



UNIVERSITÀ
DEGLI STUDI DI BARI
ALDO MORO

*CDS IN INFORMATICA E
COMUNICAZIONE DIGITALE*

Anno Accademico 2015-2016

Corso di

“Reti di Calcolatori e Comunicazione Digitale”

Modulo 5 TCP/IP : i protocolli a livello 3 e 4

*Prof. Sebastiano Pizzutilo
Dipartimento di Informatica*

Differenti versioni del protocollo IP



Caratteristiche di un nuovo protocollo IP per sostituire IPv4

- ✓ **Indirizzamento illimitato**
- ✓ **Sicurezza** (*meno NAT e obbligo di IPsec*)
- ✓ **Supporto per pacchetti di grosse dimensioni**
- ✓ **Gestione del tipo di servizio**
- ✓ **Supporto per i protocolli di livello superiore che si appoggiano ad IPv4**

IPv6 (*o IPng: next generation*)

nuova forma di indirizzamento che prevede l'uso di indirizzi a 16 byte = 128 bit = 8 sezioni esadecimali di 2 byte separate da ":"

Ad esempio :

A0D3 : FB43 : 4EFB : AA41 : FF88 : 41AD : 0000 : FD23

Indirizzi IPv6

Ottimizzazioni del nuovo formato:

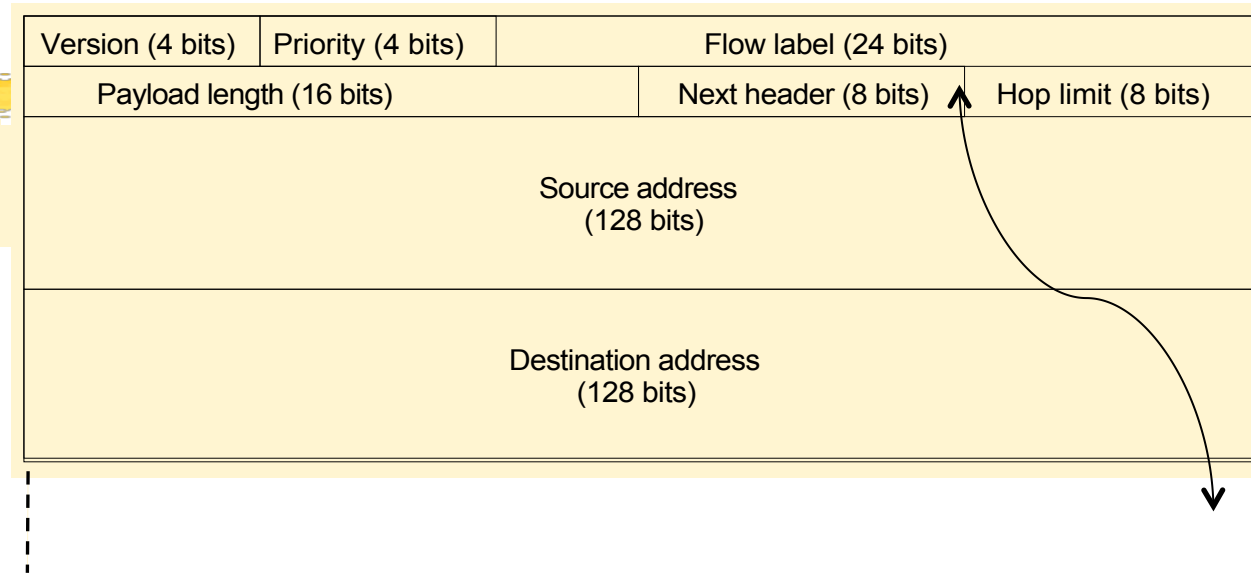
- si possono **omettere gli zeri ad inizio di un gruppo** (es. `:0123:` diventa `:123:`)
- si possono **omettere gruppi di zeri consecutivi**, con una sequenza “`::`”
(es. `2030::123:4007:F9AB:C0EF`)

Gli indirizzi IPv4 sono rappresentati con 6 gruppi di zeri, e due gruppi che rappresentano l'indirizzo IPv4 (rappresentabili anche in notazione decimale) (*ad es. `::89AB:CDEF` oppure `::137.171.205.239`*)

- Notazione CIDR per definire la parte dell'indirizzo dedicata alla rete (come in IPv4)
- In IPv6 la “rete” è identificata dall'indirizzo con tutti “0” nel campo di indirizzo dell'host

Pacchetto IPv6

IPv6
da 40 a 65535
bytes



Il **campo dati** può opzionalmente contenere **altri header** prima dei dati;

Il **campo version** assume il valore 6

Il **campo traffic class** identifica i pacchetti che necessitano di un instradamento particolare (traffico prioritario o il traffico di tipo voce o video stream).

Il **campo flow label** identifica i pacchetti appartenenti allo stesso flusso trasmissivo (traffico voce/video).

Il **campo payload length** indica la lunghezza del pacchetto in byte, esclusi i 40 byte fissi dell'header.

Il **campo next header** indica il protocollo del livello di trasporto a cui sono destinati i dati (TCP o UDP) oppure il tipo di *intestazione estesa successiva utilizzata*.

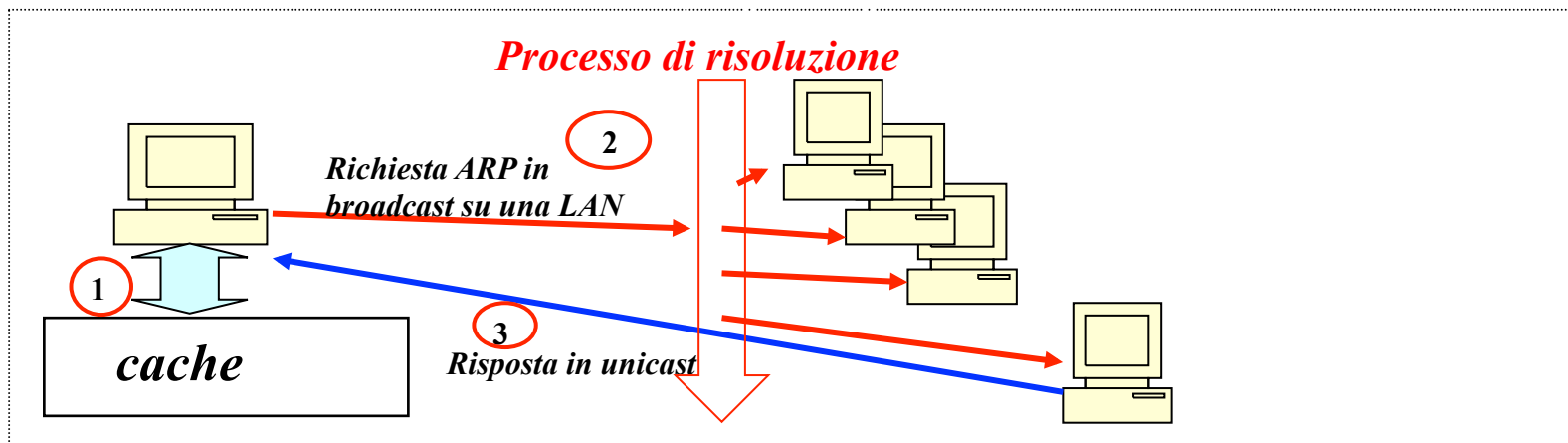
Il **campo hop limit** è il conto degli hop che viene decrementato ad ogni salto di router.

