Università di Roma Tor Vergata
Corso di Laurea triennale in Informatica

Sistemi operativi e reti

A.A. 2016-17

Pietro Frasca

Parte II

Lezione 5

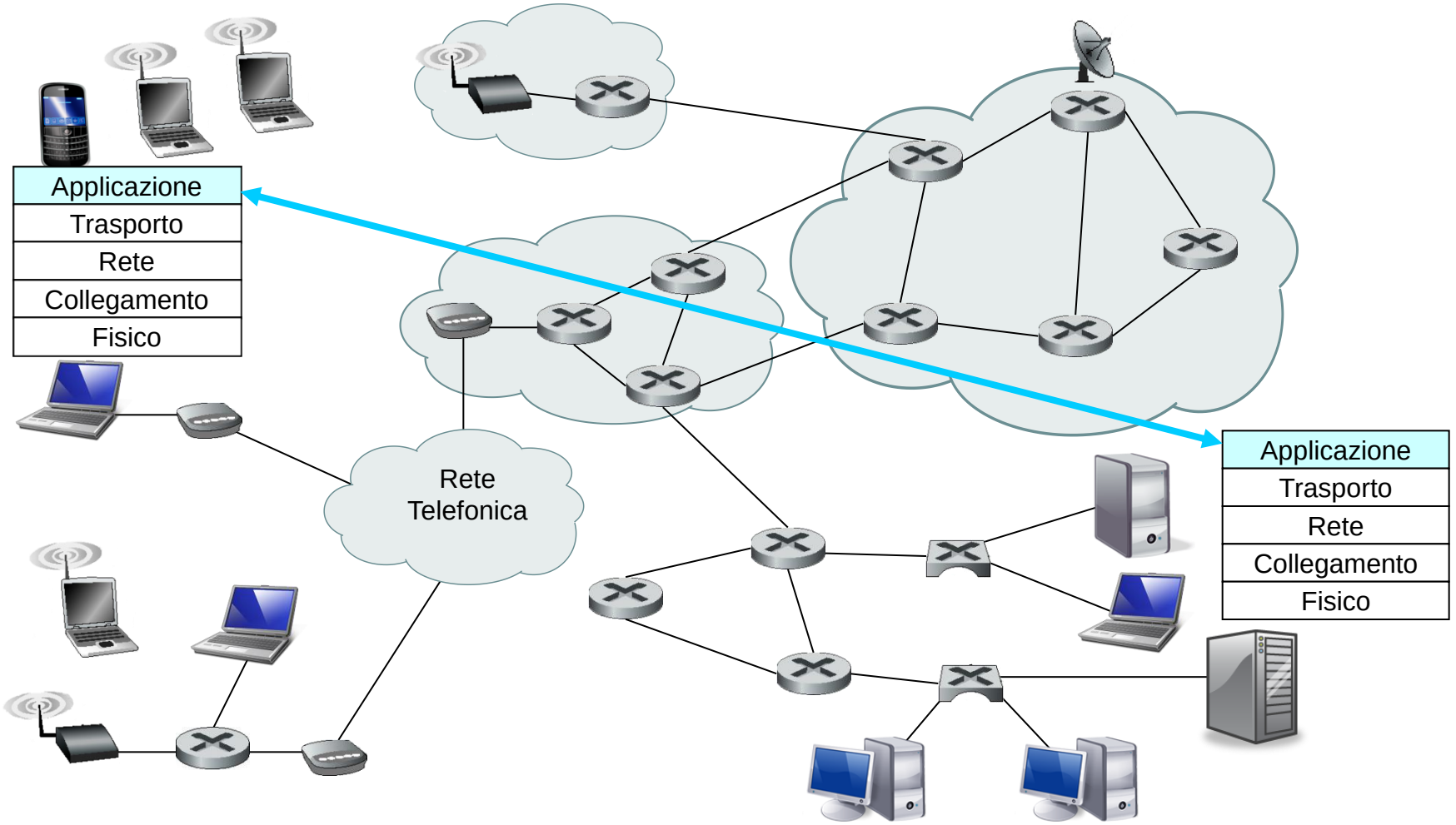
Martedì 21-03-2017

Livello di applicazione

Architetture e protocolli dello strato di applicazione

- L'architettura stratificata dello stack TCP/IP consente a processi su diversi host di comunicare tra loro **scambiandosi messaggi** attraverso la rete. Un processo mittente crea e invia messaggi e un processo destinatario riceve questi messaggi ed eventualmente rinvia messaggi di risposta al mittente.
- Le applicazioni di rete per comunicare devono utilizzare **protocolli** che definiscono:**il formato dei messaggi e l'ordine in cui essi sono scambiati, e la definizione delle operazioni da svolgere nella fase di trasmissione e alla ricezione dei messaggi.**
- Vari protocolli dello strato di applicazione sono di pubblico dominio e sono definiti nelle **RFC** (Reference For Comment).

