UNIVERSITA' DEGLI STUDI DI PARMA
Dipartimento di Ingegneria dell'Informazione

Protocolli di transporto in Internet: TCP e UDP

Luca Veltri

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

Corso di Reti di Telecomunicazioni A, a.a. 2005/2006

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

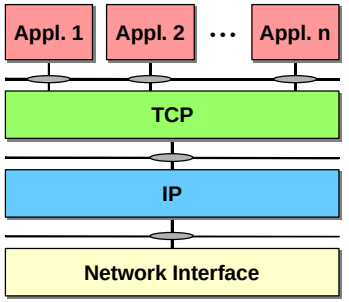
Sommario

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

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)**
- Un ulteriore protocollo di trasporto atualmente in fase di stadardizzazione in ambito IETF è:
  - **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

Indirizzamento TCP/UDP

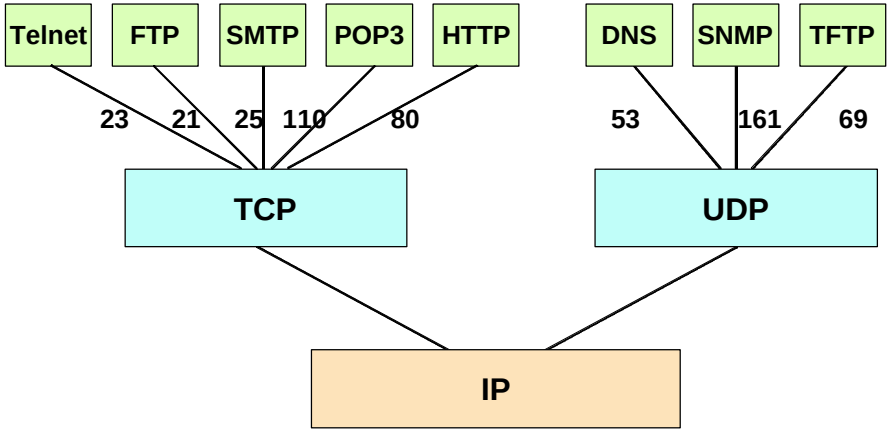


- Port
  - **identificativo di un utente TCP (o UDP), ovvero il processo applicativo**
- Socket
  - **indirizzo completo in TCP/IP (T-SAP)**
  - **(IP address , prtocol, port)**

L'assegnazione del numero di porta può essere
statico
sono identificativi staticamente associati ad applicazioni largamente utilizzate
sono spesso utilizzati identificativi inferiori a 256
dinamico
sono identificativi assegnati direttamente dal sistema operativo al momento dell'apertura della connessione (in genere maggiore di 1024)

Well Known Ports

- Sono associate agli applicativi principali, esempio



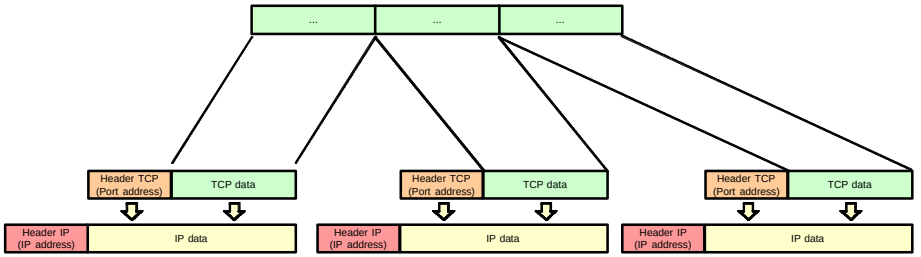
Transmission Control Protocol (TCP)

Transmission Control Protocol (TCP)

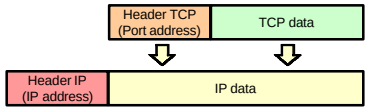
- Trasferisce un flusso informativo bi-direzionale non strutturato (byte stream) tra due host ed effettua operazioni di multiplazione e de-multiplazione
- E' un protocollo con connessione e offre un servizio stream oriente affidabile
- Funzioni eseguite
  - **controllo e recupero di errore**
  - **controllo di flusso**
  - **controllo di congestione**
  - **ri-ordinamento delle unità informative**
  - **indirizzamento di uno specifico utente all'interno di un host**

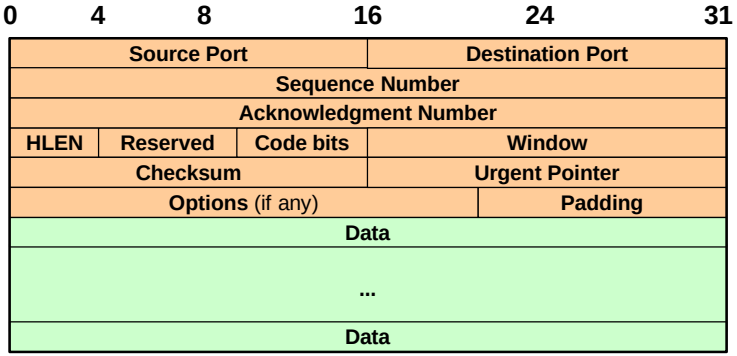
Unità di dati del TCP

- TCP è un protocollo "Stream oriented"
  - **Il TCP interpreta lo stream dati come sequenza di ottetti (bytes)**
  - **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 bytes**



