



Protocolli di trasporto in Internet: TCP e UDP

Luca Veltri

(mail.to: luca.veltri@unipr.it)

Corso di Reti di Telecomunicazione, a.a. 2012/2013

<http://www.tlc.unipr.it/veltri>

Sommario

- Protocolli di trasporto su rete IP
- User Datagram Protocol (UDP)
- Transmission Control Protocol (TCP)
- Network Address Translator (NAT)

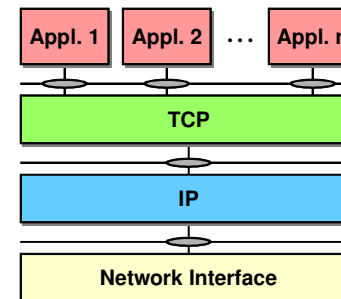
2

Strato di Trasporto in Internet

- Obiettivo: fornire una comunicazione end-to-end ai processi applicativi
- Per l'architettura Internet sono stati definiti due protocolli di trasporto:
 - User Datagram Protocol (UDP - RFC 768)
 - Transmission Control Protocol (TCP - RFC 793 e succ.)
- Ulteriori protocolli di trasporto sono stati proposti in ambito IETF
 - e.g. Stream Control Transmission Protocol (SCTP - RFC 2960, 3257, 3286)
- Funzione comune:
 - **Permettono di distinguere, all'interno di uno stesso host, il processo applicativo destinatario (o sorgente) dei dati**
 - smistamento dei dati fra differenti sorgenti o destinazioni all'interno dello stesso host
 - ogni host contiene un insieme di punti logici di accesso "port" (porte)
 - ad ogni processo applicativo viene associato un port number differente
 - i port number rappresentano insieme al protocol id e all'indirizzo IP l'indirizzo SAP (Service Access Point) tra strato di trasporto e applicativo

3

Indirizzamento TCP/UDP

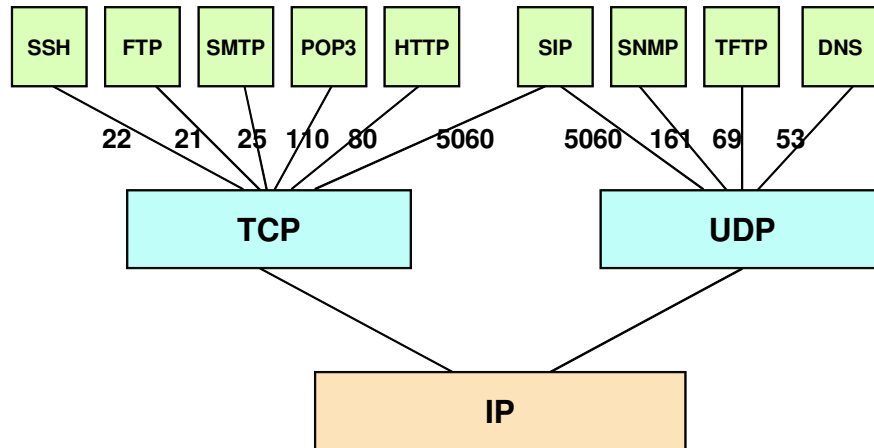


- Port number
 - **identificativo di un utente TCP (o UDP), ovvero il processo applicativo**
- Socket address
 - **indirizzo completo in TCP/IP (T-SAP)**
 - **(IP address , protocol, port)**
- L'assegnazione logica del numero di porta alla specifica applicazione può essere
 - statica
 - **gli identificativi (port number) sono staticamente associati alle applicazioni (di solito solo il lato server)**
 - **sono spesso utilizzati identificativi inferiori a 256**
 - dinamica
 - **gli identificativi sono ottenuti dal sistema operativo al momento dell'apertura della connessione (in genere identificativi maggiori di 1024)**

4

Well Known Ports

- Sono associate agli applicativi principali, esempio



5

User Datagram Protocol (UDP)

User Datagram Protocol (UDP)

- UDP (User Datagram Protocol)
- Consente alle applicazioni di scambiare messaggi singoli (protocollo Message oriented)
- Fornisce un livello di servizio minimo:
 - **E' un protocollo senza connessione**
 - **Non supporta meccanismi di riscontro e recupero d'errore**
 - **Aggiunge solo due funzionalità a quelle di IP**
 - multiplexing delle informazioni tra le varie applicazioni tramite il concetto di porta
 - checksum per verificare l'integrità dei dati
 - **Può essere utilizzato per trasmissioni multicast (a differenza di TCP)**
 - stesso messaggio UDP a più destinazioni, indirizzate attraverso un indirizzo multicast
<IP multicast address, UDP port>

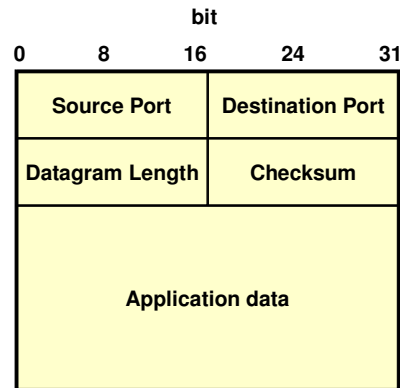
7

User Datagram Protocol (UDP)

- E' utilizzato per il supporto di transazioni semplici tra applicativi, esempio
 - **risoluzione di indirizzi (DNS)**
 - **messaggi di management (e.g. SNMP)**
- Particolarmente adatto per applicazioni Real-Time:
 - **assenza di ritrasmissione**
 - **assenza di meccanismi di controllo di flusso e congestione**
 - **minore overhead**

8

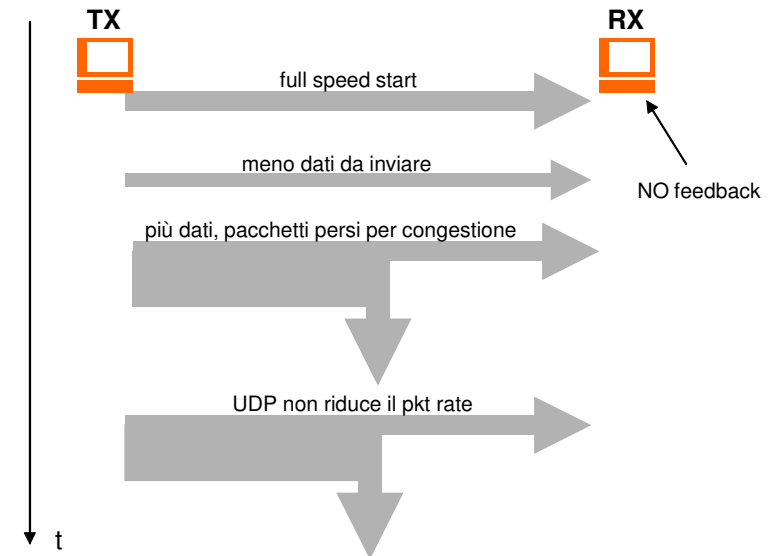
Formato del Datagramma UDP



- Source Port (16 bit) e Destination Port (16 bit)
 - identificano i processi sorgente e destinazione dei dati
- Datagram Length (16 bit)
 - è la lunghezza totale del datagramma, compreso l'header UDP
 - $UDPLen = IPLen - IPHLen$
- Checksum (16 bit)
 - protegge tutto il datagramma UDP (header UDP + dati UDP) + pseudo header IP (per proteggere info di indirizzamento)