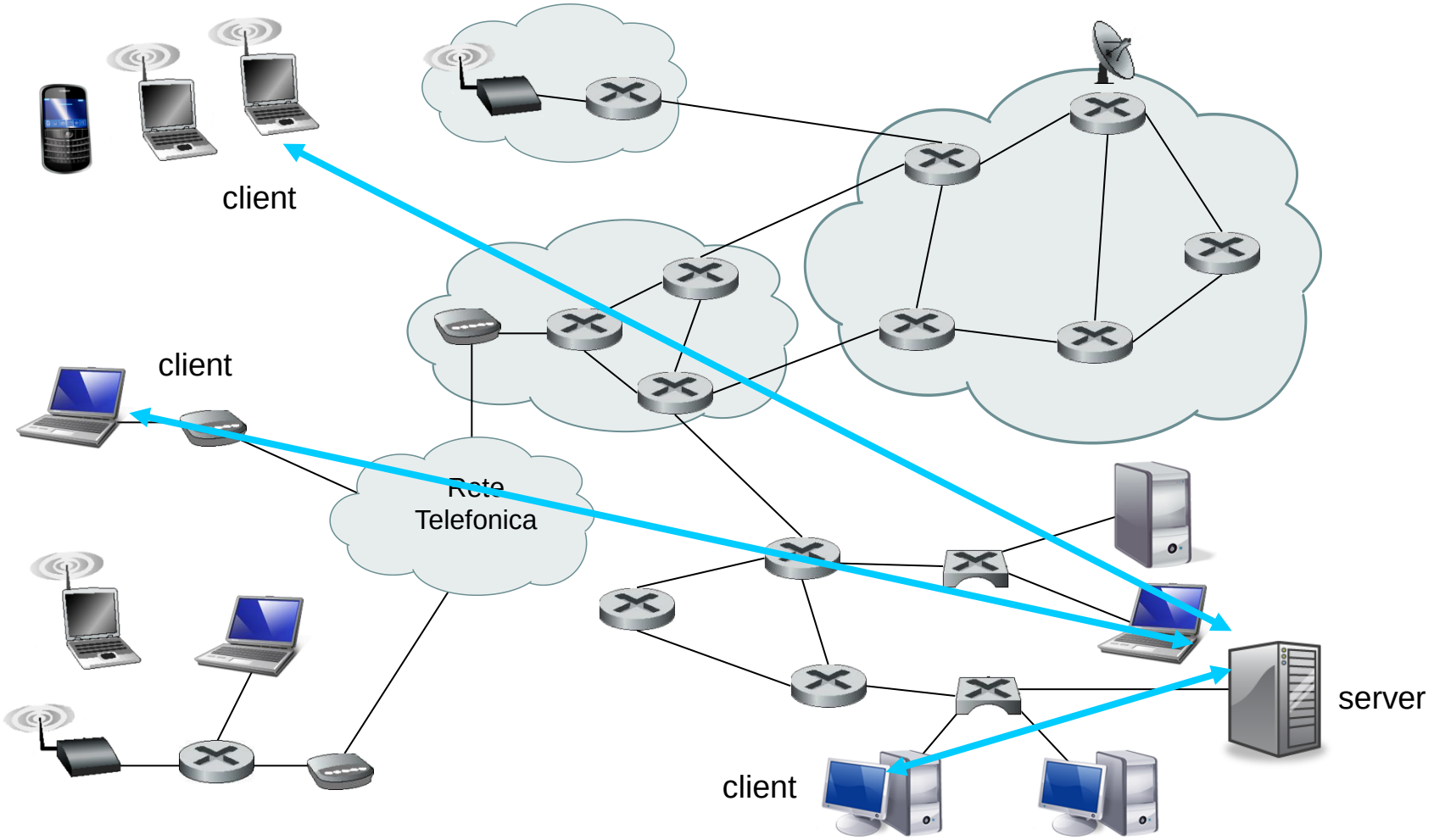
Nella figura è mostrato come i processi comunicano tra loro usando lo strato di applicazione della pila protocollare Internet.



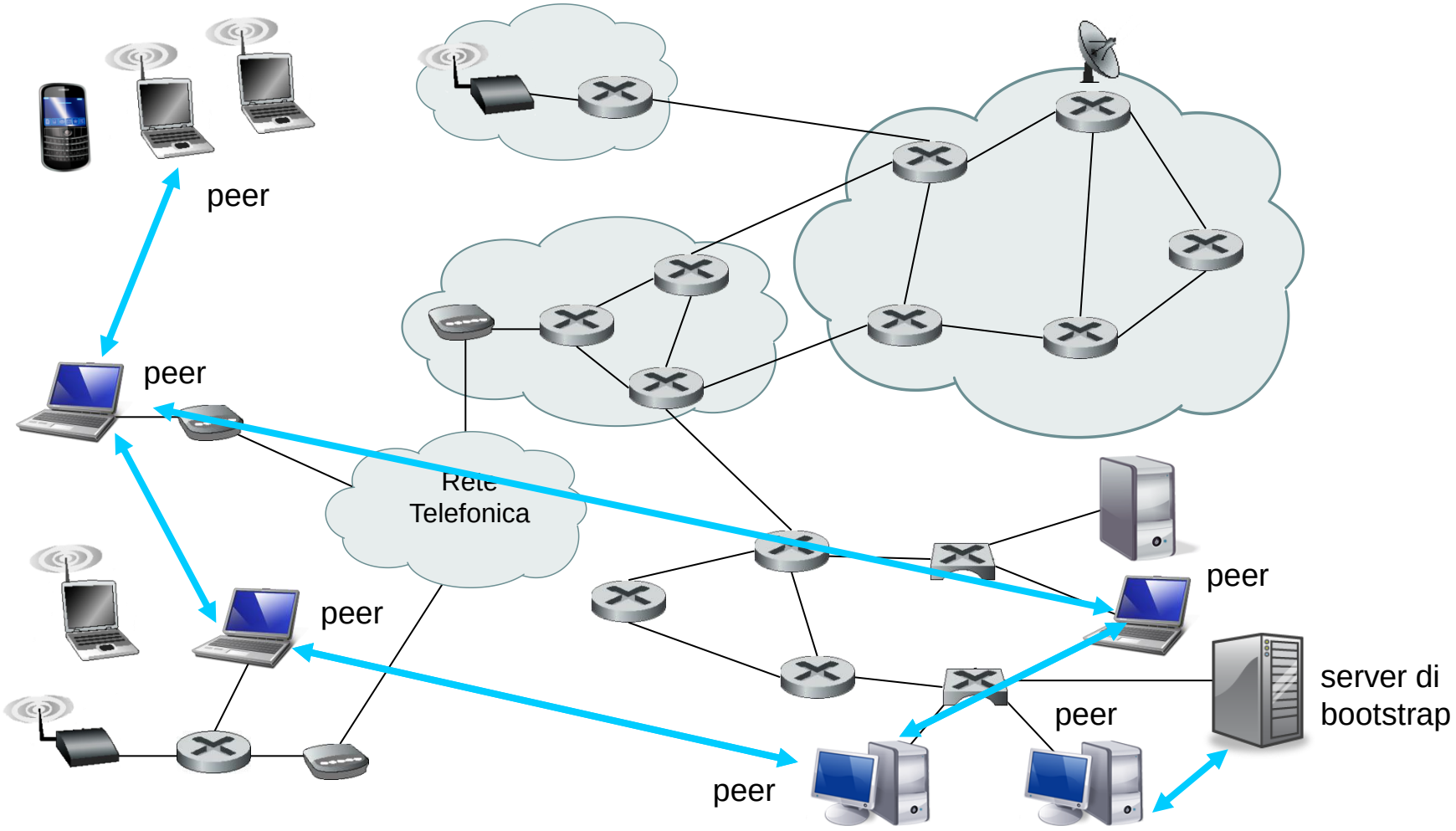
- Le principali architetture delle applicazioni di rete sono: **client/server** e **Peer To Peer** (P2P).
- Molte applicazioni di rete hanno un'architettura client/server e pertanto si realizzano in due parti, **un lato client e un lato server**. Il lato client che gira su un host comunica con il lato server che gira su un altro host. Ad esempio, in un browser è implementato il lato client del protocollo HTTP e in un server Web è implementato il lato server. Il browser invia messaggi per richiedere pagine web e il server risponde inviandole.
- Con l'architettura client/server i client non comunicano direttamente tra loro.
- Le applicazioni di rete possono anche implementare sia il lato client che il lato server sullo stesso host. In questo caso, per convenzione, si indica come client l'host che inizia la comunicazione.

