

# *Programma del corso*

Introduzione ai protocolli di rete

http

ftp

dns

smtp/pop/imap

# Capitolo 1

## Overview

### Nota per l'utilizzo:

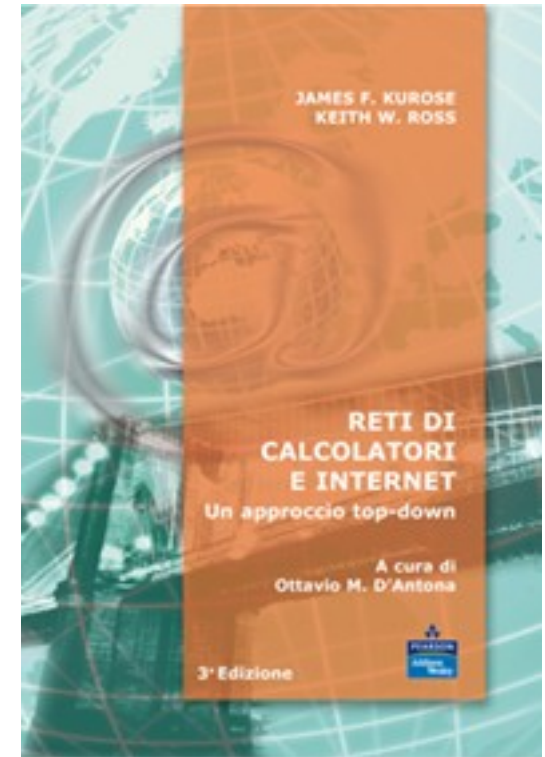
Abbiamo preparato queste slide con l'intenzione di renderle disponibili a tutti (professori, studenti, lettori). Sono in formato PowerPoint in modo che voi possiate aggiungere e cancellare slide (compresa questa) o modificarne il contenuto in base alle vostre esigenze.

Come potete facilmente immaginare, da parte nostra abbiamo fatto *un* sacco di lavoro. In cambio, vi chiediamo solo di rispettare le seguenti condizioni:

- se utilizzate queste slide (ad esempio, in aula) in una forma sostanzialmente inalterata, fate riferimento alla fonte (dopo tutto, ci piacerebbe che la gente usasse il nostro libro!)
- se rendete disponibili queste slide in una forma sostanzialmente inalterata su un sito web, indicate che si tratta di un adattamento (o che sono identiche) delle nostre slide, e inserite la nota relativa al copyright.

*Thanks and enjoy!* JFK/KWR

All material copyright 1996-2005  
J.F Kurose and K.W. Ross, All Rights Reserved



*Reti di calcolatori e Internet: Un  
approccio top-down*

*3ª edizione*

*Jim Kurose, Keith Ross*

*Pearson Education Italia ©2005*

# Capitolo 1: Introduzione

## Obiettivi:

- *introdurre la terminologia e i concetti di base*
- *approccio:*
  - ❖ *usare Internet come fonte di esempi*

## Panoramica:

- *cos'è Internet*
- *cos'è un protocollo?*
- *ai confini della rete*
- *il nucleo della rete*
- *accesso alla rete, mezzi trasmissivi*
- *ISP e dorsali internet*
- *prestazioni: ritardi e perdite*
- *livelli di protocollo, modelli di servizio*
- *modellazione di rete*

# Capitolo 1: roadmap

*1.1 Cos'è Internet?*

*1.2 Ai confini della rete*

*1.3 Il nucleo della rete*

*1.4 Accesso alla rete e mezzi trasmissivi*

*1.5 ISP e dorsali Internet*

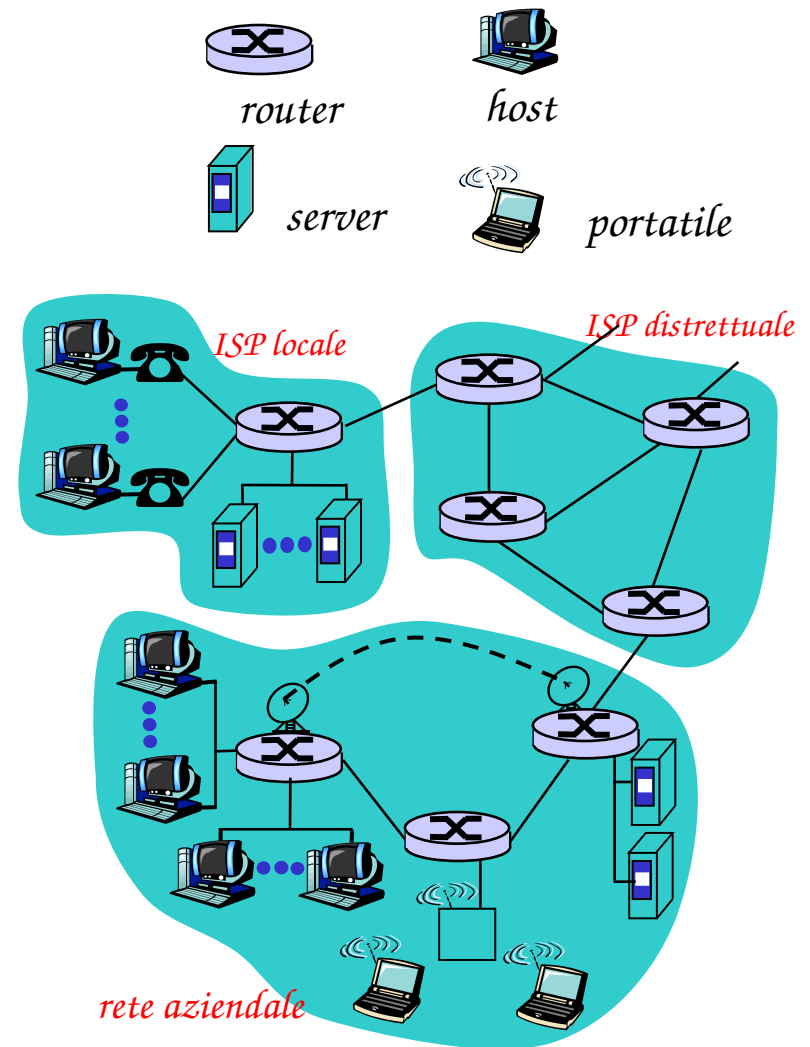
*1.6 Ritardi e perdite nelle reti a commutazione di pacchetto*

*1.7 Livelli di protocollo e loro modelli di servizio*

*1.8 Storia del computer networking e di Internet*

# Che cos'è Internet?

- ❑ *Milioni di dispositivi collegati: **host** = sistema terminale*
- ❑ *applicazioni di rete*
- ❑ *collegamenti*
  - ❖ *rame, fibra ottica, onde elettromagnetiche, satellite*
  - ❖ *Frequenza di trasmissione = **ampiezza di banda***
- ❑ ***router**: instrada i pacchetti verso la loro destinazione finale*



# Oggi Internet è anche...

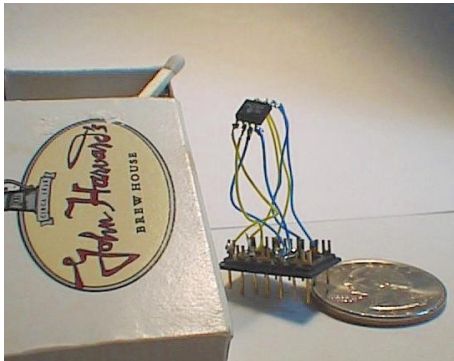


Cornice IP

<http://www.ceiva.com/>



Tostapane Web +  
previsioni del tempo



Il web server più piccolo del mondo

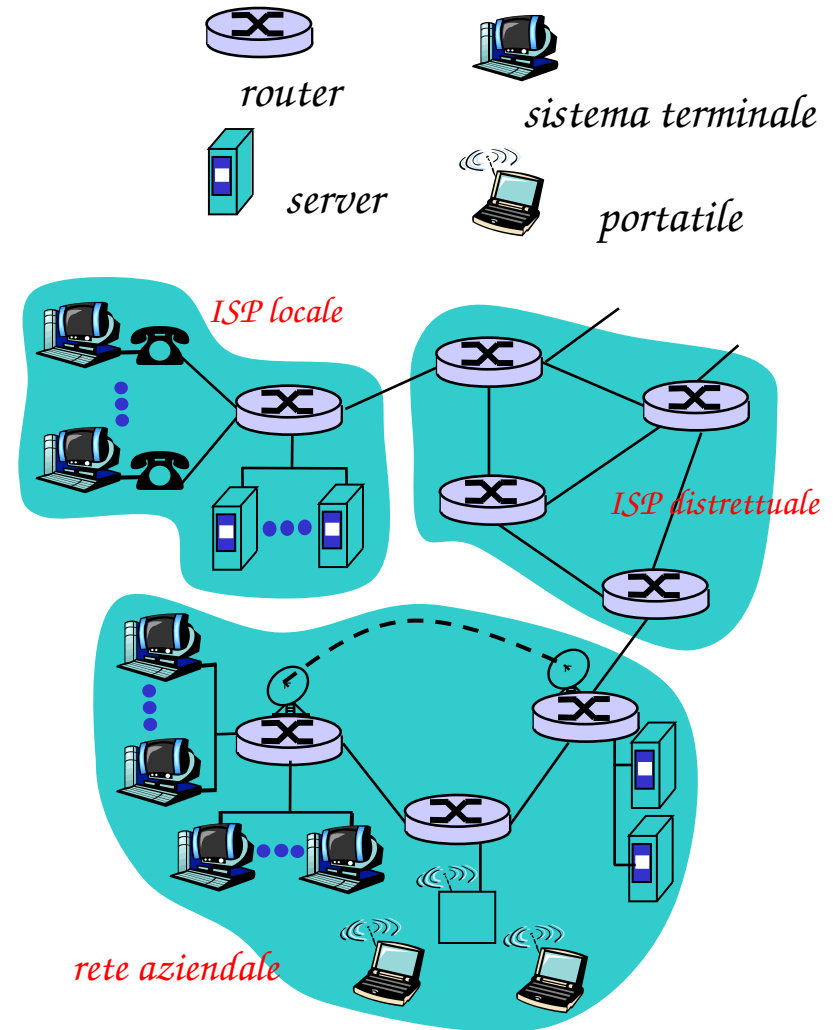
<http://www-ccs.cs.umass.edu/~shri/iPic.html>



Telefonia Internet

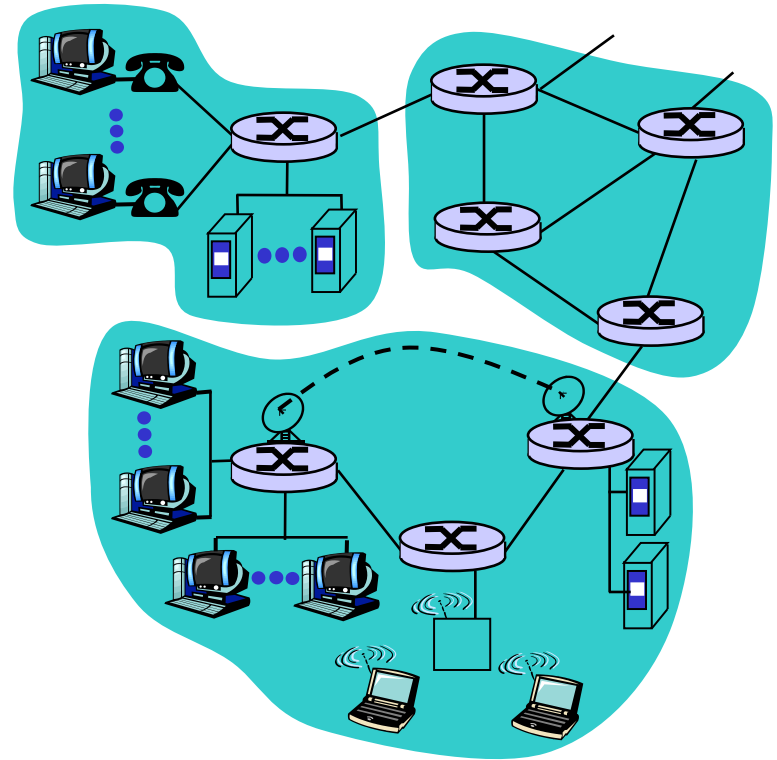
# Che cos'è Internet

- ❑ Un **protocollo** definisce il formato e l'ordine dei messaggi scambiati fra due o più entità in comunicazione
  - ❖ es.: TCP, IP, HTTP, FTP, PPP
- ❑ **Internet: "rete delle reti"**
  - ❖ struttura gerarchica
  - ❖ Internet pubblica e intranet private
- ❑ **Standard Internet**
  - ❖ RFC: Request for comments
  - ❖ IETF: Internet Engineering Task Force



# Cos'è Internet

- ❑ *Infrastruttura di comunicazione per applicazioni distribuite:*
  - ❖ *Web, e-mail, giochi, e-commerce, condivisione di file*
- ❑ *Servizi forniti alle applicazioni:*
  - ❖ *Servizio non affidabile senza connessione*
  - ❖ *servizio affidabile orientato alla connessione*





# Cos'è un protocollo?

## Protocolli umani:

- ❑ *“Che ore sono?”*
- ❑ *“Ho una domanda”*
- ❑ *Presentazioni*

*... invio di specifici messaggi*

*... quando il messaggio è ricevuto,  
vengono intraprese specifiche  
azioni, o si verificano altri eventi*

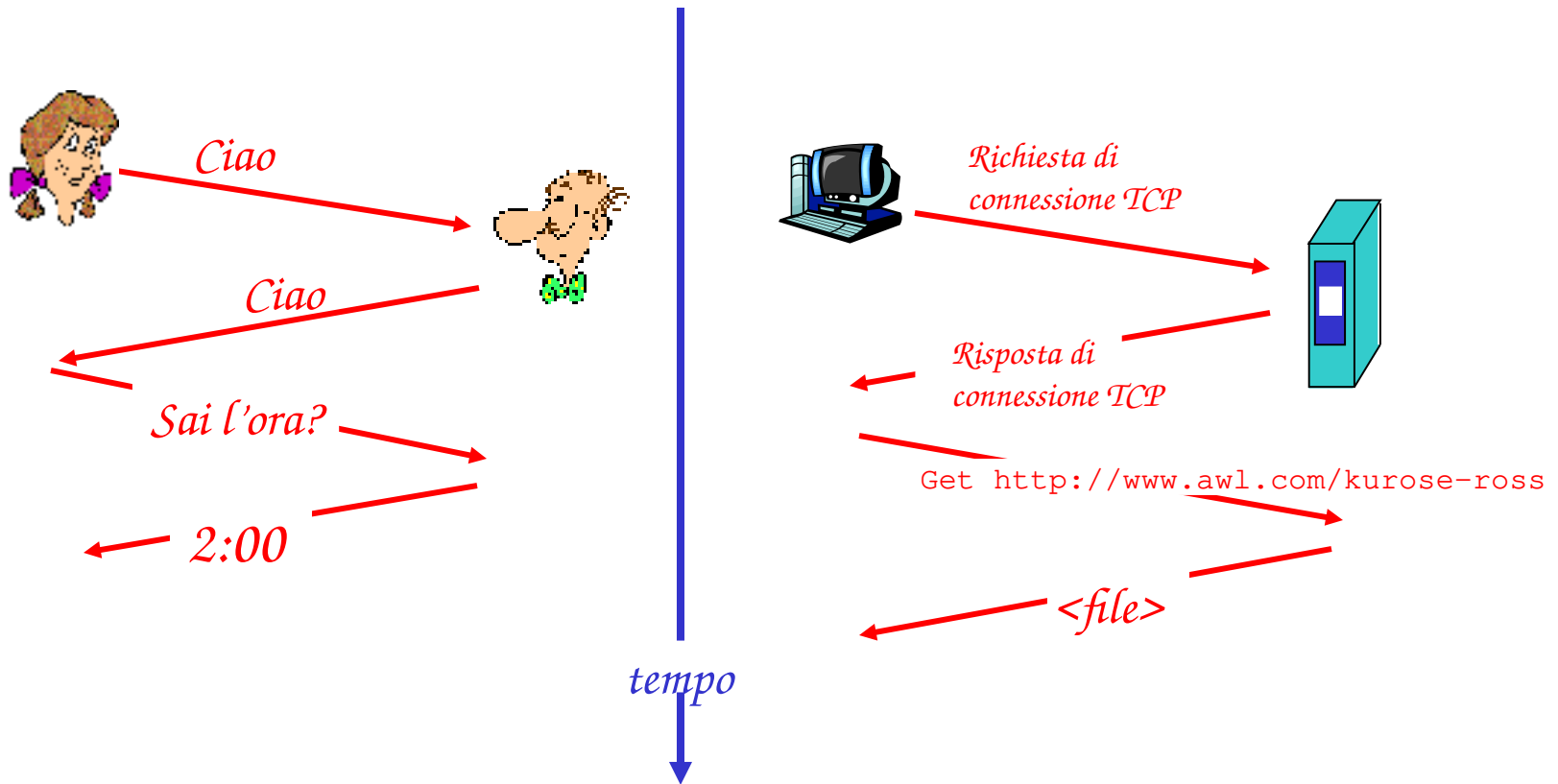
## Protocolli di rete:

- ❑ *Dispositivi hardware e software, non umani*
- ❑ *Tutta l'attività di comunicazione in Internet è governata dai protocolli*

*Un protocollo definisce il formato e l'ordine dei messaggi scambiati tra due o più entità in comunicazione, così come le azioni intraprese in fase di trasmissione e/o ricezione di un messaggio o di un altro evento*

# Cos'è un protocollo?

## *Protocollo umano e protocollo di rete*



**D:** Conoscete altri protocolli umani?

# Capitolo 1: roadmap

*1.1 Cos'è Internet?*

*1.2 Ai confini della rete*

*1.3 Il nucleo della rete*

*1.4 Accesso alla rete e mezzi trasmissivi*

*1.5 ISP e dorsali Internet*

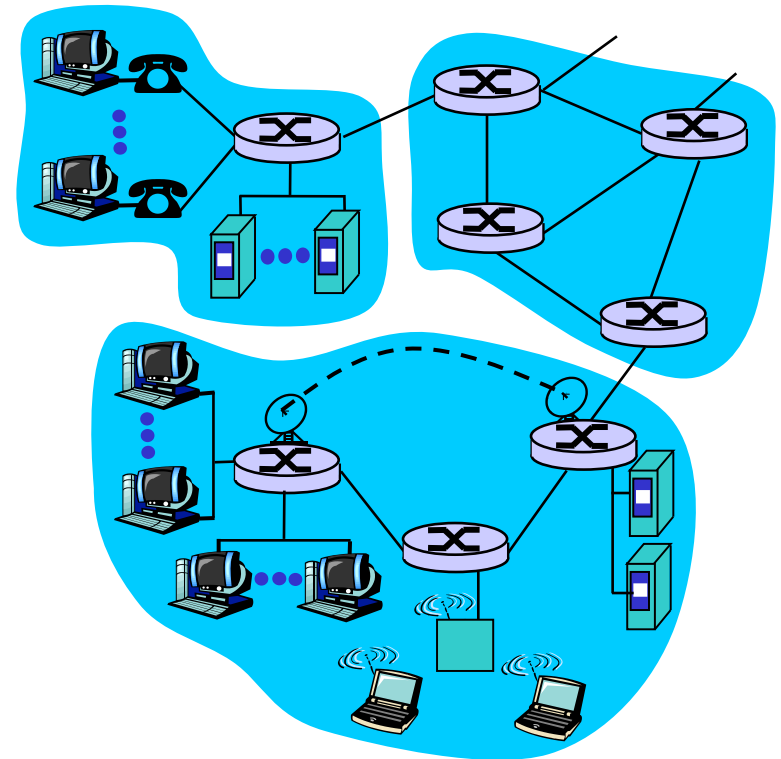
*1.6 Ritardi e perdite nelle reti a commutazione di pacchetto*

*1.7 Livelli di protocollo e loro modelli di servizio*

*1.8 Storia del computer networking e di Internet*

# Uno sguardo da vicino alla struttura di rete

- *ai confini della rete:* applicazioni e sistemi terminali
- *al centro della rete:*
  - ❖ router
  - ❖ la rete delle reti
- *reti, dispositivi fisici:* collegamenti



# Ai confini della rete

## ❑ *sistemi terminali (host)*

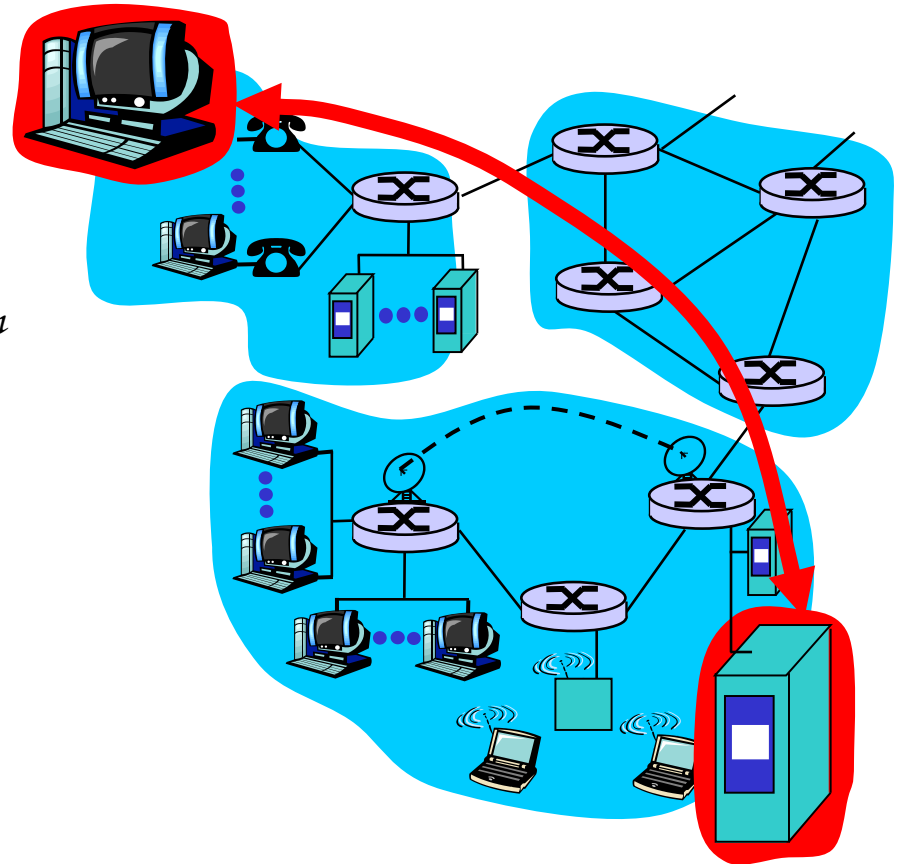
- ❖ *fanno girare programmi applicativi*
- ❖ *es.: Web, e-mail*
- ❖ *situati alle estremità di Internet*

## ❑ *architettura client/server*

- ❖ *L'host client richiede e riceve un servizio da un programma server in esecuzione su un altro terminale*
- ❖ *es.: browser/server Web ; client/server e-mail*

## ❑ *architettura peer to peer*

- ❖ *uso limitato (o inesistente) di server dedicati*
- ❖ *es.: Gnutella, KaZaA, Skype*



## Ai confini della rete: servizio orientato alla connessione

Obiettivo: trasferimento dati tra sistemi terminali

- *handshaking*: messaggi di preparazione all'invio di dati
- *TCP* - Transmission Control Protocol
  - ❖ Il servizio orientato alla connessione di Internet

### Servizio TCP [RFC 793]

- Trasporto affidabile, consegna "in ordine" del flusso di byte
  - ❖ in caso di perdita: ACK e ritrasmissioni
- Controllo del flusso
  - ❖ il mittente non sovraccarica il destinatario
- Controllo di congestione
  - ❖ i mittenti rallentano il tasso di invio quando la rete è congestionata

## Ai confini della rete: servizio senza connessione

Obiettivo: trasferimento dati tra sistemi terminali

❖ Come nel caso precedente!

□ **UDP** - User Datagram Protocol [RFC 768]:

- ❖ senza connessione
- ❖ trasferimento dati non affidabile
- ❖ nessun controllo del flusso
- ❖ nessun controllo di congestione

Applicazioni che usano TCP:

- HTTP (Web), FTP (trasferimento file), Telnet (login remoto), SMTP (e-mail)

Applicazioni che usano UDP:

- streaming multimediale, videoconferenze, DNS, telefonia Internet

# Capitolo 1: roadmap

*1.1 Cos'è Internet?*

*1.2 Ai confini della rete*

*1.3 Il nucleo della rete*

*1.4 Accesso alla rete e mezzi trasmissivi*

*1.5 ISP e dorsali Internet*

*1.6 Ritardi e perdite nelle reti a commutazione di pacchetto*

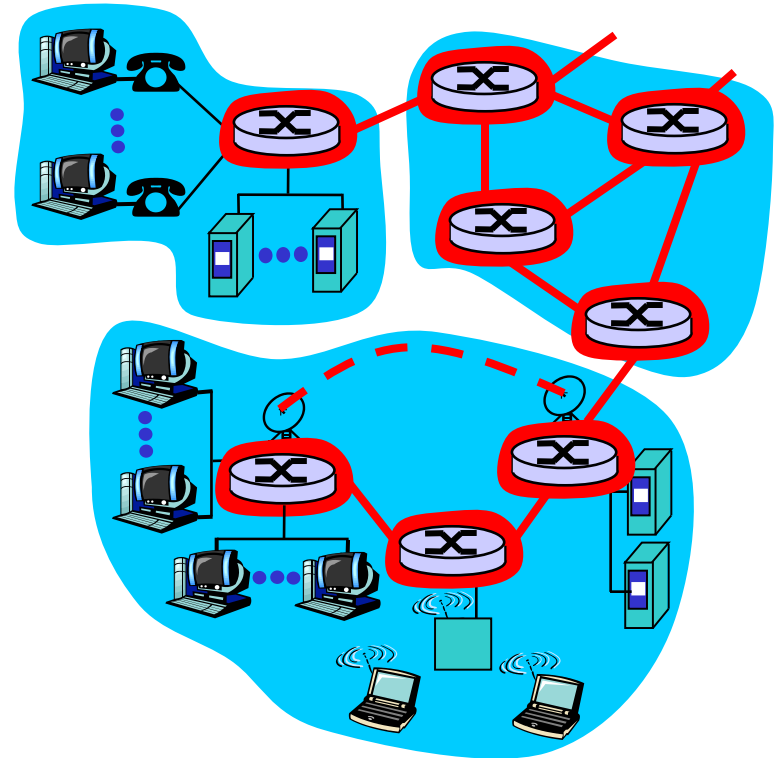
*1.7 Livelli di protocollo e loro modelli di servizio*

*1.8 Storia del computer networking e di Internet*



# Il nucleo della rete

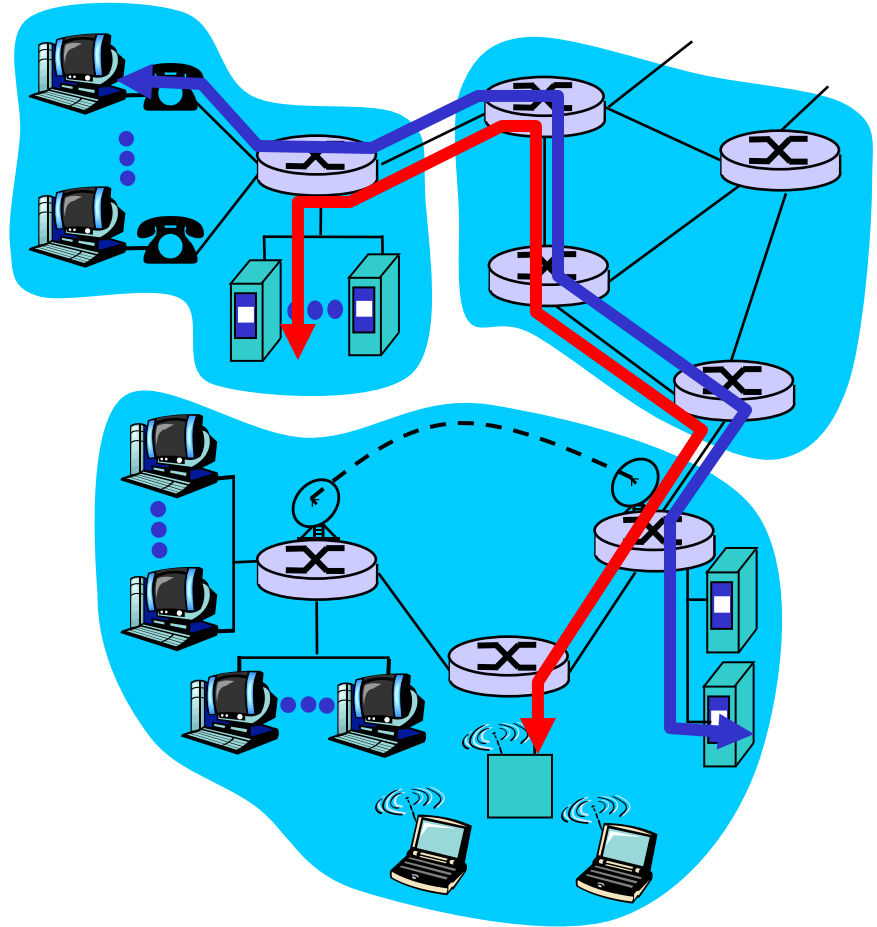
- Rete magliata di router che interconnettono i sistemi terminali
- *il quesito fondamentale:* come vengono trasferiti i dati attraverso la rete?
  - ❖ *commutazione di circuito:* circuito dedicato per l'intera durata della sessione
  - ❖ *commutazione di pacchetto:* i messaggi di una sessione utilizzano le risorse su richiesta, e di conseguenza potrebbero dover attendere per accedere a un collegamento



# Il nucleo della rete: commutazione di circuito

## *connessione punto-punto dedicata*

- ❑ *ciascun commutatore dispone di  $n$  circuiti, in modo da supportare  $n$  connessioni contemporanee*
- ❑ *risorse dedicate: non c'è condivisione*
- ❑ *necessaria l'impostazione della chiamata*



## Il nucleo della rete: commutazione di circuito

Risorse di rete (ad es. larghezza di banda, bandwidth) *suddivise in "pezzi"*

- ciascun "pezzo" viene allocato ai vari collegamenti
- le risorse rimangono *inattive* se non utilizzate (non c'è condivisione)

□ suddivisione della banda in "pezzi"

- ❖ divisione di frequenza
- ❖ divisione di tempo

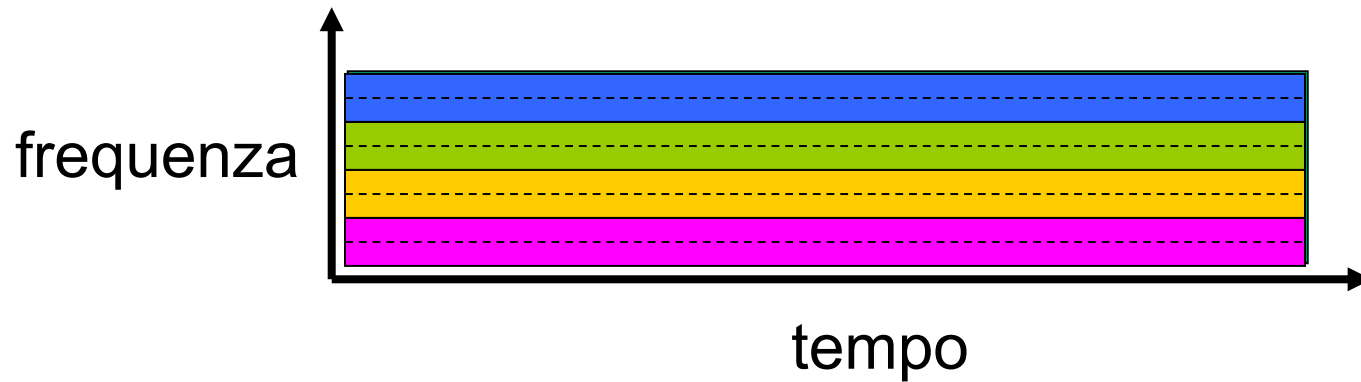
# Commutazione di circuito: FDM e TDM

Esempio:

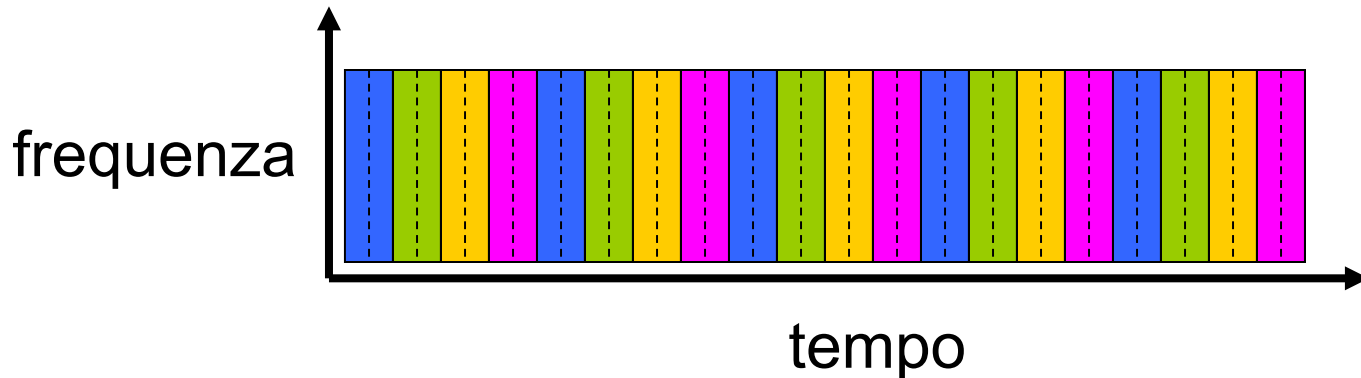
4 utenti



FDM



TDM



# Il nucleo della rete: commutazione di pacchetto

## *Il flusso di dati punto-punto viene suddiviso in pacchetti*

- ❑ *I pacchetti degli utenti A e B condividono le risorse di rete*
- ❑ *Ciascun pacchetto utilizza completamente il canale*
- ❑ *Le risorse vengono usate a seconda delle necessità*

## *Contesa per le risorse*

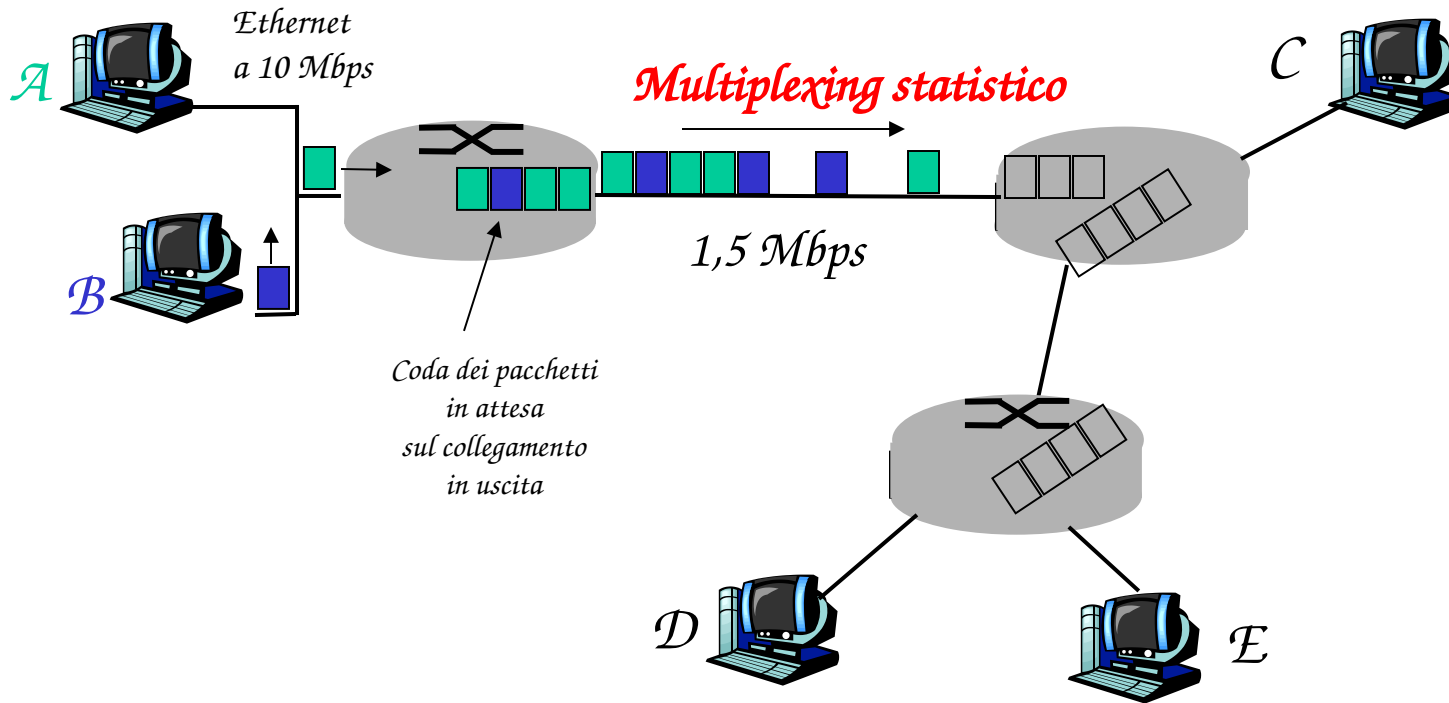
- ❑ *La richiesta di risorse può eccedere il quantitativo disponibile*
- ❑ **congestione:** *accodamento dei pacchetti, attesa per l'utilizzo del collegamento*
- ❑ **store and forward:** *il commutatore deve ricevere l'intero pacchetto prima di poter cominciare a trasmettere sul collegamento in uscita*

*Larghezza di banda suddivisa in pezzi"*

*Allocazione dedicata*

*Risorse riservate*

## Commutazione di pacchetto: multiplexing statistico



La sequenza dei pacchetti *A* e *B* non segue uno schema prefissato. Condivisione di risorse su richiesta □ **multiplexing statistico**.