9

UDP: comportamento in caso di congestione



10

Transmission Control Protocol (TCP)

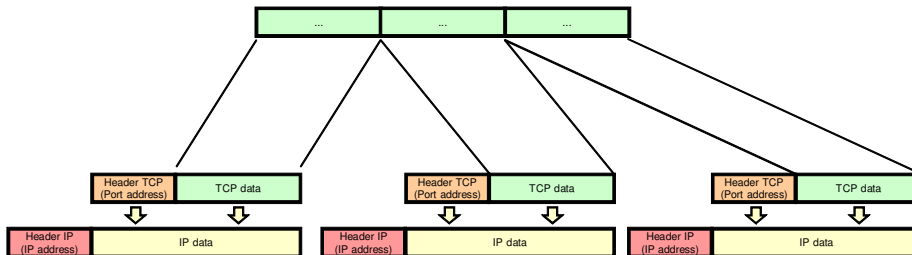
Transmission Control Protocol (TCP)

- E' un protocollo end-to-end con connessione e offre un servizio stream-oriented affidabile
- Trasferisce un flusso informativo bi-direzionale non strutturato (**byte stream**) tra due host ed effettua operazioni di moltiplicazione e de-moltiplicazione
- Non prevede nodi intermedi (TCP) e quindi non implementa la funzione di commutazione
- Funzioni eseguite
 - controllo e recupero di errore
 - controllo di flusso
 - controllo di congestione
 - ri-ordinamento delle unità informative
 - indirizzamento (di uno specifico utente all'interno di un host)

12

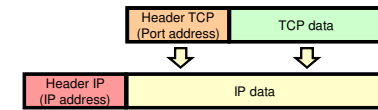
Unità di dati del TCP

- TCP è un protocollo "Stream oriented"
 - Il TCP interpreta lo stream dati come sequenza di ottetti (byte)
 - Lo stream dati è suddiviso in segmenti
 - Ogni segmento è trasferito in un pacchetto IP
 - dagli applicativi viene visto come un flusso continuo (stream) di byte



13

Unità di dati del TCP



0		4		8		16		24		31	
Source Port						Destination Port					
Sequence Number											
Acknowledgment Number											
HLEN		Reserved		Flags		Window					
Checksum						Urgent Pointer					
Options (if any)						Padding					
Data											
...											
Data											

14

Unità di dati del TCP (cont.)

- Source Port (16 bit) e Destination Port (16 bit)
 - identificano i processi sorgente e destinazione dei dati
- Sequence Number (32 bit)
 - numero di sequenza in trasmissione
 - contiene il numero di sequenza del primo byte di dati contenuti nel segmento a partire dall'inizio della sessione
- Acknowledgement Number (32 bit)
 - numero di sequenza in ricezione
 - se ACK=1, contiene il numero di sequenza del prossimo byte che il trasmettitore del segmento si aspetta di ricevere
 - è possibile la modalità piggybacking di riscontro

15

Unità di dati del TCP (cont.)

- HLEN (4 bit)
 - contiene il numero di parole di 32 bit contenute nell'header TCP
 - l'intestazione TCP non supera i 60 byte ed è sempre un multiplo di 32
- Reserved (6 bit)
 - riservato per usi futuri, per ora contiene degli zeri
- Window (16 bit)
 - larghezza della finestra in byte (controllo di flusso orientato al byte)
 - è il numero di byte che, ad iniziare dal valore di Acknowledgement Number, il trasmettitore del segmento è in grado di ricevere
- Checksum (16 bit)
 - protegge l'intero segmento più alcuni campi dell'header IP (es. indirizzi)

16

Unità di dati del TCP (cont.)

- Flags or Control bits (6 bit): i bit di controllo sono:
 - **URG**: è uguale a uno quando il campo urgent pointer contiene un valore significativo
 - **ACK**: è uguale a uno quando il campo Acknowledgement Number contiene un valore significativo
 - **PSH**: è uguale a uno quando l'applicazione esige che i dati forniti vengano trasmessi e consegnati all'applicazione ricevente prescindendo dal riempimento dei buffer allocati fra applicazione e TCP e viceversa (solitamente infatti è il riempimento dei suddetti buffer che scandisce la trasmissione e la consegna dei dati)
 - **RST**: è uguale a uno in caso di richiesta di reset della connessione
 - **SYN**: è uguale a uno solo nel primo segmento inviato (per ogni verso) durante la fase di sincronizzazione fra le entità TCP
 - **FIN**: è uguale a uno quando la sorgente ha esaurito i dati da trasmettere

17

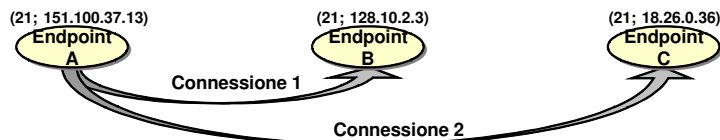
Unità di dati del TCP (cont.)

- Urgent Pointer (16 bit)
 - contiene il numero di sequenza dell'ultimo byte dei dati che devono essere consegnati urgentemente al processo ricevente
 - tipicamente sono messaggi di controllo (out-of-band traffic)
- Options (di lunghezza variabile)
 - sono presenti solo raramente
 - Esempi:
 - End of Option List, No-operation, Maximum Segment Size (MSS)
- Padding (di lunghezza variabile)
 - impone che l'intestazione abbia una lunghezza multipla di 32 bit

18

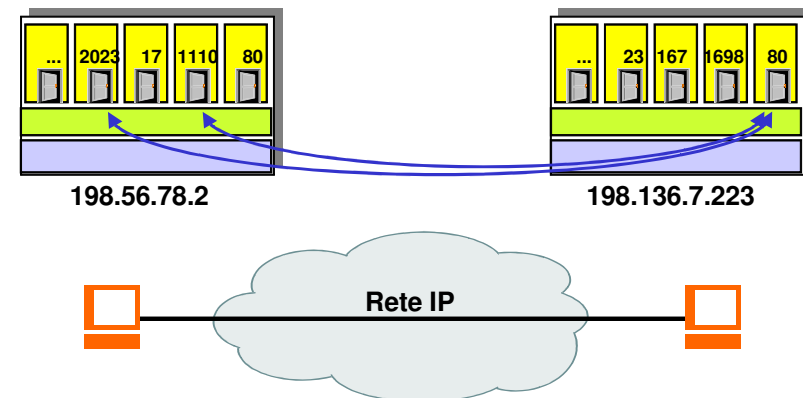
Connessioni TCP

- TCP identifica un "canale" di comunicazione con il nome di connessione
- Sono identificate dalla quadrupla: **<client IP address, client IP port, server IP address, server IP port>**
- Questa soluzione permette
 - A molti client diversi di accedere allo stesso servizio sullo stesso server
 - Allo stesso client di attivare più sessioni dello stesso servizio
- Indirizzamento TCP include indirizzi IP
 - TCP non è indipendente dalla strato sottostante
 - non richiede meccanismi aggiuntivi di Address Resolution



