



Reti di Calcolatori

Introduzione

Introduzione

► Perché le reti di calcolatori

- Reti aziendali
- Reti domestiche
- Reti Mobili



► Terminologia e classificazione delle reti

- Tecnologia di trasmissione
- Scala della rete



Convergenza delle tecnologie

- ▶ Diffusione della tecnologia digitale
- ▶ Distribuzione, elaborazione, memorizzazione dell'informazione
- ▶ Fusione di calcolatori e comunicazione
- ▶ Influenza sulle direzioni dello sviluppo tecnologico
- ▶ Impatto sociale
 - ▶ Organizzazione del lavoro
 - ▶ Comunicazione personale
 - ▶ Distribuzione e accesso alle informazioni



Reti di calcolatori

- ▶ Modello basato su calcolatori **indipendenti** e **interconnessi** fra loro
- ▶ Insieme di calcolatori **autonomi** collegati ovvero in grado di scambiare informazioni fra di loro
- ▶ Il collegamento può utilizzare tecnologie varie (fili elettrici, onde radio, fibre ottiche)
- ▶ L'insieme dei calcolatori è in genere **eterogeneo** (architettura hardware, sistema operativo)



Reti di calcolatori/Sistemi distribuiti

Reti di calcolatori

- ▶ Il fuoco è sulla trasmissione dati fra agenti (software o hardware) e sulla modalità con cui avviene
- ▶ Si specificano esplicitamente la macchina remota e il servizio da usare
- ▶ Fornisce meccanismi standardizzati e interoperabili per la trasmissione dati (dispositivi di rete, librerie software e supporto a livello di sistema operativo)

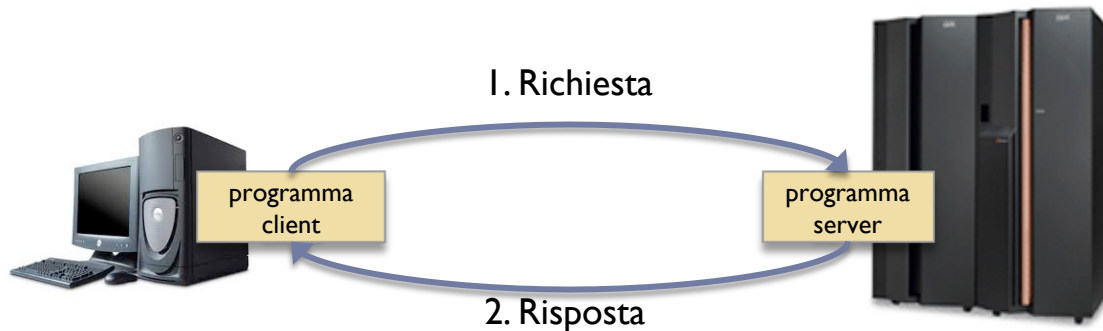
Sistemi distribuiti

- ▶ Astrazione ad un livello superiore: piattaforma software costruita sui servizi della rete ([middleware](#))
- ▶ Si enfatizzano gli aspetti funzionali (accesso ad un servizio)
 - ▶ condivisione file o risorse
 - ▶ esecuzione distribuita di programmi
 - ▶ il Web visto come un “documento” o una piattaforma di servizi distribuita
- ▶ L'accesso alla rete è trasparente: si specifica solo il servizio che si vuole ottenere (anche dal punto di vista dello sviluppo)
- ▶ Il middleware si occupa di allocare le risorse necessarie in base alla richiesta e alla disponibilità e si occupa dei dettagli della trasmissione

Applicazioni delle reti di calcolatori 1

► Reti aziendali

- Condivisione delle risorse (programmi, periferiche, dati)
 - Risorse hardware (calcolo, memoria, stampanti)
 - Informazioni e documenti (File sharing, Intranet, Internet)
 - Amministrazione e distribuzione remota del software (desktop sharing)
- Indipendenza dalla posizione fisica delle risorse e dell'utente
 - Server centralizzati e postazioni di lavoro client remote
 - Modello client/server



Applicazioni delle reti di calcolatori 2

- ▶ Supporto alla comunicazione aziendale
 - ▶ email, telefonia IP (Voice over IP – VoIP)
 - ▶ Chat e lavagna condivisa
 - ▶ Videoconferenza
- ▶ Gestione della produzione e dei servizi interni
 - ▶ monitoraggio remoto (impianti, telemedicina)
 - ▶ Reti di sensori
- ▶ Supporto al business
 - ▶ e-commerce (B2B, B2C, ..)
 - ▶ Customer care
- ▶ Le reti sono una infrastruttura indispensabile nelle aziende moderne (e non...)

Applicazioni delle reti di calcolatori 3

► Reti domestiche

- Accesso ad Internet da casa
- I dispositivi elettronici con funzionalità di rete
 - Intrattenimento (IPTV, media streaming, network gaming)
 - Console per videogiochi, decoder/set-top box, TV
- Domotica
 - sensori per monitoraggio intelligente (videosorveglianza, utenze domestiche, ..)
- Comunicazione interpersonale e servizi
 - Messaggistica istantanea, Social networks, generazione collaborativa di contenuti (Wiki)
 - Servizi finanziari (home banking), acquisti online, servizi al cittadino (e-government)



Applicazioni delle reti di calcolatori 4

► Utenti mobili

- Diffusione di computer portatili, palmari, tablet e smartphone
- Connessione ad Internet
 - reti cellulari/Wi-Fi hotspot
- Applicazioni personali e aziendali
 - Servizi georeferenziati (uso GPS)
 - Monitoraggio di flotte di autoveicoli
 - Monitoraggio dello stato dei parcheggi
 - Reti di sensori



Vantaggi delle reti

- ▶ Alta affidabilità
 - ▶ Introduzione di ridondanze (es. replicazione di archivi)
- ▶ Riduzione dei costi
 - ▶ Elaborazione su **cluster** di computer rispetto a mainframe multiprocessore
 - ▶ Architetture **client-server** rispetto a mainframe-terminali
 - ▶ Scalabilità della soluzione

Impatto sociale

- ▶ **Rivoluzione nella comunicazione**
 - ▶ Pubblicazione ed accesso alle informazioni (WWW)
 - ▶ Comunicazione personale (e-mail, chat, ICQ, VoIP)
 - ▶ Strumenti collaborativi (videoconferenza, newsgroups)
 - ▶ Social networks
- ▶ **Impatto sull'economia (e-commerce)**
- ▶ **Intrattenimento**
 - ▶ Video on demand, Web radio, network gaming
- ▶ **Problemi etici, di privacy e di sicurezza**
 - ▶ Controllo delle informazioni, verifica delle sorgenti e delle informazioni
 - ▶ Cyber-criminalità (furto di identità, spam, virus, phishing,...)

Aspetti tecnologici

La definizione di una rete include

- ▶ **Hardware**

- ▶ Apparat di interconnessione e cablaggi
- ▶ Apparat per il controllo della trasmissione
- ▶ Terminali (computer, smartphone,...)

- ▶ **Software**

- ▶ Gestione dei dispositivi di rete (schede e apparati)
- ▶ Instradamento delle informazioni fra i computer in rete
- ▶ Supporto al trasferimento di messaggi o flussi di dati (socket)
- ▶ Servizi e applicazioni

Caratteristiche del sistema di comunicazione

▶ Affidabilità

- ▶ Tecniche per individuare dati corrotti o persi
- ▶ Soluzione per evitare la duplicazione dei dati (identificazione dei pacchetti)
- ▶ Garanzia dell'integrità dell'informazione ricevuta in un trasferimento dati
- ▶ Definizione di livelli di affidabilità diversi in base al tipo di informazione (video/dati) e di servizio richiesto (Quality of Service – QoS)
- ▶ Robustezza in caso di guasti (percorsi ridondanti)

▶ Efficienza

- ▶ Uso ottimale delle risorse disponibili
- ▶ Cammino ottimale fra mittente e destinatario

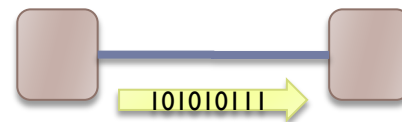
▶ Interoperabilità fra ambienti applicativi eterogenei

- ▶ Necessità di definire degli standard/protocolli comuni

Interconnessione degli apparati

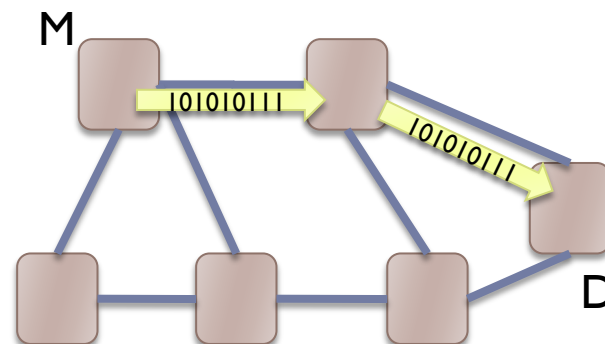
► Due nodi

- Problemi “elettrici”
- Sincronizzazione del trasferimento
 - Ready To Send (RTS)
 - Clear To Send (CTS)



► Più di due nodi

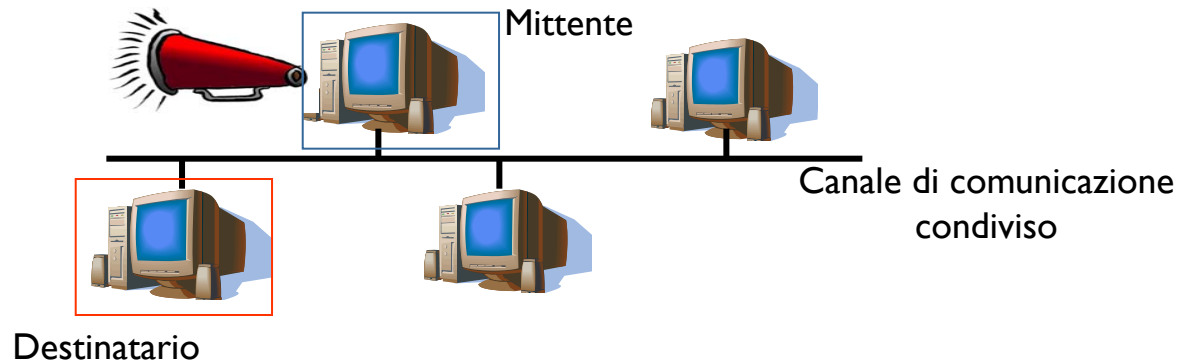
- Come connettere i nodi
- Come inoltrare i dati attraverso la rete fra un mittente e un destinatario
 - Indirizzamento
 - Routing dei messaggi
- Come gestire comunicazioni simultanee fra nodi diversi



Modalità di trasmissione - broadcast

► Broadcast

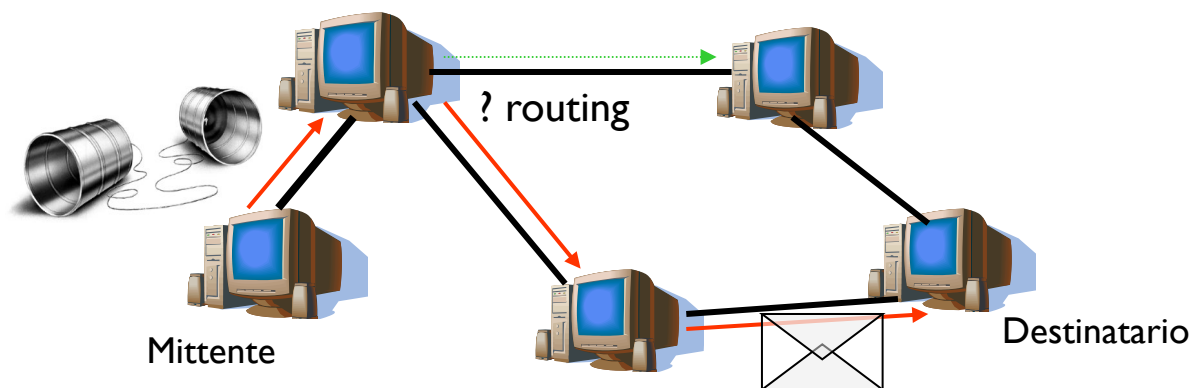
- Il canale di comunicazione è condiviso fra tutti i dispositivi
- I messaggi inviati (pacchetti) sono ricevuti da tutti i dispositivi
- Nel pacchetto è indicato l'indirizzo del destinatario
- Solo il dispositivo destinatario accetta il pacchetto trasmesso (**trasmissione unicast**)
- E' possibile trasmettere un messaggio a tutti i dispositivi (**indirizzo broadcast**) o un gruppo di essi (**indirizzo multicast**)