*TDM*: ciascun host ottiene uno slot di tempo dedicato unicamente a quella connessione.

# Capitolo 1: roadmap

*1.1 Cos'è Internet?*

*1.2 Ai confini della rete*

*1.3 Il nucleo della rete*

*1.4 Accesso alla rete e mezzi trasmissivi*

*1.5 ISP e dorsali Internet*

*1.6 Ritardi e perdite nelle reti a commutazione di pacchetto*

*1.7 Livelli di protocollo e loro modelli di servizio*

*1.8 Storia del computer networking e di Internet*

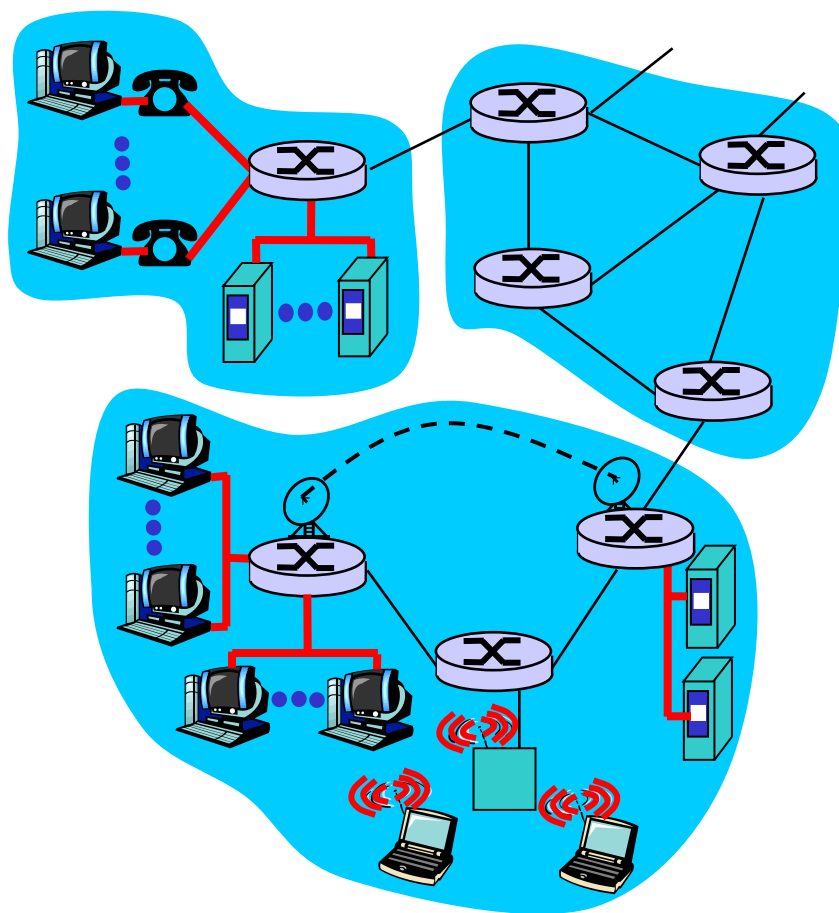
# Accesso alla rete e mezzi trasmissivi

*D: Come collegare sistemi terminali a edge router?*

- ☐ Reti di accesso residenziale
- ☐ Reti di accesso aziendale (scuole, società, istituzioni)
- ☐ Reti di accesso wireless

*Ricordate:*

- ☐ ampiezza di banda (bit al secondo) della rete di accesso?
- ☐ condivisa o dedicata?

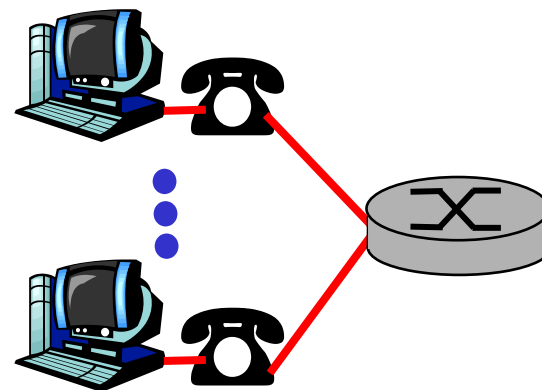




# Accesso residenziale: punto-punto

## ❑ *Modem dial-up*

- ❖ *fino a 56 Kbps di accesso diretto al router (ma spesso è inferiore)*
- ❖ *non è possibile “navigare” e telefonare allo stesso momento*



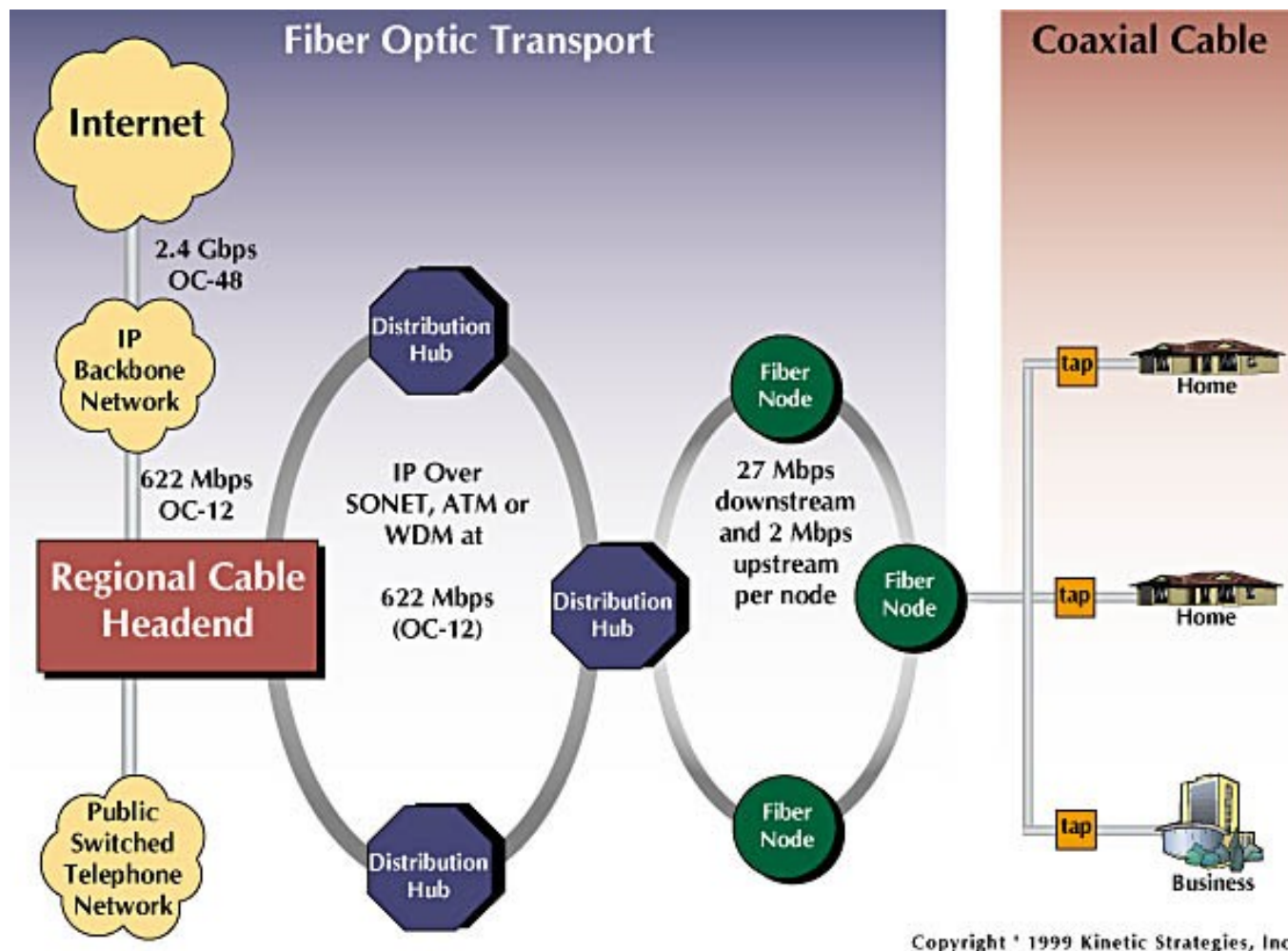
## ❑ *ADSL: asymmetric digital subscriber line*

- ❖ *fino a 1 Mbps in upstream (attualmente, in genere < 256 kbps)*
- ❖ *fino a 8 Mbps downstream (attualmente, in genere < 1 Mbps)*
- ❖ *FDMA: 50 kHz - 1 MHz per il downstream*
  - 4 kHz - 50 kHz per il canale di upstream*
  - 0 kHz - 4 kHz per il canale telefonico ordinario a due vie*

## Accesso residenziale: cable modem

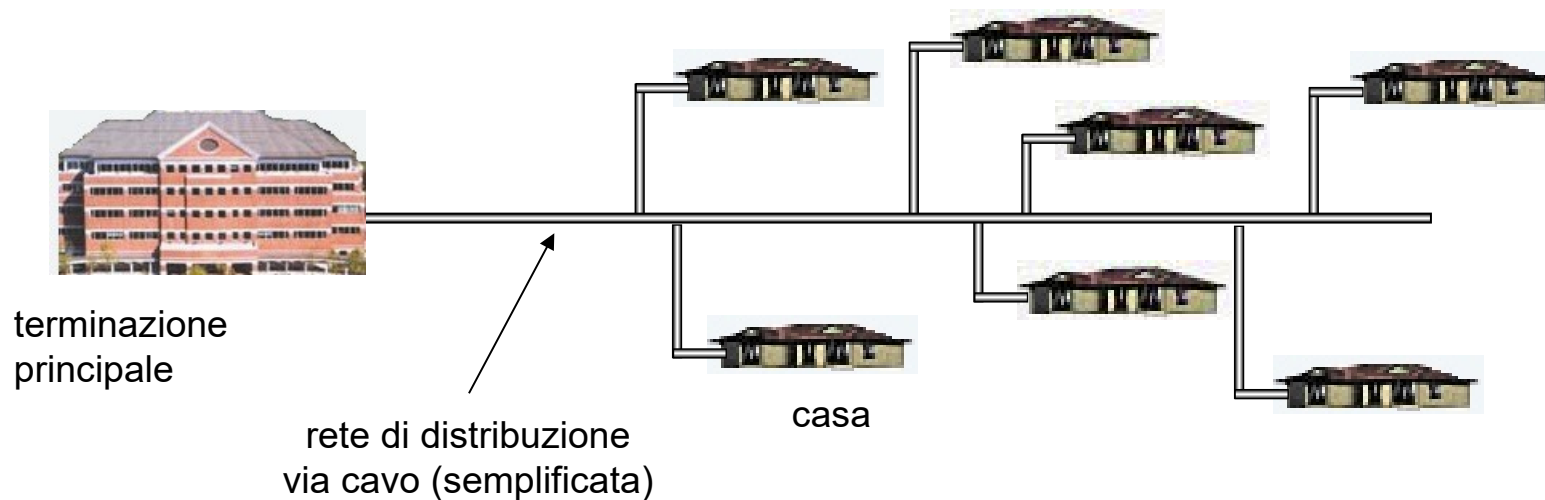
- *HFC: hybrid fiber coax*
  - ❖ *asimmetrico: fino a 30 Mbps in downstream, 2 Mbps in upstream*
- *rete ibrida a fibra e cavo coassiale collega le case ai router degli ISP*
  - ❖ *HFC rappresenta un mezzo di trasmissione condiviso*

## Accesso residenziale: cable modem

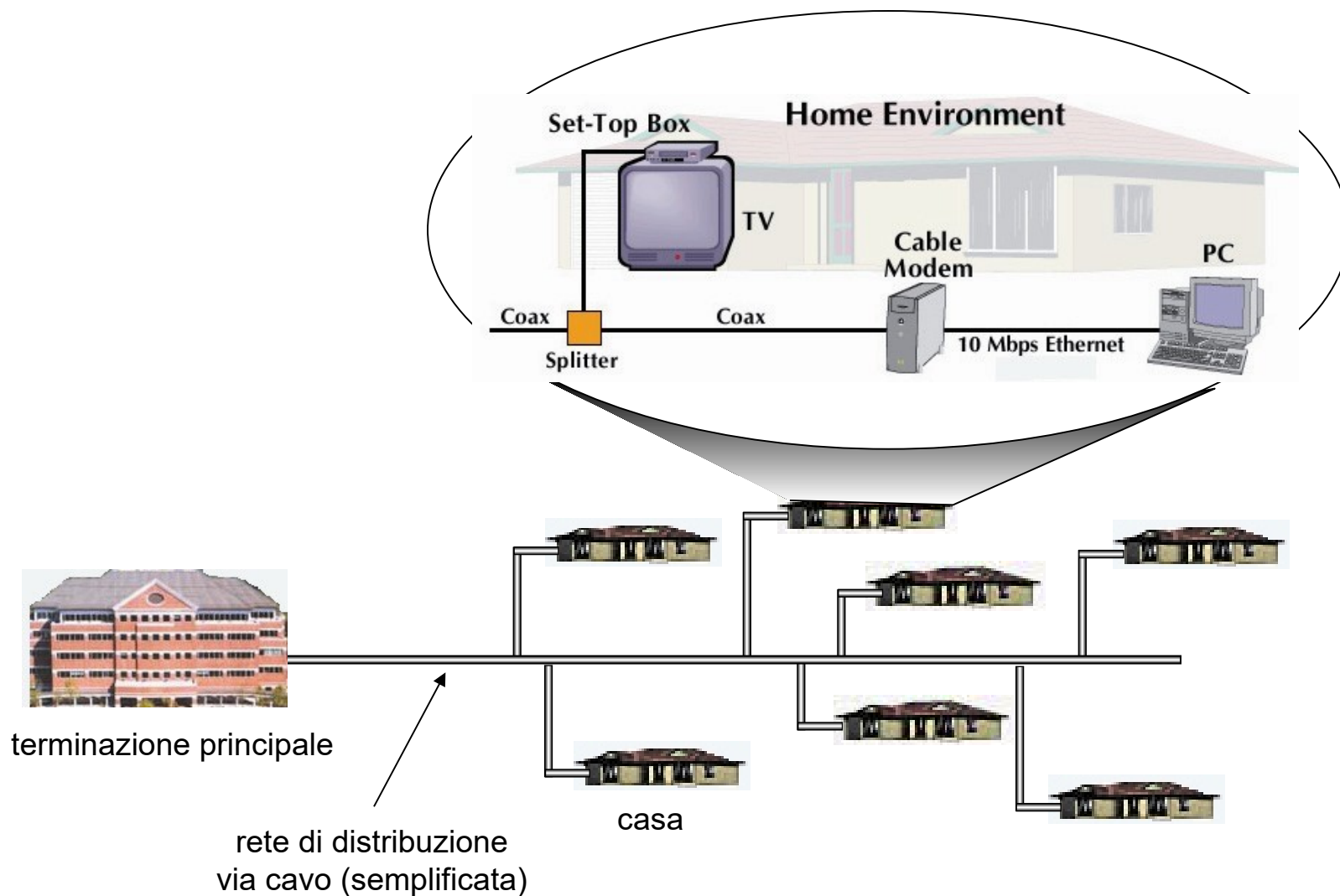


## Rete d'accesso ibrida: una visione d'insieme

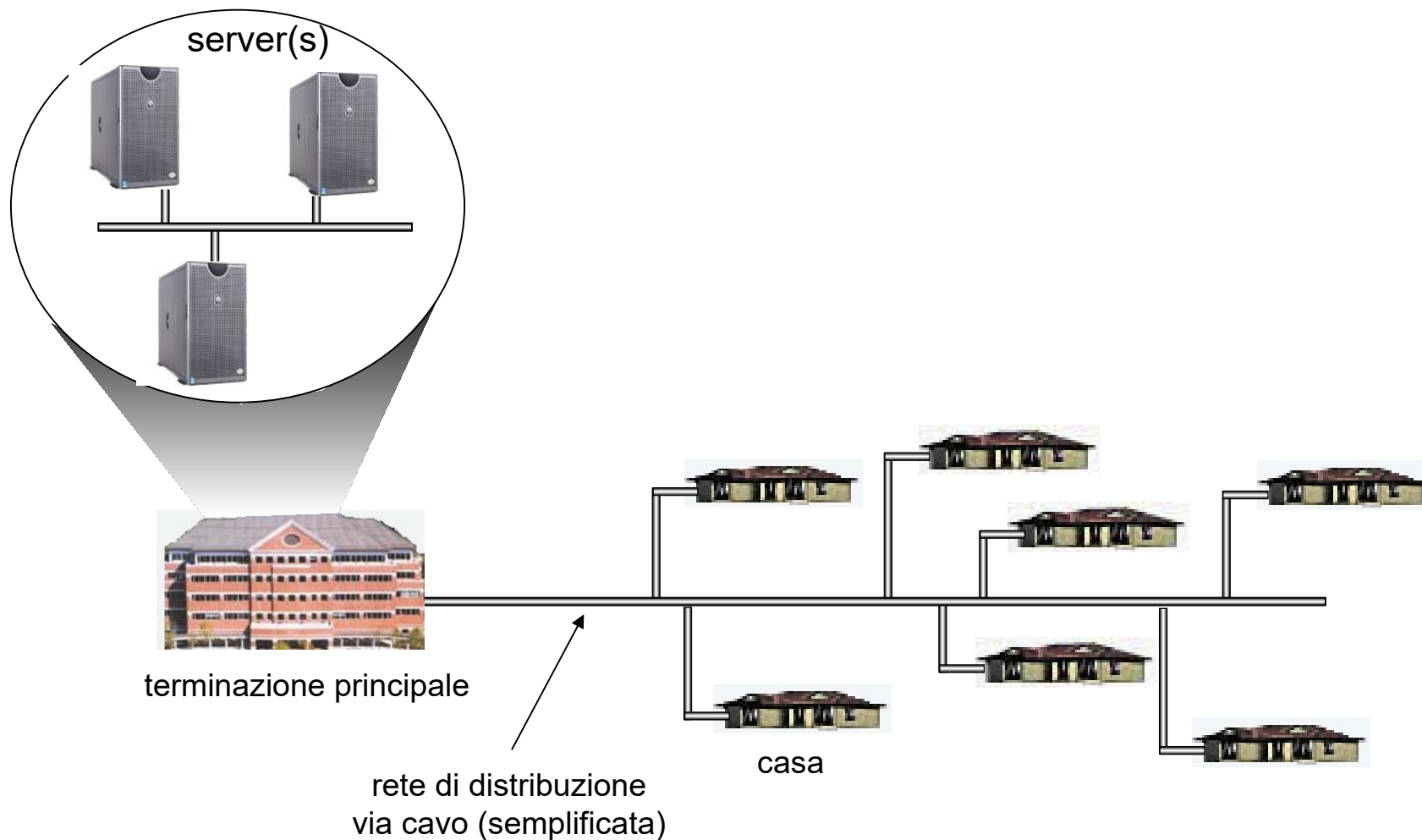
*in genere da 500 a 5.000 case*



## Rete d'accesso ibrida: una visione d'insieme

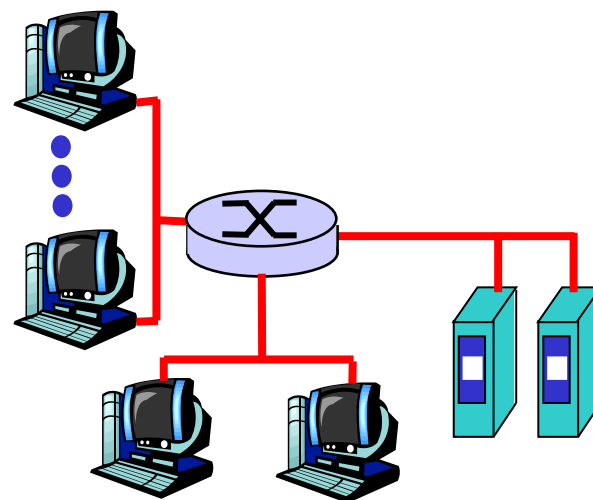


## Rete d'accesso ibrida: una visione d'insieme



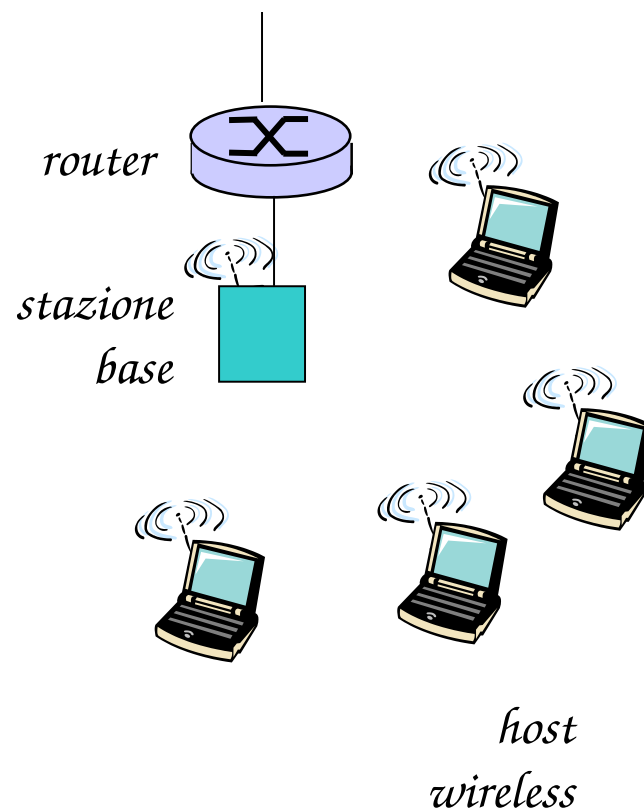
## Accesso aziendale: reti locali (LAN)

- Una LAN collega i sistemi terminali di aziende e università all'edge router
- *Ethernet:*
  - ❖ un canale condiviso o dedicato collega i sistemi terminali ai router
  - ❖ 10 Mbs, 100 Mbps, Gigabit Ethernet
- Le LAN: to-do



# Accesso wireless

- ❑ Una rete d'accesso wireless collega i sistemi terminali al router
  - ❖ attraverso la stazione base, detta anche "access point"
- ❑ *LAN wireless:*
  - ❖ 802.11b (WiFi): 11 Mbps
- ❑ *rete d'accesso wireless geografica*
  - ❖ gestita da un provider di telecomunicazioni
  - ❖ 3G ~ 384 Kbps
  - ❖ 4g
  - ❖ 5g ?

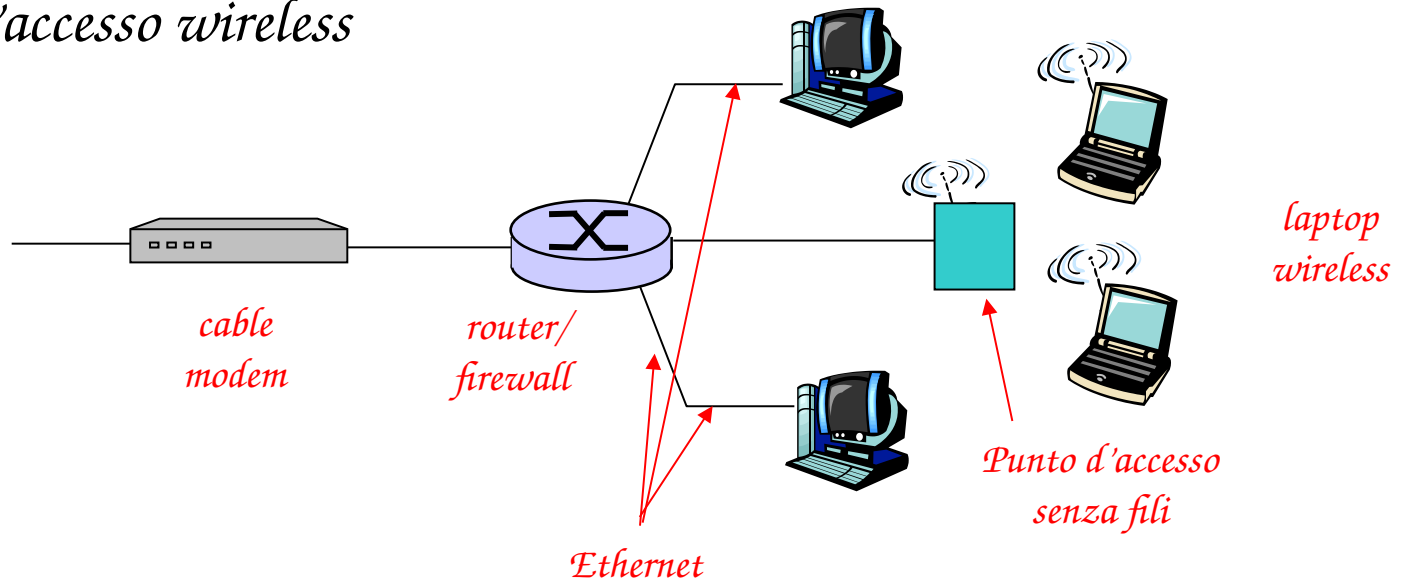




# Reti da abitazione

*Componenti di una tipica rete da abitazione:*

- ☐ *ADSL o cable modem*
- ☐ *router/firewall/NAT*
- ☐ *Ethernet*
- ☐ *Punto d'accesso wireless*



# Mezzi trasmissivi

- *Bit*: viaggia da un sistema terminale a un altro, passando per una serie di coppie trasmittente-ricevente
- *Mezzo fisico*: ciò che sta tra il trasmittente e il ricevente
- *Mezzi guidati*:
  - ❖ i segnali si propagano in un mezzo fisico: fibra ottica, filo di rame o cavo coassiale
- *Mezzi a onda libera*:
  - ❖ i segnali si propagano nell'atmosfera e nello spazio esterno

## Doppino intrecciato (TP)

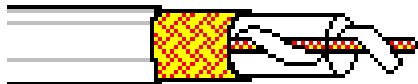
- due fili di rame distinti
  - ❖ Categoria 3: tradizionale cavo telefonico, 10 Mbps Ethernet
  - ❖ Categoria 5: 100 Mbps Ethernet



# Mezzi trasmissivi: cavo coassiale e fibra ottica

## *Cavo coassiale:*

- ❑ *due conduttori in rame concentrici*
- ❑ *bidirezionale*
- ❑ *banda base:*
  - ❖ *singolo canale sul cavo*
  - ❖ *legacy Ethernet*
- ❑ *banda larga:*
  - ❖ *più canali sul cavo*
  - ❖ *HFC*



## *Fibra ottica:*

- ❑ *Mezzo sottile e flessibile che conduce impulsi di luce (ciascun impulso rappresenta un bit)*
- ❑ *Alta frequenze trasmissiva:*
  - ❖ *Elevata velocità di trasmissione punto-punto (10's-100's Gps)*
- ❑ *Basso tasso di errore, immune all'interferenza elettromagnetica*



# Mezzi trasmissivi: canali radio

- *trasportano segnali nello spettro elettromagnetico*
- *non richiedono l'installazione fisica di cavi*
- *bidirezionali*
- *effetti dell'ambiente di propagazione:*
  - ❖ *riflessione*
  - ❖ *ostruzione da parte di ostacoli*
  - ❖ *interferenza*

## *Tipi di canali radio:*

- *microonde terrestri*
  - ❖ *es.: canali fino a 45 Mbps*
- *LAN* (es.: *Wifi*)
  - ❖ *2 Mbps, 11 Mbps, 54 Mbps*
- *wide-area* (es.: *cellulari*)
  - ❖ *es.: 3/4/5G: centinaia di kbps*
- *satellitari*
  - ❖ *canali fino a 45 Mbps channel (o sottomultipli)*
  - ❖ *ritardo punto-punto di 270 msec*
  - ❖ *geostazionari/a bassa quota*

# Capitolo 1: roadmap

*1.1 Cos'è Internet?*

*1.2 Ai confini della rete*

*1.3 Il nucleo della rete*

*1.4 Accesso alla rete e mezzi trasmissivi*

*1.5 ISP e dorsali Internet*

*1.6 Ritardi e perdite nelle reti a commutazione di pacchetto*

*1.7 Livelli di protocollo e loro modelli di servizio*

*1.8 Storia del computer networking e di Internet*

## Situazione attuale in Italia?

- ❑ *Eunet, primo fornitore di accessi in Italia*
- ❑ *Definizione di un backbone per le reti verso la fine degli anni '80*

