocol ^[12]
	No	Yes	Unofficial	
11	Yes	Yes	Official	Active Users (systat service) ^{[14][15]}
13	Yes	Yes	Official	Daytime Protocol ^[16]
Dynamic, private or ephemeral ports [hide]				
Port	TCP	UDP	Description	
49152–65535	Yes	No	Certificate Management over CMS ^[345]	
60000–61000	Port 22	Yes	Range from which Mosh – a remote-terminal application similar to SSH – typically assigns ports for ongoing sessions between Mosh servers and Mosh clients. ^[346]	
64738	Yes	Yes	Mumble ^[347]	

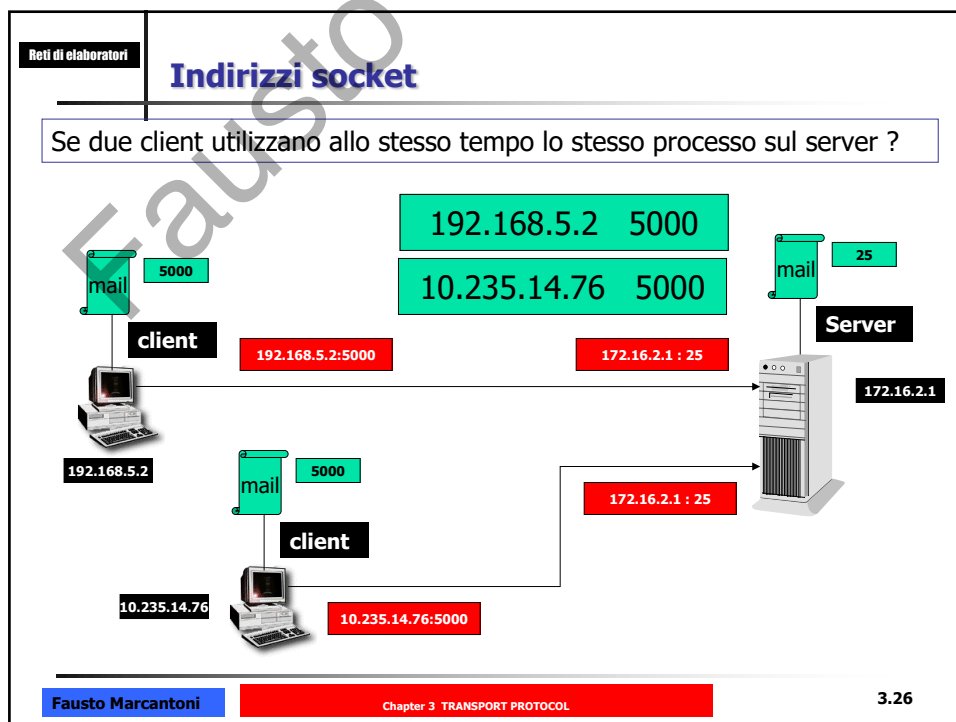
http://en.wikipedia.org/wiki/List_of_TCP_and_UDP_port_numbers

Fausto Marcantoni

Chapter 3 TRANSPORT PROTOCOL

3.25

25



Fausto Marcantoni

Chapter 3 TRANSPORT PROTOCOL

3.26

26

Reti di elaboratori

Multiplazione e Demultiplazione senza connessione

- Socket **UDP** univocamente determinato da una **coppia**:
 - <IP server> <Port Server>
 - | 32 bit | 16 bit |
- Molti client diversi accedono allo stesso processo sullo stesso server

Fausto Marcantoni

Chapter 3 TRANSPORT PROTOCOL

3.27

27

Reti di elaboratori

Multiplazione e Demultiplazione con connessione

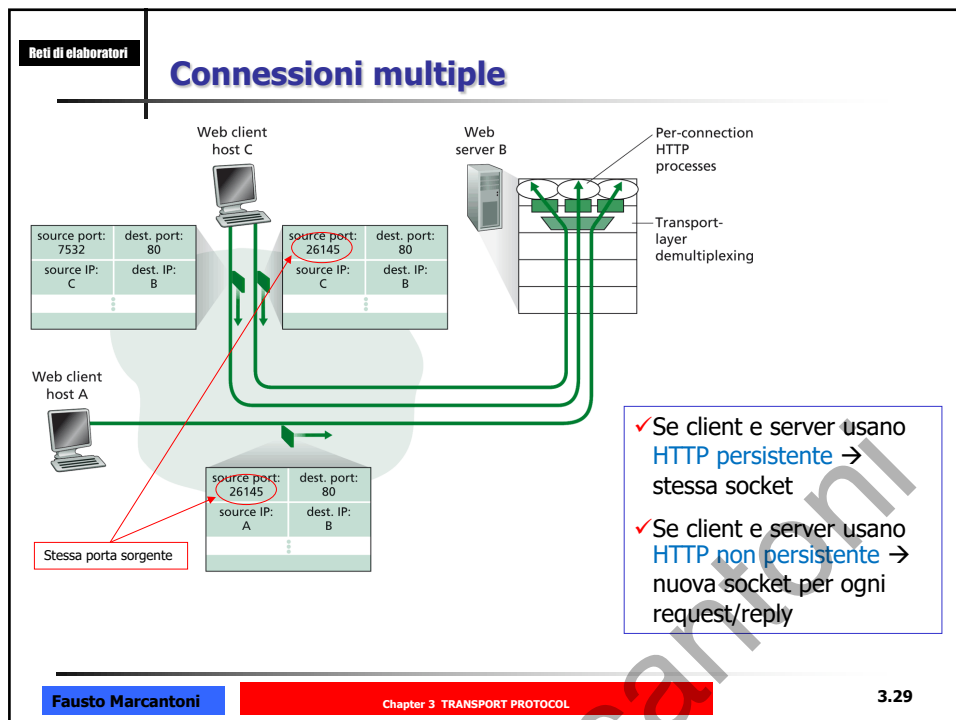
- **TCP** identifica un "canale" di comunicazione con il nome di **connessione**
- Sono identificate dalla **quadrupla**:
 - <IP client><IP server><Port client><Port Server>
 - | 32 bit | 32 bit | 16 bit | 16 bit |
- Questa soluzione permette
 1. A molti client diversi di accedere allo stesso servizio sullo stesso server
 2. Allo stesso client di attivare più sessioni dello stesso servizio
- Viola il modello a layer
 - informazioni di livello 4 mischiate con informazioni di livello 3

Fausto Marcantoni

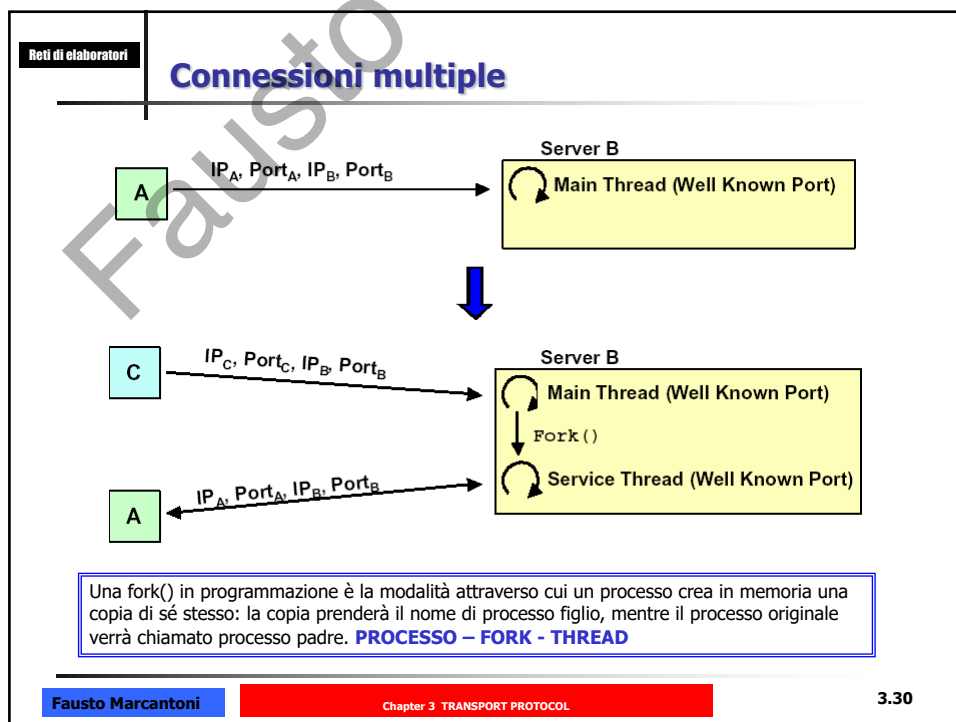
Chapter 3 TRANSPORT PROTOCOL

3.28

28



29



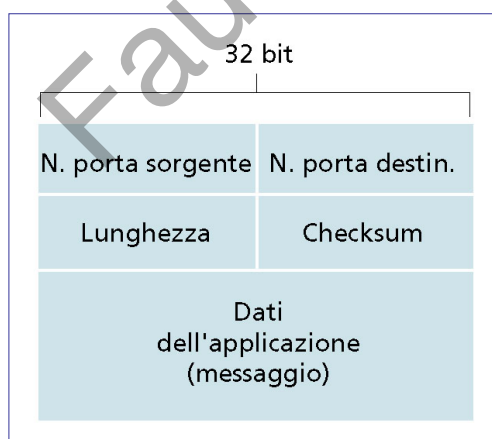
30

Server iterativi e concorrenti

- Modalità con le quali i server possono gestire le connessioni in ingresso
- Iterativi:
 - esiste un solo thread server;
 - gestisce la connessione seguente solo quando quella precedente è terminata
 - Utilizzati da servizi basati su UDP
- Concorrenti:
 - esiste un “main” thread server, che si clona ogni volta che arriva una nuova connessione;
 - possono essere servite più connessioni contemporaneamente
 - Utilizzati da servizi basati su TCP

31

Trasporto senza connessione: UDP



- Minimo necessario per lo strato di trasporto
 - Multiplexing/demultiplexing
 - Verifica degli errori

32

Checksum UDP

Mittente

- calcola la somma di tutte le parole (2 byte) contenute nel messaggio
- calcola il complemento ad 1 della somma
- invia il risultato nel segmento UDP

Messaggio di 3 parole di 2 byte
 0110011001100110
 0101010101010101
 0000111100001111

0110011001100110 +
 0101010101010101 =

1011101110111011 +
 0000111100001111 =

1100101011001010

Complemento ad 1

0011010100110101

checksum = 0011010100110101

<https://www.thegeekstuff.com/2012/05/ip-header-checksum/>

<https://www.youtube.com/watch?v=LfMyBYLw-jM>

Checksum UDP

Destinatario

- calcola la somma di tutte le parole (2 byte) contenute nel segmento + la checksum

1100101011001010 +
 0011010100110101 =

1111111111111111

- se non si sono verificati errori in trasmissione il risultato deve essere uguale ad una sequenza di 1.

- Se si e' verificato un errore = il segmento viene scartato

[RFC-1071 - Calcolare l'internet checksum](#)

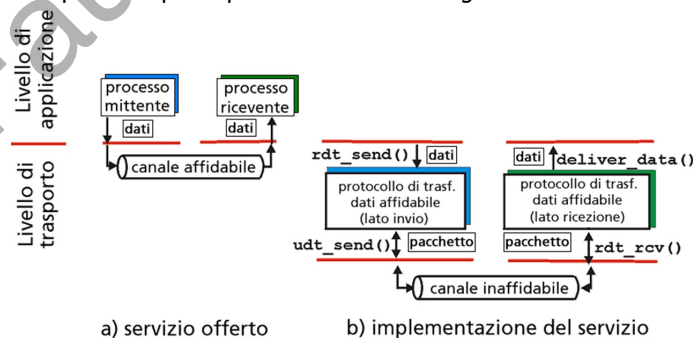
<http://www.ietf.org/rfc/rfc1071.txt>

Perché scegliere UDP

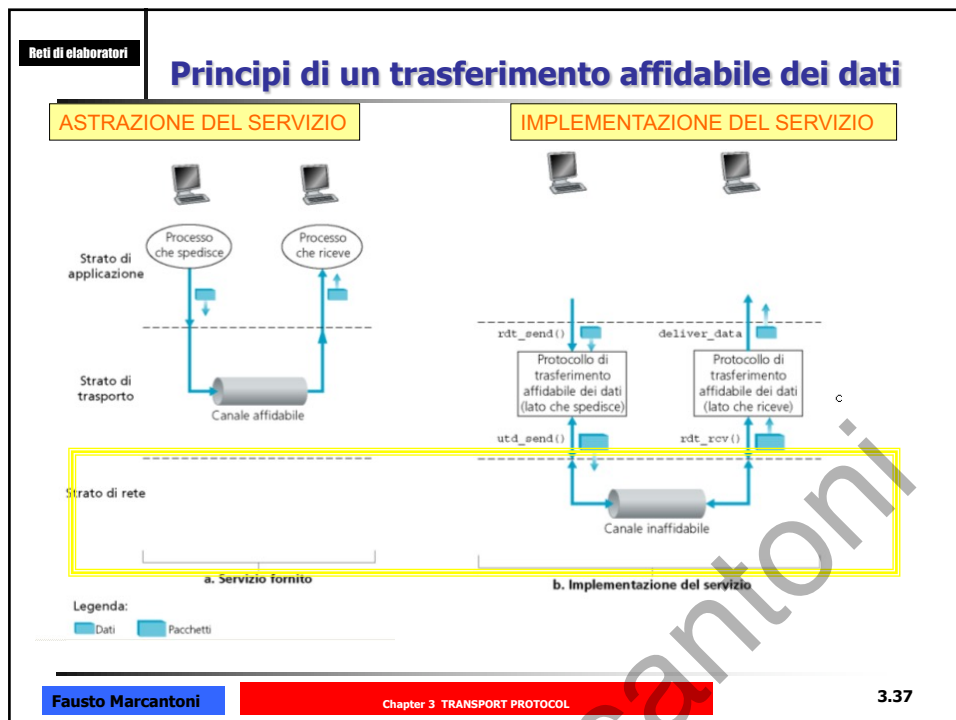
- i. **Non viene creata alcuna connessione** → non introduce alcun ritardo dovuto alla fase di impostazione della connessione
- ii. **Nessuno stato della connessione** → quindi un server può supportare molti più client attivi
- iii. **Poco sovraccarico dovuto all'instestazione del pacchetto** → 8 byte di overhead contro 20
- iv. **Controllo del livello applicativo più incisivo** → senza controllo di congestione il mittente non viene mai inondato, allagato (flooding)

Principi del trasferimento dati affidabile

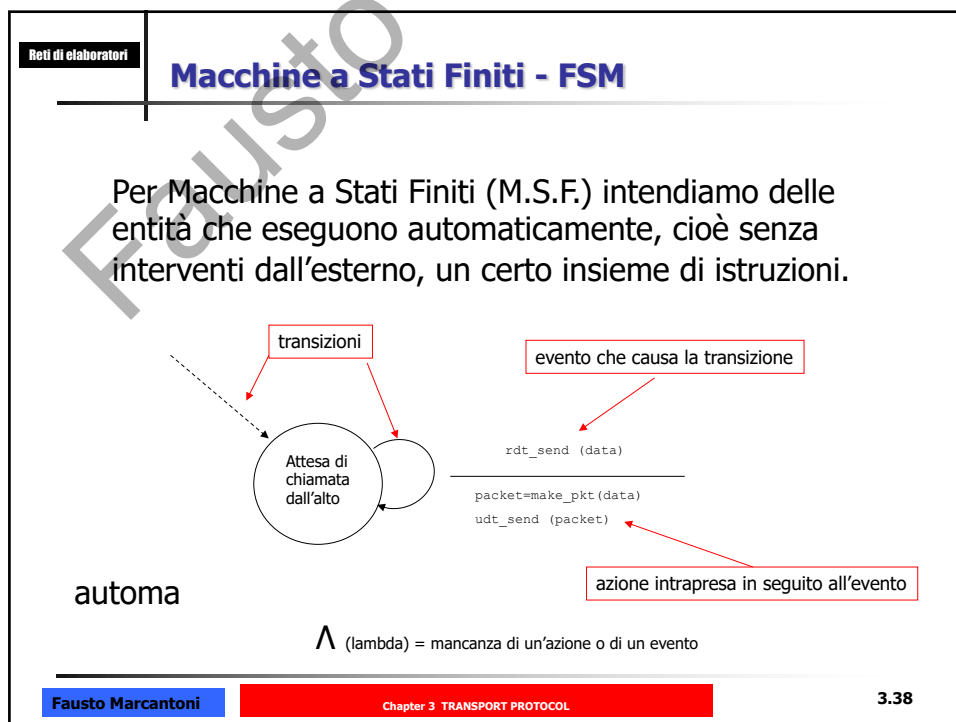
- ✓ Importante nei livelli di applicazione, trasporto e collegamento
- ✓ Tra i dieci problemi più importanti del networking!



Le caratteristiche del **canale inaffidabile** determinano la **complessità del protocollo di trasferimento dati affidabile** (reliable data transfer o rdt)



37



38

Costruzione di un protocollo di trasferimento dati affidabile

Trasferimento dati affidabile su un canale **perfettamente affidabile**:

rdt1.0 rdt=reliable data transfer

canale sottostante completamente affidabile:

- nessun errore nei bit
- nessuna perdita di pacchetti

automa distinto per il **mittente** e per il **ricevente**:

- il **mittente** **invia i dati** nel canale sottostante
- il **ricevente** **legge i dati** dal canale sottostante

Trasferimento affidabile dei dati su un canale completamente affidabile: rdt 1.0

LATO CHE SPEDISCE

Attesa di chiamata da sopra

rdt_send (data)

```
packet=make_pkt(data)
udt_send(packet)
```

rdt=reliable data transfer

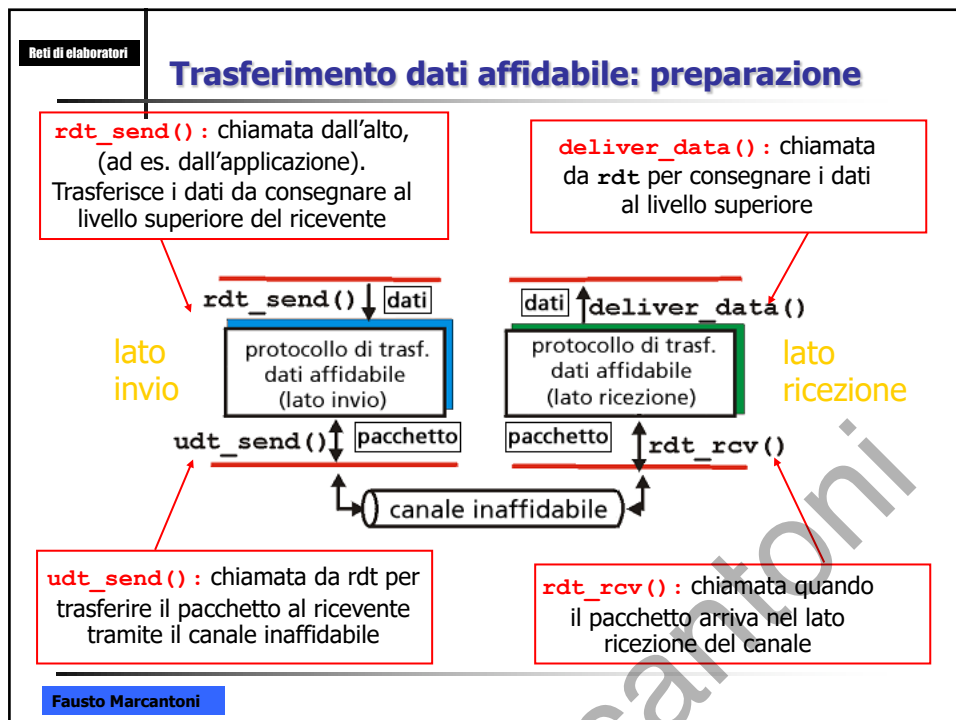
LATO CHE RICEVE

Attesa di chiamata dal basso

rdt_rcv (packet)

```
extract(packet,data)
deliver_data(data)
```

udt=unreliable data transfer



41

Reti di elaboratori

Il protocollo stop and wait

Il protocollo **stop and wait** fa parte di un sistema di telecomunicazione definito:

Automatic Repeat-reQuest (ARQ)

- è un protocollo di controllo di errore, che svolge il compito di rivelare un errore (ma non di correggerlo)
- I pacchetti corrotti vengono scartati e viene richiesta la loro ritrasmissione

Esempio: dettatura tramite telefono di un articolo sportivo

Fausto Marcantoni

Chapter 3 TRANSPORT PROTOCOL

3.42

42

Automatic Repeat-reQuest (ARQ)

3 diversi protocolli più comuni

- ✓ **PROCOLLO STOP AND WAIT**; il mittente invia un messaggio e attende dal destinatario una conferma positiva (**ACK**, acknowledge), negativa (**NACK**, contrazione di negative acknowledge) o un comando;
- ✓ se scade il tempo di attesa (time-out) per uno di questi tre, *il mittente provvederà a rispedire il pacchetto e il destinatario si incaricherà di scartare eventuali repliche.*
- ✓ Nel caso in cui si verificasse *un errore nella trasmissione del segnale di conferma (ACK), il mittente provvederà a rinviare il pacchetto*; il destinatario riceverà in questo modo una copia del pacchetto già ricevuto, credendo che gli sia pervenuto un nuovo pacchetto. Per evitare questo problema si può procedere numerando i pacchetti trasmessi, ovvero inserendo un bit di conteggio.

Automatic Repeat-reQuest (ARQ)

3 diversi protocolli più comuni

- ✓ **Go-Back-N**; **il mittente** dispone di **un buffer dove immagazzina 'N' pacchetti da spedire**, man mano che *riceve la conferma ACK svuota il buffer e lo riempie con nuovi pacchetti*;
- ✓ nell'eventualità di pacchetti persi o danneggiati e scartati avviene il rinvio del blocco di pacchetti interessati.
- ✓ I pacchetti ricevuti dal destinatario dopo quello scartato vengono eliminati.
- ✓ **Selective Repeat**; in questo caso anche **il destinatario** dispone di **un buffer dove memorizzare i pacchetti ricevuti dopo quello/quelli scartati**;
- ✓ quando i pacchetti interessati vengono correttamente ricevuti, il buffer viene svuotato (mittente) o i pacchetti contenuti salvati (destinatario).