Modalità di trasmissione – punto-punto

► Punto a punto (Point-to-point)

- Le connessioni sono fra coppie di nodi
- Per passare dal mittente al destinatario il messaggio deve attraversare nodi intermedi
- In genere ci possono essere più cammini e deve essere scelto il migliore (in base a un certo criterio)



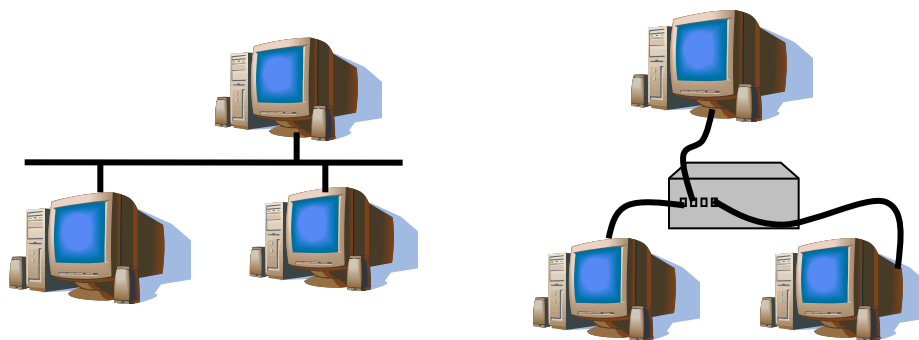
Classificazione in base alla scala

Distanza fra le unità di elaborazione	Ambiente	
0.1m	Circuito	Calcolatori multiprocessore
1m	Sistema	
10m	Stanza	
100m	Edificio	
1Km	Campus	
10Km	Città	Rete Locale (LAN)
100Km	Nazione	
1000Km	Continente	Rete Metropolitana (MAN)
10000Km	Pianeta	
		Rete Geografica (WAN)
		Internet

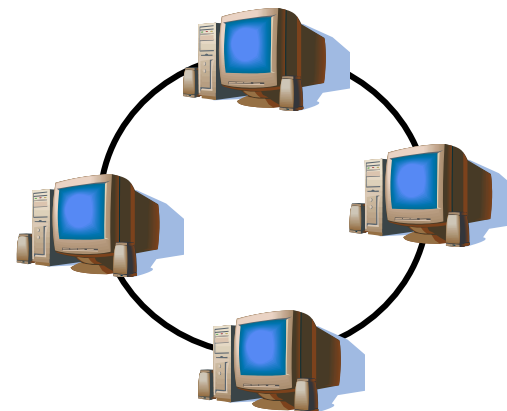
Reti Locali (LAN)

- ▶ Possono essere cablate o con copertura wireless
- ▶ Il tempo peggiore di comunicazione è limitato e noto
- ▶ Hanno una struttura broadcast con arbitraggio degli accessi
- ▶ Hanno tipicamente velocità di 100-1000 Mbps

IEEE 802.3 (Ethernet)

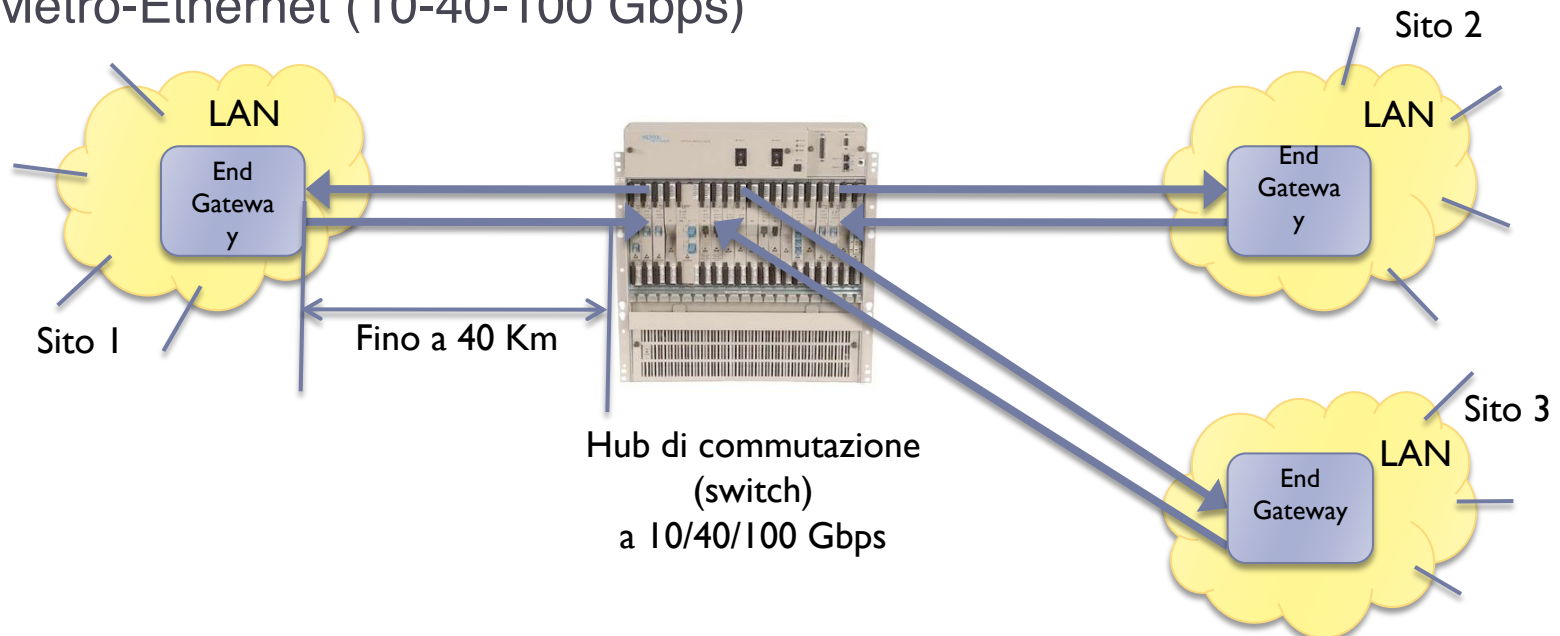


IEEE 802.5 (IBM Token Ring)



Reti Metropolitane (MAN)

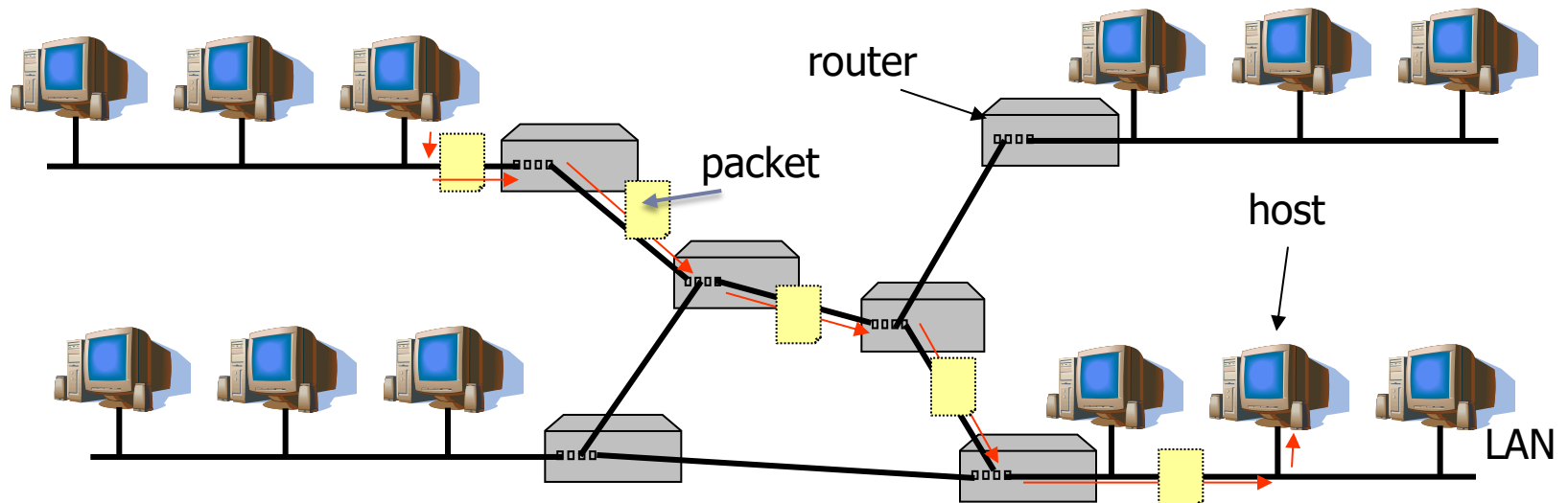
- ▶ Può supportare dati, voce, video
- ▶ Interconnette più LAN con dorsali ad alta velocità
- ▶ Utilizza tecnologie ottimizzate per distanze maggiori delle LAN
 - ▶ ATM (Asynchronous Transfer Mode) e SDH (Synchronous Digital Hierarchy)
 - ▶ Metro-Ethernet (10-40-100 Gbps)



Reti Geografiche

Wide Area Networks (WAN)

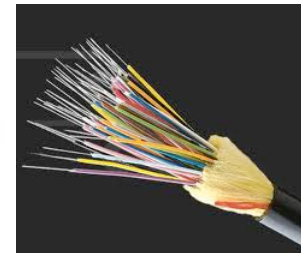
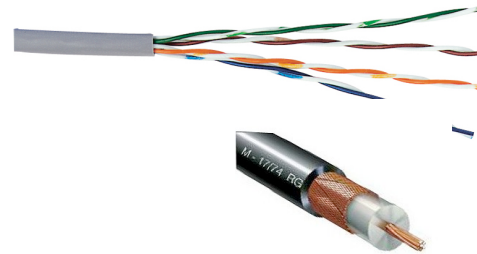
- ▶ I dispositivi (**host**) sono collegati a sottoreti
- ▶ Vengono utilizzati elementi di commutazione (**router**) che decidono su quale linea di uscita devono inoltrare il messaggio per recapitarlo al destinatario
 - ▶ point-to-point (le connessioni sono fra nodi)
 - ▶ store-and-forward (i router memorizzano i pacchetti in code interne)
 - ▶ packet-switched (l'unità di informazione scambiata è il pacchetto)



Mezzi trasmissivi

▶ Linee cablate

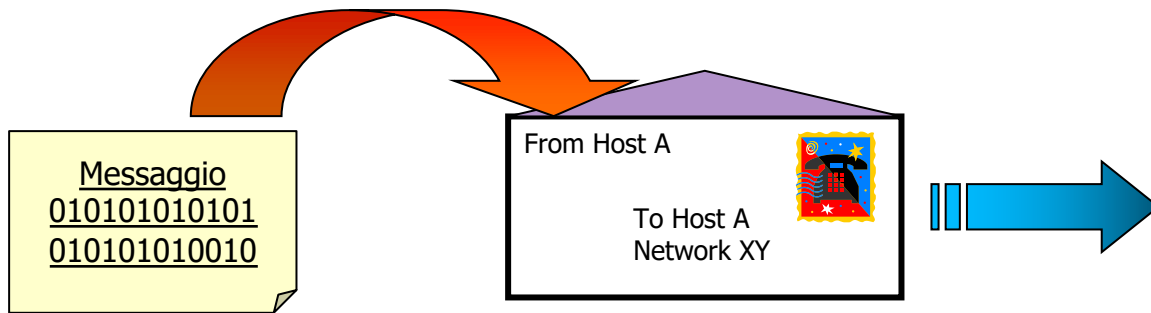
- ▶ Rame
 - ▶ Doppini Telefonici (Unshielded Twisted Pair - UTP)
 - ▶ Cavi coassiali
- ▶ Fibre ottiche