- Nell’architettura P2P, gli host connessi ad una rete P2P sono chiamati "pari" (*peer) e* possono scambiarsi file direttamente tra di loro.
- Ogni pari si comporta sia da client che da server, è cioè sia un distributore sia un fruitore di contenuti.
- In un sistema P2P, in ogni istante, sono connessi un elevato numero di pari, e ogni pari generalmente ha molti file da condividere. Se un pari **P** vuole ottenere un particolare file, allora il pari **P** deve avere funzionalità per determinare gli indirizzi IP dei pari connessi che hanno il file desiderato.
- L’architettura P2P è quindi molto più complessa della client/server.

La figura seguente mostra l'architettura client/server.



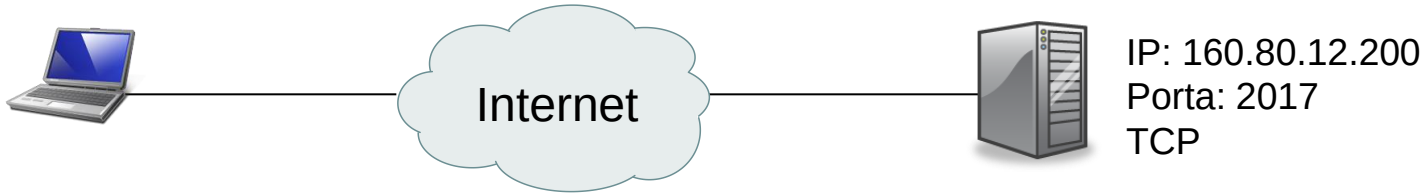
Nella figura è mostrata l'architettura P2P.



Indirizzamento dei processi

Per consentire a processi remoti di comunicare tra loro è necessario che essi siano identificati nella rete in modo univoco. Un processo è specificato da due informazioni:

1. **il nome o l'indirizzo del host su cui il processo è attivo**
2. **un identificatore del processo sull'host**.



- In Internet, un host (o meglio una sua scheda di rete) è identificato univocamente da un indirizzo di 32 bit (IPv4) e/o di 128 bit (IPv6) detto **indirizzo (o numero) IP.**
- Il processo del host è identificato da un **numero di porta** di 16 bit che deve essere univoco per uno stesso protocollo di trasporto, nel caso siano presenti più processi di rete sullo stesso host.

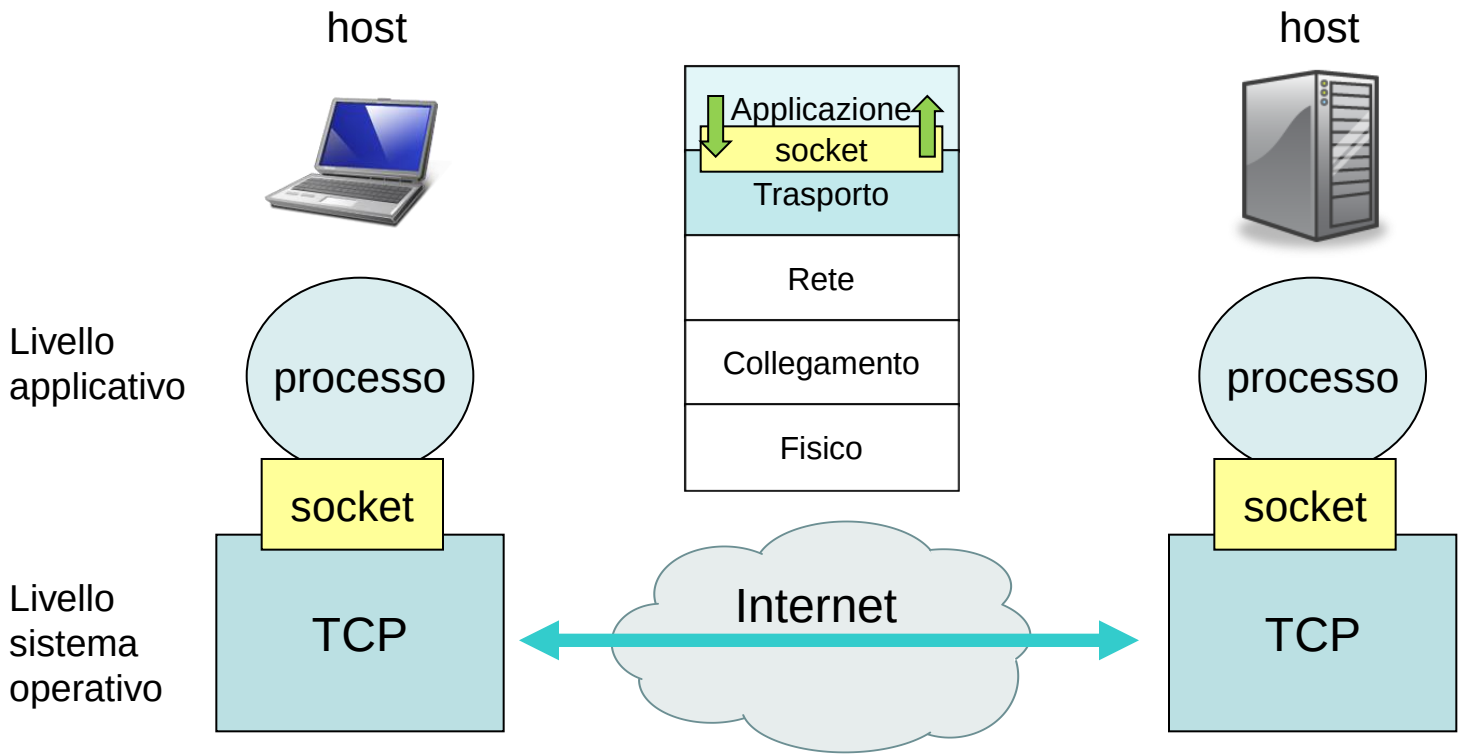
- Ai protocolli dello strato di applicazione che sono stati standardizzati sono stati assegnati numeri di porta specifici. Per esempio, il protocollo HTTP, utilizzato nel Web è identificato dal numero di porta 80; il protocollo SMTP utilizzato nella posta elettronica è identificato dal numero di porta 25.
- Questi numeri di porta assegnati sono chiamati **numeri di porta ben conosciuti (well known ports).**
- I numeri di porta ben conosciuti sono assegnati dallo **IANA (Internet Assigned Numbers Authority**) e sono compresi tra 0 e 1023.
- Quando si realizzano nuove applicazione di rete, all'applicazione non devono essere assegnati i numeri di porta ben conosciuti.