Unità di dati del TCP





Unità di dati del TCP

- 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**

Unità di dati del TCP

- 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)**

Unità di dati del TCP

- Control bit (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 durante la fase di sincronizzazione fra le entità TCP**
  - **FIN: è uguale a uno quando la sorgente ha esaurito i dati da trasmettere.**

Unità di dati del TCP

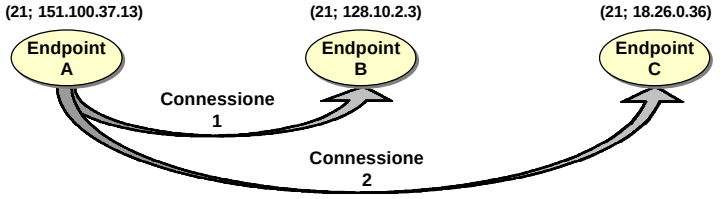
- Urgent Pointer (16 bit)
  - **contiene il numero di sequenza dell'ultimo byte dei dati che devono essere consegnati urgentemente al processo ricevente**
  - **tipicamente sono messaggi di controllo (out-of-band traffic)**
- Options (di lunghezza variabile)
  - **sono presenti solo raramente**
  - **Esempi:**
    - 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**

Connessioni TCP

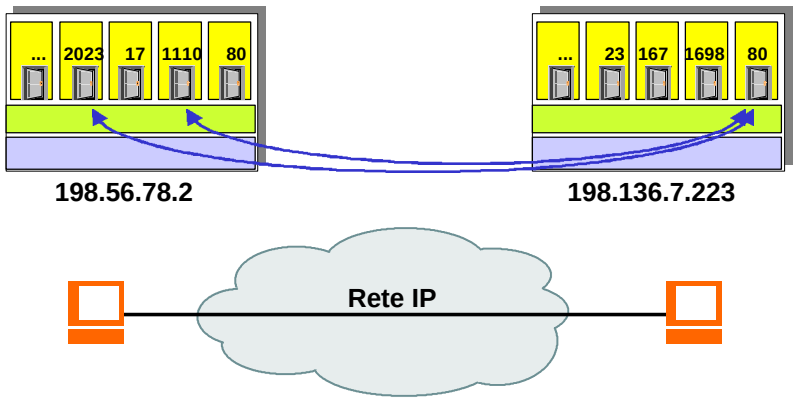
- TCP identifica un "canale" di comunicazione con il nome di *connessione*
- Sono identificate dalla quadrupla:

```
<IP client, Port client, IP server, Port server>
```

- 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**
  - **Viola il modello a layer (informazioni di livello 4 mischiate con informazioni di livello 3)**

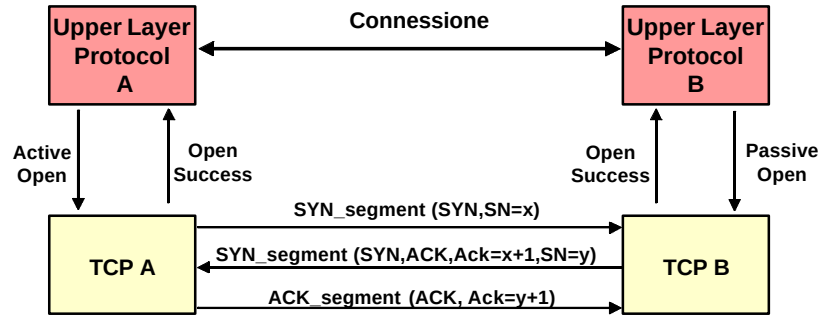


Connessioni TCP



Connessione TCP: apertura

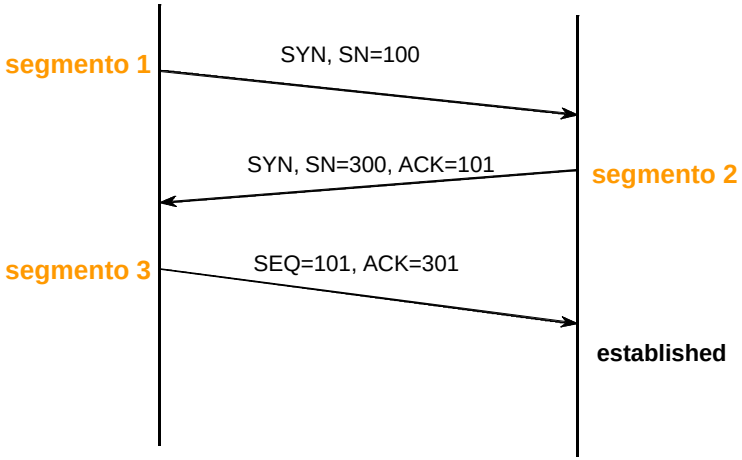
- Apertura = instaurazione della connessione (Setup)
- La sincronizzazione avviene con un meccanismo detto "three way handshaking"



lato client ("chiamante") lato server ("chiamato")