# La rete Garr-B

## • **Back bone**

- linee blu a 2.5 Gbps

- Linee rosse a 155 Mbps

## • *Collegamenti Internazionali*

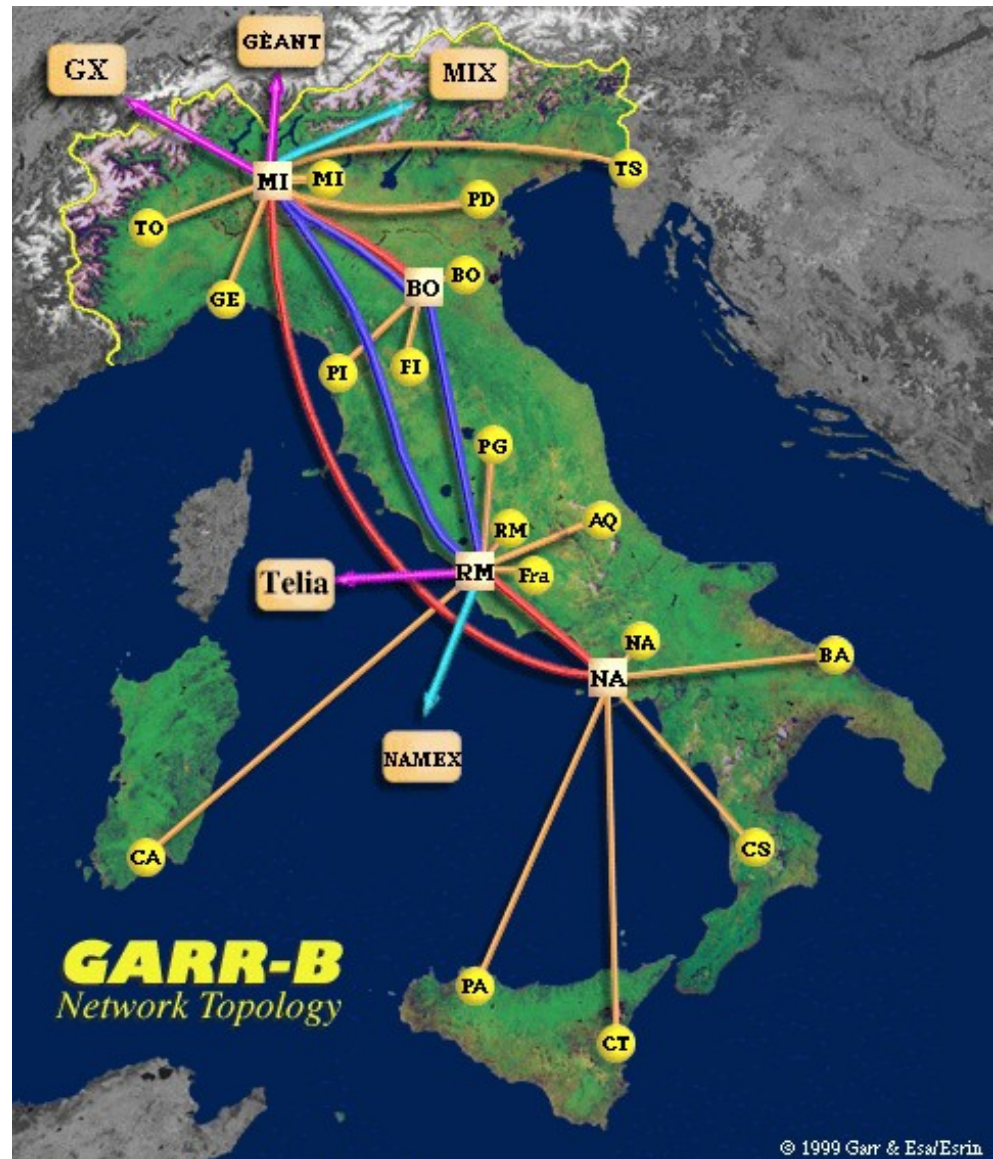
- **MI-GEANT** 2.5 Gbps

- **MI-GX** 2.5 Gbps

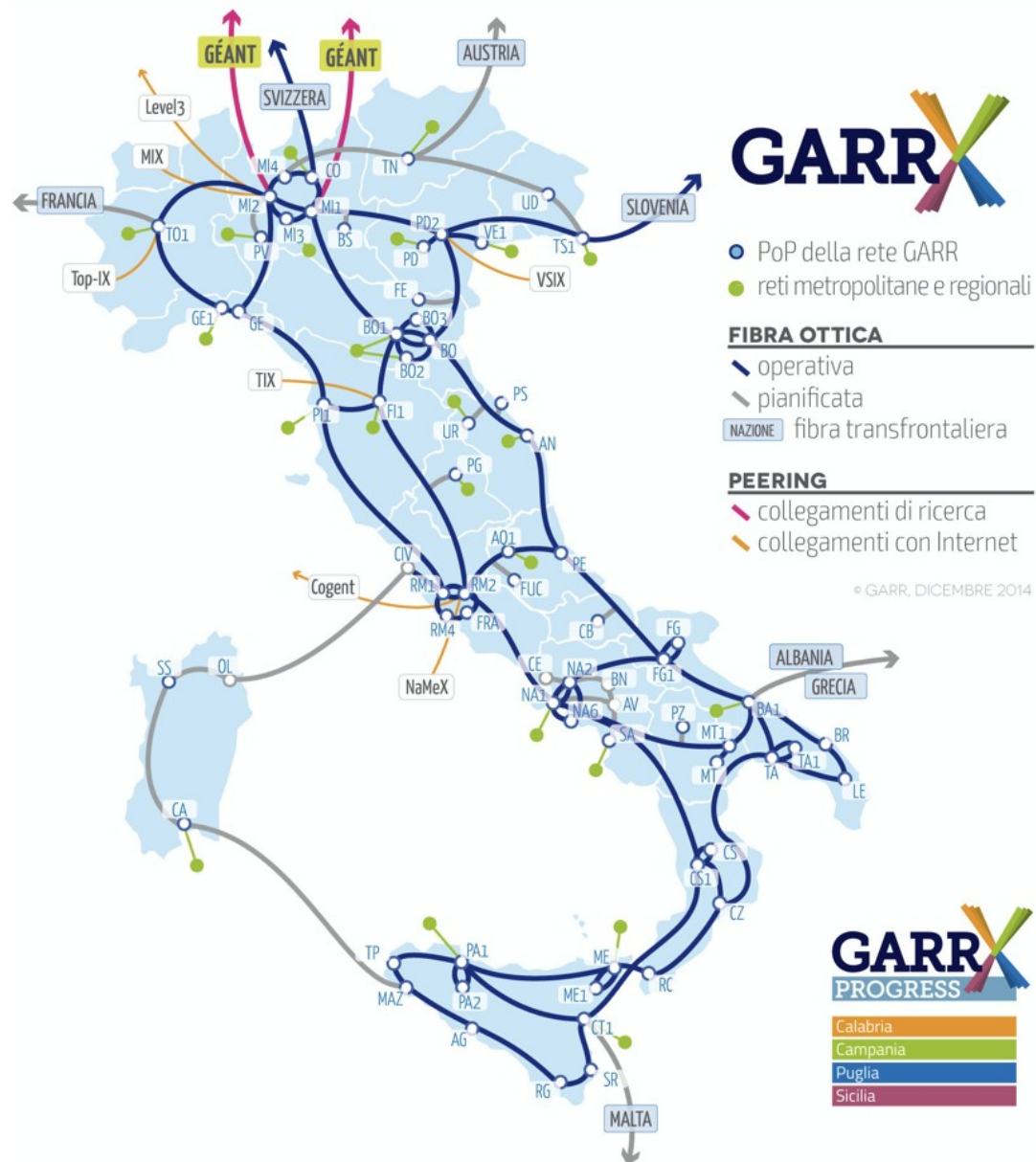
- **RM-KQ** 622 Mbps (in attivazione)

## • *Collegamenti tra Backbone e POP di accesso*

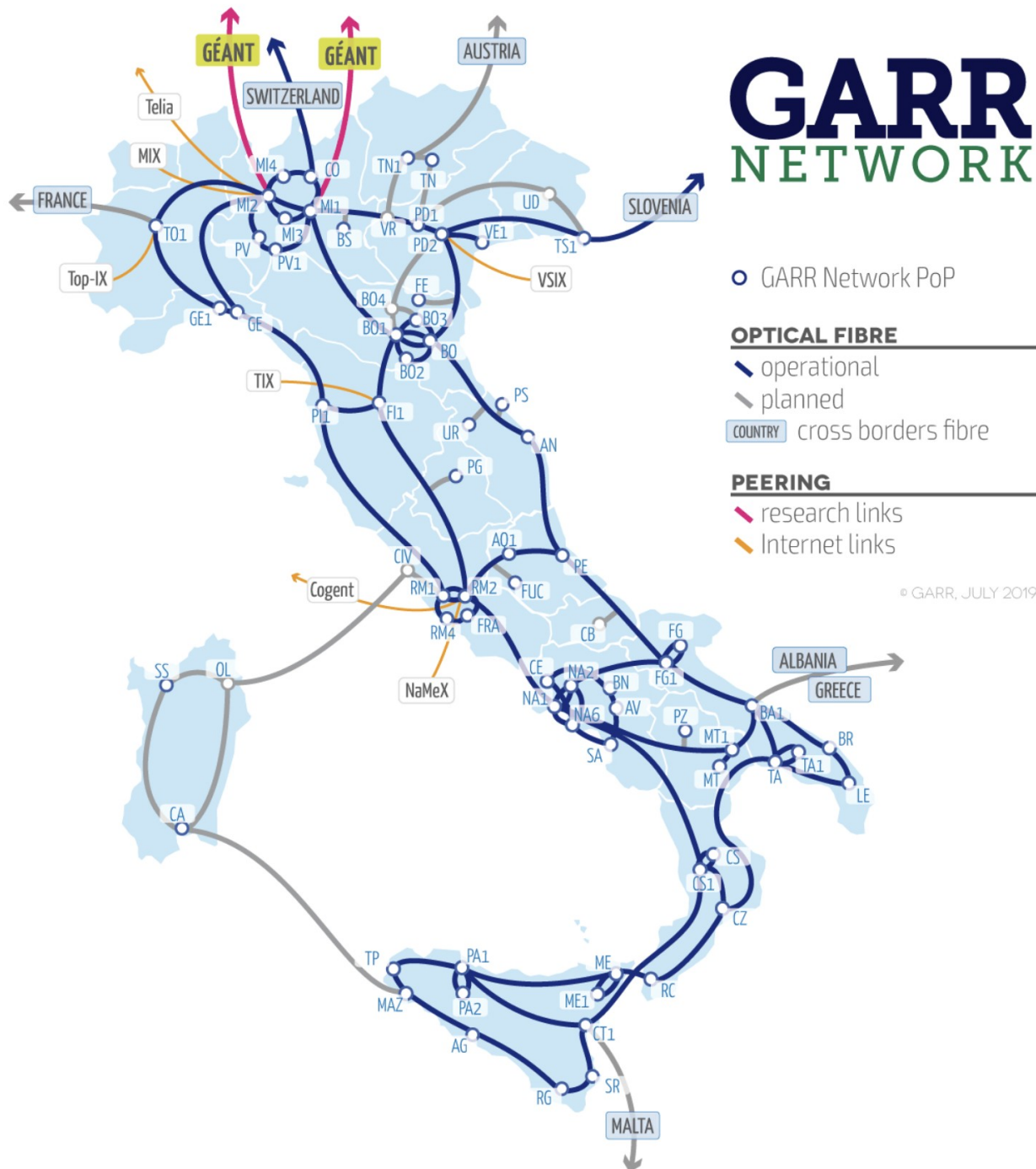
- **RM-AQ** 2 x 34 Mbps



# MAPPE STORICHE DELLA RETE GARR







# Netiquette

- *Spirito collaborativo e regole di comportamento (netiquette)*
  - ❖ *Non sprecare risorse (es. la banda di trasmissione)*
  - ❖ *Non fare niente che possa danneggiare la rete (es. Virus)*
  - ❖ *Rispetto della privacy, della proprietà*
  - ❖ *Non inviare propaganda non richiesta (spamming)*
  - ❖ *Intercettare le comunicazioni (sniffing)*
  - ❖ *Uso non autorizzato di risorse protette (cracking)*
  - ❖ *Agire sotto mentite spoglie (spoofing)*

# Capitolo 1: roadmap

*1.1 Cos'è Internet?*

*1.2 Ai confini della rete*

*1.3 Il nucleo della rete*

*1.4 Accesso alla rete e mezzi trasmissivi*

*1.5 ISP e dorsali Internet*

*1.6 Ritardi e perdite nelle reti a commutazione di pacchetto*

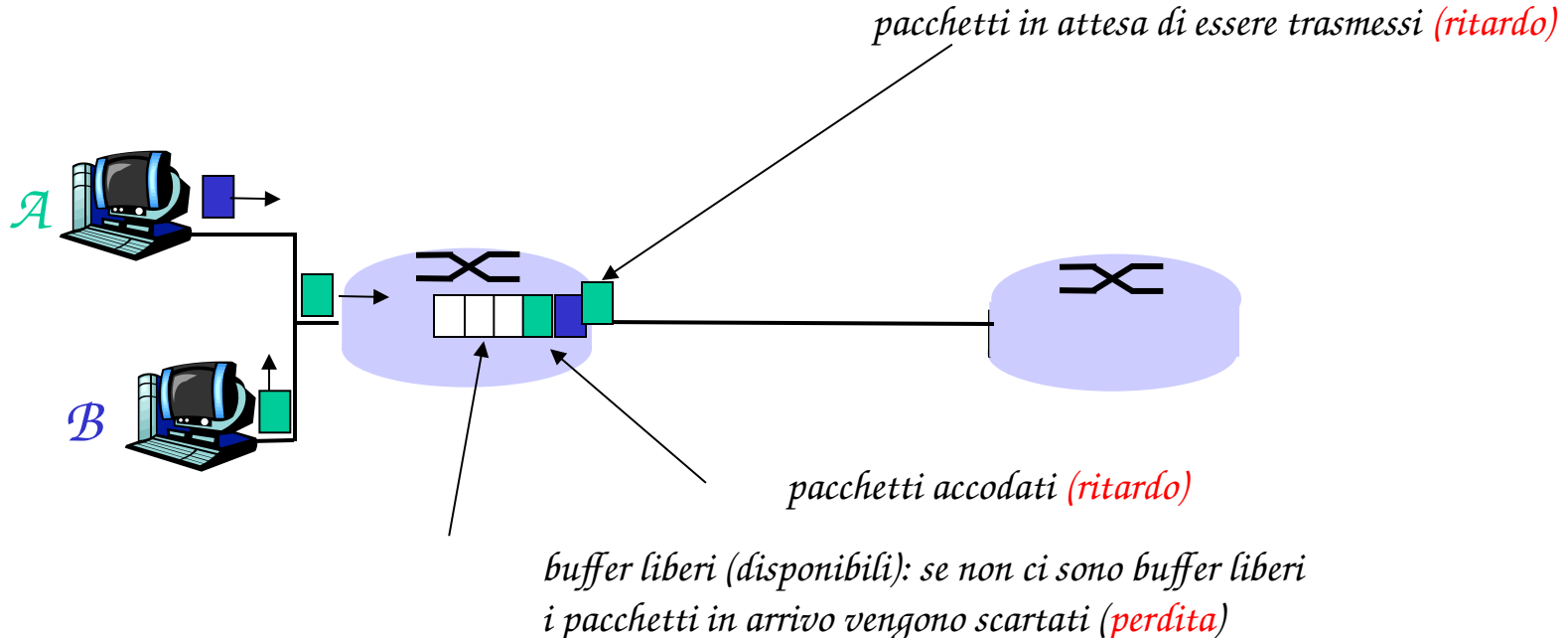
*1.7 Livelli di protocollo e loro modelli di servizio*

*1.8 Storia del computer networking e di Internet*

# Come si verificano ritardi e perdite?

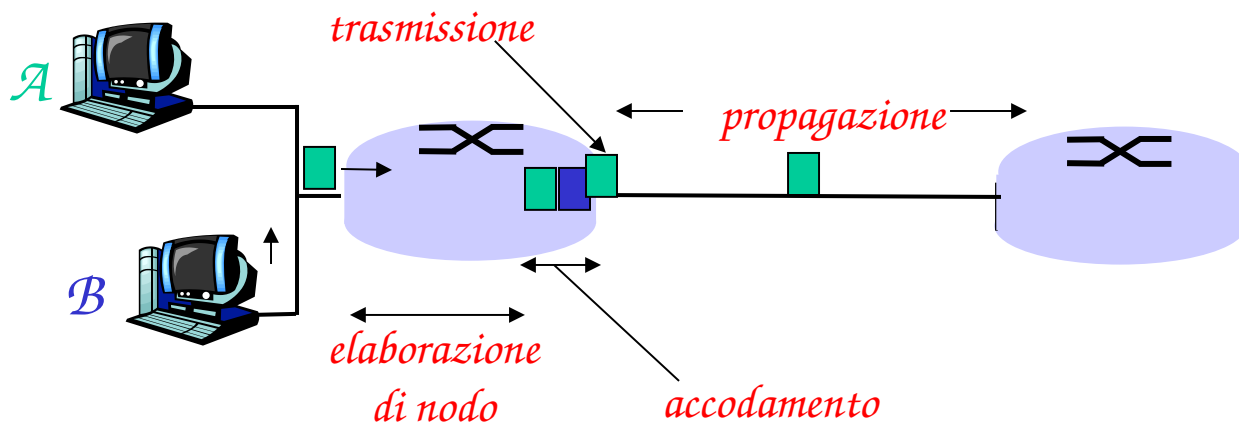
*I pacchetti si accodano nei buffer dei router*

- *il tasso di arrivo dei pacchetti sul collegamento eccede la capacità del collegamento di evaderli*
- *i pacchetti si accodano, in attesa del proprio turno*



# Quattro cause di ritardo per i pacchetti

- ❑ 1. *Ritardo di elaborazione del nodo:*
  - ❖ controllo errori sui bit
  - ❖ determinazione del canale di uscita
- ❑ 2. *Ritardo di accodamento*
  - ❖ attesa di trasmissione
  - ❖ livello di congestione del router



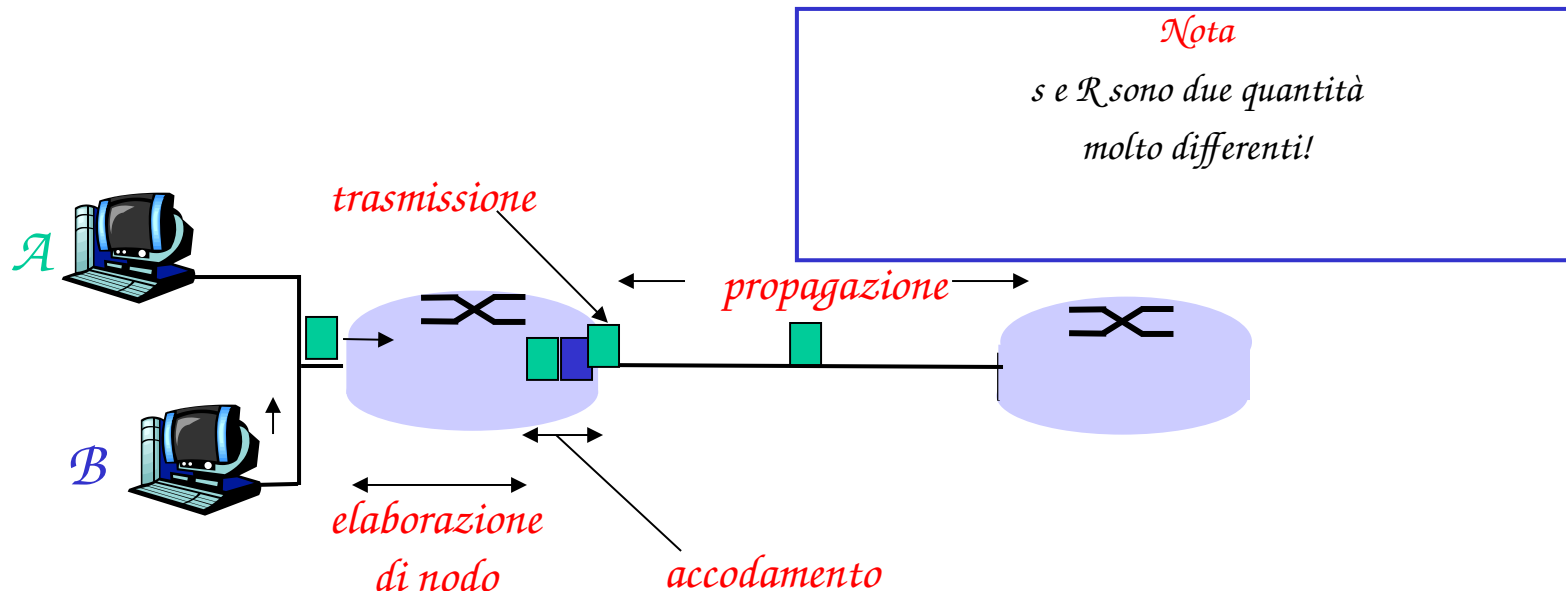
# Ritardo nelle reti a commutazione di pacchetto

## 3. Ritardo di trasmissione ( $L/R$ ):

- $R$  = frequenza di trasmissione del collegamento (in bps)
- $L$  = lunghezza del pacchetto (in bit)
- Ritardo di trasmissione =  $L/R$

## 4. Ritardo di propagazione ( $d/s$ )

- $d$  = lunghezza del collegamento fisico
- $s$  = velocità di propagazione del collegamento ( $\sim 2 \times 10^8$  m/sec)
- Ritardo di propagazione =  $d/s$



# Ritardo di nodo

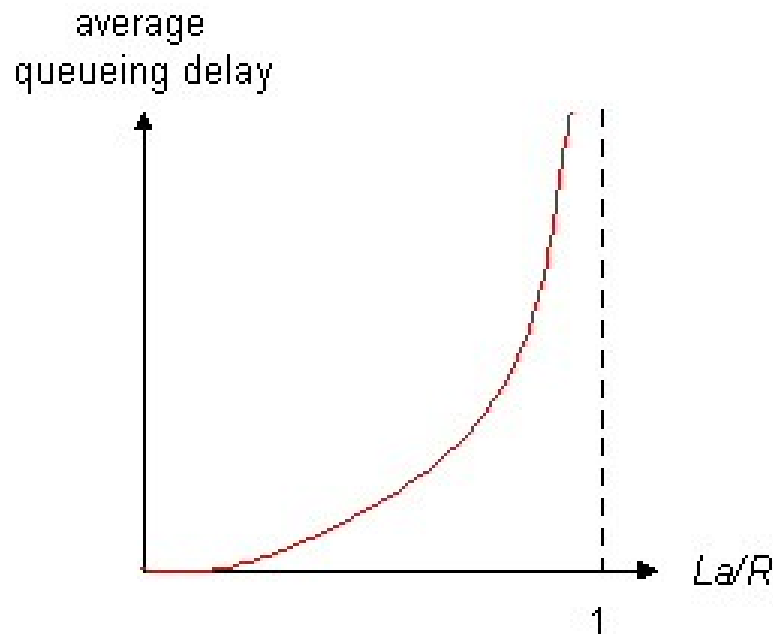
$$d_{\text{nodal}} = d_{\text{proc}} + d_{\text{queue}} + d_{\text{trans}} + d_{\text{prop}}$$

- $d_{\text{proc}}$  = ritardo di elaborazione (processing delay)
  - ❖ in genere pochi microsecondi, o anche meno
- $d_{\text{queue}}$  = ritardo di accodamento (queuing delay)
  - ❖ dipende dalla congestione
- $d_{\text{trans}}$  = ritardo di trasmissione (transmission delay)
  - ❖  $= \mathcal{L}/\mathcal{R}$ , significativo sui collegamenti a bassa velocità
- $d_{\text{prop}}$  = ritardo di propagazione (propagation delay)
  - ❖ da pochi microsecondi a centinaia di millisecondi

# Ritardo di accodamento

- $\mathcal{R}$  = frequenza di trasmissione (bps)
- $\mathcal{L}$  = lunghezza del pacchetto (bit)
- $a$  = tasso medio di arrivo dei pacchetti

$\mathcal{L}a/\mathcal{R}$  = intensità di traffico

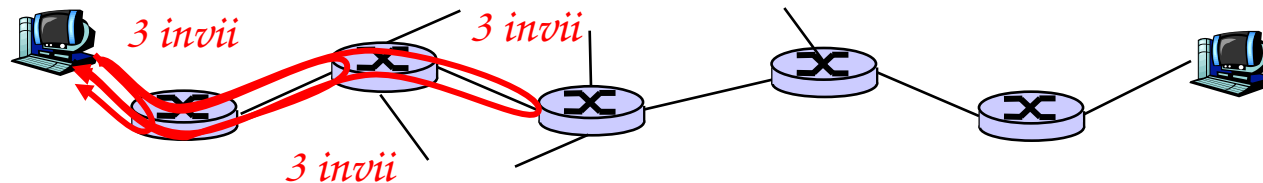


- $\mathcal{L}a/\mathcal{R} \sim 0$ : poco ritardo
- $\mathcal{L}a/\mathcal{R} \rightarrow 1$ : il ritardo si fa consistente
- $\mathcal{L}a/\mathcal{R} > 1$ : più “lavoro” in arrivo di quanto possa essere effettivamente svolto, ritardo medio infinito!



# Ritardi e percorsi in Internet


- *Ma cosa significano effettivamente ritardi e perdite nella “vera” Internet?*
- **Traceroute:** *programma diagnostico che fornisce una misura del ritardo dalla sorgente al router lungo i percorsi Internet punto-punto verso la destinazione.*
  - ❖ *invia tre pacchetti che raggiungeranno il router i sul percorso verso la destinazione*
  - ❖ *il router i restituirà i pacchetti al mittente*
  - ❖ *il mittente calcola l'intervallo tra trasmissione e risposta*



# Ritardi e percorsi in Internet


*traceroute: da gaia.cs.umass.edu a www.eurecom.fr*

*Tre misure di ritardo da  
gaia.cs.umass.edu a cs-gw.cs.umass.edu*




1 cs-gw (128.119.240.254) 1 ms 1 ms 2 ms  
2 border1-rt-fa5-1-0.gw.umass.edu (128.119.3.145) 1 ms 1 ms 2 ms  
3 cht-vbns.gw.umass.edu (128.119.3.130) 6 ms 5 ms 5 ms  
4 jn1-at1-0-0-19.wor.vbns.net (204.147.132.129) 16 ms 11 ms 13 ms  
5 jn1-so7-0-0-0.wae.vbns.net (204.147.136.136) 21 ms 18 ms 18 ms  
6 abilene-vbns.abilene.ucaid.edu (198.32.11.9) 22 ms 18 ms 22 ms  
7 nycm-wash.abilene.ucaid.edu (198.32.8.46) 22 ms 22 ms 22 ms  
8 62.40.103.253 (62.40.103.253) 104 ms 109 ms 106 ms  
9 de2-1.de1.de.geant.net (62.40.96.129) 109 ms 102 ms 104 ms  
10 de.fr1.fr.geant.net (62.40.96.50) 113 ms 121 ms 114 ms  
11 renater-gw.fr1.fr.geant.net (62.40.103.54) 112 ms 114 ms 112 ms  
12 nio-n2.cssi.renater.fr (193.51.206.13) 111 ms 114 ms 116 ms  
13 nice.cssi.renater.fr (195.220.98.102) 123 ms 125 ms 124 ms  
14 r3t2-nice.cssi.renater.fr (195.220.98.110) 126 ms 126 ms 124 ms  
15 eurecom-valbonne.r3t2.ft.net (193.48.50.54) 135 ms 128 ms 133 ms  
16 194.214.211.25 (194.214.211.25) 126 ms 128 ms 126 ms  
17 \* \* \*  
18 \* \* \*  
19 fantasia.eurecom.fr (193.55.113.142) 132 ms 128 ms 136 ms

*collegamento  
transoceanico*



*\* significa nessuna risposta (risposta persa, il router non risponde)*



## Perdita di pacchetti

- ❑ *una coda (detta anche buffer) ha capacità finita*
- ❑ *quando il pacchetto trova la coda piena, viene scartato (e quindi va perso)*
- ❑ *il pacchetto perso può essere ritrasmesso dal nodo precedente, dal sistema terminale che lo ha generato, o non essere ritrasmesso affatto*

# Capitolo 1: roadmap

*1.1 Cos'è Internet?*

*1.2 Ai confini della rete*

*1.3 Il nucleo della rete*

*1.4 Accesso alla rete e mezzi trasmissivi*

*1.5 ISP e dorsali Internet*

*1.6 Ritardi e perdite nelle reti a commutazione di pacchetto*

*1.7 Livelli di protocollo e loro modelli di servizio*

*1.8 Storia del computer networking e di Internet*

# Livelli di protocollo

## Le reti sono complesse!

- molti “pezzi”:
  - ❖ *host*
  - ❖ *router*
  - ❖ *svariate tipologie di mezzi trasmissivi*
  - ❖ *applicazioni*
  - ❖ *protocolli*
  - ❖ *hardware, software*

## Domanda:

*C'è qualche speranza di organizzare l'architettura delle reti?*

*O almeno la nostra trattazione sulle reti?*

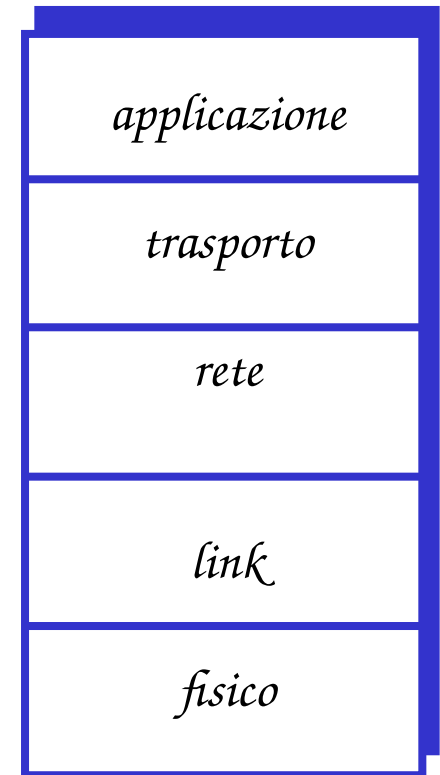
# Perché la stratificazione?

*Quando si ha a che fare con sistemi complessi:*

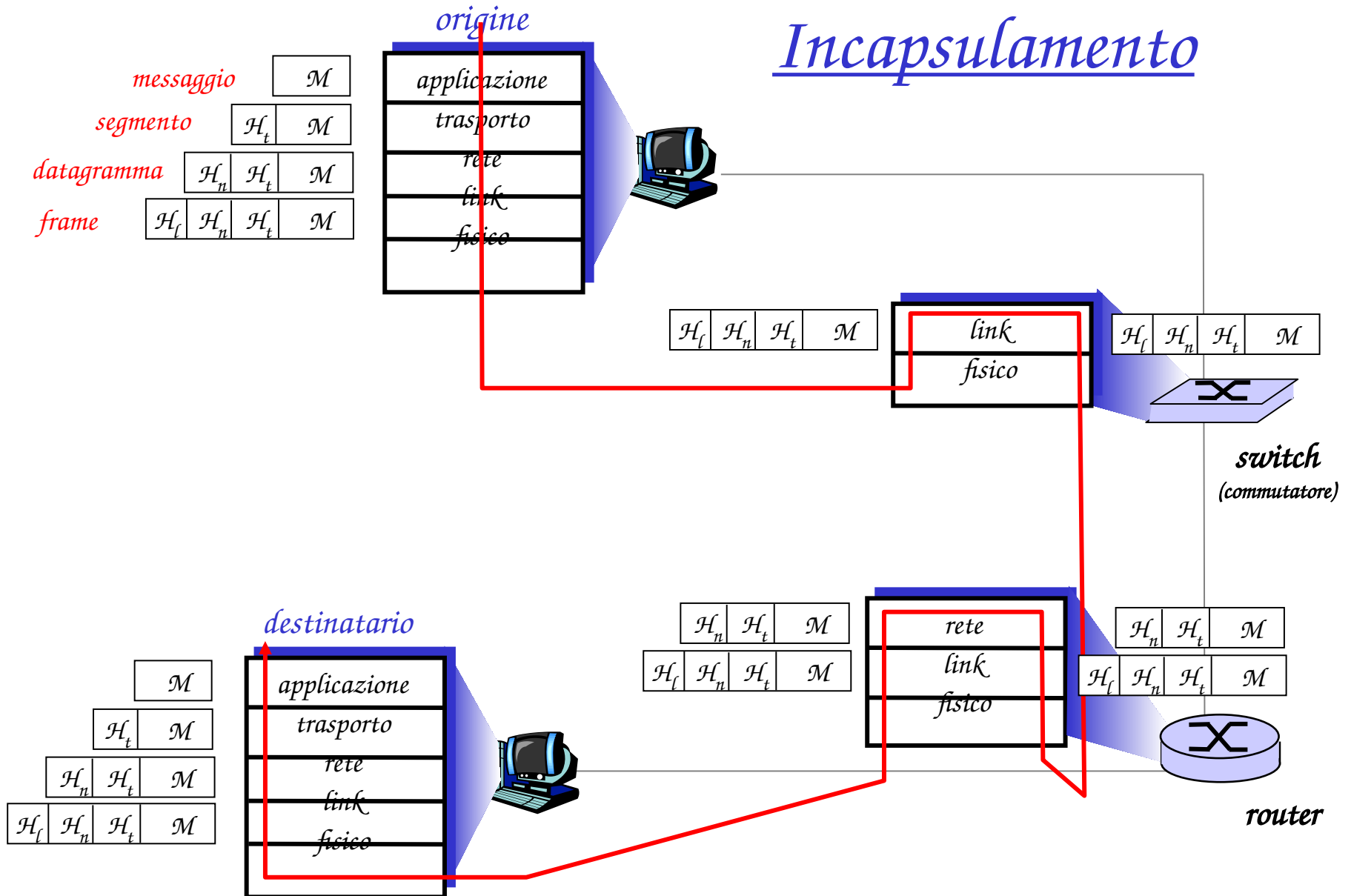
- *Una struttura “esplicita” consente l’identificazione dei vari componenti di un sistema complesso e delle loro inter-relazioni*
  - ❖ *analisi del **modello di riferimento a strati***
- *La modularizzazione facilita la manutenzione e l’aggiornamento di un sistema*
  - ❖ *modifiche implementative al servizio di uno dei livelli risultano trasparenti al resto del sistema*
  - ❖ *es.: modifiche nelle procedure effettuate al gate non condizionano il resto del sistema*
- *Il modello a strati può essere considerato dannoso?*

# Pila di protocolli Internet

- *applicazione*: di supporto alle applicazioni di rete
  - ❖ *FTP, SMTP, HTTP*
- *trasporto*: trasferimento dei messaggi a livello di applicazione tra il modulo client e server di un'applicazione
  - ❖ *TCP, UDP*
- *rete*: instradamento dei datagrammi dall'origine al destinatario
  - ❖ *IP, protocolli di instradamento*
- *link (collegamento)*: instradamento dei datagrammi attraverso una serie di commutatori di pacchetto
  - ❖ *PPP, Ethernet*
- *fisico*: trasferimento dei singoli bit



# Incapsulamento





# Capitolo 1: roadmap

*1.1 Cos'è Internet?*

*1.2 Ai confini della rete*

*1.3 Il nucleo della rete*

*1.4 Accesso alla rete e mezzi trasmissivi*

*1.5 ISP e dorsali Internet*

*1.6 Ritardi e perdite nelle reti a commutazione di pacchetto*

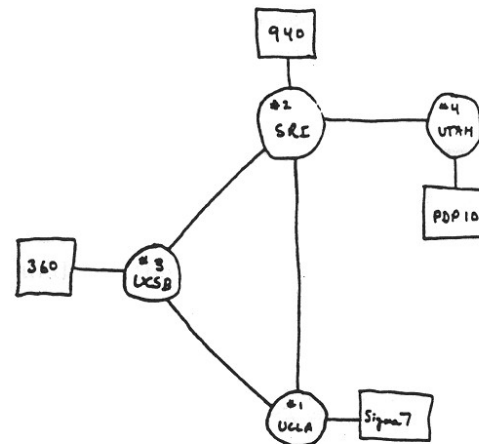
*1.7 Livelli di protocollo e loro modelli di servizio*

*1.8 Storia del computer networking e di Internet*

# Storia di Internet

## *1961-1972: sviluppo della commutazione di pacchetto*

- *1961: Kleinrock - la teoria delle code dimostra l'efficacia dell'approccio a commutazione di pacchetto*
- *1964: Baran - uso della commutazione di pacchetto nelle reti militari*
- *1967: il progetto ARPAnet viene concepito dall'Advanced Research Projects Agency*
- *1969: primo nodo operativo ARPAnet*
- *1972:*
  - ❖ *dimostrazione pubblica di ARPAnet*
  - ❖ *NCP (Network Control Protocol), primo protocollo tra nodi*
  - ❖ *Primo programma di posta elettronica*
  - ❖ *ARPAnet ha 15 nodi*



THE ARPA NETWORK

# Storia di Internet

## *1972-1980: Internetworking e reti proprietarie*

- ❑ *1970: rete satellitare ALOHAnet che collega le università delle Hawaii*
- ❑ *1974: Cerf e Kahn - architettura per l'interconnessione delle reti*
- ❑ *1976: Ethernet allo Xerox PARC*
- ❑ *Fine anni '70: architetture proprietarie: DECnet, SNA, XNA*
- ❑ *1979: ARPAnet ha 200 nodi*