Server web: porta 80

Server SMTP: porta 25

Processi di comunicazione nella rete

- Un processo invia messaggi e riceve messaggi dalla rete attraverso un interfaccia software con lo strato di trasporto detta ***socket***. Una volta che il messaggio è arrivato all'host di destinazione, il messaggio arriva al socket del processo ricevente. Possiamo pensare ad un socket come ad una porta o ad una presa.



Servizi di trasporto per le applicazioni

- Quando si progetta e sviluppa un'applicazione di rete è necessario stabilire quali requisiti di servizio debba possedere e quindi scegliere il protocollo di trasporto più adeguato.
- A grandi linee, possiamo classificare i requisiti di servizio di un'applicazione in quattro proprietà:
  - **Trasferimento affidabile dei dati**,
  - **larghezza di banda (throughput),**
  - **timing (temporizzazione)**
  - **sicurezza.**

Trasferimento affidabile dei dati

- Alcune applicazioni, come posta elettronica, trasferimento dei file, web etc, richiedono un trasferimento di dati affidabile, cioè **senza perdita di dati**.
- Altre applicazioni dette ***perdite-tolleranti***, come ad esempio le applicazioni multimediali audio/video in soft real-time possono tollerare qualche perdita di dati. In queste applicazioni multimediali, la perdita dei dati, se non eccessiva, produce accettabili difetti di riproduzione audio/video.

Larghezza di banda

- Alcune applicazioni sono dette a **larghezza di banda dipendenti** in quanto devono essere in grado di trasmettere i dati a una determinata velocità. Per esempio, se un'applicazione audio multimediale per funzionare correttamente deve trasmettere dati a 32 kbit/s, è necessario che tale larghezza di banda sia disponibile, altrimenti dovrebbe generare un'eccezione in quanto la trasmissione audio ad una velocità troppo bassa è insufficiente.
- Le **applicazioni elastiche** *(elastic application)*, invece, possono funzionare correttamente sia con una larghezza di banda grande sia piccola. Ad esempio, posta elettronica e trasferimento di file, sono applicazioni elastiche.

Timing

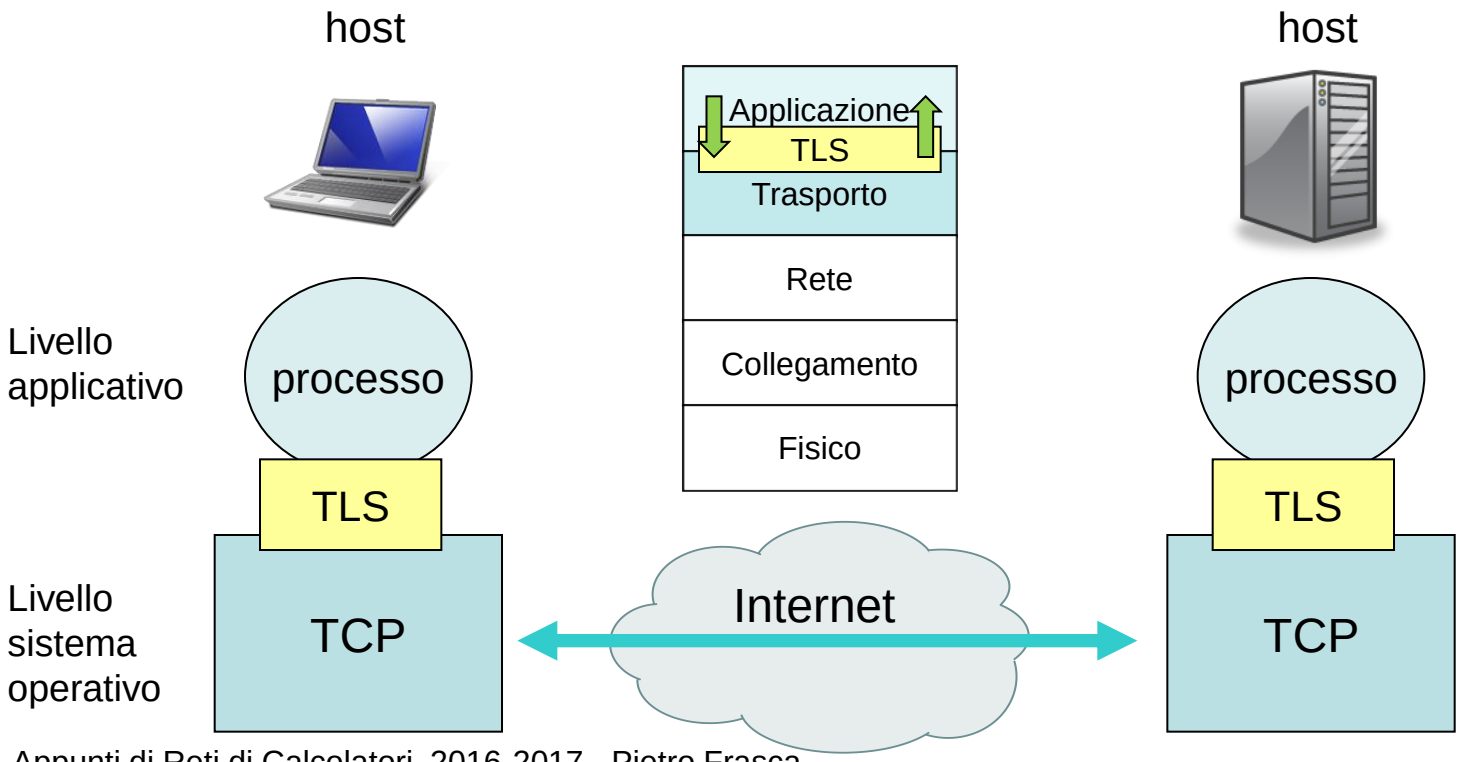
- Le applicazioni interattive in tempo reale, richiedono brevi ritardi di trasmissione dei dati. Ad esempio, la telefonia Internet richiede ritardi inferiori a circa 150 millisecondi. Se i ritardi sono compresi tra 150 e 400 millisecondi la comunicazione può essere accettabile ma ritardi superiori ai 400 millisecondi producono pause innaturali nella conversazione che diventa quindi incomprensibile.
- Per applicazioni non in tempo reale non c'è alcuna limitazione sui ritardi punto-punto, anche se ovviamente un piccolo ritardo è meglio di un grande ritardo.
- Nelle reti, ed in particolare in internet, si usa il termine **jitter** per indicare la variazione media del ritardo di ricezione dei pacchetti trasmessi, causata dalle code dei router congestionati.