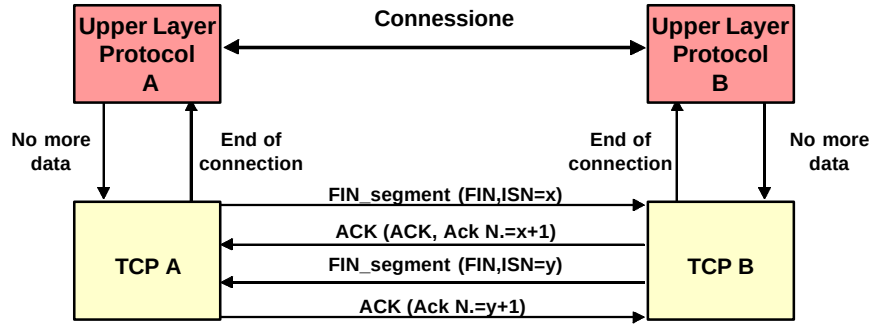
Connessione TCP: apertura

Three-way handshake

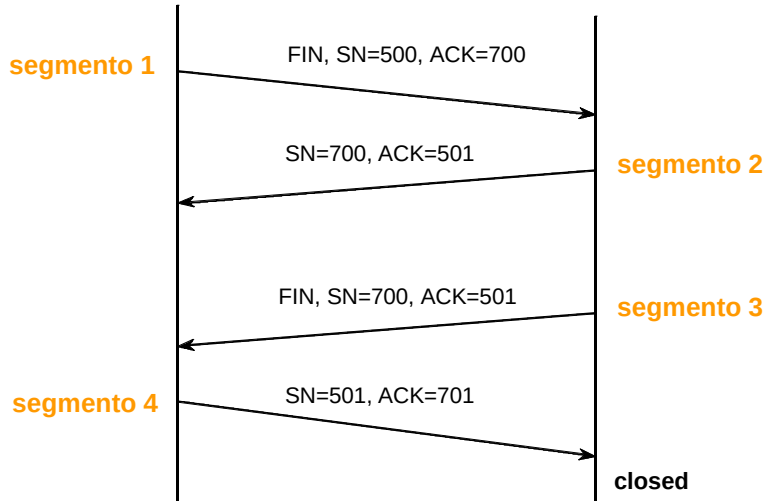


Connessione TCP: chiusura

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



Connessione TCP: Teardown



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
```

time *src* *dest* *Syn/Fin SN SN_next (size)* *ack* *flow-win*

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
```


TCP

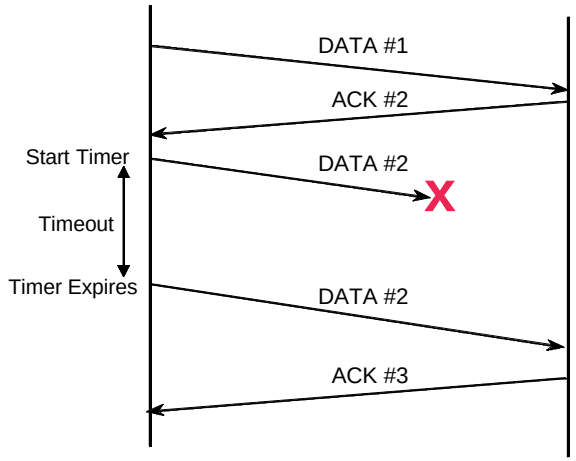
- Il TCP è un protocollo affidabile in quanto:
  - **Quando invia un segmento fa partire un timer**
    Se non viene ricevuto un riscontro prima della scadenza del timeout il segmento viene ritrasmesso
  - **Quando si riceve un segmento dati viene inviato un riscontro (ack)**
    Il riscontro viene ritardato 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**
  - **S attende che il trasmettitore vada in timeout e ritrasmetta il segmento**
  - **Poichè TCP è imbustato su IP, e poichè 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

Recupero di errore — Sequenzializzazione — Controllo di flusso

Controllo di errore

- Il controllo di errore è necessario per risolvere potenziali situazioni di perdita/errore
  - **IP non è affidabile, i datagrammi possono andare persi, essere ritardati, duplicati o consegnati fuori sequenza**
- Il TCP prevede esclusivamente riscontri positivi (Acknowledgement)
- La ritrasmissione è 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**

Controllo di errore



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 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