Gli **ultimi campi prima dei dati** sono gli indirizzi (di 128 bit) **sorgente e destinatario** del pacchetto

TCP/IP : Livello 3 di rete

ARP (Address Resolution Protocol)

Il protocollo **ARP** si occupa di tradurre un **indirizzo IP** (*indirizzo logico*) del destinatario in **indirizzo MAC** (*indirizzo fisico*) per poter costruire un **frame** a livello 2 in cui incapsulare un datagram IP inviato all'indirizzo fisico del destinatario.



1. Il **MAC address** del **destinatario** viene cercato nella cache locale.
2. Se il **MAC** non è nella cache, si cerca il destinatario con un messaggio ARP con l'**indirizzo IP** del **destinatario** in **broadcast** sulla rete.
3. Il **destinatario** riceve la richiesta ARP con il **proprio indirizzo IP** e restituisce **in unicast** al mittente (del quale conosce l'IP ed il MAC) il proprio indirizzo MAC.
4. Il **mittente memorizza nella propria cache una tabella ARP** con le associazioni IP-MAC che conosce
5. Se l'indirizzo IP del destinatario è su un'altra rete, il processo si ripete considerando mittente e ricevente i router (o i **proxy ARP**) delle rispettive reti.

TCP/IP : Livello 3 di rete

ARP per conversione *da ind.IP a ind.MAC*

Con il comando `arp` è possibile visualizzare il contenuto della cache.

```
Net to Media Table
Device      IP Address          Mask      Flags      Phys Addr
-----
hme0      sv_nt_00           255.255.255.255      00:60:08:71:de:49
hme0      www.xxxxxx.com     255.255.255.255      00:60:08:73:2f:cf
hme0      192.9.200.30       255.255.255.255      00:60:08:54:c5:ed
hme0      192.9.200.42       255.255.255.255      00:50:ba:05:fe:dd
hme0      192.9.200.98       255.255.255.255      00:10:4b:af:96:75
hme0      venus              255.255.255.255      SP      08:00:20:9a:02:64
hme0      jupiter           255.255.255.255      08:00:20:9a:08:21
hme0      224.0.0.0          240.0.0.0            SM      01:00:5e:00:00:00
```

TCP/IP : Livello 3 di rete

RARP (Reverse ARP)

RARP effettua la conversione di un **ind. MAC** in **ind. logico IP** (quando un host conosce solo il proprio indirizzo fisico (MAC), senza essere in grado di sapere quale sia il proprio indirizzo logico IP).

*Un nodo **non** conosce il proprio indirizzo IP :*

- a) quando è **senza dischi** e quindi al momento in cui viene acceso non può leggere su un file persistente l'indirizzo IP con cui è stato configurato sulla rete,*
- b) quando si è su una rete con indirizzi IP assegnati dinamicamente (**DHCP**).*

*Il nodo crea un messaggio di **richiesta RARP**, in cui inserisce il proprio indirizzo fisico e lo spedisce in broadcast allo **strato di collegamento** (tutti 1) **sulla rete locale**, fino a raggiungere **un server RARP** che conosce tutti gli indirizzi IP associati agli indirizzi fisici dei nodi connessi su quella rete.*

PROBLEMA : se si gestiscono più reti e sottoreti occorrono più server RARP uno per ogni rete.

TCP/IP : Livello 3 di rete

BOOTP e **DHCP** come supporto dal livello di applicazione

-BOOTP (*Bootstrap Protocol*), protocollo client/server **del livello applicativo** che opera impacchettando in **UDP** e in **IP** i messaggi del protocollo. *Utilizza un nodo **relay agent** come intermediario che conosce l'indirizzo IP del **server BOOTP** e invia in **unicast** la risposta al client.*
*Il **BOOTP** non consente una gestione dinamica degli indirizzi IP.*

- **DHCP** (*Dynamic Host Configuration Protocol*), è un protocollo **del livello applicativo** che consente la gestione sia di indirizzi IP statici (come BOOTP) che dinamici. *Utilizza un Data Base di indirizzi IP assegnati staticamente ed un Data Base di indirizzi assegnati dinamicamente all'interno di un range predeterminato.*

TCP/IP : Livello 3 di rete

ICMP (Internet Control Message Protocol)

ICMP (RFC 0792) fornisce il supporto a comandi per la diagnosi di problemi sulla rete.

E' un protocollo di servizio che si preoccupa di notificare o correggere errori a livello IP (trasmettere dati riguardanti malfunzionamenti, informazioni di controllo o messaggi tra i vari componenti di una rete).

Il protocollo ICMP può svolgere le seguenti funzioni:

- ✓ *fornire messaggi di **echo** e di risposta per verificare l'attendibilità di una connessione tra due host,*
- ✓ *reindirizzare il traffico per fornire un instradamento più efficiente nel caso di intasamento di un router,*
- ✓ *emettere un messaggio di **time-out** quando il TTL di un datagram viene superato,*
- ✓ *fornire un messaggio di inibizione dell'origine per dire ad un host di rallentare le proprie comunicazioni per non intasare un router.*

Header ICMP di $4 \times 8 = 32$ bit + dati



Tipo	codice	checksum
dati		

TCP/IP : Livello 3 di rete

ICMP

Tipi di messaggi

- ✓ Verifica lo stato della rete:
 - *Echo request*
 - *Echo reply*
- ✓ Riporta anomalie:
 - *Destination Unreachable*
 - *Time Exceeded for a Datagram*
 - *Parameter Problem on a Datagram*
- ✓ Scopre la netmask. Introdotto nelle ultime versioni:
 - *Mask Request*
 - *Address Mask Reply*
- ✓ Migliora il routing
 - *Redirect*

Valore	Tipo di Messaggio
0	Echo Reply
3	Destination Unreachable
4	Source Quence
5	Redirect
8	Echo Request
11	Time Exceeded for a Datagram
12	Parameter Problem on a Datagram
13	Timestamp Request
14	Timestamp Reply
15	Information Request
16	Information Reply
17	Address Mask Request
18	Address Mask Reply

TCP/IP : Livello 3 di rete

ICMP

Uno dei **comandi a livello applicativo** più comunemente usati che fanno uso di questo protocollo è il **PING (Packet INternet Groper)** :

Esecuzione di Ping venus.sysdata.it [192.9.200.199] :

Risposta da 192.9.200.199: byte=32 durata<10ms TTL=255

Risposta da 192.9.200.199: byte=32 durata<10ms TTL=255

Risposta da 192.9.200.199: byte=32 durata<10ms TTL=255

Risposta da 192.9.200.199: byte=32 durata<10ms TTL=255

Statistiche Ping per 192.9.200.199:

