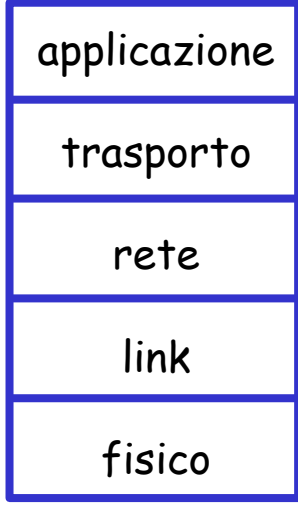
Lo strato Applicativo
URI-HTTP

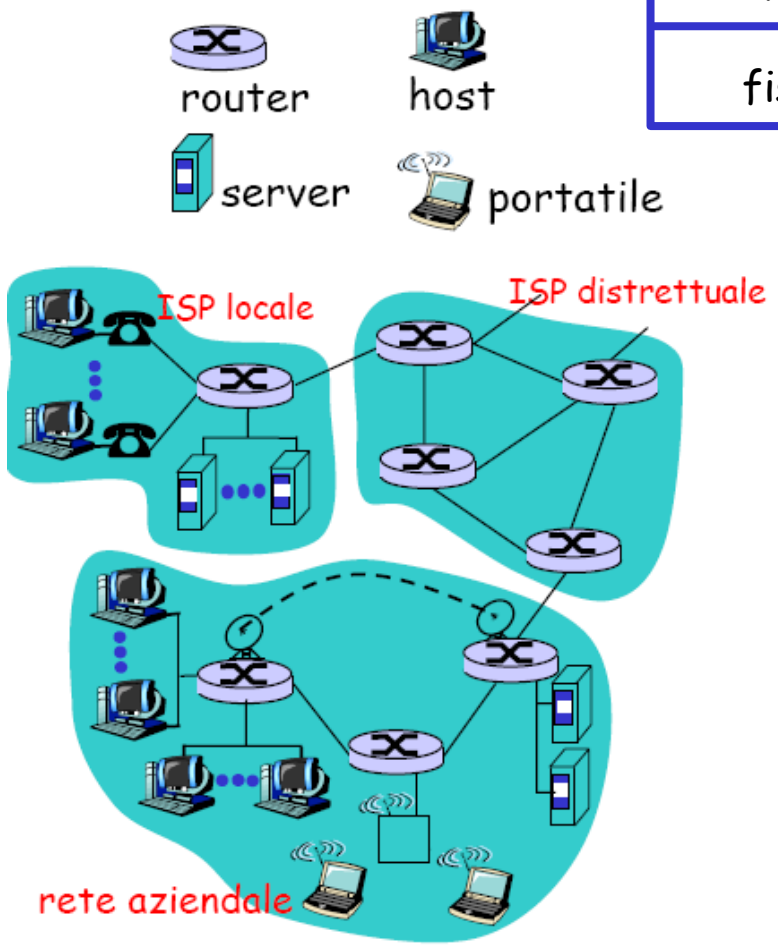
Reti di Calcolatori
AA. 2023-2024

Docente: Federica Paganelli
Dipartimento di Informatica
federica.paganelli@unipi.it

Pila protocollare di Internet

- applicazione: supporta le applicazioni di rete
  - ftp, smtp, http
- trasporto: trasferimento dati end-to-end (tra host terminali)
  - tcp, udp
- rete: instradamento dei datagrammi dalla sorgente alla destinazione
  - ip, protocolli di instradamento
- link: trasferimento dati tra elementi di rete vicini
  - ppp, ethernet
- fisico: bit “sul filo”