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

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

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

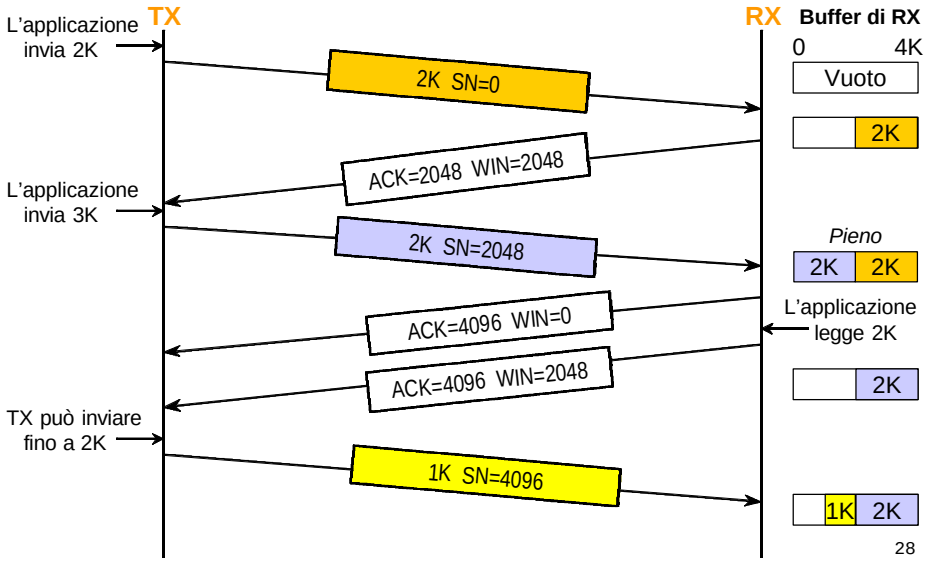
Controllo di Flusso e di Congestione

- Il controllo di flusso ha lo scopo di limitare il flusso dei dati, in base alle eseginze 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

Controllo di Flusso

- TCP utilizza un controllo di flusso a finestra basato su
  - **finestra scorrevole di ampiezza variabile**
- Il controllo di flusso opera a livello di ottetti (byte)
- Gli ottetti sono numerati sequenzialmente a partire dal numero scelto durante il 3-way handshaking
- I segmenti portano un ACK Number (utilizzato anche dal controllo di errore) e 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**

Controllo di flusso: esempio



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

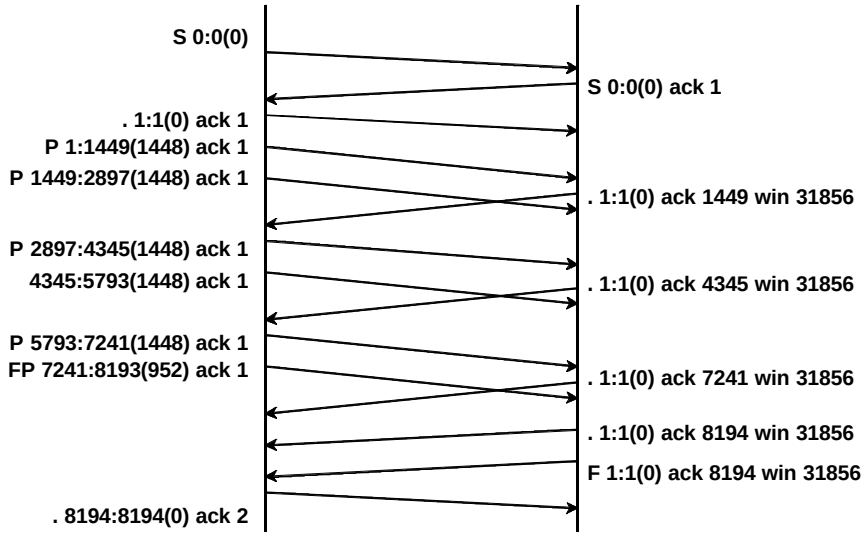
- esempio di trasmissione di 8k bytes con MSS 1460

```
[client]# ttcp -t -p 5555 -s -n1 server
```

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

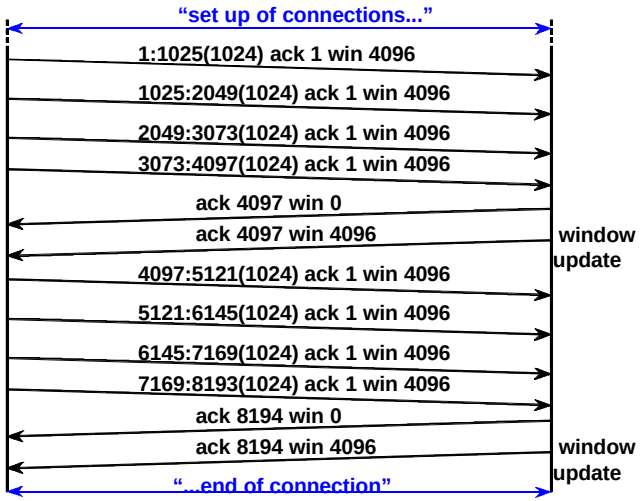
```
14.281143 client.1079 > server.5555: S 0:0(0) win 32120 <mss 1460>
14.281452 server.5555 > client.1079: S 0:0(0) ack 1 win 31856 <mss 1460>
14.281504 client.1079 > server.5555: . 1:1(0) ack 1 win 32120
14.282443 client.1079 > server.5555: P 1:1449(1448) ack 1 win 32120
14.282469 client.1079 > server.5555: 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
```