Pacchetti: Trasmessi = 4, Ricevuti = 4, Persi = 0 (0% persi),

Tempo approssimativo percorsi andata/ritorno in millisecondi: Minimo

Per verificare quale sia il percorso utilizzato dai pacchetti per raggiungere una destinazione, si utilizza il comando a livello applicativo **'traceroute'**, che determina il percorso per una destinazione inviando **messaggi ICMP di Echo Request** verso la destinazione ed incrementando di volta in volta il TTL.

traceroute portatile.plip.dg

traceroute to portatile.plip.dg (192.168.254.1), 30 hops max, 40 byte packets

1 dinkel.brot.dg (192.168.1.1) 0.433 ms 0.278 ms 0.216 ms

2 router.brot.dg (192.168.1.254) 2.335 ms 2.278 ms 3.216 ms

3 * * *

4 * * *

5 portatile.plip.dg (192.168.254.1) 10.654 ms 13.543 ms 11.344 ms

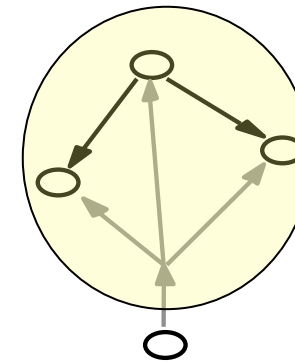
TCP/IP : Livello 3 di rete

IGMP (Internet Group Management Protocol)

L' **IGMP** permette la **gestione di gruppi di destinatari** per poter implementare a **livello applicativo** la **comunicazione multicast** (*indirizzi di classe D*).

IGMP opera fra un **host** ed il **router multicast** ad esso collegato direttamente.

*Per coordinare i **router multicast** invece è richiesto un altro protocollo, così che i datagrammi multicast possano essere instradati alle loro destinazioni finali.*

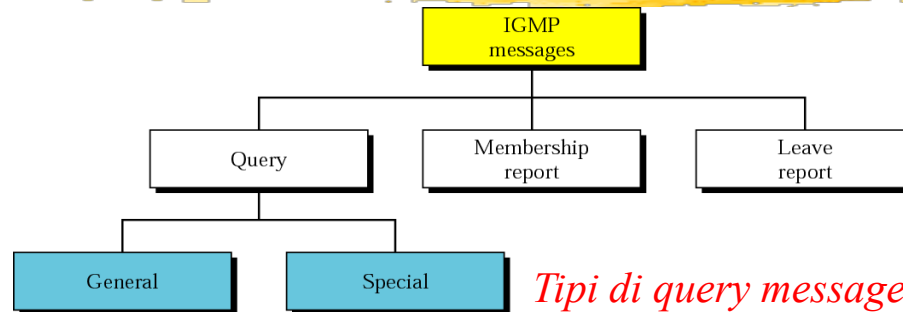


Gruppo aperto
in cui ciascun
processo del sistema
può mandare un
messaggio al gruppo

L'**Internet Group Management Protocol** permette ad un host di informare il **router** ad esso collegato che una applicazione in esecuzione su un host vuole connettersi ad uno specifico **gruppo di nodi**.

TCP/IP : Livello 3 di rete

IGMP



Tipi di messaggi dell'IGMP

Tipi di query message

Formato del messaggio IGMP 32 bit

tipo	T max di risposta	checksum
<i>Indirizzo classe D del gruppo</i>		

Tipi di messaggi

Membership query: generale : *inviato da un router*: richiede a tutti gli host a lui collegati i gruppi multicast cui gli host hanno aderito

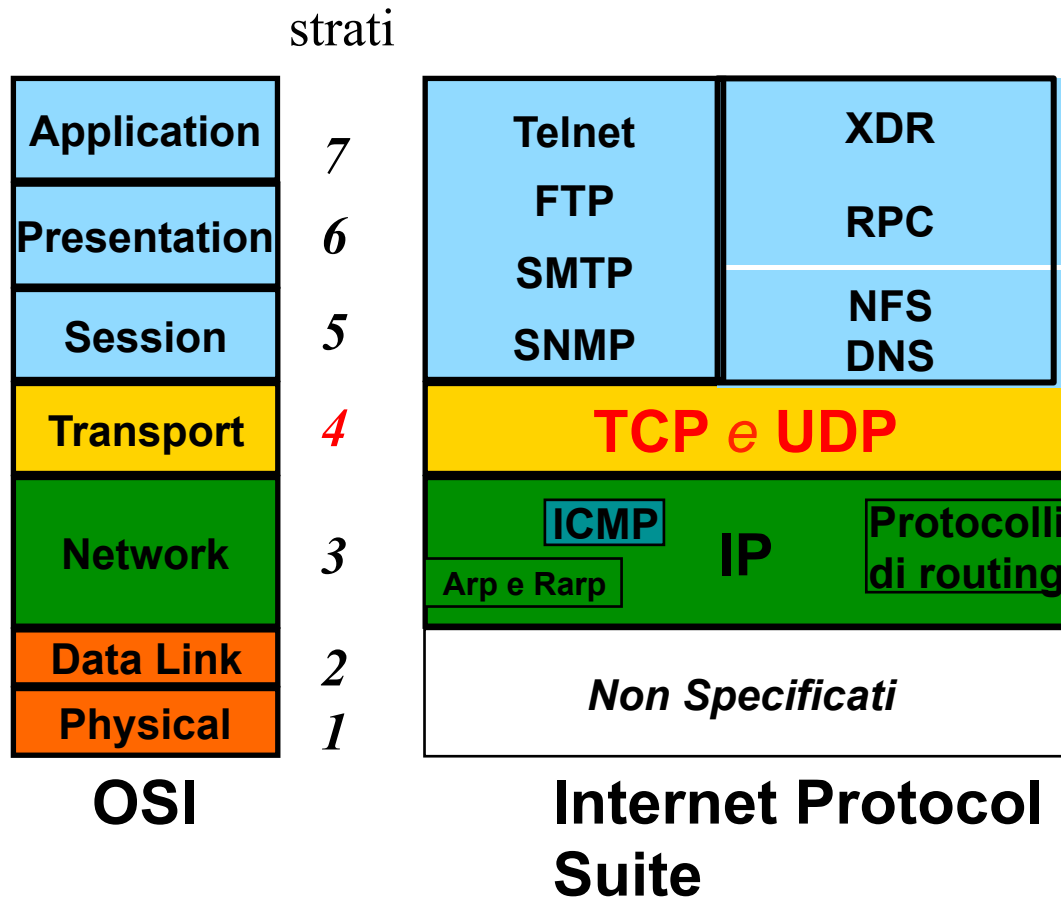
Membership query: specifica: *inviato da un router*: richiede se uno specifico gruppo si è unito agli host a cui è collegato

Membership report: *inviato da un host*: risposta di un host ad una membership query di un router per segnalare che si vuole unire o che appartiene a un dato gruppo.

Leave group: *inviato da un host*: registra l'abbandono di un certo gruppo

TCP/IP : Livello 4 di Trasporto

ISO-OSI e TCP/IP



Lo strato di **Trasporto** cura la connessione tra le applicazioni sull'host mittente e ricevente.