Più in generale, in elettronica e telecomunicazioni con il termine **jitter** si indica la variazione di una o più caratteristiche di un segnale come, ad esempio, la variazione di ampiezza, di frequenza, di fase.
La figura seguente riassume affidabilità, larghezza di banda, e timing richiesti da alcune applicazioni Internet.

Applicazione	Perdita dei dati	Larghezza di banda	Sensibile al tempo
Trasferimento di file	No	Elastica	No
E-mail	No	Elastica	No
Documenti Web	No	Elastica (pochi kbit/s)	No
Audio/Video in tempo reale	Tollerabile	Audio: pochi kbit/s-1 Mbit Video: 10 kb-5 Mbit	Sì, centinaia di ms
Audio/Video memorizzati	Tollerabile	Come sopra	Sì, pochi secondi
Giochi interattivi	Tollerabile	Pochi kbit/s-10 kbit	Sì: centinaia di ms
Messaggi istantanei	No	Elastica	Sì e no

Sicurezza

- Sia TCP che l'UDP non forniscono servizi di cifratura dei dati.
- Per ovviare a queste limitazioni sono stati sviluppati **SSL (Secure Socket Layer)** sostituito da **TLS (Transport Layer Security)** un miglioramento, a livello di sicurezza, delle socket del TCP.
- Quando un'applicazione utilizza TLS i dati vengono cifrati nel lato mittente e vengono decifrate nella destinazione.



Servizi forniti dai protocolli di trasporto

- Lo strato di trasporto di Internet ha due protocolli (dal 2000 è presente anche **SCTP**):

  UDP *(User Datagram Protocol)*
  TCP *(Transmission Control Protocol).*

- Per realizzare un'applicazione di rete, è necessario decidere dapprima se scegliere di usare UDP o TCP. I due protocolli offrono un modello di servizio molto diverso alle applicazioni.

Servizi TCP

- Il TCP fornisce alle applicazioni un servizio orientato alla connessione e un servizio di trasferimento affidabile dei dati.
- ***Servizio orientato alla connessione****:* Prima che due processi applicativi inizino a comunicare tra loro, il TCP effettua una procedura iniziale detta **handshake** ("**stretta di mano**"), terminata la quale si instaura una connessione TCP fra i **socket** dei due processi.
- La comunicazione è di tipo **full-duplex** che consente ai due processi di inviare contemporaneamente messaggi in entrambe le direzioni. La comunicazione può essere chiusa sia dal client che dal server.
- ***Servizio di trasporto affidabile****: Il TCP* garantisce ai processi che tutti i dati trasmessi arrivano a destinazione senza errori e nello stesso ordine di partenza.

- Il TCP fornisce anche un servizio di **controllo della congestione**, un servizio per regolare il traffico sulla rete. Il **controllo della congestione** del TCP modula la velocità di trasmissione dei dati di un processo (client o server) quando la rete è congestionata.
- Con il TCP la velocità di trasmissione è regolata dal servizio di controllo della congestione, che riduce la velocità media di trasmissione del mittente quando la rete è congestionata. Pertanto, il TCP non garantisce una velocità minima di trasmissione dei dati e quindi non garantisce anche un ritardo minimo.

Servizi UDP

- L'UDP è un protocollo di trasporto semplice.
- L'UDP è un protocollo *senza connessione*, quindi non c'è la fase di handshake prima che i due processi inizino a comunicare.
- L'UDP fornisce un servizio di trasferimento dati non affidabile. Pertanto, i messaggi che arrivano al socket ricevente possono non arrivare in ordine o non arrivare per niente.
- L'UDP non implementa il servizio di controllo della congestione, così un processo può inviare dati nel socket fino alla massima velocità consentita.
- L'UDP è più adeguato per applicazioni soft real-time le quali generalmente possono tollerare qualche perdita di dati, ma richiedono che la velocità di trasmissione non scenda sotto una determinata soglia.
- Come il TCP, l'UDP non dà garanzie sul ritardo.