Un canale reale può alterare i bit di un pacchetto

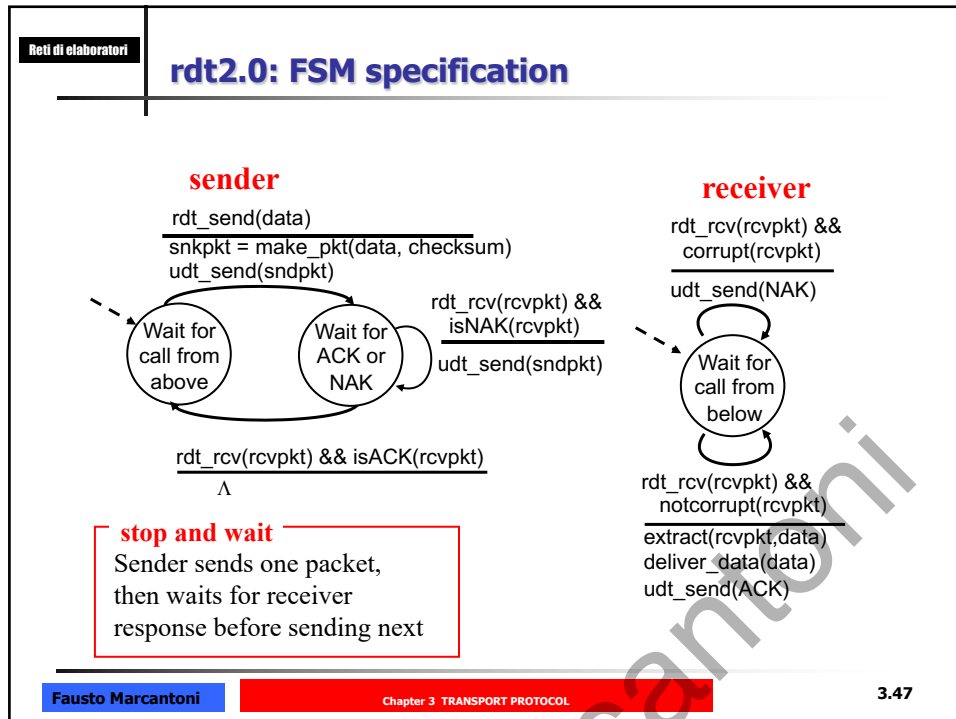
Protocolli di tipo ARQ (Automatic Repeat reQuest)

- Rilevamento errori (errori dovuti alle componenti fisiche delle reti)
- Feedback dal ricevente
 - Riscontri positivi ACK = Acknowledge
 - Riscontri negativi NAK = Not Acknowledge
- Ritrasmissione

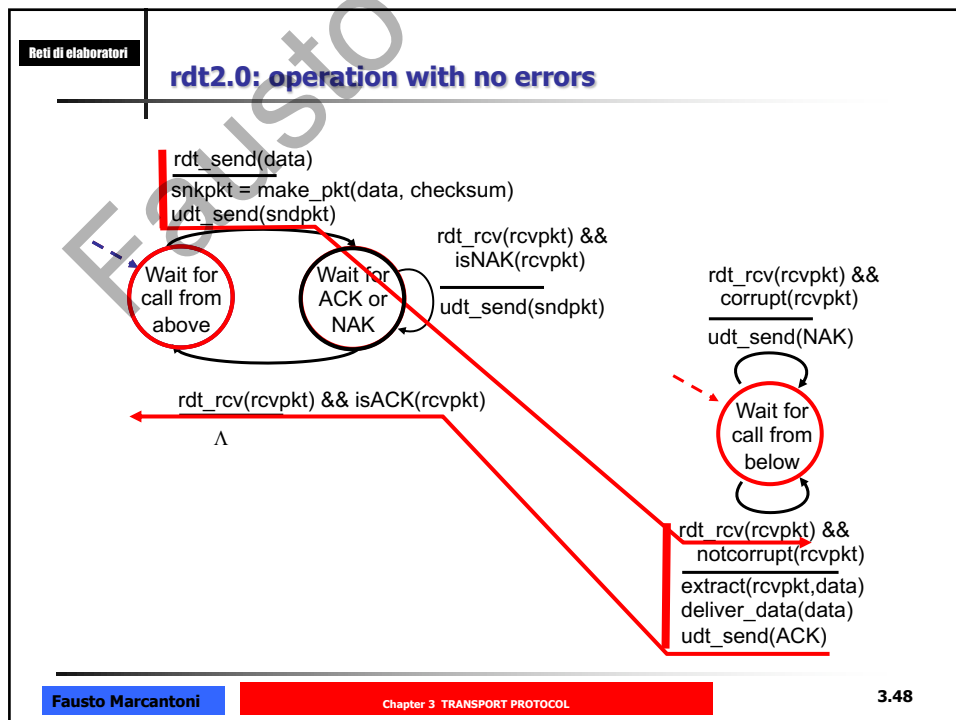
Protocolli di tipo ARQ (Automatic Repeat reQuest)

- Rilevamento di errore
 - ✓ Il destinatario deve rilevare eventuali errori sui bit
- Feedback del destinatario
 - ✓ Le risposte di notifica positiva (ACK) e negativa (NAK) devono essere trasmesse tra mittente e destinatario
- Ritrasmissione
 - ✓ Un pacchetto ricevuto con errori sarà ritrasmesso

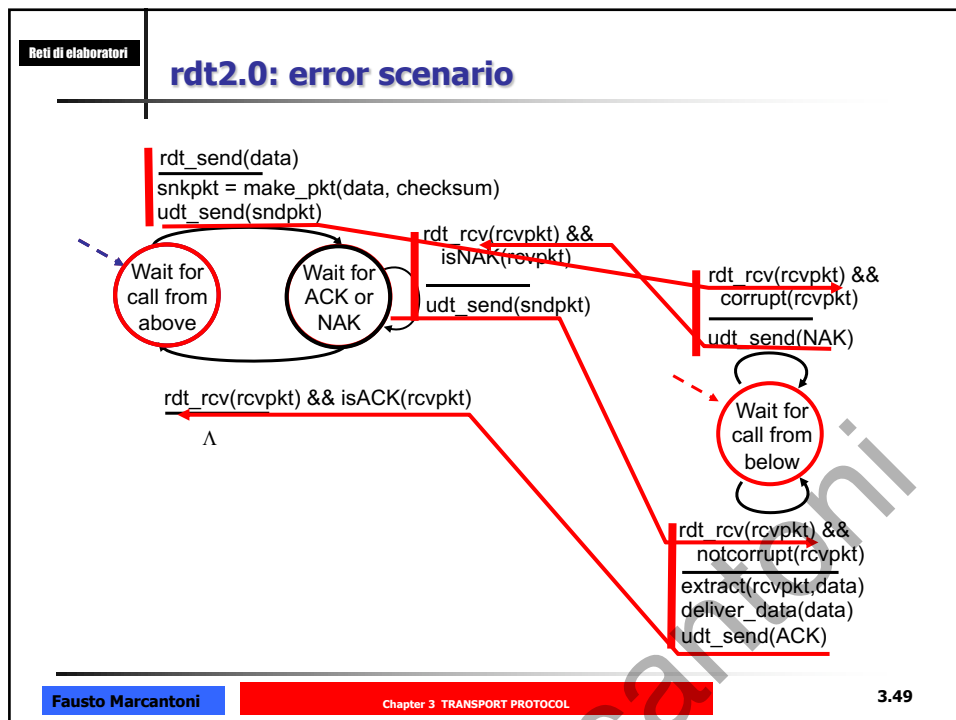




47



48



49

Reti di elaboratori

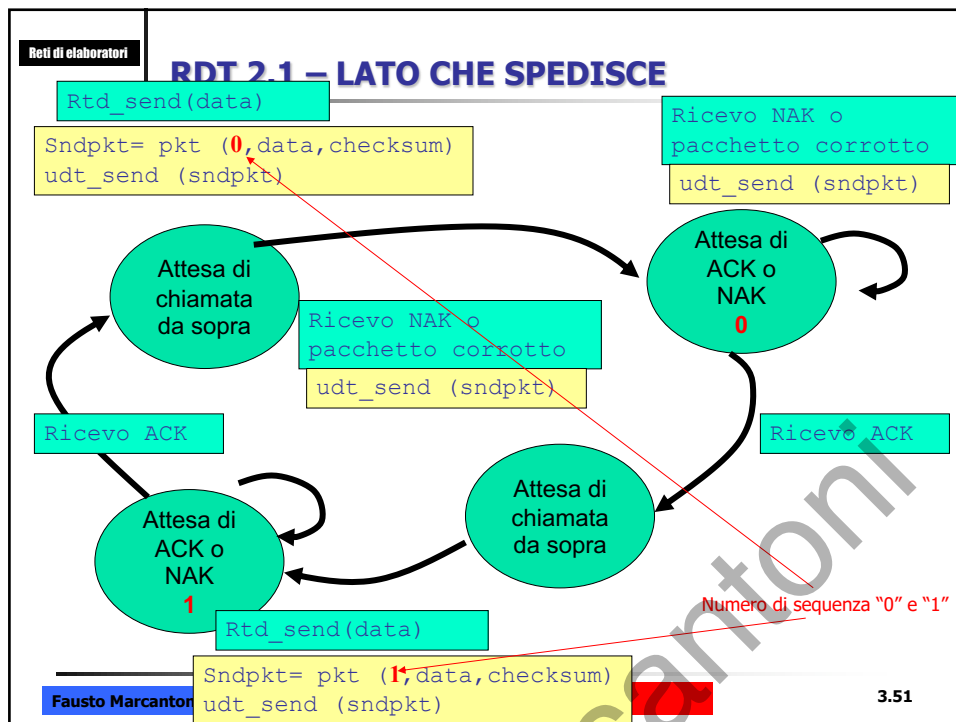
I pacchetti ACK e NAK possono essere alterati

- Se il mittente non comprende l'assenso o il dissenso potrà richiedere di ritrasmettere.
 - "che cosa hai detto?" <disturbo> "che cosa hai detto TU ?"
 - ma se poi anche questa richiesta sarà corrotta ...
- Aggiungo bit al checksum permettendo al receiver non solo da rilevare, ma anche di correggere.
 - dispendioso, costoso da realizzare
 - se poi il pacchetto va perso ?
- Il sender ritrasmette il pacchetto quando riceve un ACK/NAK difettoso
 - pacchetti duplicati
 - il receiver non sa se il pacchetto è nuovo o duplicato

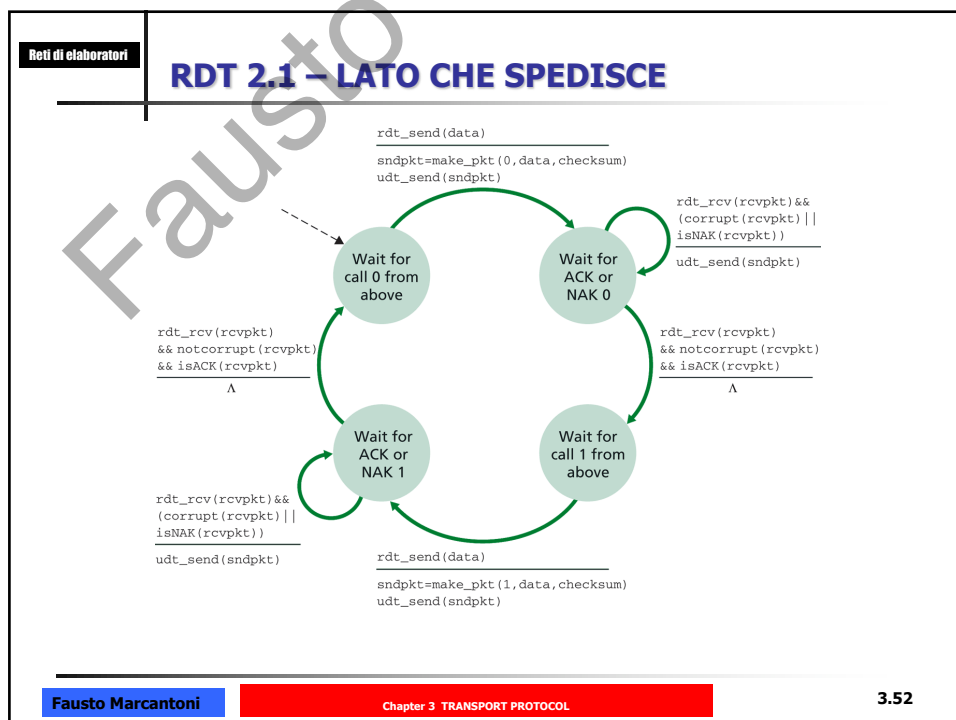
NUMERO DI SEQUENZA inserito nel pacchetto dati

Fausto Marcantoni Chapter 3 TRANSPORT PROTOCOL 3.50

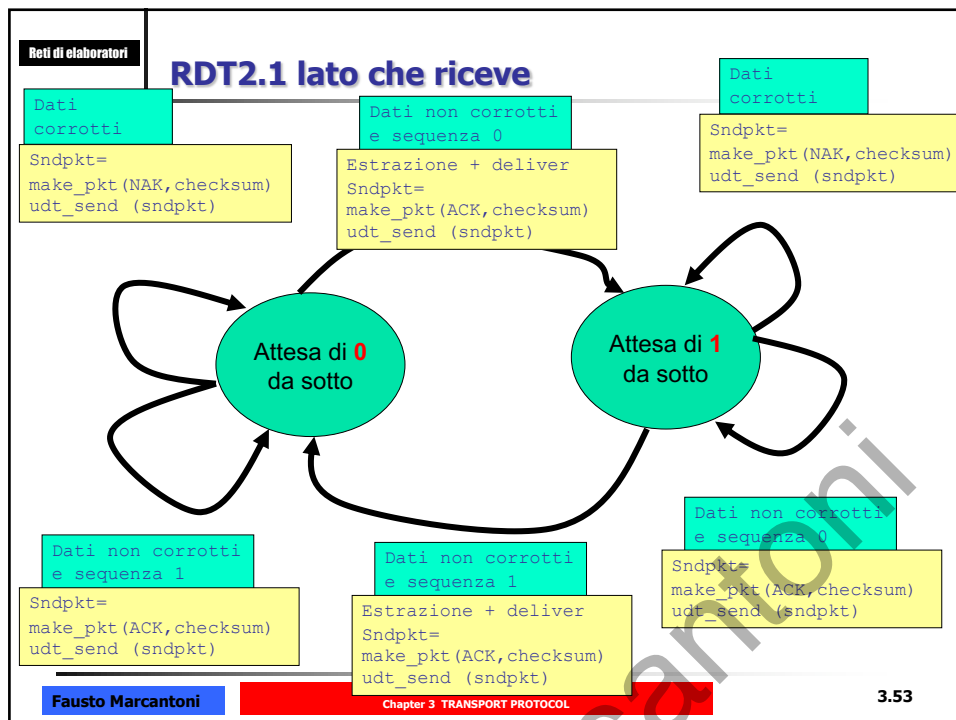
50



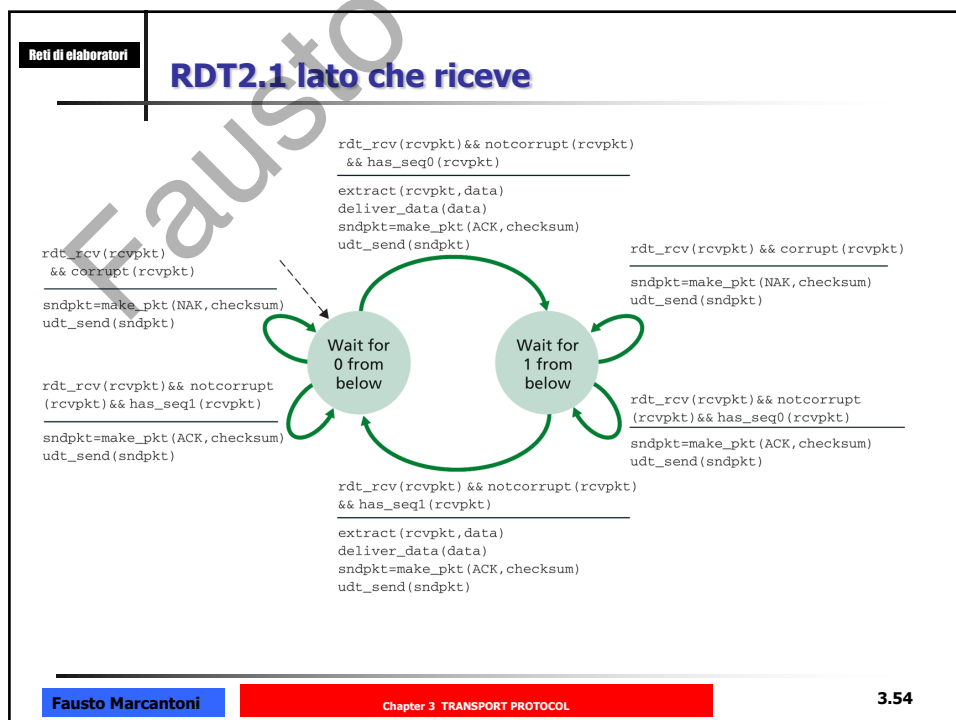
51



52



53

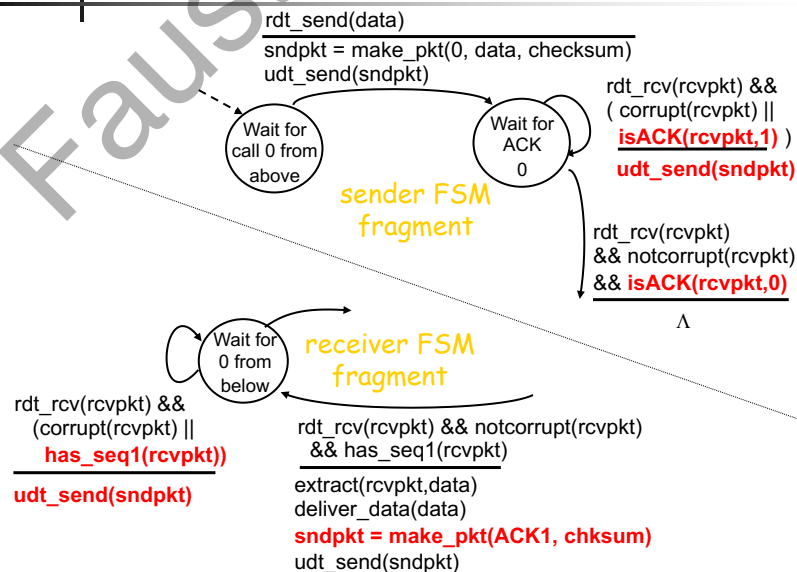


54

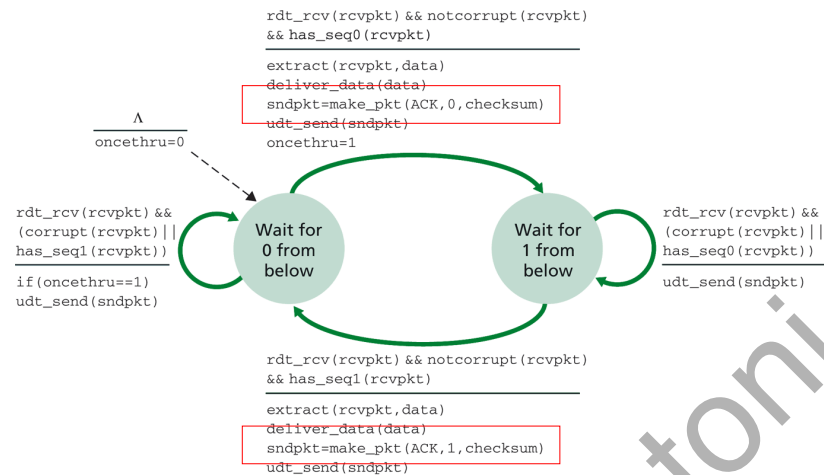
Evoluzione versione 2.2

- Possiamo ottenere lo stesso effetto di un NAK, se inviamo solo un **ACK per l'ultimo pacchetto** ricevuto correttamente
- Un sender che riceve due ACK per lo stesso pacchetto riconosce che il receiver non ha avuto nel modo corretto il pacchetto successivo (**ACK duplicati**)
- DUE MODIFICHE
 1. Il **ricevente** deve **includere** il numero di sequenza del pacchetto in un messaggio ACK
 2. Il **mittente** deve **controllare** il numero di sequenza che accompagna un ACK

rdt2.2: sender, receiver fragments



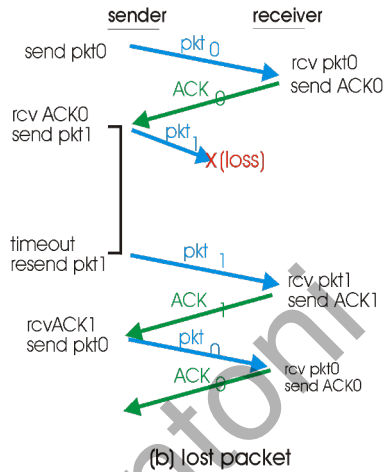
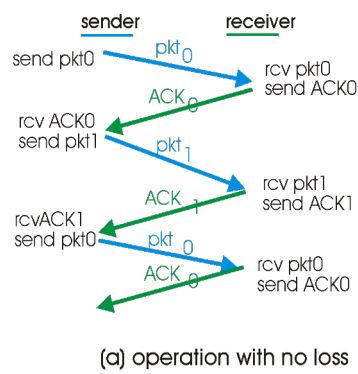
rdt2.2: sender, receiver fragments



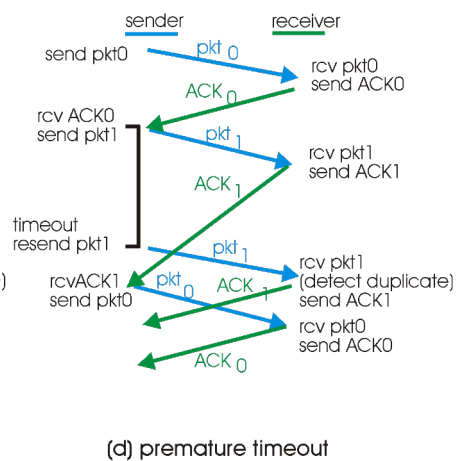
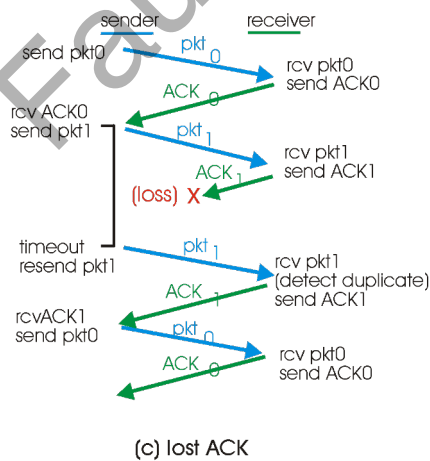
Perdita di pacchetti rdt 3.0

- Operazione di recupero compiuta dal **sender**
 - Aspetta un tempo prefissato e se non succede niente ritrasmette il pacchetto
 - Tempo di attesa predefinito → può ritrasmettere un pacchetto anche se il suo ACK arriva in ritardo
- Meccanismo di conto alla rovescia (countdown timer)
 - **Avviare il timer** ogni qual volta un pacchetto è inviato per la prima volta o ritrasmesso
 - **Rispondere alle interruzioni del timer** eseguendo le azioni appropriate
 - **Arrestare il timer**
 - Il **receiver** inoltre dovrà **inserire nell'ACK il numero di sequenza del pacchetto** per consentire al sender di valutare se l'ACK è di un pacchetto recente o di un pacchetto in ritardo

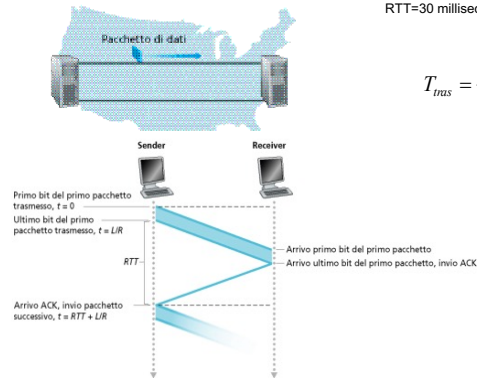
rdt3.0 in action



rdt3.0 in action (II)



I problemi dei protocolli stop-and-wait



RTT=30 millisecondi - 1Gb/s = 10^9 bit/s - Pacchetto = 1000 Byte

$$T_{tras} = \frac{L}{R} = \frac{8000 \text{ bit/pacchetto}}{10^9 \text{ bit/s}} = 8 \text{ microsecondi}$$

l'ultimo bit del pacchetto che arriva al receiver al tempo

$$t = RTT/2 + L/R = 15,008 \text{ ms}$$

l'ACK ritorna al mittente al tempo

$$t = RTT + L/R = 30,008 \text{ ms}$$

$U_{senders}$: utilizzazione del sender come la frazione di tempo in cui il sender è davvero occupato a spedire bit attraverso il canale

$$U_{sender} = \frac{\frac{L}{R}}{RTT + \frac{L}{R}} = \frac{0,008}{30,008} = 0,000266595$$

in grado di spedire solo 1000 byte in 30,008 millisecondi
una capacità effettiva (throughput) di soli 267 kbit/s

$$\text{throughput} = \frac{(1000 \cdot 8)}{30,008} = 266595 \text{ bit/sec}$$

Fausto Marcantoni

Chapter 3 TRANSPORT PROTOCOL

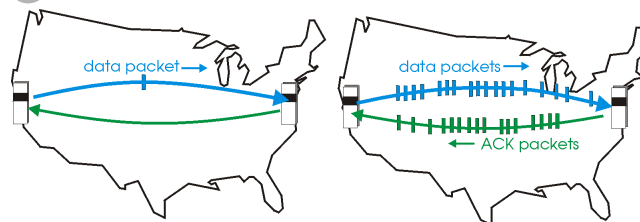
3.61

61

Pipelined protocols

Pipelining: invio di più pacchetti senza aspettare i riscontri

- range of sequence numbers must be increased
- buffering at sender and/or receiver



(a) a stop-and-wait protocol in operation

(b) a pipelined protocol in operation

Two generic forms of pipelined protocols:

- **go-Back-N**
- **selective repeat**

Fausto Marcantoni

Chapter 3 TRANSPORT PROTOCOL

3.62

62

Reti di elaboratori

Pipelining: increased utilization

first packet bit transmitted, $t = 0$

last bit transmitted, $t = L/R$

RTT

ACK arrives, send next packet, $t = RTT + L/R$

receiver

first packet bit arrives

last packet bit arrives, send ACK

last bit of 2nd packet arrives, send ACK

last bit of 3rd packet arrives, send ACK

Increase utilization
by a factor of 3!

$$U_{\text{sender}} = \frac{3 * L / R}{RTT + L / R} = \frac{.024}{30.008} = 0.0008$$

$$U_{\text{sender}} = \frac{L / R}{RTT + L / R} = \frac{.008}{30.008} = 0.00027$$

Fausto Marcantoni

Chapter 3 TRANSPORT PROTOCOL

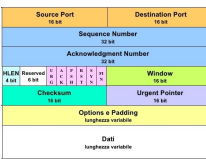
3.63

63

Reti di elaboratori

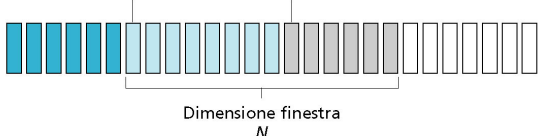
GBN: finestra scorrevole (sliding-window protocol)

- Quattro intervalli nei numeri di sequenza
 1. Pacchetti già trasmessi e riscontrati $[0, \text{base}-1]$
 2. Pacchetti trasmessi e non riscontrati $[\text{base}, \text{nextseqnum}-1]$
 3. Pacchetti disponibili ma non ancora inviati $[\text{nextseqnum}, \text{base}+N-1]$
 4. Pacchetti non disponibili $[\geq \text{base}+N]$
- N **non può essere grande a piacere** (controllo di flusso e congestione)
- Gamma dei numeri di sequenza $[0, 2^k-1]$ dove **K (32bit) è il numero dei bit disponibili nell'intestazione del pacchetto** per i numeri di sequenza



base

nextseqnum



Dimensione finestra
N

Legenda:

Già riscontrato

Inviato, non ancora riscontrato

Disponibile, non ancora inviato

Non disponibile

Fausto Marcantoni

Chapter 3 TRANSPORT PROTOCOL

3.64

64

Relative sequence number

Modifica → Preferenze → Protocollo → TCP

Transmission Control Protocol, Src Port: 4399, Dst Port: 80, Seq: 1, Ack: 1, Len: 0

Source Port: 4399

Destination Port: 80

[Stream index: 0]

[Conversation completeness: Complete, WITH_DATA (31)]

[TCP Segment Len: 0]

Sequence Number: 1 (relative sequence number)

Sequence Number (raw): 1525978352

[Next Sequence Number: 1 (relative sequence number)]

Acknowledgment Number: 1 (relative ack number)

Acknowledgment number (raw): 1923965645

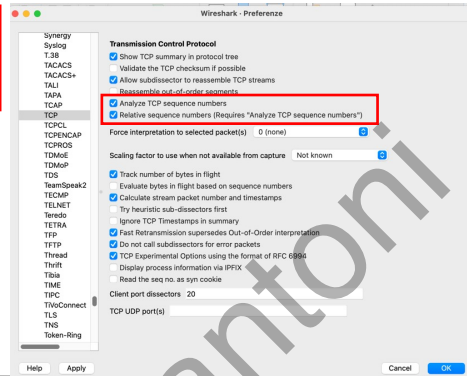
0101 = Header Length: 20 bytes (5)

> Flags: 0x010 (ACK)

Window: 8212

[Calculated window size: 2102272]

[Window size scaling factor: 256]



Fausto Marcantoni

Chapter 3 TRANSPORT PROTOCOL

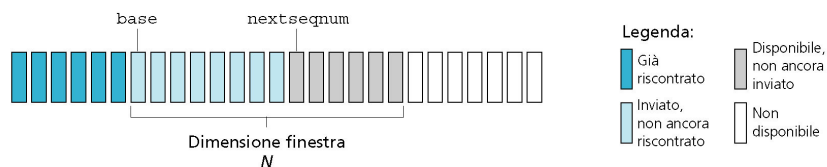
3.65

65

Descrizione di GBN basato su ACK (lato sender)

■ Chiamata da sopra.