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



Fast sender, slow receiver

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



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 cooeperano tra loro, anzi competono per l'uso delle risorse distribuite**

Controllo di congestione

- 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**
  - **il meccanismo adattativo di timeout evita ri-trasmissioni che porterebbero ad un aumento della congestione invece che ad una sua diminuzione**
- Il controllo di errore e di flusso end-to-end riesce ad adattare il rate di emissione della sorgente a quello corrispondente al bottleneck della rete (proprietà di self-clocking)

Controllo di congestione

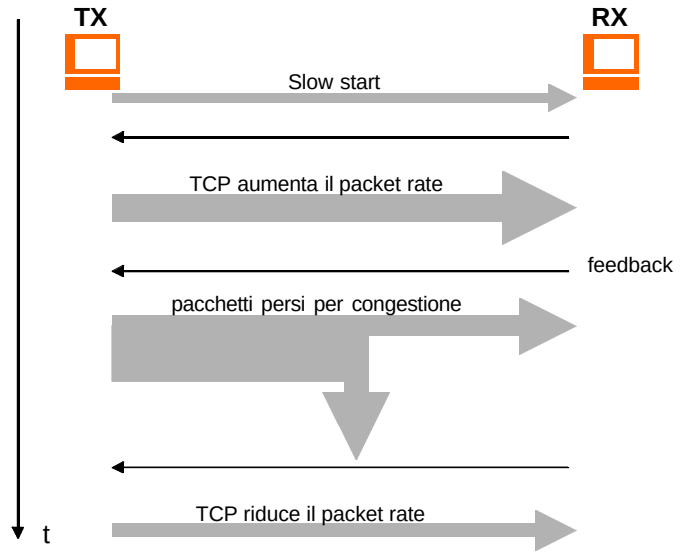
- I bottleneck in rete possono essere
  - **causati dalla limitazione della banda nei collegamenti fisici**
  - **causati dalla congestione nei router**
  - **dovuti alla limitata capacità elaborativa del ricevitore**
- Il controllo di congestione di TCP
  - **non è in grado di distinguere il tipo di bottleneck di rete**
  - **non può stabilire il tipo di contromisura più adatta**
- TCP utilizza la stima di RTT come misura di congestione, lo scadere del timeout di ritrasmissione è considerato un sintomo di congestione
- Il controllo di congestione si basa su una finestra di emissione di dimensione variabile (congestion window)
- Tale finestra indica il numero massimo di byte che possono essere emessi senza riscontro
- Fissato il RTT, aumentando la dimensione della finestra aumenta il rate

Controllo di congestione

- Sono definiti dei meccanismi addizionali rispetto alla semplice stima del RTT
- Esistono varie implementazioni di TCP
  - **Berkeley**
  - **Tahoe**
  - **Reno**

	Meccanismo	TCP Berkeley	TCP Tahoe	TCP Reno
Timer	RTT variance estimation	u	u	u
	Exponential RTO Backoff	u	u	u
	Karn Algorithm	u	u	u
Congest. window	Slow Start	u	u	u
	Congestion Avoidance	u	u	u
	Fast retransmit		u	u
	Fast recovery			u

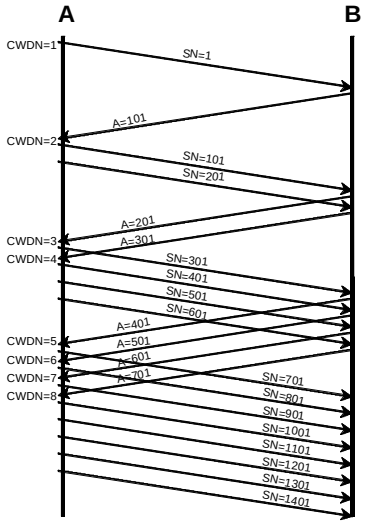
TCP: comportamento in caso di congestione



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 definisce una Congestion Window (cwnd) che tende ad aumentare progressivamente
- La cwnd viene misurata in segmenti
- La cwnd viene incrementata di un segmenti per ogni segmento trasmesso e confermato da un ACK
- La crescita risulta di tipo esponenziale
  - **cwnd=N ⇒ si possono inviare sino a N segmenti**
  - **quando vengono ricevuti tutti gli N Ack ⇒ nuova cwnd=2N**

Slow Start



- L'ampiezza della finestra (awnd) di trasmissione in segmenti è

  awdn = min[fwin , cwin]

  - **fwin:**
    - numero di crediti concessi dalla flow window nell'ultimo ACK (in segmenti)
  - **cwnd:**
    - congestion window (in segmenti)
  - **per il primo segmento**
    - cwnd=1
  - **per ogni segmento riscontrato**
    - cwnd=min[Mwnd, cwnd+1]
    - Mwnd è il massimo valore della congestion window

Congestion Avoidance

- Regola l'ampiezza della finestra in caso di congestione durante la connessione
- Il meccanismo è innescato in caso di ritrasmissione di un segmento
- Riducendo la cwdn si riduce il flusso in modo da
  - **consentire l'esaurimento della congestione**
  - **evitare successivamente un nuovo sovraccarico**
- Una procedura di incremento della cwnd come la "Slow start" è troppo aggressiva
- Si procede in Slow start sino al raggiungimento di una soglia limite,
  superata tale soglia la cwnd cresce linearmente