19

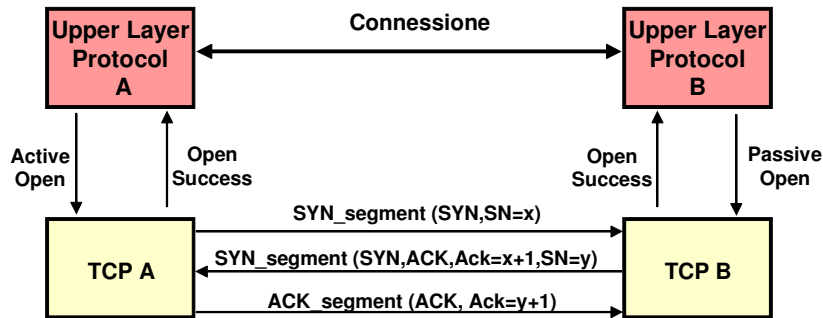
Connessioni TCP



20

Connessione TCP: apertura

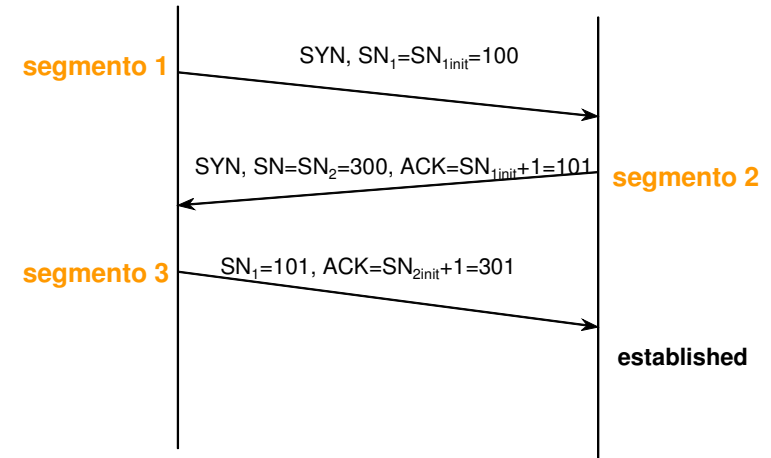
- Apertura = instaurazione della connessione (Setup)
- La sincronizzazione avviene con un meccanismo detto "three way handshaking"



21

Connessione TCP: apertura

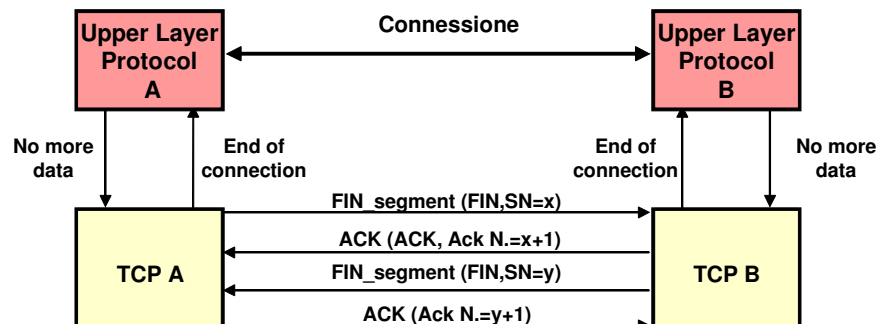
Three-way handshake



22

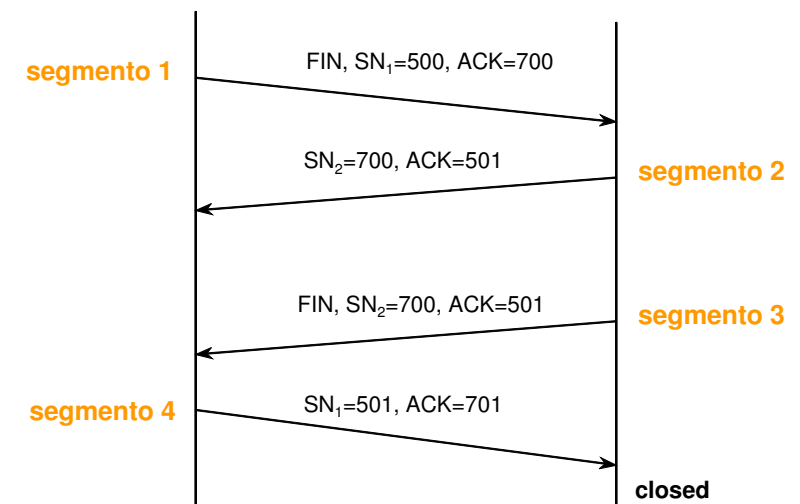
Connessione TCP: chiusura

- Chiusura = abbattimento della connessione (Teardown)
- Nella fase di rilascio le due vie sono chiuse indipendentemente
- Meccanismo del tipo two way handshaking



23

Connessione TCP: Teardown



24

Esempio: Instaurazione e abbattimento di una connessione (tcpdump)

Client=193.205.242.33
Server=193.205.242.11

```
[client]# telnet server 9
[client]# tcpdump
09:56:06.0094 client.1057 > server.9 : S 0:0(0) win 32120 <mss 1460>
09:56:06.0098 server.9 > client.1057: S 0:0(0) ack 128582 win 32120 <mss 1460>
09:56:06.0098 client.1057 > server.9 : . 1:1(0) ack 1 win 32120
...
09:56:22.4891 client.1057 > server.9 : F 1:1(0) ack 1 win 32120
09:56:22.4894 server.9 > client.1057: . 1:1(0) ack 2 win 32120
09:56:22.4898 server.9 > client.1057: F 1:1(0) ack 2 win 32120
09:56:22.4898 client.1057 > server.9 : . 2:2(0) ack 2 win 32120
```

25

netstat : TCP/UDP status

```
UDP:
Local Address      State
-----
*.talk             Idle
*.time             Idle
*.echo             Idle
*.discard          Idle

TCP:
Local Address      Remote Address    Swind Send-Q Rwind Recv-Q State
-----
*.ftp              *.*               0      0      0      0 LISTEN
*.telnet           *.*               0      0      0      0 LISTEN
*.pop3             *.*               0      0      0      0 LISTEN
*.smtp             *.*               0      0      0      0 LISTEN
*.80               *.*               0      0      0      0 LISTEN
cartesio.telnet    galileo.1262      7904   0      8760   0 ESTABLISHED
cartesio.54770     keplero.telnet    32120  0      8760   0 ESTABLISHED
cartesio.telnet    galileo.1328      8179   0      8760   0 ESTABLISHED
cartesio.54771     keplero.telnet    32120  0      8760   0 ESTABLISHED
cartesio.telnet    galileo.1330      8513   0      8760   0 ESTABLISHED
cartesio.43390     pascal.80         8760   0      8760   0 CLOSE_WAIT
cartesio.pop3      platone.1457      17346  0      8760   0 TIME_WAIT
cartesio.smtp      atlante.57897     8760   0      8760   0 TIME_WAIT
cartesio.54777     newton.smtp       8760   0      8760   0 TIME_WAIT
```