Lo strato 4 **segmenta** i dati prodotti dal mittente (che provengono dal livello di sessione) e trasmette i segmenti al livello inferiore 3 di rete.

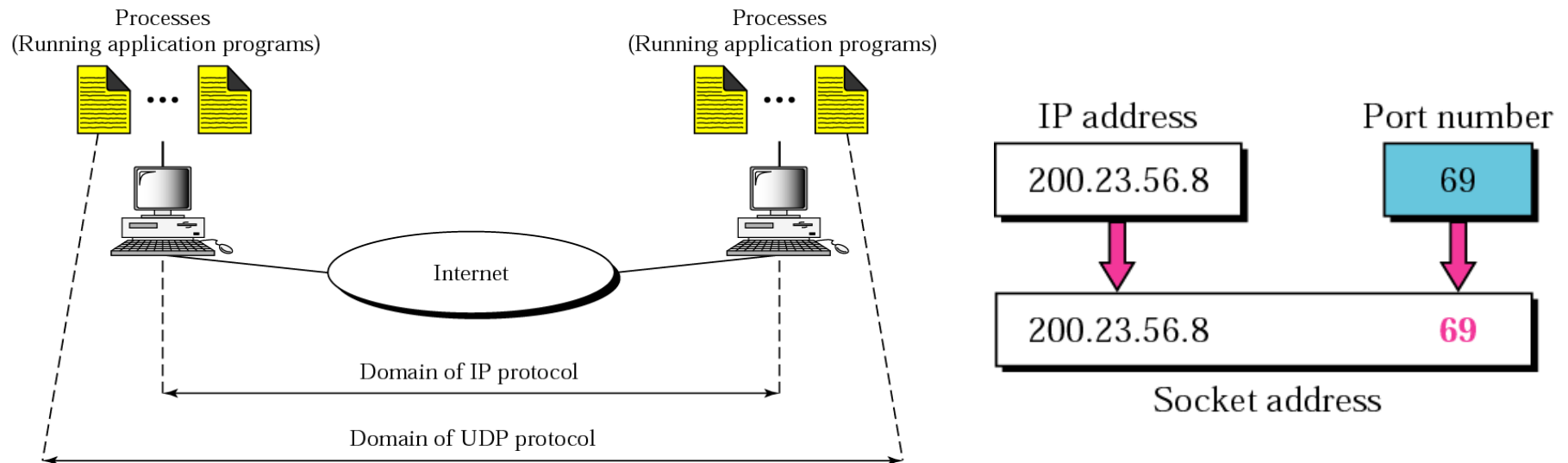
Lo strato 4 **riasmonta** i segmenti ricevuti dallo strato 3 di rete in un flusso verso il livello di sessione.

TCP/IP: livello 4 di TRASPORTO


Lo **strato di trasporto** permette la **comunicazione tra processi**, permette cioè di creare un **canale di comunicazione logica end-to-end tra processi** (applicazioni in esecuzione) remoti.

Naming dei processi applicativi connessi = **numero di porta** del processo mittente e del processo ricevente.

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



TCP/IP : Livello 4 di Trasporto



Per far comunicare due processi che operano su host diversi è necessaria la combinazione del **numero di porta** con **l'indirizzo IP** dell'host .

A **livello di sessione** si usano le **socket** per identificare completamente (host e processo) il mittente ed il ricevente.

La comunicazione può avvenire mediante **socket con connessione** (**sock-stream**: il processo mittente ed il processo destinatario prima instaurano una connessione e poi spediscono i **segmenti** che compongono il messaggio) o **senza connessione** (**sock-datagram**: il segmento viene trasmesso senza stabilire prima una connessione ed i dati viaggiano indipendentemente l'uno dall'altro).

Differenti protocolli a livello di Trasporto:

TCP (connection oriented)

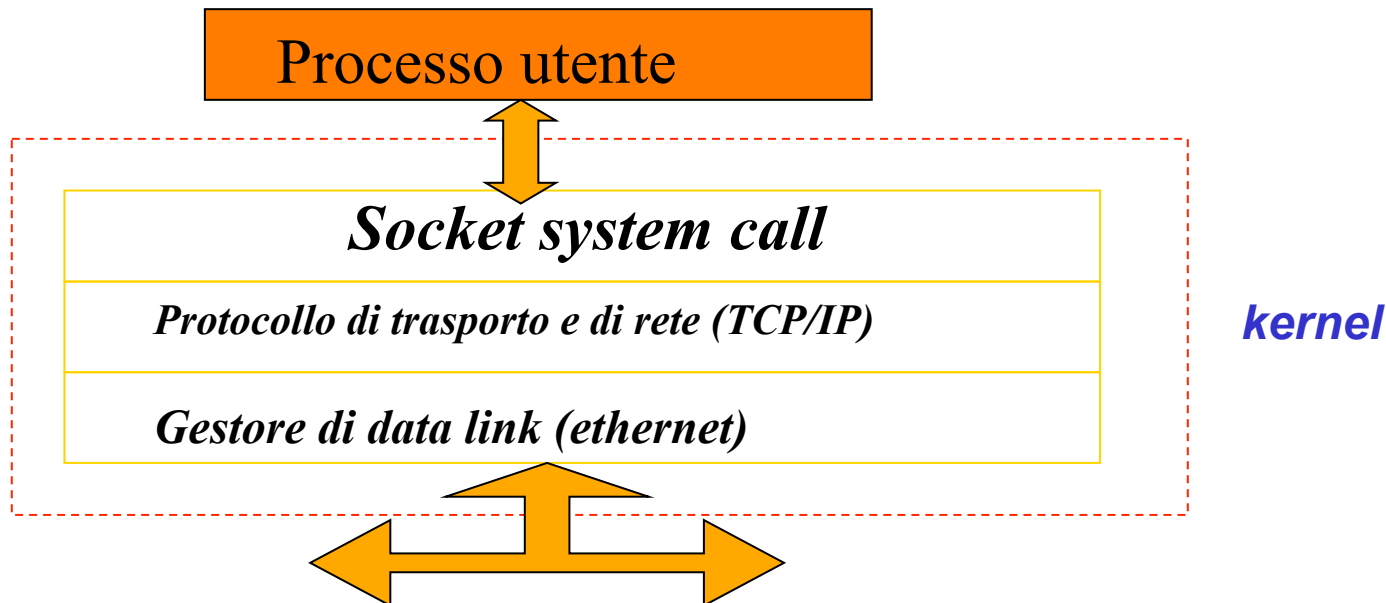
UDP (connection less).