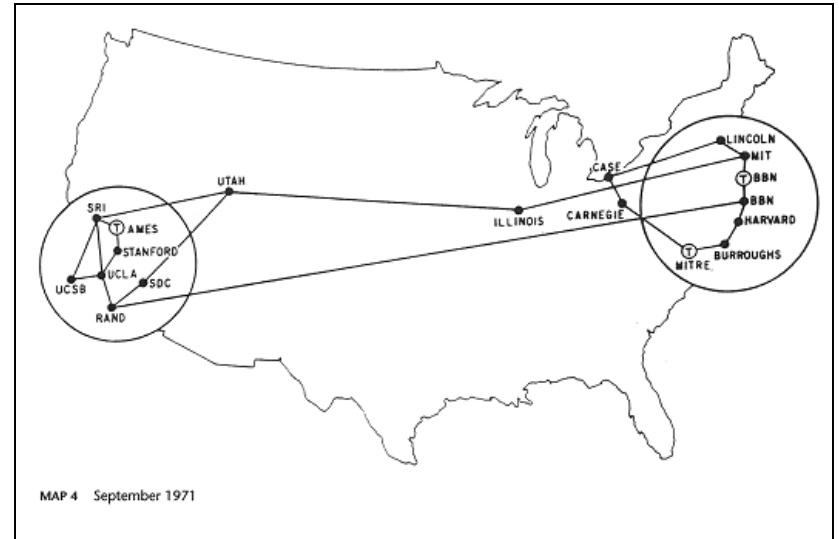
### *Le linee guida di Cerf e Kahn sull'internetworking:*

- ❖ *minimalismo, autonomia - per collegare le varie reti non occorrono cambiamenti interni*
- ❖ *modello di servizio best effort*
- ❖ *router stateless*
- ❖ *controllo decentralizzato*

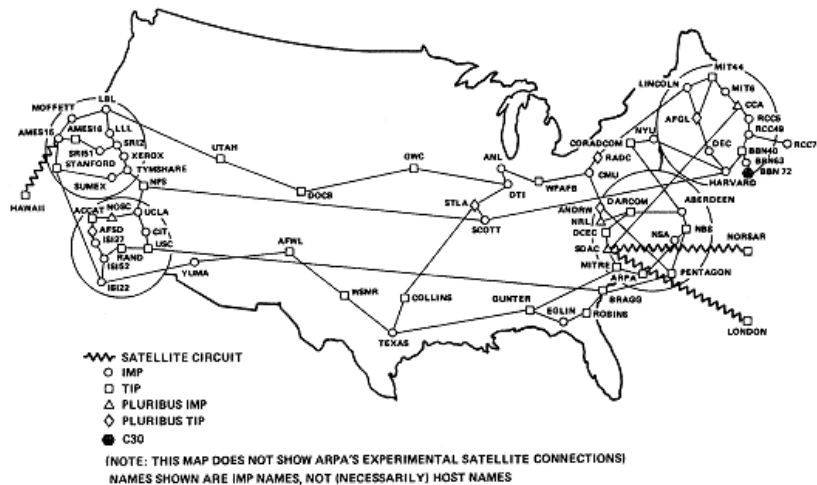
*definiscono l'attuale architettura di Internet*

# ARPANET

□ Settembre 1971



ARPANET GEOGRAPHIC MAP, OCTOBER 1980



□ Ottobre 1980

# Storia di Internet

## *1980-1990: nuovi protocolli, proliferazione delle reti*

- *1983: rilascio di TCP/IP*
- *1982: definizione del protocollo smtp per la posta elettronica*
- *1983: definizione del DNS per la traduzione degli indirizzi IP*
- *1985: definizione del protocollo ftp*
- *1988: controllo della congestione TCP*
- *nuove reti nazionali: Csnnet, BITnet, NSFnet, Minitel*
- *100.000 host collegati*

# NSFNet

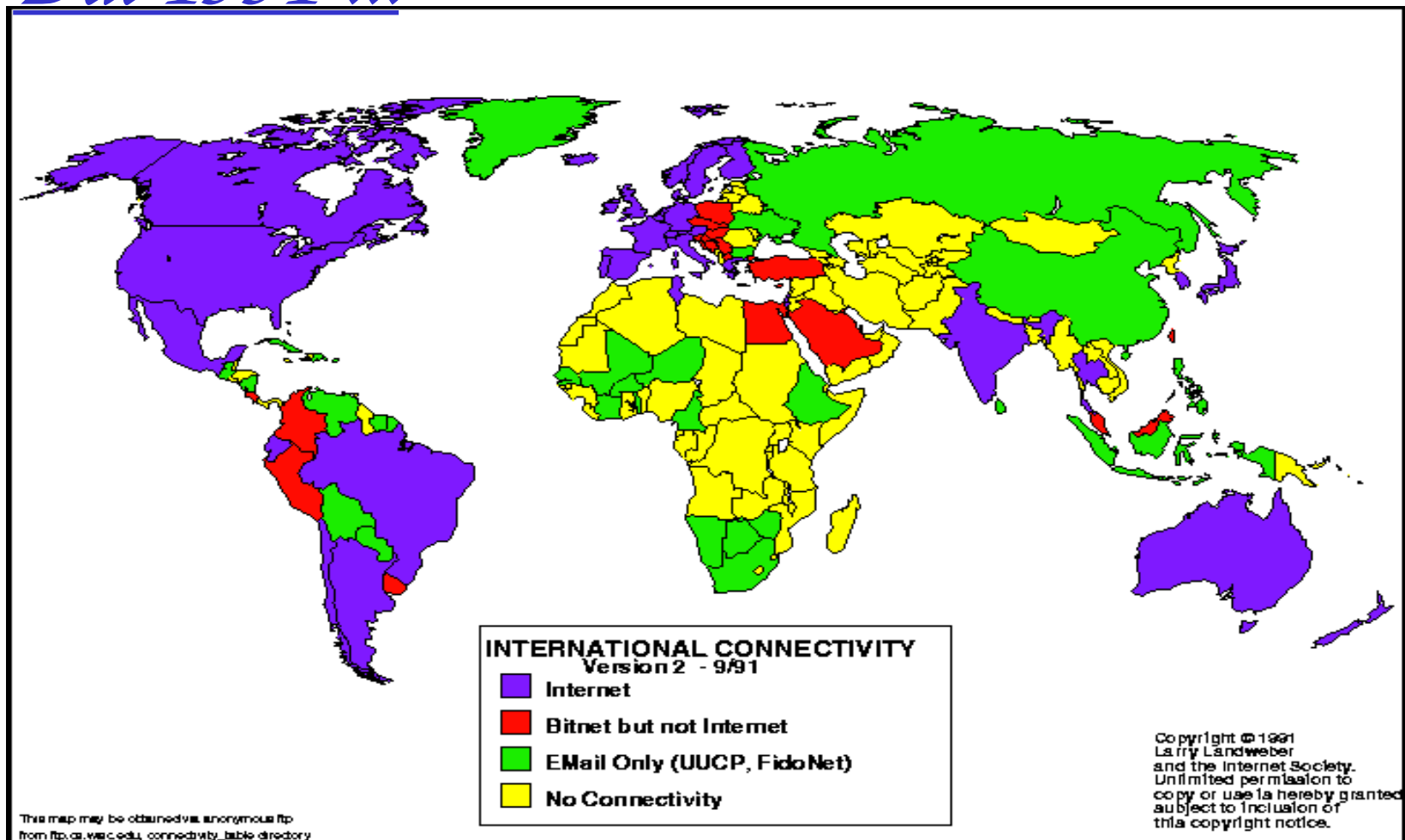
- 1988: CA, DK, FI, FR, IS, NO, SE
- 1989: AU, DE, IL, IT, JP, MX, NL, NZ, PR, UK
- 1990: AR, AT, BE, BR, CL, GR, IN, IE, KR, ES, CH
- 1991: HR, CZ, HK, HU, OL, PT, SG, ZA, TW, TN
- 1992: AQ, CM, CY, EC, EE, KW, LV, LU, MY, SK, SI, TH, VE
- 1993: BG, CR, EG, FJ, GH, GU, ID, KZ, KE, LI, PE, RO, RU, TR, UA, AE, VI
- 1994: ...

# Storia di Internet

## *1990-oggi: commercializzazione, il Web, nuove applicazioni*

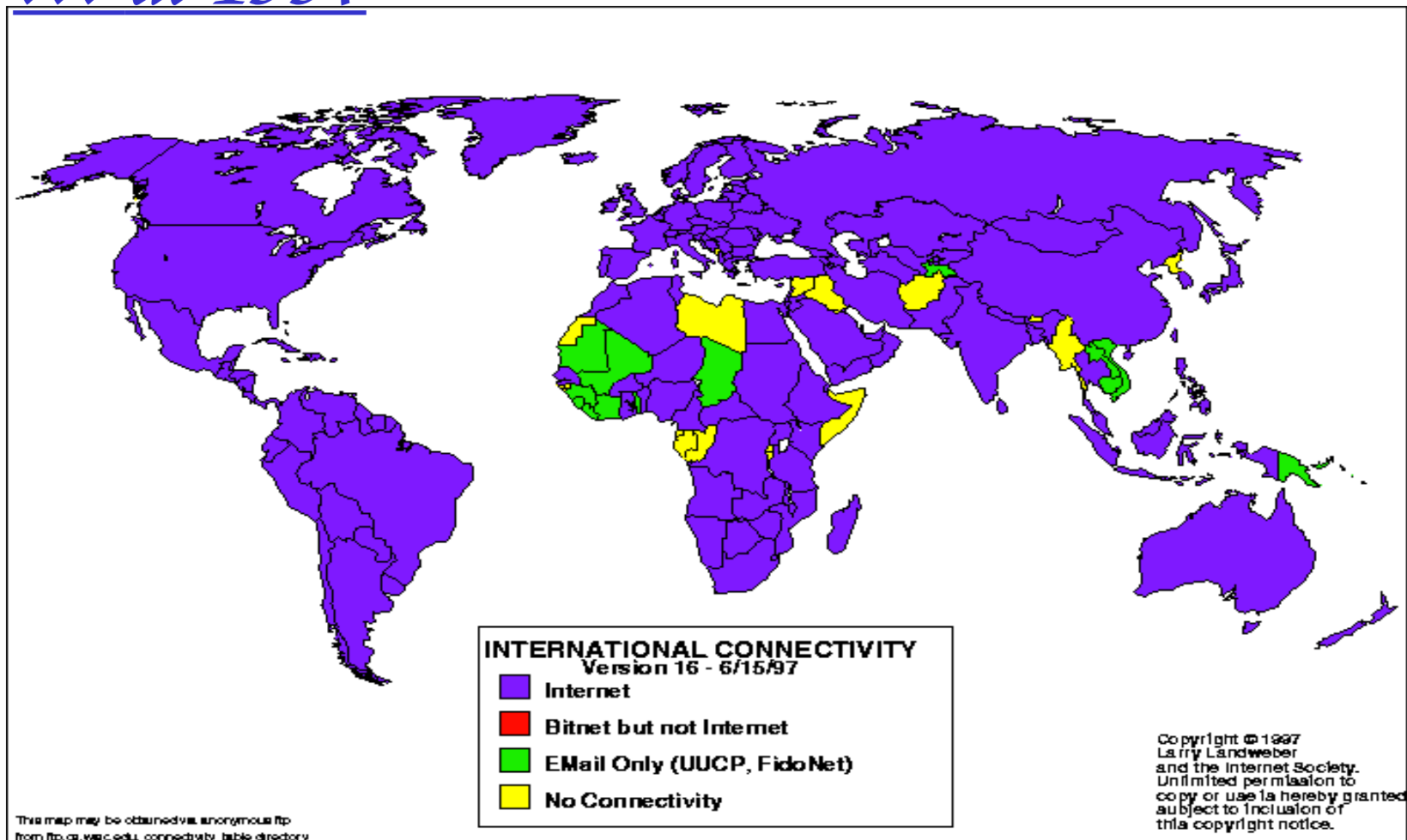
- ❑ *Primi anni '90: ARPAnet viene dismessa*
  - ❑ *1991: NSF lascia decadere le restrizioni sull'uso commerciale di NSFnet*
  - ❑ *Primi anni '90: il Web*
    - ❖ *ipertestualità [Bush 1945, Nelson 1960's]*
    - ❖ *HTML, HTTP: Berners-Lee*
    - ❖ *1994: Mosaic, poi Netscape*
  - ❑ *Fine '90 : commercializzazione del Web*
- Fine anni '90 – 2005:*
    - ❑ *arrivano le “killer applications”:  
messaggistica istantanea, condivisione di file P2P*
    - ❑ *sicurezza di rete*
    - ❑ *50 milioni di host, oltre 100 milioni di utenti*
    - ❑ *velocità nelle dorsali dell'ordine di Gbps*

## *Dal 1991 ...*

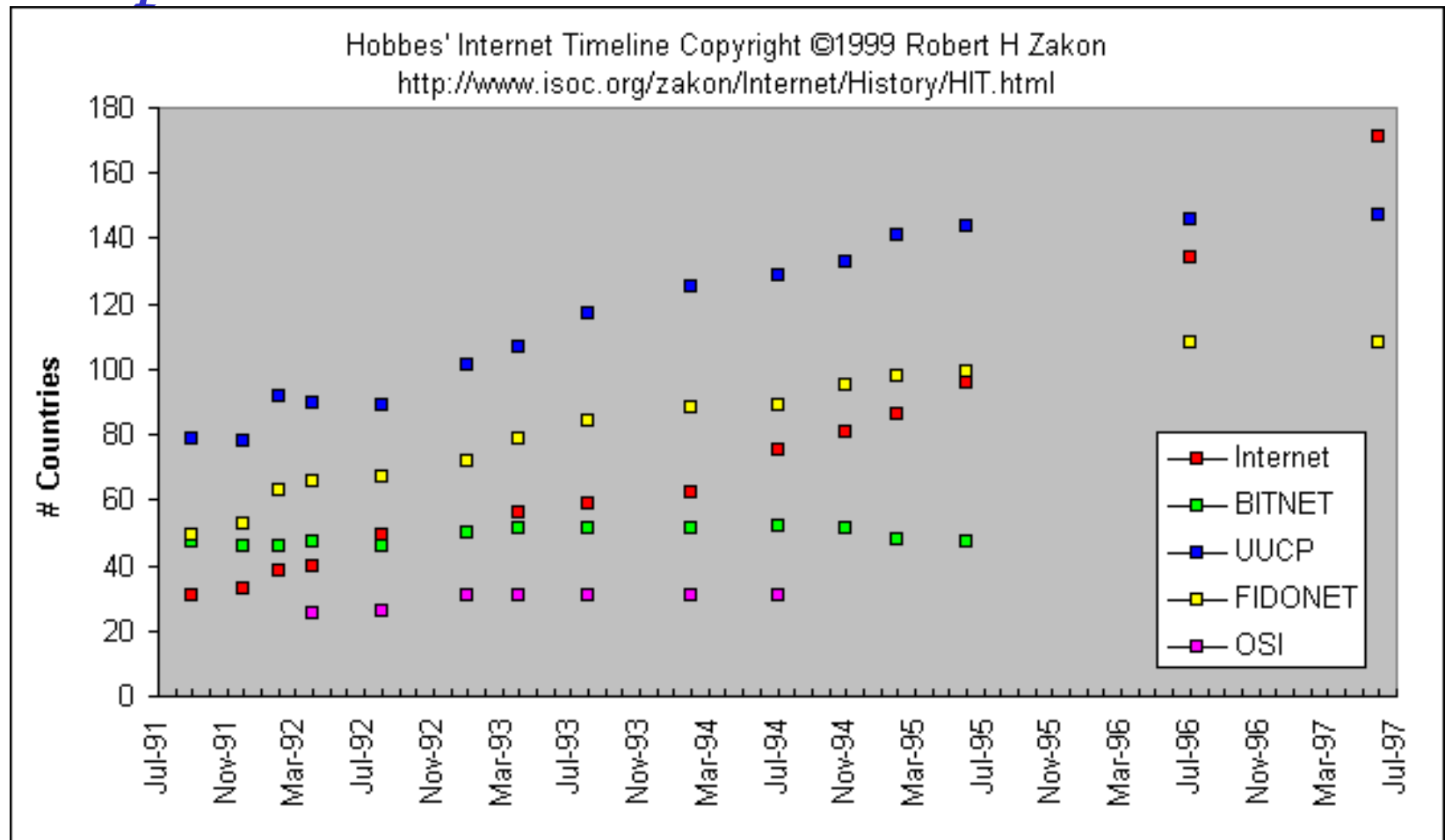




... as 1997



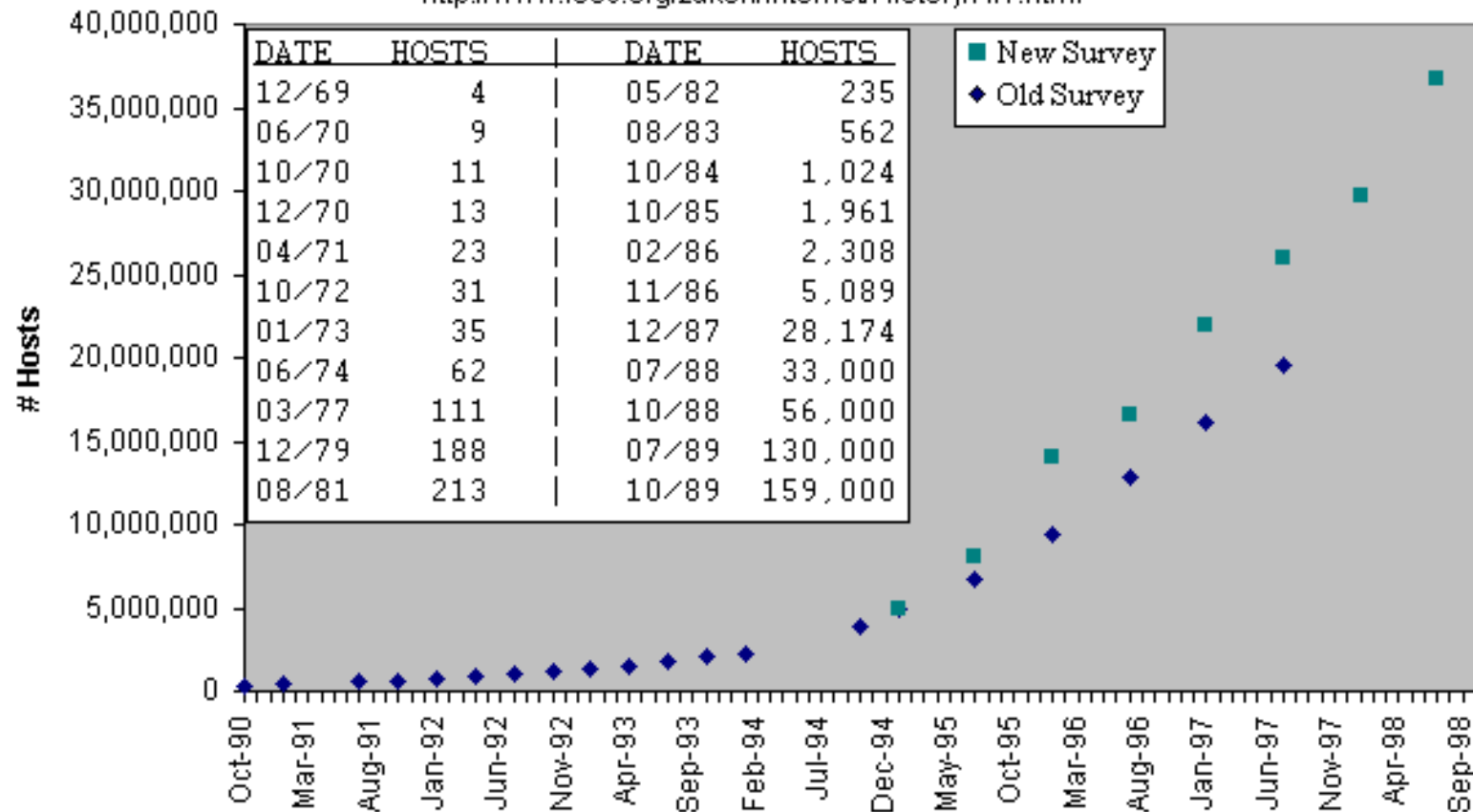
## Un po' di numeri ...



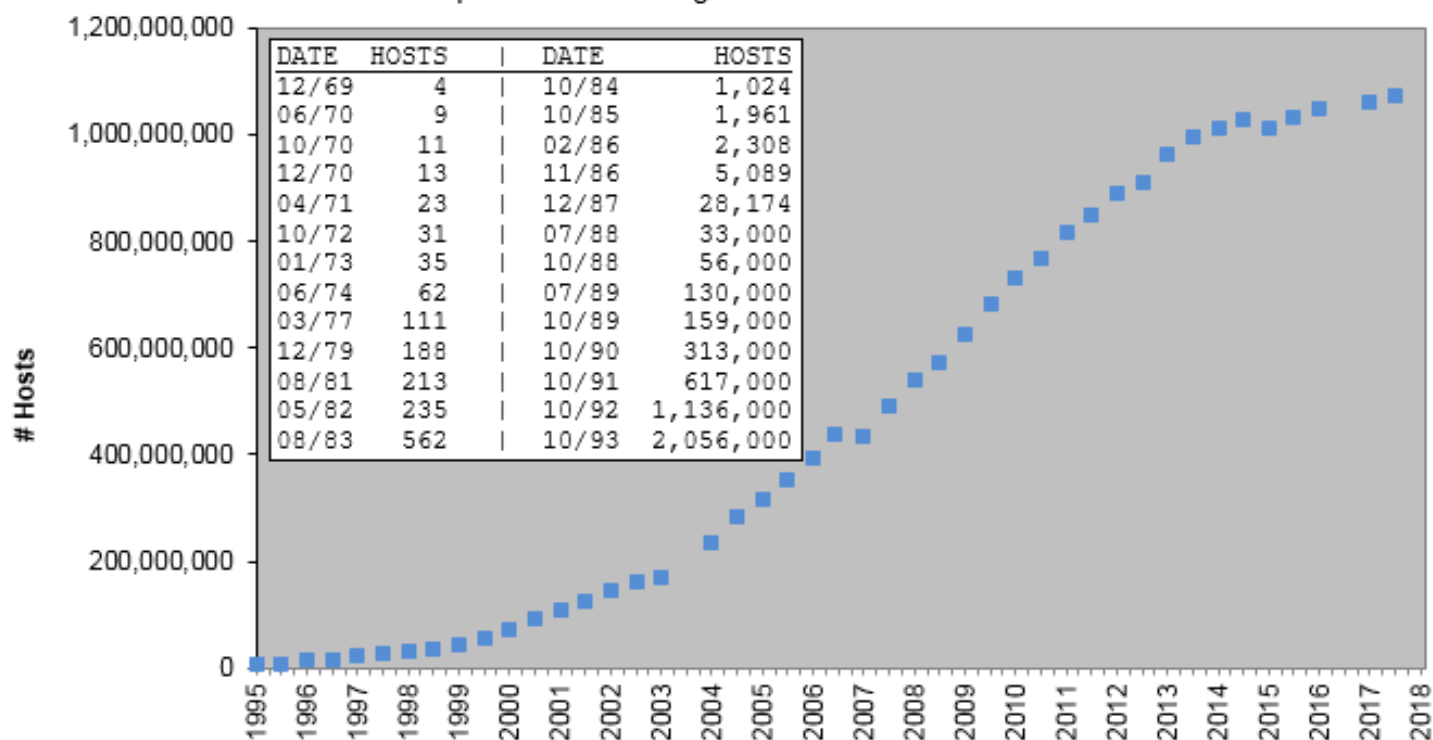
## ... e gli hosts ...

Hobbes' Internet Timeline Copyright ©1999 Robert H Zakon

<http://www.isoc.org/zakon/Internet/History/HIT.html>



Hobbes' Internet Timeline Copyright ©2018 Robert H Zakon  
<https://www.zakon.org/robert/internet/timeline/>



*If  $w\tau w \dots$*

www

facebook

*dns*



# Riassunto

## Abbiamo visto un sacco di argomenti!

- ❑ *Panoramica di Internet*
- ❑ *Cos'è un protocollo?*
- ❑ *Il vasto mondo delle reti*
  - ❖ *Commutazione di pacchetto e commutazione di circuito*
- ❑ *Internet/struttura degli ISP*
- ❑ *Prestazioni: perdite, ritardo*
- ❑ *Stratificazioni e modello di servizio*
- ❑ *Cenni storici*

## Adesso siete in grado di:

- ❑ *contestualizzare, fornire una panoramica sulle reti, avere un'idea precisa di che cosa si intende per "networking"*
- ❑ *maggiori approfondimenti e dettagli nei prossimi capitoli!*

# Capitolo 2

## Livello di applicazione

### Nota per l'utilizzo:

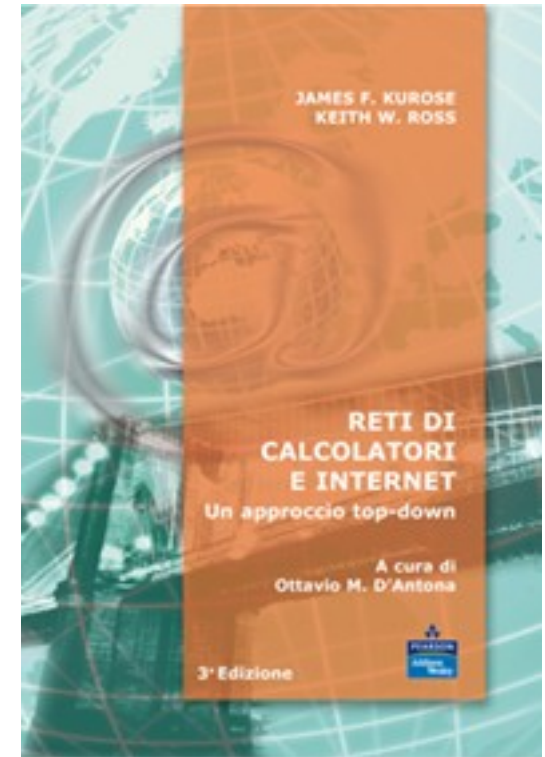
Abbiamo preparato queste slide con l'intenzione di renderle disponibili a tutti (professori, studenti, lettori). Sono in formato PowerPoint in modo che voi possiate aggiungere e cancellare slide (compresa questa) o modificarne il contenuto in base alle vostre esigenze.

Come potete facilmente immaginare, da parte nostra abbiamo fatto *un* sacco di lavoro. In cambio, vi chiediamo solo di rispettare le seguenti condizioni:

- se utilizzate queste slide (ad esempio, in aula) in una forma sostanzialmente inalterata, fate riferimento alla fonte (dopo tutto, ci piacerebbe che la gente usasse il nostro libro!)
- se rendete disponibili queste slide in una forma sostanzialmente inalterata su un sito web, indicate che si tratta di un adattamento (o che sono identiche) delle nostre slide, e inserite la nota relativa al copyright.

*Thanks and enjoy!* JFK/KWR

All material copyright 1996-2005  
J.F Kurose and K.W. Ross, All Rights Reserved



*Reti di calcolatori e Internet: Un  
approccio top-down*

*3ª edizione*

*Jim Kurose, Keith Ross*

*Pearson Education Italia ©2005*

# Capitolo 2: Livello di applicazione

- 2.1 Principi delle applicazioni di rete
- 2.2 Web e HTTP
- 2.3 FTP
- 2.4 Posta elettronica
  - ❖ SMTP, POP3, IMAP
- 2.5 DNS
- 2.6 Condivisione di file P2P
- 2.7 Programmazione delle socket con TCP
- 2.8 Programmazione delle socket con UDP
- 2.9 Costruire un semplice server web

# Capitolo 2: Livello di applicazione

## Obiettivi:

- *Fornire i concetti base e gli aspetti implementativi dei protocolli delle applicazioni di rete*
  - ❖ *modelli di servizio del livello di trasporto*
  - ❖ *paradigma client-server*
  - ❖ *paradigma peer-to-peer*
- *Apprendere informazioni sui protocolli esaminando quelli delle più diffuse applicazioni di rete*
  - ❖ *HTTP*
  - ❖ *FTP*
  - ❖ *SMTP / POP3 / IMAP*
  - ❖ *DNS*

# Alcune diffuse applicazioni di rete

- ❑ *Posta elettronica*
- ❑ *Web*
- ❑ *Messaggistica istantanea*
- ❑ *Autenticazione in un calcolatore remoto (Telnet e SSH)*
- ❑ *Condivisione di file P2P*
- ❑ *Giochi multiutente via rete*
- ❑ *Streaming di video-clip memorizzati*
- ❑ *Telefonia via Internet*
- ❑ *Videoconferenza in tempo reale*

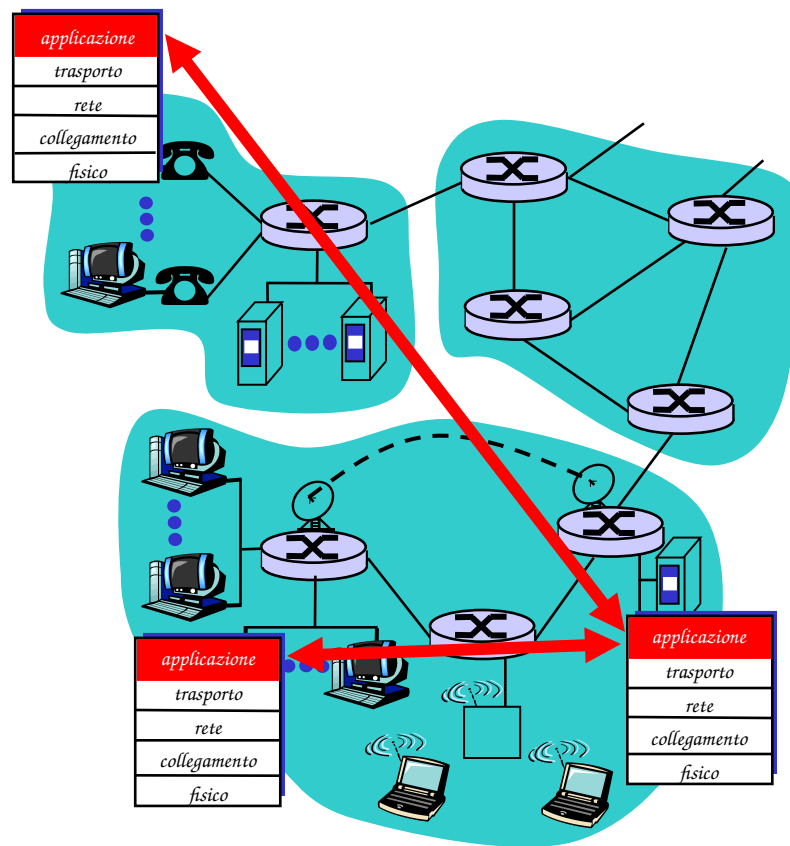
# Creare un'applicazione di rete

## *Scrivere programmi che*

- ❖ *girano su sistemi terminali diversi*
- ❖ *comunicano attraverso la rete*
- ❖ *Ad es. il Web: il software di un server Web comunica con il software di un browser*

## *software in grado di funzionare su più macchine*

- ❖ *non occorre predisporre programmi per i dispositivi del nucleo della rete, quali router o commutatori Ethernet*



# Capitolo 2: Livello di applicazione

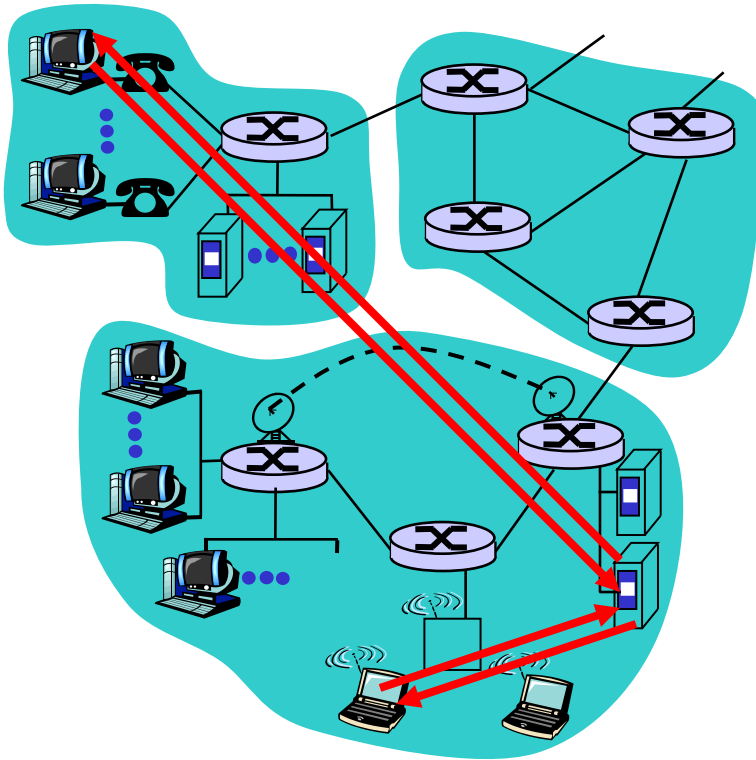
- ❑ *2.1 Principi delle applicazioni di rete*
- ❑ *2.2 Web e HTTP*
- ❑ *2.3 FTP*
- ❑ *2.4 Posta elettronica*
  - ❖ *SMTP, POP3, IMAP*
- ❑ *2.5 DNS*
- ❑ *2.6 Condivisione di file P2P*
- ❑ *2.7 Programmazione delle socket con TCP*
- ❑ *2.8 Programmazione delle socket con UDP*
- ❑ *2.9 Costruire un semplice server web*

# Architetture delle applicazioni di rete

- ❑ *Client-server*
- ❑ *Peer-to-peer (P2P)*
- ❑ *Architetture ibride (client-server e P2P)*



# Architettura client-server



## *server:*

- ❖ *host sempre attivo*
- ❖ *indirizzo IP fisso*
- ❖ *server farm per creare un potente server virtuale*