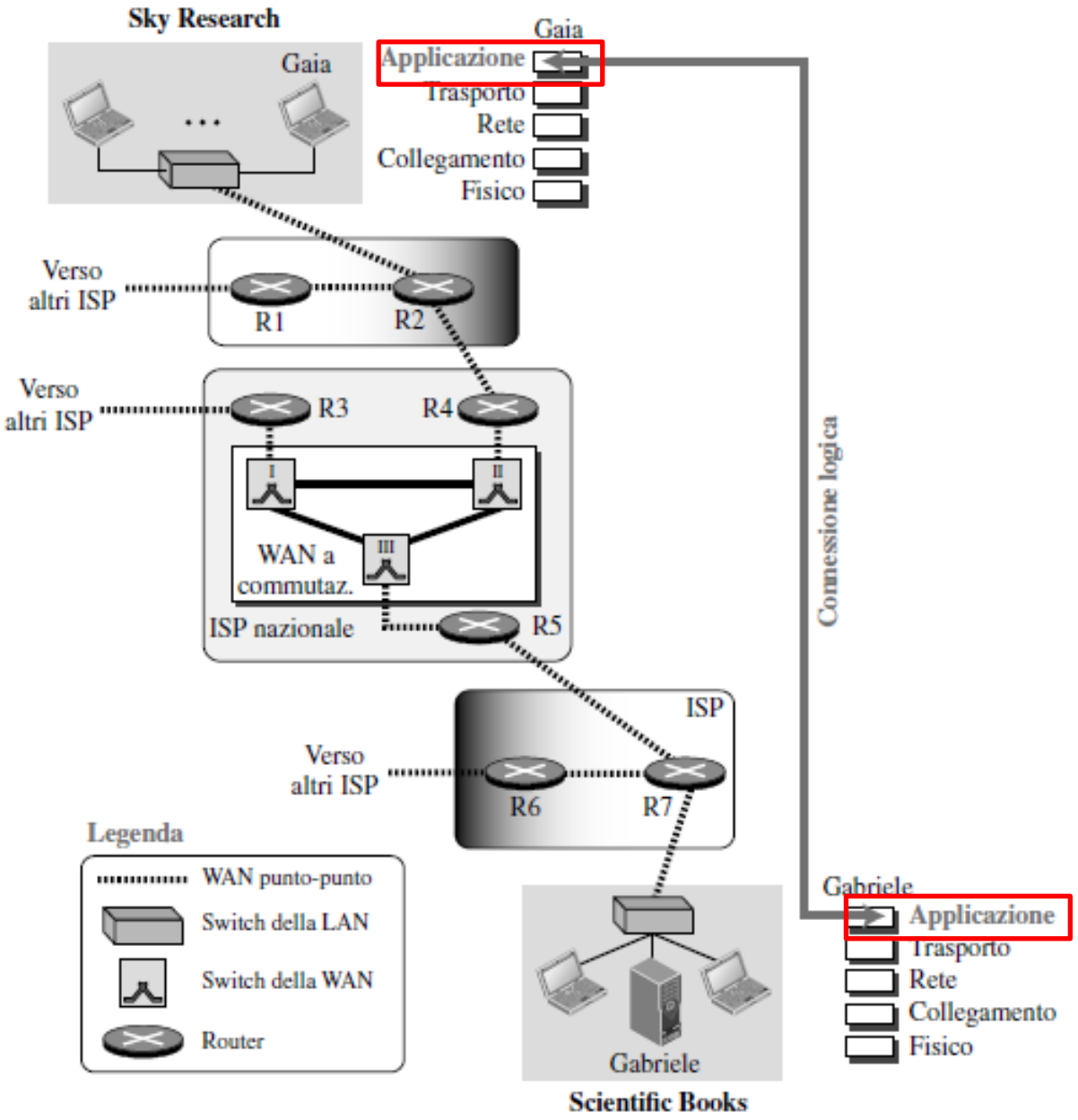


Applicazioni di rete

- Applicazioni formate da processi distribuiti comunicanti
- I processi sono programmi eseguiti dai dispositivi terminali (o host o "end system") di una rete
- All'interno dello stesso host, due processi possono anche comunicare attraverso la comunicazione inter-processo definita dal sistema operativo
- Nella comunicazione a livello applicativo fra due dispositivi terminali diversi di una rete, due o più processi girano su ciascuno degli host comunicanti e si scambiano **messaggi**.

Livello applicazione

- I livelli applicazione nei due lati della comunicazione agiscono come se esistesse un collegamento diretto attraverso il quale poter inviare e ricevere messaggi



Protocollo dello Strato di Applicazione

- Definisce i tipi di messaggi scambiati a livello applicativo (es: di richiesta e di risposta)
- la sintassi dei vari tipi di messaggio (i campi del messaggio)
- la semantica dei campi (significato)
- le regole per determinare quando e come un processo invia messaggi o risponde ai messaggi

Paradigmi del livello applicazione

- Programmi applicativi su host diversi che comunicano tra di loro scambiandosi messaggi
- Es. WEB, gestione di un elaboratore remoto, trasferimento e condivisione file, posta elettronica, comunicazione multimediale

- I due programmi applicativi devono essere entrambi in grado di richiedere e offrire servizi, oppure ciascuno deve occuparsi di uno dei due compiti?

- Paradigmi
  - Client-server:
    - numero limitato di processi server che offrono un servizio e sono in esecuzione, in attesa di ricevere richieste
    - Client: programma che richiede un servizio
  - Peer-to-peer: peer che possono offrire servizi e inviare richieste
  - Misto