Socket a livello di sessione : una rapida parentesi (1)

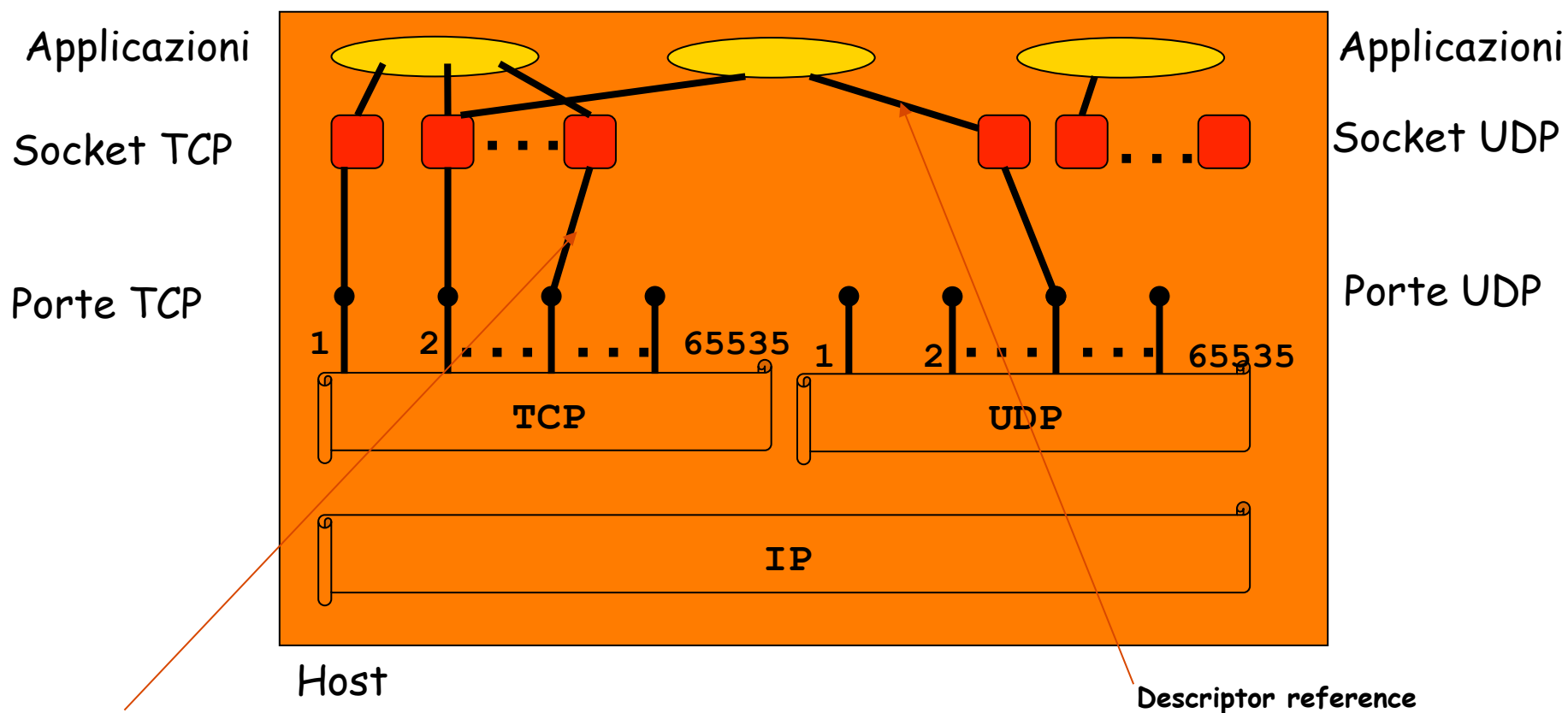
Una socket è un oggetto software che consente la comunicazione (trasferimento di dati) tra due processi remoti in una rete di calcolatori.

- ↗ *Interfaccia indipendente dalla rete*
- ↗ *Una astrazione unica per nodi comunicanti*



Socket, Protocolli e Porte su un singolo host (2)

Una socket che usa la famiglia di protocolli TCP/IP è univocamente determinata da *un indirizzo internet*, un protocollo di comunicazione (TCP o UDP) e un numero di porta



Sockets bound to ports

Socket: una visione di insieme (3)



- Nella comunicazione mediante socket uno dei processi che comunica è il **mittente** (**sender o client**) e l'altro **ricevente** (**receiver o server**).
- Tra i due processi il **server** è quello che ha controllo maggiore, poiché è il processo che inizialmente **crea la socket**.
- **Più client possono comunicare attraverso la stessa socket**, ma solo un server può essere associato ad una definita socket.

Il fatto che un programma agisca come client o come server determina un differente uso delle API Socket

- Il **Client**(il **sender**) ha bisogno di conoscere l'indirizzo del server (ma non il viceversa)
- Il **Server** (il **receiver**) può apprendere informazioni sull'indirizzo del client una volta stabilita la connessione

Le famiglie di Socket (4)

- Esistono varie **famiglie di socket**. Ogni famiglia
 - riunisce i socket che utilizzano gli stessi protocolli (Protocol Family) sottostanti,
 - supporta un sottoinsieme di stili di comunicazione e possiede un proprio formato di indirizzamento (Address Family)

Alcuni esempi di famiglie

- **Unix Domain socket**: file in una directory di un computer local host. Consentono il trasferimento di dati tra processi sulla stessa macchina Unix
- **Internet socket (AF_INET)**: consentono il trasferimento di dati tra processi posti su macchine remote connesse tramite una LAN

I tipi di Socket (5)

- Il tipo di una socket definisce una **modalità di comunicazione** che una socket usa per inviare dati:
 - **Streaming Socket (SOCK_STREAM)**: Fornisce una connessione sequenziale, affidabile e full-duplex. Il protocollo **TCP** supporta questo tipo di socket.
 - **Datagram socket (SOCK_DGRAM)**: Supporta i datagrammi (privo di connessione, messaggi inaffidabili di una lunghezza massima prefissata). Il protocollo **UDP** supporta questo tipo di socket.

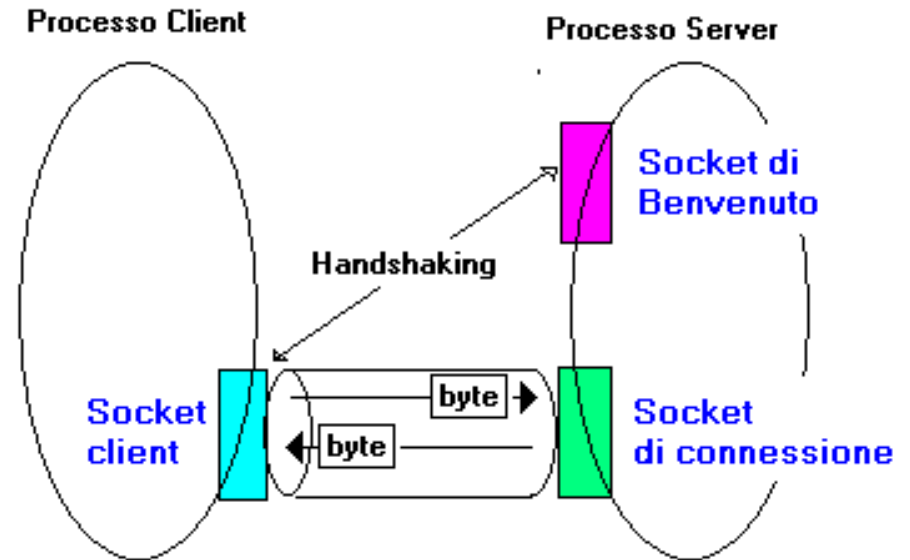
Programmazione socket (6)

➤ Il client deve contattare il server

- Il programma **server** deve essere in esecuzione come processo.
- Il programma server deve avere una porta (socket) che dia il benvenuto al contatto iniziale stabilito da un processo client in esecuzione

➤ Il client contatta il server tramite:

- la creazione di una socket locale
- la specifica di un indirizzo del processo server (IP, numero di porta relativi al processo)
- Dopo la creazione della socket nel client: TCP avvia un **handshake a tre vie** (tre messaggi di **SYNC** e **ACK**) e stabilisce una connessione TCP con il server .