## *client:*

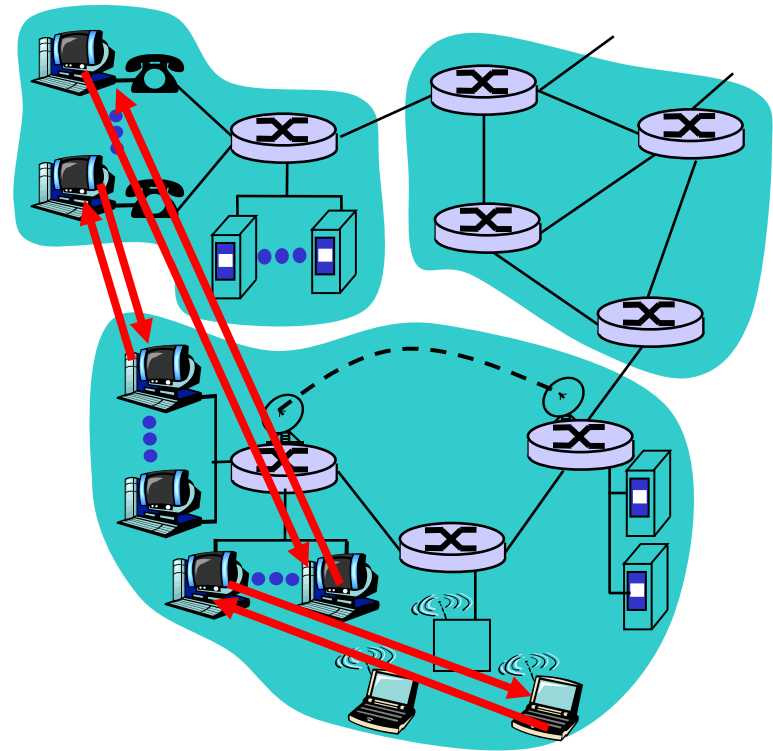
- ❖ *comunica con il server*
- ❖ *può contattare il server in qualunque momento*
- ❖ *può avere indirizzi IP dinamici*
- ❖ *non comunica direttamente con gli altri client*

# Architettura P2P pura

- ❑ *non c'è un server sempre attivo*
- ❑ *coppie arbitrarie di host (peer) comunicano direttamente tra loro*
- ❑ *i peer non devono necessariamente essere sempre attivi, e cambiano indirizzo IP*
- ❑ *Un esempio: Gnutella*

*Facilmente scalabile*

*Difficile da gestire*



# Ibridi (client-server e P2P)

## *Napster*

- ❖ *Scambio di file secondo la logica P2P*
- ❖ *Ricerca di file centralizzata:*
  - *i peer registrano il loro contenuto presso un server centrale*
  - *i peer chiedono allo stesso server centrale di localizzare il contenuto*

## *Messaggistica istantanea*

- ❖ *La chat tra due utenti è del tipo P2P*
- ❖ *Individuazione della presenza/location centralizzata:*
  - *l'utente registra il suo indirizzo IP sul server centrale quando è disponibile online*
  - *l'utente contatta il server centrale per conoscere gli indirizzi IP dei suoi amici*

# Processi comunicanti

***Processo:** programma in esecuzione su di un host.*

- *All'interno dello stesso host, due processi comunicano utilizzando **schemi interprocesso** (definiti dal SO).*
- *processi su host differenti comunicano attraverso lo scambio di **messaggi***

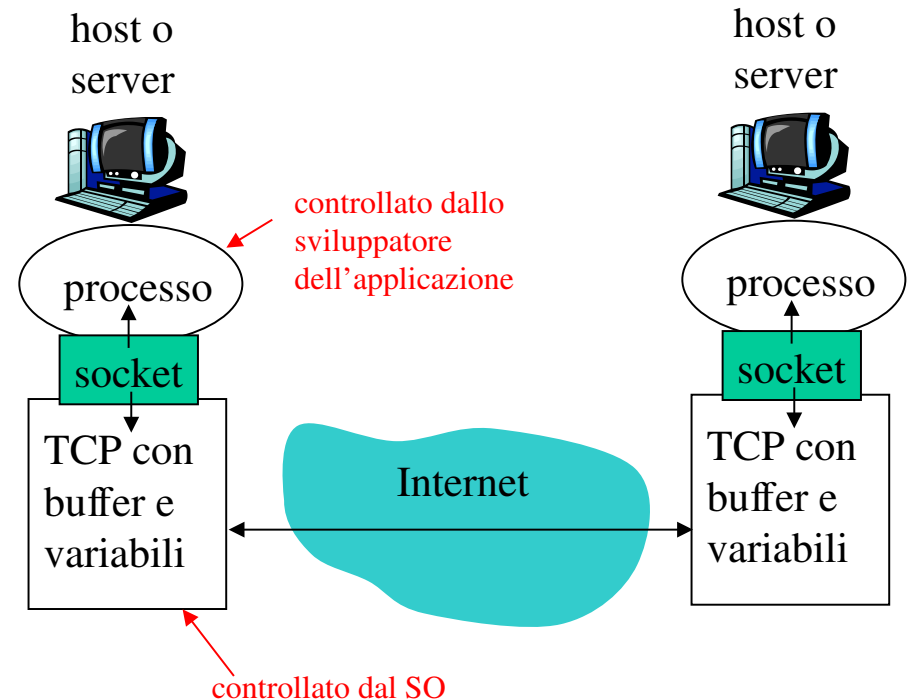
***Processo client:** processo che dà inizio alla comunicazione*

***Processo server :** processo che attende di essere contattato*

- *Nota: le applicazioni con architetture P2P hanno processi client e processi server*

# Socket

- *un processo invia/riceve messaggi a/da la sua **socket***
- *una socket è analoga a una porta*
  - ❖ *un processo che vuole inviare un messaggio, lo fa uscire dalla propria "porta" (socket)*
  - ❖ *il processo presuppone l'esistenza di un'infrastruttura esterna che trasporterà il messaggio attraverso la rete fino alla "porta" del processo di destinazione*



- *API: (1) scelta del protocollo di trasporto; (2) capacità di determinare alcuni parametri*

## Processi di indirizzamento

- *Affinché un processo su un host invii un messaggio a un processo su un altro host, il mittente deve identificare il processo destinatario.*
- *Un host  $A$  ha un indirizzo IP univoco a 32 bit*
- ***D:** È sufficiente conoscere l'indirizzo IP dell'host su cui è in esecuzione il processo per identificare il processo stesso?*
- ***Risposta:** No, sullo stesso host possono essere in esecuzione molti processi.*
- *L'identificatore comprende sia l'indirizzo IP che i **numeri di porta** associati al processo in esecuzione su un host.*
- *Esempi di numeri di porta:*
  - ❖ *HTTP server: 80*
  - ❖ *Mail server: 25*
- *Approfondiremo questi argomenti più avanti*

# Protocollo a livello di applicazione

- *Tipi di messaggi scambiati, ad esempio messaggi di richiesta e di risposta*
- *Sintassi dei tipi di messaggio: quali sono i campi nel messaggio e come sono descritti*
- *Semantica dei campi, ovvero significato delle informazioni nei campi*
- *Regole per determinare quando e come un processo invia e risponde ai messaggi*

## *Protocolli di pubblico dominio:*

- *Definiti nelle RFC*
- *Consente l'interoperabilità*
- *Ad esempio, HTTP, SMTP*

## *Protocolli proprietari:*

- *Ad esempio, KaZaA*

# Quale servizio di trasporto richiede un'applicazione?

## *Perdita di dati*

- ❑ alcune applicazioni (ad esempio, audio) possono tollerare qualche perdita
- ❑ altre applicazioni (ad esempio, trasferimento di file, telnet) richiedono un trasferimento dati affidabile al 100%

## *Temporizzazione*

- ❑ alcune applicazioni (ad esempio, telefonia Internet, giochi interattivi) per essere "realistiche" richiedono piccoli ritardi

## *Ampiezza di banda*

- ❑ alcune applicazioni (ad esempio, quelle multimediali) per essere "efficaci" richiedono un'ampiezza di banda minima
- ❑ altre applicazioni ("le applicazioni elastiche") utilizzano l'ampiezza di banda che si rende disponibile



## Requisiti del servizio di trasporto di alcune applicazioni comuni

<b>Applicazione</b>	<b>Tolleranza alla perdita di dati</b>	<b>Ampiezza di banda</b>	<b>Sensibilità al tempo</b>
Trasferimento file	No	Variabile	No
Posta elettronica	No	Variabile	No
Documenti Web	No	Variabile	No
Audio/video in tempo reale	Sì	Audio: da 5 Kbps a 1 Mbps Video: da 10 Kbps a 5 Mbps	Sì, centinaia di ms
Audio/video memorizzati	Sì	Come sopra	Sì, pochi secondi
Giochi interattivi	Sì	Fino a pochi Kbps	Sì, centinaia di ms
Messaggistica istantanea	No	Variabile	Sì e no

# Servizi dei protocolli di trasporto Internet

## Servizio di TCP:

- ❑ *orientato alla connessione:* è richiesto un setup fra i processi client e server
- ❑ *trasporto affidabile* fra i processi d'invio e di ricezione
- ❑ *controllo di flusso:* il mittente non vuole sovraccaricare il destinatario
- ❑ *controllo della congestione:* “strozza” il processo d'invio quando la rete è sovraccaricata
- ❑ *non offre:* temporizzazione, ampiezza di banda minima

## Servizio di UDP:

- ❑ *trasferimento dati inaffidabile* fra i processi d'invio e di ricezione
- ❑ *non offre:* setup della connessione, affidabilità, controllo di flusso, controllo della congestione, temporizzazione né ampiezza di banda minima

D: Perché preoccuparsi? Perché esiste UDP?

## Applicazioni Internet: protocollo a livello applicazione e protocollo di trasporto

<b>Applicazione</b>	<b>Protocollo a livello applicazione</b>	<b>Protocollo di trasporto sottostante</b>
Posta elettronica	SMTP [RFC 2821]	TCP
Accesso a terminali remoti	Telnet [RFC 854]	TCP
Web	HTTP [RFC 2616]	TCP
Trasferimento file	FTP [RFC 959]	TCP
Multimedia in streaming	Proprietario (ad esempio, RealNetworks)	TCP o UDP
Telefonia Internet	Proprietario (ad esempio, Vonage, Dialpad)	Tipicamente UDP

# Capitolo 2: Livello di applicazione

- ❑ 2.1 Principi delle applicazioni di rete
  - ❖ Architetture delle applicazioni
  - ❖ Servizi richiesti dalle applicazioni
- ❑ 2.2 Web e HTTP
- ❑ 2.3 FTP
- ❑ 2.4 Posta elettronica
  - ❖ SMTP, POP3, IMAP
- ❑ 2.5 DNS
- ❑ 2.6 Condivisione di file P2P
- ❑ 2.7 Programmazione delle socket con TCP
- ❑ 2.8 Programmazione delle socket con UDP
- ❑ 2.9 Costruire un semplice server web

# Web e HTTP

## Terminologia

- Una *pagina web* è costituita da *oggetti*
- Un oggetto può essere un file *HTML*, un'immagine *JPEG*, un'applet *Java*, un file audio, ...
- Una pagina web è formata da un *file base HTML* che include diversi oggetti referenziati
- Ogni oggetto è referenziato da un *URL*
- Esempio di URL:

www.someschool.edu / someDept/pic.gif

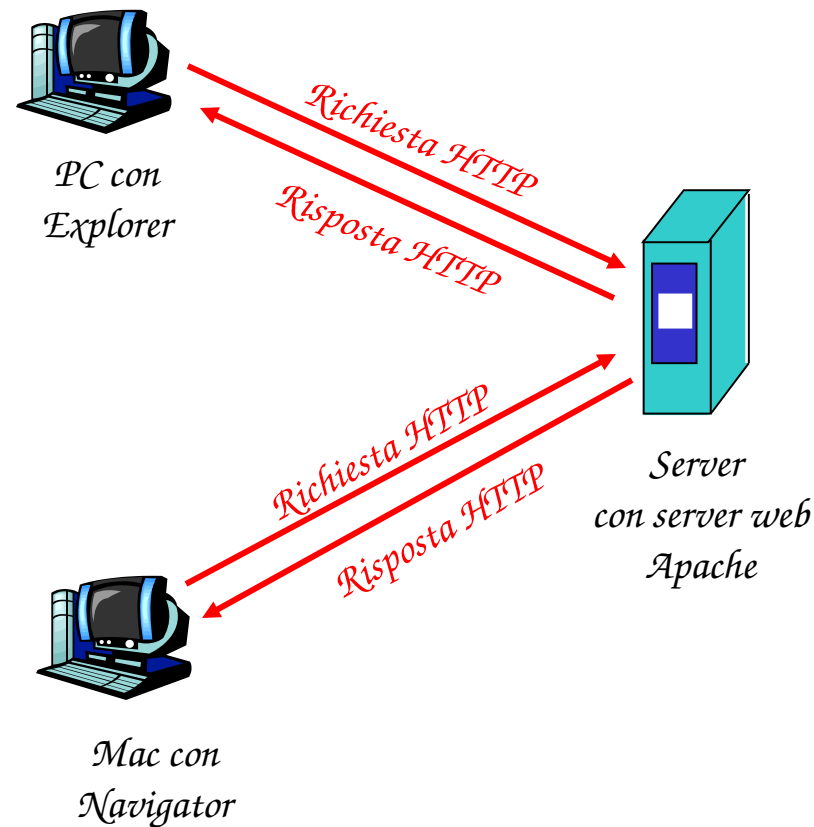
*nome dell'host*

*nome del percorso*

# Panoramica su HTTP

## *HTTP: hypertext transfer protocol*

- Protocollo a livello di applicazione del Web
- Modello client/server
  - ❖ *client*: il browser che richiede, riceve, “visualizza” gli oggetti del Web
  - ❖ *server*: il server web invia oggetti in risposta a una richiesta
- HTTP 1.0: RFC 1945
- HTTP 1.1: RFC 2068



# Panoramica su HTTP (continua)

## *Usa TCP:*

- *Il client inizializza la connessione TCP (crea una socket) con il server, la porta 80*
- *Il server accetta la connessione TCP dal client*
- *Messaggi HTTP scambiati fra browser (client HTTP) e server web (server HTTP)*
- *Connessione TCP chiusa*

## *HTTP è un protocollo “senza stato” (stateless)*

- *Il server non mantiene informazioni sulle richieste fatte dal client*

*nota*

*I protocolli che mantengono lo “stato” sono complessi!*

- *La storia passata (stato) deve essere memorizzata*
- *Se il server e/o il client si bloccano, le loro viste dello “stato” potrebbero essere contrastanti e dovrebbero essere riconciliate*

# Connessioni HTTP

## Connessioni non persistenti

- ❑ *Almeno un oggetto viene trasmesso su una connessione TCP*
- ❑ *HTTP/1.0 usa connessioni non persistenti*

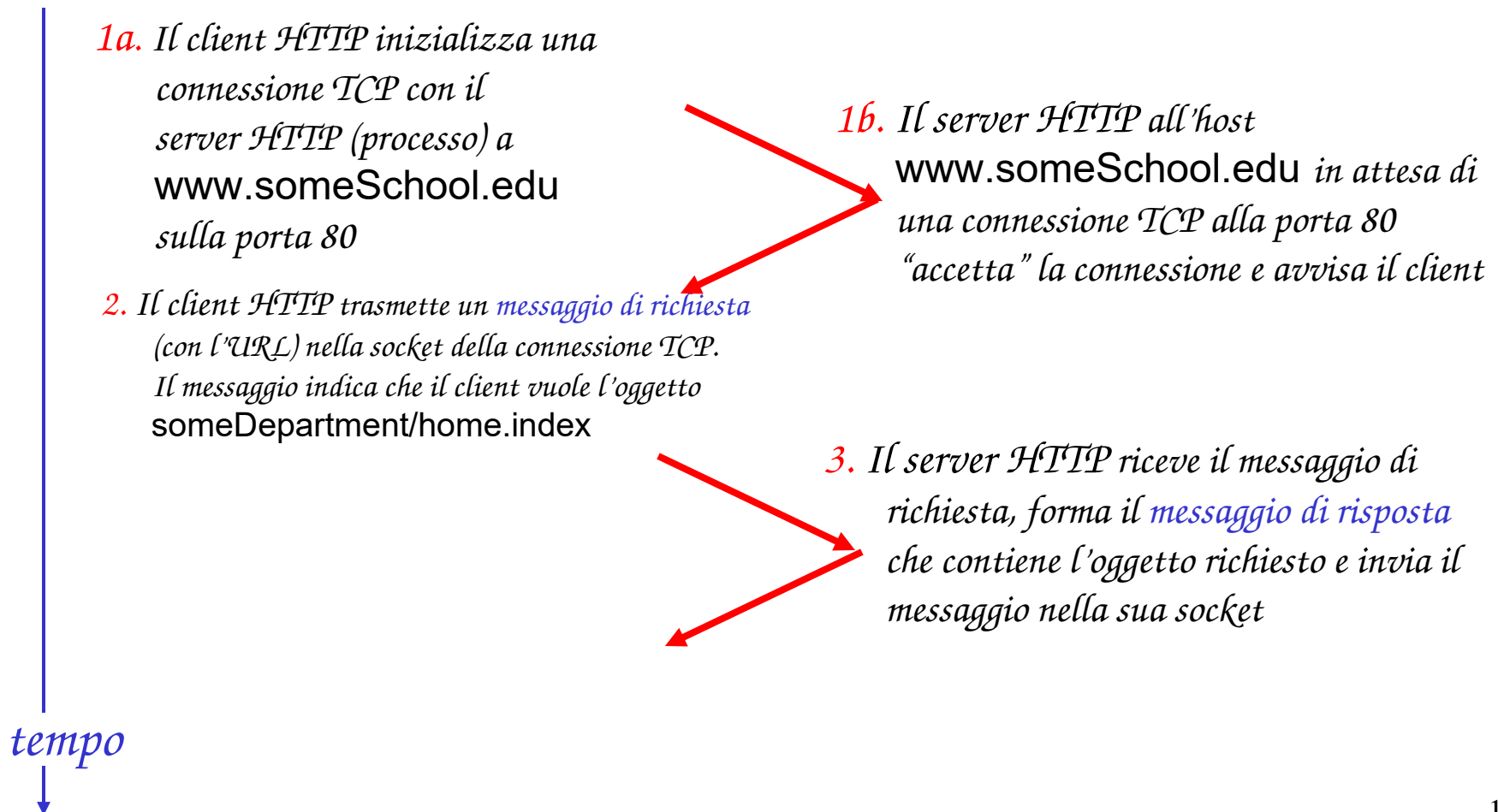
## Connessioni persistenti

- ❑ *Più oggetti possono essere trasmessi su una singola connessione TCP tra client e server*
- ❑ *HTTP/1.1 usa connessioni persistenti nella modalità di default*

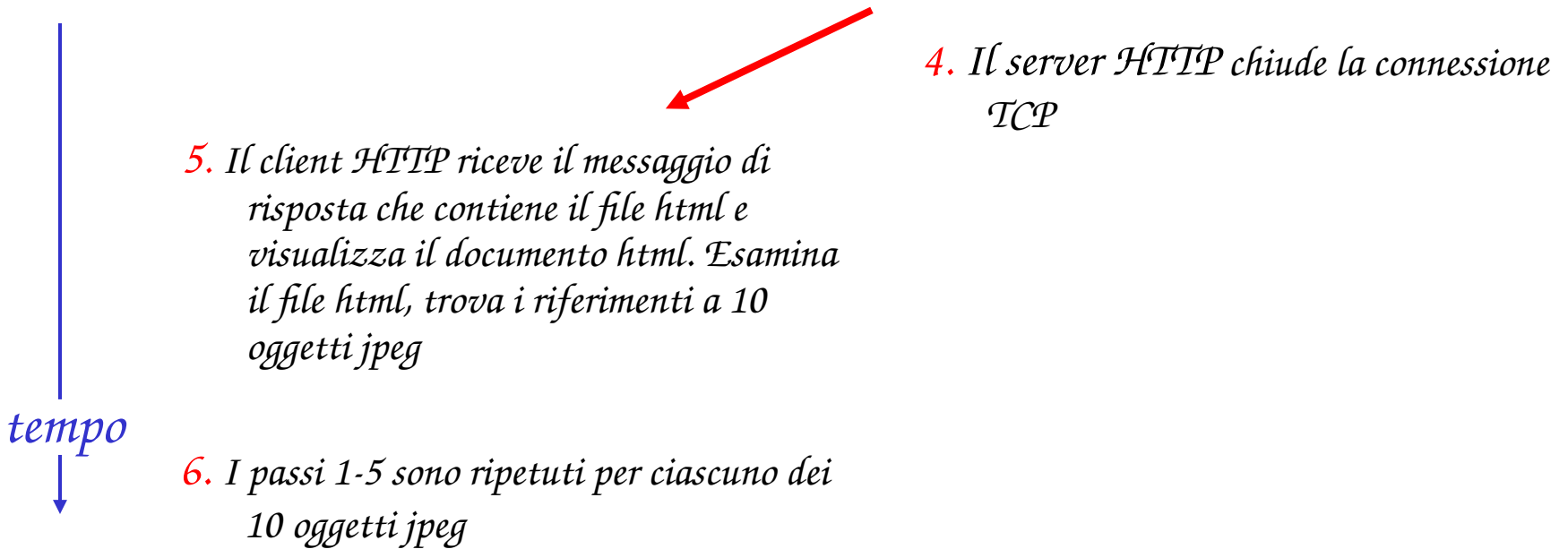


# Connessioni non persistenti

Supponiamo che l'utente immetta l'URL `www.someSchool.edu/someDepartment/home.index` (contiene testo, riferimenti a 10 immagini jpeg)



## Connessioni non persistenti (continua)



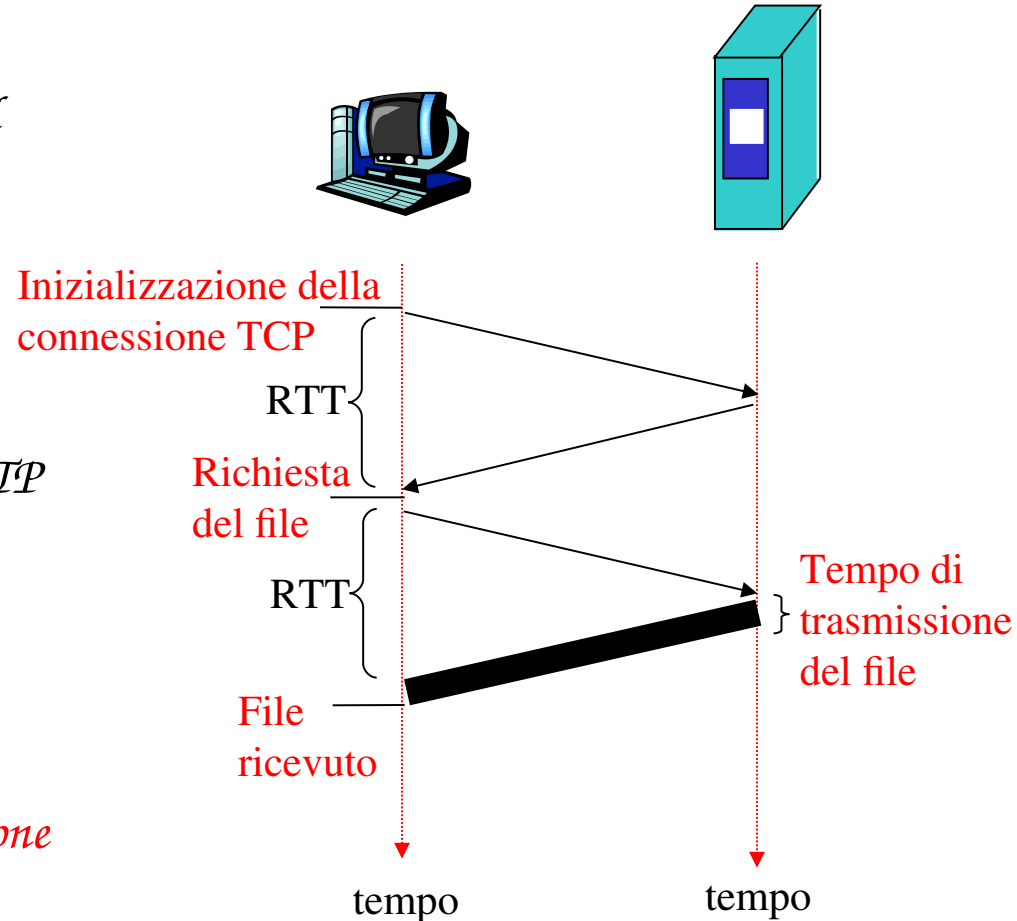
# Schema del tempo di risposta

*Definizione di  $\mathcal{RTT}$ : tempo impiegato da un piccolo pacchetto per andare dal client al server e ritornare al client.*

## Tempo di risposta:

- un  $\mathcal{RTT}$  per inizializzare la connessione TCP
- un  $\mathcal{RTT}$  perché ritornino la richiesta HTTP e i primi byte della risposta HTTP
- tempo di trasmissione del file

$$\text{totale} = 2\mathcal{RTT} + \text{tempo di trasmissione}$$



# Connessioni persistenti

## Svantaggi delle connessioni non persistenti:

- ❑ richiede 2  $RTT$  per oggetto
- ❑ overhead del sistema operativo per ogni connessione  $TCP$
- ❑ i browser spesso aprono connessioni  $TCP$  parallele per caricare gli oggetti referenziati

## Connessioni persistenti

- ❑ il server lascia la connessione  $TCP$  aperta dopo l'invio di una risposta
- ❑ i successivi messaggi tra gli stessi client/server vengono trasmessi sulla connessione aperta

## Connessione persistente

### senza pipelining:

- ❑ il client invia una nuova richiesta solo quando ha ricevuto la risposta precedente
- ❑ un  $RTT$  per ogni oggetto referenziato

## Connessione persistente

### con pipelining:

- ❑ è la modalità di default in  $HTTP/1.1$
- ❑ il client invia le richieste non appena incontra un oggetto referenziato
- ❑ un solo  $RTT$  per tutti gli oggetti referenziati

# *Non Persistente vs persistente*

## Persistente: non pipeling vs pipeling

# Messaggi HTTP

- ❑ due tipi di messaggi HTTP: *richiesta, risposta*
- ❑ *Messaggio di richiesta HTTP:*
  - ❖ ASCII (formato leggibile dall'utente)

*Riga di richiesta  
(comandi GET,  
POST, HEAD)*

*Righe di  
intestazione*

```
GET /somedir/page.html HTTP/1.1
Host: www.someschool.edu
User-agent: Mozilla/4.0
Connection: close
Accept-language: fr
```

*Un carriage return  
e un line feed  
indicano la fine  
del messaggio*

(carriage return e line feed extra)

# Messaggio di richiesta HTTP: formato generale



# Upload dell'input di un form

## Metodo Post:

- ❑ *La pagina web spesso include un form per l'input dell'utente*
- ❑ *L'input arriva al server nel corpo dell'entità*

## Metodo URL:

- ❑ *Usa il metodo GET*
- ❑ *L'input arriva al server nel campo URL della riga di richiesta:*

`www.somesite.com/animalsearch?monkeys&banana`

# Tipi di metodi

## HTTP/1.0

- ❑ *GET*
- ❑ *POST*
- ❑ *HEAD*
  - ❖ *chiede al server di escludere l'oggetto richiesto dalla risposta*

## HTTP/1.1

- ❑ *GET, POST, HEAD*
- ❑ *PUT*
  - ❖ *include il file nel corpo dell'entità e lo invia al percorso specificato nel campo URL*
- ❑ *DELETE*
  - ❖ *cancella il file specificato nel campo URL*

# Messaggio di risposta HTTP

*Riga di stato  
(protocollo  
codice di stato  
espressione di stato)*

*Righe di  
intestazione*

**HTTP/1.1 200 OK**

**Connection close**

**Date: Thu, 06 Aug 1998 12:00:15 GMT**

**Server: Apache/1.3.0 (Unix)**

**Last-Modified: Mon, 22 Jun 1998 ...**

**Content-Length: 6821**

**Content-Type: text/html**

*dati, ad esempio  
il file HTML  
richiesto*

**dati dati dati dati dati ...**

# Codici di stato della risposta HTTP

*Nella prima riga nel messaggio di risposta server->client.*

*Alcuni codici di stato e relative espressioni:*

## **200 OK**

- ❖ *La richiesta ha avuto successo; l'oggetto richiesto viene inviato nella risposta*

## **301 Moved Permanently**

- ❖ *L'oggetto richiesto è stato trasferito; la nuova posizione è specificata nell'intestazione **Location:** della risposta*

## **400 Bad Request**

- ❖ *Il messaggio di richiesta non è stato compreso dal server*

## **404 Not Found**

- ❖ *Il documento richiesto non si trova su questo server*

## **505 HTTP Version Not Supported**

- ❖ *Il server non ha la versione di protocollo HTTP*

## Provate HTTP (lato client)

1. Collegatevi via Telnet al vostro server web preferito:

**telnet cis.poly.edu 80**

Apri una connessione TCP alla porta 80 (porta di default per un server HTTP) dell'host cis.poly.edu.

Tutto ciò che digitate viene trasmesso alla porta 80 di cis.poly.edu

2. Digitate una richiesta GET:

**GET /~ross/ HTTP/1.1**  
**Host: cis.poly.edu**

Digitando questo (premete due volte il tasto Invio), trasmettete una richiesta GET minima (ma completa) al server HTTP

3. Guardate il messaggio di risposta trasmesso dal server HTTP!

# Interazione utente-server: i cookie

*Molti dei più importanti siti web usano  
i cookie*

## Quattro componenti:

- 1) Una riga di intestazione  
nel messaggio di risposta HTTP*
- 2) Una riga di intestazione nel  
messaggio di richiesta HTTP*
- 3) Un file cookie mantenuto sul sistema  
terminale dell'utente e gestito dal  
browser dell'utente*
- 4) Un database sul sito*

## Esempio:

- ❖ Susan accede sempre a Internet  
dallo stesso PC*
- ❖ Visita per la prima volta un  
particolare sito di commercio  
elettronico*
- ❖ Quando la richiesta HTTP  
iniziale giunge al sito, il sito  
crea un identificativo unico  
(ID) e una entry nel database  
per ID*

## Cookie (continua)



# Cookie (continua)

## Cosa possono contenere i cookie:

- ☐ autorizzazione
- ☐ carta per acquisti
- ☐ raccomandazioni
- ☐ stato della sessione dell'utente (e-mail)

*nota*

## Cookie e privacy:

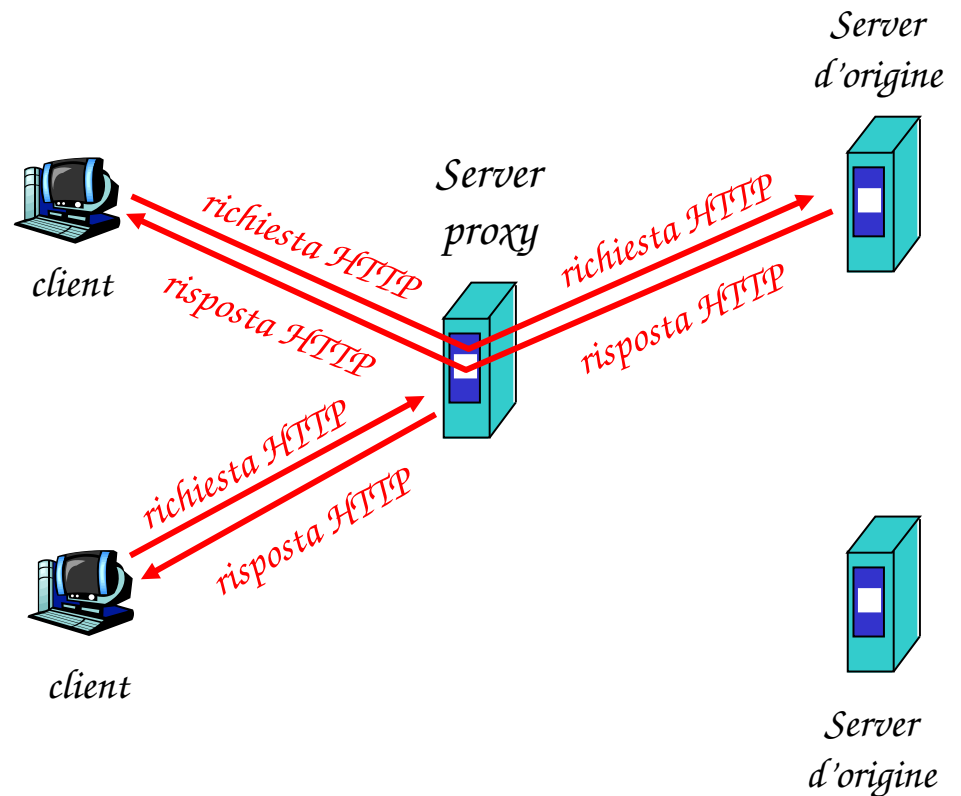
- ☐ i cookie permettono ai siti di imparare molte cose sugli utenti
- ☐ l'utente può fornire al sito il nome e l'indirizzo e-mail
- ☐ i motori di ricerca usano il reindirizzamento e i cookie per sapere ancora di più
- ☐ le agenzie pubblicitarie ottengono informazioni dai siti



# Cache web (server proxy)

**Obiettivo:** soddisfare la richiesta del client senza coinvolgere il server d'origine

- L'utente configura il browser: accesso al Web tramite la cache
- Il browser trasmette tutte le richieste HTTP alla cache
  - ❖ oggetto nella cache: la cache fornisce l'oggetto
  - ❖ altrimenti la cache richiede l'oggetto al server d'origine e poi lo inoltra al client



## Cache web (continua)

- ❑ *La cache opera come client e come server*
- ❑ *Tipicamente la cache è installata da un ISP (università, aziende o ISP residenziali)*

### Perché il caching web?

- ❑ *Riduce i tempi di risposta alle richieste dei client.*
- ❑ *Riduce il traffico sul collegamento di accesso a Internet.*
- ❑ *Internet arricchita di cache consente ai provider “scadenti” di fornire dati con efficacia (ma così fa la condivisione di file P2P)*