- Il sender prima controlla per valutare se la finestra è satura, cioè se ci sono N pacchetti in circolazione, non riscontrati.
- Se la finestra non è satura, viene creato e spedito un pacchetto, e le variabili sono aggiornate nel modo appropriato.
- Se la finestra è satura, il sender ritorna semplicemente i dati allo strato superiore, un' indicazione implicita che la finestra è piena.



Fausto Marcantoni

Chapter 3 TRANSPORT PROTOCOL

3.66

66

Reti di elaboratori

Descrizione di GBN basato su ACK
(lato sender)

- Ricezione di un ACK
 - un riscontro per un pacchetto con numero di sequenza n sarà interpretato come un **riscontro cumulativo**, che indica che tutti i pacchetti con un numero di sequenza fino a n , **n compreso**, sono stati correttamente ricevuti dal receiver.

Fausto Marcantoni

Chapter 3 TRANSPORT PROTOCOL

3.67

67

Reti di elaboratori

Descrizione di GBN basato su ACK
(lato sender)

- Un evento timeout.
 - Il nome del protocollo, "Go-Back-N", è derivato dal comportamento del sender in presenza di pacchetti persi o in ritardo sovrapposti.
 - Si userà ancora un **timer** per recuperare i dati o i pacchetti di riscontro persi.
 - Se interviene un **timeout**, il **sender rispedisce tutti pacchetti che sono già stati spediti, ma che non hanno avuto riscontro**.
 - Se si riceve un ACK, ma ci sono ancora pacchetti spediti, non riscontrati, il timer viene riavviato.
 - Se non sono presenti pacchetti non riscontrati, il timer si arresta.

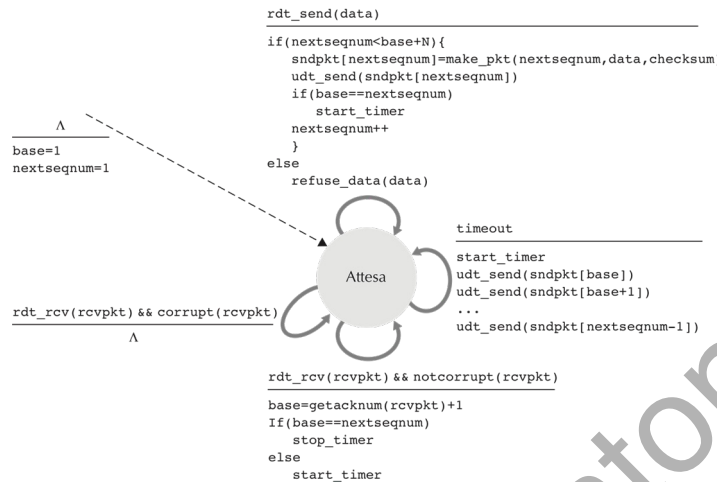
Fausto Marcantoni

Chapter 3 TRANSPORT PROTOCOL

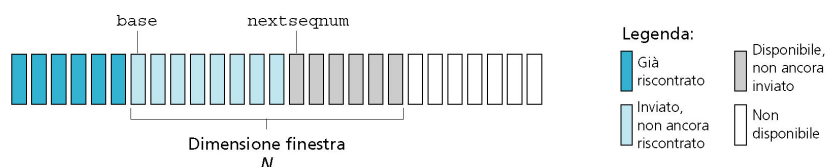
3.68

68

Descrizione di GBN basato su ACK (lato sender)

Descrizione di GBN basato su ACK
(lato receiver)

- Se un pacchetto con un numero di sequenza n è **ricevuto correttamente** ed è in ordine (cioè, i dati inoltrati per ultimi allo strato superiore derivano da un pacchetto con numero di sequenza $n-1$), il **receiver invia un ACK** per il pacchetto n e invia la porzione di dati del pacchetto allo strato superiore.
- In tutti gli altri casi, il receiver **scarta il pacchetto** e rispedisce un ACK per il pacchetto **ricevuto più di recente con l'ordine esatto**.



Reti di elaboratori

Descrizione di GBN basato su ACK (lato receiver)

- il receiver **scarta i pacchetti non in ordine**.
 - Ricordiamo che il receiver deve inviare i dati, nel corretto ordine, allo strato superiore.
 - Supponete ora che si aspetti il pacchetto n , e che arrivi il pacchetto $n + 1$.
 - il receiver potrebbe salvare il pacchetto $n + 1$ e spedirlo allo strato superiore dopo che ha ricevuto e spedito il pacchetto n in ritardo.
 - Se il pacchetto n si è perso, questo e il pacchetto $n + 1$ **saranno probabilmente ritrasmessi** come risultato della regola di ritrasmissione che GBN ha al sender
- Il vantaggio di questo approccio è la semplicità del buffering del receiver: **il receiver non deve mantenere in memoria alcun pacchetto fuori ordine.**

Fausto Marcantoni

Chapter 3 TRANSPORT PROTOCOL

3.71

71

Reti di elaboratori

Descrizione di GBN basato su ACK (lato receiver)

- il **sender** deve mantenere i collegamenti tra la parte superiore e quella inferiore della sua finestra e la posizione di nextseqnum all'interno di questa finestra
- la sola parte di informazione che il **receiver** ha la necessità di mantenere è il **numero di sequenza del prossimo pacchetto in ordine**.

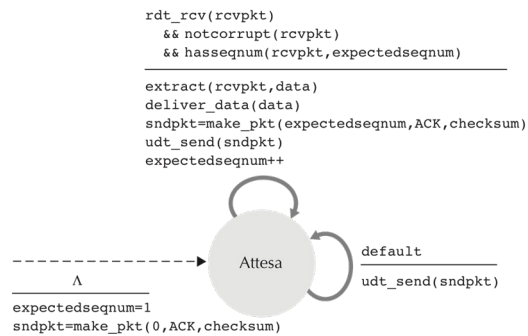
Fausto Marcantoni

Chapter 3 TRANSPORT PROTOCOL

3.72

72

Descrizione di GBN basato su ACK (lato receiver)



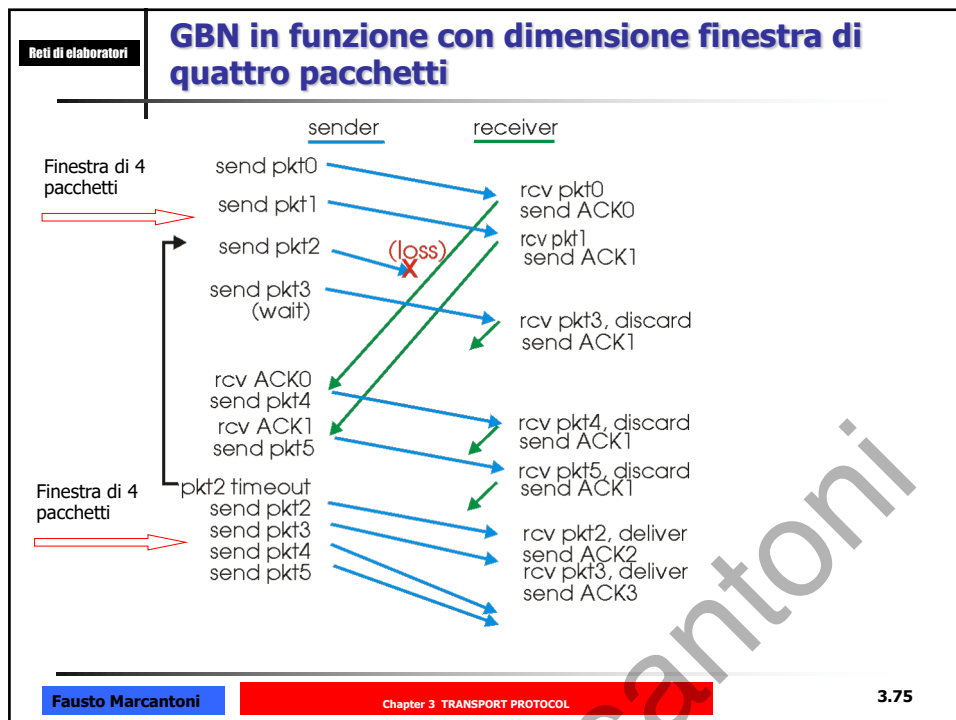
GBN considerazioni finali

Azioni del mittente GBN:

- Invocazione dall'alto
- Ricezione di un ACK → **acknowledgment cumulativo**
- Evento di timeout

Azioni del destinatario GBN:

- Se un pacchetto con numero di sequenza n viene ricevuto correttamente ed è in ordine → manda un ACK per quel pacchetto e consegna i suoi dati al livello superiore
- Oppure il destinatario scarta i pacchetti e rimanda un ACK per il pacchetto in ordine ricevuto più di recente
- Il destinatario scarta i pacchetti fuori sequenza



75

Reti di elaboratori

GBN – SR selective-repeat

Il protocollo GBN incorpora quasi tutte le tecniche relative a TCP per il trasferimento dati affidabile:

- ✓ l'uso di numeri di sequenza
- ✓ riscontri cumulativi
- ✓ checksum
- ✓ operazioni di timeout/ritrasmissione

Ripetizione selettiva

I **protocolli a ripetizione selettiva** (SR, *selective-repeat protocol*) **evitano le ritrasmissioni non necessarie facendo ritrasmettere al mittente solo quei pacchetti su cui esistono sospetti di errore** (ossia, smarrimento o alterazione)

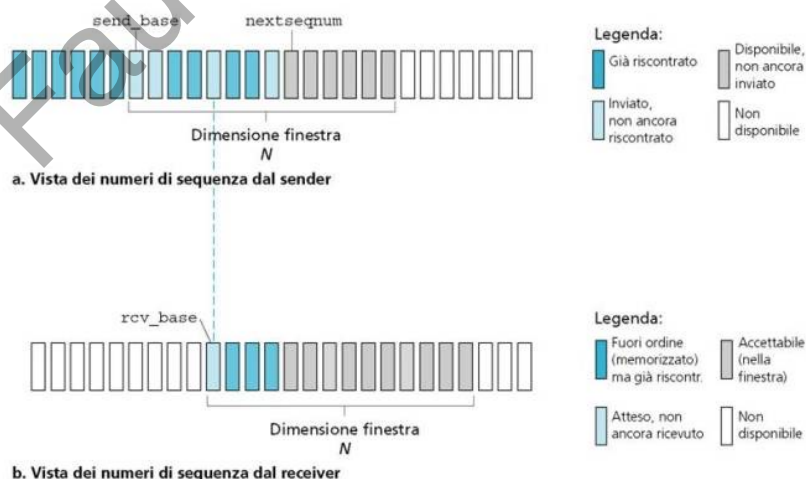
Fausto Marcantoni **Chapter 3 TRANSPORT PROTOCOL** **3.76**

76

Ripetizione selettiva

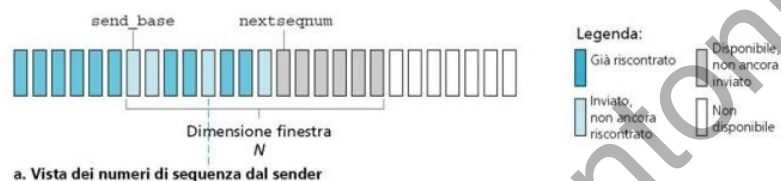
- Il protocollo GBN può riempire la pipeline
- Quando la **dimensione della finestra** ed il prodotto **larghezza di banda * ritardo** sono grandi → nella pipeline possono trovarsi numerosi pacchetti
 - Numerosità dovuta alla ritrasmissione dei pacchetti ricevuti correttamente ma non ancora riscontrati
- SOLUZIONE → Ripetizione selettiva
- Evitano le ritrasmissioni non necessarie
 - Receiver dovrà riscontrare individualmente i pacchetti
 - Visioni diverse della finestra dal lato sender e dal lato receiver

SR: numeri di sequenza visti dal sender e dal receiver



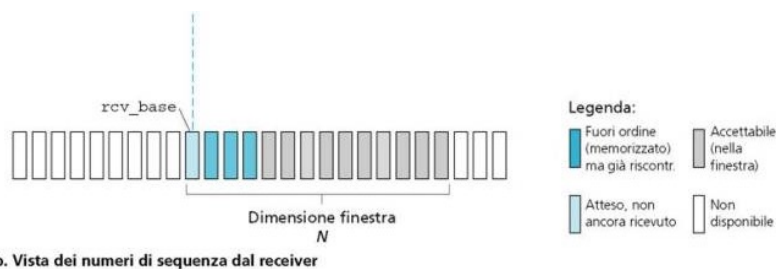
Eventi ed azioni del sender

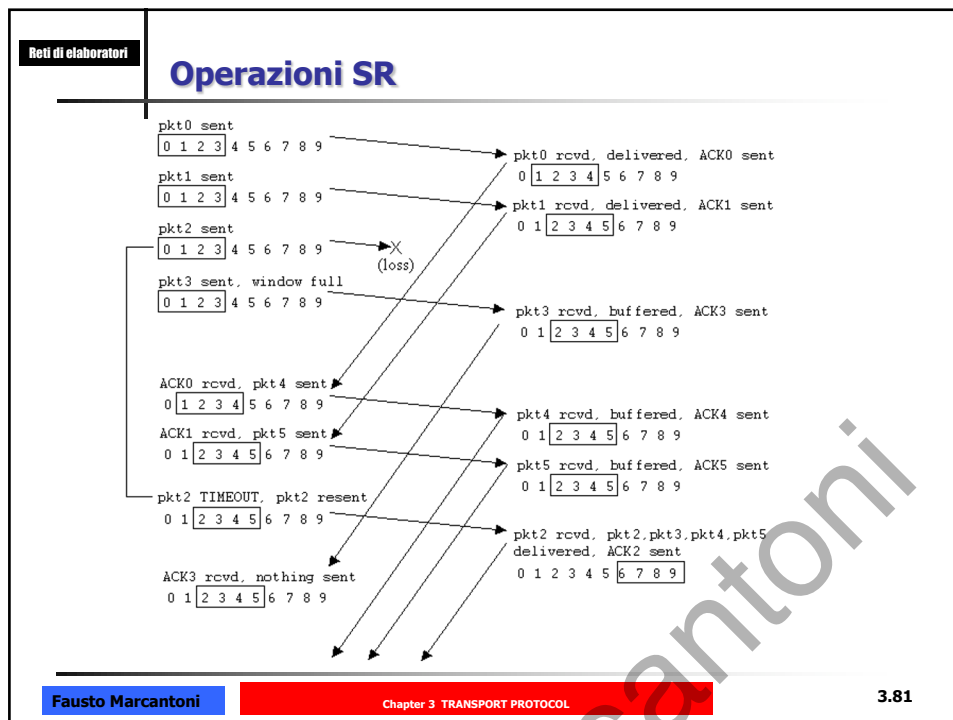
- Dati ricevuti da sopra
 - Numero di sequenza disponibile dentro la finestra → impacchettamento e spedizione
 - Altrimenti o ignora o li salva
- Timeout
 - Timer logico su ogni pacchetto per ritrasmettere solo questo pacchetto
- ACK ricevuti
 - Contrassegna il pacchetto corrispondente come ricevuto quando è nella finestra
 - Se numero di sequenza = send_base la finestra si sposta in avanti al successivo pacchetto non riscontrato
 - Se scorrendo la finestra trova pacchetti non trasmessi che ora rientrano nella finestra, li trasmette



Eventi ed azioni del receiver

- Ricevuto integro un pacchetto dentro la finestra del receiver
 - Invio al sender di un pacchetto selettivo ACK
 - Se il pacchetto non era stato ricevuto viene salvato
 - Se il numero di sequenza del pacchetto è uguale alla base della finestra del receiver allora questo pacchetto ed i successivi in sequenza salvati vengono inviati allo strato superiore
 - La finestra del receiver si sposta in avanti di tanti quanti ne ha inviati allo strato superiore
- Numero di sequenza compreso tra rcv_base-N e rcv_base-1
 - ACK selettivo anche se è stato già riscontrato (*serve per far scorrere la finestra del sender quando esso non riceve un ACK per il send_base*)
- Ignora il pacchetto





81

Reti di elaboratori

Go-back-N $\leftrightarrow$ Selective Repeat

Go-back-N

- ✓ dimensione della finestra W: non invia più di W segmenti non riscontrati
- ✓ acknowledgement cumulativi
- ✓ la ricezione di un numero di sequenza n implica che tutti i segmenti fino ad n sono stati ricevuti
- ✓ timeout: ritrasmette tutti i segmenti non riscontrati

Selective Repeat (SR)

- ✓ riscontra i singoli segmenti
- ✓ finestra del mittente: N segmenti a partire dal primo pacchetto non riscontrato
- ✓ un timer per ogni pacchetto: ritrasmette solo un pacchetto al timeout
- ✓ finestra del ricevente: N segmenti a partire dal primo pacchetto mancante
- ✓ **il ricevente bufferizza i segmenti riscontrati, ma fuori ordine**
- ✓ il ricevente consegna all'applicazione i segmenti, in ordine

Fausto Marcantoni Chapter 3 TRANSPORT PROTOCOL 3.82

82

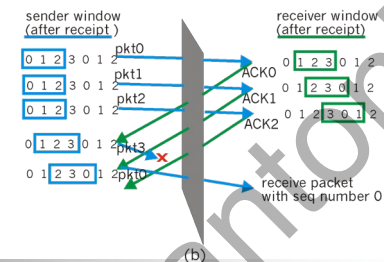
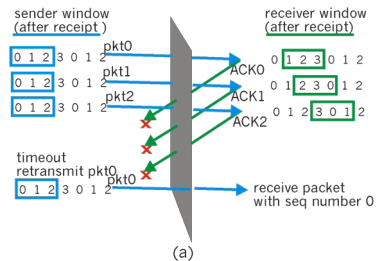
Selective repeat: dilemma

Example:

- seq #'s: 0, 1, 2, 3
- window size=3
- Il receiver non vede differenze nei due scenari!
- incorrettamente passa dati duplicati come nel caso (a)

Q: Quale relazione esiste tra seq # size e window size?

La dimensione della finestra deve essere inferiore o uguale alla metà dello spazio dei numeri di sequenza



Fausto Marcantoni

Chapter 3 TRANSPORT PROTOCOL

3.83

83

Go-Back-N Protocol Demo

Send New Stop Animation Faster Slower Kill Packet Reset

Sender

Base = 5
NextSeq = 9

Receiver

Packet 7 received. Acknowledge sent.

Packet Acknowledge Received Pack Selected

Esempio:

<http://computerscience.unicam.it/marcantoni/reti/applet/GoBackProtocol/goback.html>

https://www2.tkn.tu-berlin.de/teaching/rn/animations/gbn_sr/

<https://nikhilsahu.me/Sliding-window-simulator/GoBackn.html>

Fausto Marcantoni

Chapter 3 TRANSPORT PROTOCOL

3.84

84

Reti di elaboratori

Selective Repeat Protocol Demo

Send New
Pause
Kill Packet/Ack
Reset

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19

Sender
(Send Window Size = 5)
Timeout = 10s

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19

Receiver

Packet
Received
Ack
Ack Received
Selected
Buffered

https://www2.tkn.tu-berlin.de/teaching/rn/animations/gbn_sr/
<https://nikhilsahu.me/Sliding-window-simulator/SelectiveRepeat.html>

Fausto Marcantoni
Chapter 3 TRANSPORT PROTOCOL
3.85

85

Reti di elaboratori

Sommario dei meccanismi per il trasferimento affidabile

Meccanismo	Uso e Commenti
Checksum	Utilizzato per rilevare errori sui bit in un pacchetto trasmesso.
Timer	Serve a far scadere un pacchetto e ritrasmetterlo, forse perché il pacchetto (o il suo ACK) si è smarrito all'interno del canale. I timeout si possono verificare per via dei ritardi anziché degli smarrimenti (timeout prematuro), o quando il pacchetto è stato ricevuto dal destinatario, ma è andato perduto il relativo ACK dal destinatario al mittente. Per questi motivi il destinatario può ricevere copie duplicate di un pacchetto.
Numero di sequenza	Usato per numerare sequenzialmente i pacchetti di dati che fluiscono tra mittente e destinatario. Le discontinuità nei numeri di sequenza di pacchetti ricevuti consentono al destinatario di rilevare i pacchetti persi. I pacchetti con numero di sequenza ripetuto consentono al destinatario di rilevare pacchetti duplicati.

Fausto Marcantoni
Chapter 3 TRANSPORT PROTOCOL
3.86

86

Reti di elaboratori

Sommario dei meccanismi per il trasferimento affidabile

Meccanismo	Uso e Commenti
Acknowledgment (ACK)	Utilizzato dal destinatario per comunicare al mittente che un pacchetto o un insieme di pacchetti sono stati ricevuti correttamente. Gli acknowledgement trasporteranno generalmente i numeri di sequenza del pacchetto o dei pacchetti da confermare. A seconda del protocollo, i riscontri possono essere individuali o cumulativi.
Acknowledgment negativo (NAK)	Usato dal destinatario per comunicare al mittente che un pacchetto non è stato ricevuto correttamente. Gli acknowledgement negativi trasporteranno generalmente il numero di sequenza del pacchetto che non è stato ricevuto correttamente.

Fausto Marcantoni

Chapter 3 TRANSPORT PROTOCOL

3.87

87

Reti di elaboratori

Sommario dei meccanismi per il trasferimento affidabile

Meccanismo	Uso e Commenti
Finestra e pipelining	Il mittente può essere forzato a inviare soltanto pacchetti con numeri di sequenza che ricadono in un determinato intervallo. Consentendo a più pacchetti di essere trasmessi e non aver ancora ricevuto acknowledgement si può migliorare l'utilizzo del canale rispetto alla modalità operativa stop-and-wait. L'ampiezza della finestra può essere impostata sulla base della capacità del destinatario di ricevere e memorizzare messaggi in un buffer, su quella del livello di congestione della rete, o su entrambe.

Fausto Marcantoni

Chapter 3 TRANSPORT PROTOCOL

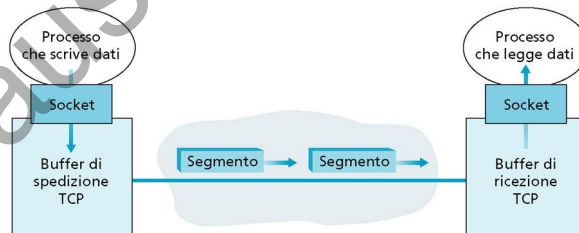
3.88

88

La connessione TCP

- TCP è orientato alla connessione
 - Prima di iniziare la trasmissione i due processi debbono accordarsi
 - Inizializzazione delle variabili di stato (ambo i lati) [verificare con lo sniffer]
- La connessione TCP non è un circuito end to end e non è nemmeno un circuito virtuale
 - Lo stato della connessione risiede solo nei terminali (end system)
- La connessione TCP è full-duplex
- La connessione TCP è point to point
 - Singolo sender e singolo receiver

Bufferizzazione



- RFC 793 specifica che TCP può spedire i dati salvati nel buffer secondo la sua propria convenienza
- **DIMENSIONE MASSIMA DEL SEGMENTO (MSS)**
 - quantità massima di dati da inserire nel segmento
 - **MSS è legato all'MTU delle frame**
- TCP unisce ai dati una intestazione formando i segmenti TCP

Dimensioni del segmento TCP

MSS (Maximum Segment Size) 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
 - ❑ Esempio: ping -f -l 1500 193.205.92.56

ping -f - frammentazione

```
Microsoft Windows [Versione 10.0.26200.7019]
(c) Microsoft Corporation. Tutti i diritti riservati.

C:\Users\stude>ping -l 2000 193.205.92.204

Esecuzione di Ping 193.205.92.204 con 2000 byte di dati:
Risposta da 193.205.92.204: byte=2000 durata=20ms TTL=64
Risposta da 193.205.92.204: byte=2000 durata=1ms TTL=64
Risposta da 193.205.92.204: byte=2000 durata=1ms TTL=64
Risposta da 193.205.92.204: byte=2000 durata=1ms TTL=64

Statistiche Ping per 193.205.92.204:
    Pacchetti: Trasmissi = 4, Ricevuti = 4,
    Persi = 0 (0% persi),
    Tempo approssimativo percorsi andata/ritorno in millisecondi:
    Minimo = 1ms, Massimo = 20ms, Medio = 5ms

C:\Users\stude>
```

```
C:\Users\stude>ping -f -l 2000 193.205.92.204

Esecuzione di Ping 193.205.92.204 con 2000 byte di dati:
C'è necessario frammentare il pacchetto ma DF è attivo.
E' necessario frammentare il pacchetto ma DF è attivo.
E' necessario frammentare il pacchetto ma DF è attivo.
E' necessario frammentare il pacchetto ma DF è attivo.

Statistiche Ping per 193.205.92.204:
    Pacchetti: Trasmissi = 4, Ricevuti = 0,
    Persi = 4 (100% persi),

C:\Users\stude>
```

```
faustomarcantoni@MacBook-Pro-di-Fausto ~ % ping -s 2000 193.205.92.204
PING 193.205.92.204 (193.205.92.204): 2000 data bytes
2000 bytes from 193.205.92.204: icmp_seq=0 ttl=64 time=27.613 ms
2000 bytes from 193.205.92.204: icmp_seq=1 ttl=64 time=1.321 ms
2000 bytes from 193.205.92.204: icmp_seq=2 ttl=64 time=1.492 ms
2000 bytes from 193.205.92.204: icmp_seq=3 ttl=64 time=1.244 ms
^C
--- 193.205.92.204 ping statistics ---
4 packets transmitted, 4 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 1.244/7.893/27.613/11.328 ms
faustomarcantoni@MacBook-Pro-di-Fausto ~ %
```

```
faustomarcantoni@MacBook-Pro-di-Fausto ~ % ping -f -s 2000 193.205.92.204
ping: -f flag: Operation not permitted
faustomarcantoni@MacBook-Pro-di-Fausto ~ %
```

Reti di elaboratori

MTU standard

Tipo di rete	MTU (byte)
Hyperchannel/link	65535
Token ring IBM (16mbps)	17914
Token ring IEEE 802.5 (4mbps)	4464
FDDI	4352
Ethernet	1500
X.25	576
Point to point (low delay)	296

Per cambiare il valore del MTU su Windows è sufficiente cambiare la voce `HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\Class\NetTrans\0002` nel registro di sistema mentre su Linux possiamo editare il file `/etc/ppp/options` per la connessione ad Internet oppure usare `ifconfig <nome interfaccia> mtu <valore del mtu>` per il transito dei pacchetti in locale.