Il paradigma client-server

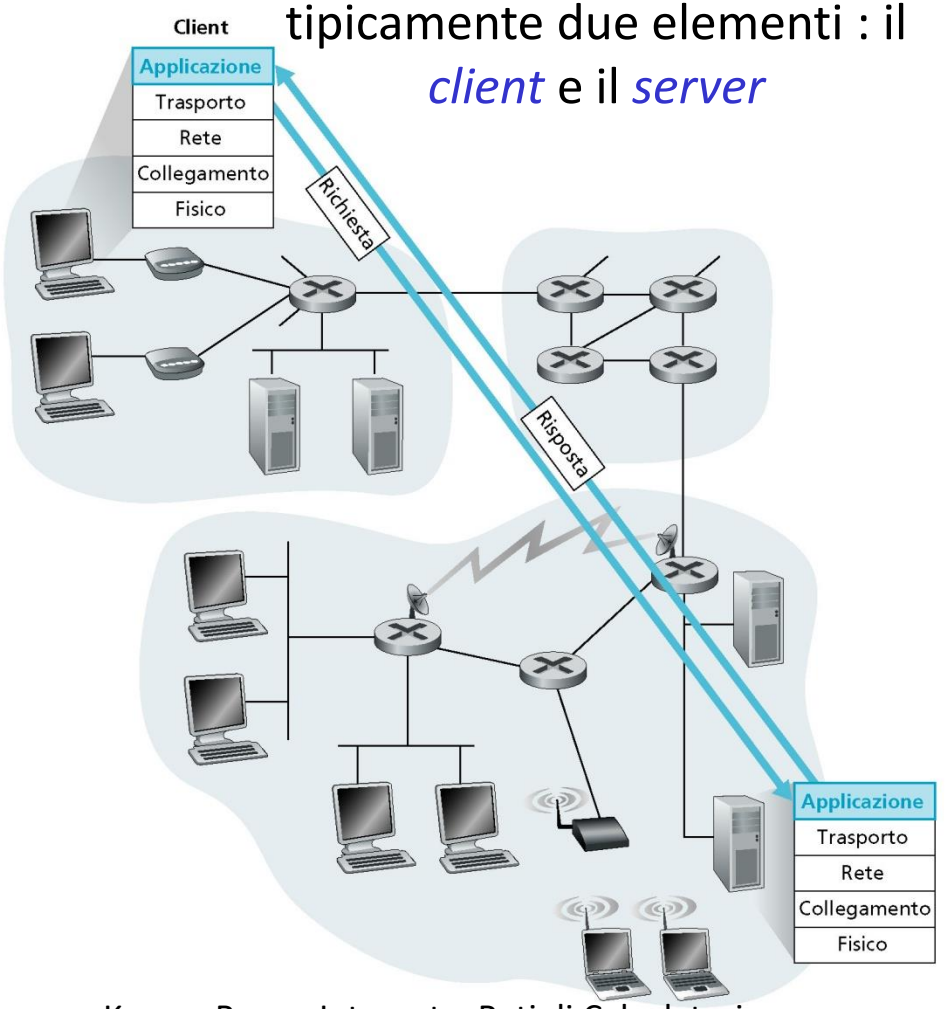
Client:

- inizia il contatto con il server (“parla per primo”)
- tipicamente richiede un servizio al server
- es.: per il Web, il client è implementato nel browser; per l’e-mail, nel mail reader

Server:

- fornisce al client il servizio richiesto
- Sempre attivo
- es.: i Web server inviano le pagine Web richieste, il mail server invia le e-mail

Le applicazioni di rete hanno tipicamente due elementi : il *client* e il *server*



Kurose Ross – Internet e Reti di Calcolatori

Componenti di un'applicazione di rete: due esempi

Web

- Browser sul client
- Server Web
- Standard per il formato dei documenti (risorse)
- Protocollo HTTP

Posta elettronica

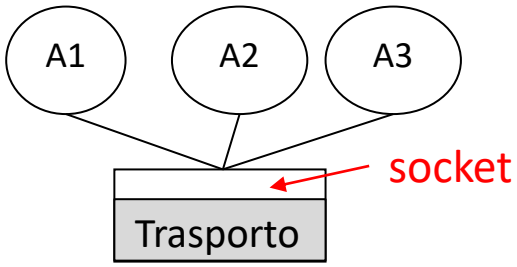
- Standard per il formato dei messaggi
- Programmi di lettura/scrittura sul client
- Server di posta di Internet
- Protocolli SMTP, POP3, ecc.

N.B il protocollo a livello applicativo è solo una parte dell'applicazione di rete!

Applicazioni di rete: terminologia

API: application programming interface

- Insieme di regole che un programmatore deve rispettare per utilizzare delle risorse.