# GET condizionale

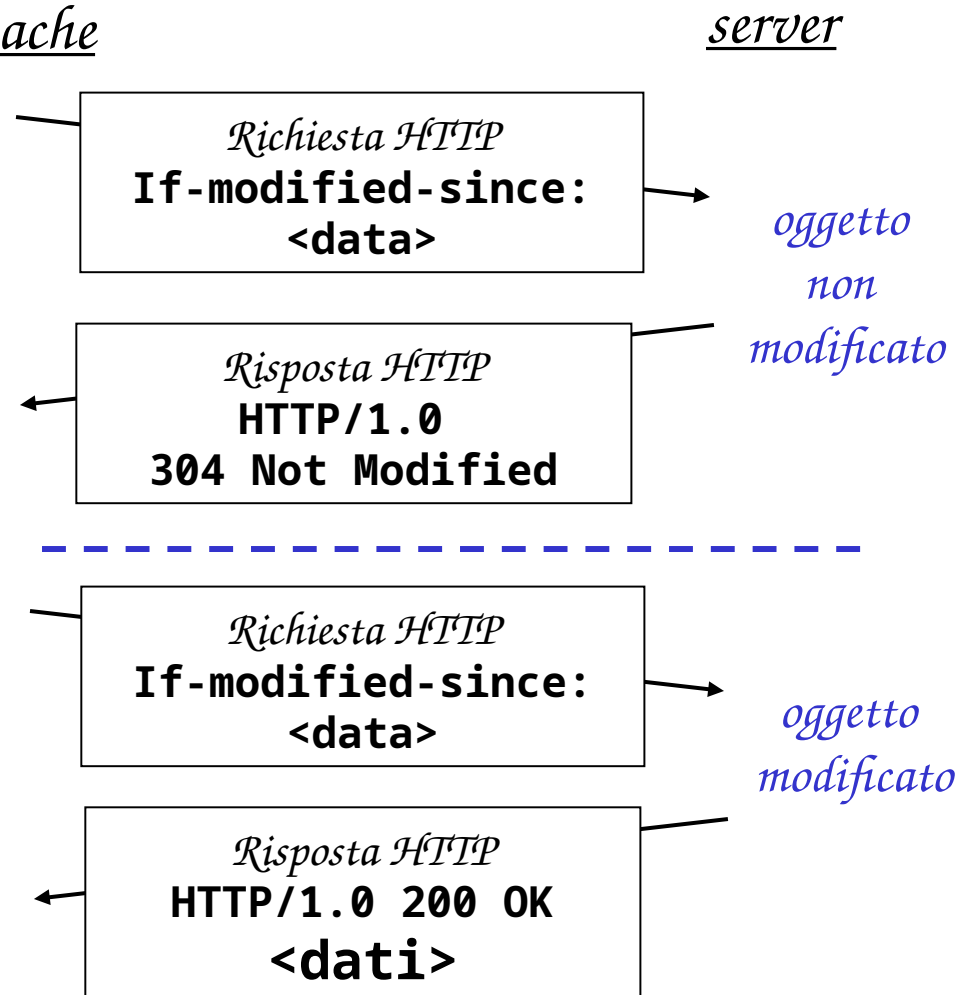
- **Obiettivo:** non inviare un oggetto se la cache server ha una copia aggiornata dell'oggetto

- *cache:* specifica la data della copia dell'oggetto nella richiesta HTTP

**If-modified-since:  
<data>**

- *server:* la risposta non contiene l'oggetto se la copia nella cache è aggiornata:

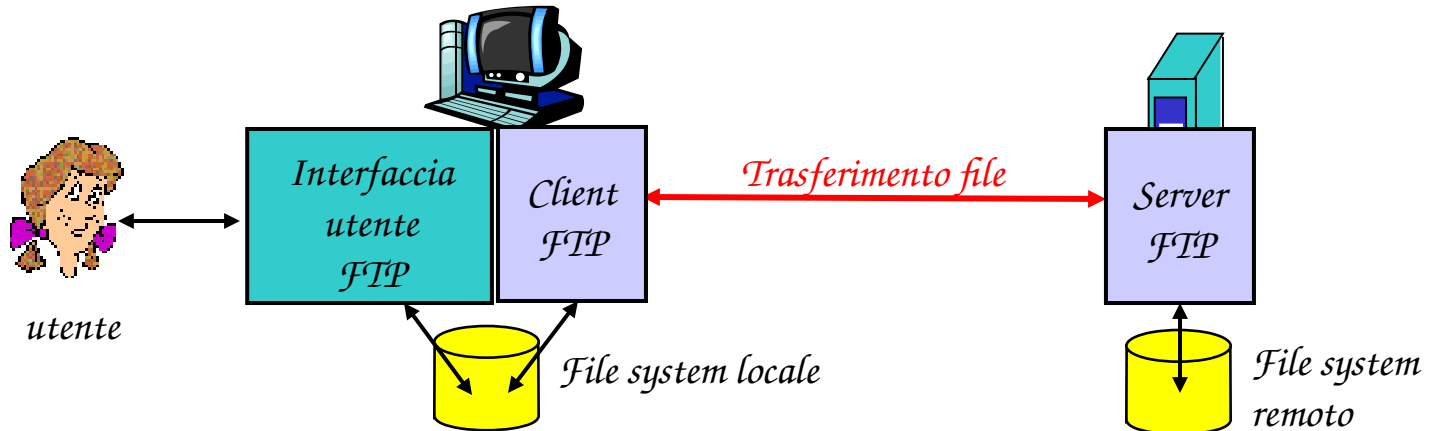
**HTTP/1.0 304 Not  
Modified**



# Capitolo 2: Livello di applicazione

- ❑ 2.1 Principi delle applicazioni di rete
- ❑ 2.2 Web e HTTP
- ❑ 2.3 FTP
- ❑ 2.4 Posta elettronica
  - ❖ SMTP, POP3, IMAP
- ❑ 2.5 DNS
- ❑ 2.6 Condivisione di file P2P
- ❑ 2.7 Programmazione delle socket con TCP
- ❑ 2.8 Programmazione delle socket con UDP
- ❑ 2.9 Costruire un semplice server web

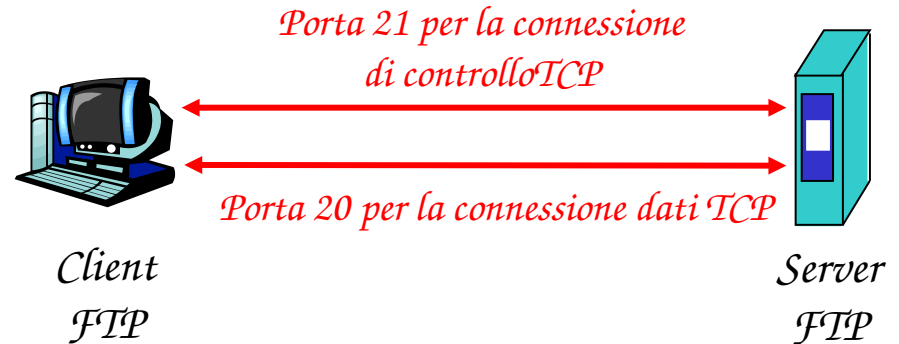
# FTP: file transfer protocol



- ❑ *Trasferimento file a/da un host remoto*
- ❑ *Modello client/server*
  - ❖ *client*: il lato che inizia il trasferimento (a/da un host remoto)
  - ❖ *server*: host remoto
- ❑ *ftp: RFC 959*
- ❑ *server ftp: porta 21*

# FTP: connessione di controllo, connessione dati

- Il client FTP contatta il server FTP alla porta 21, specificando TCP come protocollo di trasporto
- Il client ottiene l'autorizzazione sulla connessione di controllo
- Il client cambia la directory remota inviando i comandi sulla connessione di controllo
- Quando il server riceve un comando per trasferire un file, apre una connessione dati TCP con il client
- Dopo il trasferimento di un file, il server chiude la connessione



- Il server apre una seconda connessione dati TCP per trasferire un altro file.
- Connessione di controllo: *"fuori banda"* (out of band)
- Il server FTP mantiene lo "stato": directory corrente, autenticazione precedente

# Comandi e risposte FTP

## Comandi comuni:

- ❑ *Inviati come testo ASCII sulla connessione di controllo*
- ❑ **USER *username***
- ❑ **PASS *password***
- ❑ **LIST**  
*elenca i file della directory corrente*
- ❑ **RETR *filename***  
*recupera (get) un file dalla directory corrente*
- ❑ **STOR *filename*** *memorizza (put) un file nell'host remoto*

## Codici di ritorno comuni:

- ❑ *Codice di stato ed espressione (come in HTTP)*
- ❑ **331 Username OK, password required**
- ❑ **125 data connection already open; transfer starting**
- ❑ **425 Can't open data connection**
- ❑ **452 Error writing file**

# Capitolo 2: Livello di applicazione

- ❑ 2.1 Principi delle applicazioni di rete
- ❑ 2.2 Web e HTTP
- ❑ 2.3 FTP
- ❑ 2.4 Posta elettronica
  - ❖ SMTP, POP3, IMAP
- ❑ 2.5 DNS
- ❑ 2.6 Condivisione di file P2P
- ❑ 2.7 Programmazione delle socket con TCP
- ❑ 2.8 Programmazione delle socket con UDP
- ❑ 2.9 Costruire un semplice server web



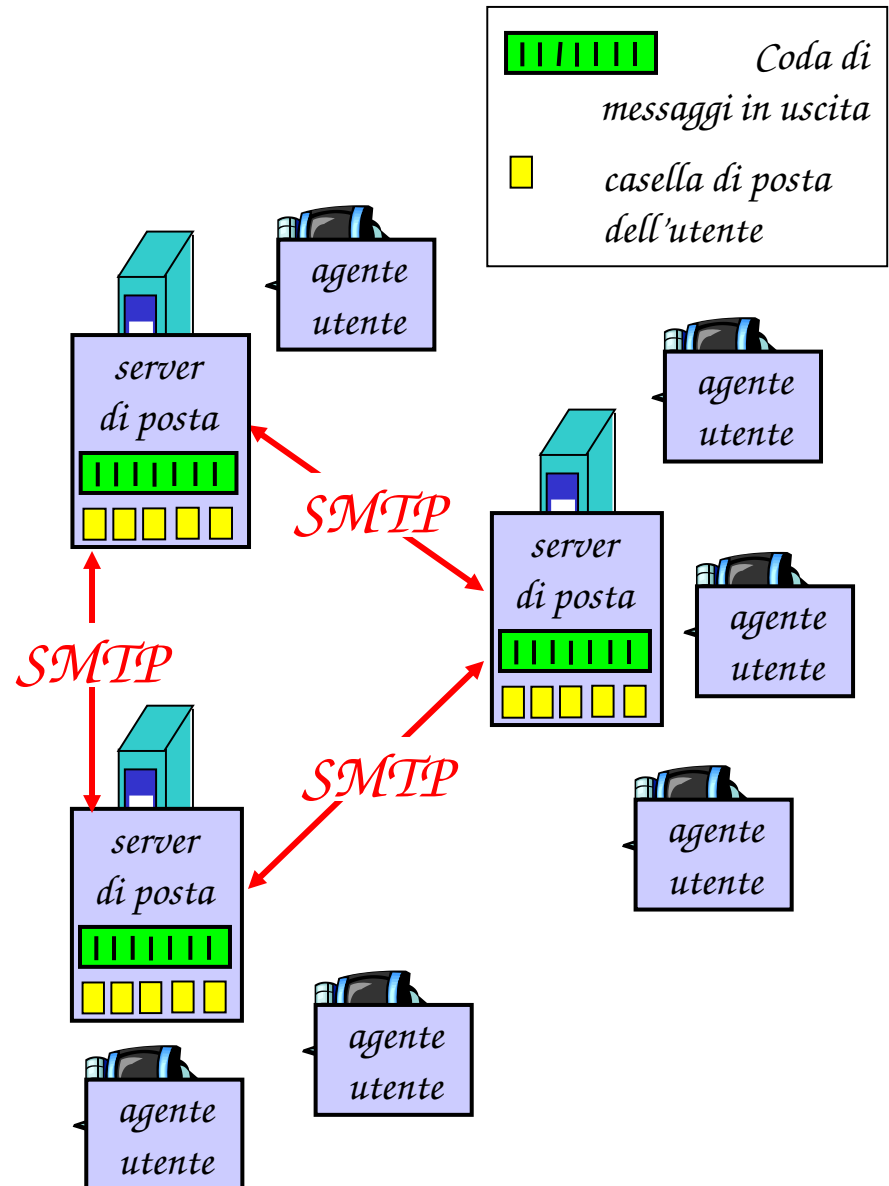
# Posta elettronica

## *Tre componenti principali:*

- ❑ agente utente
- ❑ server di posta
- ❑ simple mail transfer protocol: *SMTP*

## Agente utente

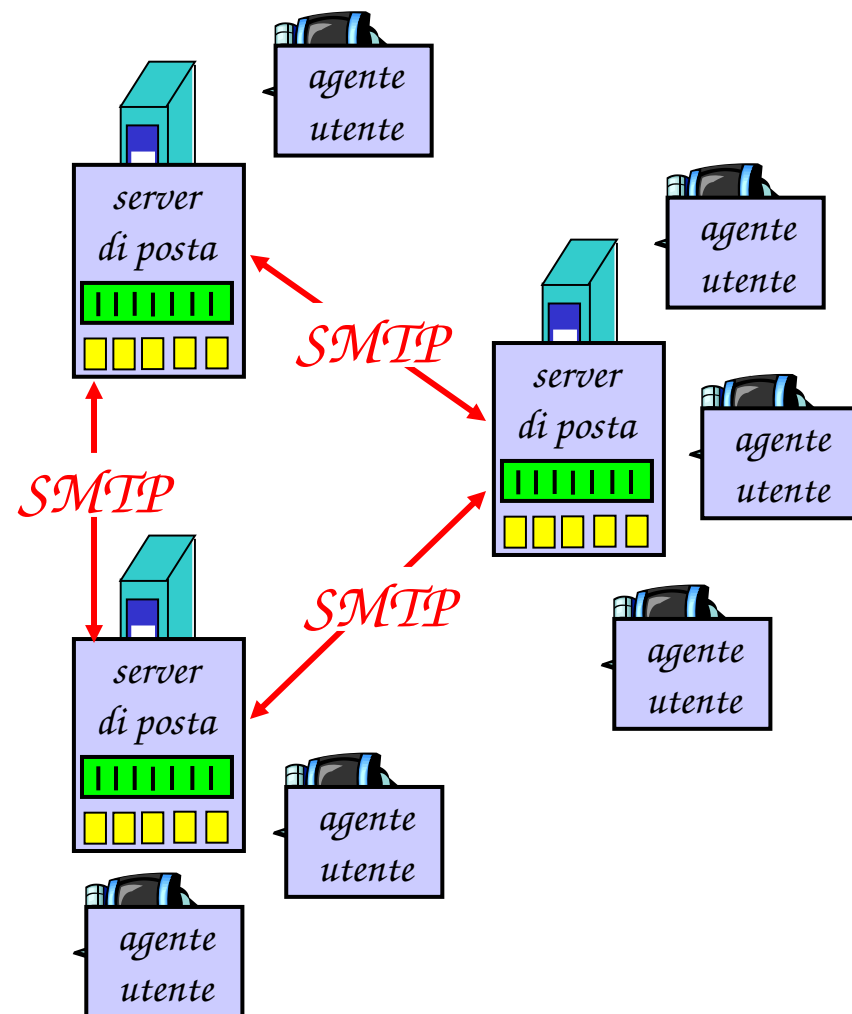
- ❑ detto anche "mail reader"
- ❑ composizione, editing, lettura dei messaggi di posta elettronica
- ❑ esempi: Eudora, Outlook, elm, Netscape Messenger
- ❑ i messaggi in uscita o in arrivo sono memorizzati sul server



# Posta elettronica: server di posta

## Server di posta

- ❑ *Casella di posta* (mailbox) contiene i messaggi in arrivo per l'utente
- ❑ *Coda di messaggi* da trasmettere
- ❑ *Protocollo SMTP* tra server di posta per inviare messaggi di posta elettronica
  - ❖ client: server di posta trasmittente
  - ❖ "server": server di posta ricevente

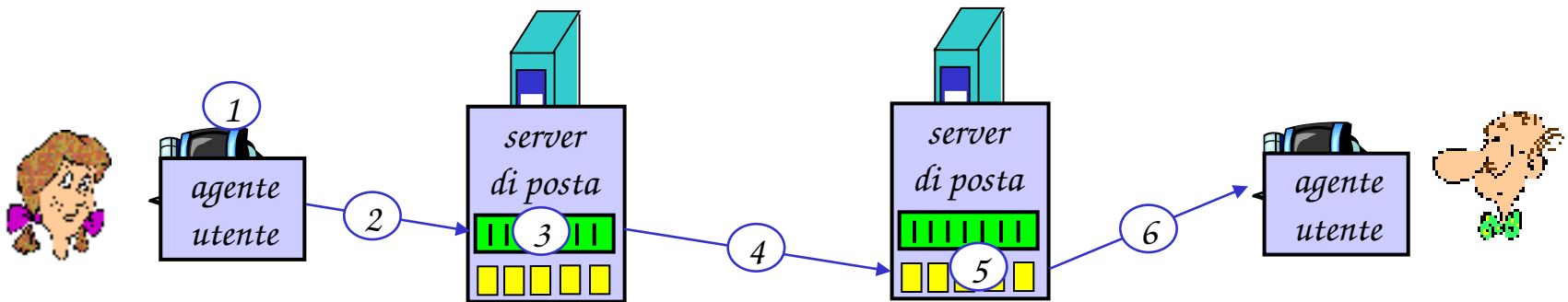


## Posta elettronica: SMTP [RFC 2821]

- *usa TCP per trasferire in modo affidabile i messaggi di posta elettronica dal client al server, porta 25*
- *trasferimento diretto: il server trasmittente al server ricevente*
- *tre espressioni per il trasferimento*
  - ❖ *handshaking (saluto)*
  - ❖ *trasferimento di messaggi*
  - ❖ *chiusura*
- *interazione comando/risposta*
  - ❖ *comandi: testo ASCII*
  - ❖ *risposta: codice di stato ed espressione*
- *i messaggi devono essere nel formato ASCII a 7 bit*

# Scenario: Alice invia un messaggio a Bob

- 1) Alice usa il suo agente utente per comporre il messaggio da inviare "a" bob@someschool.edu
- 2) L'agente utente di Alice invia un messaggio al server di posta di Alice; il messaggio è posto nella coda di messaggi
- 3) Il lato client di SMTP apre una connessione TCP con il server di posta di Bob
- 4) Il client SMTP invia il messaggio di Alice sulla connessione TCP
- 5) Il server di posta di Bob pone il messaggio nella casella di posta di Bob
- 6) Bob invoca il suo agente utente per leggere il messaggio



## Esempio di interazione SMTP

```
S: 220 hamburger.edu
C: HELO crepes.fr
S: 250 Hello crepes.fr, pleased to meet you
C: MAIL FROM: <alice@crepes.fr>
S: 250 alice@crepes.fr... Sender ok
C: RCPT TO: <bob@hamburger.edu>
S: 250 bob@hamburger.edu ... Recipient ok
C: DATA
S: 354 Enter mail, end with "." on a line by itself
C: Do you like ketchup?
C: How about pickles?
C: .
S: 250 Message accepted for delivery
C: QUIT
S: 221 hamburger.edu closing connection
```

## Provate un'interazione SMTP:

- ❑ **telnet servername 25**
- ❑ *Riceverete la risposta **220** dal server*
- ❑ *Digitate i comandi **HELO**, **MAIL FROM**, **RCPT TO**, **DATA**, **QUIT***

*Questo vi consente di inviare messaggi di posta elettronica senza usare il client di posta (lettore)*

# SMTP: note finali

- ❑ *SMTP usa connessioni persistenti*
- ❑ *SMTP richiede che il messaggio (intestazione e corpo) sia nel formato ASCII a 7 bit*
- ❑ *Il server SMTP usa CRLF . CRLF per determinare la fine del messaggio*

## *Confronto con HTTP:*

- ❑ *HTTP: pull*
- ❑ *SMTP: push*
- ❑ *Entrambi hanno un'interazione comando/risposta in ASCII, codici di stato*
- ❑ *HTTP: ogni oggetto è incapsulato nel suo messaggio di risposta*
- ❑ *SMTP: più oggetti vengono trasmessi in un unico messaggio*

# Formato dei messaggi di posta elettronica

*SMTP*: protocollo per scambiare messaggi di posta elettronica

*RFC 822*: standard per il formato dei messaggi di testo:

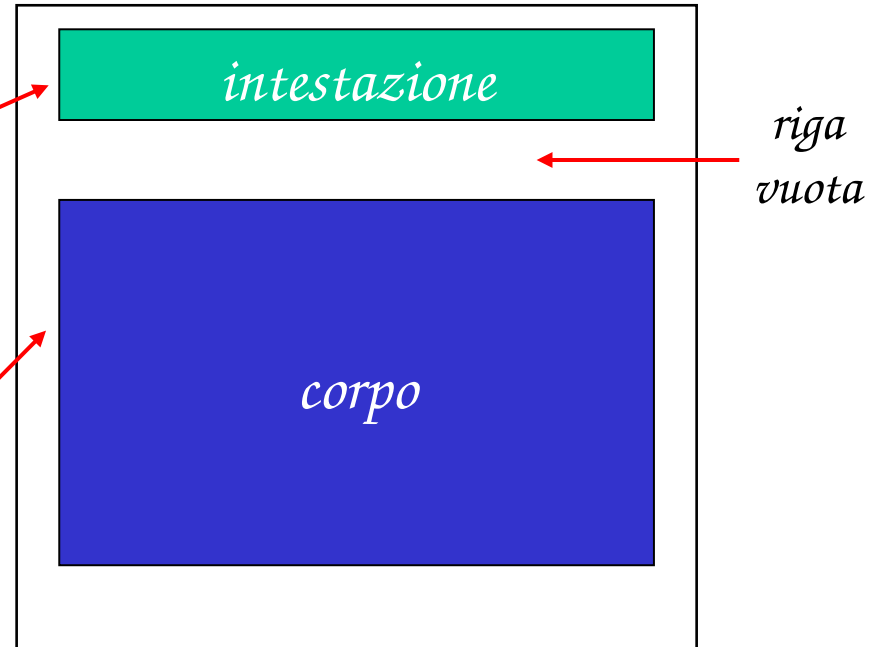
□ Righi di intestazione, per esempio

- ❖ *To:*
- ❖ *From:*
- ❖ *Subject:*

*differenti dai comandi SMTP !*

□ corpo

- ❖ il "messaggio", soltanto caratteri ASCII





## Formato del messaggio: estensioni di messaggi multimediali

- ❑ *MIME: estensioni di messaggi di posta multimediali, RFC 2045, 2056*
- ❑ *Alcune righe aggiuntive nell'intestazione dei messaggi dichiarano il tipo di contenuto MIME*

*Versione MIME*

*metodo usato  
per codificare i dati*

*Tipo di dati  
multimediali, sottotipo,  
dichiarazione  
dei parametri*

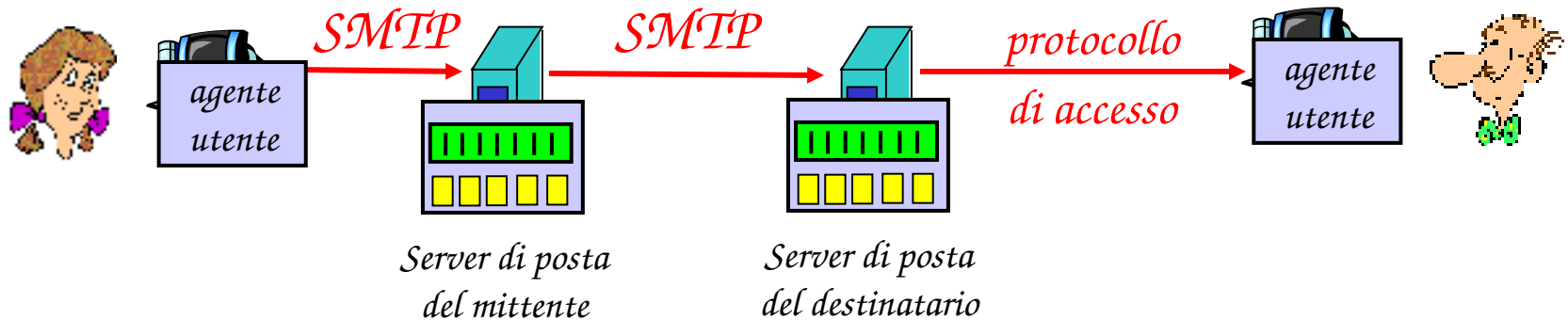
*Dati codificati*

```
From: alice@crepes.fr
To: bob@hamburger.edu
Subject: Picture of yummy crepe.
MIME-Version: 1.0
Content-Transfer-Encoding: base64
Content-Type: image/jpeg

base64 encoded data .....
.....
.....base64 encoded data
```

*Return-Path: <fancello@sci.unich.it>*  
*Received: from phobos.unich.it (phobos.unich.it [192.167.13.101])*  
*by gotham.sci.unich.it (8.12.8/8.12.8) with ESMTP id i8GAZMaS011065*  
*for <bista@sci.unich.it>; Thu, 16 Sep 2004 12:35:23 +0200*  
*Received: from phobos.unich.it (phobos.unich.it [127.0.0.1])*  
*by phobos.unich.it (8.12.5/8.12.8) with ESMTP id i8GAZ7Rp024306*  
*for <bista@sci.unich.it>; Thu, 16 Sep 2004 12:35:07 +0200*  
*Received: from sci111.sci.unich.it ([192.167.92.11])*  
*by phobos.unich.it (MailMonitor for SMTP v1.2.2 );*  
*Thu, 16 Sep 2004 12:35:06 +0200 (CEST)*  
*Message-ID: <001801c49bd8\$b54118c0\$0b5ca7c0@sci.unich.it>*  
*From: "Maura Fancello" <fancello@sci.unich.it>*  
*To: "stefano Bistarelli" <bista@sci.unich.it>*  
*References: <000801c3d5fd\$56b8ce20\$0b5ca7c0@sci.unich.it> <6c7301c49bd0\$23e50150\$bd603092@iit.cnr.it>*  
*Subject: Re: lavagna luminosa e proiettore*  
*Date: Thu, 16 Sep 2004 12:34:10 +0200*  
*MIME-Version: 1.0*  
*Content-Type: multipart/alternative;*  
*boundary="-----\_NextPart\_000\_0015\_01C49BE9.77729620"*  
*X-Priority: 3*  
*X-MSMail-Priority: Normal*  
*X-Mailer: Microsoft Outlook Express 6.00.2800.1106*  
*X-MimeOLE: Produced By Microsoft MimeOLE V6.00.2800.1106*  
*X-Antivirus: Scanned by F-Prot Antivirus (http://www.f-prot.com)*  
*X-Antivirus-Summary: Mod score: 0*  
*X-Antivirus: Scanned by F-Prot Antivirus (http://www.f-prot.com)*  
*X-Spam-Checker-Version: SpamAssassin 2.63 (2004-01-11) on gotham.sci.unich.it*  
*X-Spam-Level:*  
*X-Spam-Status: No, hits=-4.8 required=3.0 tests=BAYES\_00,HTML\_MESSAGE*  
*autolearn=no version=2.63*

# Protocolli di accesso alla posta



- *SMTP: consegna/memorizzazione sul server del destinatario*
- *Protocollo di accesso alla posta: ottenere i messaggi dal server*
  - ❖ *POP: Post Office Protocol [RFC 1939]*
    - *autorizzazione (agente <--> server) e download*
  - ❖ *IMAP: Internet Mail Access Protocol [RFC 1730]*
    - *più funzioni (più complesse)*
    - *manipolazione di messaggi memorizzati sul server*
  - ❖ *HTTP: Hotmail, Yahoo! Mail, ecc.*


# Protocollo POP3

## *Fase di autorizzazione*

- *Comandi del client:*
  - ❖ **user:** dichiara il nome dell'utente
  - ❖ **pass:** password
- *Risposte del server*
  - ❖ **+OK**
  - ❖ **-ERR**

## *Fase di transazione, client:*

- **list:** elenca i numeri dei messaggi
- **retr:** ottiene i messaggi per numero
- **dele:** cancella
- **quit**



```
S: +OK POP3 server ready
C: user bob
S: +OK
C: pass hungry
S: +OK user successfully logged on

C: list
S: 1 498
S: 2 912
S: .
C: retr 1
S: <message 1 contents>
S: .
C: dele 1
C: retr 2
S: <message 1 contents>
S: .
C: dele 2
C: quit
S: +OK POP3 server signing off
```

# POP3 (altro) e IMAP

## *Ancora su POP3*

- ❑ *Il precedente esempio usa la modalità “scarica e cancella”*
- ❑ *Bob non può rileggere le e-mail se cambia client*
- ❑ *Modalità “scarica e mantieni”: copia i messaggi su più client*
- ❑ *POP3 è un protocollo senza stato tra le varie sessioni*

## *IMAP*

- ❑ *Mantiene tutti i messaggi in un unico posto: il server*
- ❑ *Consente all'utente di organizzare i messaggi in cartelle*
- ❑ *IMAP conserva lo stato dell'utente tra le varie sessioni:*
  - ❖ *I nomi delle cartelle e l'associazione tra identificatori dei messaggi e nomi delle cartelle*

# Convenzioni e netiquette

## ❑ *Comunicazione di stati d'animo con le faccette: ([emoticons](#))*

*:-) sorridente e scherzoso*

*;-) malizioso*

*:-( triste*

*:-I indifferente*

*:-> sarcastico*

*>:-> diabolico*

*:-/ perplesso*

*:-D sorpreso*

*:O molto sorpreso*

*>:-> ammiccante e diabolico*

## ❑ *Usare lettere maiuscole equivale ad URLARE*

# Il lingo

- ❑ **AFAIK** *As Far As I Know*
- ❑ **AKA** *Also Known As*
- ❑ **BBIAB** *Be Back in a Bit*
- ❑ **BBIAF** *Be Back in a Few*
- ❑ **BBL** *Be Back Later*
- ❑ **BFN** *Bye For Now*
- ❑ **BTW** *By The Way*
- ❑ **CID** *Consider It Done*
- ❑ **CIO** *Check It Out*
- ❑ **CUL8R** *See You Later*
- ❑ **FYA** *For Your Amusement*
- ❑ **FYI** *For Your Information*
- ❑ **GTSY** *Glad To See Ya*

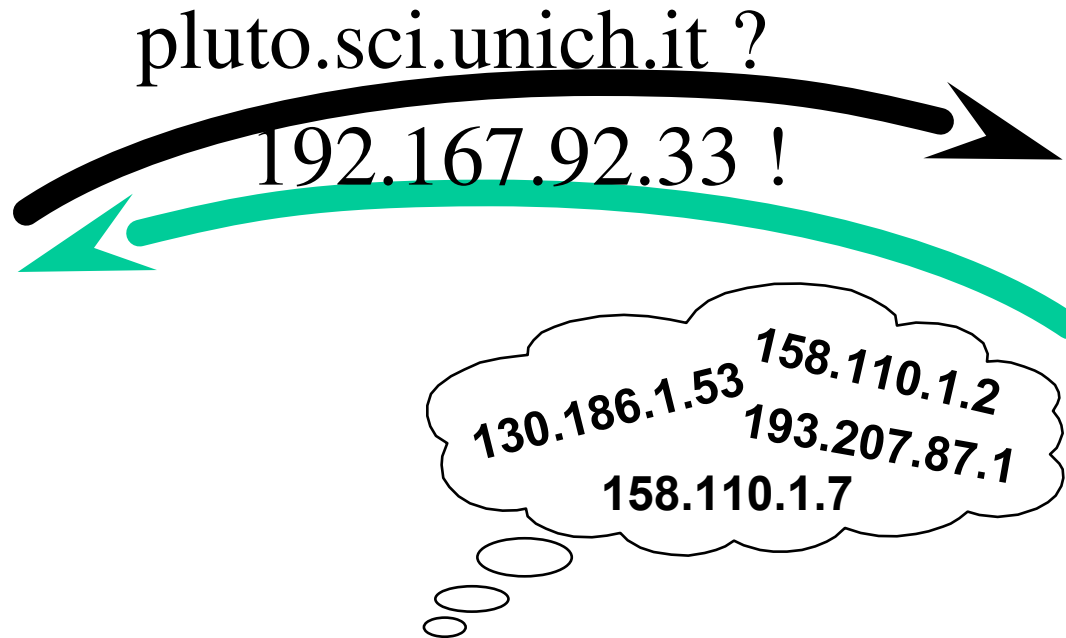
- ❑ **GYPPO** *Get Your Pants Off*
- ❑ **IMO** *In My Opinion*
- ❑ **IOW** *In Other Words*
- ❑ **IRL** *In Real Life*
- ❑ **KIT** *Keep In Touch*
- ❑ **MOTD** *Message Of The Day*
- ❑ **POV** *Point of View*
- ❑ **RSN** *Real Soon Now*
- ❑ **RTM** *Read The Manual*
- ❑ **TIA** *Thanks in Advance*
- ❑ **TX** *Thanks*
- ❑ **TYVM** *Thank You Very Much*
- ❑ **WB** *Welcome Back*

# Capitolo 2: Livello di applicazione

- ❑ 2.1 Principi delle applicazioni di rete
- ❑ 2.2 Web e HTTP
- ❑ 2.3 FTP
- ❑ 2.4 Posta elettronica
  - ❖ SMTP, POP3, IMAP
- ❑ 2.5 DNS
- ❑ 2.6 Condivisione di file P2P
- ❑ 2.7 Programmazione delle socket con TCP
- ❑ 2.8 Programmazione delle socket con UDP
- ❑ 2.9 Costruire un semplice server web



# Domain Name Server (DNS)



# *DNS: le funzioni*

- *ad ogni risorsa TCP/IP può essere assegnato un nome simbolico*  
*sono necessari:*
  - ❖ *un metodo per associare il nome simbolico di una macchina all'indirizzo (o agli indirizzi) IP: risoluzione diretta*
  - ❖ *un metodo per associare ad un indirizzo IP al nome simbolico della macchina: risoluzione inversa*
- *Domain Name System (DNS)*
  - ❖ *definito presso ISI - USC 1984*
  - ❖ *RFC 882, RFC 883, RFC 973 (obsolete)*
  - ❖ *RFC 1034, RFC 1035, RFC 1123, RFC 1537, RFC 1912*

## Un po' di storia

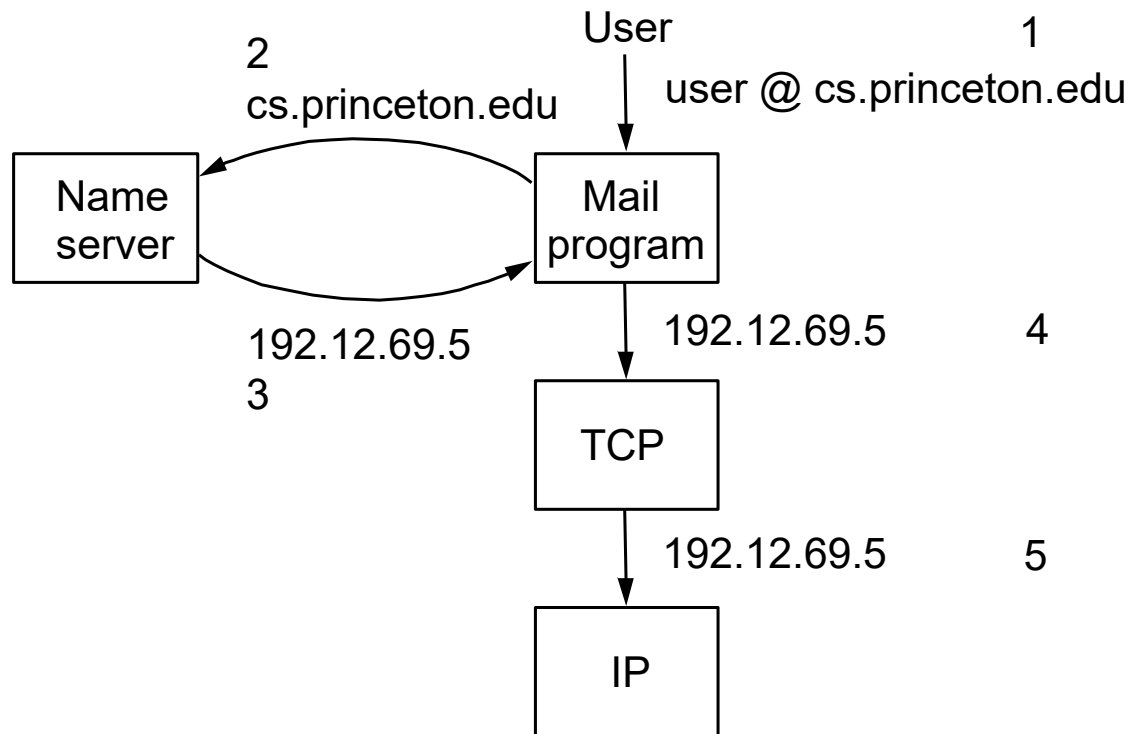
- ❑ *Ai tempi di ARPANET esisteva in ogni sistema operativo un unico file, `hosts.txt`, che elencava tutti gli host e i loro indirizzi IP. Ogni notte tutti gli host della rete lo copiavano dal sito in cui era mantenuto*
- ❑ *Quando la rete comprendeva solo qualche centinaio di grosse macchine questo approccio funzionava bene; quando la rete crebbe venne inventato il servizio DNS (Domain Name Server), definito nei documenti RFC 1034 e 1035*

# *DNS: caratteristiche principali*

- ❑ *database distribuito*
- ❑ *basato sul modello client/server*
- ❑ *tre componenti principali:*
  - *spazio dei nomi e informazioni associate*  
*(Resource Record -  $\mathcal{RR}$ )*
  - *nameserver (application server che mantiene i dati)*
  - *resolver (client per l'interrogazione del nameserver)*
- ❑ *accesso veloce ai dati (database in memoria centrale e meccanismo di caching)*

# Esempio

## □ Mailboxes



# DNS: Domain Name System

*Persone:* molti identificatori:

- ❖ nome, codice fiscale, carta d'identità

*Host e router di Internet:*

- ❖ indirizzo IP (32 bit) - usato per indirizzare i datagrammi
- ❖ “nome”, ad esempio, *www.yahoo.com* – usato dagli esseri umani

D: Come associare un indirizzo IP a un nome?