Applicazioni	Protocollo dello strato di applicazione	Protocollo di trasporto sottostante
Posta elettronica	SMTP (RFC 821)	TCP
Accesso a terminale remoto	Telnet (RFC 854)	TCP
Web	HTTP (RFC 2616)	TCP
Trasferimento di file	FTP (RFC 959)	TCP
File server remoto	NFS (McKusik 1996)	UDP o TCP
Streaming multimediale	Spesso proprietario (per esempio, Real Networks)	UDP o TCP
Telefonia Internet	Spesso proprietario (per esempio, Dialpad)	Tipicamente UDP

Applicazioni di rete

- Descriveremo quattro applicazioni oggi molto diffuse in Internet: il **Web**, **FTP** (**trasferimento di file)**, la **posta elettronica** e il **DNS** ***(Domain Name System,*** **Sistema dei nomi**). Inoltre parleremo, di alcuni tecnologie e protocolli usati nelle applicazioni di **condivisione di file da pari a pari (P2P, Peer to Peer).**

Il web

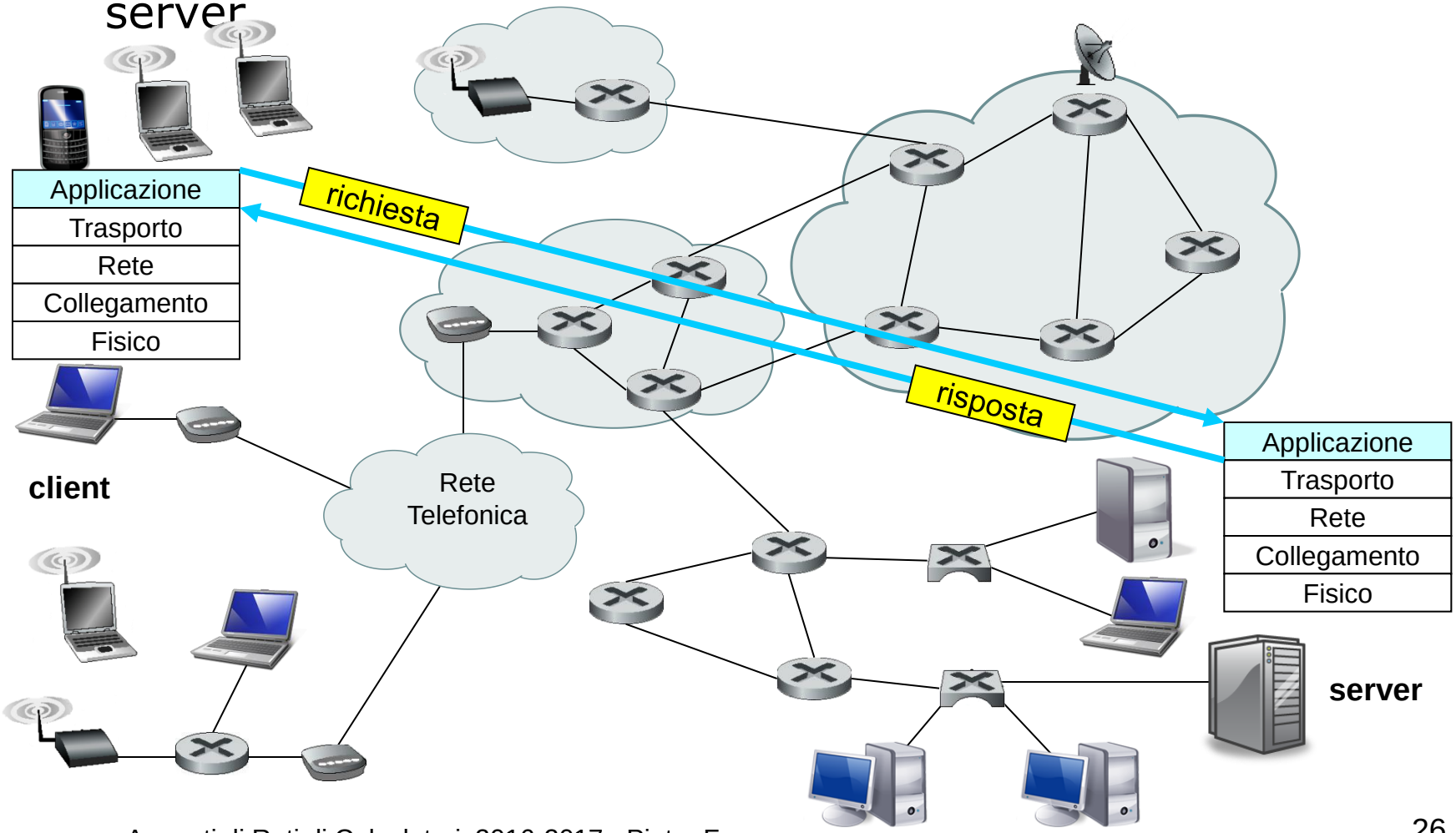
- Fino al 1990 Internet era usata soprattutto usata in ambito universitario per il collegamento a host remoti, per trasferire dati da host locali a host remoti e viceversa, per accedere a notizie e per scambiarsi messaggi di posta elettronica.
- All'inizio degli anni '90, fu realizzata l'applicazione **World Wide Web (Web)** che attirò l'interesse del grande pubblico. Il Web ha cambiato profondamente il modo di interagire delle persone sia all'interno che all'esterno dell'ambiente lavorativo cambiando vari comportamenti e modalità di lavorare delle persone.
- Ha permesso la nascita di migliaia di società.
- Ha portato Internet a diventare praticamente l'unica e sola rete per dati. Prima, soprattutto presso le università e le grandi società, erano molto diffuse le reti SNA di IBM e DECNET di Digital (ora HP).

- Una delle caratteristiche più importanti del Web è che permette la fruizione delle informazioni a ***richiesta***. Gli utenti accedono ai contenuti informativi e multimediali quando lo desiderano. Questo funzionamento è molto diverso dalle trasmissioni radio/televisive, che richiedono agli utenti di sintonizzarsi in determinati orari per ottenere informazioni che sono trasmesse in base ad un palinsesto programmato.
- Oltre a essere a richiesta, il Web ha molte altre interessanti caratteristiche. Consente agli utenti di pubblicare informazioni in modo molto semplice ed economico.
- I **motori di ricerca** aiutano nella ricerca delle informazioni.
- Una **pagina web** (**documento**) è formata da file di vario formato, come file di testo HTML, immagine JPEG, video divx, etc.) che sono indirizzabili attraverso **URL (Uniform Resource Locator).**