Congestion Avoidance

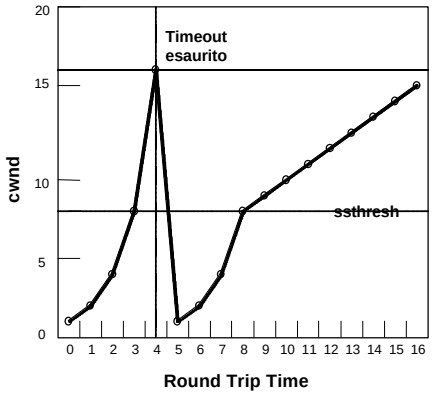
- Procedura
  - **in caso di congestione si fissata una soglia di slow start uguale alla metà del valore corrente della congestion window**

    $$\text{sstresh} = \frac{\text{cwnd}}{2}$$

  - **si esegue la procedura slow start fino a che**

    $$\text{cwnd} = \text{sstresh}$$

  - **se cwnd ≥ sstresh, cwnd è incrementato di uno ogni round trip delay (non più per ogni segmento riscontrato)**



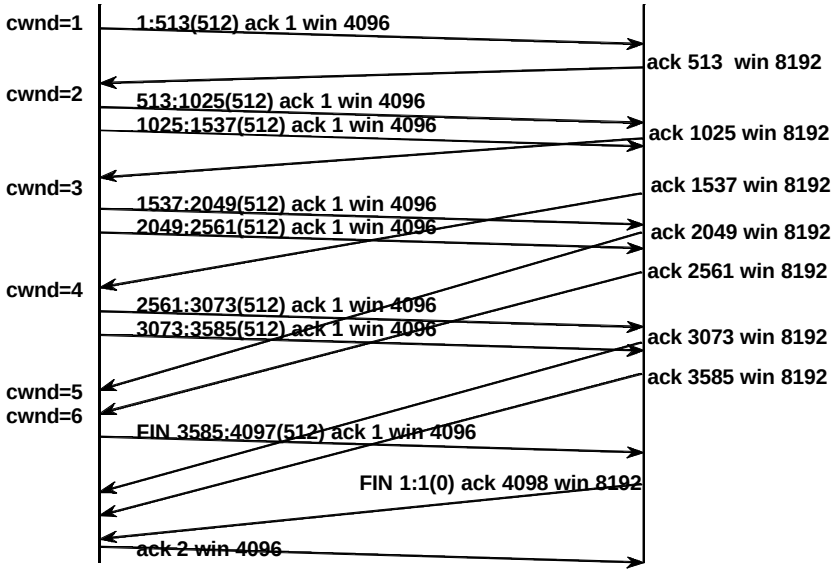
Fast Retransmit

- Migliora le prestazioni se è perso un singolo segmento
  - **velocizza la ritrasmissione del segmento**
  - **può evitare la ritrasmissione dei segmenti successivi**
- Assunzioni su cui la procedura si basa sono
  - **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

Fast Recovery

- Evita l'innesco 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 cwdn è maggiore e fissato a sstresh**
  - **l'incremento di cwdn è sempre lineare**
  - **si evita la fase iniziale di aumento esponenziale di cwdn (Slow start)**

Esempio: Slow start



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**
  - **Supporta funzionalità di multiplazione e demultiplazione attraverso l'introduzione delle porte (in modo analogo al TCP)**
  - **Può essere utilizzato per trasmissioni multicast (a differenza di TCP)**
    - stesso messaggio UDP a più destinazioni, indirizzate attraverso un indirizzo
      `< IP_multicast_addr, UDP port >`

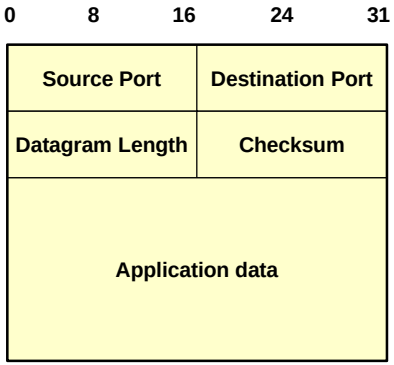
User Datagram Protocol (UDP)

- 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**
- Non prevede un controllo di flusso
- Non è in grado di adattarsi autonomamente a variazioni di traffico
- Non prevede meccanismi di ritrasmissione in caso di errori/perdite
  - **eventuali meccanismi di ritrasmissione (se necessari) vengono gestiti direttamente dall'applicazione**

User Datagram Protocol (UDP)