26

TCP Protocol Overview

- Il TCP è un protocollo affidabile in quanto:
 - Quando invia un segmento fa partire un timer
Se non viene ricevuto un riscontro prima della scadenza del timeout il segmento viene ritrasmesso
 - Quando si riceve un segmento dati viene inviato un riscontro (ack)
Il riscontro viene ritardato di una frazione di secondo (delayed acknowledgement)
 - Se un segmento arriva con un checksum non valido, il TCP scarta il segmento e non invia il riscontro
 - Si attende che il trasmettitore vada in timeout e ritrasmetta il segmento
 - Poiché TCP è imbustato su IP, e poiché i pacchetti IP possono arrivare fuori sequenza, i segmenti TCP vengono riordinati in ricezione e passati in ordine all'applicativo
 - Ciascuna estremità di una connessione TCP ha a disposizione un buffer di ricezione di dimensione finita
il TCP ricevente non consente alla stazione trasmittente di inviare una quantità di dati che superi lo spazio disponibile nel buffer

27

Controllo di errore

- Il controllo di errore (rivelazione e recupero) è necessario per risolvere potenziali situazioni di perdita/errore introdotte e non recuperate dagli strati inferiori
 - IP non è affidabile, i datagrammi possono andare persi, essere ritardati, duplicati o consegnati fuori sequenza
- TCP utilizza un meccanismo di recupero di errore (ARQ) a finestra scorrevole (Sliding Window)
 - sono presenti esclusivamente riscontri positivi (Acknowledgement)
 - la parte dati dei segmenti TCP contengono ottetti (byte) numerati sequenzialmente a partire dal numero scelto durante il 3-way handshaking
 - i segmenti portano un ACK Number (utilizzato anche dal controllo di flusso e dal controllo di congestione)

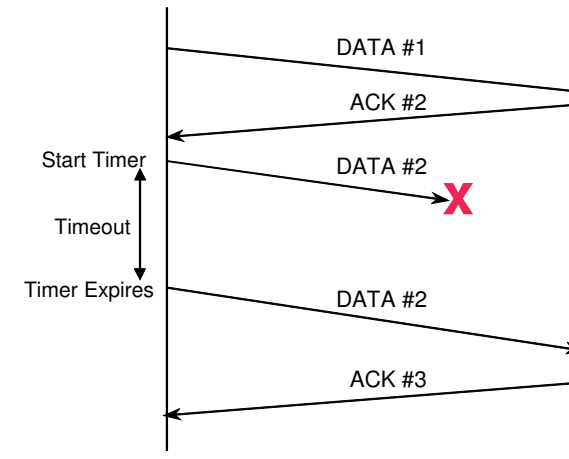
28

Controllo di errore (cont.)

- **ritrasmissione innescata dalla mancata ricezione degli ACK entro un fissato tempo limite (timeout)**
 - dopo l'invio di un segmento si attende un certo tempo e, se nessuna conferma (ACK) viene ricevuta nel frattempo, si assume che il segmento si sia perso
- **finestra di trasmissione di dimensione variabile (twin)**
 - possono essere inviati sino a *twin* byte consecutivi non riscontrati
 - $twin = \min(fwin, cwin)$
- **Un riscontro con ACK Number=N significa che**
 - sono riscontrati tutti gli ottetti ricevuti fino a quello numerato con $N-1$
 - il trasmittente è autorizzato a trasmettere fino a ulteriori *twin* ottetti, ovvero fino all'ottetto numerato con $N+twin-1$

29

Controllo di errore



30

Retransmission Timeout (RTO)

- Il dimensionamento del RTO è un aspetto critico nelle prestazioni del TCP
 - **se troppo piccolo, alcuni segmenti in ritardo, a causa dei ritardi di trasmissione o congestione nella rete, potrebbero essere considerati persi e quindi ritrasmessi, ciò aumenterebbe la congestione e di conseguenza le ritrasmissioni facendo tendere la portata a zero**
 - **se troppo grande, la risposta ad un evento di perdita sarebbe troppo lenta con conseguente perdita di efficienza**
- Il RTO è fissato con uno schema adattativo
- Il TCP misura dinamicamente l'Round Trip Time (RTT), definito come il ritardo tra l'invio di un segmento e la ricezione del relativo ACK

$$E\{RTT\}_{k+1} = \frac{1}{K+1} \sum_{i=1}^{K+1} RTT_i = \frac{K}{K+1} E\{RTT\}_k + \frac{1}{K+1} RTT_k$$

$$\widehat{RTT}_k = \alpha E\{RTT\}_{k-1} + (1-\alpha) RTT_k \quad (\text{normalmente } 0.8 \leq \alpha \leq 0.9)$$

$$RTO_k = \beta \widehat{RTT}_k \quad (1.3 \leq \beta \leq 2.0)$$

31

Controllo di Flusso e di Congestione

- Il controllo di flusso ha lo scopo di limitare il flusso dei dati, in base alle esigenze del terminale ricevente, prescindendo dal traffico presente nella rete
 - **per permettere la comunicazione tra terminali di dimensione e velocità molto diverse tra loro**
- Il controllo della congestione ha lo scopo di recuperare eventuali situazioni di sovraccarico (congestione) della rete