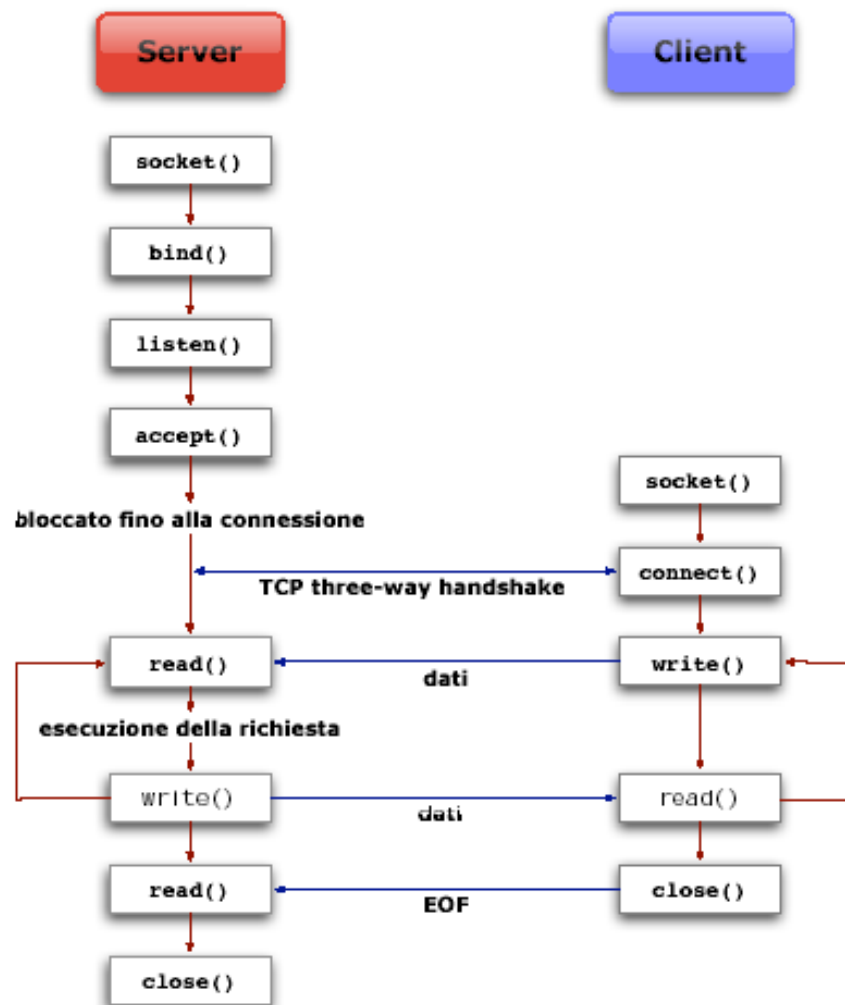


- Durante l'**handshake a tre vie** il TCP server crea una nuova socket (di connessione) dedicata a quel particolare client

Esempio 1: Interazione **TCP Client/Server** con **WSA** (7)

Le **Windows Sockets** (dette anche *WinSock*) **API (WSA)** sono una specifica estensione delle socket BSD UNIX elaborata per l'ambiente operativo Microsoft Windows.

STREAM SOCKET



Server

1. (Inizializzare una WSA Winsock API)
2. Creare una socket
3. Assegnare un local address alla socket
4. Settare la socket all' ascolto
5. Iterativamente (processo di background):
 - a. Accettare una nuova connessione
 - b. Inviare e ricevere dati
 - c. Chiudere la connessione

Client

1. (Inizializzare una WSA)
2. Creare una Socket
3. Connettersi al server
4. Inviare e ricevere dati
5. Chiudere la connessione

Esempio1b: BSD Unix Socket used for streams (8)

sender (client)

Requesting a connection

```
s = socket(AF_INET, SOCK_STREAM, 0)
•
•
connect(s, ServerAddress)
•
•
write(s, "message", length)
```

receiver (server)

Listening and accepting a connection

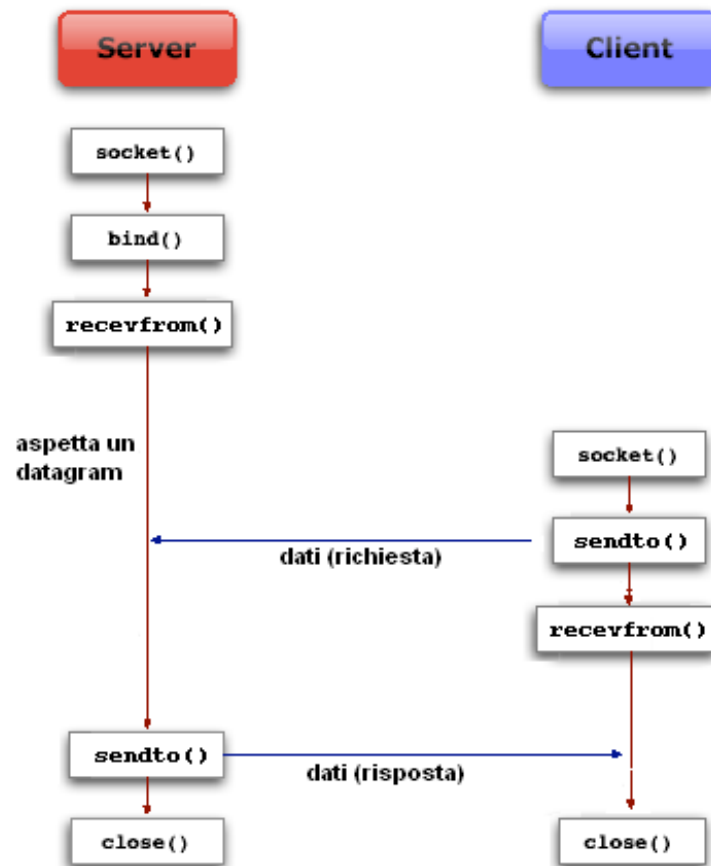
```
s = socket(AF_INET, SOCK_STREAM, 0)
•
bind(s, ServerAddress);
listen(s, 5);
•
sNew = accept(s, ClientAddress);
•
n = read(sNew, buffer, amount)
```

ServerAddress and *ClientAddress* are socket addresses

Esempio2 : Interazione **UDP Client/Server con WSA** (9)

Le **Windows Sockets** (dette anche *WinSock*) **API (WSA)** sono una specifica estensione delle socket BSD UNIX elaborata per l'ambiente operativo Microsoft Windows.