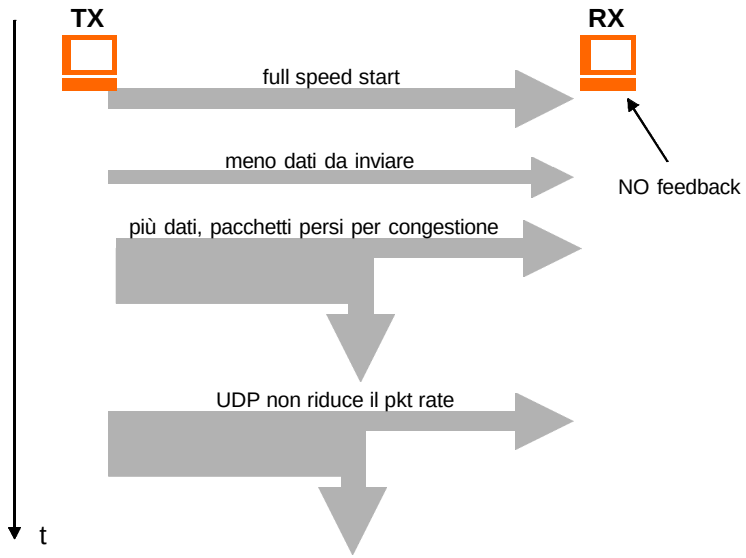
- E' utilizzato per il supporto di transazioni semplici tra applicativi, esempio
  - **risoluzione di indirizzi (DNS)**
  - **messaggi di management (e.g. SNMP)**
  - **supporto di File System distribuiti (e.g. NFS)**
- Particolarmente adatto per applicazioni Real-Time:
  - **assenza di ritrasmissione**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

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**
- Checksum (16 bit)
  - **protegge tutto il datagramma UDP (header + dati)**

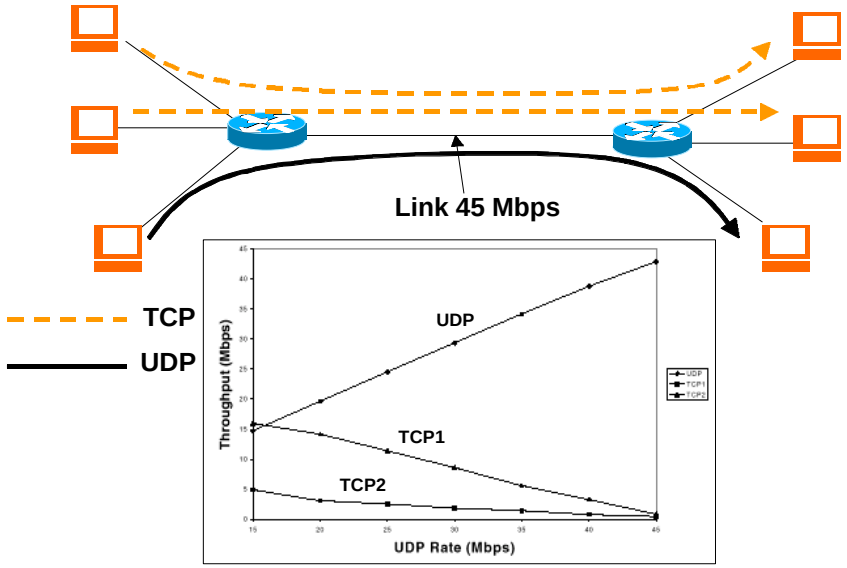
UDP: comportamento in caso di congestione



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

TCP vs. UDP sulla stessa rete: Esempio



Network Address Translation

Indirizzi IP privati

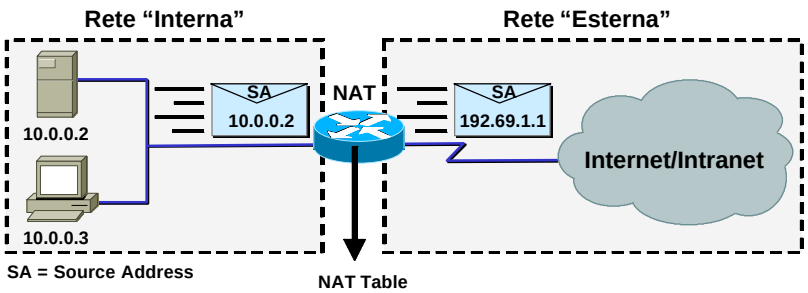
IANA-Allocated, Non-Internet Routable, IP Address Schemes

Class	Network Address Range
A	10.0.0.0-10.255.255.255 (10.0.0.0/8)
B	172.16.0.0-172.31.255.255 (172.16.0.0/12)
C	192.168.0.0-192.168.255.255 (192.168.0.0/16)

NAT: tipi di traduzione

- Network Address Translation (NAT)
  - **Traduce solo gli indirizzi**
  - **Traduzione uno-a-uno**
  - **Funzione in ambedue i versi (interno<->esterno)**
- Network Address Port Translation (NAPT)
  - **Traduce le coppie Indirizzo/Port**
  - **Traduzione N-a-uno**
  - **Riduce il consumo di indirizzi IP registrati pubblicamente**
  - **Funziona in un solo verso**
    - l'host che inizia una comunicazione pto-pto deve essere l'host interno