▶ Wireless

- ▶ Onde acustiche
- ▶ Luce
- ▶ Raggi infrarossi
- ▶ Radiofrequenza
- ▶ Microonde





Reti di calcolatori

Protocolli e standard

Organizzazione in livelli e protocolli

- ▶ Necessità di definire **protocolli** e **standard**
 - ▶ Un protocollo definisce la **modalità** con la quale avviene la comunicazione fra due agenti
 - ▶ La standardizzazione del protocollo garantisce l'**interoperabilità** fra dispositivi e/o agenti software sviluppati indipendentemente
 - ▶ I protocolli di rete implementati nel sistema operativo Linux e in Windows
 - ▶ Una scheda di rete e l'apparato di interconnessione (switch) a cui si collega
 - ▶ Il server Web Apache e il browser Web Firefox
- ▶ La rete è organizzata come una serie di strati (**livelli**)
 - ▶ Ogni livello fornisce una serie di servizi al livello superiore
 - ▶ Ogni livello di un host comunica con il livello corrispondente di un altro host utilizzando un opportuno protocollo

Protocolli e standard

- ▶ Per scambiarsi dati due agenti devono attenersi ad un protocollo predefinito e completamente specificato
- ▶ Il protocollo definisce le regole da seguire per il trasferimento dei dati
 - ▶ **Sintassi** - formato dei dati (campi del pacchetto)
 - ▶ **Semantica** - significato (uso dei campi del pacchetto)
 - ▶ **Sincronizzazione** - modalità di invio dei dati quando mittente e destinatario elaborano i dati a velocità diverse
 - ▶ Notifica della disponibilità di dati da inviare (Mittente) *Request-To-Send*
 - ▶ Notifica della disponibilità a ricevere dati (Ricevente) *Clear-To-Send*
- ▶ Gli **standard** forniscono le linee guida per assicurare il livello di interconnessione
 - ▶ **de facto** – descrivono implementazioni già esistenti (es. Internet)
 - ▶ **de jure** – si definisce lo standard e le implementazioni si devono attenere ad esso

Organizzazioni di standardizzazione

- ▶ ISO (International Standards Organization)
 - ▶ 82 nazioni dal 1947
- ▶ ITU-T (International Telecommunications Union-Telecommunication Standards Sector)
 - ▶ Standard per le telecomunicazioni – telefonia e trasferimento dati (ex. CCITT)
- ▶ ANSI (American National Standards Institute)
 - ▶ Organizzazione privata per l'unificazione degli standard in USA
- ▶ IEEE (Institute of Electrical and Electronics Engineers)
- ▶ EIA (Electronic Industries Association)

Forum

- ▶ Necessità di maggiore dinamismo rispetto ai tempi di ratifica degli standard
- ▶ Controllano, valutano e rendono uniformi le nuove tecnologie
 - ▶ ATM forum
 - ▶ E' costituito dai rivenditori di hardware e software che sostengono la tecnologia ATM (Asynchronous Transfer Mode)
 - ▶ Metro-Ethernet Forum (MEF) dal 2001
 - ▶ Ha la missione di accelerare l'adozione di servizi e reti Ethernet come portanti



- 186 Member Companies
- 92 Service Providers
- Global Representation

Gli standard Internet

- ▶ **Internet draft**

- ▶ Documento non ufficiale che vive per circa 6 mesi

- ▶ **Request for Comment (RFC)**

- ▶ Internet draft pubblicato su suggerimento delle autorità per la regolamentazione della rete Internet
 - ▶ Ogni RFC è protocollato, gli viene assegnato un codice e reso disponibile (www.rfc-editor.org)

- ▶ **Livelli di maturazione degli RFC**

- ▶ Un RFC attraversa sei livelli di maturazione

Livelli di un RFC

- ▶ Proposta di standard
 - ▶ Specificazione stabile e interessante – è provata e usata
- ▶ Bozza
 - ▶ Richiede almeno due realizzazioni indipendenti e inter-operative
- ▶ Standard Internet
 - ▶ Richiede che sia stato dimostrato l'uso concreto con successo
- ▶ Livello storico
- ▶ Livello sperimentale
- ▶ Livello informativo

Livelli di impiego di un RFC

▶ Necessario

- ▶ Deve essere usato necessariamente da tutti i sistemi Internet per assicurare un livello di conformità minimale (es. IP)

▶ Consigliato

- ▶ Non è necessario ma viene consigliato per la sua utilità (es. TELNET)

▶ Facoltativo

- ▶ Può essere usato per scopi specifici in un sistema particolare

▶ D'uso limitato

- ▶ Dovrebbe essere usato in situazioni particolari

▶ Sconsigliato

RFC 0791 – Internet Protocol 1

0791 Internet Protocol. J. Postel. Sep-01-1981. (Format: TXT=97779 bytes) (Obsoletes RFC0760) (Updated by RFC1349) (Also STD0005) (Status: STANDARD)

RFC: 791

INTERNET PROTOCOL
DARPA INTERNET PROGRAM
PROTOCOL SPECIFICATION

September 1981

prepared for

Defense Advanced Research Projects Agency
Information Processing Techniques Office
1400 Wilson Boulevard
Arlington, Virginia 22209

by

Information Sciences Institute
University of Southern California
76 Admiralty Way
Marina del Rey, California 90291

September 1981

Internet Protocol

TABLE OF CONTENTS

PREFACE	iii
1. INTRODUCTION.....	1
1.1 Motivation	1
1.2 Scope	1
1.3 Interfaces	1
1.4 Operation	2
2. OVERVIEW	5
2.1 Relation to Other Protocols	9
2.2 Model of Operation	5
2.3 Function Description	7
2.4 Gateways	9
3. SPECIFICATION	11
3.1 Internet Header Format	11
3.2 Discussion	23
3.3 Interfaces	31
APPENDIX A: Examples & Scenarios	34
APPENDIX B: Data Transmission Order	39
GLOSSARY	41
REFERENCES	45

[Page i]

RFC 0791 – Internet Protocol 2

INTERNET PROTOCOL

DARPA INTERNET PROGRAM PROTOCOL SPECIFICATION

1. INTRODUCTION

1.1. Motivation

The Internet Protocol is designed for use in interconnected systems of packet-switched computer communication networks. Such a system has been called a "catenet" [1]. The internet protocol provides for transmitting blocks of data called datagrams from sources to destinations, where sources and destinations are hosts identified by fixed length addresses. The internet protocol also provides for fragmentation and reassembly of long datagrams, if necessary, for transmission through "small packet" networks.

1.2. Scope

The internet protocol is specifically limited in scope to provide the functions necessary to deliver a package of bits (an internet datagram) from a source to a destination over an interconnected system of networks. There are no mechanisms to augment end-to-end data reliability, flow control, sequencing, or other services commonly found in host-to-host protocols. The internet protocol can capitalize on the services of its supporting networks to provide various types and qualities of service.

1.3. Interfaces

This protocol is called on by host-to-host protocols in an internet environment. This protocol calls on local network protocols to carry the internet datagram to the next gateway or destination host.

For example, a TCP module would call on the internet module to take a TCP segment (including the TCP header and user data) as the data portion of an internet datagram. The TCP module would provide the addresses and other parameters in the internet header to the internet module as arguments of the call. The internet module would then create an internet datagram and call on the local network interface to transmit the internet datagram.

In the ARPANET case, for example, the internet module would call on a

[Page 1]

RFC 0791 – Internet Protocol 3

September 1981

Internet Protocol

3. SPECIFICATION

3.1. Internet Header Format

A summary of the contents of the internet header follows:

0				1				2				3			
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5
+	-	+	-	+	-	+	-	+	-	+	-	+	-	+	-
Version				IHL				Type of Service				Total Length			
+	-	+	-	+	-	+	-	+	-	+	-	+	-	+	-
								Identification				Flags		Fragment Offset	
+	-	+	-	+	-	+	-	+	-	+	-	+	-	+	-
				Time to Live				Protocol				Header Checksum			
+	-	+	-	+	-	+	-	+	-	+	-	+	-	+	-
								Source Address							
+	-	+	-	+	-	+	-	+	-	+	-	+	-	+	-
								Destination Address							
+	-	+	-	+	-	+	-	+	-	+	-	+	-	+	-
								Options				Padding			
+	-	+	-	+	-	+	-	+	-	+	-	+	-	+	-

Example Internet Datagram Header
Figure 4.

Note that each tick mark represents one bit position.

Version: 4 bits

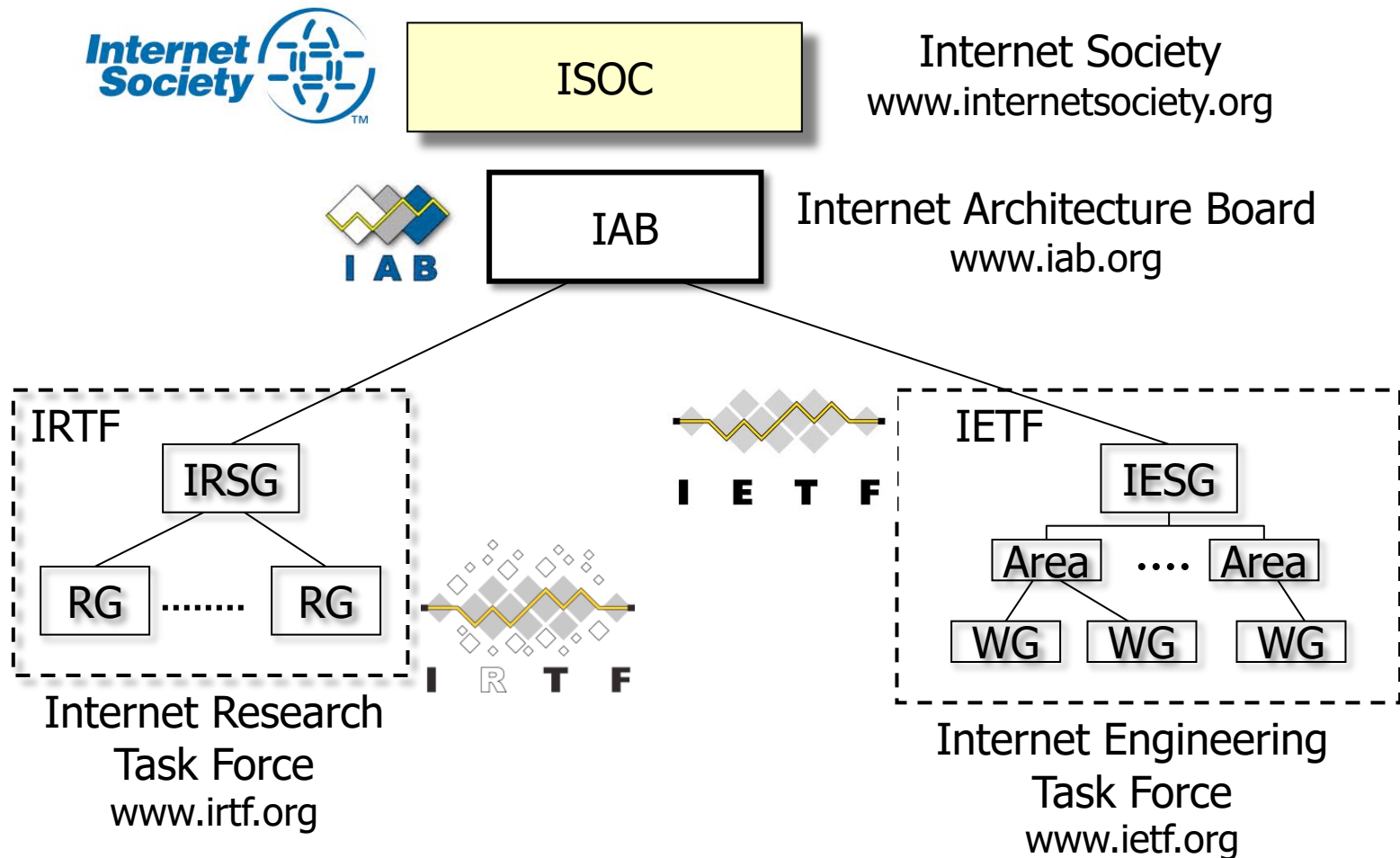
The Version field indicates the format of the internet header. This document describes version 4.