DATAGRAM SOCKET



Server

1. (Inizializzare una WSA)
2. Creare una socket
3. Assegnare un local address alla socket
4. Iterativamente:
 - a. Inviare e ricevere dati
 - b. Chiudere la connessione

Client

1. (Inizializzare una WSA)
2. Creare una Socket
3. Inviare e ricevere dati
4. Chiudere la connessione

Esempio 2b: BSD Unix Sockets used for datagrams (10)

sender (client)

Sending a message

```
s = socket(AF_INET, SOCK_DGRAM, 0)
•
•
bind(s, ClientAddress)
•
•
sendto(s, "message", ServerAddress)
```

receiver (server)

Receiving a message

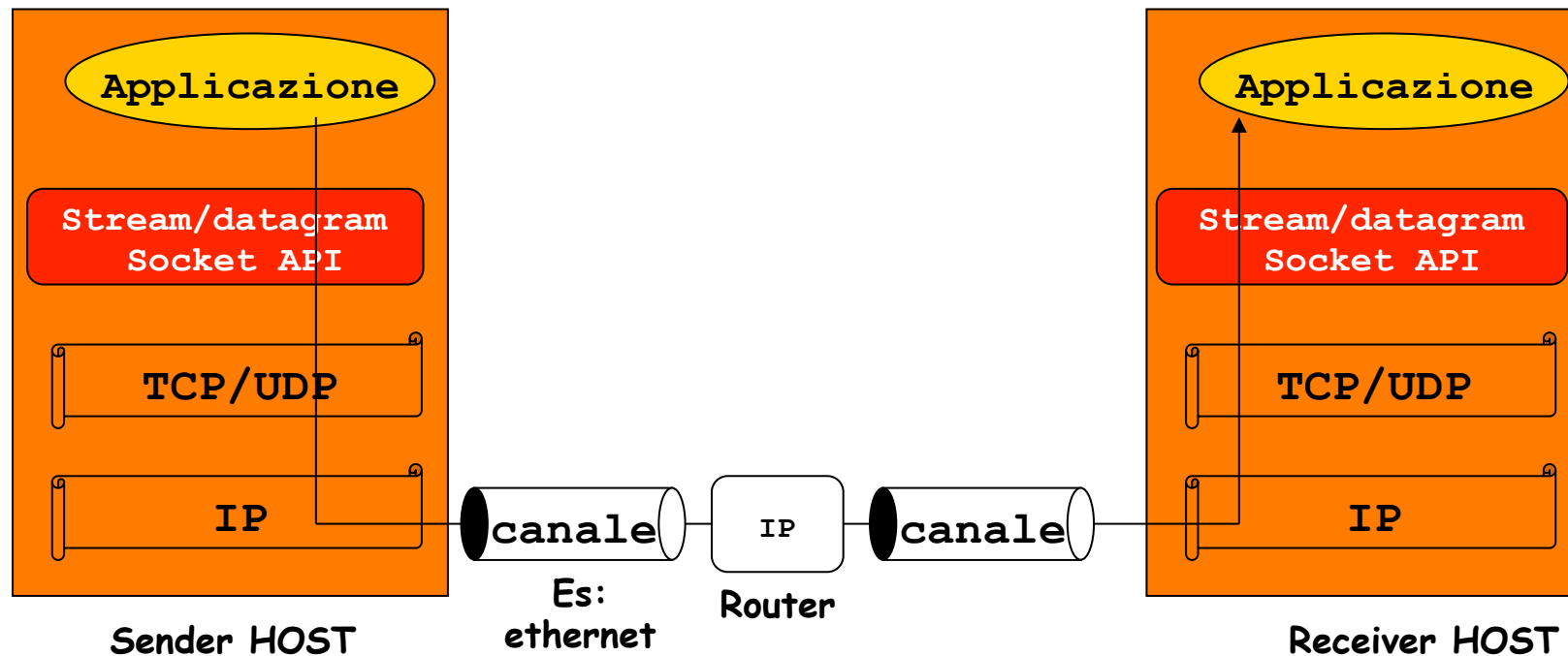
```
s = socket(AF_INET, SOCK_DGRAM, 0)
•
•
bind(s, ServerAddress)
•
•
amount = recvfrom(s, buffer, from)
```

ServerAddress and *ClientAddress* are socket addresses

TCP/IP : Livello 4 di Trasporto

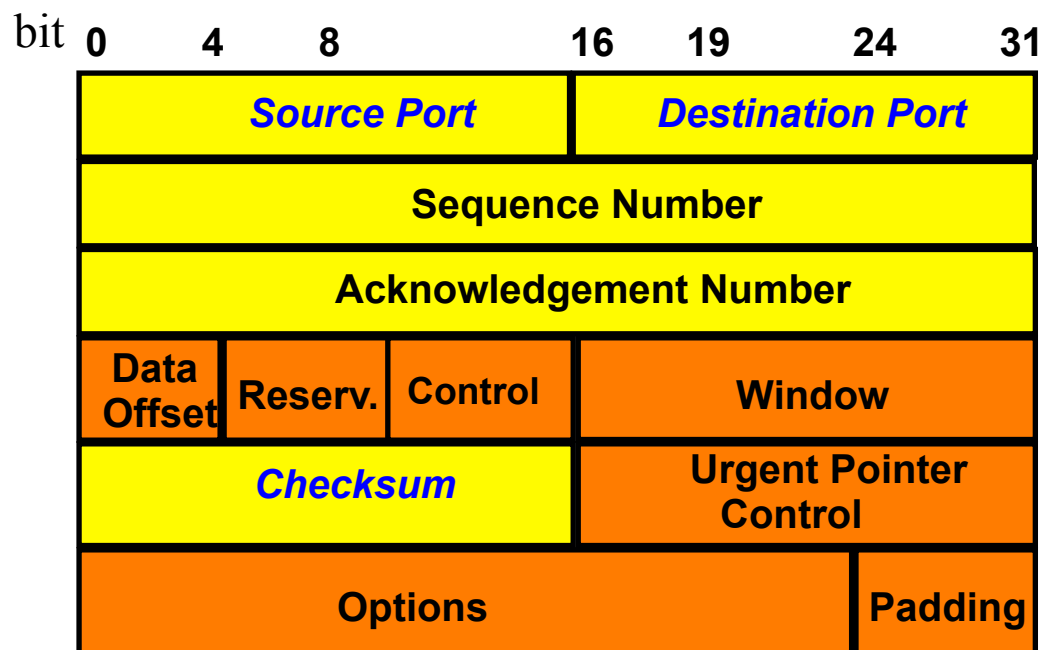
TCP (Transmission Control Protocol)

- *TCP svolge le funzioni di controllo di errore, di controllo di flusso, di controllo di sequenza e di multiplazione delle connessioni su una singola socket.*
- *TCP opera in modalità **connection oriented (streamed)**, utilizzando come identificazione dei punti di connessione le **socket** (combinazione di indirizzo IP dell'host ed il numero di porta su cui avviene la comunicazione) **di tipo stream**.*



TCP/IP : Livello 4 di Trasporto

Header TCP



TCP Header = 24 byte

Nel corso di trasmissioni di questo tipo ci si serve di numeri sequenziali e di conferme (*acknowledge*) per assicurare il trasferimento con successo dei dati.