Simple NAT

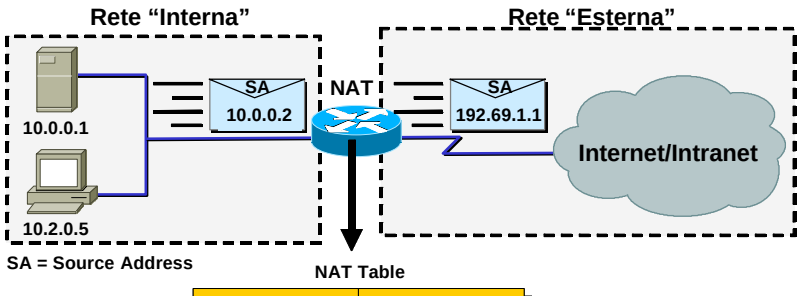


NAT Table

Inside Local IP Address	Inside Global IP Address
10.0.0.2	192.69.1.1
10.0.0.3	192.69.1.2

- viene tradotto il SA dei pacchetti uscenti (da sinistra a destra) da indirizzo privato a indirizzo pubblico
- viene tradotto il DA nei pacchetti entranti da indirizzo pubblico a privato (mappaggio inverso)

Network Address Port Translation (NAPT)

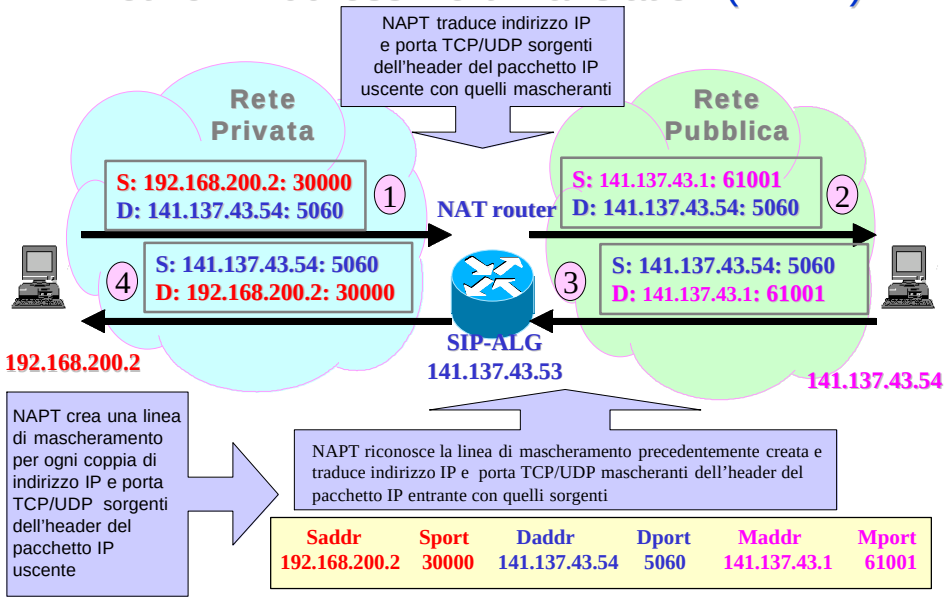


NAT Table

Inside Local Address/port	Inside Global Address/port
10.0.0.2:1045	192.69.1.1:50001
10.0.0.3:2340	192.69.1.1:5002

- Tutti gli host interni utilizzano un singolo indirizzo IP pubblico esterno
- Vengono utlizzate 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

Network Address Port Translation (NAPT)



NAT/NAPT Motivazioni

- Risparmio di indirizzi IP pubblici
  - **accedere alla Internet pubblica senza richiedere nuovi indirizzi IP pubblici, impiegando spazi di indirizzamento privato (RFC 1918)**
    - interconnettere reti IP con spazi di indirizzamento sovrapposti (indirizzi privati)
    - riduzione del consumo di indirizzi pubblici
  - **attraverso meccanismi tipo Port Address Translation (NAPT)**
- Cambiamento di ISP
  - **evitare la rinumerazione degli host**
  - **sia attraverso NAT che NAPT**
- Sicurezza
  - **migliorare la sicurezza della rete "nascondendo" gli indirizzi reali degli host**

NAT/NAPT Vantaggi e Svantaggi

- Vantaggi:
  - Applicativi lavorano trasparentemente (no cambiamenti di hosts e routers)
  - Solo un unico indirizzo IP pubblico richiesto per la rete
  - Aumento sicurezza rete (NAPT come caratteristica di un Firewall di rete)
- Svantaggi:
  - Mancato funzionamento di alcuni applicativi di rete, e.g. FTP, H323(Netmeeting)..
    - e.g. indirizzi IP e porte TCP/UDP privati inviati nel payload del messaggio TCP (non gestito dal NAPT)
    - se tale indirizzo viene usato dal host esterno, quest'ultimo non riuscirebbe ad istradare correttamente i pacchetti
  - In questi casi, sono necessari dei Application Level Gateway (ALG), ovvero funzioni implementate nel nodo NAPT che elaborano i pacchetti analizzando e modificando il contenuto dati di livello applicativo