Fausto Marcantoni

Chapter 3 TRANSPORT PROTOCOL

3.93

93

Reti di elaboratori

MTU in Windows

netsh interface ipv4 show subinterfaces

```

C:\Users\fausto.mfausto>netsh interface ipv4 show subinterfaces

MTU      Stato Media Sense  Byte in ingresso  Byte in uscita  Interfaccia
-----
1500     1      0      0      1097808  VirtualBox Host-Only Network
1500     1      802825 1477735 Wi-Fi - Interna
1500     5      0      0      0      Connessione alla rete locale (LAN)* 7
1500     1      0      1097684 VMware Network Adapter VMnet1
1500     5      0      0      0      Connessione alla rete locale (LAN)* 11
4294967295 1      1      0      523598  Loopback Pseudo-Interface 1
1500     1      818475103 99334269 Ethernet
1500     1      0      1099259 VMware Network Adapter VMnet8
65536    1      0      1232797 VirtualBox Host-Only Network #3
65536    1      0      1098354 VirtualBox Host-Only Network #2
1500     5      0      0      0      Connessione di rete Bluetooth
1500     1      0      1251720 VMware Network Adapter VMnet10

C:\Users\fausto.mfausto>

```

Fausto Marcantoni

Chapter 3 TRANSPORT PROTOCOL

3.94

94

MTU in MacOS

ifconfig | grep -e "^[a-z]" -e "mtu"

```
faustomarcantoni@MacBook-Pro-di-Fausto ~ % ifconfig | grep -e "^[a-z]" -e "mtu"
lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> mtu 16384
gif0: flags=8010<POINTOPOINT,MULTICAST> mtu 1280
stf0: flags=0<> mtu 1280
anp10: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
anp12: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
anp11: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
en4: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
en5: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
en6: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
en1: flags=8963<UP,BROADCAST,SMART,RUNNING,PROMISC,SIMPLEX,MULTICAST> mtu 1500
en2: flags=8963<UP,BROADCAST,SMART,RUNNING,PROMISC,SIMPLEX,MULTICAST> mtu 1500
en3: flags=8963<UP,BROADCAST,SMART,RUNNING,PROMISC,SIMPLEX,MULTICAST> mtu 1500
ap1: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
en0: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
bridge0: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
awd10: flags=8822<BROADCAST,SMART,SIMPLEX,MULTICAST> mtu 1500
llw0: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
utun6: flags=8050<POINTOPOINT,RUNNING,MULTICAST> mtu 1280
utun0: flags=8051<UP,POINTOPOINT,RUNNING,MULTICAST> mtu 1500
utun1: flags=8051<UP,POINTOPOINT,RUNNING,MULTICAST> mtu 1380
utun2: flags=8051<UP,POINTOPOINT,RUNNING,MULTICAST> mtu 2080
utun3: flags=8051<UP,POINTOPOINT,RUNNING,MULTICAST> mtu 2080
utun4: flags=8051<UP,POINTOPOINT,RUNNING,MULTICAST> mtu 1380
utun5: flags=8051<UP,POINTOPOINT,RUNNING,MULTICAST> mtu 1380
utun7: flags=8051<UP,POINTOPOINT,RUNNING,MULTICAST> mtu 1500
en9: flags=8963<UP,BROADCAST,SMART,RUNNING,PROMISC,SIMPLEX,MULTICAST> mtu 1500
en7: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
vmenet0: flags=8963<UP,BROADCAST,SMART,RUNNING,PROMISC,SIMPLEX,MULTICAST> mtu 1500
bridge100: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
vmenet1: flags=8963<UP,BROADCAST,SMART,RUNNING,PROMISC,SIMPLEX,MULTICAST> mtu 1500
bridge101: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
vmenet2: flags=8963<UP,BROADCAST,SMART,RUNNING,PROMISC,SIMPLEX,MULTICAST> mtu 1500
vmenet3: flags=8963<UP,BROADCAST,SMART,RUNNING,PROMISC,SIMPLEX,MULTICAST> mtu 1500
bridge102: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
faustomarcantoni@MacBook-Pro-di-Fausto ~ %
```

MTU in Windows

How to set MTU size to 100 on Windows 10

```
Administrator: Prompt dei comandi
C:\Users\fausto.mfausto>netsh interface ipv4 set subinterface "Ethernet" mtu=352
OK.

C:\Users\fausto.mfausto>netsh interface ipv4 set subinterface "Ethernet" mtu=1500
OK.

C:\Users\fausto.mfausto>netsh interface ipv4 set subinterface "Ethernet" mtu=100
OK.

C:\Users\fausto.mfausto>
```

Looks like Windows do not allow MTU size smaller than 352 bytes:

<https://support.microsoft.com/en-us/topic/packet-loss-occurs-when-mtu-is-below-576-and-pmtu-discovery-is-enabled-in-windows-e7a54c21-6202-9788-fb24-99530c0e9d64>

Manage Network Adapter Settings via PowerShell

Version: Windows Server 2025 and Windows 11 PowerShell

Search

Commands

- Disable-NetAdapter
- Disable-NetAdapterBinding
- Disable-NetAdapterChecksumOffload
- Disable-NetAdapterEncapsulatedPacketTxOffload

Download PDF

NetAdapter Module

This reference provides cmdlet descriptions and syntax for all Network Adapter-specific cmdlets. It lists the cmdlets in alphabetical order based on the verb at the beginning of the cmdlet.

NetAdapter Cmdlets

Cmdlet	Description
Disable-NetAdapter	Disables a network adapter.
Disable-NetAdapterBinding	Disables a binding to a network adapter.
Disable-NetAdapterChecksumOffload	Disables the selected checksum offloads on the network adapter.

<https://learn.microsoft.com/en-us/powershell/module/netadapter/?view=windowsserver2025-ps>

networksetup - MacOS

networksetup – Change Network Settings from the Command Line

```
networksetup -listnetworkserviceorder
networksetup -listallnetworkservices
networksetup -listallhardwareports
networksetup -detectnewhardware
networksetup -getcomputername
```

<https://www.unix.com/man-page/osx/8/networksetup/>

ifconfig | grep mtu

MTU in Linux

```

studente@studente-virtual-machine: ~
Unpacking net-tools (1.60+git20181103.0eebece-1ubuntu5) ...
Setting up net-tools (1.60+git20181103.0eebece-1ubuntu5) ...
Processing triggers for man-db (2.10.2-1) ...
studente@studente-virtual-machine:~$ ifconfig
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 193.205.92.226 netmask 255.255.255.0 broadcast 193.205.92.255
    inet6 fe80::ca25:a39f:a116:bb3b prefixlen 64 scopeid 0x20<link>
    ether 00:0c:29:26:c3:b0 txqueuelen 1000 (Ethernet)
    RX packets 68604 bytes 25416197 (25.4 MB)
    RX errors 0 dropped 13 overruns 0 frame 0
    TX packets 3707 bytes 424947 (424.9 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 864 bytes 100575 (100.5 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 864 bytes 100575 (100.5 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

studente@studente-virtual-machine:~$

```

To change the MTU size of an interface, use the following syntax:

```

$ ifconfig <Interface_name> mtu <mtu_size> up
$ ifconfig ens33 mtu 1000 up

```

MTU in Linux

```

$ ip link list
$ ip link set dev eth0 mtu 1400
$ service network restart

```

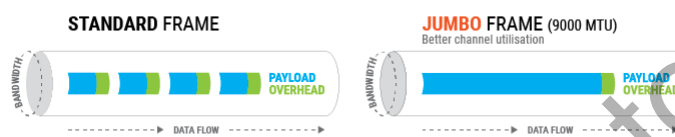
```

studente@studente-virtual-machine: ~
studente@studente-virtual-machine:~$ ip link list
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT
    group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP mode
    DEFAULT group default qlen 1000
    link/ether 00:0c:29:26:c3:b0 brd ff:ff:ff:ff:ff:ff
    altname enp2s1
studente@studente-virtual-machine:~$

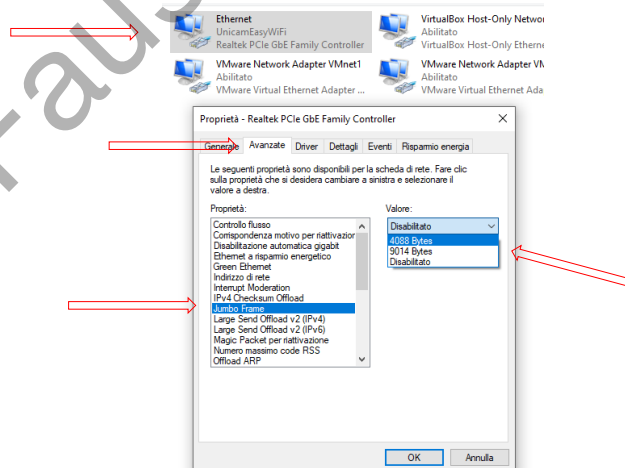
```

Jumbo frames and maximum transmission unit

- ✓ Jumbo frames are when the Ethernet MTU is larger than the standard 1,500 bytes.
- ✓ This may be possible on fast Ethernet links, such as with a gigabit LAN, and can be as large as **9,000 bytes**.
- ✓ Using jumbo packets can reduce the overhead and increase efficiency of data transmission.



Abilitare Jumbo Frame in Windows



Reti di elaboratori

Ping

Usage: ping [-t] [-a] [-n count] [-l size] [-f] [-i TTL] [-v TOS] [-r count] [-s count] [[-j host-list] | [-k host-list]] [-w timeout] [-R] [-S srcaddr] [-4] [-6 target_name]

Options:

-t	Pings the specified host until stopped. To see statistics and continue - Type Control-Break; To stop - press Ctrl + C.
-a	Resolve addresses to hostnames.
-n	count Number of echo requests to send.
-l size	Send buffer size.
-f	Set Don't Fragment flag in packet (IPv4-only).
-i	TTL Time To Live.
-v	TOS Type Of Service (IPv4-only. This setting has been deprecated and has no effect on the type of service field in the IP Header).
-r count	Record route for count hops (IPv4-only).
-s count	Timestamp for count hops (IPv4-only).
-j host-list	Loose source route along host-list (IPv4-only).
-k host-list	Strict source route along host-list (IPv4-only).
-w timeout	Timeout in milliseconds to wait for each reply.
-R	Use routing header to test reverse route also (IPv6-only). Per RFC 5095 the use of this routing header has been deprecated. Some systems may drop echo requests if this header is used.
-S srcaddr	Source address to use.
-4	Force using IPv4.
-6	Force using IPv6.

Fausto Marcantoni

Chapter 3 TRANSPORT PROTOCOL

3.103

103

Reti di elaboratori

Wireshark: MSS

Frame 1067: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0

Ethernet II, Src: AsustekC_0a:e6:1d (d8:50:e6:0a:e6:1d), Dst: CiscoInc_39:1d:1d:00:00:00

Internet Protocol Version 4, Src: 193.205.92.123 (193.205.92.123), Dst: 68.87.7.14

Transmission Control Protocol, Src Port: 14507 (14507), Dst Port: 443 (443)

Source Port: 14507 (14507)

Destination Port: 443 (443)

[Stream index: 22]

[TCP Segment Len: 0]

Sequence number: 0 (relative sequence number)

Acknowledgment number: 0

Header Length: 32 bytes

... 0000 0000 0010 = Flags: 0x002 (SYN)

Window size value: 8192

[Calculated window size: 8192]

Checksum: 0xe59b [validation disabled]

Urgent pointer: 0

Options: (12 bytes), Maximum segment size, No-operation (NOP), window scale

Maximum segment size: 1460 bytes

Kind: Maximum segment size (2)

Length: 4

MSS Value: 1460

No-Operation (NOP)

Window scale: 8 (multiply by 256)

No-Operation (NOP)

No-Operation (NOP)

No-Operation (NOP)

TCP SACK Permitted Option: True

0000 00 0e 38 39 10 3f d8 50 e6 0a e6 1d 08 00 45 00 ..89.?P.....E.

0010 00 34 40 5a 40 00 80 06 34 71 c1 cd 5c 7b 44 e8 .4@Z@...4q..{D.

0020 22 c8 38 ab 01 bb 6e 72 3a a3 00 00 00 00 80 02 ".8...nr.....

0030 20 00 e5 9b 00 00 02 04 05 b4 01 03 03 08 01 01B.....

0040 04 02

Fausto Marcantoni

Chapter 3 TRANSPORT PROTOCOL

3.104

104

MSS (Maximum Segment Size)

- Se il TCP trasmittente non specifica nel primo segmento (SYN) il valore di MSS, si usa il valore di default di MSS pari a 536 byte
- Un valore di MSS di 536 byte, sommata ai 40 byte di header IP + TCP, dà luogo a un datagramma IP di dimensione pari a 576 byte, ovvero un datagramma che tutti i sistemi di Internet devono necessariamente accettare e gestire
- Oggi è diffusa la tendenza a usare valori molto più grandi dell'MSS per aumentare l'efficienza e la velocità del collegamento

MSS (Maximum Segment Size)

Primo pacchetto della connessione

MSS proposto dal client

connessione web.pcapng

Reti di elaboratori

MSS (Maximum Segment Size)

Pacchetto di risposta della connessione

MSS dichiarato dal server

1460=1500-20

Fausto Marcantoni

Chapter 3 TRANSPORT PROTOCOL

3.107

107

Reti di elaboratori

TCP: header del segmento

32 bit

Numero di porta di origine

Numero di porta di destinazione

Numero di sequenza

Numero di acknowledgement

Lungh. intest. Non usato URG ACK PSH RST SYN FIN Finestra di ricezione

Checksum Internet Puntatore ai dati urgenti

Opzioni

Dati

Fausto Marcantoni

Chapter 3 TRANSPORT PROTOCOL

3.108

108

Flags header TCP

SYN - se settato a 1 indica che l'host mittente del segmento vuole **aprire una connessione** TCP con l'host destinatario e specifica nel campo Sequence number il valore dell'Initial Sequence Number (ISN); ha lo scopo di sincronizzare i numeri di sequenza dei due host. L'host che ha inviato il SYN deve attendere dall'host remoto un pacchetto SYN/ACK.

ACK - se settato a 1 indica che il campo Acknowledgment number **è valido**;

URG - se settato a 1 indica che nel flusso sono presenti **dati urgenti** alla posizione (offset) indicata dal campo Urgent pointer. Urgent Pointer punta alla fine dei dati urgenti;

PSH - se settato a 1 indica che i dati in arrivo non devono essere bufferizzati ma **passati subito** ai livelli superiori dell'applicazione;

RST - se settato a 1 indica che la connessione **non è valida**; viene utilizzato in caso di grave errore; a volte utilizzato insieme al flag ACK per la chiusura di una connessione.

FIN - se settato a 1 indica che l'host mittente del segmento vuole **chiudere la connessione** TCP aperta con l'host destinatario. Il mittente attende la conferma dal ricevente (con un FIN-ACK). A questo punto la connessione è ritenuta chiusa per metà: l'host che ha inviato FIN non potrà più inviare dati, mentre l'altro host ha il canale di comunicazione ancora disponibile. Quando anche l'altro host invierà il pacchetto con FIN impostato la connessione, dopo il relativo FIN-ACK, sarà considerata completamente chiusa.

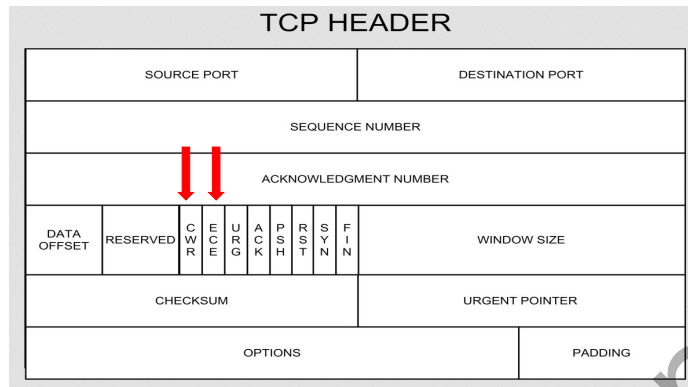
TCP: segmento (esempio)

The screenshot shows a Wireshark capture of a network packet. The packet list on the left shows a packet of type TCP. The packet details pane on the right shows the structure of the TCP segment. Red arrows point to specific fields in the packet details pane:

- Source port: http (80)
- Destination port: 4712 (4712)
- Sequence number: 1 (relative sequence number)
- Acknowledgement number: 451 (relative ack number)
- Header length: 20 bytes
- Flags: 0x10 (ACK)
- Window size: 11764
- Checksum: 0x6144 [correct]
- [Good Checksum: True]
- [Bad Checksum: False]
- [Seq/Ack analysis]
- [This is an ACK to the segment in frame 75]
- [The RTT to ACK the segment was: 0.042539000 seconds]

The packet bytes pane at the bottom shows the raw data of the packet in hexadecimal and ASCII.

Flags header TCP CWR - ECE



CWR (Congestion Window Reduced) - se impostato a 1 indica che l'host sorgente ha ricevuto un segmento TCP con il flag ECE impostato a 1 (aggiunto all'header in RFC 3168).

ECE [ECN (Explicit Congestion Notification) -Echo] - se impostato a 1 indica che l'host supporta ECN durante il 3-way handshake (aggiunto all'header in RFC 3168).

Wireshark filtro ECN e/o CWR

1 Answer

active oldest votes

You can load the packet trace in Wireshark, and apply the filter

```
tcp.flags.ecn==1
```

to see only packets with the ECN-Echo bit set in the TCP header, and

```
tcp.flags.cwr==1
```

to see only the packets with the ECN-CWR (Congestion Window Reduced) bit set in the TCP header.

[Here is the list of TCP filters](#) in Wireshark.

You may also be interested in filtering on ECN-CE flags in the IP header, which you can do with

```
ip.dsfield.ecn == 0x03
```

[Here is the list of IP filters](#) in Wireshark.

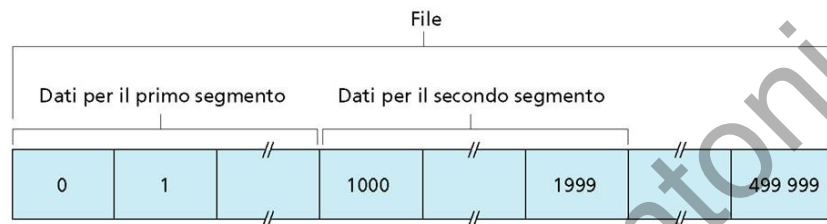
To see what it *should* look like, you can download [this sample capture](#) from the [Wireshark Sample Captures](#) page. This capture includes packets with ECN events.

<https://www.wireshark.org/docs/dfref/i/ip.html>

Numeri di sequenza

- TCP vede i dati come flusso di byte non strutturati
- Numero di sequenza per un segmento è **il numero, all'interno del flusso, del primo byte del segmento**
 - Al primo segmento numero di sequenza 0
 - Al secondo numero di sequenza 1000 e così via

Flusso di 500.000 byte con MSS (maximum segment size) = 1000



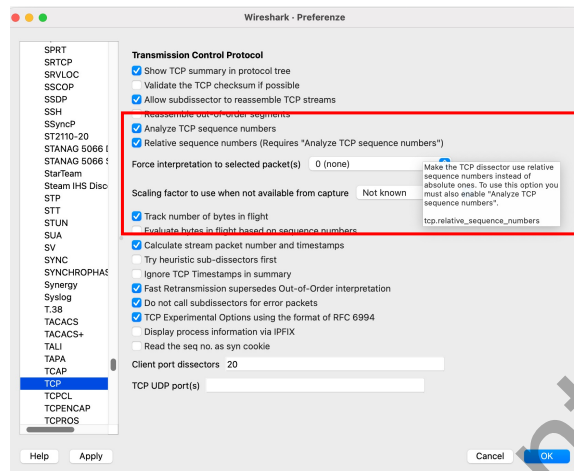
Numeri di riscontro

Il numero di riscontro che un host inserisce nel suo segmento è il **numero di sequenza del prossimo byte** che l'host si aspetta dal suo corrispondente

- TCP riscontra solo i byte fino al primo mancante nel flusso
→ riscontro cumulativo
- I numeri di sequenza iniziale sono scelti a caso
- Esempio:
 - Ricevo due segmenti con byte 0,535 e byte 900,1000
 - Che numero di riscontro inserisco nell'intestazione del mio prossimo segmento ?
 - Che faccio del segmento fuori sequenza ?

Numeri di sequenza in wireshark

https://wiki.wireshark.org/TCP_Relative_Sequence_Numbers



ISN Initial Sequence Number

Initial Sequence Number

- Ogni volta che si instaura una nuova connessione l'host trasmittente sceglie in modo pseudo-casuale numero di sequenza iniziale (ISN) di 32 bit da cui partire, ponendolo uguale al valore del suo contatore binario, sincronizzato al clock locale
- Il numero di sequenza si sceglie in modo pseudo-casuale tra 0 e $(2^{32})-1 = 4.294.967.295$

$$\text{ISN} = (\text{tempo_corrente_in_ms} \times \text{costante}) \bmod 2^{32}$$

da provare in shell

```
while true; do
  echo $(( ( $(date +%s) * 1103515245 % 4294967296 ))
  sleep 1
done

echo $(( ( $(date +%s) * 1000 + $(jot -r 1 0 999) ) * 1103515245 ) % 4294967296 ))
```

Reti di elaboratori

MSL *Maximun Segment Lifetime*

Maximum Segment Lifetime

Storicamente, gli standard RFC 793 (TCP originale) dicono:

"The Maximum Segment Lifetime is the maximum time a segment can exist in the internetwork system before being discarded."

E raccomandano un valore tipico:

MSL = 2 minuti

(anche se non viene imposto rigidamente — è solo una raccomandazione).