IHL: 4 bits

Internet Header Length is the length of the internet header in 32 bit words, and thus points to the beginning of the data. Note that the minimum value for a correct header is 5.

[Page 11]

Internet Ecosystem



ISOC e IAB

- ▶ ISOC - www.internetsociety.org
 - ▶ Costituita nel 1992 per coadiuvare il processo della creazione degli standard Internet
- ▶ IAB - www.iab.org
 - ▶ E' il consulente tecnico dell'ISOC
 - ▶ Sovraintende lo sviluppo del TCP/IP Protocol Suite
 - ▶ Coadiuvava i ricercatori della comunità Internet
 - ▶ Gestisce la pubblicazione degli RFC
 - ▶ Ha due componenti: IETF e IRTF

The "Internet Architecture Board" (IAB) sounds as if it is something rather grand, perhaps consisting of a group of people in formal business clothes, sitting around an impressive oak table, under the watchful eyes of an oil painting of The Founder of the Internet. The reality is rather different...

..

Today, the IAB consists of thirteen members. Of these, six are nominated each year by a nominating committee drawn from the Internet Engineering Task Force (IETF) for a two year term.

..

Currently, the IAB holds two 90-minute business meetings via telephone conference each month.

IETF e IRTF

▶ IETF - www.ietf.org

- ▶ E' un forum di gruppi di lavoro gestiti dall'Internet Engineering Steering Group (IESG)
- ▶ Identifica i problemi operazionali e propone soluzioni
- ▶ Sviluppa e recensisce le specifiche
- ▶ I gruppi di lavoro sono divisi in aree su argomenti diversi
 - ▶ Applicazioni, protocolli Internet, Routing, Operazioni e gestione, Applicazioni Real-Time e infrastruttura, Routing, Sicurezza, Trasporto
 - ▶ Un gruppo di lavoro è tipicamente creato per affrontare un problema specifico ed hanno un tempo di vita breve

▶ IRTF - www.irtf.org

- ▶ E' un forum di gruppi di lavoro su aspetti di ricerca a lungo termine
 - ▶ Anti-Spam Research Group (ASRG), Peer to Peer Research Group (P2PRG), Routing Research Group (RRG),...

IANA e NIC

- ▶ IANA (Internet Assigned Numbers Authority)
www.iana.org



- ▶ Gestione dei nomi di domini (DNS Root Zone)
- ▶ Gestione degli indirizzi IP e degli Autonomous System Numbers
 - ▶ IPv4 (32 bit) e IPv6 (128 bit)
- ▶ Gestione dei numeri e nomi di protocollo

- ▶ ICANN (Internet Corporation for Assigned Names and Numbers) - www.icann.org



- ▶ ICANN è una corporazione privata gestita da una commissione Internazionale che assunto il ruolo di IANA dal 1998

- ▶ NIC (Network Information Center) - www.internic.net

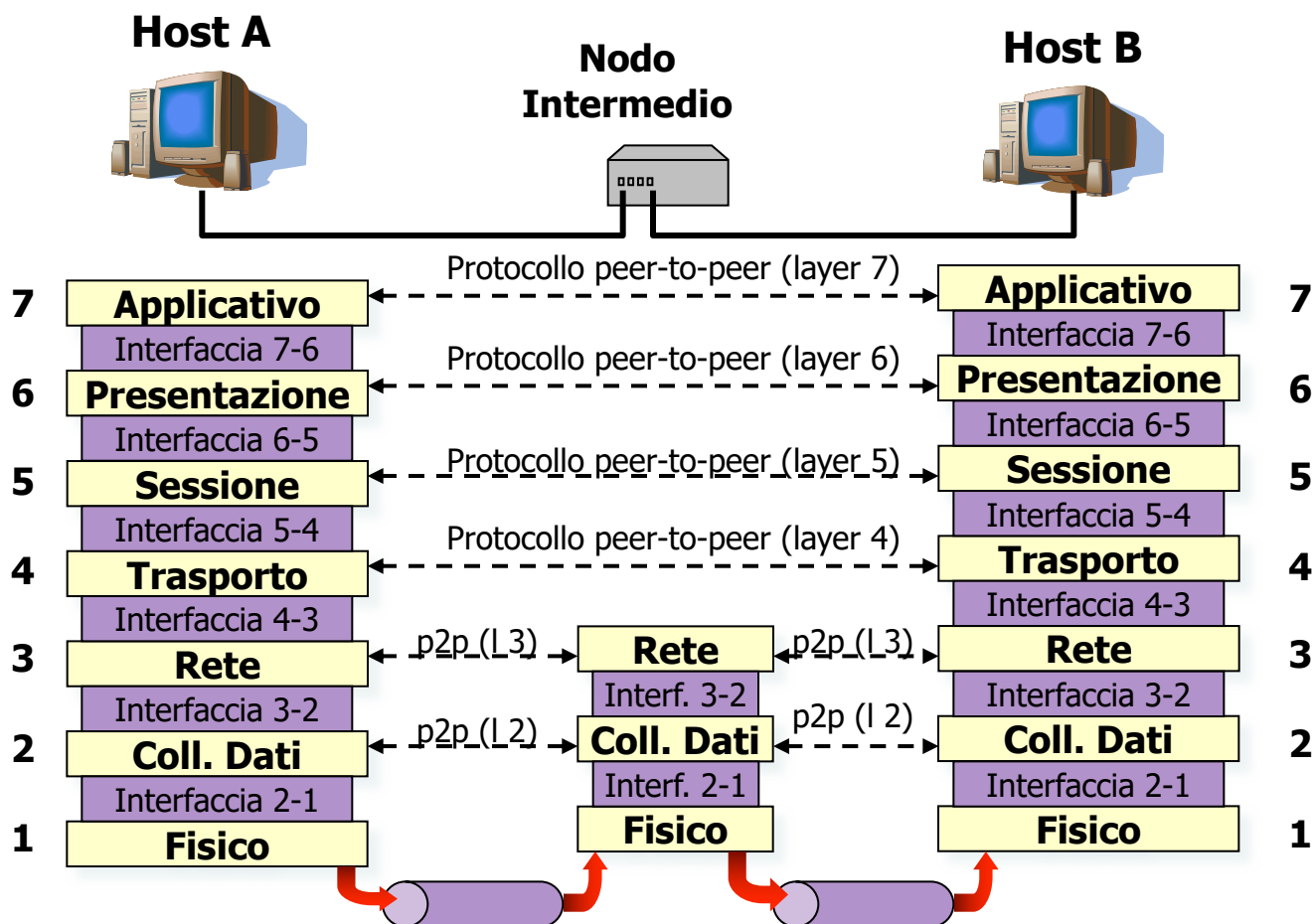
- ▶ Si occupa di raccogliere e diffondere le informazioni relative alla registrazione dei domini e indirizzi IP (WhoIS)

Modello ISO-OSI

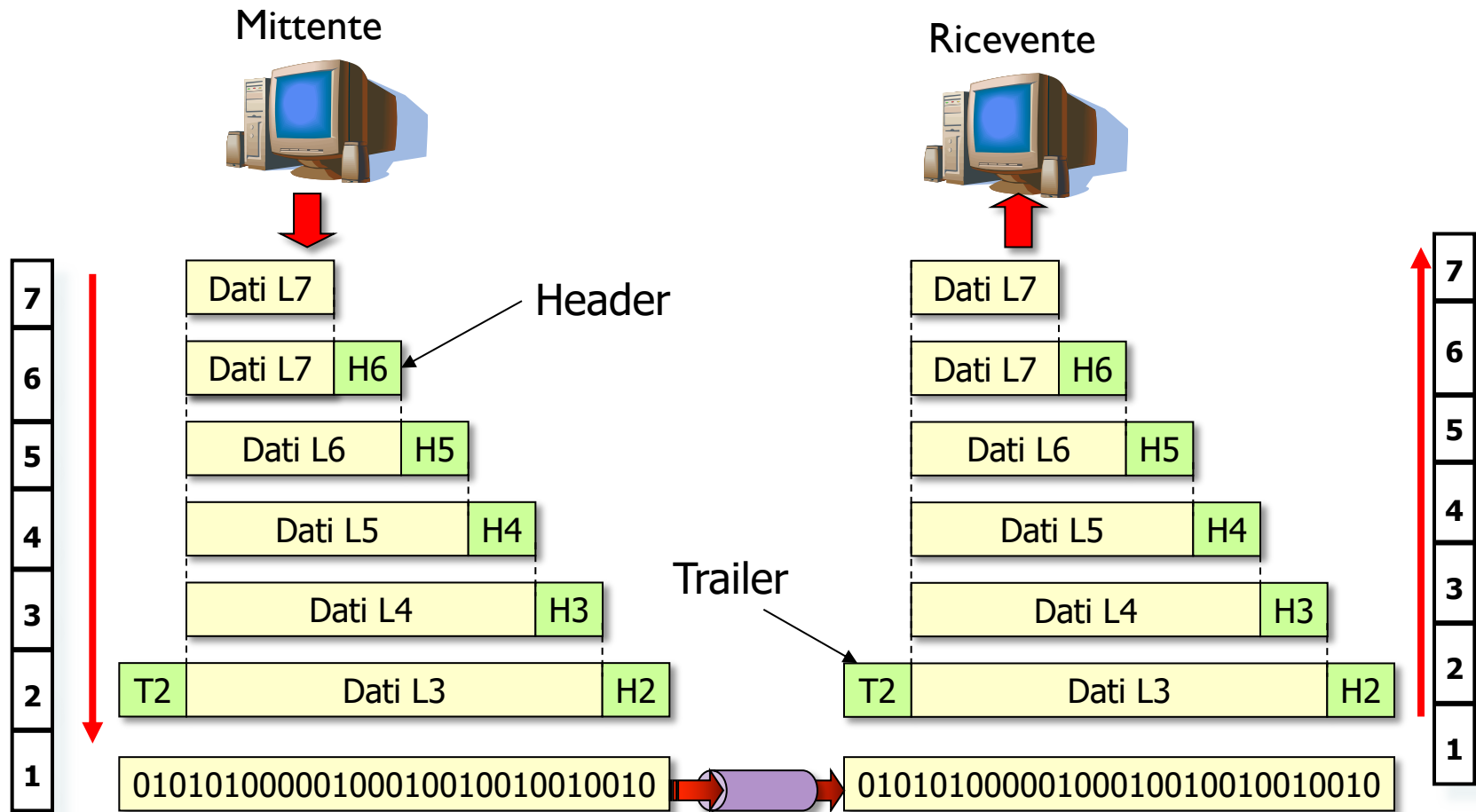
► OSI – Open System Interconnection

- Un **sistema aperto** è un insieme di protocolli per permettere la comunicazione fra due sistemi indipendentemente dalla loro architettura
- E' un modello per la progettazione di un'**architettura di rete** flessibile, robusta e inter-operativa
- Utilizza un riferimento a **strati** (layer) che descrivono la comunicazione a livelli di astrazione successivi (stack OSI)
 - Ogni strato considera uno specifico servizio necessario per gestire la comunicazione
 - A ciascun strato corrispondono funzioni diverse da quelle degli altri
 - Ciascun livello utilizza i servizi dello strato inferiore attraverso **interfacce** ai servizi
 - Gli strati elaborano in sequenza il messaggio da inviare secondo l'ordine predefinito