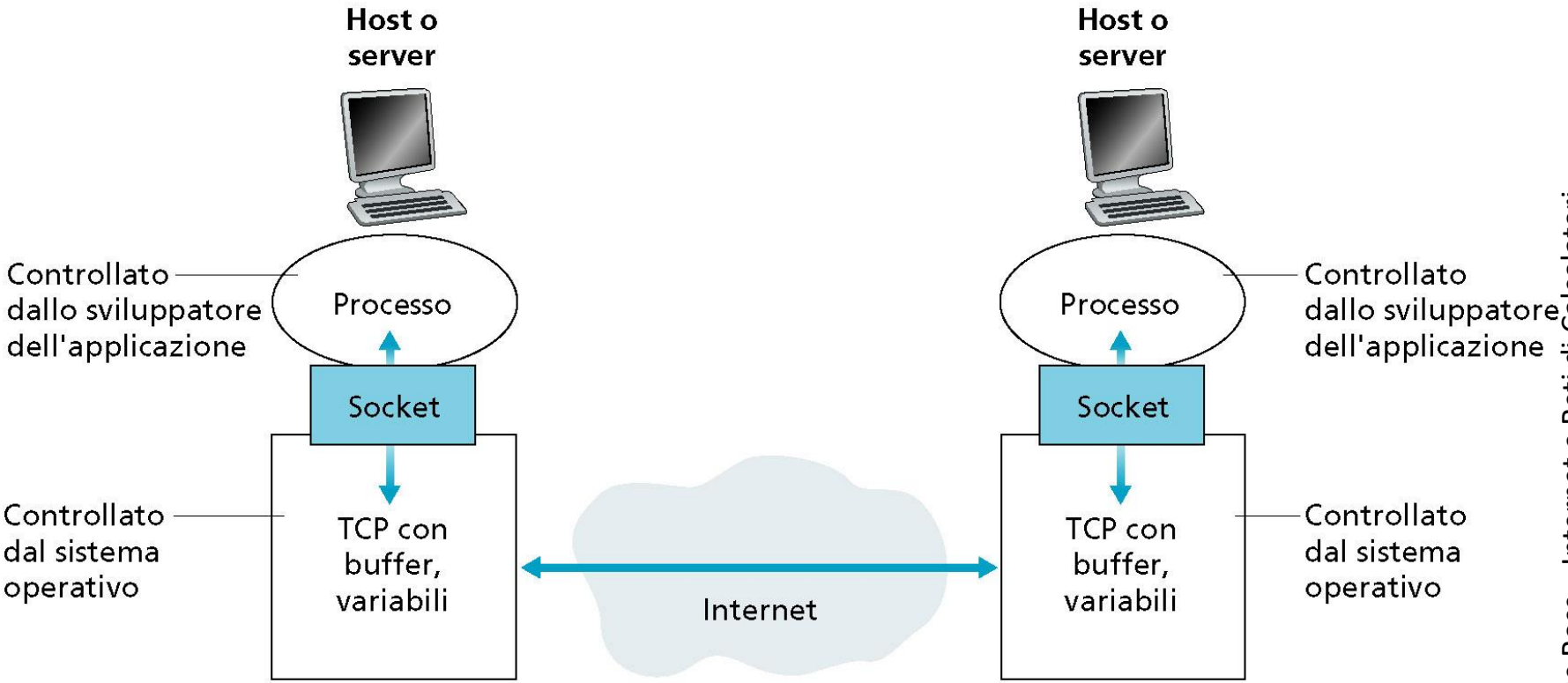
- Un processo a livello applicazione vuole inviare un messaggio al suo omologo su un altro host
- Interagisce con il sistema operativo che implementa i sottostanti livelli dello stack TCP/IP attraverso un'interfaccia

Interfaccia Socket:

API che funge da interfaccia tra gli strati di applicazione e di trasporto

Socket è la API di Internet per eccellenza

Interfaccia socket

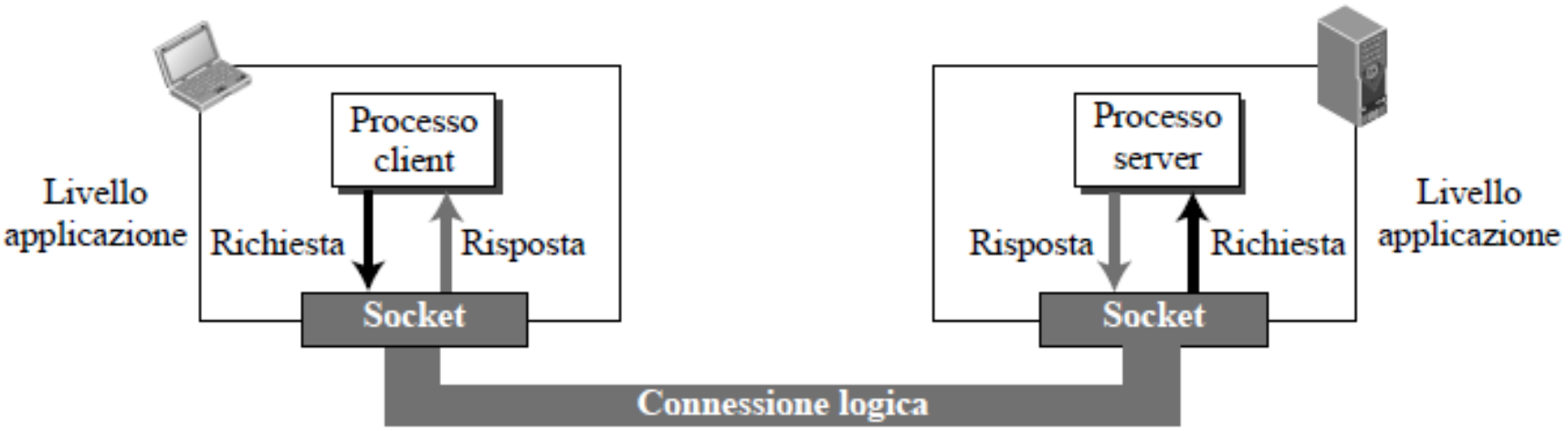


Due processi comunicano mandando dati alla socket, e leggendoli da questa

- Connessione logica tra i due processi, l'invio/ricezione dei dati è responsabilità dei restanti quattro livelli dello stack TCP/IP nel sistema operativo

Interfaccia socket

- Socket è un'astrazione. Si tratta di una struttura dati utilizzata dal programma applicativo.
- La comunicazione tra un processo client e server è realizzata attraverso la comunicazione tra le due socket (**endpoint della comunicazione, socket = connettore, presa**)
- Il client considera la socket come l'entità che riceve le richieste e fornisce le risposte, per il server la socket è l'entità da cui riceve le richieste e a cui inviare le risposte



Indirizzare processi

- Per ricevere messaggi, un processo deve avere un identificatore
- La macchina host ha un indirizzo IP di 32 bit
- *Q:* l'indirizzo IP dell'host su cui il processo è in esecuzione è sufficiente per identificare il processo?