- Il Web utilizza vari standard e tecnologie:
  - Il **protocollo HTTP *(HyperText Transfer Protocol*)**. E' il protocollo dello strato di applicazione Web, che consente il trasferimento di file (hyperText) tra server web e browser.
  - **HTML e CSS** per la formattazione dei documenti;
  - i **browser** (per esempio, Google Chrome, Netscape Navigator, Mozilla Firefox, Safari e Microsoft Internet Explorer) che sono il lato client dell'applicazione web.
  - i **server Web** (per esempio, i server Apache, IIS (Internet Information Services) di Microsoft e Netscape) che sono il lato server dell'applicazione web.
  - **URL (URI)**, (Uniform Resource Locator) identificatore di risorse.
  - **DBMS** (Sistemi di gestione di basi di dati)
  - **Linguaggi di programmazione (lato client)** come Java (applet), JavaScript, VBScript, etc.
  - **Linguaggi di programmazione (lato server)** come java, JavaScript, php, VBScript (asp), etc, per la generazione dinamica delle pagine

Lati client e server di un'applicazione web

- Un'applicazione web ha due "lati", un **lato client** e un **lato server**. Un browser implementa il lato client del protocollo HTTP, e un server Web ne implementa il lato server

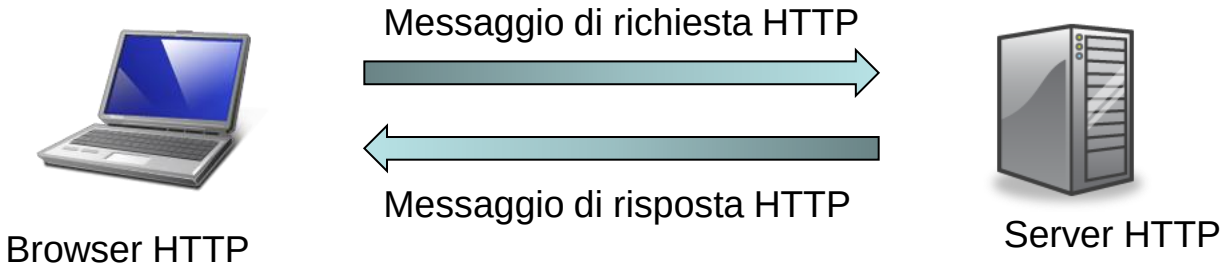


- Per esempio, se una pagina Web contiene testo HTML e cinque immagini JPEG, allora la pagina Web è composta da sei file: il file base HTML più le cinque immagini.
- Il file base HTML contiene gli URL (indirizzi) dei file. Ciascun URL ha tre componenti fondamentali: il **protocollo** utilizzato, il **nome** *(o l'indirizzo IP)* del server che contiene gli oggetti e il nome del **percorso (path)** per raggiungere il file.
- Per esempio, l'URL **http://www.pf.uniroma2.it/reti/lezioni/lezione5.html** ha come hostname **www.pf.uniroma2.it** e **/reti/lezioni/lezione4.html** come nome del path del file html e come protocollo http.
- Un browser visualizza la pagina Web richiesta e consente molte caratteristiche di navigazione e configurazione. I server Web implementano il lato server dell'HTTP e memorizzano i file web, ciascuno indirizzabile da un URL.
- Server Web molto diffusi sono Apache, Microsoft Internet Information Server e Netscape Enterprise Server.

Il protocollo HTTP

- L'**HTTP (Hypertext Transfer Protocol)** è il protocollo dello strato di applicazione del Web. L'HTTP è implementato in due parti: un programma **client** e uno **server** i quali comunicano tra loro, scambiandosi **messaggi di richiesta e di risposta**.
- La prima versione HTTP/1.0 fu realizzata nel 1991. Nel 1997 il protocollo fu aggiornato alla versione HTTP/1.1 che eliminò alcuni problemi e limitazioni della versione precedente.
- Le due versioni sono compatibili ed entrambe usano il **TCP** come protocollo dello strato di trasporto.
- Le due versioni dell'HTTP sono descritte negli [RFC 1945] e [RFC 2616] rispettivamente per le versioni 1.0 e 1.1.
- HTTP/2 è la nuova versione di HTTP. E' stato sviluppato dal Working Group Hypertext Transfer Protocol dell'Internet Engineering Task Force. E' stato proposto come standard nel dicembre 2014.

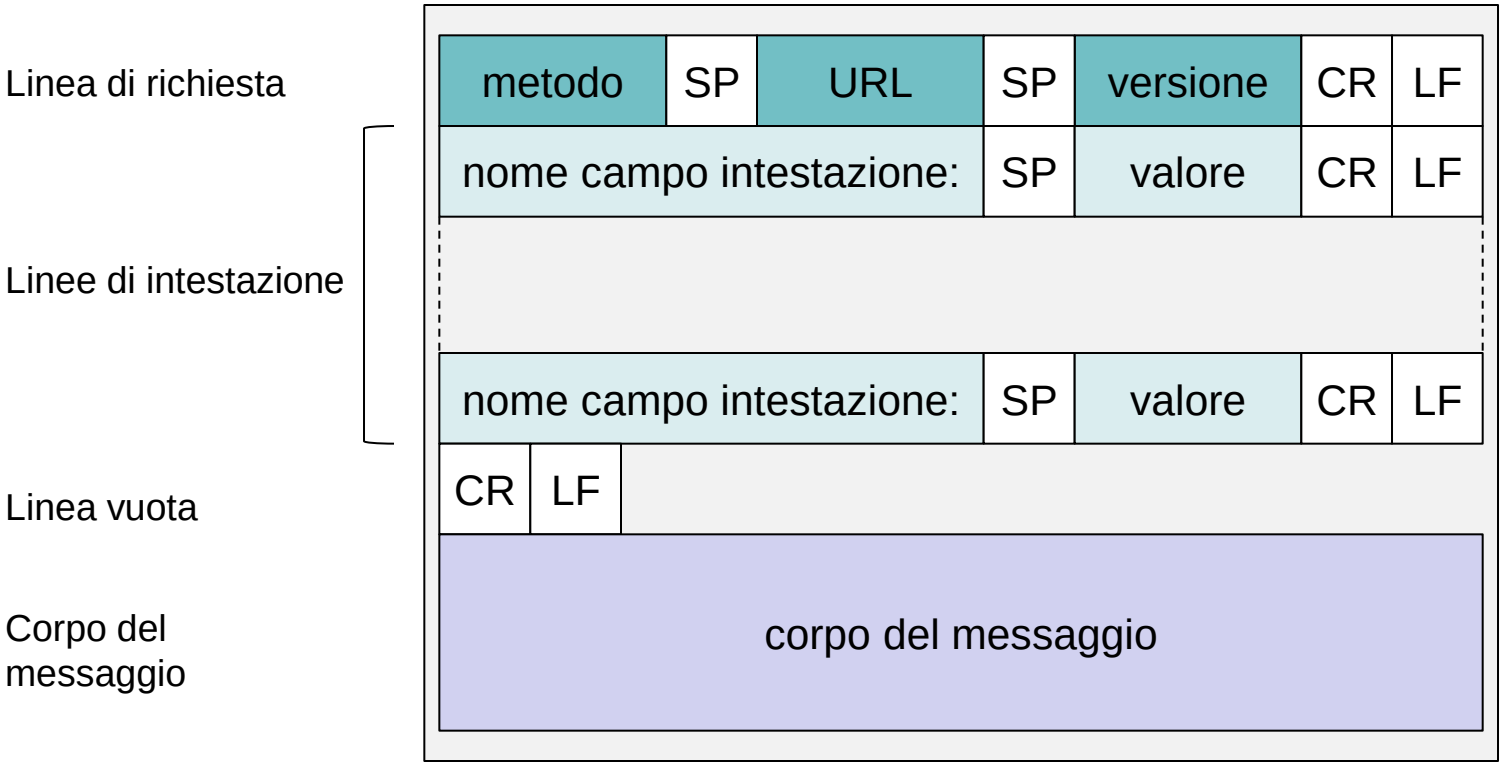
- Il funzionamento, a grandi linee, dell'interazione tra client e server è illustrata in figura.