Layer del modello ISO-OSI

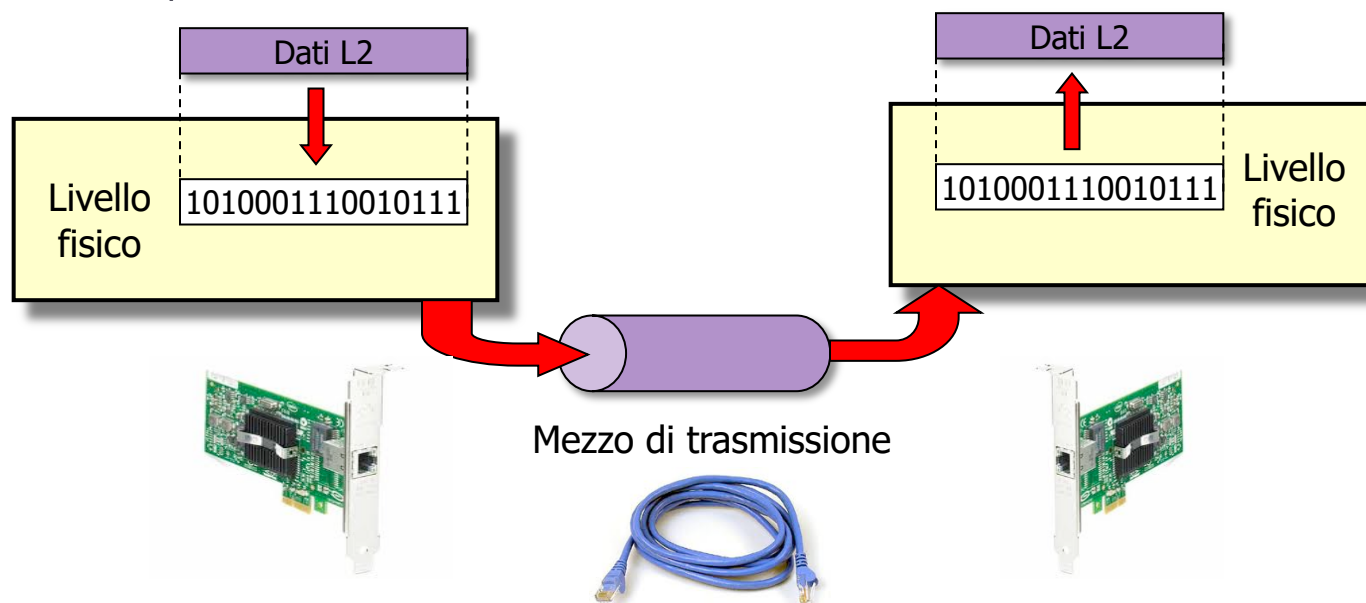


Costruzione dei messaggi nei Layer



Livello fisico

- ▶ Riguarda la trasmissione dei bit sul canale fisico
 - ▶ Caratteristiche fisiche (linee comunicazione, propagazione onde)
 - ▶ Codifica (rappresentazione dei bit, frequenza di trasmissione)
 - ▶ Modalità di trasmissione (simplex, half-duplex, full-duplex)
 - ▶ Caratteristiche Meccaniche ed elettriche (connettori, livelli di tensione)

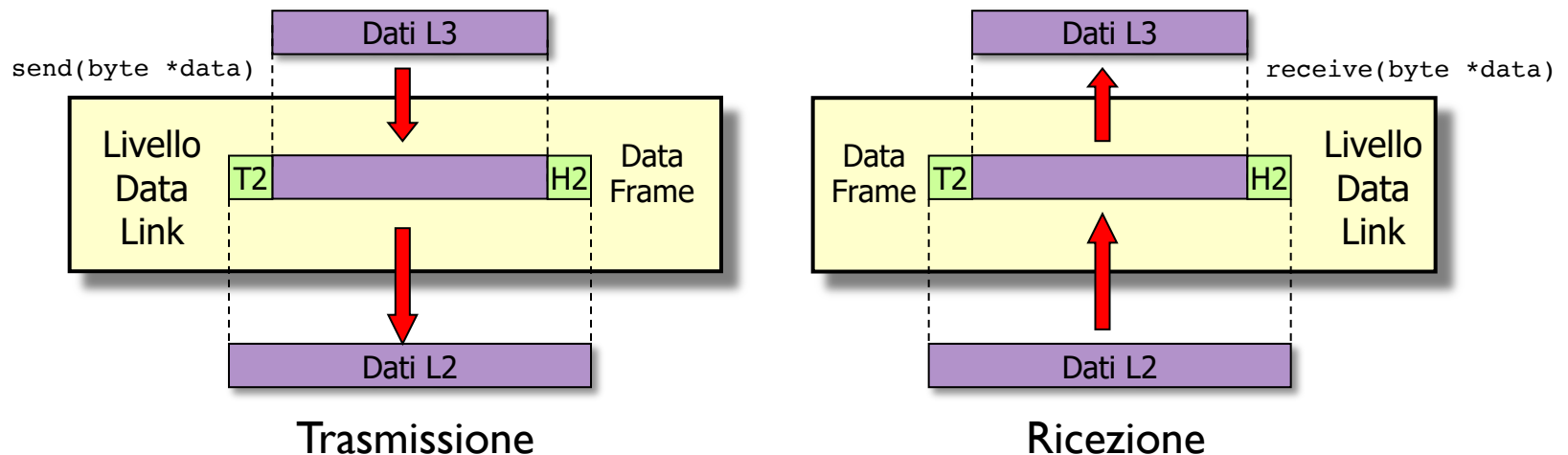


Data Link Layer 1

- ▶ Gestisce gli errori dello strato fisico aumentando l'affidabilità
 - ▶ codifica a rilevazione o correzione di errore
- ▶ Divide le informazioni in pacchetti trasmettendoli sul mezzo fisico (data frame)
 - ▶ Definizione di marker per l'inizio e la fine del pacchetto
- ▶ Sincronizza la trasmissione fra i dispositivi
 - ▶ temporizzazione, controllo di flusso
- ▶ Gestisce l'eventuale duplicazione dei data frame dovuta alle ritrasmissioni
- ▶ Gestisce l'accesso al canale per reti broadcast
 - ▶ MAC (Medium Access Control)
 - ▶ arbitraggio dell'accesso
 - ▶ indirizzamento fisico

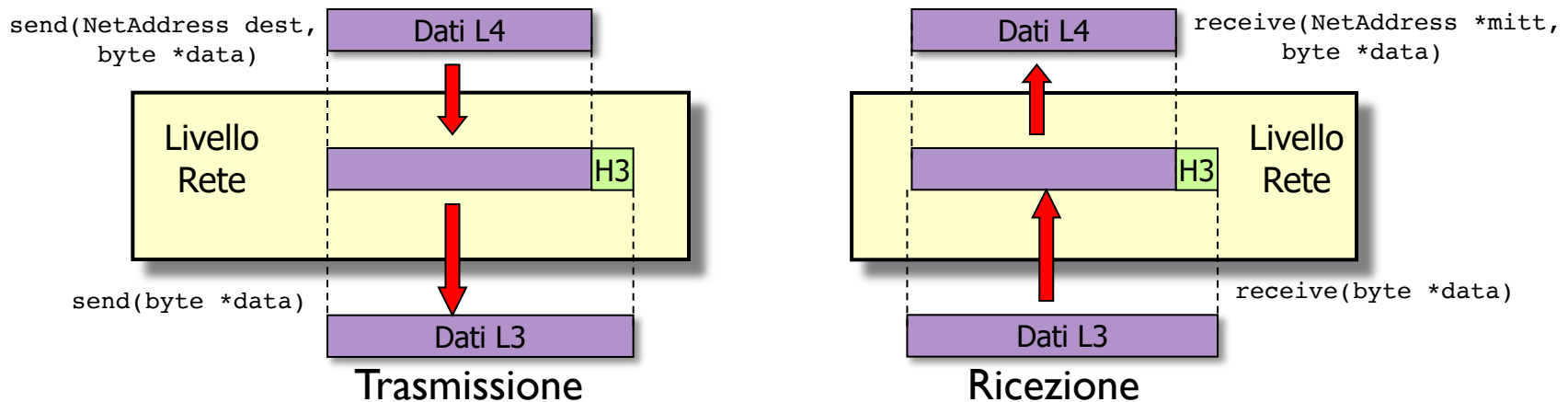
Data Link Layer 2

- ▶ E' spesso realizzato nell'hardware
- ▶ Le soluzioni adottate fanno riferimento a parametri noti e poco variabili (ritardo, tasso di errore, ..)
- ▶ Viene creato un pacchetto che prevede in genere
 - ▶ **Intestazione** (header) H2 (indirizzi MAC mittente/destinatario, lunghezza del pacchetto, marker di inizio pacchetto)
 - ▶ **Coda** (trailer) T2 (marker di fine pacchetto, bit per il controllo di errore)



Network Layer

- ▶ E' responsabile della consegna mittente-destinatario di un pacchetto attraverso reti diverse (**routing**)
- ▶ Realizza l'**indirizzamento** logico globale dei pacchetti
 - ▶ inserisce gli indirizzi logici del mittente e destinatario nell'header
 - ▶ un indirizzo individua un dispositivo sulla rete
- ▶ Definisce gli algoritmi di instradamento per i router
 - ▶ ottimizzazione dei percorsi (throughput, fault-tolerance, ...)

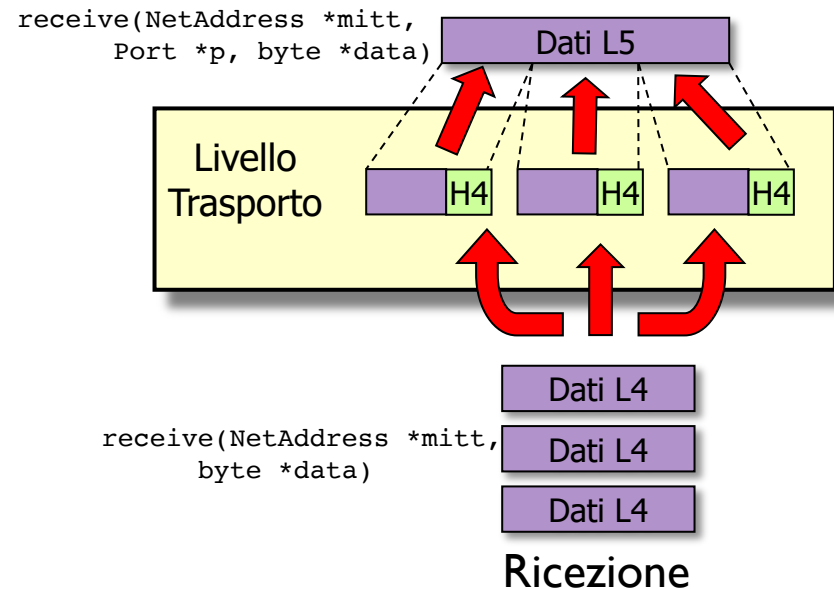
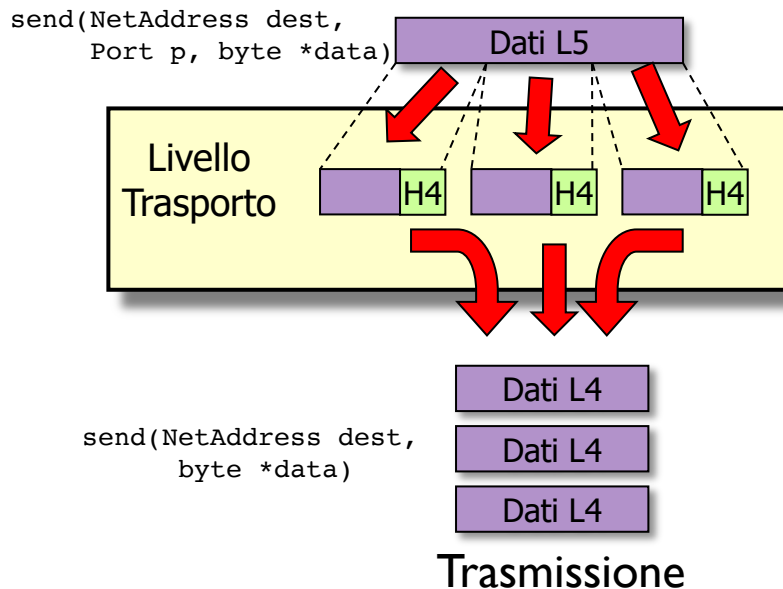


Transport Layer 1

- ▶ E' responsabile della consegna mittente-destinatario dell'intero messaggio (**punto-a-punto**)
- ▶ Realizza un indirizzamento service-point
 - ▶ gli indirizzi specificano non solo i dispositivi fisici (host) ma anche i processi mittente e destinazione (numero di **porta**)
 - ▶ definisce una comunicazione a livello di agenti software (programmi)
- ▶ Può supportare servizi di tipo diverso
 - ▶ **Con connessione**
 - ▶ Gestisce un flusso di dati affidabile
 - ▶ Gestisce la divisione in pacchetti, il riassemblaggio, l'ordinamento, la ritrasmissione, il controllo di errore
 - ▶ **Senza connessione**
 - ▶ Permette di inviare messaggi come singoli pacchetti indipendenti senza garanzia di consegna