Sistema operativo	Valore tipico di MSL	Timeout TIME_WAIT (≈ 2×MSL)
Linux	60 s	120 s
Windows	120 s	240 s
macOS / BSD	30 s	60 s

Fausto Marcantoni

Chapter 3 TRANSPORT PROTOCOL

3.117

117

Reti di elaboratori

MSL – TIME_WAIT

```

faustomarcantoni -- zsh -- 80x24
tcp4 0 0 mbp-di-fausto.am.61589 1.80.190.35.bc.g.https ESTABLISHED
tcp4 0 0 mbp-di-fausto.am.61588 mil04s50-in-f5.1.https ESTABLISHED
tcp4 0 0 mbp-di-fausto.am.61587 123.208.120.34.b.https ESTABLISHED
tcp4 0 0 mbp-di-fausto.am.61586 216.239.38.223.https ESTABLISHED
tcp4 0 0 mbp-di-fausto.am.61585 edge-star-shv-01.https ESTABLISHED
tcp4 0 0 mbp-di-fausto.am.61584 edge-star-shv-01.https ESTABLISHED
tcp4 0 0 mbp-di-fausto.am.61583 edge-star-shv-01.https ESTABLISHED
tcp4 0 0 mbp-di-fausto.am.61573 216.239.32.223.https CLOSE_WAIT
tcp4 0 0 mbp-di-fausto.am.61525 93.243.107.34.bc.https ESTABLISHED
tcp4 0 0 mbp-di-fausto.am.60426 en-in-f188.1e100.5228 ESTABLISHED
tcp6 0 0 macbook-pro-di-f.60403 fe80::e4d6:24ff:49951 ESTABLISHED
tcp6 0 0 macbook-pro-di-f.60404 fe80::e4d6:24ff:49950 ESTABLISHED
tcp6 0 0 macbook-pro-di-f.60402 fe80::e4d6:24ff:58783 ESTABLISHED
tcp4 0 0 mac.amministrazi.60381 edge-dgw-shv-01-.https ESTABLISHED
tcp4 0 0 mac.amministrazi.60377 edge-star-shv-01.https ESTABLISHED
tcp4 0 0 mac.amministrazi.60373 170.72.245.159.https ESTABLISHED
tcp4 0 0 mac.amministrazi.60370 edge-z-p3-shv-01.https ESTABLISHED
tcp4 0 0 mac.amministrazi.60334 mil04s24-in-f42.https ESTABLISHED
tcp4 24 0 fabriziofornari..53885 172.66.0.165.https CLOSE_WAIT
tcp4 0 0 mbp-di-fausto.am.61602 web-lnx287.ergon.https TIME_WAIT
tcp4 0 0 mbp-di-fausto.am.61601 web-lnx287.ergon.https TIME_WAIT
tcp4 0 0 mbp-di-fausto.am.61599 web-lnx287.ergon.https TIME_WAIT
tcp4 0 0 mbp-di-fausto.am.49990 17.57.146.23.443 ESTABLISHED
udp4 0 0 mbp-di-fausto.am.61858 par21s06-in-f227.https

```

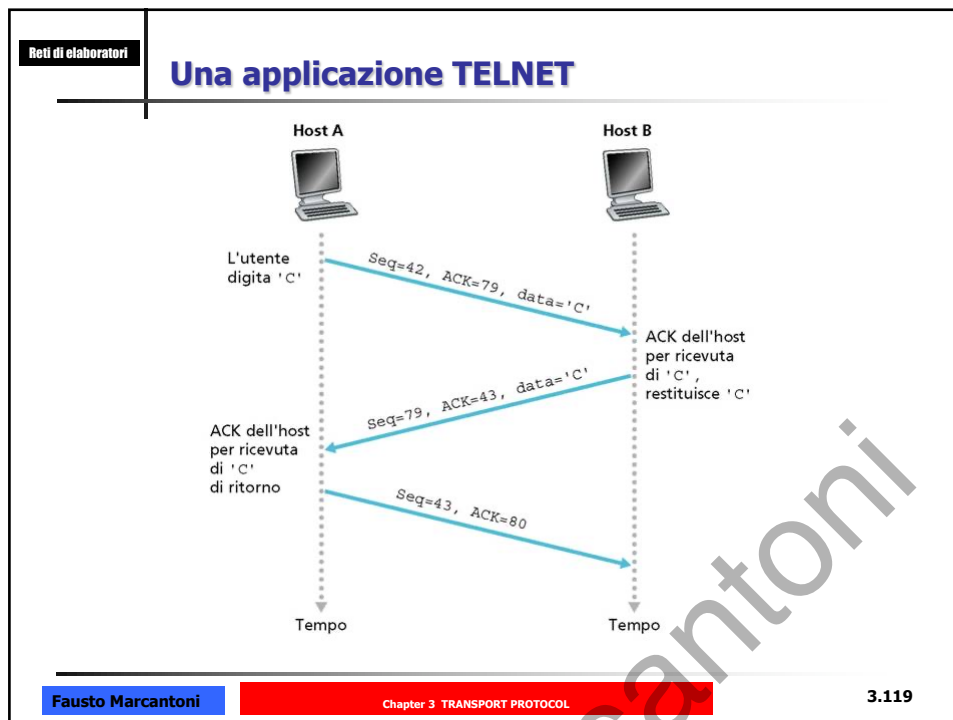
Dopo che una connessione TCP viene chiusa, il punto finale che entra nello stato TIME_WAIT manterrà alcune risorse (come il numero di porta) per un periodo di tempo pari al doppio dell'MSL (noto come attesa 2MSL)

Fausto Marcantoni

Chapter 3 TRANSPORT PROTOCOL

3.118

118



119

Reti di elaboratori

Esempio ssh 1/3

Time	Source	Destination	Protocol	Details
35	193.205.92.161	193.205.92.56	TCP	3900 > 22 [ACK] Seq=80 Ack=48 Win=65103
36	193.205.92.56	193.205.92.161	SSH	Encrypted response packet len=48
37	193.205.92.56	193.205.92.161	SSH	Encrypted request packet len=80
38	193.205.92.161	193.205.92.56	TCP	3900 > 22 [ACK] Seq=160 Ack=96 Win=65055
39	193.205.92.8	193.205.92.248	TCP	netbios-ssn > 2830 [ACK] Seq=0 Ack=0 Win=
40	193.205.92.161	193.205.92.56	SSH	Encrypted request packet len=80
41	193.205.92.56	193.205.92.161	SSH	Encrypted response packet len=48
42	193.205.92.161	193.205.92.56	TCP	3900 > 22 [ACK] Seq=240 Ack=144 Win=6500
43	193.205.92.56	193.205.92.161	SSH	Encrypted response packet len=64
44	193.205.92.56	193.205.92.161	SSH	Encrypted response packet len=112
45	193.205.92.161	193.205.92.56	TCP	3900 > 22 [ACK] Seq=240 Ack=320 Win=6483
46	193.205.92.56	193.205.92.161	SSH	Encrypted response packet len=112
47	193.205.92.56	193.205.92.161	SSH	Encrypted response packet len=112
48	193.205.92.161	193.205.92.56	TCP	3900 > 22 [ACK] Seq=240 Ack=544 Win=6460
49	193.205.92.56	193.205.92.161	SSH	Encrypted response packet len=112

```

p Frame 36 (134 bytes on wire, 134 bytes captured)
p Ethernet II, Src: 00:0f:b0:44:69:10, Dst: 00:0a:e6:76:ca:92
p Internet Protocol, Src Addr: 193.205.92.161 (193.205.92.161), Dst Addr: 193.205.92.56 (193.205.92.56)
v Transmission Control Protocol, Src Port: 3900 (3900), Dst Port: 22 (22), Seq: 80, Ack: 48, Len: 80
  Source port: 3900 (3900)
  Destination port: 22 (22)
  Sequence number: 80 (relative sequence number)
  [Next sequence number: 160 (relative sequence number)]
  Acknowledgement number: 48 (relative ack number)
  Header length: 20 bytes
  p Flags: 0x0018 (PSH, ACK)
  window size: 65103
  Checksum: 0x3cdf (incorrect, should be 0xb9b7)
  p SSH Protocol
  
```

Fausto Marcantoni Chapter 3 TRANSPORT PROTOCOL 3.120

120

Reti di elaboratori

Esempio ssh 2/3

```

35 5.136052 193.205.92.161 193.205.92.56 TCP 3900 > 22 [ACK] seq=80 Ack=48 win=65103 [CHECKSUM
36 5.368259 193.205.92.161 193.205.92.56 SSH Encrypted request packet len=80
37 5.368805 193.205.92.56 193.205.92.161 SSH Encrypted response packet len=48
38 5.537337 193.205.92.161 193.205.92.56 TCP 3900 > 22 [ACK] Seq=160 Ack=96 win=65055 [CHECKSUM
39 5.623335 193.205.92.8 193.205.92.248 TCP netbios-ssn > 2830 [ACK] Seq=0 Ack=0 win=2920 Len
40 5.629221 193.205.92.161 193.205.92.56 SSH Encrypted request packet len=80
41 5.629623 193.205.92.56 193.205.92.161 SSH Encrypted response packet len=48
42 5.737973 193.205.92.161 193.205.92.56 TCP 3900 > 22 [ACK] Seq=240 Ack=144 win=65007 [CHECKSUM
43 5.738667 193.205.92.56 193.205.92.161 SSH Encrypted response packet len=64
44 5.739065 193.205.92.56 193.205.92.161 SSH Encrypted response packet len=112
45 5.739100 193.205.92.161 193.205.92.56 TCP 3900 > 22 [ACK] Seq=240 Ack=320 win=64831 [CHECKSUM
46 5.739241 193.205.92.56 193.205.92.161 SSH Encrypted response packet len=112
47 5.739332 193.205.92.56 193.205.92.161 SSH Encrypted response packet len=112
48 5.739359 193.205.92.161 193.205.92.56 TCP 3900 > 22 [ACK] Seq=240 Ack=544 win=64607 [CHECKSUM
49 5.739431 193.205.92.56 193.205.92.161 SSH Encrypted response packet len=112

```

▸ Frame 37 (102 bytes on wire, 102 bytes captured)
 ▸ Ethernet II, Src: 00:0a:e6:76:ca:92, Dst: 00:0f:b0:44:69:10
 ▸ Internet Protocol, Src Addr: 193.205.92.56 (193.205.92.56), Dst Addr: 193.205.92.161 (193.205.92.161)
 ▸ Transmission Control Protocol, Src Port: 22 (22), Dst Port: 3900 (3900), Seq: 48, Ack: 160, Len: 48

Source port: 22 (22)
 Destination port: 3900 (3900)
 Sequence number: 48 (relative sequence number)
 [Next sequence number: 96 (relative sequence number)]
 Acknowledgement number: 160 (relative ack number)

Header length: 20 bytes
 ▸ Flags: 0x0018 (PSH, ACK)
 Window size: 14144
 Checksum: 0x812e (correct)
 ▸ [SEQ/ACK analysis]
 ▸ SSH Protocol

Fausto Marcantoni

Chapter 3 TRANSPORT PROTOCOL

3.121

121

Reti di elaboratori

Esempio ssh 3/3

```

35 5.136052 193.205.92.161 193.205.92.56 TCP 3900 > 22 [ACK] Seq=80 Ack=48 win=65103 [CHECKSUM
36 5.368259 193.205.92.161 193.205.92.56 SSH Encrypted request packet len=80
37 5.368805 193.205.92.56 193.205.92.161 SSH Encrypted response packet len=48
38 5.537337 193.205.92.161 193.205.92.56 TCP 3900 > 22 [ACK] Seq=160 Ack=96 win=65055 [CHECKSUM
39 5.623335 193.205.92.8 193.205.92.248 TCP netbios-ssn > 2830 [ACK] Seq=0 Ack=0 win=2920 L
40 5.629221 193.205.92.161 193.205.92.56 SSH Encrypted request packet len=80
41 5.629623 193.205.92.56 193.205.92.161 SSH Encrypted response packet len=48
42 5.737973 193.205.92.161 193.205.92.56 TCP 3900 > 22 [ACK] Seq=240 Ack=144 win=65007 [CHECKSUM
43 5.738667 193.205.92.56 193.205.92.161 SSH Encrypted response packet len=64
44 5.739065 193.205.92.56 193.205.92.161 SSH Encrypted response packet len=112
45 5.739100 193.205.92.161 193.205.92.56 TCP 3900 > 22 [ACK] Seq=240 Ack=320 win=64831 [CHECKSUM
46 5.739241 193.205.92.56 193.205.92.161 SSH Encrypted response packet len=112
47 5.739332 193.205.92.56 193.205.92.161 SSH Encrypted response packet len=112
48 5.739359 193.205.92.161 193.205.92.56 TCP 3900 > 22 [ACK] Seq=240 Ack=544 win=64607 [CHECKSUM
49 5.739431 193.205.92.56 193.205.92.161 SSH Encrypted response packet len=112

```

▸ Frame 40 (134 bytes on wire, 134 bytes captured)
 ▸ Ethernet II, Src: 00:0f:b0:44:69:10, Dst: 00:0a:e6:76:ca:92
 ▸ Internet Protocol, Src Addr: 193.205.92.161 (193.205.92.161), Dst Addr: 193.205.92.56 (193.205.92.56)
 ▸ Transmission Control Protocol, Src Port: 3900 (3900), Dst Port: 22 (22), Seq: 160, Ack: 96, Len: 80

Source port: 3900 (3900)
 Destination port: 22 (22)
 Sequence number: 160 (relative sequence number)
 [Next sequence number: 240 (relative sequence number)]
 Acknowledgement number: 96 (relative ack number)

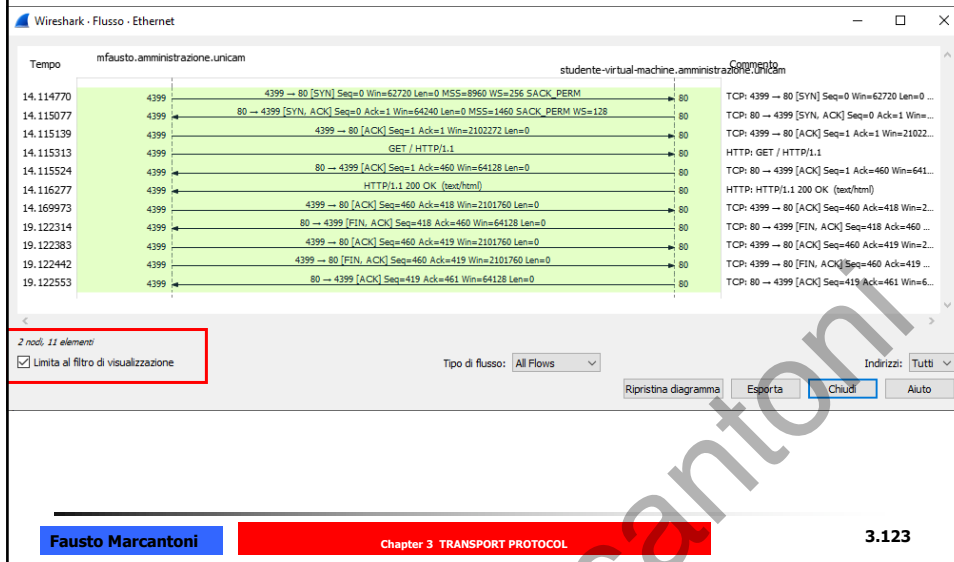
Header length: 20 bytes
 ▸ Flags: 0x0018 (PSH, ACK)
 Window size: 65055
 Checksum: 0x3cdf (incorrect, should be 0xb53e)
 ▸ SSH Protocol

Fausto Marcantoni

Chapter 3 TRANSPORT PROTOCOL

3.122

122



123

Installa la VM metasploitable2

<https://docs.rapid7.com/metasploit/metasploitable-2>

- [illegible]

124

Reti di elaboratori

Esercitazioni

internet reti sicurezza

Esercitazioni

Fausto Marcantoni

Chapter 2 APPLICATION PROTOCOL

2.125

125

Reti di elaboratori

Procedure di timeout e ritrasmissione

- Il timeout dovrebbe essere maggiore del RTT (Round Trip Time)
- RTT = tempo richiesto dal momento che il segmento viene spedito (passato a IP) al momento in cui si riceve un riscontro per il segmento
 - > TCP Option - SACK permitted
 - > TCP Option - No-Operation (NOP)
 - > TCP Option - Window scale: 8 (multiply by 256)
 - ▼ [SEQ/ACK analysis]
 - [\[This is an ACK to the segment in frame: 65\]](#)
 - [The RTT to ACK the segment was: 0.026848000 seconds]
 - [iRTT: 0.026935000 seconds]
 - ▼ [Timestamps]
 - [Time since first frame in this TCP stream: 0.026848000 seconds]
 - [Time since previous frame in this TCP stream: 0.026841000 seconds]

iRTT = Initial Round Trip Time

Fausto Marcantoni

Chapter 3 TRANSPORT PROTOCOL

3.126

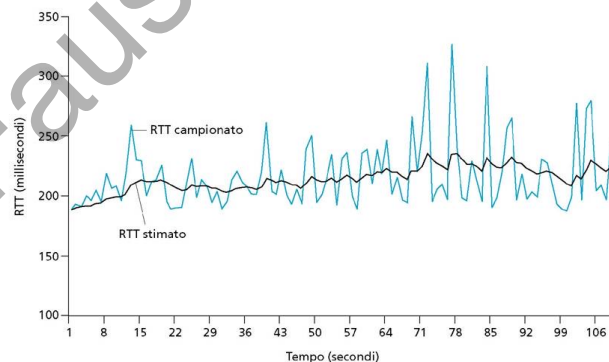
126

Procedure di timeout e ritrasmissione

- Stima di RTT da parte del TCP
 1. Misura RTT una volta ogni tempo di andata e ritorno (**sampleRTT**)
 2. Mantiene una **media pesata** dei valori misurati (**estimatedRTT**)
 3. $\text{estimatedRTT} = (1-\alpha) \text{estimatedRTT} + \alpha \text{SampleRTT}$
(α normalmente uguale a 0,125 - definita in RFC2988)

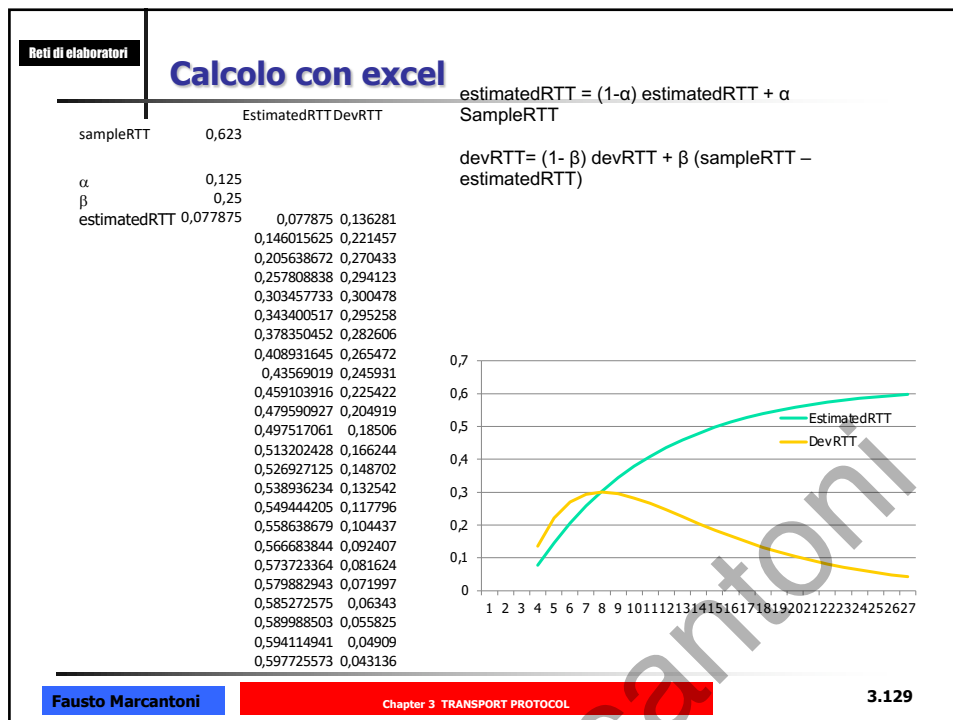
127

RTT campionati e stimati



- Oltre avere una stima di RTT è necessario avere una **misura della Variabilità** di RTT
- $\text{devRTT} = (1 - \beta) \text{devRTT} + \beta (\text{sampleRTT} - \text{estimatedRTT}) \rightarrow \text{variazione di RTT}$
- [definita in RFC2988]
- $\beta \rightarrow 0,25$
- devRTT è una media pesata tra sampleRTT e estimatedRTT

128



129

Reti di elaboratori

Plot RTT histogram using wireshark

Wireshark or tshark can give you the TCP RTT for each received ACK packet using tcp.analysis.ack_rtt which measures the time delta between capturing a TCP packet and the ACK for that packet.

You need to be careful with this as most of your ACK packets will be from your office machines ACKing packets received from the internet, so you will be measuring the RTT between your router seeing the packet from the internet and seeing the ACK from your office machine.

To measure your internet RTT you need to look for ACKS from the internet (ACKing data sent from your network). Assuming your office machines have IP addresses like 192.168.1.x and you have logged all the data on the LAN port of your router you could use a display filter like so:

tcp.analysis.ack_rtt and ip.dst==192.168.1.255/24

To dump the RTTs into a .csv for analysis you could use a tshark command like so;

```
tshark -r router.pcap -R "tcp.analysis.ack_rtt and ip.dst==192.168.1.255/24" -e tcp.analysis.ack_rtt -T fields -E separator=, -E quote=d > rtt.csv
```

The -r option tells tshark to read from your .pcap file
 The -R option specifies the display filter to use
 The -e option specifies the field to output
 The -T options specify the output formatting

<http://stackoverflow.com/questions/6962133/plot-rtt-histogram-using-wireshark-or-other-tool>

Fausto Marcantoni

Chapter 3 TRANSPORT PROTOCOL

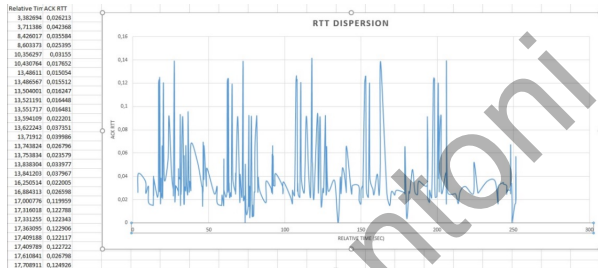
3.130

130

Plot RTT histogram using wireshark by student

1. Aprire Wireshark e selezionare l'interfaccia su cui iniziare a catturare i pacchetti
2. Dopo qualche tempo a scelta (nel mio caso circa 250 secondi) fermare la cattura
3. File->Salva come...->selezionare come tipo di file "Wireshark /tcpdump/ ... -pcap" e salvare (in questo caso l'ho chiamato "rtt.pcap")
4. Aprire un cmd nella cartella di Wireshark e inserire questo comando:
5. **tshark -r rtt.pcap -Y "tcp.analysis.ack_rtt and ip.dst==192.168.1.255/24" -e frame.time_relative -e tcp.analysis.ack_rtt -T fields -E separator=, -E quote=d > rtt.csv**
dove rtt.pcap è il percorso del dump tcp salvato da Wireshark e rtt.csv è il percorso del csv di destinazione
1. Aprire Excel (NON il file csv direttamente) e importare il csv da Dati->Carica dati da testo, impostando come separatore la virgola e cambiando il punto decimale
2. (Wireshark tratta il "." come separatore decimale, di default Excel lo usa come separatore delle migliaia)
3. Selezionare tutti i valori nelle due colonne e disegnare un grafico a scelta (nel mio caso "Dispersione con linee smussate")
4. FINE!

Riccardo Cannella 01/12/2016



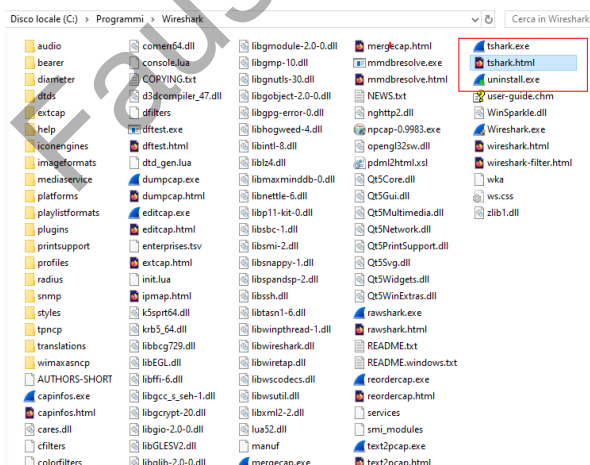
Fausto Marcantoni

Chapter 3 TRANSPORT PROTOCOL

3.131

131

tshark



tshark.exe

Fausto Marcantoni

Chapter 3 TRANSPORT PROTOCOL

3.132

132

tshark

```

C:\Users\fausto.mfausto>"C:\Program Files\Wireshark\tshark.exe" --help
TShark (Wireshark) 3.0.6 (V3.0.6-0-g908c8e357d0f)
Dump and analyze network traffic.
See https://www.wireshark.org for more information.

Usage: tshark [options] ...

Capture interface:
-i <interface>          name or idx of interface (def: first non-loopback)
-f <capture filter>      packet filter in libpcap filter syntax
-s <snaplen>            packet snapshot length (def: appropriate maximum)
-p                      don't capture in promiscuous mode
-I                      capture in monitor mode, if available
-B <buffer size>         size of kernel buffer (def: 2MB)
-y <link type>           link layer type (def: first appropriate)
--time-stamp-type <type> timestamp method for interface
-D                      print list of interfaces and exit
-L                      print list of link-layer types of iface and exit
--list-time-stamp-types print list of timestamp types for iface and exit

Capture stop conditions:
-c <packet count>       stop after n packets (def: infinite)
-a <autostop cond.> ... duration:NUM - stop after NUM seconds
                        filesize:NUM - stop this file after NUM KB
                        files:NUM - stop after NUM files

Capture output:
-b <ringbuffer opt.> ... duration:NUM - switch to next file after NUM secs
                        interval:NUM - create time intervals of NUM secs
                        filesize:NUM - switch to next file after NUM KB
                        files:NUM - ringbuffer: replace after NUM files

```

tshark.html

```

NAME
tshark - Dump and analyze network traffic

SYNOPSIS
tshark [-a] [-a <capture autostop condition>] ... [-b <capture ring buffer option>] ... [-B <capture buffer size>]
[-c <capture packet count>] [-C <configuration profile>] [-d <layer type>] ... [-e <decoder-as-protocol>] [-D]
[-e <fields>] [-E <field print options>] [-f <capture filter>] [-F <file format>] [-g] [-h] [-H <input hosts file>]
[-i <capture interface>] [-I] [-j <protocol match filter>] [-J <protocol match filter>] [-K <keytab>] [-L] [-l] [-n]
[-N <name resolving flags>] [-o <preference settings>] ... [-O <protocol>] [-p] [-P] [-q] [-Q] [-r <infile>]
[-R <Read filter>] [-s <capture snaplen>] [-S <separator>] [-t <ajadajodjdjddlelrludludj>]
[-T <fields(json|jsonraw|pdml|psj|psml|table|text)>] [-u <seconds type>] [-U <tap_name>] [-v] [-V] [-w <outfile>] [-]
[-W <file format option>] [-x] [-X <extension option>] [-y <capture link type>] [-Y <display filter>]
[-M <auto session reset>] [-z <statistics>] [-capture-comment <comment>] [-list-time-stamp-types]
[-time-stamp-type <type>] [-color] [-no-duplicate-keys] [-export-objects <protocol>,<destid>]
[-enable-protocol <proto_name>] [-disable-protocol <proto_name>] [-enable-heuristic <short_name>]
[-disable-heuristic <short_name>] [-filter]

tshark -G [-report type] [-elastic-mapping-filter <protocol>]

DESCRIPTION
TShark is a network protocol analyzer. It lets you capture packet data from a live network, or read packets from a previously
saved capture file, either printing a decoded form of those packets to the standard output or writing the packets to a file.
TShark's native capture file format is pcapng format, which is also the format used by Wireshark and various other tools.

Without any options set, TShark will work much like tcpdump. It will use the pcap library to capture traffic from the first
available network interface and displays a summary line on the standard output for each received packet.