Ciascun segmento di dati viene numerato e spedito al livello inferiore di rete. Qui il segmento viene trasformato in datagramma e spedito al destinatario.

Il destinatario una volta ricevuti i datagrammi provvede ad inviare al mittente un messaggio di avvenuta ricezione.

Questo meccanismo ovviamente appesantisce la trasmissione, ma assicura un elevato grado di affidabilità.

Le porte di processo in TCP

TCP è utilizzato da **applicativi** che richiedono la trasmissione affidabile dell'informazione:

- **TELNET** su porta 23
- **FTP (file transfer protocol)** su porta 22
- **SMTP (simple mail transfer protocol)** su porta 25
- **DNS** su porta 53
- **HTTP** su porta 80
- **POP** su porta 110

In UNIX, I numeri di porta “well-known” sono memorizzate nel file `/etc/services`. Ciascuna linea del file contiene il nome del processo server e il corrispondente numero di porta associato.

Well-known ports per TCP

Port	Protocol	Description
7	Echo	Echoes a received datagram back to the sender
9	Discard	Discards any datagram that is received
11	Users	Active users
13	Daytime	Returns the date and the time
17	Quote	Returns a quote of the day
19	Chargen	Returns a string of characters
20	FTP, Data	File Transfer Protocol (data connection)
21	FTP, Control	File Transfer Protocol (control connection)
23	TELNET	Terminal Network
25	SMTP	Simple Mail Transfer Protocol
53	DNS	Domain Name Server
67	BOOTP	Bootstrap Protocol
79	Finger	Finger
80	HTTP	Hypertext Transfer Protocol
111	RPC	Remote Procedure Call

Trasmissione TCP su Ethernet

1. L'applicazione apre **una socket** per trasmettere i dati al livello di trasporto.

2. **In trasmissione**, TCP accetta un flusso di byte **dal livello di applicazione**, frammenta il flusso di byte in **segmenti numerati** e li consegna singolarmente al protocollo IP per la costruzione di datagrammi;

In ricezione, TCP riordina i pacchetti **in arrivo dall'IP** (livello sottostante), elimina i duplicati, calcola il checksum per intero (e non solo del suo header, come fa l'IP), esegue il **controllo di flusso** per regolare differenti velocità tra macchina mittente e destinatario, **smista i segmenti ai processi** secondo la porta di destinazione.

3. **Il protocollo IP accetta i segmenti** e li include in pacchetti IP (**datagrammi**) secondo il formato previsto. IP decide inoltre le forme di routing.

4. Una volta costruiti i pacchetti, l'IP deve utilizzare il protocollo dello strato inferiore (**data link**) per trasmetterli. In particolare dovrà ottenere dall'ARP l'indirizzo MAC dell'host destinazione.

5. Il protocollo dello strato **data link** aggiunge ai pacchetti IP altri campi di controllo ed ottenuto così il **frame Ethernet** da trasmettere, utilizza i servizi messi a disposizione dal livello fisico per trasmettere i segnali (i bit) che compongono il frame.

TCP/IP : Livello 4 di Trasporto

UDP (User Datagram Protocol)

- ✓ Il protocollo *UDP* è **connectionless**, cioè non gestisce il riordinamento dei segmenti né la ritrasmissione di quelli persi, non vi è certezza dell'avvenuta ricezione da parte del destinatario dei dati spediti ma in compenso la trasmissione risulta più semplice e veloce.

Questo protocollo viene utilizzato nei casi in cui la velocità di trasmissione sia più importante della sicurezza della trasmissione (per es. in applicazioni real-time).

UDP Header = 8 byte

0	4	8	16	19	24	31
UDP Source Port				UDP Destination Port		
UDP Message Length				Checksum del datagram		

- ✓ L'UDP fornisce soltanto i servizi basilari del **livello di trasporto**:
 - moltiplicazione delle connessioni, ottenuta attraverso il meccanismo delle **porte**,
 - verifica degli errori mediante una **checksum**, inserita in un campo dell'intestazione del pacchetto.
- ✓ L'UDP è un protocollo **stateless**, ovvero non tiene nota dello stato della connessione, dunque ha rispetto al TCP informazioni in meno da memorizzare.

Le porte di processo in UDP

Le applicazioni principali che utilizzano UDP :

- **NFS (Network File System);**
- **SNMP (Simple Network Management Protocol);**
- **ICMP**
- **Applicazioni RUNIX (rwho, ruptime, ...).**

In UNIX, I numeri di porta “well-known” sono memorizzate nel file [/etc/services](#). Ciascuna linea del file contiene il nome del processo server e il corrispondente numero di porta associato.

Well-known ports used with UDP

<i>Port</i>	<i>Protocol</i>	<i>Description</i>
7	Echo	Echoes a received datagram back to the sender
9	Discard	Discards any datagram that is received
11	Users	Active users
13	Daytime	Returns the date and the time
17	Quote	Returns a quote of the day
19	Chargen	Returns a string of characters
53	Nameserver	Domain Name Service
67	Bootps	Server port to download bootstrap information
68	Bootpc	Client port to download bootstrap information
69	TFTP	Trivial File Transfer Protocol
111	RPC	Remote Procedure Call
123	NTP	Network Time Protocol
161	SNMP	Simple Network Management Protocol
162	SNMP	Simple Network Management Protocol (trap)

TCP/IP : Livello 4 di Trasporto

Trasmissione UDP

1. *Il pacchetto UDP viene incapsulato all'interno di un pacchetto IP.*
2. *Giunto a destinazione, il pacchetto viene inviato alla porta di destinazione indicata nell'intestazione UDP.*
3. *Qualora la porta non fosse disponibile, viene inviato un pacchetto ICMP (Internet Control Message Protocol) all'host mittente con messaggio di **port unreachable**.*

- ✓ *Il protocollo di trasmissione UDP realizza di fatto un modello di comunicazione di tipo “message passing” con ricezione bloccante e invio non bloccante. E' demandato al programmatore il compito di assicurare la corretta ricezione dei messaggi (datagram) nel caso essa sia considerata importante per l'applicazione.*
- ✓ *La comunicazione in UDP avviene mediante indirizzamento indiretto di tipo “molti a uno”: la primitiva **send** indirizza il proprio messaggio ad uno **specifico identificatore della porta** di accesso al processo destinazione (**hostname e port**).*
- ✓ *Il processo destinazione conosce l'identità del processo sorgente mediante **hostname e port** specificati in testa al messaggio.*
- ✓ *Mediante UDP, è anche possibile che molti client si mettano in ascolto di messaggi inviati (in broadcast) da un processo server.*