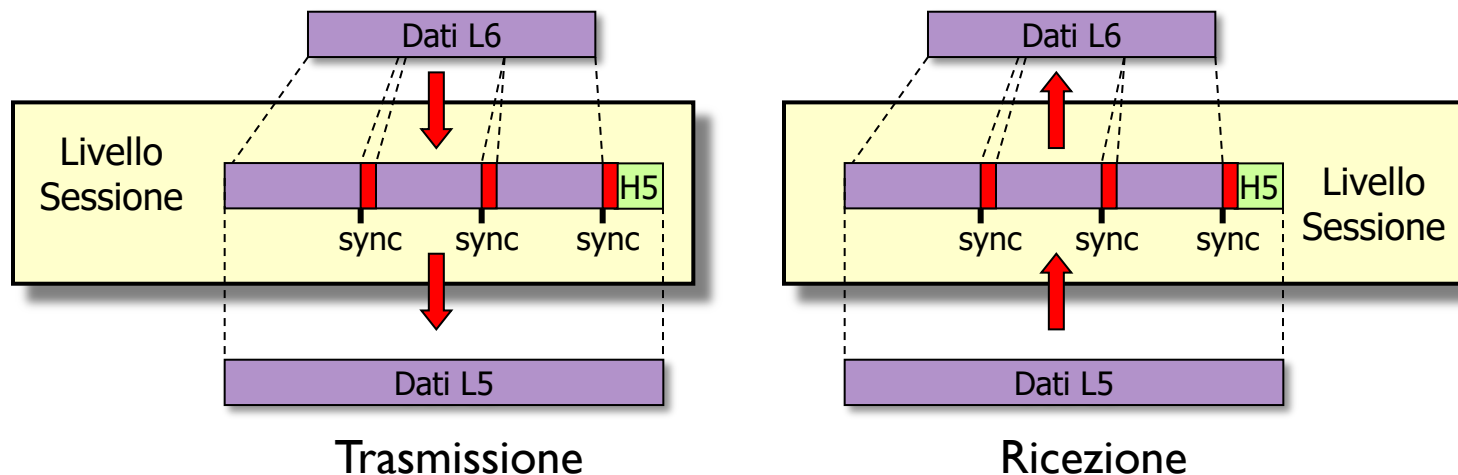
Transport Layer 2

- ▶ L'header specifica
 - ▶ la porta mittente e destinaria (gestite dal Sistema Operativo)
 - ▶ il numero d'ordine e la ricevuta di ritorno (ack) [con connessione]
- ▶ Il servizio senza connessione
 - ▶ ha un header più semplice
 - ▶ non richiede la fase di apertura della connessione per inizializzare la trasmissione del flusso dati



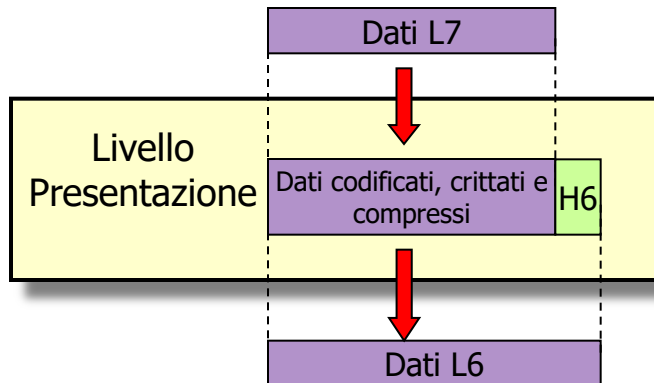
Session Layer

- ▶ Controlla il dialogo fra due sistemi assicurando le modalità half-duplex o full-duplex
- ▶ Permette la sincronizzazione con checkpoint di un flusso di dati
- ▶ Permette di correlare fra loro più comunicazioni successive all'interno di una sessione

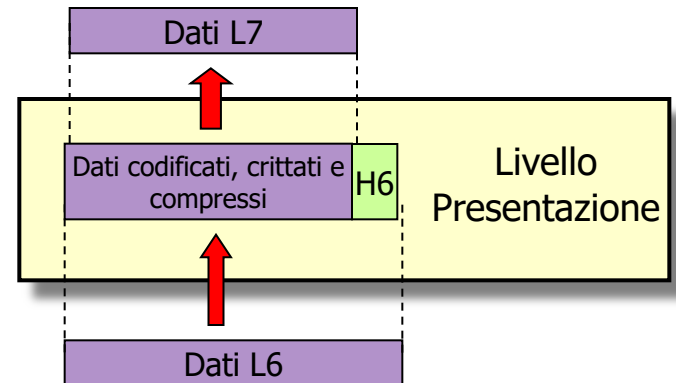


Presentation Layer

- ▶ Fornisce servizi riguardanti la codifica dei dati
 - ▶ Inter-operatività nello scambio di dati fra sistemi con codifiche diverse dei dati (**Traslazione**)
 - ▶ I dati sono tradotti in un formato comune e poi trasmessi
 - ▶ Sicurezza (**Crittografia**)
 - ▶ Codifica dati con **compressione**



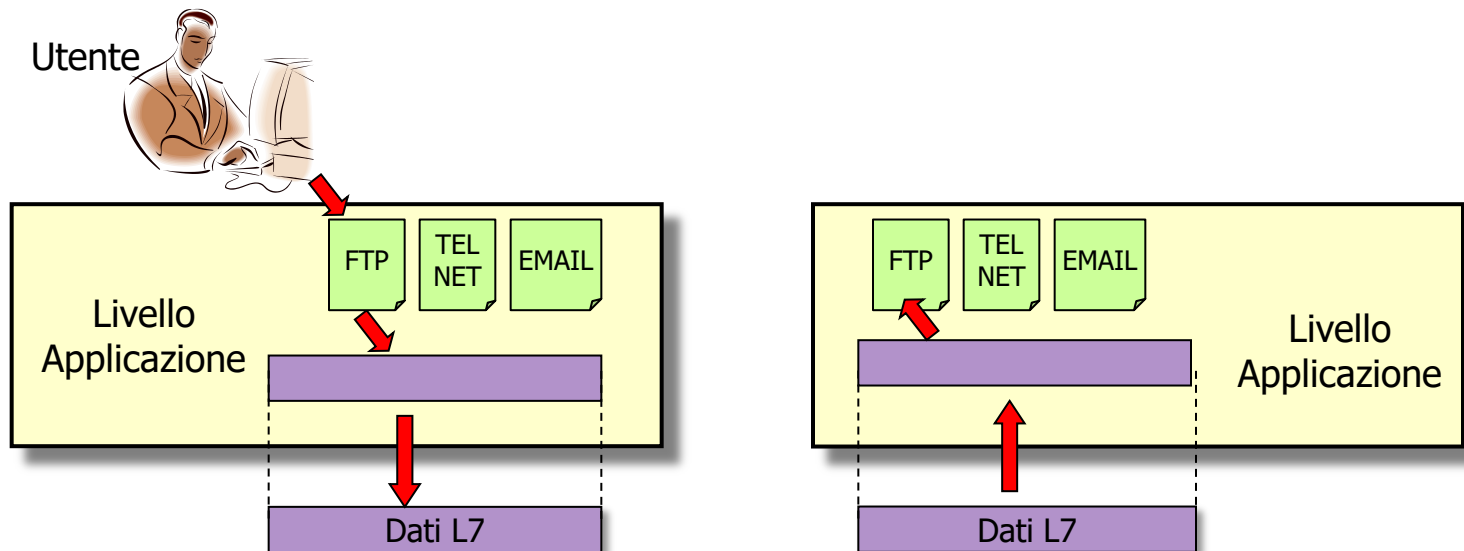
Trasmissione



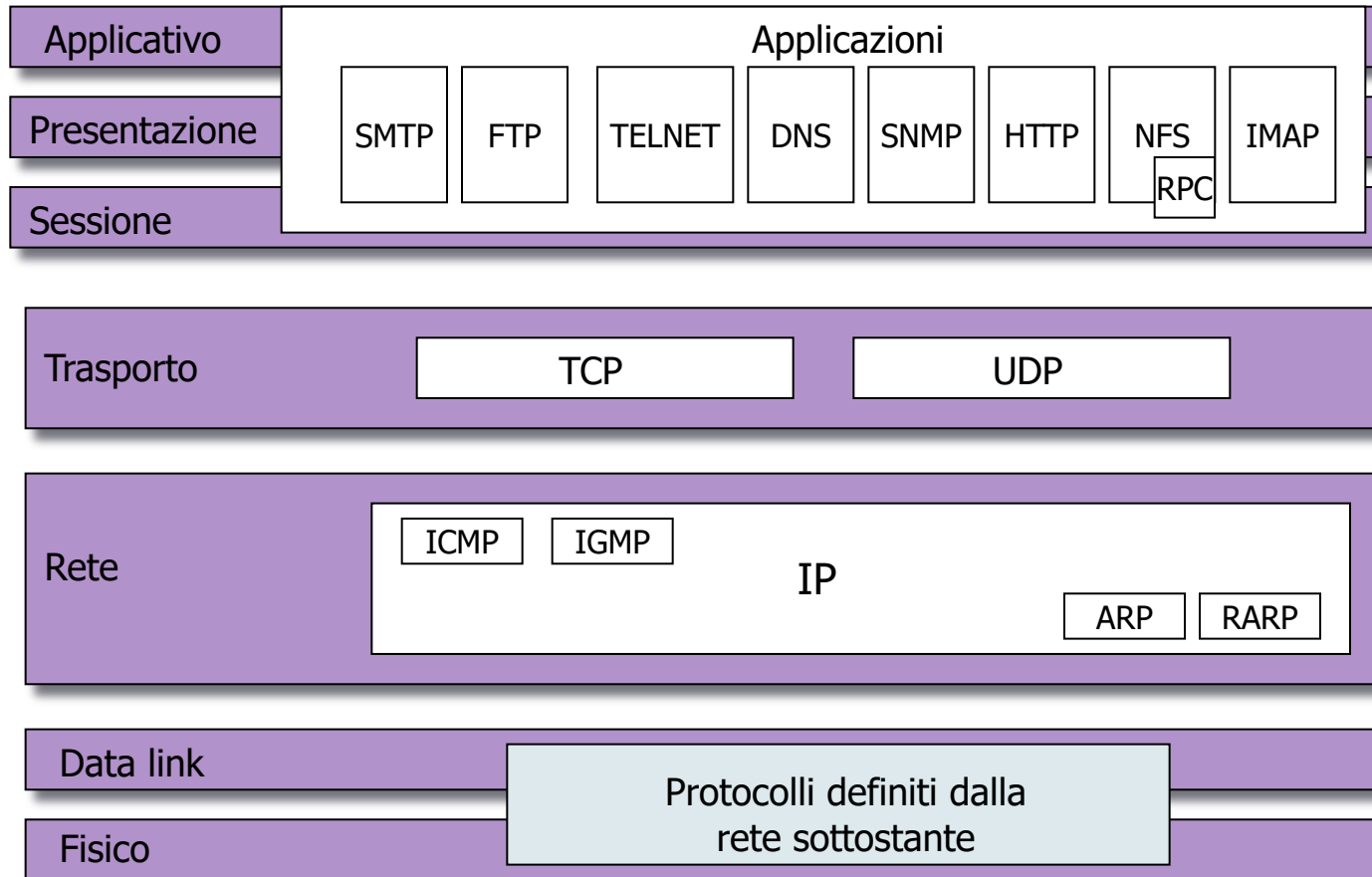
Ricezione

Application Layer

- ▶ Applicazioni che utilizzano i servizi di rete
 - ▶ e-mail, trasferimento di file remoti, accesso a database, Web
- ▶ I protocolli applicativi specificano la struttura dei messaggi scambiati fra gli agenti software per eseguire operazioni
 - ▶ Modello richiesta/risposta