Indirizzare processi

- Per ricevere messaggi, un processo deve avere un identificatore
- La macchina host ha un indirizzo IP di 32 bit
- *Q:* l'indirizzo IP dell'host su cui il processo è in esecuzione è sufficiente per identificare il processo?

- *L'identificativo* include l'indirizzo IP e numero di porta associate al processo.
- Esempi di numeri di porta:
  - HTTP server: 80
  - mail server: 25
- Per inviare messaggi HTTP a al server web www.example.com:
  - IP address: 128.119.245.12

Socket Address

Esempio di API: TCP

connection **TCPopen** (IPaddress, int) //per aprire una connessione
void **TCPsend** (connection, data) //per spedire dati su una connessione
data **TCPreceive** (connection) //per ricevere dati su una connessione
void **TCPclose** (connection) //per chiudere una connessione

int **TCPbind** (int) //per rich. assegnaz. porta su cui attendere rich. di conn.
void **TCPunbind** (int) //per liberare una porta
connection **TCPaccept** (int) //per attendere richieste di connessione

Note: (1) connection identificata da una quadrupla
(2) astraiamo da possibili eccezioni sollevate (e da loro trattamento)

Uso dei servizi di trasporto

- Una coppia di processi fornisce servizi agli utenti di Internet, siano questi persone o applicazioni.
- La coppia di processi, tuttavia, deve utilizzare i servizi offerti dal livello trasporto per la comunicazione, poiché non vi è una comunicazione fisica a livello applicazione.

- Nel livello trasporto della pila di protocolli TCP/IP sono previsti due protocolli principali
  - TCP: servizio connection-oriented, basato su stream
  - UDP: servizio connection-less, orientato al messaggio

Concetti generali

Applicazioni di rete realizzate “sopra” servizi Internet di trasporto dati

*Servizio **TCP***

- Connection-oriented: setup richiesto tra client e server
- Trasporto affidabile tra processo mittente e destinatario
- Controllo del flusso: il mittente non “inonderà” di dati il destinatario
- Controllo di congestione: “strangola” il mittente quando la rete è sovraccarica
- Non offre garanzie di timing né di ampiezza minima di banda

*Servizio **UDP***

- Non orientato alle connessioni (connection-less)
- Trasporto NON affidabile
- NO controllo di flusso
- NO controllo congestione
- No garanzie timing né ampiezza minima di banda

Ma allora quali applicazioni usano UDP? (e perché?)

Che tipo di trasporto è richiesto da un'applicazione?

Throughput

- tasso al quale il processo mittente può inviare i bit al processo ricevente
- alcune applicazioni (es. multimedia) richiedono un certo throughput per essere efficaci
- altre apps ("elastic apps") possono far uso di poco o tanto throughput (a seconda di quanto ne trovano a disposizione)

Perdita dei dati

- alcune applicazioni (es., audio) possono tollerare alcune perdite
- altre applicazioni richiedono un trasferimento dati affidabile al 100%

Sensibilità ai ritardi

- alcune applicazioni (es., teleconferenza, giochi interattivi) richiedono un basso ritardo per essere efficaci

Trasporto richiesto da applicazioni comuni

applicazione	Tolleranza alla Perdita di dati	throughput	Sensibilità al tempo
file transfer	no	elastico	no
e-mail	no	elastico	no
Documenti Web	no	elastico	no
Audio/video real time	si	audio: 5kbps-1Mbps video:10kbps-5Mbps	Si, decine di msec
audio/video memorizzato	si	come sopra	Si, pochi secondi
Giochi interattivi	si	Pochi kbps	Si, decine di msec
Messaggistica istantanea	no	elastico	Si e no

Applicazioni Internet

applicazione	protocollo di livello applicazione	Protocollo di trasporto
e-mail	SMTP [RFC 2821]	TCP
remote terminal access	Telnet [RFC 854]	TCP
Web	HTTP [RFC 2616]	TCP
file transfer	FTP [RFC 959]	TCP
streaming multimedia	HTTP (e.g., YouTube), RTP [RFC 1889]	TCP or UDP
Internet telephony	SIP, RTP, proprietary (e.g., Skype)	TCP or UDP

Applicazione

Web e HTTP

Cos'è il World Wide Web?