```

Reti di elaboratori

tshark

Prompt dei comandi

```

C:\Users\fausto.mfausto>"C:\Program Files\Wireshark\tshark.exe" -D
1. \Device\NPF_{62F9A358-3ACE-4162-B113-E5460AB8EF6E} (Connessione alla rete locale (LAN)* 14)
2. \Device\NPF_{5C818218-2D68-4BF2-8CAC-CBFF1802F53D} (VMware Network Adapter VMnet8)
3. \Device\NPF_{A6894AAD-BFF7-467A-A49F-AE879C25261D} (VMware Network Adapter VMnet1)
4. \Device\NPF_{0FB6FE01-E122-46D1-B574-B016E0AF66E4} (Connessione alla rete locale (LAN)* 4)
5. \Device\NPF_{8E12E87A-AEB3-412F-982A-494C700BE8C5} (Connessione alla rete locale (LAN)* 6)
6. \Device\NPF_{31F20730-8C4F-4C02-B264-D3E5F3F123B7} (Ethernet)
7. \Device\NPF_{0260747F-8B6A-4CAB-B28D-B18D23188CA9} (Connessione alla rete locale (LAN)* 16)
8. \Device\NPF_{3812677C-83D8-4FB1-A92E-F282E453CBA9} (Ethernet 2)
9. \Device\NPF_{EDE4DCAD-038A-4E20-A3B4-8D11412C0B67} (Ethernet 4)
10. \Device\NPF_{3211EE50-C2B0-4606-88A7-A501CCF9F562} (Wi-Fi)
11. \Device\NPF_{4C7584C0-3136-40A8-AD09-BF4895451869} (Connessione alla rete locale (LAN)* 13)
12. \Device\NPF_{Loopback} (Adapter for loopback traffic capture)

C:\Users\fausto.mfausto>

```

C:\Program Files\Wireshark\tshark.exe -D
Per visualizzare le interfacce di rete

Fausto Marcantoni
Chapter 3 TRANSPORT PROTOCOL
3.135

135

Reti di elaboratori

tshark

Prompt dei comandi

```

C:\Users\fausto.mfausto>"C:\Program Files\Wireshark\tshark.exe" -i 6
Capturing on 'Ethernet'
1 0.000000 mfausto.amministrazione.unicam → ec2-35-165-44-141.us-west-2.
assembled PDU]
2 0.000008 mfausto.amministrazione.unicam → ec2-35-165-44-141.us-west-2.
n-discover(2799) → https(443) [ACK] Seq=1 Ack=1 Win=63685 Len=1
3 0.092492 mfausto.amministrazione.unicam → 192.168.1.2 BJNP 58 Scanner
4 0.092502 mfausto.amministrazione.unicam → 192.168.1.2 BJNP 58 Scanner
5 0.133777 SeltaTel_01:4b:55 → Broadcast ARP 60 who has 192.168.176.10
6 0.142705 SeltaTel_01:c0:4d → Broadcast ARP 60 who has 192.168.176.10
7 0.186684 ec2-35-165-44-141.us-west-2.compute.amazonaws.com → mfausto.am
scover(2799) [ACK] Seq=1 Ack=2 Win=29095 Len=0
8 0.209715 SeltaTel_03:19:1a → Broadcast ARP 60 who has 192.168.176.10
9 0.272809 SeltaTel_02:9f:e5 → Broadcast ARP 60 who has 192.168.176.10
10 0.292559 mfausto.amministrazione.unicam → 192.168.1.3 BJNP 58 Scanner
11 0.292568 mfausto.amministrazione.unicam → 192.168.1.3 BJNP 58 Scanner
12 0.334208 Cisco_f6:85:00 → Broadcast ARP 60 who has 192.168.176.10?
13 0.359898 SeltaTel_03:18:41 → Broadcast ARP 60 who has 192.168.176.10
14 0.388879 Cisco_f6:85:00 → Broadcast ARP 60 who has 193.205.92.33? T
15 0.493397 mfausto.amministrazione.unicam → 192.168.1.4 BJNP 58 Scanner
16 0.493405 mfausto.amministrazione.unicam → 192.168.1.4 BJNP 58 Scanner
17 0.532632 SeltaTel_00:ac:6f → Broadcast ARP 60 who has 192.168.176.10
18 0.578338 SeltaTel_01:5f:02 → Broadcast ARP 60 who has 192.168.176.10
19 0.580568 SeltaTel_03:33:37 → Broadcast ARP 60 who has 192.168.176.10

```

C:\Program Files\Wireshark\tshark.exe -i 6
Per catturare i pacchetti dall interfaccia di rete **i6**

Fausto Marcantoni
Chapter 3 TRANSPORT PROTOCOL
3.136

136

Reti di elaboratori

Grafico RTT

Statistiche → Grafici dei Flussi TCP → Round Trip Time

Fausto Marcantoni

Chapter 3 TRANSPORT PROTOCOL

3.137

137

Reti di elaboratori

Retransmission Time Out

Il timeout (**RTO Retransmission Time Out**) indica il tempo entro il quale la sorgente si aspetta di ricevere il riscontro (ack)
 nel caso in cui il riscontro non arrivi, la sorgente procede alla ritrasmissione

RTO **non** può essere un **valore statico predefinito**
 il tempo di percorrenza sperimentato dai segmenti è variabile e dipende

- **dalla distanza tra sorgente e destinazione**
- **dalle condizioni della rete**
- **dalla disponibilità della destinazione**

esempio: fattorino che deve consegnare un pacco in città

RTO deve dunque essere **calcolato dinamicamente di volta in volta**

- durante la fase di instaurazione della connessione
- durante la trasmissione dei dati

Il calcolo di RTO si basa sulla misura del RTT (Round Trip Time)

Fausto Marcantoni

Chapter 3 TRANSPORT PROTOCOL

3.138

138

Retransmission Time Out

RTO ha un valore minimo (normalmente intorno a 200ms) e uno massimo (normalmente 60s e oltre)

Implementazioni moderne usano l'opzione "timestamp" per il calcolo di RTT, ma i principi base di adattamento alle condizioni della rete rimane lo stesso:

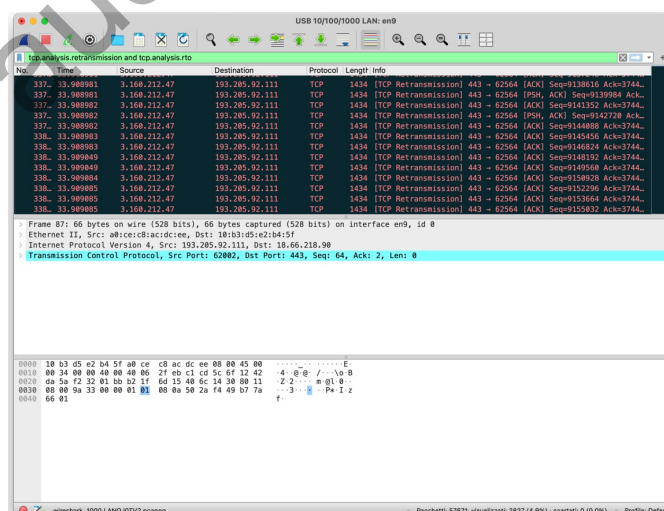
- viene stimato un valore medio e una deviazione media di RTT
- RTO viene calcolato in base a questi valori medi

```
> TCP Option - SACK permitted
> TCP Option - No-Operation (NOP)
> TCP Option - Window scale: 8 (multiply by 256)
▼ [SEQ/ACK analysis]
  [This is an ACK to the segment in frame: 65]
  [The RTT to ACK the segment was: 0.026848000 seconds]
  [RTT: 0.026935000 seconds]
```

```
▼ [Timestamps]
  [Time since first frame in this TCP stream: 0.026848000 seconds]
  [Time since previous frame in this TCP stream: 0.026841000 seconds]
```

RTO – Filtro Wireshark

tcp.analysis.retransmission and tcp.analysis.rto



Reti di elaboratori

creare nuove colonne in wireshark

Fausto Marcantoni

Chapter 3 TRANSPORT PROTOCOL

3.141

141

Reti di elaboratori

Impostazione TIME OUT

- Timeout maggiore o uguale a estimatedRTT
 - Altrimenti molte ritrasmissioni non necessarie
- Non molto maggiore
 - Quando un segmento viene perso non verrebbe tempestivamente ritrasmesso
- $TIMEOUT = estimatedRTT + \text{margine}$
 - Margine grande quando ci sono ampie fluttuazioni
 - Margine piccolo quando ci sono piccole fluttuazioni
- **$TIMEOUT = estimatedRTT + 4 * devRTT$**

Fausto Marcantoni

Chapter 3 TRANSPORT PROTOCOL

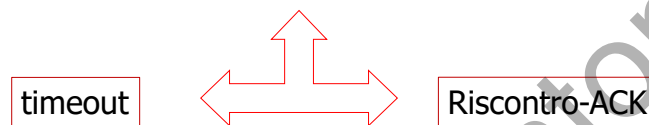
3.142

142

Trasferimento affidabile dei dati (lato mittente)

TCP riceve i dati dall'applicazione,

- li incapsula in un segmento, e passa il segmento a IP.
- se il timer non è già in funzione per qualche altro segmento, il TCP avvia il timer quando il segmento viene passato a IP. (È utile pensare che il timer sia associato con il più vecchio segmento non riscontrato.)
- L'intervallo di scadenza per questo timer è il TimeoutInterval, che è calcolato da EstimatedRTT e DevRTT.



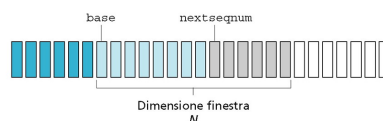
Trasferimento affidabile dei dati (lato mittente)

evento timeout

- Il TCP risponde ritrasmettendo il segmento che ha causato il timeout stesso
- Il TCP quindi riavvia il timer.

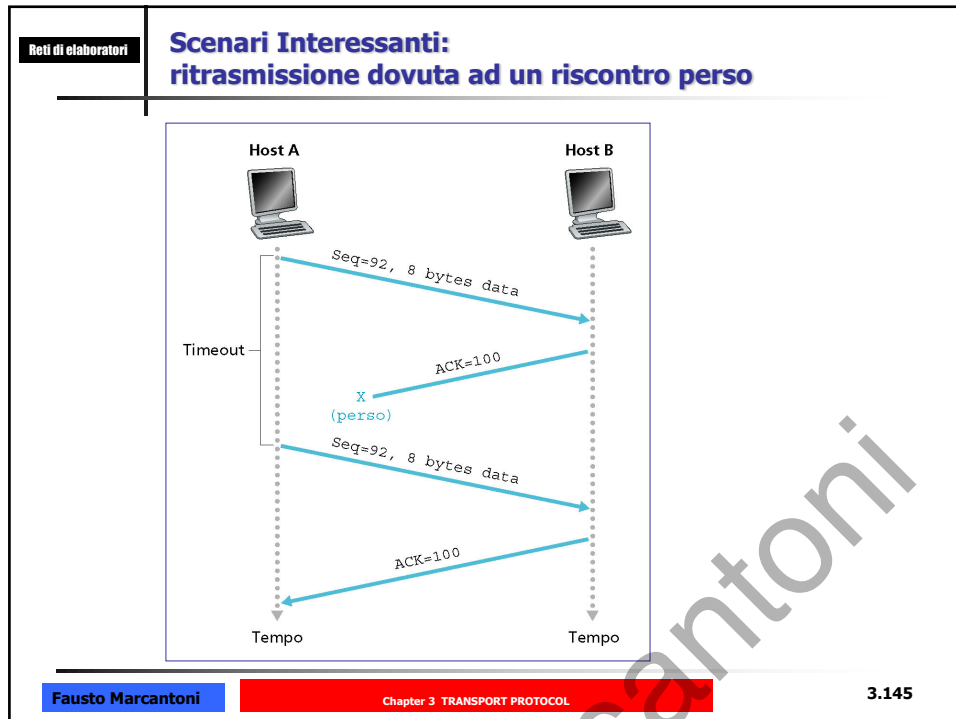
arrivo di un segmento di riscontro (ACK) dal ricevente

- il TCP confronta il valore y di ACK con la sua variabile SendBase. La variabile di stato SendBase di TCP è il numero di sequenza del più vecchio byte non riscontrato.
- Se $y > \text{SendBase}$, allora ACK sta riscontrando uno o più segmenti non riscontrati prima.
- Quindi il mittente aggiorna la sua variabile SendBase;
- inoltre riavvia il timer se ci sono segmenti attualmente non ancora riscontrati.

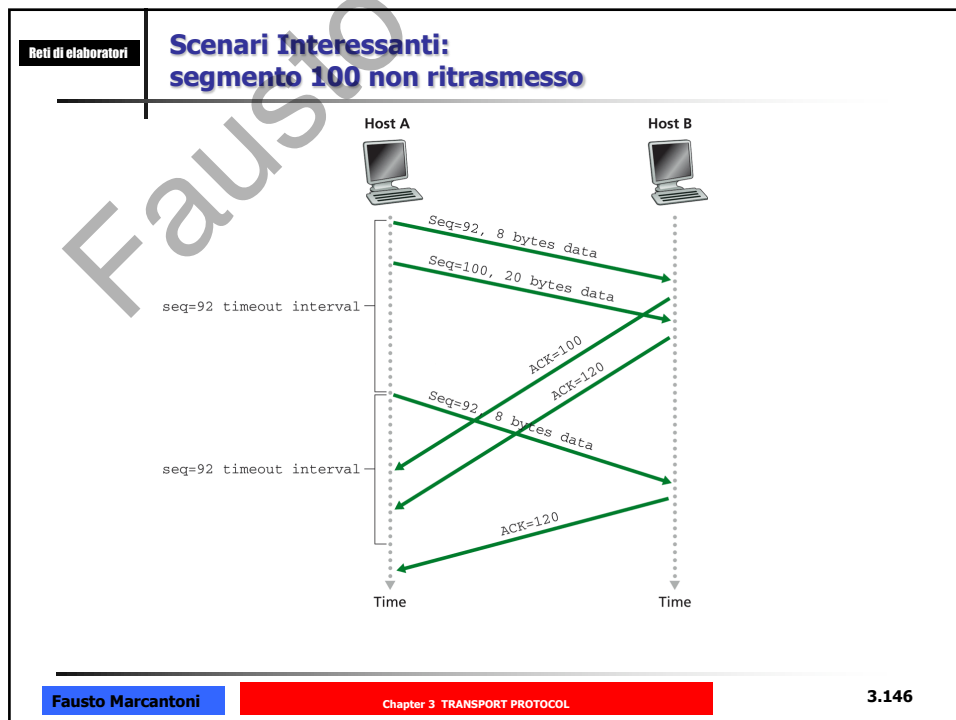


Legenda:

 Già riscontrato	 Disponibile, non ancora inviato
 Inviato, non ancora riscontrato	 Non disponibile

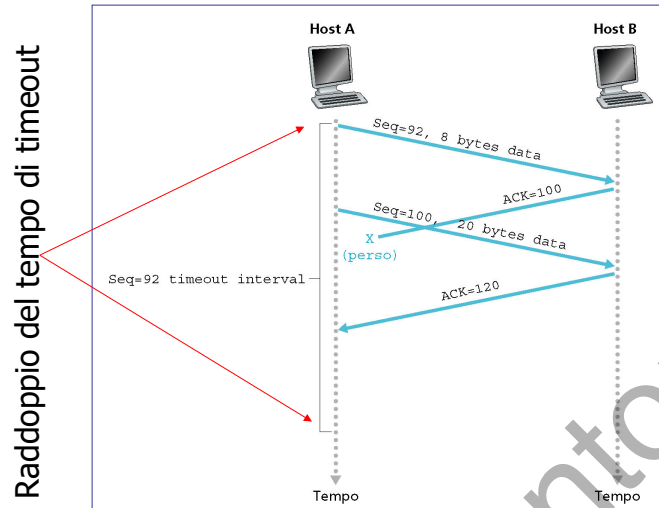


145



146

Scenario: riscontri cumulativi

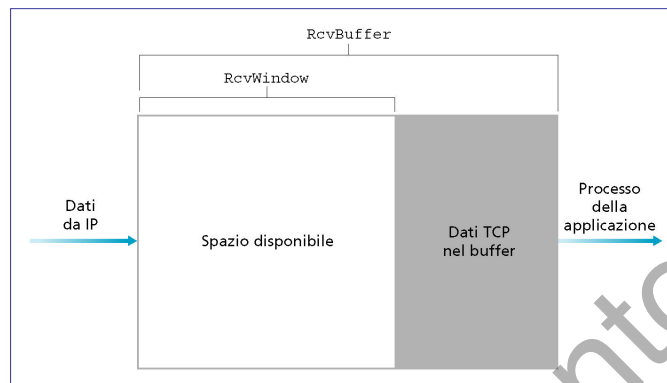


Alcune modifiche adottate

- Raddoppio del tempo di timeout
 - Quando un segmento fa scattare il timeout viene ritrasmesso
 - Il tempo viene **raddoppiato** anziché **calcolato**
 - La scadenza del timer è probabilmente dovuta a congestione
 - Si evita di ritrasmettere il segmento → forma limitativa di controllo di congestione
- Ritrasmissione veloce
 - Periodo di timeout a volte lungo
 - Il mittente può svelare la perdita di un pacchetto da riscontri duplicati (**tre ACK**)
 - Il mittente prende coscienza che il segmento successivo a quello riscontrato è andato perso
 - Esegue una ritrasmissione veloce

Controllo di flusso

- Servizio per evitare che il sender saturi il buffer di ricezione
- Mantenimento nel sender di una variabile detta **finestra di ricezione**
- La finestra di ricezione è dinamica



Meccanismo del controllo del flusso

1. Il receiver comunica al sender l'ampiezza della finestra di ricezione
2. La finestra di ricezione (RcvWindow) è uguale all'ampiezza del buffer meno la differenza tra l'ultimo byte arrivato e l'ultimo byte passato all'applicazione

$$\text{LastByteRcvd} - \text{LastByteRead} \leq \text{RcvBuffer}$$



Reti di elaboratori

Meccanismo del controllo del flusso

- All'inizio la finestra di ricezione è uguale all'ampiezza del buffer

$$RcvWindow = RcvBuffer - [LastByteRcvd - LastByteRead]$$
- Il mittente dovrà assicurarsi sempre di trasmettere i dati solo quando la differenza tra l'ultimo byte trasmesso e l'ultimo byte riscontrato è minore o uguale alla finestra di ricezione
- Problema quando la finestra è 0 ed il ricevente non deve mandare a dire più niente
 - Specifiche TCP impongono **la trasmissione con 1 byte di dati**

Fausto Marcantoni

Chapter 3 TRANSPORT PROTOCOL

3.151

151

Reti di elaboratori

Controllo di flusso [simulazione]

Numero di sequenza	Numero di riscontro	Dimensione finestra
1001	4000	
5001	0	
5001	1000	
5001	1000	

Fausto Marcantoni

Chapter 3 TRANSPORT PROTOCOL

3.152

152

Reti di elaboratori

Dimensione della finestra - lato sender

```

> Frame 65: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface \Device\NPF_{50C0A4FC-92A8-4BC2-A300-06F46D38F065}, id 0
> Ethernet II, Src: AzureIav_43:53:23 (24:0a:64:43:53:23), Dst: Vodafone_2d:eb:20 (14:14:59:2d:eb:20)
> Internet Protocol Version 4, Src: 192.168.1.13, Dst: 34.107.221.82
> Transmission Control Protocol, Src Port: 2476, Dst Port: 80, Seq: 0, Len: 0
  Source Port: 2476
  Destination Port: 80
  [Stream index: 0]
  [TCP Segment Len: 0]
  Sequence Number: 0 (relative sequence number)
  Sequence Number (raw): 3642849256
  [Next Sequence Number: 1 (relative sequence number)]
  Acknowledgment Number: 0
  Acknowledgment number (raw): 0
  1000 .... = Header Length: 32 bytes (8)
  Flags: 0x002 (SYN)
  Window: 64240
  [Calculated window size: 64240]
  Checksum: 0x57a6 [unverified]
  [Checksum Status: Unverified]
  Urgent Pointer: 0
  > Options: (12 bytes), Maximum segment size, No-Operation (NOP), Window scale, No-Operation (NOP), No-Operation (NOP), SACK permitted
  [Timestamps]
    [Time since first frame in this TCP stream: 0.00000000 seconds]
    [Time since previous frame in this TCP stream: 0.00000000 seconds]

```

Dimensione della finestra - lato sender

Fausto Marcantoni

Chapter 3 TRANSPORT PROTOCOL

3.153

153

Reti di elaboratori

Dimensione della finestra - lato receiver

```

> Frame 77: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface \Device\NPF_{50C0A4FC-92A8-4BC2-A300-06F46D38F065}, id 0
> Ethernet II, Src: Vodafone_2d:eb:20 (14:14:59:2d:eb:20), Dst: AzureIav_43:53:23 (24:0a:64:43:53:23)
> Internet Protocol Version 4, Src: 34.107.221.82, Dst: 192.168.1.13
> Transmission Control Protocol, Src Port: 80, Dst Port: 2476, Seq: 0, Ack: 1, Len: 0
  Source Port: 80
  Destination Port: 2476
  [Stream index: 0]
  [TCP Segment Len: 0]
  Sequence Number: 0 (relative sequence number)
  Sequence Number (raw): 3615737765
  [Next Sequence Number: 1 (relative sequence number)]
  Acknowledgment Number: 1 (relative ack number)
  Acknowledgment number (raw): 3642849257
  1000 .... = Header Length: 32 bytes (8)
  Flags: 0x012 (SYN, ACK)
  Window: 65535
  [Calculated window size: 65535]
  Checksum: 0xb37a [unverified]
  [Checksum Status: Unverified]
  Urgent Pointer: 0
  > Options: (12 bytes), Maximum segment size, No-Operation (NOP), No-Operation (NOP), SACK permitted, No-Operation (NOP), Window scale
  [SEQ/ACK analysis]
    [This is an ACK to the segment in frame: 65]
    [The RTT to ACK the segment was: 0.026848000 seconds]
    [RTT: 0.026935000 seconds]
  [Timestamps]
    [Time since first frame in this TCP stream: 0.026848000 seconds]
    [Time since previous frame in this TCP stream: 0.026841000 seconds]

```

Dimensione della finestra - lato receiver

Fausto Marcantoni

Chapter 3 TRANSPORT PROTOCOL

3.154

154

windows size scaling factor

$$513 \times 256 = 131328$$

Dimensione della finestra - lato sender - ricalcolata

tcp.window_size

Inserire colonna con il valore window_size

tcp.window_size == 0

Sequence Number (raw): 4112861313
 (Next Sequence Number: 2848 - (relative sequence number))
 Acknowledgment Number: 0
 Acknowledgment number (raw): 0
 Window length: 20 bytes (5)
 Flags: RST
 Window: 0
 (calculated window size: 0)
 Window size scaling factor: 64
 Checksum: 8x56d2 [unverified]
 (Checksum Status: Unverified)
 Urgent Pointer: 0
 (Timestamps)

Questo filtro mostra i pacchetti in cui il destinatario non è in grado di ricevere più dati (la finestra di ricezione è a zero). Questa condizione si verifica quando il ricevitore segnala che il suo buffer è pieno.

157

Filtri Wireshark

Finestra di ricezione TCP maggiore di un valore specifico:

Per vedere i pacchetti con una finestra di ricezione più grande di un determinato valore:

```
tcp.window_size_value > 10000
```

Cambiamenti nella finestra di ricezione:

Se vuoi vedere pacchetti in cui ci sono cambiamenti nella finestra di ricezione (ad esempio, la finestra si sta riducendo o aumentando):

```
tcp.analysis.window_update
```

Finestra ridotta (Shrinking Window):

Filtra pacchetti dove la finestra di ricezione si è ridotta rispetto al valore precedente, il che potrebbe indicare congestione o un'ACK che tarda ad arrivare.

```
tcp.analysis.window_full
```

Analisi della finestra a zero

Se vuoi analizzare i pacchetti in cui il ricevitore ha dichiarato di non poter ricevere più dati (fenomeno chiamato "finestra zero"), usa:

```
tcp.window_size_value == 0
```

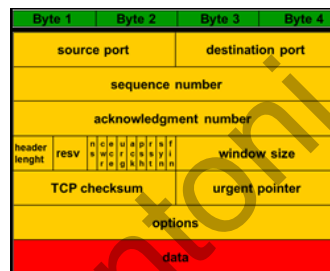
158

Extensions for high speed

Extensions for high speed (window scale opt.)

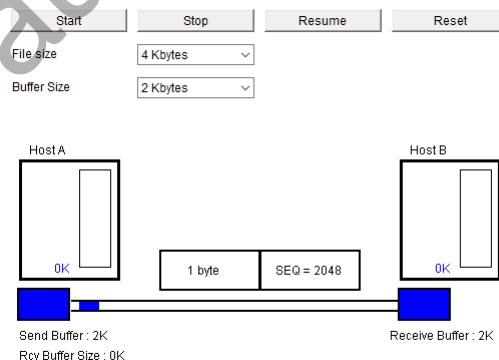
Il window scaling venne introdotto nell' RFC 1072 e poi ridefinito nel RFC 1323.

La logica che pilota il window scaling consiste nell' ampliare il window size da 16 a 32 bits.



<https://tools.ietf.org/html/rfc1323>

Flowcontrol simulation

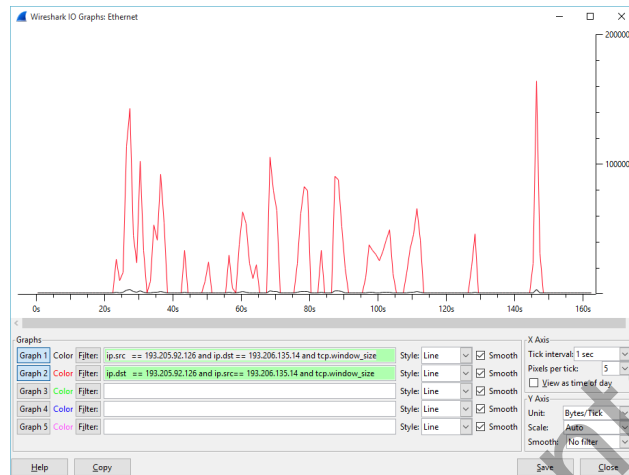


<http://computerscience.unicam.it/marcantoni/reti/applet/FlowControl/flow.html>

<https://www2.tkn.tu-berlin.de/teaching/rn/animations/flow/>

Wireshark – windows size graph

ip.dst == 193.205.92.126 and ip.src == 193.206.135.13 and tcp.window_size



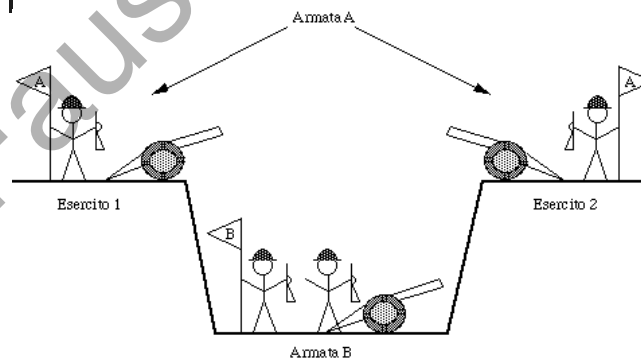
Fausto Marcantoni

Chapter 3 TRANSPORT PROTOCOL

3.161

161

Il problema dei due eserciti



Problema dei due eserciti:

Vi sono due armate di uno stesso esercito e nel mezzo vi è un'armata nemica. Le due armate insieme riuscirebbero a sconfiggere l'esercito nemico ma singolarmente perderebbero; dovrebbero quindi mettersi d'accordo sul momento dell'attacco. Per comunicare con l'altra armata, il messaggero deve per forza passare dall'esercito nemico, e viene così catturato. In pratica un'armata non è mai sicura che il messaggio di richiesta arrivi. Ma cosa succede se è il messaggero di conferma che viene catturato? Nasce quindi lo stesso problema.