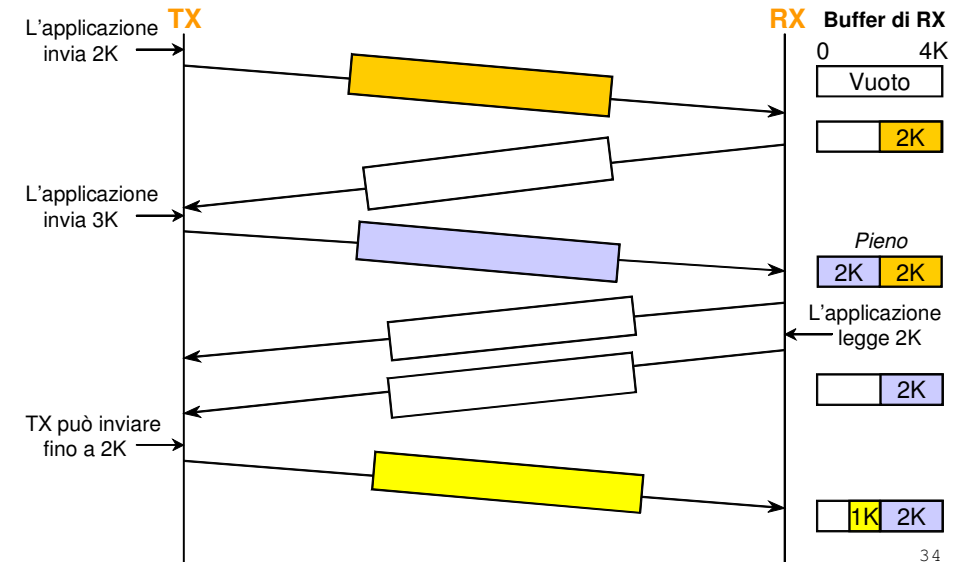
32

Controllo di Flusso

- TCP utilizza un controllo di flusso a finestra basato su finestra scorrevole di ampiezza variabile (*fwin*)
- Anche il controllo di flusso opera a livello di ottetti (byte)
- I segmenti portano oltre al ACK Number (utilizzato pure dal controllo di errore) anche un Flow Window
- Un riscontro (ACK Number=N e Window=w) significa che
 - sono riscontrati tutti gli ottetti ricevuti fino a quello numerato con N-1
 - il trasmittente è autorizzato a trasmettere fino a ulteriori w ottetti, ovvero fino all'ottetto numerato con N+w-1
 - $tw = \min(fwin, cwin)$
 - $fwin = w$

33

Controllo di flusso: esempio



34

Esempio: trasmissione di un blocco di dati (1/2)

- esempio di trasmissione di 8k bytes con MSS 1460

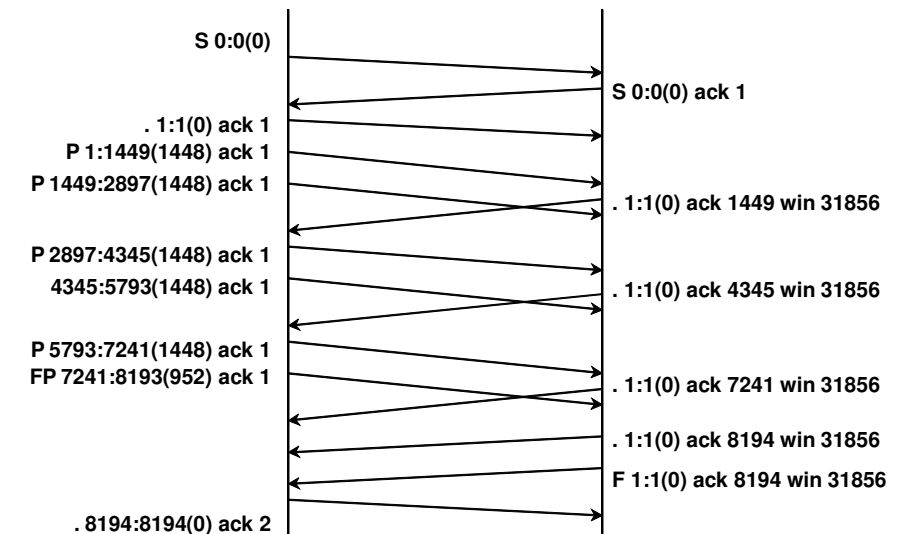
```
[client]# ttcp -t -p 5555 -s -nl server
```

```
timestamp addr.port > addr.port: flag SN:SN ack flow-window <options>
```

```
14.281143 client.1079 > server.5555: S 0:0(0) win 32120 <mss 1460>
14.281452 server.5555 > client.1079: S 0:0(0) ack 1 win 31856 <mss 1460>
14.281504 client.1079 > server.5555: . 1:1(0) ack 1 win 32120
14.282443 client.1079 > server.5555: P 1:1449(1448) ack 1 win 32120
14.282469 client.1079 > server.5555: P 1449:2897(1448) ack 1 win 32120
14.284997 server.5555 > client.1079: . 1:1(0) ack 1449 win 31856
14.285056 client.1079 > server.5555: P 2897:4345(1448) ack 1 win 32120
14.285068 client.1079 > server.5555: P 4345:5793(1448) ack 1 win 32120
14.287628 server.5555 > client.1079: . 1:1(0) ack 4345 win 31856
14.287672 client.1079 > server.5555: P 5793:7241(1448) ack 1 win 32120
14.287683 client.1079 > server.5555: FP 7241:8193(952) ack 1 win 32120
14.290019 server.5555 > client.1079: . 1:1(0) ack 7241 win 31856
14.291264 server.5555 > client.1079: . 1:1(0) ack 8194 win 31856
14.292922 server.5555 > client.1079: F 1:1(0) ack 8194 win 31856
14.292961 client.1079 > server.5555: . 8194:8194(0) ack 2 win 32120
```

35

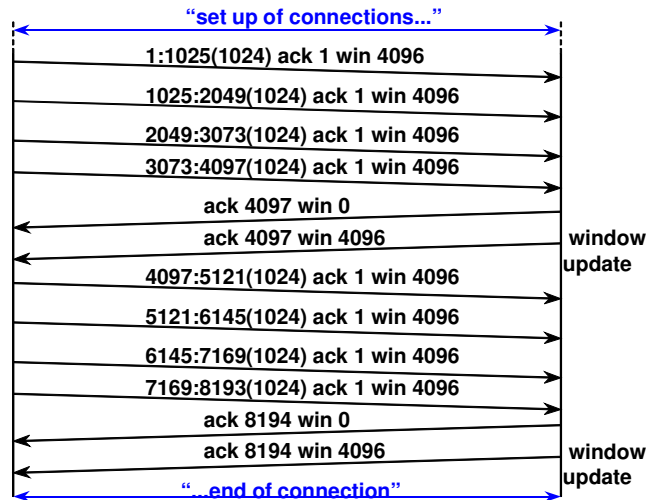
Esempio: trasmissione di un blocco di dati (2/2)



36

Fast sender, slow receiver

- Esempio di trasmissione di 8k bytes da un host sorgente veloce ad un host destinatario lento



37

Controllo di congestione

- Ha lo scopo di recuperare situazioni di sovraccarico nella rete limitando il traffico offerto alla stessa
- Difficoltà:
 - il protocollo IP (protocollo di rete) non possiede alcun meccanismo per rivelare e controllare la congestione
 - il TCP è un protocollo end-to-end e può rivelare e controllare la congestione solo in modo indiretto (senza collaborazione dei nodi di rete)
 - conseguenza: la conoscenza dello stato della rete da parte delle entità TCP è imperfetta
 - le entità TCP che usano la rete non cooperano tra loro, anzi competono per l'uso delle risorse distribuite

38

Controllo di congestione

- Osservazione: in caso di congestione, il controllo di errore e di flusso a finestra proteggono implicitamente anche la rete
 - se la rete è congestionata arriveranno meno riscontri e quindi saranno emessi un numero minore di segmenti (finestra di trasmissione)
 - il meccanismo adattativo di timeout evita ritrasmissioni che porterebbero ad un aumento della congestione invece che ad una sua diminuzione
- Il controllo di errore e di flusso riesce ad adattare il rate di emissione della sorgente a quello corrispondente al bottleneck della rete (proprietà di self-clocking)

39

Controllo di congestione (cont.)