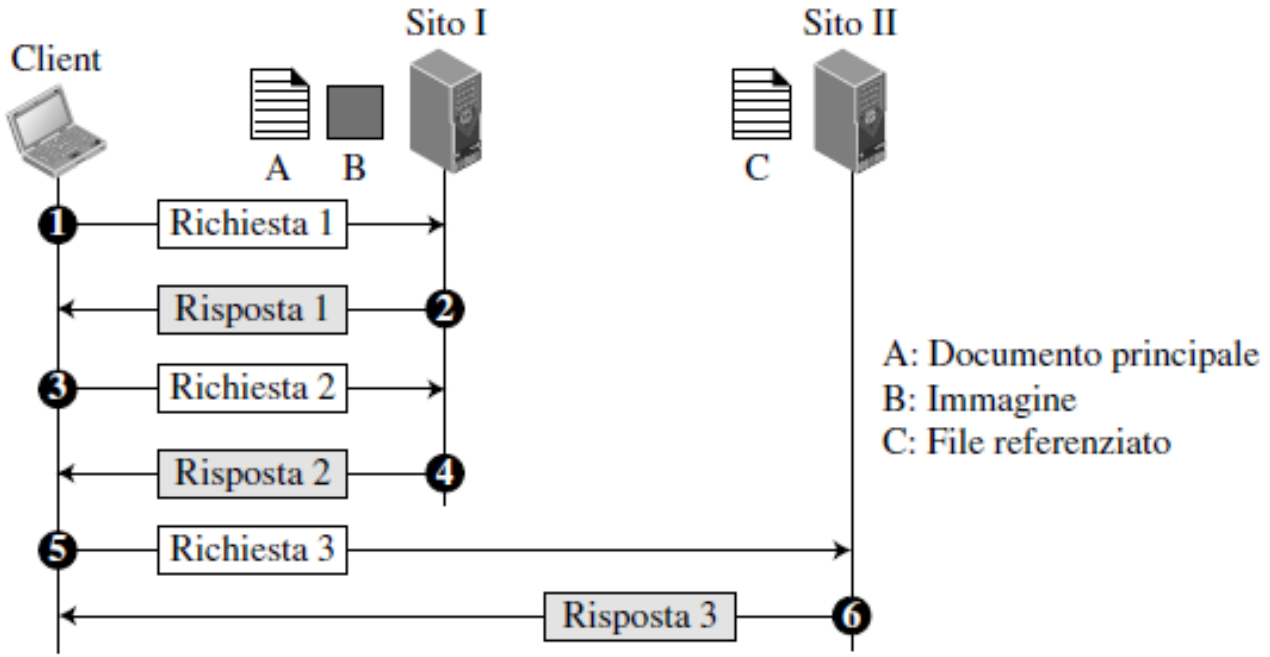
- Web = Ragnatela
- Ad oggi enorme collezione di informazioni nella quale le risorse sono distribuite e collegate l'una all'altra
- Qualsiasi server web nel mondo può aggiungere risorse
- Hypertext (o meglio hypermedia): una risorsa (es. pagina testuale) può contenere riferimenti a altre risorse (documenti testuali, immagini, risorse audio/video…)

Nascita e evoluzione del Web

- Fino agli anni ‘90 Internet usata principalmente per il servizio di posta elettronica e news in comunità accademiche e di ricerca
- Il World Wide Web viene ideato da Tim Berners Lee (CERN) per consentire ai ricercatori di accedere alle pubblicazioni scientifiche (tra il 1989 e 1991 ideazione e primo sito web)
- Varie evoluzioni (Web 1.0, Web 2.0, Web 3.0, …) fino a piattaforma di riferimento per servizi di e-commerce, news, social network, email, telefonia, videoconferenza,…

Il Web: terminologia

- Pagina Web: consiste di “oggetti” formati da:
  - Un file HTML di base
  - diversi oggetti referenziati (altre pagine, immagine JPEG, file audio)
  - Ciascun oggetto è indirizzabile tramite una URL (Uniform Resource Locator)



URI, URL E URN

Uniform Resource Identifier (URI)

- Una URI è una forma generale per identificare una risorsa presente sulla rete (vedi IETF rfc 2396)
  - *A Uniform Resource Identifier (URI) is a compact string of characters for identifying an abstract or physical resource.*

- Uniform: uniformità della sintassi dell'identificatore anche se i meccanismi per accedere alle risorse possono variare
- Resource: qualsiasi cosa abbia un'identità (!)
  - Documento, servizio, immagine, collezione di altre risorse
- Identifier: informazioni che permettono di distinguere un oggetto da altri oggetti (entro un ambito definito di identificazione)

Uniform Resource Identifier (URI)

- Ci sono due tipi di URI:
- Uniform Resource Locator (URL): sottoinsieme di URI che identifica le risorse attraverso il loro meccanismo di accesso (e.g., protocollo HTTP)
  - https://doi.org/10.1109/LCN.1988.10239
  - ftp://ftp.is.co.za/rfc/rfc1808.txt
  - http://www.apple.com/index.html

- Uniform Resource Name (URN): sottoinsieme di URI che devono rimanere globalmente unici e persistente anche quando la risorsa cessa di esistere e diventa non disponibile
  - urn:oasis:names:specification:docbook:dtd:xml:4.1.2:
  - urn:doi:10.1109/LCN.1988.10239