Fausto Marcantoni

Chapter 3 TRANSPORT PROTOCOL

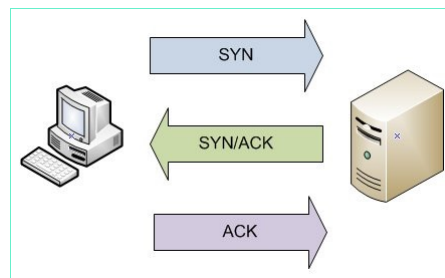
3.162

162

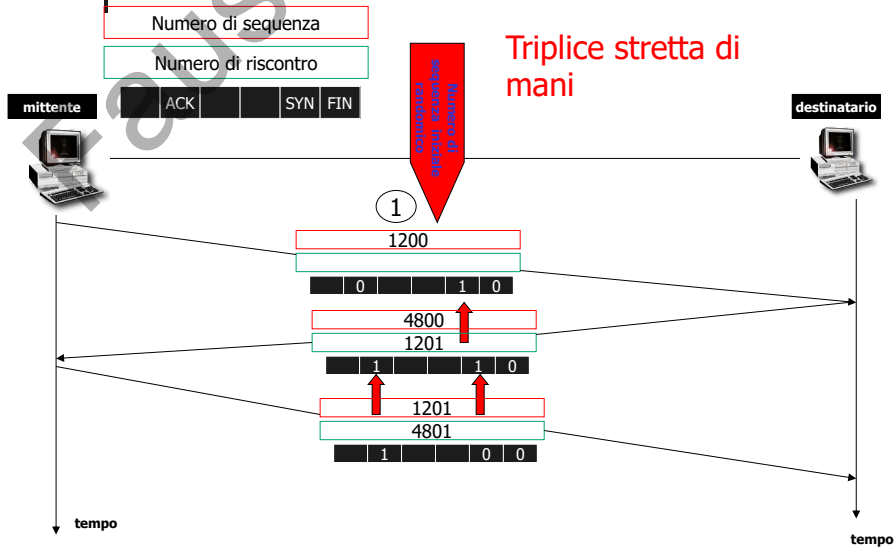
three way handshake

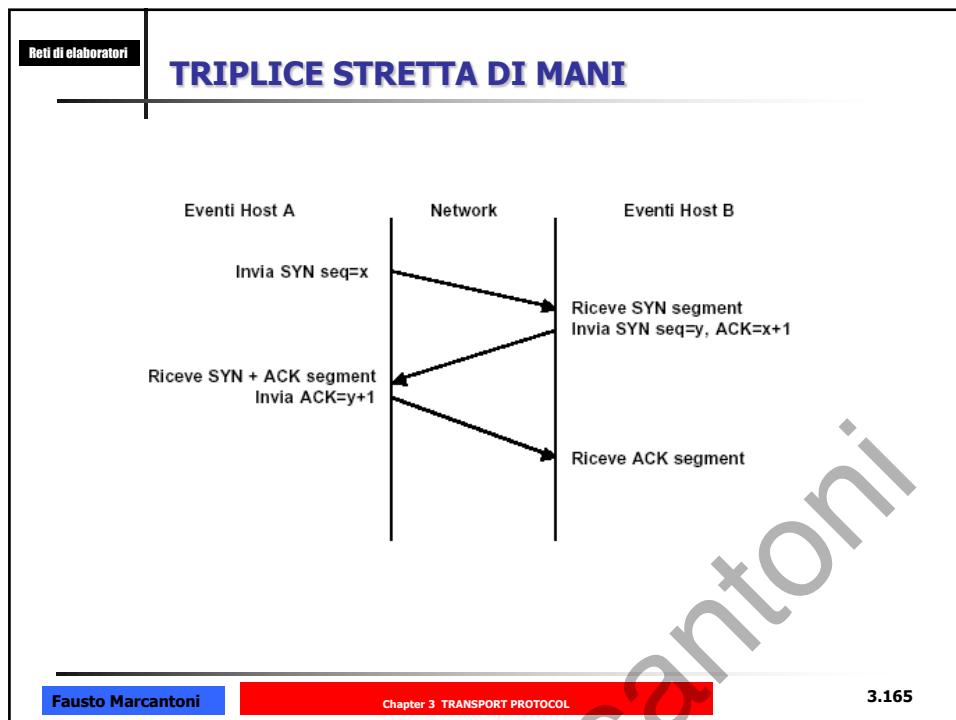
La creazione della connessione:

il three way handshake

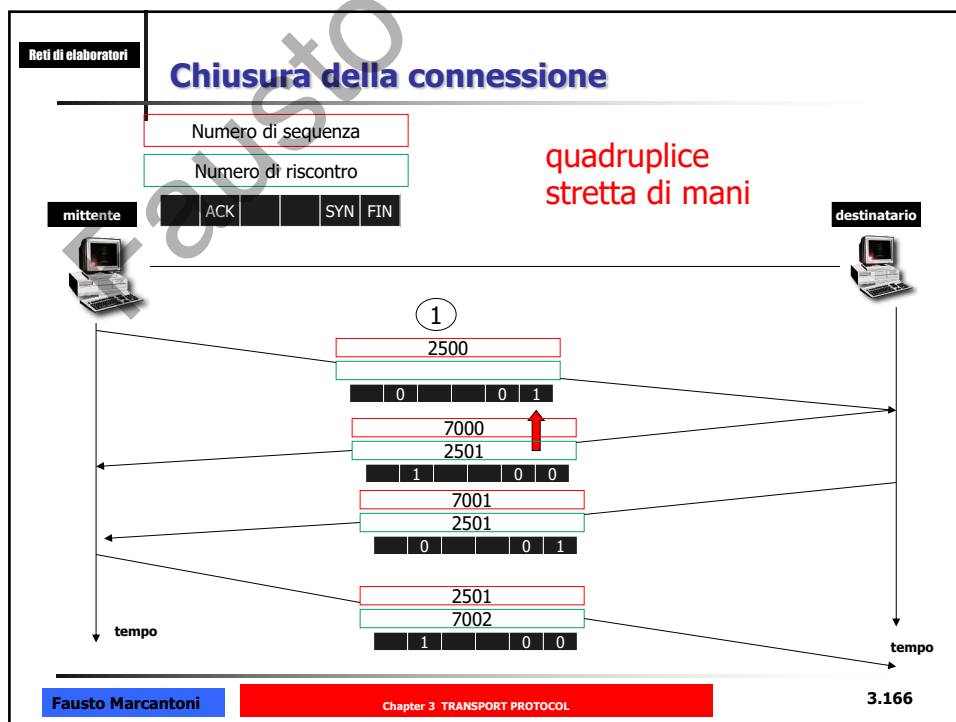


Apertura della connessione (funzionamento del protocollo)



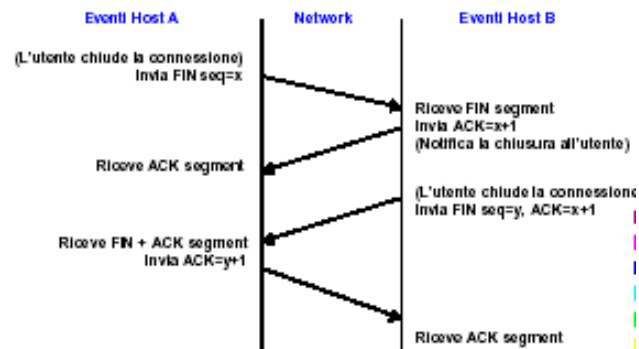


165



166

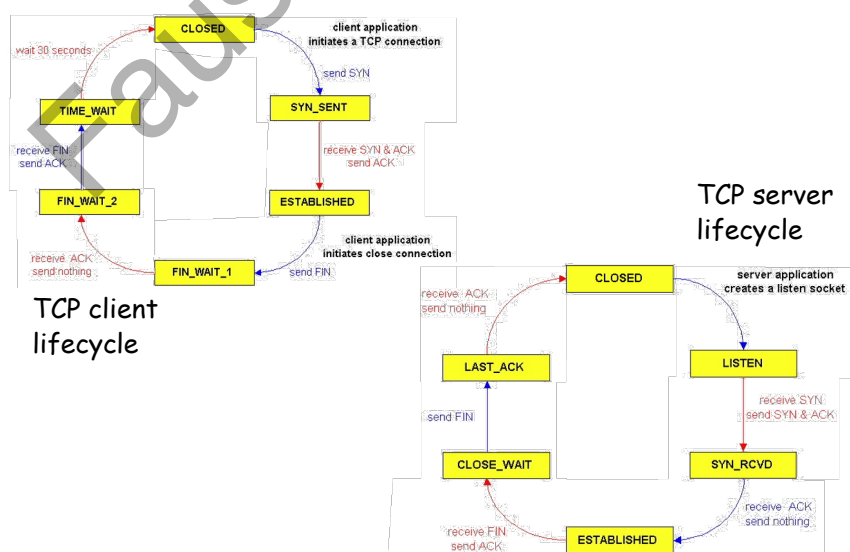
Quadruplice stretta di mani

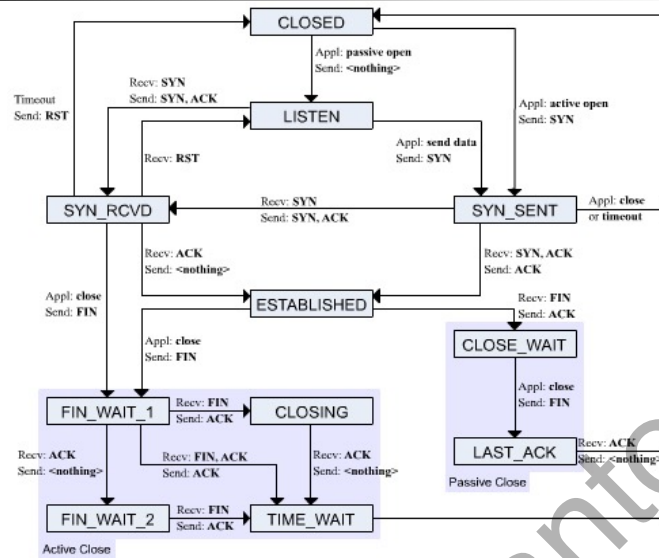


3-way handshake modificato:

l'host B può ancora voler trasmettere dei dati, anche se l'host A ha chiuso la connessione; quindi la chiusura della connessione (lato host B) può avvenire in un secondo tempo

TCP Connection Management





169

Principi sul controllo di congestione

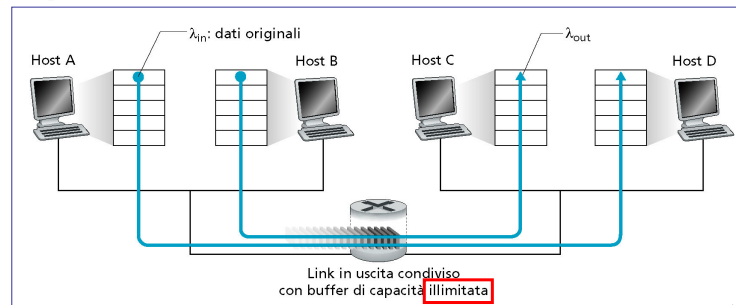
Congestione:

- ❖ troppe sorgenti trasmettono troppi dati, a una velocità talmente elevata che la rete non è in grado di gestirli
- ❖ differisce dal controllo di flusso!
- ❖ manifestazioni:
 - ✓ pacchetti smarriti (overflow nei buffer dei router)
 - ✓ lunghi ritardi (accodamento nei buffer dei router)
- ❖ tra i dieci problemi più importanti del networking!

170

Principi del controllo di congestione

1° Scenario: due connessioni si dividono un singolo hop con **buffer infinito**



λ = frequenza media byte/s

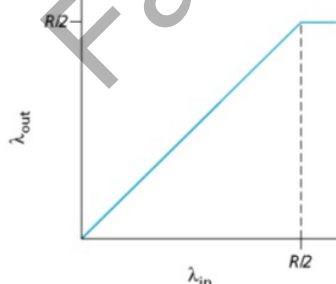
λ_{in} → velocità media con cui l'applicazione sta passando i dati alla connessione

λ_{out} → duale

R → capacità del link

Principi del controllo di congestione

1° scenario : **throughput della connessione**

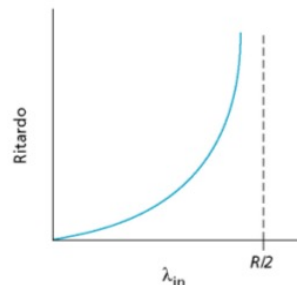


La banda di trasmissione viene condivisa tra A e B.

$R/2$ è il limite di banda tra i due Host

Principi del controllo di congestione

1° scenario : ritardo

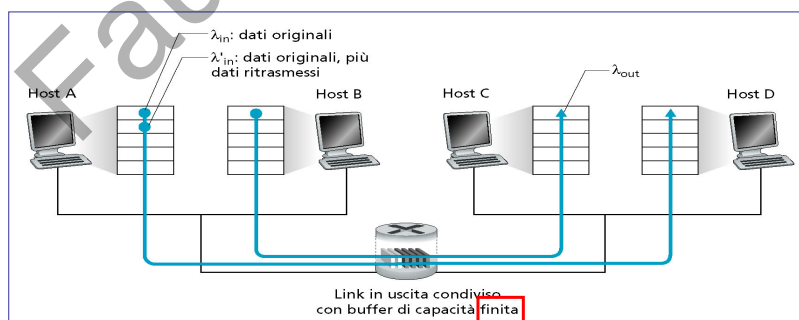


Avvicinandosi a $R/2$ il ritardo medio aumenta sempre di più e il numero di pacchetti in coda diventa illimitato

- Ci si devono aspettare grandi ritardi di coda quando la velocità di arrivo dei pacchetti è prossima alla capacità del link

Principi del controllo di congestione

2° Scenario: due host con ritrasmissioni e un router con buffer finito



λ_{in} → velocità media con cui l'applicazione sta passando i dati alla connessione

λ'_{in} → velocità media con cui l'applicazione sta passando i dati alla connessione + **ritrasmissioni**

Principi del controllo di congestione

2° scenario : prestazioni

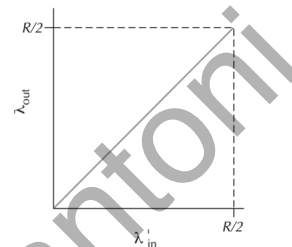
Le prestazioni che si riscontrano in questo scenario dipenderanno ora fortemente da come si effettua la ritrasmissione.

Innanzitutto, consideriamo il caso **poco probabile** in cui **l'Host A sia in grado di determinare se il buffer nel router abbia spazio a disposizione** e trasmetta pertanto un pacchetto solo quando il buffer è libero.

In questo caso non si verificherebbe smarrimento

$$\lambda_{in} = \lambda'_{in}$$

e il throughput della connessione sarebbe

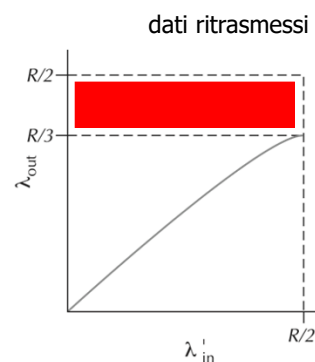


Principi del controllo di congestione

2° scenario : prestazioni

Consideriamo ora il caso, un po' più realistico, in cui il **mittente ritrasmette solo quando è certo che un pacchetto sia andato perduto**.

L'host mittente potrebbe impostare il proprio **timeout con un valore sufficientemente grande** da essere praticamente certo che il pacchetto di cui non si è ancora ricevuto acknowledgement sia stato perduto.

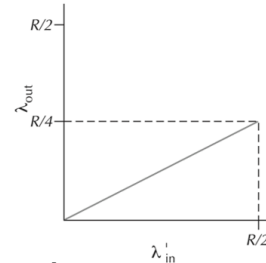


Principi del controllo di congestione

2° scenario : prestazioni

Il mittente possa andare **in timeout prematuramente** e ritrasmettere un pacchetto che abbia subito ritardi in coda, **ma non sia stato perduto**.

In questo caso, sia il pacchetto originale sia quello ritrasmesso possono raggiungere il destinatario.



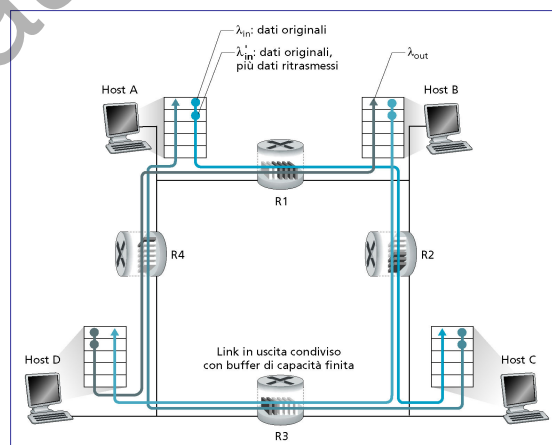
Al destinatario è sufficiente una copia di tale pacchetto e **scarnerà le altre**.

In questo caso, il lavoro effettuato dal router per instradare la copia ritrasmessa del pacchetto è sprecato (**duplica le spedizioni**), dato che il destinatario avrà già ricevuto la copia originale.

ritrasmissioni non necessarie da parte del mittente come risposta a lunghi ritardi possono costringere un router a utilizzare la larghezza di banda del collegamento per instradare copie non necessarie di un pacchetto

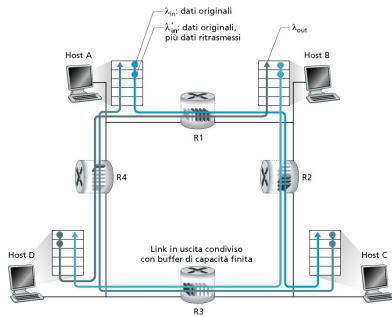
Principi del controllo di congestione

3° scenario : quattro sender e quattro router con buffer finito e percorsi a salti multipli



Principi del controllo di congestione

3° scenario: prestazioni



Consideriamo la connessione A_C

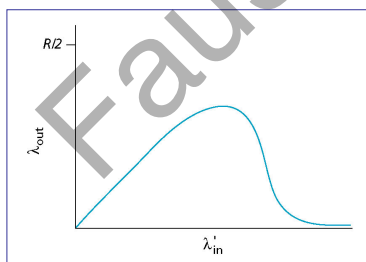
Divide il router R1 con D-B

Divide il router R2 con B-D

- Per valori piccoli di λ_{in} il sovraccarico del buffer è raro ed il throughput è circa uguale al carico offerto
- Per valori di λ_{in} considerevoli :
 - Un grande carico sulla tratta B-D fa diminuire la quantità di traffico di A-C che attraversa R2

Principi del controllo di congestione

3° scenario: prestazioni



Consideriamo la connessione A_C

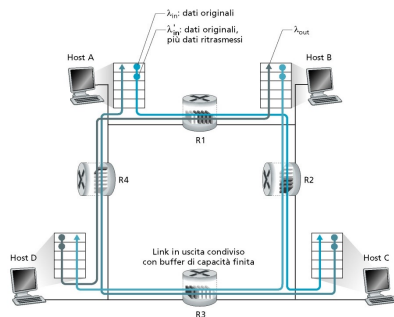
Divide il router R1 con D-B

Divide il router R2 con B-D

- Per valori piccoli di λ_{in} il sovraccarico del buffer è raro ed il throughput è circa uguale al carico offerto
- Per valori di λ_{in} considerevoli :
 - Quando questo traffico si avvicina all'infinito può far sì che il traffico A-C su R2 vada a zero

Principi del controllo di congestione

3° scenario: prestazioni



Consideriamo la connessione A_C

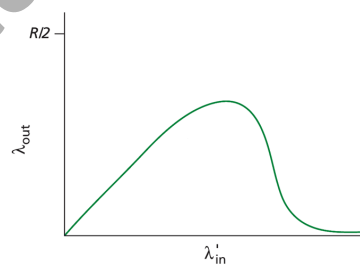
Divide il router R1 con D-B

Divide il router R2 con B-D

- ogni qualvolta un pacchetto viene scartato sul router del secondo hop, il lavoro effettuato dal router del primo hop nell'instradamento del pacchetto verso il secondo router finisce per essere **"sprecato"**.
- La capacità trasmissiva utilizzata dal primo router per instradare il pacchetto al secondo potrebbe essere utilizzata in modo più proficuo trasmettendo un altro pacchetto

Principi del controllo di congestione

3° scenario : costi della congestione



Quando un pacchetto è perso lungo il percorso, la capacità di trasmissione che è stata usata in ciascuno dei router a monte per inoltrare quel pacchetto al punto in cui si è perso è andata sprecata

Reti di elaboratori

Approcci al controllo di congestione

- Controllo della congestione assistito dalla rete (Available Bit Rate di ATM)
 - I componenti a livello di rete (ossia i router) forniscono un feedback esplicito al mittente sullo stato di congestione della rete. Questo avviso può essere semplicemente un bit che indica traffico su un collegamento.

Fausto Marcantoni

Chapter 3 TRANSPORT PROTOCOL

3.183

183

Reti di elaboratori

Approcci al controllo di congestione

- Controllo della congestione end-end
 - Lo strato di rete non fornisce alcun supporto esplicito (TCP)
 - I timeout e la perdita di pacchetti indicano una congestione della rete
 - TCP ridimensiona la propria finestra

Fausto Marcantoni

Chapter 3 TRANSPORT PROTOCOL

3.184

184

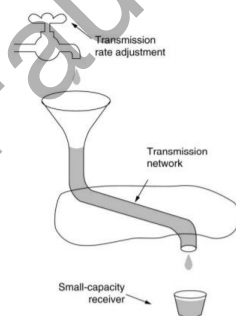
Controllo di congestione del TCP

tre domande

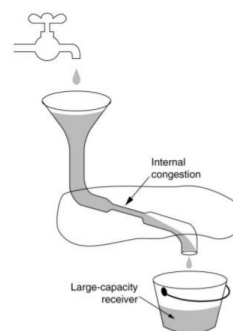
1. come può il **mittente** TCP **limitare la velocità di invio** del traffico sulla propria connessione?
2. come percepisce il **mittente** la **congestione sul percorso** che porta alla destinazione?
3. quale algoritmo dovrebbe essere usato dal **mittente** per **variare la velocità di invio** in funzione della congestione end-to-end?

185

Congestione TCP



una rete **veloce** che alimenta
un ricevitore a **bassa capacità**



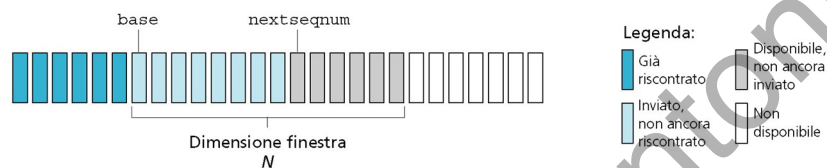
una rete **lenta** che alimenta
un ricevitore ad **alta capacità**

186

Controllo di congestione del TCP

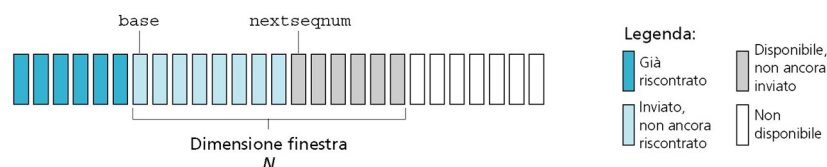
In che modo un sender in TCP limita il ritmo di trasmissione?

- Tenendo traccia di un'altra variabile → **finestra di congestione**
- La quantità di dati non riscontrati che un host può avere all'interno di una connessione non deve superare
- **il minimo tra la finestra di congestione e la finestra di ricezione**
 - $\text{LastByteSent} - \text{LastByteAcked} \leq \min(\text{CongWin}, \text{RcvWindow})$



Controllo di congestione del TCP

- Il vincolo **limita la quantità di dati non riscontrati** al mittente e quindi il **ritmo della trasmissione**
- In effetti (supponendo il buffer di ricezione infinito) ad ogni RTT il mittente riceve i riscontri per i dati e quindi lavora ad un ritmo pari a **CongWin/RTT** byte/sec



Controllo di congestione del TCP

In che modo un sender percepisce la congestione ?

- "evento di perdita" = timeout o ACK Duplicati (3ACK)
- Quando c'è congestione eccessiva uno o più buffer dei router durante il percorso traboccano → **perdita datagramma**
- TCP utilizza i riscontri per scatenare (temporizzare) gli incrementi della finestra di congestione
- TCP è auto-temporizzante (self-clocking)

Controllo di congestione del TCP

Quale algoritmo usare per cambiare il ritmo di invio in funzione della congestione percepita ?

ALGORITMI DI CONTROLLO DELLA CONGESTIONE

1. Congestion Avoidance (**AIMD** additive-increase/multiplicative-decrease)
Incremento additivo, decremento moltiplicativo
Per ogni calcolo di RTT, la finestra di congestione aumenta di 1.
2. Partenza lenta (**SLOW START**)
Incremento moltiplicativo
Per ogni ACK ricevuto, la finestra di congestione raddoppia fino ad arrivare alla soglia.
3. Fast recovery (**SLOW START**)
facoltativa

Slow Start e Congestion Avoidance

Due variabili importanti:

Congwin = la finestra di congestione corrente;

Threshold = limite tra incremento moltiplicativo e incremento additivo.