- I bottleneck in rete possono essere
 - causati dalla limitazione della banda nei collegamenti fisici
 - causati dalla congestione nei router
 - dovuti alla limitata capacità elaborativa del ricevitore
- Il controllo di congestione di TCP
 - non è in grado di distinguere il tipo di bottleneck di rete
 - non può stabilire il tipo di contromisura più adatta
- TCP utilizza il TO di ritrasmissione (funzione della stima di RTT) come indicatore di congestione
 - lo scadere del TO è considerato un sintomo di congestione
- Il controllo di congestione del TCP introduce una finestra di emissione (Congestion Window) di dimensione variabile (cwin)
 - tale finestra indica il numero massimo di byte che possono essere emessi senza riscontro in base al controllo di congestione
- Fissato il RTT, aumentando la dimensione della finestra aumenta il rate

40

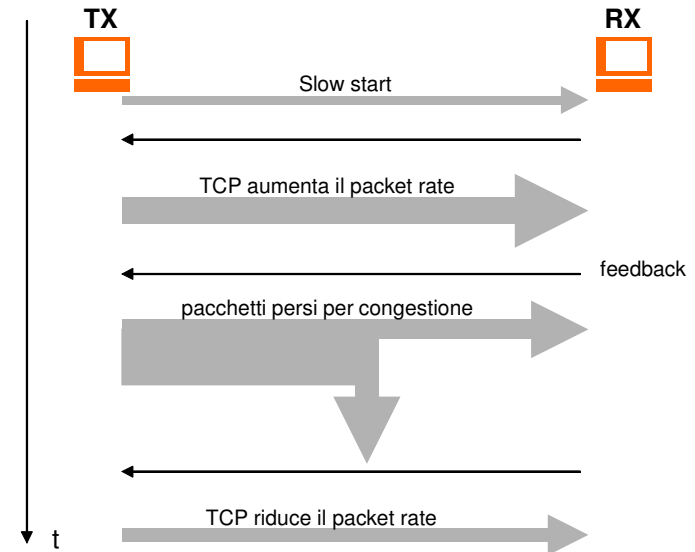
Controllo di congestione

- Sono definiti dei meccanismi aggiuntivi rispetto alla semplice stima del RTT
- Sono stati definiti vari meccanismi di adattamento della finestra di congestione
- Come risultato esistono varie implementazioni di TCP
 - Berkeley, Tahoe, Reno

Meccanismo	TCP Berkeley	TCP Tahoe	TCP Reno
RTT variance estimation	◆	◆	◆
Exponential RTO Backoff	◆	◆	◆
Karn Algorithm	◆	◆	◆
Slow Start	◆	◆	◆
Congestion Avoidance	◆	◆	◆
Fast retransmit		◆	◆
Fast recovery			◆

41

TCP: comportamento in caso di congestione



42

Slow Start

- Regola l'emissione dei segmenti ad ogni ripartenza (all'inizio della connessione e dopo ogni situazione di congestione)
- Ha lo scopo aumentare il ritmo di emissione sino al raggiungimento del limite di congestione
 - provocando così una nuova ripartenza
- Si considera una Congestion Window (*cwin*) che tende ad aumentare progressivamente
 - *cwin* viene misurata in segmenti (o più precisamente varia con multipli di MSS)
 - *cwin* viene incrementata di un segmento per ogni segmento trasmesso e confermato da un ACK
 - $cwin = \min(MAXwin, cwin + 1MSS)$
 - MAXwin è il massimo valore della Congestion Window

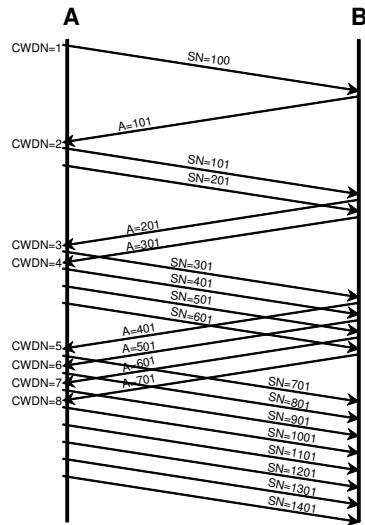
43

Slow Start (cont.)

- La crescita risulta di tipo esponenziale
 - all'inizio $cwin=1$
 - può essere inviato 1 segmento
 - $t=1$ RTT, viene ricevuto 1 ACK, $cwin=2$
 - possono essere inviati sino a 2 segmenti
 - $t=2$ RTT, vengono ricevuti 2 ACK, $cwin=4$
 - possono essere inviati sino a 4 segmenti
 - $t=n$ RTT, vengono ricevuti $N=2^{(n-1)}$ ACK, $cwin=cwin+N=2^n$
 - si possono inviare sino a 2^n segmenti
- In caso di ritrasmissione (segmento perso e timeout) si riparte da
 - $cwin=1$

44

Slow Start (cont.)



- L'ampiezza della finestra (twin) di trasmissione in segmenti è
 - $twdn = \min(fwin, cwin)$
- **fwin:**
 - numero di crediti concessi dalla flow window nell'ultimo ACK (in segmenti)
- **cwin:**
 - numero di crediti concessi dalla congestion window
- **per il primo segmento**
 - $cwin=1$
- **per ogni segmento riscontrato**
 - $cwin = \min(MAXwin, cwin + MSS)$

45

Congestion Avoidance

- In caso di ritrasmissione (congestione) si riparte da $cwin=1$ con la procedura "Slow Start"
- Una procedura di incremento come la "Slow Start" è molto aggressiva
- La procedura di Congestion Avoidance riduce la crescita della $cwin$ dopo una ripartenza
- Riducendo la crescita della $cwin$ si riduce il flusso in modo da
 - **consentire l'esaurimento della congestione**
 - **evitare successivamente un nuovo sovraccarico**
- Si procede in Slow start sino al raggiungimento di una soglia limite ($ssthresh$), superata tale soglia la $cwin$ cresce linearmente

46

Congestion Avoidance (cont.)

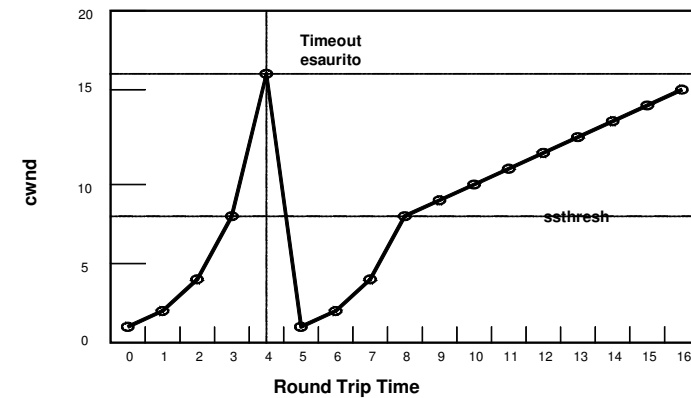
- La procedura è la seguente
 - in caso di congestione si fissa una soglia di slow start ($ssthresh$) uguale alla metà del valore corrente della Congestion Window

$$ssthresh = \frac{cwin}{2}$$
 - si esegue la procedura slow start fino a che

$$cwin = ssthresh$$
 - quando $cwin \geq ssthresh$, $cwin$ è incrementato di $1MSS$ ogni RTT (non più ogni segmento riscontrato)

47

Congestion Avoidance (cont.)