*Domain Name System:*

- ❑ Database distribuito implementato in una gerarchia di server DNS
- ❑ Protocollo a livello di applicazione che consente agli host, ai router e ai server DNS di comunicare per risolvere i nomi (tradurre indirizzi/nomi)
  - ❖ nota: funzioni critiche di Internet implementate come protocollo a livello di applicazione
  - ❖ complessità nelle parti periferiche della rete

# DNS

## Servizi DNS

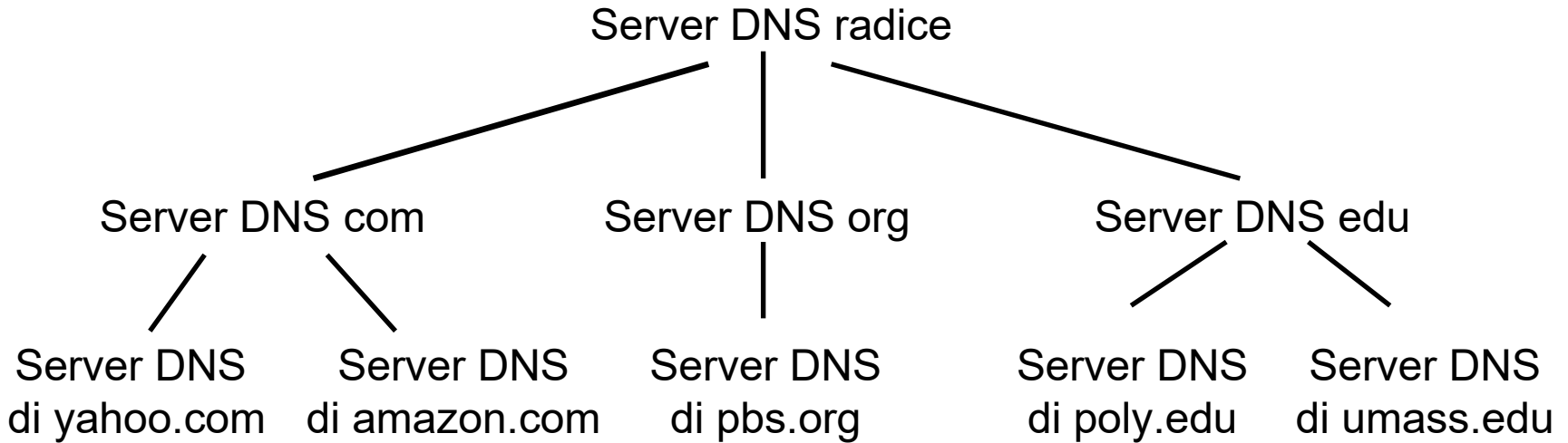
- *Traduzione degli hostname in indirizzi IP*
- *Host aliasing*
  - ❖ *un host può avere più nomi*
- *Mail server aliasing*
- *Distribuzione locale*
  - ❖ *server web replicati: insieme di indirizzi IP per un nome canonico*

## Perché non centralizzare DNS?

- *singolo punto di guasto*
- *volume di traffico*
- *database centralizzato distante*
- *manutenzione*

*Un database centralizzato su un singolo server DNS non è scalabile !*

# Database distribuiti e gerarchici



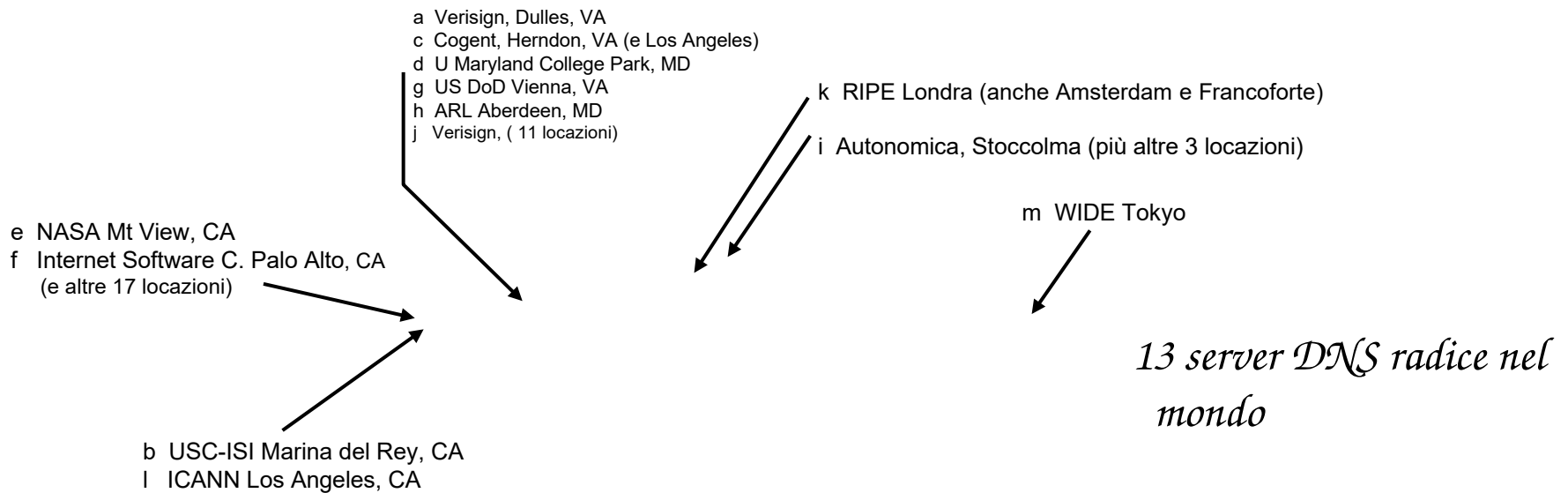
Il client vuole l'IP di *www.amazon.com*; 1<sup>a</sup> approssimazione:

- Il client interroga il server radice per trovare il server DNS com
- Il client interroga il server DNS com per ottenere il server DNS amazon.com
- Il client interroga il server DNS amazon.com per ottenere l'indirizzo IP di *www.amazon.com*



# *DNS: server DNS radice*

- *contattato da un server DNS locale che non può tradurre il nome*
- *server DNS radice:*
  - ❖ *contatta un server DNS autorizzato se non conosce la mappatura*
  - ❖ *ottiene la mappatura*
  - ❖ *restituisce la mappatura al server DNS locale*



# Server TLD e server di competenza

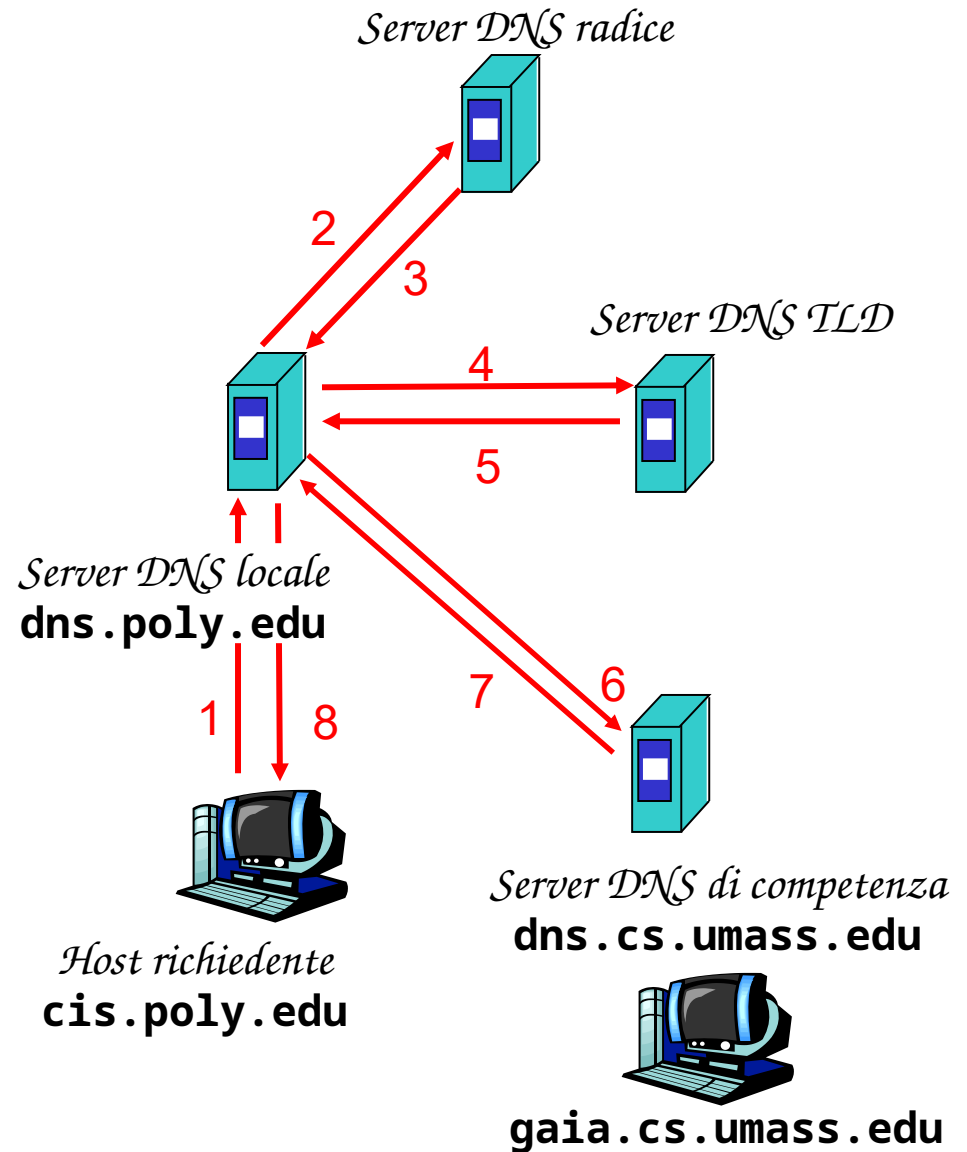
- ❑ *Server TLD (top-level domain):* si occupano dei domini com, org, net, edu, ecc. e di tutti i domini locali di alto livello, quali uk, fr, ca e jp.
  - ❖ *Network Solutions gestisce i server TLD per il dominio com*
  - ❖ *Educause gestisce quelli per il dominio edu*
- ❑ *Server di competenza (authoritative server):* ogni organizzazione dotata di host Internet pubblicamente accessibili (quali i server web e i server di posta) deve fornire i record DNS di pubblico dominio che mappano i nomi di tali host in indirizzi IP.
  - ❖ *possono essere mantenuti dall'organizzazione o dal service provider*

## Server DNS locale

- ❑ *Non appartiene strettamente alla gerarchia dei server*
- ❑ *Ciascun ISP (università, società, ISP residenziale) ha un server DNS locale.*
  - ❖ *detto anche “default name server”*
- ❑ *Quando un host effettua una richiesta DNS, la query viene inviata al suo server DNS locale*
  - ❖ *il server DNS locale opera da proxy e inoltra la query in una gerarchia di server DNS*

## Esempio

- L'host *cis.poly.edu* vuole l'indirizzo IP di *gaia.cs.umass.edu*



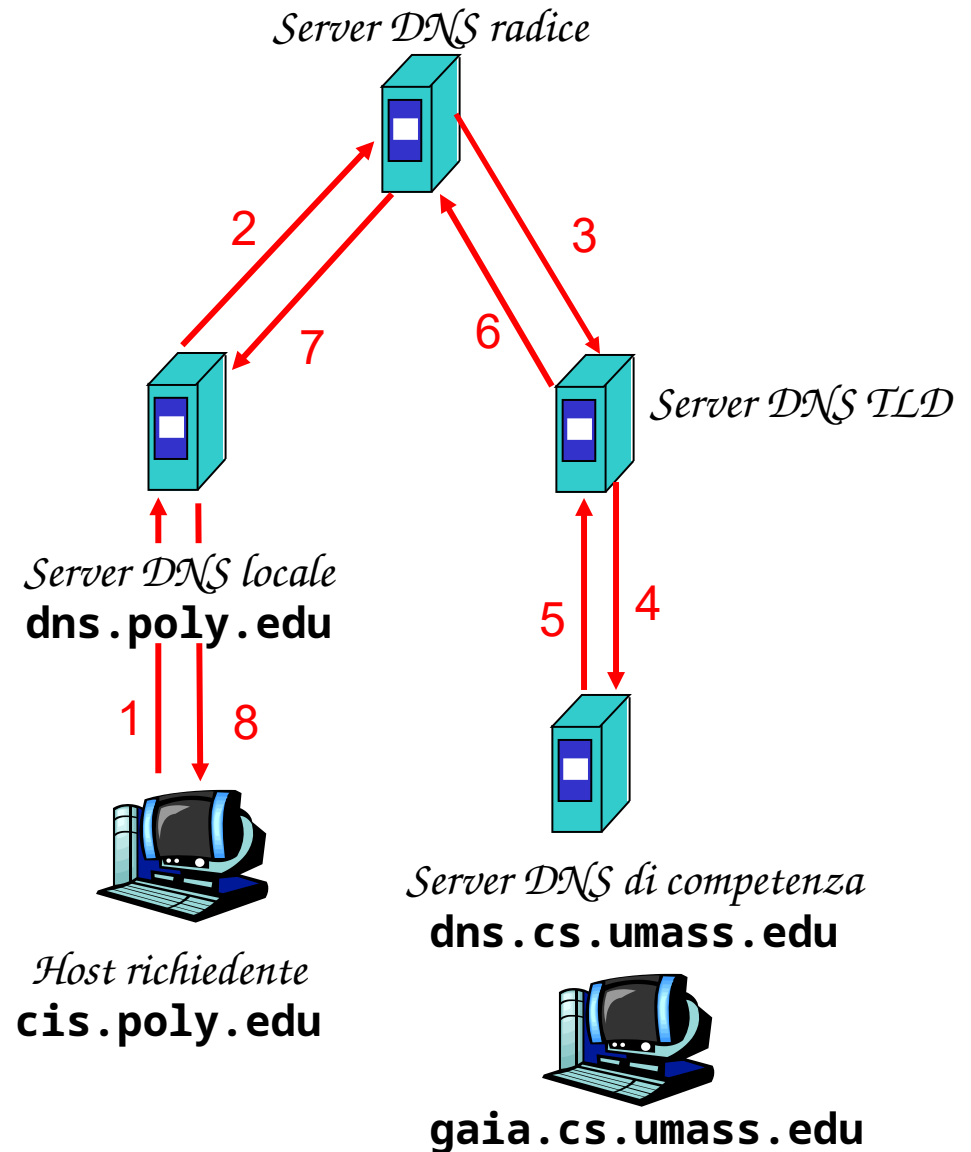
# Query ricorsiva

## Query ricorsiva:

- ❑ Affida il compito di tradurre il nome al server DNS contattato
- ❑ Compito difficile?

## Query iterativa:

- ❑ Il server contattato risponde con il nome del server da contattare
- ❑ “Non conosco questo nome, ma chiedi a questo server”



# *DNS: caching e aggiornamento dei record*

- *Una volta che un server DNS impara la mappatura, la mette nella **cache***
  - ❖ *le informazioni nella cache vengono invalidate (spariscono) dopo un certo periodo di tempo*
  - ❖ *tipicamente un server DNS locale memorizza nella cache gli indirizzi IP dei server TLD*
    - *quindi i server DNS radice non vengono visitati spesso*
- *I meccanismi di aggiornamento/notifica sono progettati da IETF*
  - ❖ *RFC 2136*
  - ❖ *<http://www.ietf.org/html.charters/dnsind-charter.html>*

# Record DNS

DNS: database distribuito che memorizza i record di risorsa (*RR*)

*Formato RR: (name, value, type, ttl)*

## □ *Type=A*

- ❖ **name** è il nome dell'host
- ❖ **value** è l'indirizzo IP

## □ *Type=NS*

- ❖ **name** è il dominio  
(ad esempio foo.com)
- ❖ **value** è il nome dell'host del server di competenza di questo dominio

## □ *Type=CNAME*

- ❖ **name** è il nome alias di qualche nome  
"canonico" (nome vero)

www.ibm.com è in realtà

servereast.backup2.ibm.com

- ❖ **value** è il nome canonico

## □ *Type=MX*

- ❖ **value** è il nome del server di posta associato a **name**

```

$TTL      43200
@          IN      SOA      ns.mesys.it.
hostmaster.mesys.it. (
                2002053101 ; serial
                86400 ; refresh
                3600 ; retry
                604800 ; expire
                86400 ; default_ttl
                )
@          IN      MX       5          mail.mesys.it.
@          IN      NS       ns.mesys.it.
@          IN      NS       dns2.nic.it.
localhost IN      A        127.0.0.1
ns         IN      A        151.4.83.2
ns1        IN      A        151.4.83.3
mail       IN      A        151.4.83.2
www        IN      CNAME    turtle.mesys.it.
ftp        IN      CNAME    dolphin.mesys.it.

```



# Messaggi DNS

Protocollo DNS: *domande* (query) e messaggi di *risposta*, entrambi con lo stesso *formato*

## *Intestazione del messaggio*

- ❑ *Identificazione*: numero di 16 bit  
per la domanda;  
la risposta alla domanda usa lo  
stesso numero
- ❑ *Flag*:
  - ❖ domanda o risposta
  - ❖ richiesta di ricorsione
  - ❖ ricorsione disponibile
  - ❖ risposta di competenza

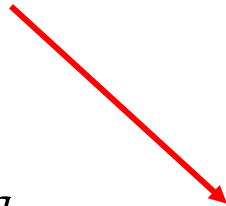
# Messaggi DNS

*Campi per  
il nome richiesto  
e il tipo di domanda*

*RR nella risposta  
alla domanda*

*Record per  
i server di competenza*

*Informazioni extra che  
possono essere usate*



# Inserire record nel database DNS

- *Esempio: abbiamo appena avviato la nuova società "Network Utopia"*
- *Registriamo il nome **networkutopia.com** presso registrar (ad esempio, Network Solutions)*
  - ❖ *Forniamo a registrar i nomi e gli indirizzi IP dei server DNS di competenza (primario e secondario)*
  - ❖ *Registrar inserisce due RR nel server TLD com:*

(networkutopia.com, dns1.networkutopia.com, NS)  
(dns1.networkutopia.com, 212.212.212.1, A)

- *Inseriamo nel server di competenza un record tipo A per www.networkutopia.com e un record tipo MX per networkutopia.com*
- *In che modo gli utenti otterranno l'indirizzo IP del nostro sito web?*

# Esercizi

- ❑ *Protocollo di trasporto: UDP*
- ❑ *Porta: 53*
- ❑ *... bugia ..*
- ❑ *Scoprire per quali messaggi DNS usa la porta 53 e il TCP  
(invece che l'UDP)*

# Capitolo 2: Livello di applicazione

- ❑ 2.1 Principi delle applicazioni di rete
- ❑ 2.2 Web e HTTP
- ❑ 2.3 FTP
- ❑ 2.4 Posta elettronica
  - ❖ SMTP, POP3, IMAP
- ❑ 2.5 DNS
- ❑ 2.6 *Condivisione di file P2P*
- ❑ 2.7 Programmazione delle socket con TCP
- ❑ 2.8 Programmazione delle socket con UDP
- ❑ 2.9 Costruire un semplice server web

# Condivisione di file P2P

## Esempio

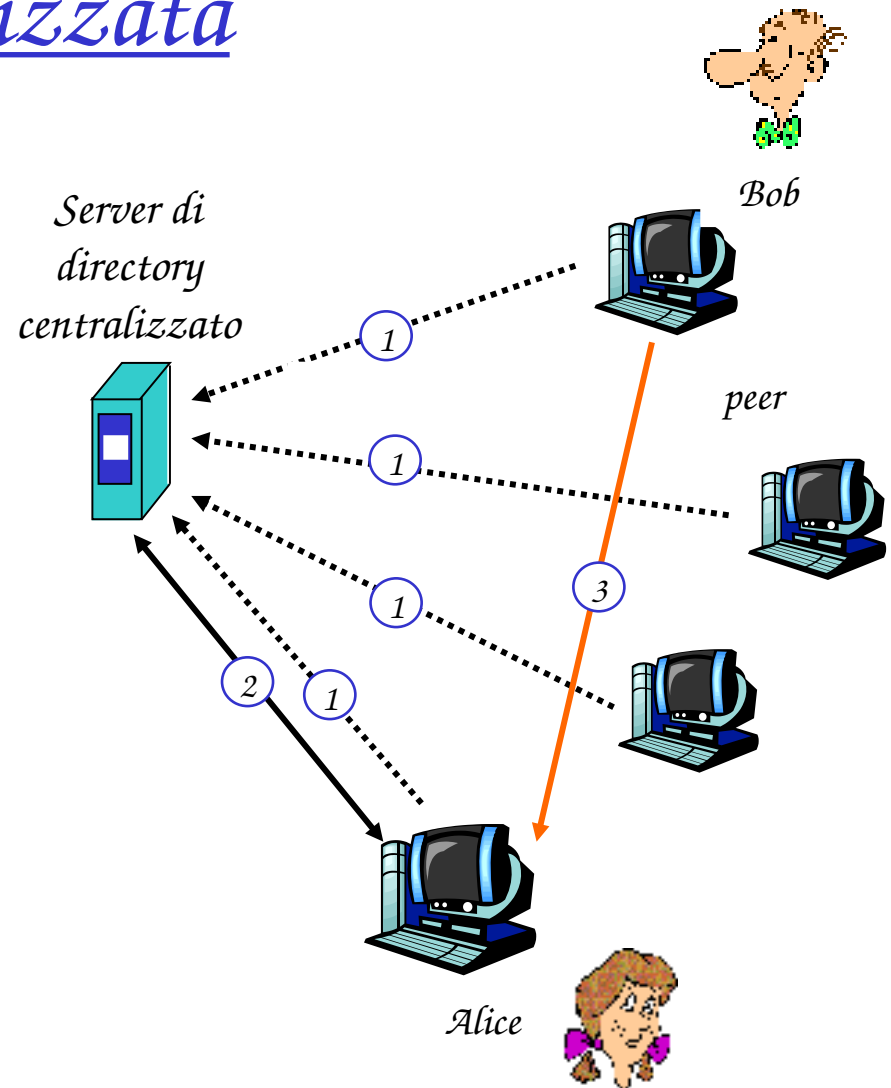
- Alice esegue un'applicazione di condivisione file P2P sul suo notebook
- Si collega in modo intermittente a Internet; ottiene un nuovo indirizzo IP ogni volta che si collega
- Cerca la canzone intitolata "Hey Jude"
- L'applicazione visualizza altri peer che hanno una copia di "Hey Jude"
- Alice sceglie uno dei peer, Bob
- Il file viene inviato dal PC di Bob al notebook di Alice: HTTP
- Mentre Alice scarica il file, altri utenti potrebbero scaricare dei file da Alice
- Il peer di Alice è sia client web sia server web transitorio

*Tutti i peer sono server = grande scalabilità!*

# P2P: directory centralizzata

*Progetto originale di "Napster"*

- 1) *quando il peer si collega, informa il server centrale:*
  - ❖ *indirizzo IP*
  - ❖ *contenuto*
- 2) *Alice cerca la canzone "Hey Jude"*
- 3) *Alice richiede il file a Bob*



## P2P: problemi con la directory centralizzata

- ❑ *Unico punto di guasto*
- ❑ *Collo di bottiglia per le prestazioni*
- ❑ *Violazione del diritto d'autore*

*Il trasferimento dei file è distribuito,  
ma il processo  
di localizzazione è fortemente  
centralizzato*



# Query flooding: Gnutella

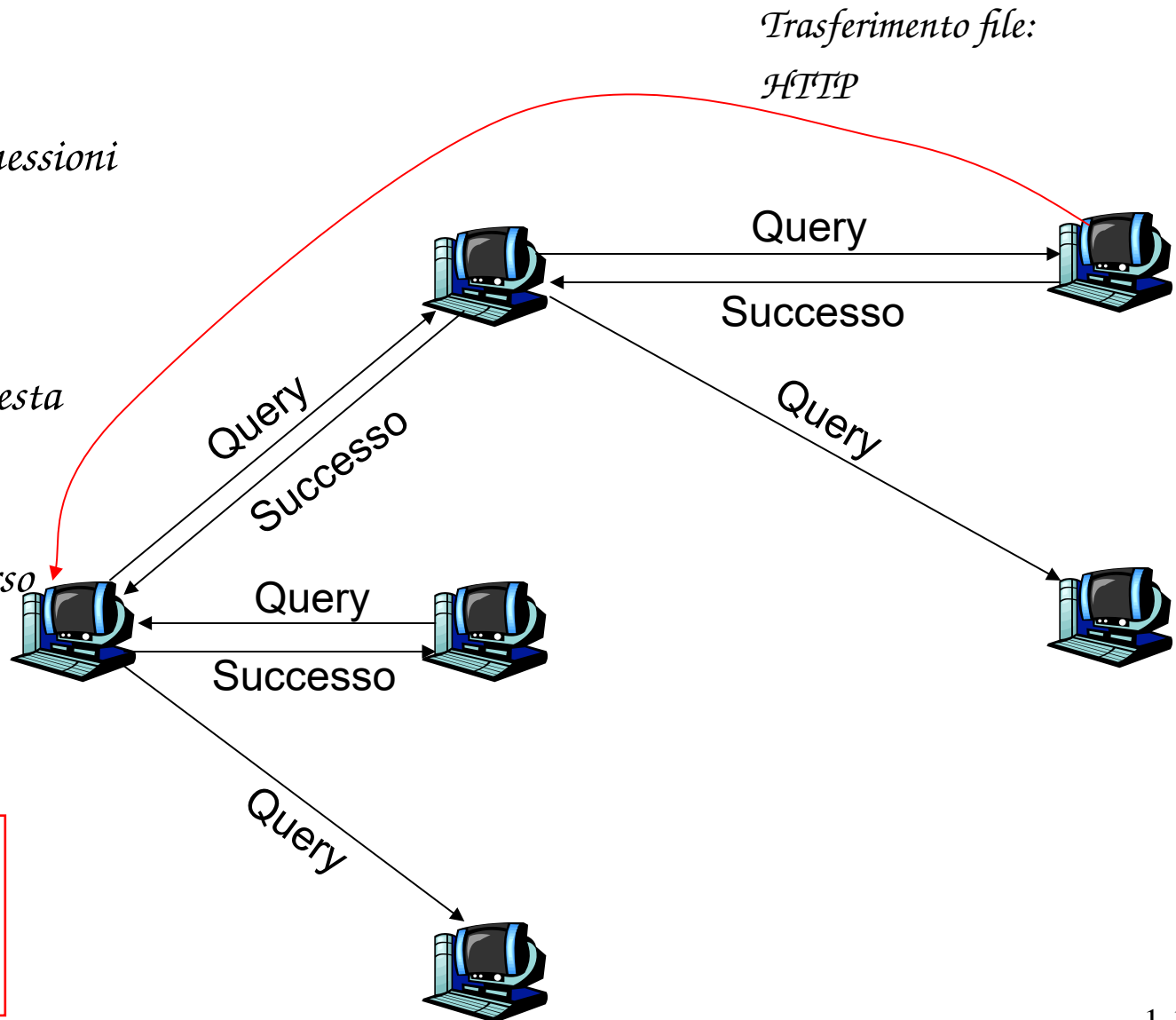
- ❑ *Completamente distribuito*
  - ❖ *nessun server centrale*
- ❑ *Protocollo di pubblico dominio*
- ❑ *Molti client Gnutella implementano il protocollo*

## *Rete di copertura: grafo*

- ❑ *Arco tra i peer  $X$  e  $Y$  se c'è una connessione TCP*
- ❑ *Tutti i peer attivi e gli archi formano la rete di copertura*
- ❑ *Un arco non è un collegamento fisico*
- ❑ *Un dato peer sarà solitamente connesso con meno di 10 peer vicini nella rete di copertura*

# Gnutella: protocollo

- Il messaggio di richiesta è trasmesso sulle connessioni TCP esistenti
- Il peer inoltra il messaggio di richiesta
- Il messaggio di successo è trasmesso sul percorso inverso



Scalabilità:  
query flooding  
a raggio limitato

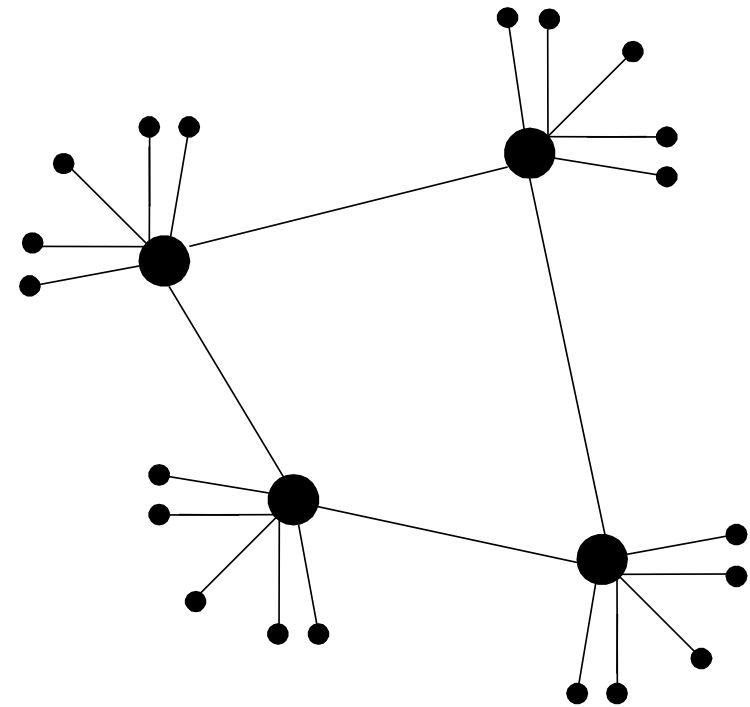
## Gnutella: unione di peer

1. *Per unire il peer X alla rete, bisogna trovare qualche altro peer della rete Gnutella: usate la lista dei peer candidati*
2. *X tenta in sequenza di impostare una connessione TCP con i peer della lista finché non stabilisce una connessione con Y*
3. *X invia un messaggio Ping a Y; Y inoltra il messaggio Ping*
4. *Tutti i peer che ricevono il messaggio Ping rispondono con un messaggio Pong*
5. *X riceve molti messaggi Pong. Quindi può impostare delle connessioni TCP aggiuntive*

*Distacco dei peer: consultate il problema alla fine del capitolo!*

# Sfruttare l'eterogeneità: KaZaA

- Ogni peer è un leader di gruppo o è assegnato a un leader di gruppo
  - ❖ Connessione TCP tra peer e il suo leader di gruppo
  - ❖ Connessioni TCP tra qualche coppia di leader di gruppo
- Il leader di gruppo tiene traccia del contenuto di tutti i suoi figli.