URL

- Sintassi URL (RFC1738) – Internet Syntax

<scheme>://<user>:<password>@<host>:<port>/<path>

  - <user> e <password> opzionali e generalmente deprecato
  - scheme: protocollo di accesso alla risorsa
  - host = nome di dominio di un host o indirizzo IP (in notazione decimale puntata),
  - port = numero di porta del server. Molti protocolli hanno porte di default (es. 80 per http)
  - Path: contiene dati specifici per l'host (o scheme) e identifica la risorsa nel contesto di quello schema e host. Può consistere in una sequenza di segmenti separati da "/". Ad esempio path nel file system del server ma non solo! Il path specifica la Request-URI*

* lo vedremo nei messaggi HTTP

HTTP URL

- HTTP URL

  http://<host>:<port>/<path>?<query>

  – <query> stringa di informazioni che deve essere interpretata dal server

  http://www.esempio.com/scarpe?ordina=prezzo

URL Assolute e Relative

- Le URL possono essere assolute o relative
- URL assoluta: identifica una risorsa indipendentemente dal contesto in cui è usata
- URL relativa: informazioni per identificare una risorsa in relazione ad un'altra URL (è priva dello schema e della authority).

URL Relativa

Sia http://a/b/c/d;p?q il documento di partenza, allora:

```
g       =   http://a/b/c/g
/g      =   http://a/g
//g     =   http://g
?y      =   http://a/b/c/?y
#s      =   (current document)#s
g;x?y#s       =   http://a/b/c/g;x?y#s
..      =   http://a/b/
../../g       =   http://a/g
```

https://www.ietf.org/rfc/rfc3986.txt

Le URL Relative…

- NON viaggiano sulla rete
- Sono interpretate dal browser in relazione al documento di partenza

Lo strato Applicativo
HTTP

Reti di Calcolatori
AA. 2022-2023

Docente: Federica Paganelli
Dipartimento di Informatica
federica.paganelli@unipi.it

HYPERTEXT TRANSFER PROTOCOL

HTTP: HyperText Transfer Protocol

- Usato dal 1990 come protocollo di trasferimento per il World Wide Web
- protocollo di livello applicazione per sistemi di informazione distribuiti, collaborativi ed ipermediali (RFC 2616)

Dall'abstract in RFC 2616 (https://tools.ietf.org/html/rfc2616)

It is a generic, stateless, protocol which can be used for many tasks beyond its use for hypertext, such as name servers and distributed object management systems, through extension of its request methods, error codes and headers.

A feature of HTTP is the typing and negotiation of data representation, allowing systems to be built independently of the data being transferred.

- Non limitato a ipertesto
- I metodi possono essere estesi
- Protocollo senza stato
- Tipizzazione e negoziazione dei formati di rappresentazione dei dati

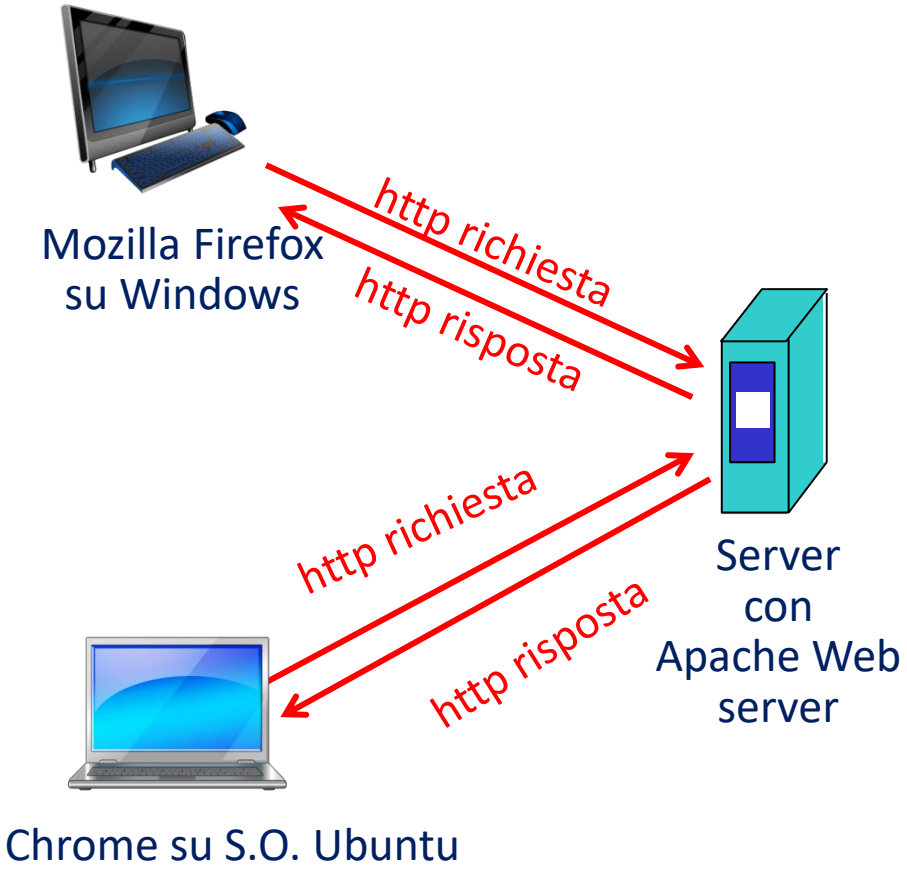
Il protocollo http: client/server

Client HTTP

- programma che stabilisce una connessione con un server HTTP e invia una richiesta per effettuare operazioni su risorse web
- Es. browser che richiede, riceve e visualizza gli oggetti Web