48

Fast Retransmit

- Migliora le prestazioni se è perso un singolo segmento
 - **velocizza la ritrasmissione del segmento**
 - **può evitare la ritrasmissione dei segmenti successivi**
- La procedura di Fast Retransmit si basa sulle seguenti assunzioni:
 - **il ricevitore emette un ACK non appena rivela un fuori sequenza ed emette un ACK per ogni segmento successivo fuori sequenza (in pratica viene emesso un ACK ogni volta che viene ricevuto un segmento; l'ACK fa riferimento all'ultimo segmento ricevuto in sequenza)**
 - **la ricezione di tre ACK duplicati è sintomo di un segmento perso**
 - **la scelta di tre ACK tende ad escludere il caso in cui il segmento sia effettivamente ritardato in rete e ricevuto successivamente fuori sequenza (e non perso)**
- La ritrasmissione del segmento inizia non appena sono ricevuti quattro ACK di richiesta trasmissione (confermando tutti i segmenti precedenti) anche se il timeout non è scaduto

49

Fast Recovery

- Evita l'insacco della procedura standard di Congestion Avoidance associata alla procedura di Fast Retransmission
 - **l'arrivo di ACK multipli assicura che i segmenti ricevuti sono stati ricevuti e quindi la eventuale congestione è già stata superata**
- Rispetto alla procedura di Congestion Avoidance
 - **il valore iniziale di *cwin* è maggiore e fissato a *sstresh***
 - **l'incremento di *cwin* è sempre lineare**
 - **si evita la fase iniziale di aumento esponenziale di *cwin* (Slow start)**

50

Fast Recovery

- La procedura è la seguente
 - **quando sono stati ricevuti tre ACK duplicati**
 - si pone

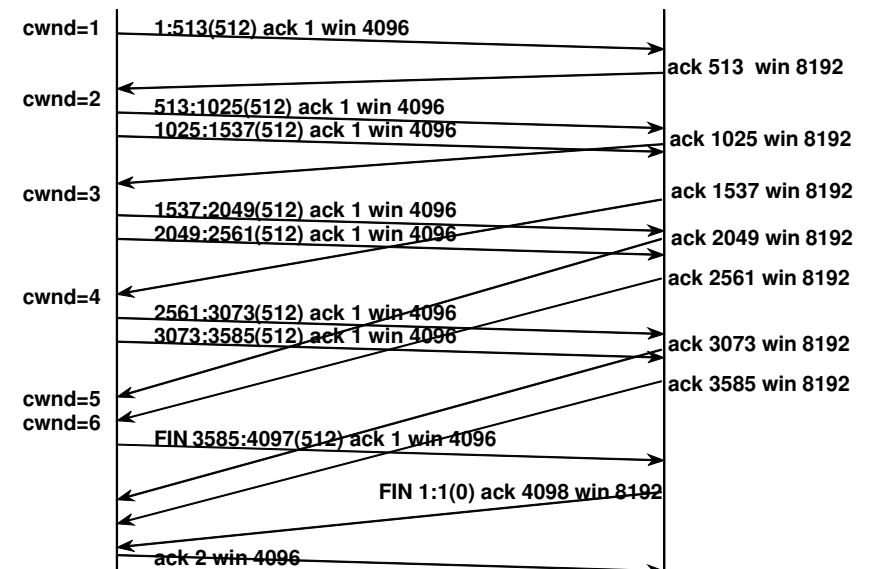
$$sstresh = cwin/2$$
 - viene ritrasmesso il segmento perduto
 - per tener conto dei segmenti già ricevuti si pone

$$cwin = sstresh + 3MSS$$
 - **ogni volta che arriva un ACK duplicato, il valore di *cwin* viene incrementato di 1MSS e trasmesso (se possibile) un segmento**
 - **quando viene ricevuto un ACK (riscontro cumulativo)**
 - si pone

$$cwin = ssthresh$$

51

Esempio: Slow start



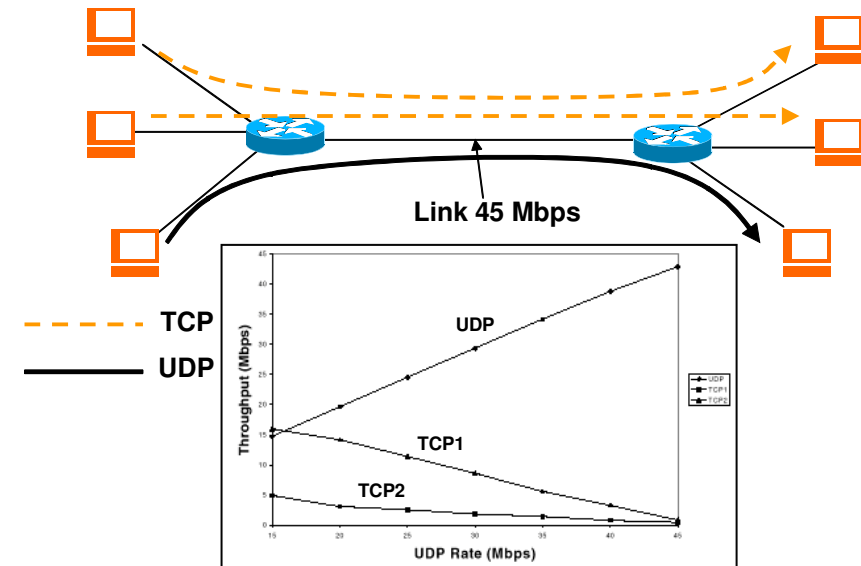
52

Confronto fra TCP ed UDP

- TCP è affidabile (Connection Oriented), mentre UDP non lo è (Connectionless)
- TCP consente esclusivamente connessioni punto-punto, mentre UDP supporta broadcast e multicast
- TCP è stream-oriented, UDP message-oriented
- TCP recupera eventuali errori trasmissivi ed effettua controllo di flusso e di congestione, UDP no
- TCP introduce maggior overhead rispetto ad UDP

53

TCP vs. UDP sulla stessa rete: Esempio



54

Network Address Translation (NAT)

- RFC 2663 - "NAT Terminology and Considerations"
 - Network Address Translation is a method by which IP addresses are mapped from one address realm to another, providing transparent routing to end hosts
 - There are many variations of address translation that lend themselves to different applications