● Peer ordinario

● Peer leader di gruppo

— Relazioni di adiacenza  
nella rete di copertura

## KaZaA: query

- ❑ *Ogni file ha un identificatore hash e un descrittore*
- ❑ *Il client invia al suo leader di gruppo una query con una parola chiave*
- ❑ *Il leader di gruppo risponde con un elenco di peer che condividono i file i cui descrittori corrispondono alle parole chiave:*
  - ❖ *Per ogni corrispondenza: metadata, hash, indirizzo IP*
- ❑ *Se il leader di gruppo inoltra la query ad altri leader di gruppo, questi rispondono con le corrispondenze*
- ❑ *Il client quindi seleziona i file per il downloading*
  - ❖ *Le richieste HTTP che usano un identificatore hash sono trasmesse ai peer che hanno il file desiderato*

# Tecniche KaZaA

- ❑ *Limitare il numero di upload simultanei*
- ❑ *Accodamento delle richieste*
- ❑ *Priorità di incentivo*
- ❑ *Downloading parallelo*

# Capitolo 2: Livello di applicazione

- ❑ 2.1 Principi delle applicazioni di rete
- ❑ 2.2 Web e HTTP
- ❑ 2.3 FTP
- ❑ 2.4 Posta elettronica
  - ❖ SMTP, POP3, IMAP
- ❑ 2.5 DNS
- ❑ 2.6 Condivisione di file P2P
- ❑ 2.7 Programmazione delle socket con TCP
- ❑ 2.8 Programmazione delle socket con UDP
- ❑ 2.9 Costruire un semplice server web

# Programmazione delle socket

Obiettivo: imparare a costruire un'applicazione client/server che comunica utilizzando le socket

## *Socket API*

- introdotta in BSD4.1 UNIX, 1981
- esplicitamente creata, usata, distribuita dalle applicazioni
- paradigma client/server
- due tipi di servizio di trasporto tramite una socket API:
  - ❖ datagramma inaffidabile
  - ❖ affidabile, orientata ai byte

## *socket*

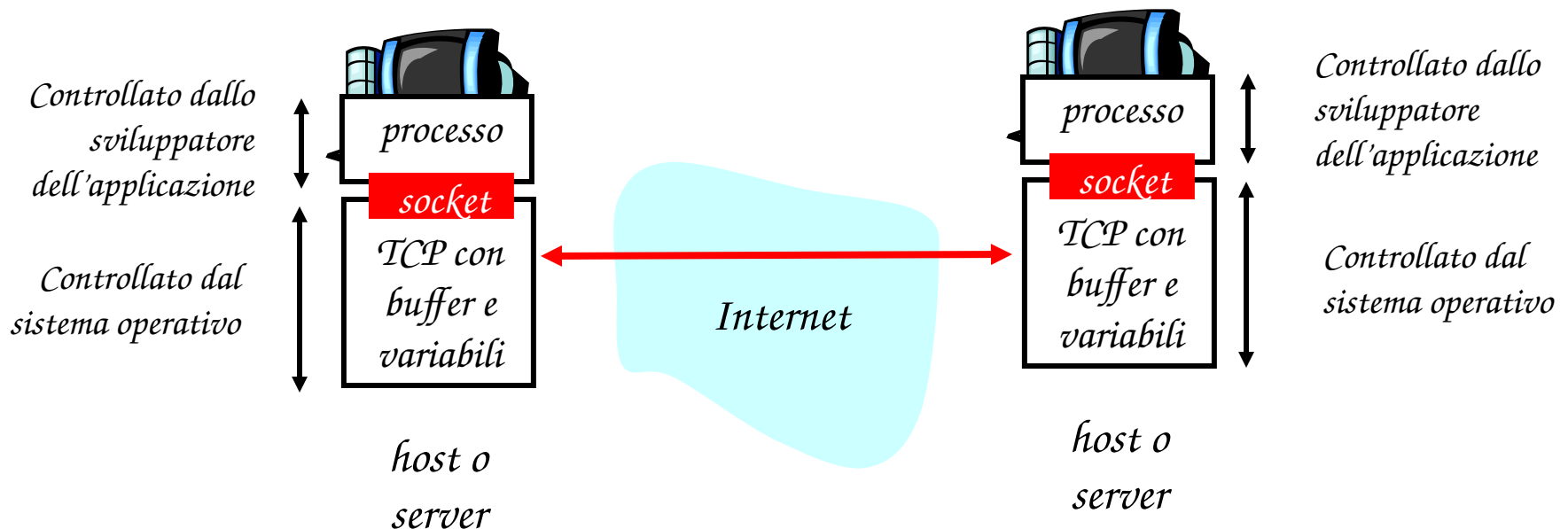
Interfaccia di un  
*host locale,*  
*creata dalle applicazioni,*  
*controllata dal SO*  
(una "porta") in cui  
il processo di un'applicazione può  
*inviare e ricevere*  
messaggi al/dal processo di  
un'altra applicazione



# Programmazione delle socket con TCP

Socket: una porta tra il processo di un'applicazione e il protocollo di trasporto end-end (UDP o TCP)

Servizio TCP: trasferimento affidabile di **byte** da un processo all'altro



# Programmazione delle socket con TCP

## *Il client deve contattare il server*

- Il processo server deve essere in corso di esecuzione
- Il server deve avere creato una socket (porta) che dà il benvenuto al contatto con il client

## *Il client contatta il server:*

- Creando una socket TCP
- Specificando l'indirizzo IP, il numero di porta del processo server
- Quando il *client crea la socket*: il client TCP stabilisce una connessione con il server TCP

- Quando viene contattato dal client, il *server TCP crea una nuova socket* per il processo server per comunicare con il client

- ❖ consente al server di comunicare con più client
- ❖ numeri di porta origine usati per distinguere i client (*maggiori informazioni nel Capitolo 3*)

## *Punto di vista dell'applicazione*

*TCP fornisce un trasferimento di byte affidabile e ordinato ("pipe") tra client e server*

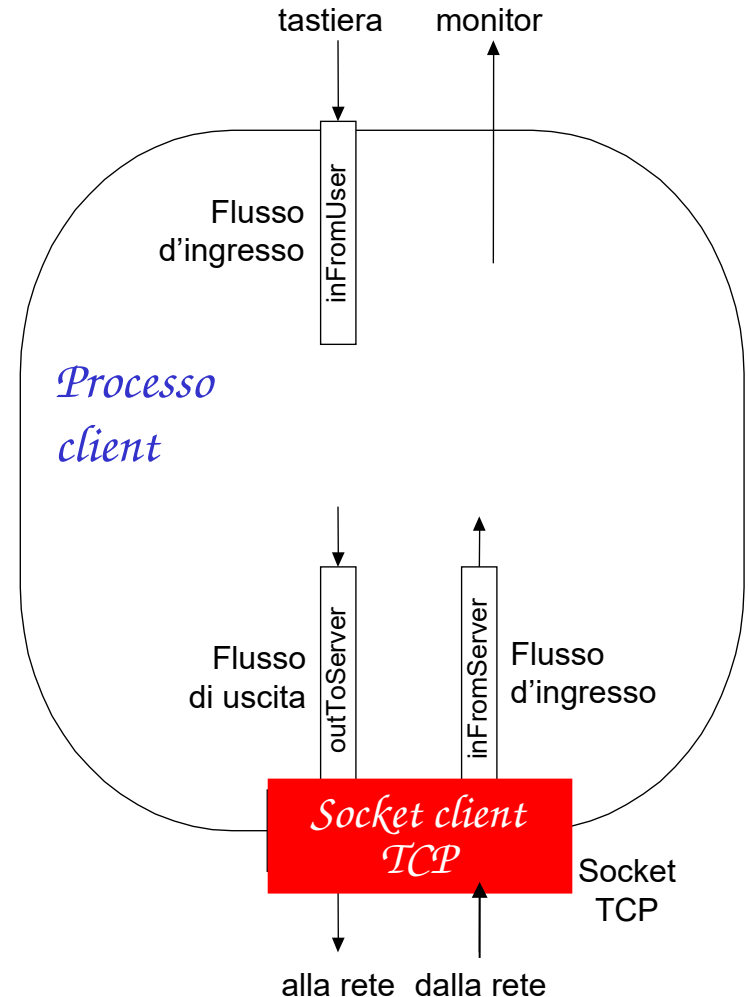
# Termini

- Un *flusso* (stream) è una sequenza di caratteri che fluisce verso/da un processo.
- Un *flusso d'ingresso* (input stream) è collegato a un'origine di input per il processo, ad esempio la tastiera o la socket.
- Un *flusso di uscita* (output stream) è collegato a un'uscita per il processo, ad esempio il monitor o la socket.

# Programmazione delle socket con TCP

## *Esempio di applicazione client-server:*

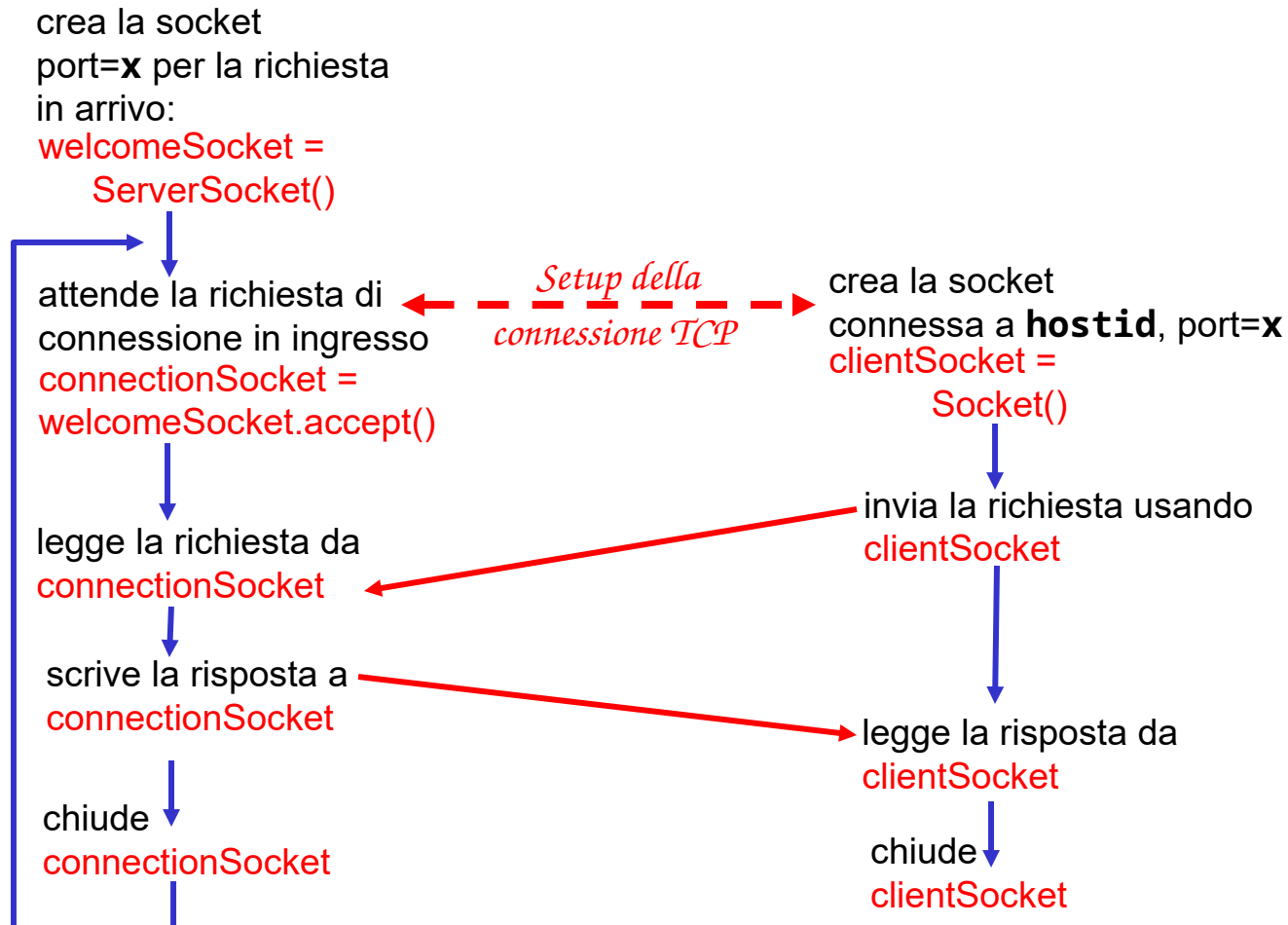
- 1) Il client legge una riga dall'input standard (flusso **inFromUser**) e la invia al server tramite la socket (flusso **outToServer**)
- 2) Il server legge la riga dalla socket
- 3) Il server converte la riga in lettere maiuscole e la invia al client
- 4) Il client legge nella sua socket la riga modificata e la visualizza (flusso **inFromServer**)



# Interazione delle socket client/server: TCP

*Server* (gira su **hostid**)

*Client*



## Esempio: client Java (TCP)

```
import java.io.*;
import java.net.*;
class TCPClient {
```

```
    public static void main(String argv[]) throws Exception
    {
```

```
        String sentence;
        String modifiedSentence;
```

*Crea un  
flusso d'ingresso*

```
        BufferedReader inFromUser =
            new BufferedReader(new InputStreamReader(System.in));
```

*Crea una  
socket client,  
connessa al server*

```
        Socket clientSocket = new Socket("hostname", 6789);
```

*Crea un  
flusso di uscita  
collegato alla socket*

```
        DataOutputStream outToServer =
            new DataOutputStream(clientSocket.getOutputStream());
```

## Esempio: client Java (TCP), continua

*Crea  
un flusso d'ingresso  
collegato alla socket* → `BufferedReader inFromServer =  
new BufferedReader(new  
InputStreamReader(clientSocket.getInputStream()));`

*Invia una riga  
al server* → `sentence = inFromUser.readLine();  
outToServer.writeBytes(sentence + '\n');`

*Legge la riga  
dal server* → `modifiedSentence = inFromServer.readLine();  
System.out.println("FROM SERVER: " + modifiedSentence);  
clientSocket.close();`

`}`  
`}`

## Esempio: server Java (TCP)

```
import java.io.*;  
import java.net.*;
```

```
class TCPServer {
```

```
    public static void main(String argv[]) throws Exception  
    {
```

```
        String clientSentence;  
        String capitalizedSentence;
```

*Crea una socket  
di benvenuto  
sulla porta 6789*

```
        ServerSocket welcomeSocket = new ServerSocket(6789);
```

*Attende, sulla socket  
di benvenuto,  
un contatto dal client*

```
        while(true) {
```

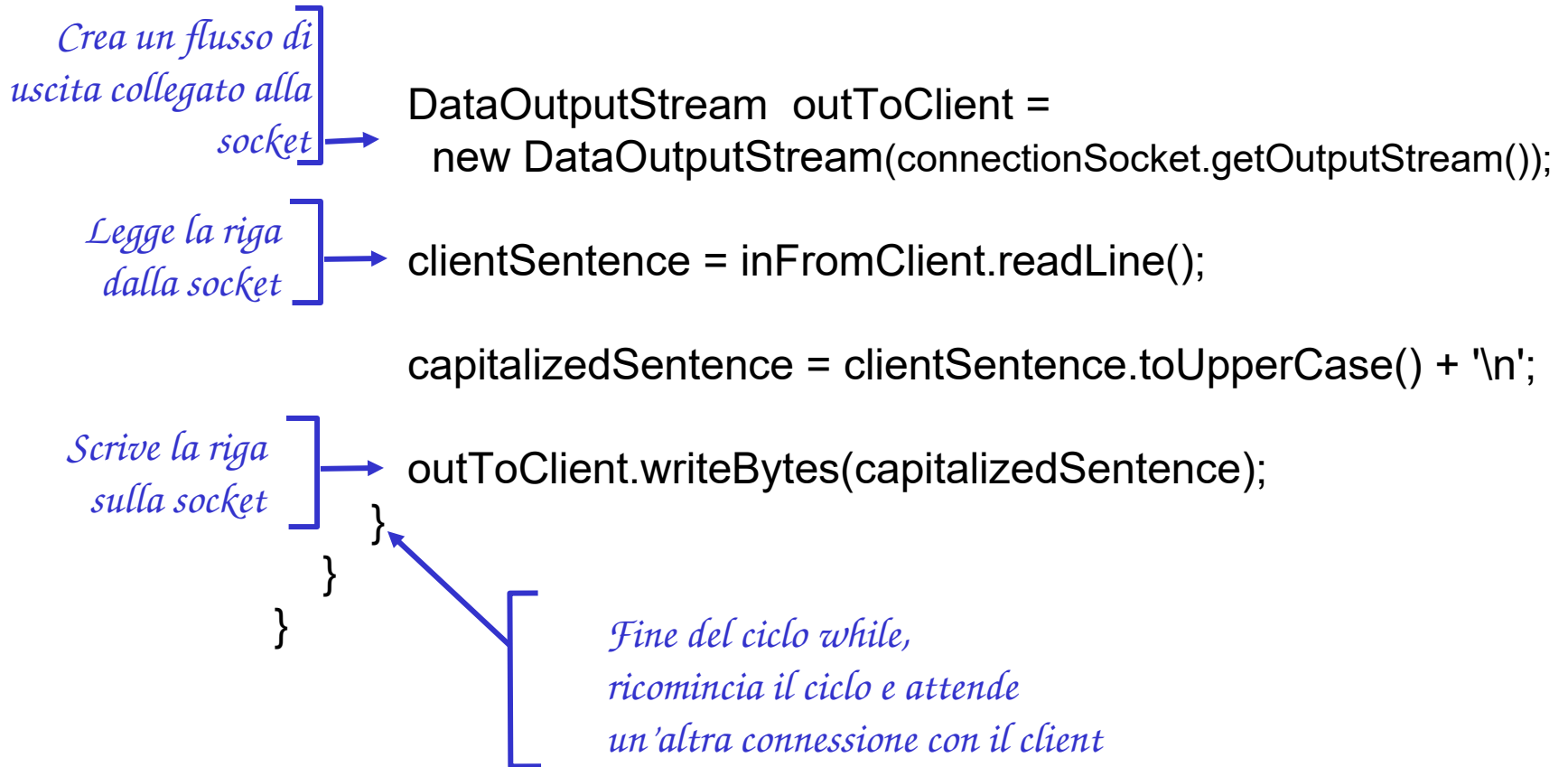
```
            Socket connectionSocket = welcomeSocket.accept();
```

*Crea un  
flusso d'ingresso  
collegato alla socket*

```
            BufferedReader inFromClient =  
                new BufferedReader(new  
                    InputStreamReader(connectionSocket.getInputStream()));
```



## Esempio: server Java (TCP), continua



# Capitolo 2: Livello di applicazione

- ❑ 2.1 Principi delle applicazioni di rete
- ❑ 2.2 Web e HTTP
- ❑ 2.3 FTP
- ❑ 2.4 Posta elettronica
  - ❖ SMTP, POP3, IMAP
- ❑ 2.5 DNS
- ❑ 2.6 Condivisione di file P2P
- ❑ 2.7 Programmazione delle socket con TCP
- ❑ 2.8 Programmazione delle socket con UDP
- ❑ 2.9 Costruire un semplice server web

# Programmazione delle socket con UDP

*UDP: non c'è "connessione" tra client e server*

- ❑ *Non c'è handshaking*
- ❑ *Il mittente allega esplicitamente a ogni pacchetto l'indirizzo IP e la porta di destinazione*
- ❑ *Il server deve estrarre l'indirizzo IP e la porta del mittente dal pacchetto ricevuto*

*UDP: i dati trasmessi possono perdersi o arrivare a destinazione in un ordine diverso da quello d'invio*

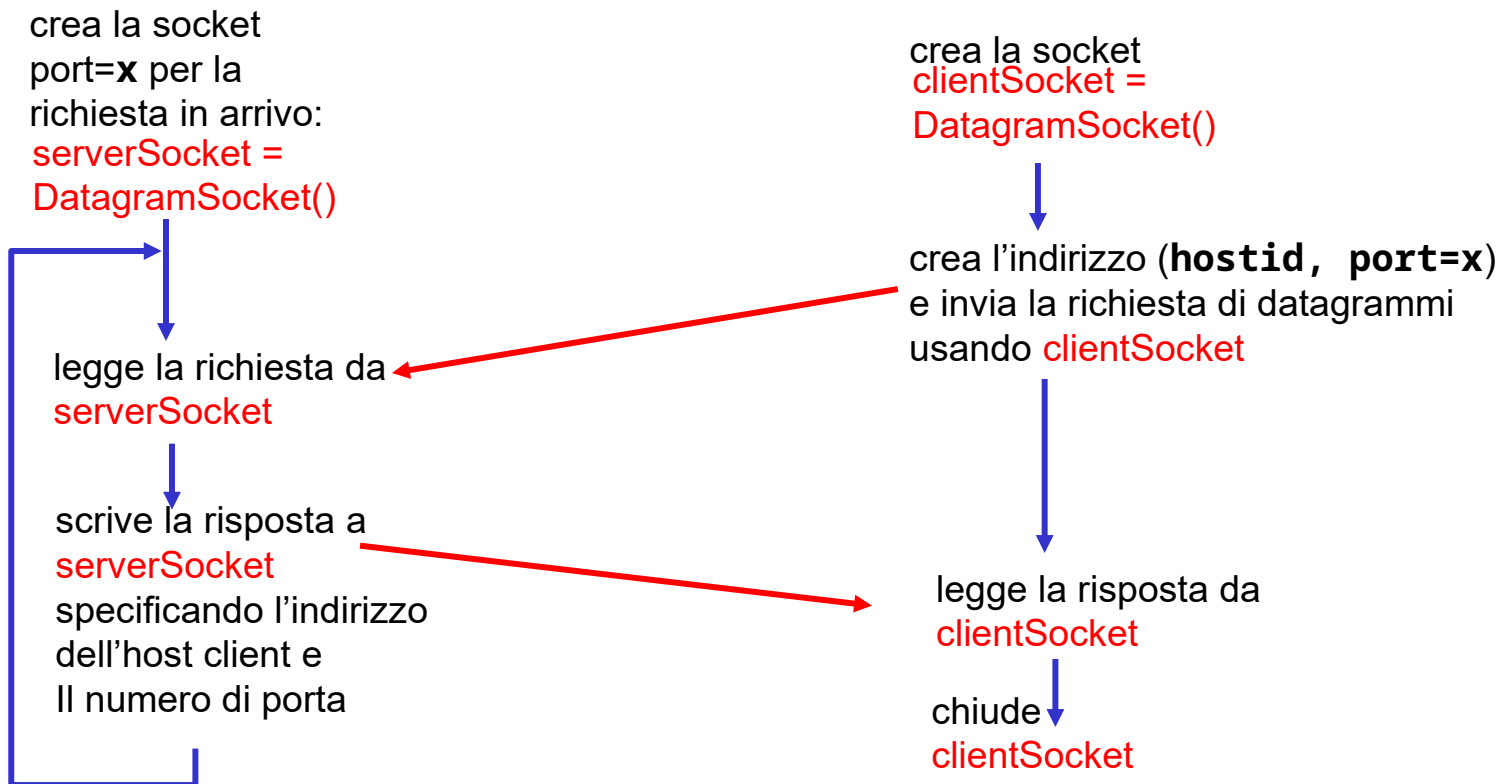
*Punto di vista dell'applicazione*

*UDP fornisce un trasferimento  
inaffidabile di gruppi di  
byte ("datagrammi")  
tra client e server*

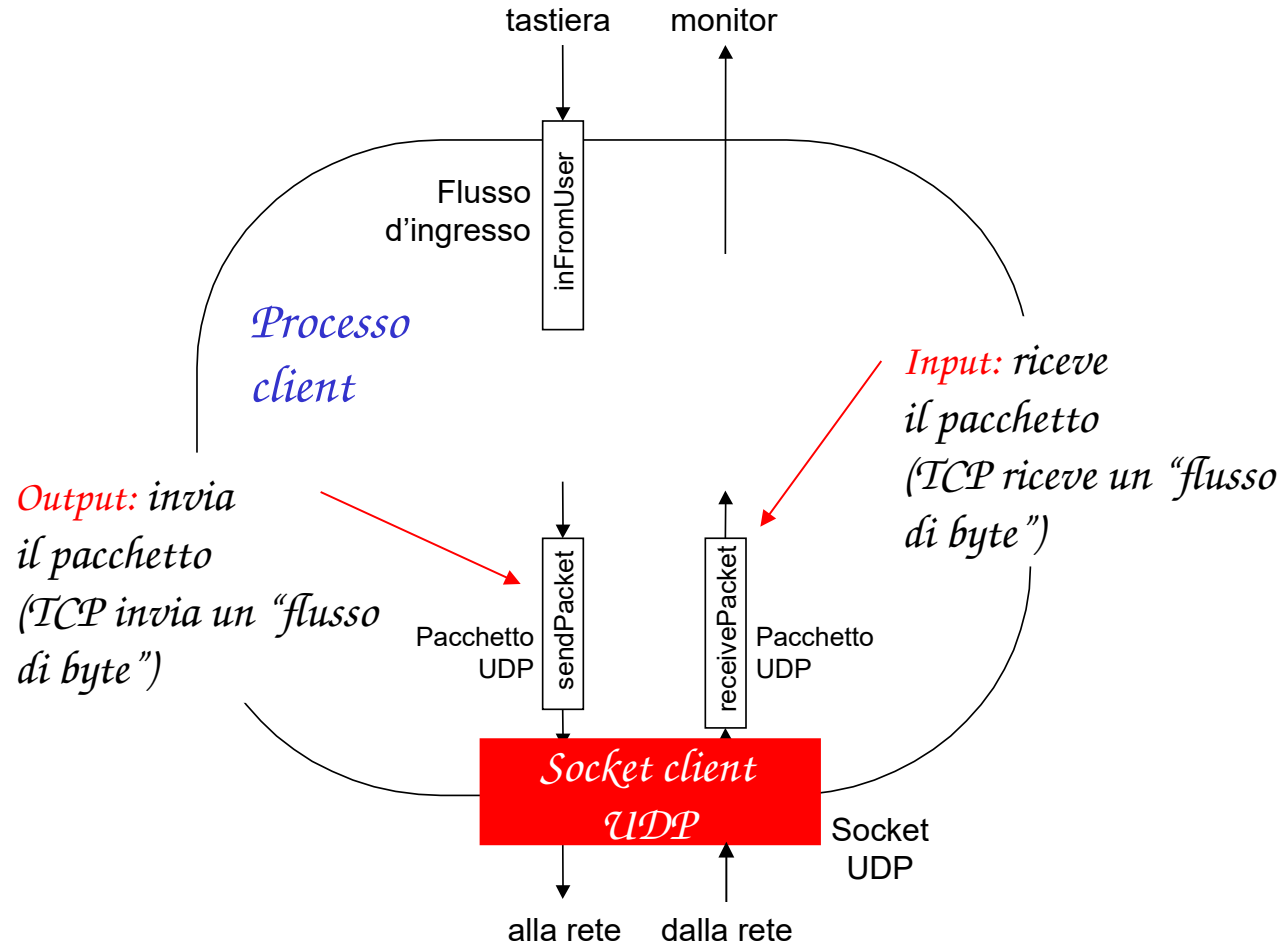
# Interazione delle socket client/server: UDP

*Server* (gira su **hostid**)

*Client*



# Esempio: client Java (UDP)



## Esempio: client Java (UDP)

```
import java.io.*;  
import java.net.*;
```

```
class UDPClient {  
    public static void main(String args[]) throws Exception  
    {
```

*Crea un  
flusso d'ingresso*



```
        BufferedReader inFromUser =
```

*Crea una  
socket client*



```
        new BufferedReader(new InputStreamReader(System.in));
```

```
        DatagramSocket clientSocket = new DatagramSocket();
```

*Traduce il  
nome dell'host  
nell'indirizzo IP  
usando DNS*



```
        InetAddress IPAddress = InetAddress.getByName("hostname");
```

```
        byte[] sendData = new byte[1024];
```

```
        byte[] receiveData = new byte[1024];
```

```
        String sentence = inFromUser.readLine();
```

```
        sendData = sentence.getBytes();
```

## Esempio: client Java (UDP), continua

*Crea il datagramma con i  
dati da trasmettere,  
lunghezza,  
indirizzo IP, porta* → DatagramPacket sendPacket =  
new DatagramPacket(sendData, sendData.length, IPAddress, 9876);

*Invia  
il datagramma  
al server* → clientSocket.send(sendPacket);

*Legge  
il datagramma  
dal server* → DatagramPacket receivePacket =  
new DatagramPacket(receiveData, receiveData.length);

clientSocket.receive(receivePacket);

String modifiedSentence =  
new String(receivePacket.getData());

System.out.println("FROM SERVER:" + modifiedSentence);  
clientSocket.close();  
}

}

## Esempio: server Java (UDP)

```
import java.io.*;  
import java.net.*;
```

```
class UDPServer {  
    public static void main(String args[]) throws Exception  
    {
```

*Crea una socket per  
datagrammi  
sulla porta 9876*



```
        DatagramSocket serverSocket = new DatagramSocket(9876);
```

```
        byte[] receiveData = new byte[1024];  
        byte[] sendData = new byte[1024];
```

```
        while(true)  
        {
```

*Crea lo spazio per  
i datagrammi*



```
            DatagramPacket receivePacket =  
                new DatagramPacket(receiveData, receiveData.length);
```

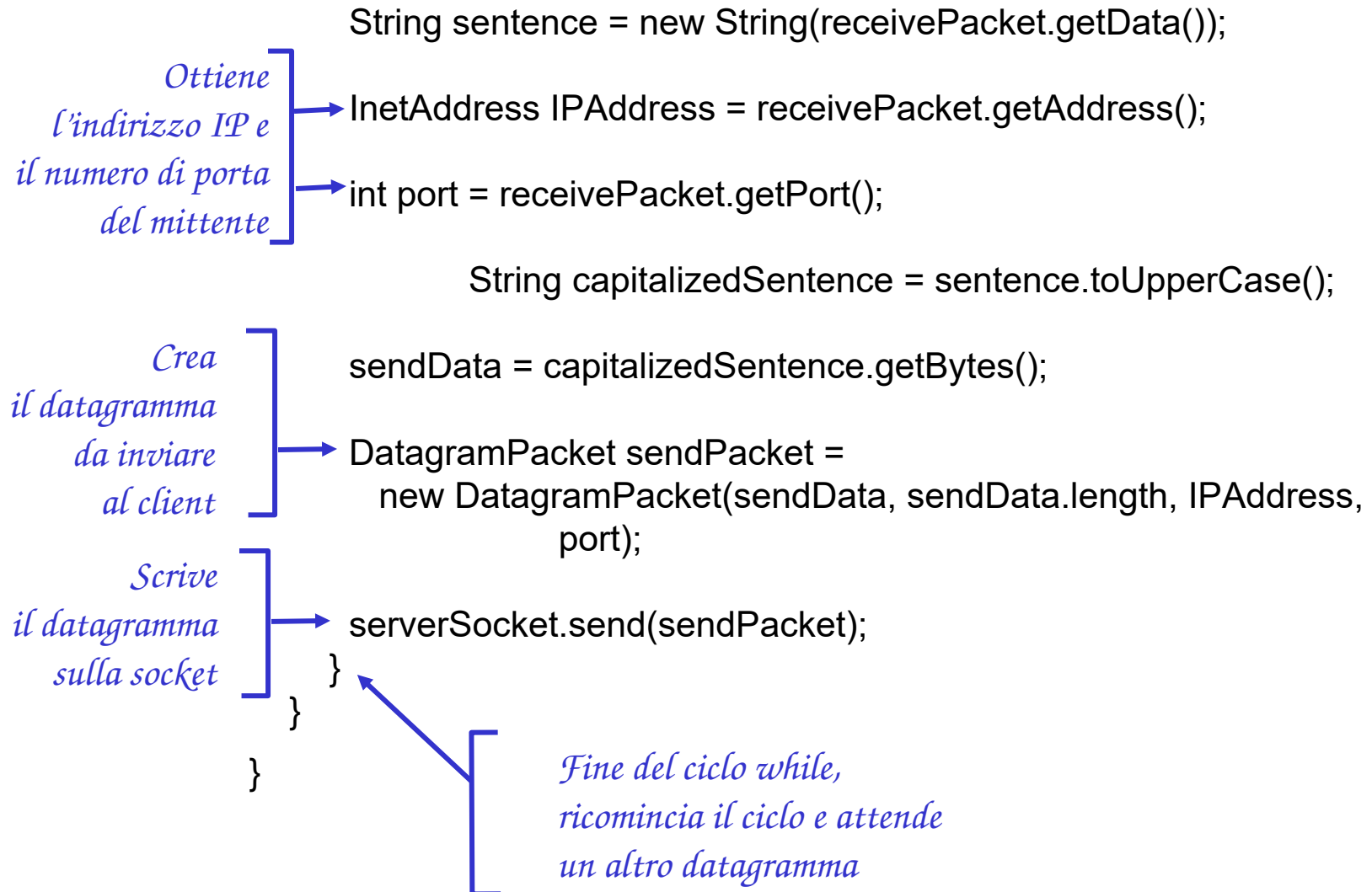
*Riceve i  
datagrammi*



```
            serverSocket.receive(receivePacket);
```



## Esempio: server Java (UDP), continua



# Capitolo 2: Livello di applicazione

- ❑ 2.1 Principi delle applicazioni di rete
- ❑ 2.2 Web e HTTP
- ❑ 2.3 FTP
- ❑ 2.4 Posta elettronica
  - ❖ SMTP, POP3, IMAP
- ❑ 2.5 DNS
- ❑ 2.6 Condivisione di file P2P
- ❑ 2.7 Programmazione delle socket con TCP
- ❑ 2.8 Programmazione delle socket con UDP
- ❑ 2.9 Costruire un semplice server web

# Costruire un semplice server web

- ❑ *gestisce una richiesta HTTP*
- ❑ *accetta la richiesta*
- ❑ *analizza l'intestazione*
- ❑ *prende il file richiesto dal file system del server*
- ❑ *crea un messaggio di risposta HTTP:*
  - ❖ *righe di intestazione + file*
- ❑ *invia la risposta al client*
- ❑ *dopo avere creato il server, potete richiedere il file utilizzando un browser (ad esempio, Internet Explorer)*
- ❑ *Vedere il testo per i dettagli*

## domande

- ❑ *E' possibile implementare un servizio di comunicazione affidabile usando udp?*
  - ❖ *SI (implemntando I controlli a lato applicazione)*
- ❑ *Quale sarebbe il vantaggio?*
  - ❖ *No controllo congestione!! ☺*

# Capitolo 2: Riassunto

*Lo studio delle applicazioni di rete adesso è completo!*

- *Architetture delle applicazioni*
  - ❖ *client-server*
  - ❖ *P2P*
  - ❖ *ibride*
- *Requisiti dei servizi delle applicazioni:*
  - ❖ *affidabilità, ampiezza di banda, ritardo*
- *Modello di servizio di trasporto di Internet*
  - ❖ *orientato alle connessioni, affidabile: TCP*
  - ❖ *inaffidabile, datagrammi: UDP*
- *Protocolli specifici:*
  - ❖ *HTTP*
  - ❖ *FTP*
  - ❖ *SMTP, POP, IMAP*
  - ❖ *DNS*
- *Programmazione delle socket*

# Riassunto

## Molto importante: conoscere i protocolli

- ❑ *Tipico scambio di messaggi di richiesta/risposta:*
  - ❖ *il client richiede informazioni o servizi*
  - ❖ *il server risponde con dati e codici di stato*
- ❑ *Formati dei messaggi:*
  - ❖ *intestazioni: campi che forniscono informazioni sui dati*
  - ❖ *dati: informazioni da comunicare*
- ❑ *Controllo o messaggi di dati*
  - ❖ *in banda, fuori banda*
- ❑ *Architettura centralizzata o decentralizzata*
- ❑ *Protocollo senza stato o con stato*
- ❑ *Trasferimento di messaggi affidabile o inaffidabile*
- ❑ *“Complessità nelle parti periferiche della rete”*