I protocolli TCP/IP (1974) vs ISO-OSI



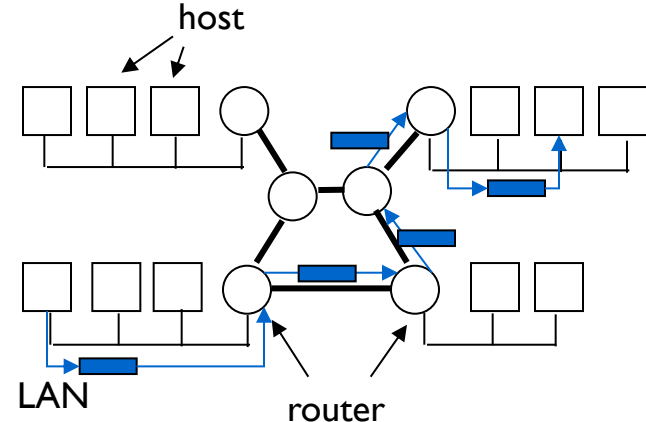
Internet Protocol (IP)

- ▶ I protocolli TCP/IP hanno una struttura gerarchica
 - ▶ Gli strati contengono protocolli relativamente indipendenti
- ▶ IP (Internetworking Protocol) – Network Layer
 - ▶ È un protocollo per la trasmissione di singoli pacchetti (**datagram**)
 - ▶ I pacchetti inviati sono indipendenti a livello IP
 - ▶ Il protocollo IP è non affidabile e senza connessione (consegna best-effort)
 - ▶ I datagram possono seguire percorsi diversi e raggiungere la destinazione in un ordine diverso a quello di spedizione
 - ▶ Il protocollo IP non si occupa di riordinare i pacchetti dopo il loro arrivo

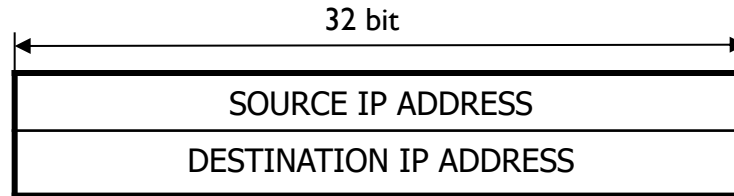
Il livello IP

- ▶ Trasmissione punto-a-punto di pacchetti (da dispositivo a dispositivo)
- ▶ Connessione di reti con tecnologie diverse (**internetworking**)
- ▶ Schema di **indirizzamento globale** (indirizzi IPv4 a 32 bit)
- ▶ L'instradamento viene fatto localmente ad ogni nodo in base all'indirizzo IP destinatario (tabelle di routing)

- I router instradano i pacchetti sulle linee di comunicazione in modo da consegnarli alla destinazione
- I percorsi sono scelti in modo ottimale (es. per evitare sovraccarichi)



Indirizzamento IPv4



- ▶ L'indirizzo IP a 32 bit identifica **univocamente** un dispositivo (host o router) sulla rete
- ▶ Gli indirizzi IP **pubblici** sulla rete Internet sono assegnati da **ICANN**
- ▶ Due dispositivi connessi alla rete Internet pubblica non possono avere lo stesso IP
- ▶ Un indirizzo IP identifica una connessione alle rete
 - ▶ Un host che ha più interfacce di rete ha più indirizzi IP
 - ▶ In alcuni SO si può assegnare più di un indirizzo IP alla stessa interfaccia (aliasing)

Struttura degli indirizzi IP

- ▶ Un indirizzo IP è composto di 4 byte
 - ▶ Si rappresenta con 4 numeri decimali (0-255) separati da punti (.)
- ▶ L'indirizzo può essere scomposto in due parti
 - ▶ Indirizzo di rete
 - ▶ Indirizzo di host nella rete
- ▶ La struttura a due livelli (rete/host) permette una gestione più efficiente dell'instradamento

Indirizzo IP binario (32 bit)				Indirizzo numerico
10000001	00110100	00000110	00000000	129.52.6.0
11000000	00000101	00110000	00000011	192.5.48.3
00001010	00000010	00000000	00100101	10.2.0.37
10000000	00001010	00000010	00000011	128.10.2.3
10000000	10000000	11111111	00000000	128.128.255.0

Classi di indirizzi IP

- ▶ La divisione logica dell'indirizzo in due parti definisce un'organizzazione gerarchica degli indirizzi
- ▶ Sono state definite suddivisioni standard per gestire gli indirizzi a livello globale che definiscono delle **classi di indirizzi**

	0 1 2 3 4	8	16	31	
classe A	0	rete	host		1.0.0.0 127.255.255.255
classe B	1 0	rete		host	128.0.0.0 191.255.255.255
classe C	1 1 0	rete		host	192.0.0.0 223.255.255.255
classe D	1 1 1 0	indirizzo multicast			224.0.0.0 239.255.255.255
classe E	1 1 1 1	riservato per scopi futuri			240.0.0.0 247.255.255.255

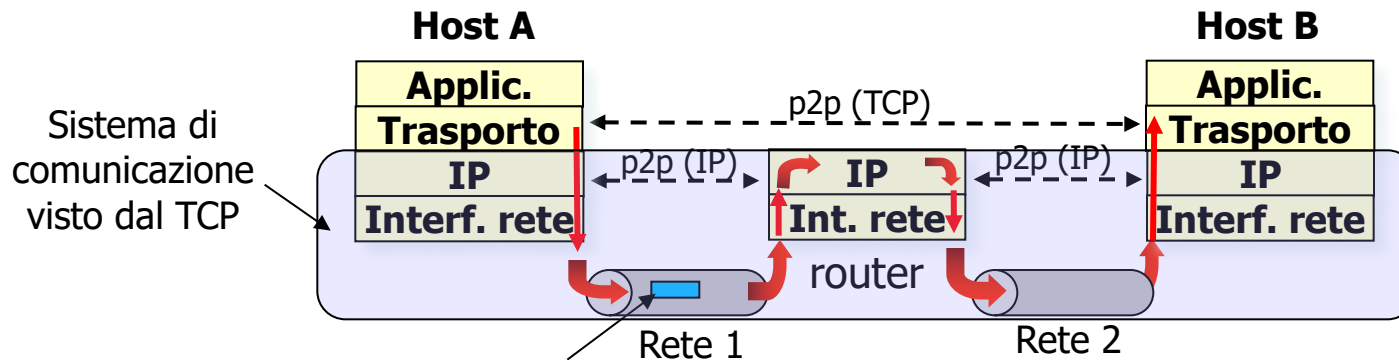
I protocolli di supporto al TCP/IP

- ▶ **ARP (Address Resolution Protocol)**
 - ▶ Associa l'indirizzo IP a quello fisico che caratterizza la scheda hardware
- ▶ **RARP (Reverse Address Resolution Protocol)**
 - ▶ Permette di ottenere l'indirizzo IP dall'indirizzo fisico
- ▶ **ICMP (Internet Control Message Protocol)**
 - ▶ Utilizzati per inviare messaggi sullo stato della trasmissione dei datagram
- ▶ **IGMP (Internet Group Message Protocol)**
 - ▶ Permette la trasmissione simultanea di un messaggio a più destinatari

Il livello di trasporto: TCP e UDP

- ▶ UDP e TCP sono protocolli che prevedono lo scambio dati fra processi indirizzati da
 - ▶ indirizzo IP a 32 bit e porta a 16 bit
- ▶ **UDP** (User Datagram Protocol)
 - ▶ E' senza connessione e prevede l'invio/ricezione di pacchetti datagram
 - ▶ Prevede il controllo di errore per scartare i pacchetti con dati corrotti
- ▶ **TCP** (Transmission Control Protocol)
 - ▶ E' un protocollo affidabile per il trasporto di un **flusso di dati**
 - ▶ Richiede l'apertura di una **connessione** fra i due nodi
 - ▶ Suddivide il flusso dei dati in segmenti che vengono numerati per essere riordinati dal ricevente

II TCP



- ▶ Il **TCP** su un host usa **IP** per comunicare con il **TCP** di un altro host
 - ▶ IP si occupa della consegna host-host dei pacchetti
 - ▶ TCP garantisce la comunicazione affidabile fra due processi che sono in esecuzione sui due host
 - ▶ E' un servizio di tipo **stream di dati** full-duplex

Funzionalità del TCP

▶ Trasmissione

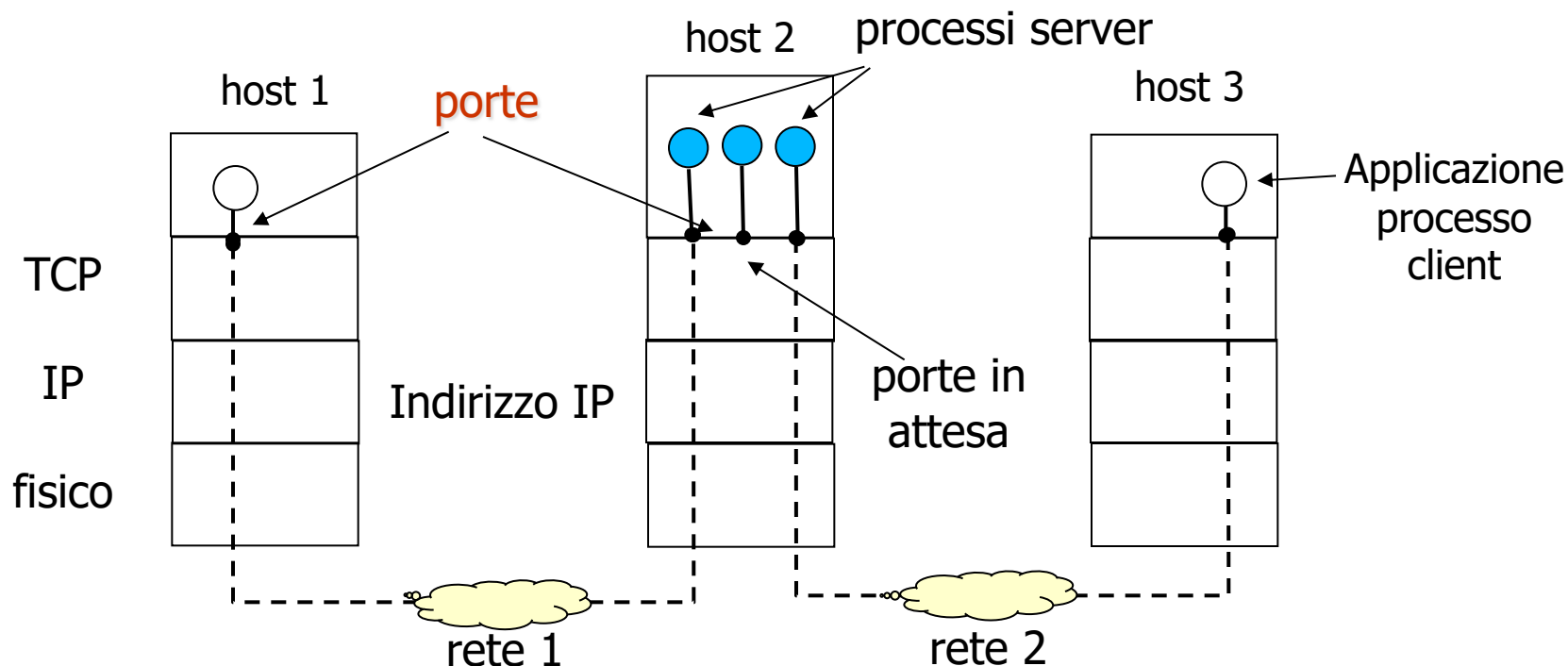
- ▶ Riceve un flusso di dati dall'applicazione
- ▶ Li organizza in pacchetti di massimo 64Kb identificati dal numero di sequenza del primo byte contenuto
- ▶ Eventualmente bufferizza i dati prima di spedire il pacchetto (es. input da tastiera)
- ▶ Spedisce le unità di dati come datagram IP
- ▶ Alla scadenza di un timeout ritrasmette i pacchetti per cui non ha ricevuto riscontro di ricezione

▶ Ricezione

- ▶ Riceve i datagram IP
- ▶ Ricostruisce il flusso di byte originale nella sequenza corretta in un buffer di ricezione usando i numeri di sequenza
- ▶ Invia un riscontro per i pacchetti di byte ricevuti correttamente
- ▶ Rende disponibile il flusso dati all'applicazione destinataria

Indirizzamento TCP (porte)

- Un **indirizzo TCP** identifica l'host (indirizzo IP a 32 bit) e la specifica applicazione sull'host (porta a 16 bit)



Socket

- ▶ Il concetto di **socket** è stato introdotto su UNIX BSD
- ▶ Indirizzo del socket
 - ▶ indirizzo IP dell'host
 - ▶ numero di porta a 16 bit (locale all'host)
- ▶ Una connessione TCP connette un socket dell'host client e un socket dell'host server
 - ▶ Una volta aperto un socket è utilizzato come un file in lettura e/o scrittura
 - ▶ Le connessioni sono identificate con gli identificatori dei socket dei due endpoint ($IP_c, Porta_c$)-($IP_s, Porta_s$)
 - ▶ Sono disponibili primitive nei linguaggi di programmazione per aprire e usare socket (C, Java...)