Network Address Translation

56

Quando si usa il NAT

- Un nodo NAT si usa quando si vuole:
 - **intradare verso la rete Internet del traffico proveniente da una rete interna (intranet) in cui si fa uso di indirizzamento privato**
 - lo scambio dei pacchetti deve essere bidirezionale
 - **semplificare lo spostamento (migrazione) di una rete (e.g. aziendale) da un un punto di accesso alla rete Internet ad un altro**
 - ad esempio a causa di cambio ISP
 - si vuole evitare la riconfigurazione di tutti i nodi (host e router) interni alla rete
 - **semplificare le tabelle di routing (avoidance of routing table explosion)**
 - isolando una porzione di rete dietro ad un router NAT e rendendo invisibile la sua struttura interna all'esterno
 - **nascondere l'indirizzamento utilizzato all'interno della propria rete**
 - **proteggere i nodi interni ad una rete da attacchi provenienti dall'esterno**

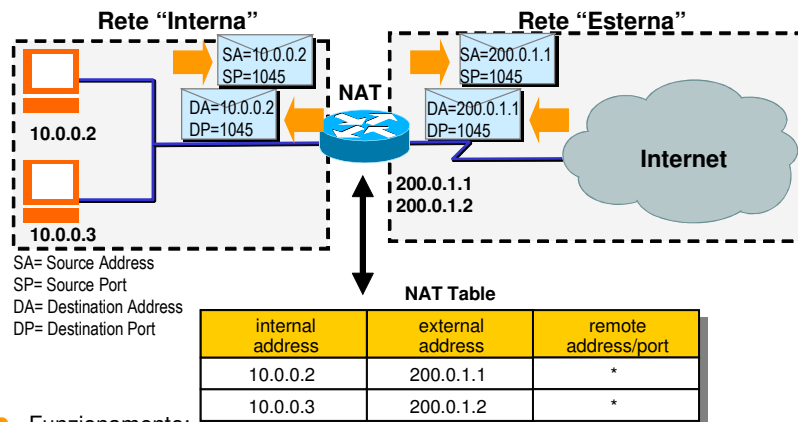
57

Tipologie di NAT: Simple NAT

- Mappaggio del tipo uno-a-uno (n indirizzi ↔ n indirizzi)
 - **ciascun indirizzo IP interno ad una rete viene tradotto in un indirizzo IP esterno distinto, appartenente ad un pool di indirizzi predefinito**
 - alcune volte riferito col termine "simple NAT"
 - NAT statico: la corrispondenza indirizzo interno e esterno è permanente, ovvero ogni indirizzo viene mappato sempre nel medesimo indirizzo esterno
 - funziona in ambedue i versi, nel senso che i pacchetti possono iniziare a transitare sia da dentro a fuori che viceversa
 - NAT dinamico: la corrispondenza tra indirizzo interno e esterno è variabile, ovvero ogni indirizzo viene mappato in un indirizzo casuale prelevato d un insieme di indirizzi disponibili
 - funziona solo da dentro a fuori, nel senso che un nodo interno deve inviare un pacchetto prima di poterne ricevere dall'esterno

58

Simple NAT



- Funzionamento:
 - viene tradotto il SA dei pacchetti uscenti (da sinistra a destra) da indirizzo interno a indirizzo esterno
 - viene tradotto il DA nei pacchetti entranti da indirizzo esterno a interno (mappaggio inverso)
 - tale mappaggio è mantenuto da una apposita tabella (NAT table)

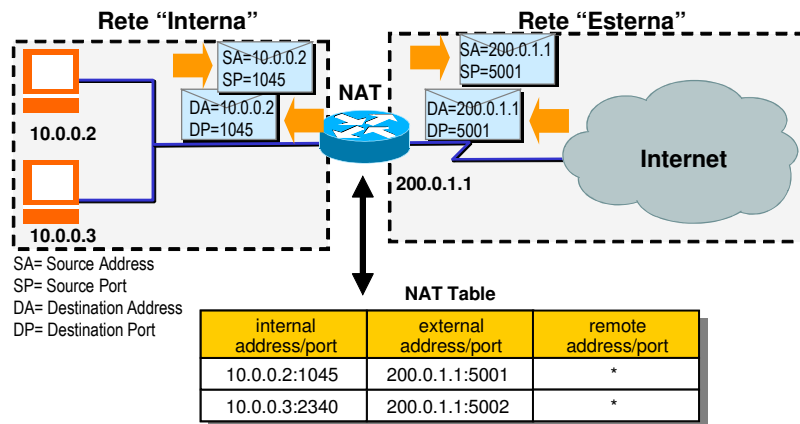
59

Tipologie di NAT: NATP

- Mappaggio del tipo n↔1 (n indirizzi ↔ 1 indirizzo)
 - **tutti gli indirizzo IP interni ad una rete vengono tradotti in un medesimo indirizzo IP esterno**
 - per permettere il mappaggio inverso si deve far ricorso all'utilizzo delle informazioni di numero porta che vengono opportunamente modificate
 - più propriamente riferito col termine NATP (Network Address and Port Translator)
 - funziona solo da dentro a fuori, nel senso che un nodo interno deve inviare un pacchetto prima di poterne ricevere dall'esterno
 - è il tipo di NAT più utilizzato
- In entrambi i casi gli indirizzi "interni" hanno valore solo dietro al NAT e possono essere quindi di tipo privato (RFC 1918)
 - 10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16

60

NAPT



- Funzionamento:
 - tutti gli host interni utilizzano un singolo indirizzo IP pubblico esterno
 - vengono utilizzate le porte TCP/UDP per individuare il reale destinatario del pacchetto
 - nei pacchetti uscenti vengono modificate anche le S_port
 - nei pacchetti entranti vengono rimessi a posto sia il D_addr che il D_port

61

Differenti tipi di NAPT

- Il NAT (e NAPT) non è un protocollo poiché non richiede un accordo tra due o più parti
 - **NAT opera il mappaggio nei due versi in modo trasparente agli altri nodi intermedi e terminali della comunicazione**
 - **ogni implementazione può utilizzare specifici meccanismi di mappaggio**
 - tipo di tabelle
 - informazione utilizzata per il mappaggio (e.g. solo gli indirizzi da mappare o anche l'indirizzo del host remoto)
 - supporto di eventuali protocolli applicativi che richiedono trattamento a parte (tramite appositi ALG)
- In letteratura differenti comportamenti NAPT vengono classificati:
 - **Full cone NAT**
 - **Restricted cone NAT**
 - **Port restricted cone NAT**
 - **Symmetric NAT**

62

Svantaggi del NAT/NAPT

- Complicato l'impiego di server all'interno della rete accessibili dall'esterno
 - in questi casi si ricorre al "static natting" o "destination NAT" (DNAT)
 - vengono configurati (staticamente) mappaggi tra porte esterne e coppie indirizzo/porta interne
 - il mappaggio viene fatto a partire da pacchetti entranti e in base alla porta di destinazione (destination NAT)
- Complicato il funzionamento di alcuni applicativi che non mantengono una relazione fissa client/server a livello di trasporto, e.g. FTP, H323, SIP
 - ad esempio quando gli indirizzi IP e porte dei nodi interni vengono inviati nel payload dei messaggi applicativi per poi essere usati per instaurare comunicazioni nel verso opposto
 - In questi casi, sono necessari degli Application Level Gateway (ALG) implementati nel nodo NAPT che modificando il contenuto dei dati di livello applicativo dei pacchetti

63