- Quando un utente richiede una pagina Web (per esempio, cliccando su un hyperlink), il browser crea e invia messaggi di richiesta HTTP per gli oggetti nella pagina al server. Il server riceve la richiesta e risponde con messaggi di risposta HTTP contenenti gli oggetti richiesti.
- E' importante notare che l'HTTP non gestisce alcuna informazione relativa ai client che richiedono file al server. Dato che un server non conserva le informazioni relative al client è detto **protocollo senza stato** *(stateless protocol).*

Formato del messaggio HTTP

- Ci sono due tipi di messaggi HTTP, **messaggi di richiesta** e **messaggi di risposta**.
- Il formato del messaggio di richiesta è illustrato nella figura seguente.



Esempio di messaggio di richiesta HTTP

- Un tipico messaggio di richiesta HTTP:

```
GET /lezioni/lez2.html HTTP/1.1
Accept-Language: it-IT
User-Agent: Mozilla/4.0
Host: www.pf.uniroma2.it
Connection: close
```

- Il messaggio di richiesta è scritto in formato ASCII che è leggibile. Quindi, non è garantita la riservatezza delle informazioni.
- Un messaggio di richiesta può avere un numero di linee variabile, anche una sola linea se la versione HTTP è la 1.0.
- La prima linea di un messaggio di richiesta HTTP è detta **linea di richiesta** *(request line).* Le linee successive prendono il nome di **linee di intestazione** *(header line).*
- La linea di richiesta ha tre campi:
  - **il campo metodo**
  - **il campo URL**
  - **il campo versione dell'HTTP**.
- Il campo metodo può assumere vari valori, tra cui **GET**, **POST** e HEAD.

- Il metodo GET è il metodo di richiesta più usato. Il metodo GET è usato quando il browser richiede un oggetto, con l'oggetto richiesto identificato nel campo URL.
- Nel nostro messaggio di esempio, la linea di richiesta indica che il browser richiede la pagina `/lezioni/lez2.html` e che viene utilizzata la versione HTTP/1.1.
- Ora guardiamo le linee di intestazione dell'esempio.

  - **Accept-Language: it-IT**, indica che il browser è configurato per richiedere preferibilmente una versione del file in italiano, se disponibile.
  - **User-Agent: Mozilla/4.0**, specifica il nome del browser utilizzato per la richiesta. Mozilla/4.0 è un browser prodotto da Netscape. Questa linea di intestazione è utile per problemi di compatibilità, in quanto il server potrebbe inviare differenti versioni dello stesso file a differenti tipi di browser per ottenere una corretta visualizzazione.
  - **Host: www.pf.uniroma2.it** specifica il nome del server sul quale è memorizzato il file richiesto.
  - **Connection: close**, indica che il browser richiede al server di utilizzare la connessione non persistente, nonostante che il browser che invia questo messaggio di richiesta implementi HTTP/1.1 che per default funziona con le connessioni persistenti.

- Dopo le linee dell'intestazione c'è una riga vuota (CR+LF) e quindi il "**corpo del messaggio**" *(entity body).* Il corpo del messaggio è usato con il metodo POST ma non è usato con il metodo GET.
- Generalmente, il metodo POST si usa quando nella pagina web fornita al client sono presenti ***form*** (moduli), costituiti da elementi per l'input di dati. In questo caso il corpo del messaggio contiene ciò che l'utente ha inserito nei campi di input del form.
- Tuttavia, anche se con dei limiti, il metodo GET può essere usato per inviare i valori inseriti nei campi di un form. In questo caso i campi del form sono accodati all'url sottoforma di una stringa detta **QUERY_STRING**. Ad esempio, nell'URL:

  http://www.cs.uniroma2.it/persone/rubrica.php?nome=mario&numero=0672545411

- la parte che segue il punto interrogativo prende il nome di **query_string**.
- Il metodo HEAD è utile per il debugging. Quando un server riceve una richiesta che usa il metodo HEAD, invia una risposta ma non invia l'oggetto richiesto.
- La versione HTTP/1.0 permette tre tipi di metodi: GET, POST e HEAD. La versione HTTP/1.1 oltre a questi tre metodi consente altri metodi, tra cui PUT e DELETE. Il metodo PUT consente ad un utente, di caricare un oggetto su una specifica directory di un Web server (funzione di upload) mentre DELETE si usa per cancellare un oggetto da un server Web.