- "La finestra di congestione (cwnd) è un limite lato mittente sulla quantità di dati che il mittente può trasmettere alla rete prima di ricevere un riscontro (ACK)»
- è un parametro all'interno dello stack di rete TCP - il suo valore non viene trasmesso in pacchetti di rete, quindi non lo vedrai come un campo nella dissezione di un pacchetto TCP.

Slow Start e Congestion Avoidance

- Due variabili importanti:
 - **Congwin** = la finestra di congestione corrente;
 - **Threshold** = limite tra incremento moltiplicativo e incremento additivo.
- al di **sotto** della threshold si adotta la **slow start**
- al di **sopra** della threshold si adotta la **congestion avoidance**
- **slow start**, **Congwin** cresce **geometricamente**;
- **congestion avoidance** **Congwin** cresce **additivamente**.
- N.B: **Congwin** e **Threshold** varieranno nel corso del tempo di vita di una connessione TCP!

Reti di elaboratori

Incremento additivo, decremento moltiplicativo (AIMD)

- TCP **dimezza il valore** attuale della finestra di congestione dopo un riscontro di perdita
- La finestra non può scendere sotto a un MSS
- "Riaumento" lento del ritmo di un MSS per ogni tempo di round-trip fino a quando non si verificano ulteriori perdite (PREVENZIONE DELLA CONGESTIONE)

Fausto Marcantoni

Chapter 3 TRANSPORT PROTOCOL

3.193

193

Reti di elaboratori

Incremento additivo, decremento moltiplicativo (AIMD)

ADDITIVE INCREASE

Sorgente

Destinazione

Fausto Marcantoni

Chapter 3 TRANSPORT PROTOCOL

3.194

194

Incremento additivo, decremento moltiplicativo (AIMD)

Se il traffico di rete fosse uniforme, l'algoritmo dovrebbe convergere ad un valore che mantiene il traffico pressoché costante, eventualmente al limite della congestione

L'approccio del decremento moltiplicativo stabilisce che:

il mittente dimezza la dimensione della finestra di perdita fino ad un minimo di un MSS

evento di perdita:

```
CongWin = CongWin / 2;
if (CongWin < MSS)
    CongWin = MSS;
```

Incremento additivo, decremento moltiplicativo (AIMD)

Quando un mittente non rileva eventi di perdita, allora può immaginare che esista della banda disponibile che potrebbe essere sfruttata

Per questo **aumenta il valore della finestra di congestione**

La rilevazione di una assenza di eventi di perdita viene effettuata alla ricezione di un ACK

Se ricevo gli ACK significa che tutto OK

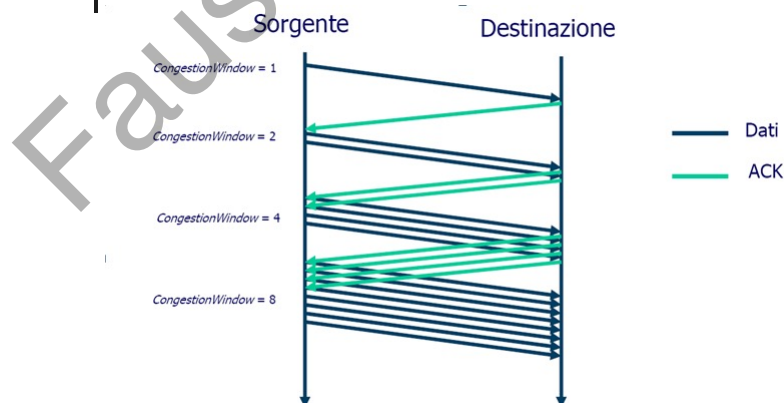
L'aumento della finestra di congestione è pari ad un MSS
in realtà l'algoritmo è un poco più complesso

ricezione di un ACK: // ovviamente non duplicato!
 $\text{CongWin} = \text{MSS} \cdot (\text{MSS} / \text{CongWin});$

Partenza lenta

- **Quando si inizia una connessione TCP, il valore della finestra di congestione è posto, prudenzialmente, a 1 MSS**
 - Se la rete è libera, il normale incremento lineare renderebbe lenta la connessione iniziale, con un ritmo che piano piano (linearmente) cresce fino a raggiungere il massimo prima di congestionare la rete
- Invece, nella fase iniziale, **ogni incremento è un raddoppio** (incremento esponenziale)
 - la fase iniziale termina quando si rileva il primo evento di perdita
- Questa fase iniziale, caratterizzata dal **raddoppio del valore ad ogni incremento della finestra di congestione**, è detta **partenza lenta** (slow start – SS)

Partenza lenta



$\text{CongestionWindow} = \text{CongestionWindow} * 2$
All'arrivo di ogni ACK

Reazioni ad eventi di timeout

- Dopo un timeout si entra in una fase di partenza lenta
 - Finestra di congestione ad 1 MSS
 - Aumento esponenziale
- Viene fissata una variabile chiamata **soglia (threshold)** che determina la dimensione della finestra alla quale deve terminare la partenza lenta ed iniziare la prevenzione della congestione
- Questa variabile è posta **inizialmente ad un valore grande** in modo che non abbia alcun effetto iniziale
- Ad ogni perdita il valore di soglia è posto uguale **alla metà del valore dell'attuale** finestra di congestione

FUNZIONAMENTO DI SINTESI DELL'ALGORITMO

Finestra di congestione **sotto la soglia** → **partenza lenta ed incremento esponenziale**

Finestra di congestione **sopra la soglia** → **prevenzione congestione ed aumento lineare**

Evento di perdita riscontrato con **ACK consecutivi** → **dimezzata la finestra di congestione ed il valore di soglia**

Evento di perdita di tipo **timeout** → **finestra di congestione pari a 1 MSS e valore di soglia pari a metà finestra di congestione**

Reti di elaboratori

FUNZIONAMENTO DI SINTESI DELL'ALGORITMO (TCP TAHOE & TCP RENO)

- PERCHE' COMPORTAMENTI DIVERSI PER LO STESSO FENOMENO DI PERDITA ?
- **TCP TAHOE**
 - taglia incondizionatamente la finestra di congestione ad 1 MSS ed entra nella fase di partenza lenta sempre
- **TCP RENO**
 - elimina la fase di partenza lenta ed effettua un ripristino rapido (fast recovery)

Fausto Marcantoni

Chapter 3 TRANSPORT PROTOCOL

3.201

201

Reti di elaboratori

Congestion control - timeout

Congestion Window

The graph shows the Congestion Window (Y-axis) over time (X-axis). It starts with a 'Slow Start' phase where the window increases exponentially. This is followed by a 'Timeout' phase where the window drops to 1. Then, it enters an 'Additive Increase' phase where the window increases linearly until it reaches the 'Congestion Threshold'. After the threshold, it enters a 'multiplicative decrease' phase where the window is halved. The graph then shows a 'slow start' phase where the window increases exponentially again until it reaches the threshold.

Additive increase multiplicative decrease

threshold = $\max(CW/2, 2)$

CW = 1

slow start fino alla threshold

Timeout

Additive Increase

Congestion Threshold

Tempo

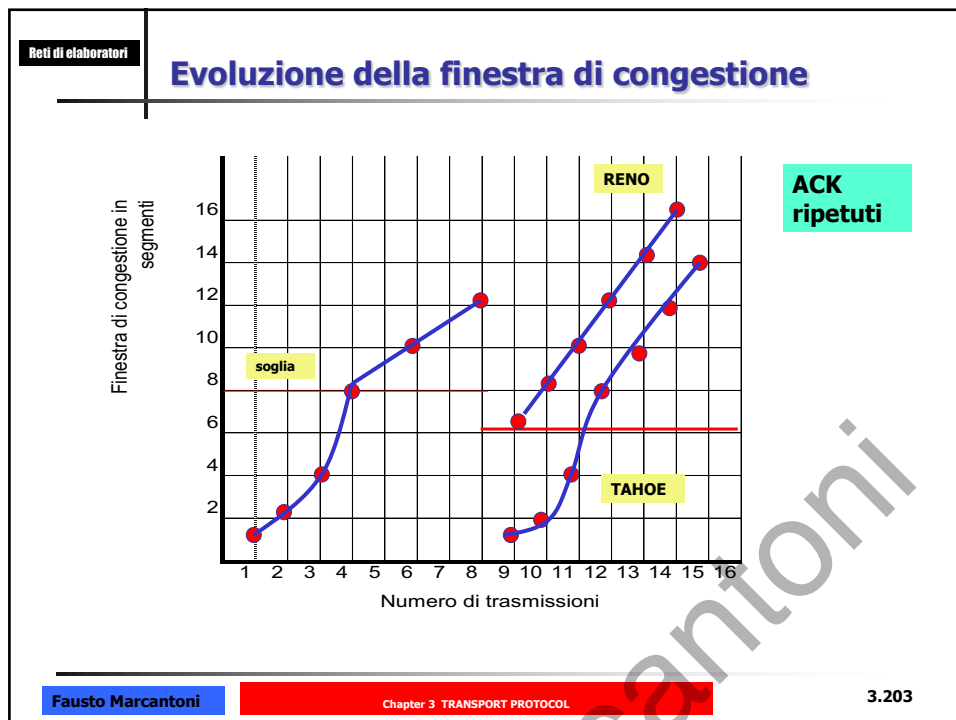
$CongestionThreshold = \text{Max}(CongestionWindow/2, 2)$

Fausto Marcantoni

Chapter 3 TRANSPORT PROTOCOL

3.202

202



203

Reti di elaboratori

Ricapitolando...

- Finestra di congestione sotto la soglia:
 - Slow start
 - Crescita esponenziale della finestra
- Finestra di congestione sopra la soglia:
 - Prevenzione della congestione
 - Crescita lineare della finestra
- Evento di perdita dedotto da ACK duplicato 3 volte:
 - Soglia posta alla metà del valore attuale della finestra
 - TCP Reno:
 - Finestra posta pari alla soglia
 - TCP Tahoe:
 - Finestra posta pari ad un segmento (MSS -- Maximum Segment Size)
- Evento di perdita dedotto da timeout:
 - Soglia posta alla metà del valore attuale della finestra
 - Finestra posta pari ad un segmento (MSS -- Maximum Segment Size)

Fausto Marcantoni Chapter 3 TRANSPORT PROTOCOL 3.204

204

Reti di elaboratori

TCP Congestion Control

TCP Congestion Control

TCP congestion control is described in Chapter 3 of the text. In this interactive animation, you can view how TCP behaves when multiple clients are sending data over the same link. You may add stations either before or during the simulation. (The default is one station.) When the maximum amount of bandwidth is consumed, all of the sending clients reduce their transmissions rates, depending on the recovery method (Reno or Tahoe).

The green background in the graph shows the total amount of consumed bandwidth. The "3 ack's" button and the "timeout" button trigger the corresponding event for that particular workstation. The "Reno" and "Tahoe" buttons set the recovery method for that particular workstation. The sliders can be used to adjust the speed and the maximum bandwidth.

https://media.pearsoncmg.com/curriculum/intl/it/lab/9788891902559_KUROSE_ROSS/studente/interactive%20animations/tcp-congestion/index.html

Fausto Marcantoni

Chapter 3 TRANSPORT PROTOCOL

3.205

205

Reti di elaboratori

Attuali implementazioni

Due algoritmi per evitare la congestione sono ampiamente utilizzati nei sistemi operativi comuni:

- CUBIC (Linux)
- CTCP (Microsoft Windows)

L'obiettivo di questi algoritmi è mantenere un throughput il più vicino possibile alla larghezza di banda disponibile.

<https://companyprideplatform.org/it/tcp-congestion-control-italiano/>

Algoritmi storici

- TCP tahoe
- Reno
- Nuovo Reno
- Vegas
- Westwood (+)
- TCP SACK

Attuali implementazioni

- BIC
- CUBIC
- TCP composto

Fausto Marcantoni

Chapter 3 TRANSPORT PROTOCOL

3.206

206

Osservazioni

Entrambi i sistemi operativi utilizzano CUBIC di default.

Windows 10 Client / Server

Nella versione di windows 10 "Client" non è possibile cambiare l'algoritmo assegnato in quanto la variabile di ambiente è settata su "read only".

Nella versione server è possibile modificare il profilo utilizzato .

Da Powershell : `Get-NetTcpSetting` (il Profilo di Default usato è quello INTERNET che utilizza appunto cubic.)

LINUX

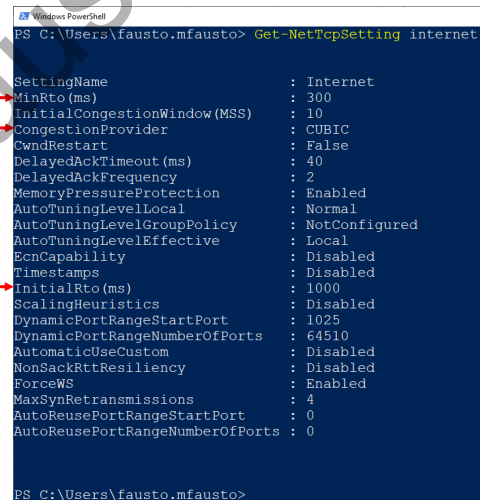
In linux è possibile visualizzare l'algoritmo in uso con il comando

```
cat /proc/sys/net/ipv4/tcp_congestion_control
```

mentre è possibile modificarlo usando il comando come amministratore es:

```
sysctl net.ipv4.tcp_congestion_control=reno
```

Get-NetTcpSetting internet



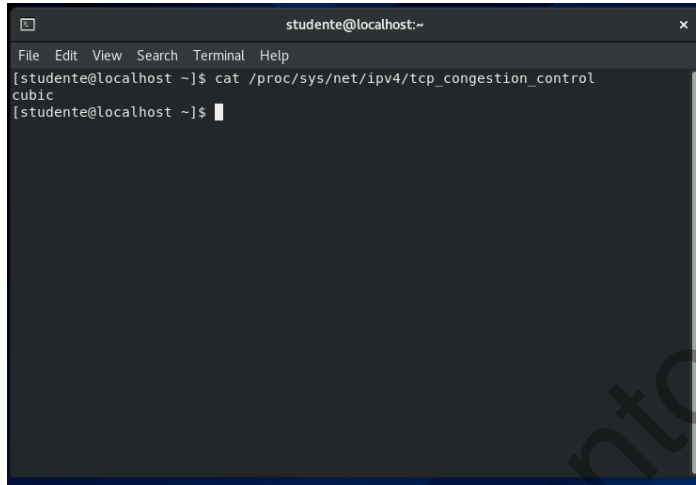
```
PS C:\Users\fausto.mfausto> Get-NetTcpSetting internet

SettingName           : Internet
MinRto (ms)           : 300
InitialCongestionWindow (MSS) : 10
CongestionProvider     : CUBIC
CwndRestart            : False
DelayedAckTimeout (ms) : 40
DelayedAckFrequency    : 2
MemoryPressureProtection : Enabled
AutoTuningLevelLocal   : Normal
AutoTuningLevelGroupPolicy : NotConfigured
AutoTuningLevelEffective : Local
EcnCapability          : Disabled
Timestamps             : Disabled
InitialRto (ms)        : 1000
ScalingHeuristics      : Disabled
DynamicPortRangeStartPort : 1025
DynamicPortRangeNumberOfPorts : 64510
AutomaticUseCustom     : Disabled
NonSackRttResiliency   : Disabled
ForceWS                : Enabled
MaxSynRetransmissions  : 4
AutoReusePortRangeStartPort : 0
AutoReusePortRangeNumberOfPorts : 0

PS C:\Users\fausto.mfausto>
```

tcp_congestion_control

```
cat /proc/sys/net/ipv4/tcp_congestion_control
```



```
studente@localhost:~  
File Edit View Search Terminal Help  
[studente@localhost ~]$ cat /proc/sys/net/ipv4/tcp_congestion_control  
cubic  
[studente@localhost ~]$
```

Verifica cosa espone davvero il tuo sistema

```
# Vedi se esiste la famiglia net.inet.tcp  
sysctl net.inet.tcp
```

```
# Elenco completo dei TCP sysctl disponibili  
sysctl -a | grep '^net\.inet\.tcp'
```

```
# Cerca chiavi legate alla congestione (se ci fossero)  
sysctl -a | grep -iE 'cong|cwnd|cc\.|cubic|bbr|reno'
```

provare con iperf3

```
iperf3 -c iperf3.moji.fr -t 20 -i 1
```

```
faustomarcantoni@MacBook-Pro-di-Fausto ~ % iperf3 -c iperf3.moji.fr -t 20 -i 1
Connecting to host iperf3.moji.fr, port 5201
[ 7] local 193.205.92.88 port 51875 connected to 45.147.210.189 port 5201
[ ID] Interval      Transfer      Bitrate
[ 7] 0.00-1.00 sec  43.0 MBytes  360 Mbits/sec
[ 7] 1.00-2.00 sec  49.5 MBytes  416 Mbits/sec
[ 7] 2.00-3.00 sec  51.1 MBytes  429 Mbits/sec
[ 7] 3.00-4.01 sec  51.2 MBytes  429 Mbits/sec
[ 7] 4.01-5.00 sec  49.0 MBytes  413 Mbits/sec
[ 7] 5.00-6.00 sec  50.5 MBytes  423 Mbits/sec
[ 7] 6.00-7.00 sec  51.1 MBytes  429 Mbits/sec
[ 7] 7.00-8.00 sec  51.5 MBytes  432 Mbits/sec
[ 7] 8.00-9.00 sec  51.1 MBytes  429 Mbits/sec
[ 7] 9.00-10.00 sec 46.5 MBytes  398 Mbits/sec
[ 7] 10.00-11.00 sec 47.2 MBytes  396 Mbits/sec
[ 7] 11.00-12.00 sec 48.5 MBytes  407 Mbits/sec
[ 7] 12.00-13.00 sec 50.4 MBytes  422 Mbits/sec
[ 7] 13.00-14.00 sec 50.4 MBytes  422 Mbits/sec
[ 7] 14.00-15.00 sec 49.5 MBytes  416 Mbits/sec
[ 7] 15.00-16.00 sec 49.8 MBytes  418 Mbits/sec
[ 7] 16.00-17.00 sec 49.5 MBytes  415 Mbits/sec
[ 7] 17.00-18.00 sec 51.5 MBytes  433 Mbits/sec
[ 7] 18.00-19.00 sec 51.0 MBytes  427 Mbits/sec
[ 7] 19.00-20.00 sec 50.9 MBytes  428 Mbits/sec
-----
[ ID] Interval      Transfer      Bitrate          sender
[ 7] 0.00-20.00 sec  993 MBytes  417 Mbits/sec
[ 7] 0.00-20.03 sec  992 MBytes  415 Mbits/sec          receiver

iperf Done.
faustomarcantoni@MacBook-Pro-di-Fausto ~ %
```

<https://iperf.fr/iperf-servers.php>

CUBIC è un derivato meno aggressivo e più sistematico di BIC TCP

la dimensione della finestra è una funzione cubica del tempo dall'ultimo evento di congestione, con il punto di flesso impostato sulla dimensione della finestra prima dell'evento.

E' una funzione cubica, ci sono due componenti per la crescita della finestra.

1. La prima è una porzione concava in cui le dimensioni della finestra aumentano rapidamente fino a raggiungere le dimensioni prima dell'ultimo evento di congestione.
2. La prossima è la crescita convessa in cui CUBIC cerca una maggiore larghezza di banda, prima lentamente e poi molto rapidamente

CUBIC - approfondimenti

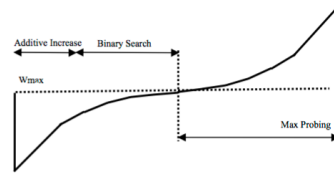


Fig. 1: The Window Growth Function of BIC

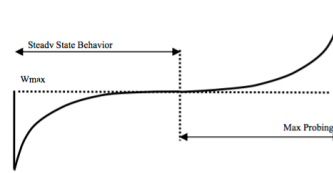


Fig. 2: The Window Growth Function of CUBIC

Cubic functions can have both convex and concave portions, meaning that they can grow more slowly in some portions (concave) and more quickly in others (convex). In addition to that CUBIC also has a TCP friendly region that operates when window is small to ensure that CUBIC is not penalized relative to regular TCP. More specifically, the window size of standard TCP in terms of time elapsed.

<https://tools.ietf.org/html/rfc8312>

https://en.wikipedia.org/wiki/CUBIC_TCP

<https://gastack.it/superuser/355143/on-the-performance-of-tcp-implementations-of-linux-and-windows>

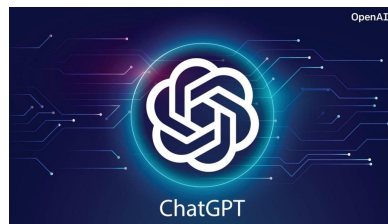
Esercitazione/Laboratorio

An Analysis of TCP Congestion Control Mechanisms using Wireshark.

<https://www.scribd.com/doc/55589000/An-Analysis-of-TCP-Congestion-Control-Mechanisms-using-Wireshark>

Wireshark IO Graph - TCP Performance rwin cwnd

<https://www.google.com/search?q=Wireshark+IO+Graph+-+TCP+Performance+rwin+cwnd&client=firefox-b-d&uact=5&oq=Wireshark+IO+Graph+-+TCP+Performance+rwin+cwnd>



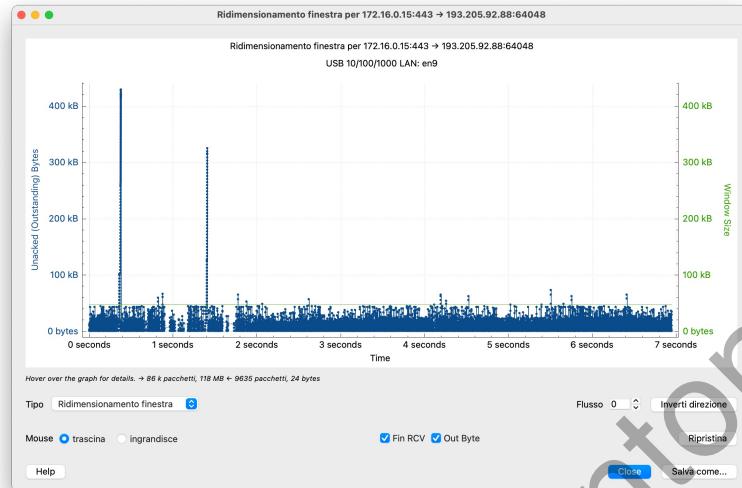
Wireshark IO Graph - Prestazioni TCP rwin cwnd (Finestra di Ricezione e Finestra di Congestione)

Per visualizzare le metriche delle prestazioni TCP come la **finestra di ricezione (rwin)** e una stima della **finestra di congestione (cwnd)**, i "Grafici di flusso TCP" (TCP Stream Graphs) dedicati in Wireshark sono più efficaci del Grafico I/O generico. *Il valore della finestra di congestione è un valore del kernel lato mittente e non è presente nei pacchetti di rete stessi, quindi non può essere tracciato direttamente nel Grafico I/O utilizzando un filtro di visualizzazione.*

Rappresentare graficamente la Finestra di Ricezione (rwin) e i Byte in Sospeso (Bytes in Flight)

Il grafico più utile per analizzare i problemi relativi alle dimensioni della finestra è il grafico "Window Scaling" (Scalatura della Finestra) o "Bytes in Flight" (Byte in Sospeso). Questo grafico traccia la finestra pubblicizzata dal ricevitore (rwin) rispetto alla quantità effettiva di dati non confermati che il mittente ha "in sospeso" sulla rete.

1. **Seleziona un pacchetto** appartenente allo specifico flusso TCP che desideri analizzare nel pannello dell'elenco dei pacchetti.
2. Vai su **Statistiche > Grafici di flusso TCP (TCP Stream Graphs) > Window Scaling** (o Bytes in Flight).
3. Si aprirà una nuova finestra con un grafico:
 1. La **linea verde** rappresenta la dimensione della finestra pubblicizzata dal ricevitore (rwin) nel tempo, mostrando quanti dati il ricevitore può accettare.
 2. La **linea blu** rappresenta i "byte in sospeso" (dati non confermati sulla rete), che fornisce una buona stima della velocità di invio corrente del mittente, spesso limitata dalla finestra di congestione (cwnd) o dalla rwin.
4. **Analisi:** Se la linea blu raggiunge costantemente la linea verde e poi si appiattisce, indica che la finestra del ricevitore è il collo di bottiglia (uno scenario "window limited"). Gli spazi tra le due linee possono indicare problemi di latenza della rete o un mittente limitato dalla propria finestra di congestione.

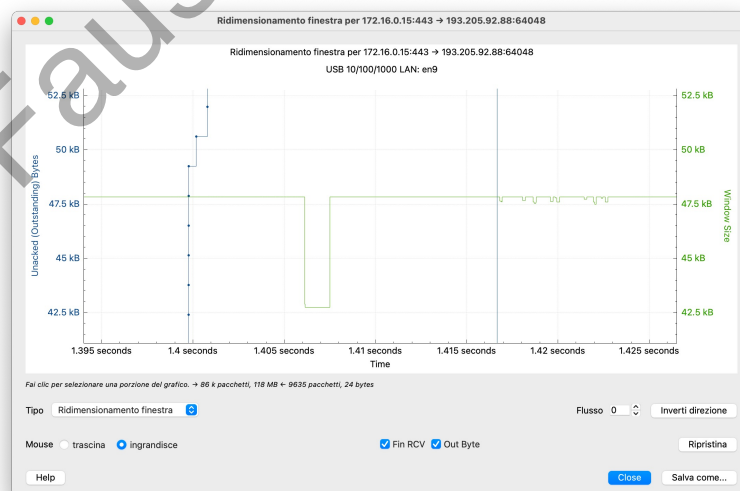


Fausto Marcantoni

Chapter 3 TRANSPORT PROTOCOL

3.217

217



Fausto Marcantoni

Chapter 3 TRANSPORT PROTOCOL

3.218

218