Server HTTP

- programma che accetta connessioni per servire richieste inviando messaggi di risposta
- Es. Server web che ospita risorse Web, indirizzabili tramite URL



Il protocollo http

- E' un protocollo di tipo richiesta/risposta
  - La connessione viene iniziata dal client, che invia un messaggio di richiesta (request).
  - Il server risponde con un messaggio di risposta (response)
- Protocollo stateless (senza memoria di stato)
  - le coppie richiesta/risposta sono indipendenti
  - ogni richiesta viene eseguita indipendentemente da quelle che l'hanno preceduta

Quale servizio di trasporto usa? TCP o UDP?

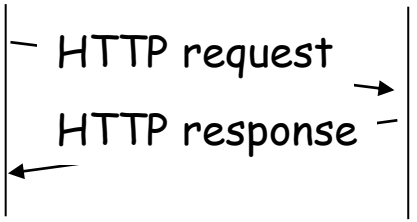
Il protocollo http

- E' un protocollo di tipo richiesta/risposta
  - La connessione viene iniziata dal client, che invia un messaggio di richiesta (request).
  - Il server risponde con un messaggio di risposta (response)
- Protocollo stateless (senza memoria di stato)
  - le coppie richiesta/risposta sono indipendenti
  - ogni richiesta viene eseguita indipendentemente da quelle che l'hanno preceduta
- Usa TCP (connection-oriented)
  - Il client richiede l'instaurazione di una connessione TCP col server
  - I processi si scambiano messaggi attraverso le rispettive socket

HTTP

– tipica interazione HTTP client-server:

– **HTTP utilizza TCP**




```
//esempio client
c = TCPopen("131.115.7.24", 80);
TCPsend(c,"GET /index.html");
d = TCPreceive(c);
```

```
//esempio server
p = TCPbind(80);
d = TCPaccept(p);
 r = TCPreceive(d)
        ...
TCPsend(d,pag)
TCPclose(d)
```

Connessione persistente e non persistente

- *Connessione*: Un circuito logico di livello trasporto stabilito tra due programmi applicativi per comunicare tra loro.
- *Connessione non persistente* (http1.0: RFC 1945): viene stabilita una connessione TCP separata per recuperare ciascuna URL
- *Connessione persistente* (http1.1: RFC 2616): se non diversamente indicato, il client può assumere che il server manterrà una connessione persistente
  - Lo standard specifica un meccanismo con cui client e server possono indicare la chiusura della connessione TCP (Connection header field).
  - Dopo la chiusura, il client non deve più inviare richieste su quella connessione.

Esempio: trasferimento di una pagina web

Supponiamo che l'utente digiti la URL

www.someSchool.edu/someDepartment/home.index

(file html contenente testo e riferimenti a 10 immagini jpg)

1a. Il client http inizia la connessione TCP verso il server http al www.someSchool.edu. Socket Porta 80 è default per il server http.

1b. Il server http dell'host www.someSchool.edu aspetta le richieste di connessione TCP connection alla porta 80. "Accetta" la connessione, e lo notifica al client

2. Il client http invia un *messaggio di richiesta http* (che contiene la URL) alla socket di connessione con lo strato di trasporto (TCP)

3. Il server http riceve il msg di richiesta, compila un *messaggio di risposta* che contiene l'oggetto richiesto (someDepartment/home.index), manda il messaggio alla socket

tempo

Esempio http (cont.)

4. Il server http chiude la connessione TCP

5. Il client http riceve il messaggio di risposta che contiene il file html e lo visualizza. Percorrendo il file trova il riferimento a 10 oggetti jpg

tempo

6. Ripete i passaggi da 1-5 per ognuno dei 10 oggetti jpg

Ipotesi: connessione non persistente

Il protocollo http è "stateless"

ogni risposta è collegata solo alla richiesta che l'ha generata, indipendentemente dalle richieste precedenti

Connessione persistente (HTTP 1.1)

- La stessa connessione HTTP può essere utilizzata per una serie di richieste e una serie corrispondente di risposte
- Il server lascia aperta la connessione TCP dopo aver spedito la risposta e può quindi ricevere le richieste successive sulla stessa connessione
- Il server HTTP chiude la connessione quando viene specificato nell'header del messaggio (desiderata da parte del cliente) oppure quando non riceve richieste per un certo intervallo di tempo (time out)

Come cambia il diagramma il flusso richieste/risposte nella slide precedente?

Domanda

- Quali sono i vantaggi della connessione persistente introdotta nella versione 1.1 ?
- Suggerimento: cerca nelle RFC…