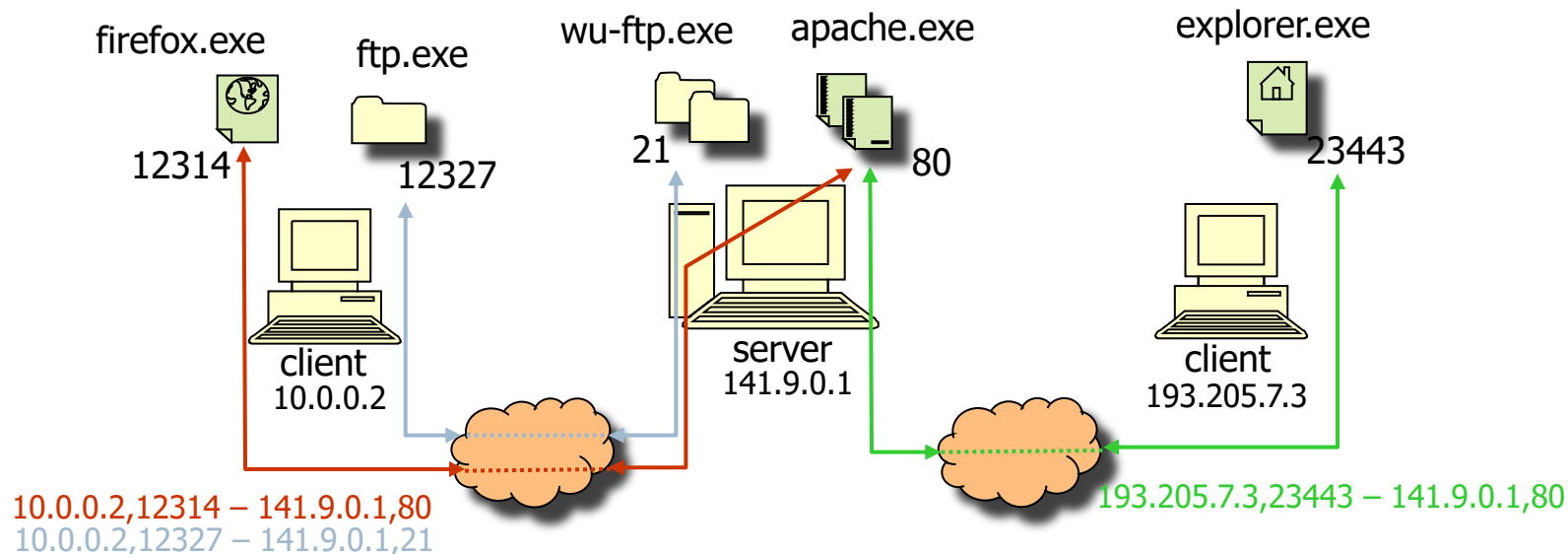
Porte dei server socket

- ▶ Le porte aperte in ascolto definiscono i servizi TCP disponibili su un host
- ▶ Per connettersi ad un servizio specifico su un server si devono conoscere
 - ▶ l'indirizzo IP dell'host server
 - ▶ il numero di porta su cui il processo server accetta le connessioni
- ▶ Le porte inferiori alla 1024 sono dette **porte ben note** (well-known ports) e corrispondono a servizi standard
 - ▶ la porta 21 di TCP corrisponde al servizio FTP (File Transfer Protocol)
 - ▶ la porta 80 di TCP corrisponde al servizio HTTP (Hypertext Transfer Protocol) ovvero al server Web
- ▶ Un servizio “standard” può anche essere attivato su una porta diversa (es. HTTP su 8080)

Porte dei client socket

- ▶ Il client definisce la porta di ogni sua connessione utilizzando numeri in genere elevati
 - ▶ si dicono **numeri di porta effimeri**
 - ▶ sono assegnati dal sistema operativo in modo da garantire che siano unici sull'host
 - ▶ non è necessario che sia noto a priori e per questo viene assegnato al momento della richiesta di connessione
- ▶ esempio di coppia relativa alla connessione ad un server HTTP
 - ▶ client port 23443
 - ▶ server port 80
- ▶ Le connessioni sono quindi **punto-a-punto** e **full duplex**

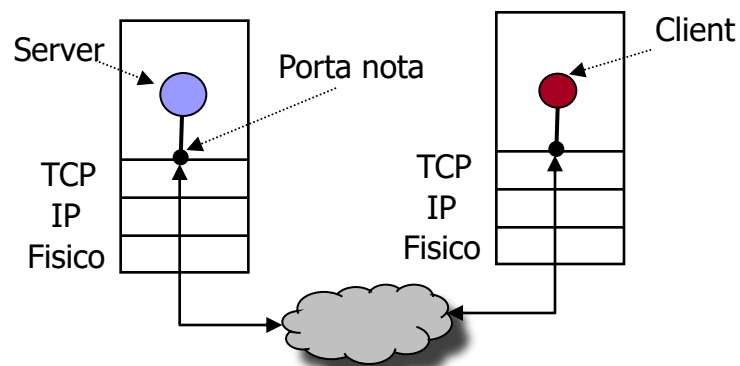
Socket connessioni



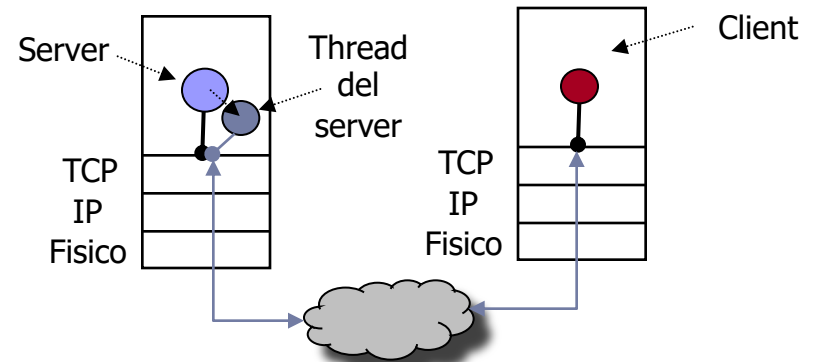
- ▶ Ogni socket mette in comunicazione due processi (server-client)
- ▶ Ogni connessione è individuata dalle due coppie (IP,porta) per i due host
 - ▶ E' possibile aprire più connessioni dalla stessa applicazione sullo stesso host allo stesso processo server (cambia la porta effimera del client)

Connessioni multiple 1

- ▶ Un server multi-thread può gestire più connessioni contemporaneamente
 - ▶ Un thread è un flusso di esecuzione



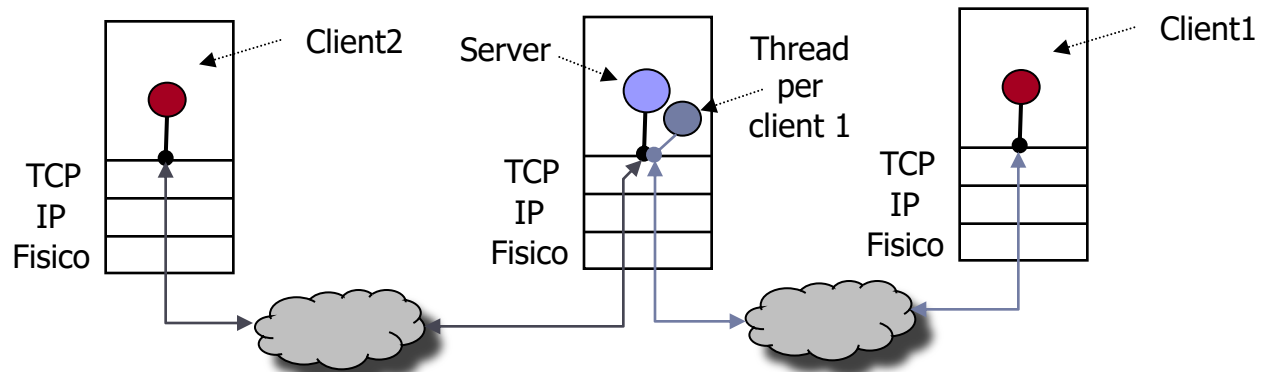
Connessione dal client al server sulla porta nota



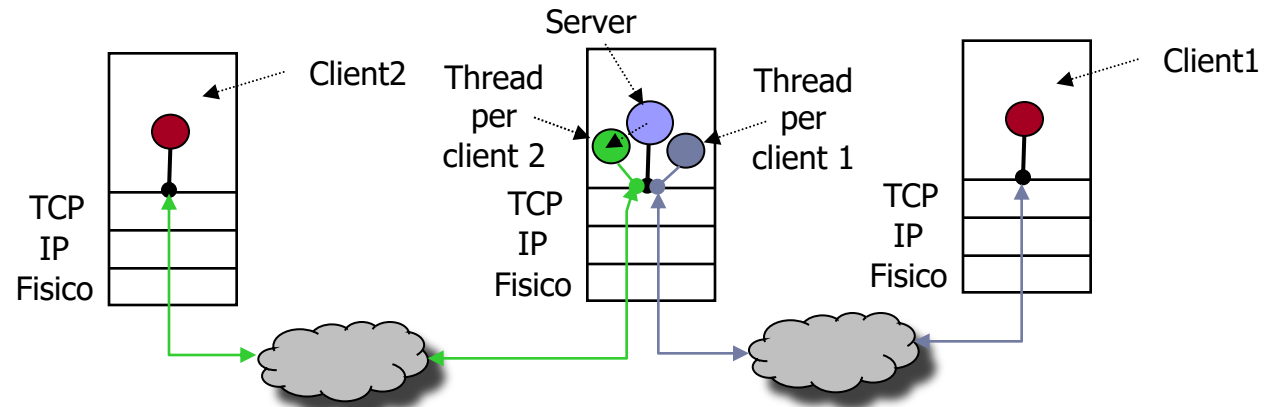
L'applicativo server genera un thread di esecuzione che gestisce la connessione. Il server è in grado di accettare una nuova connessione

Connessioni multiple 2

Un secondo client può fare una richiesta di connessione con lo stesso server

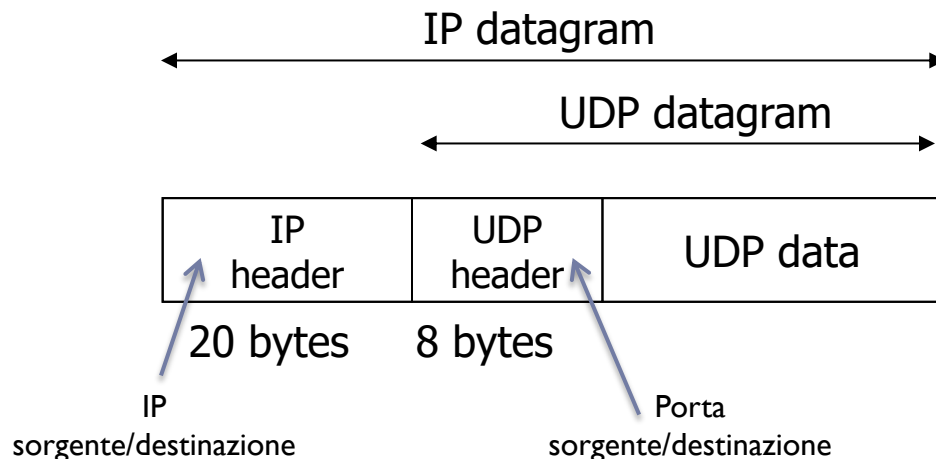


Viene generato un nuovo thread per gestire la connessione col client 2

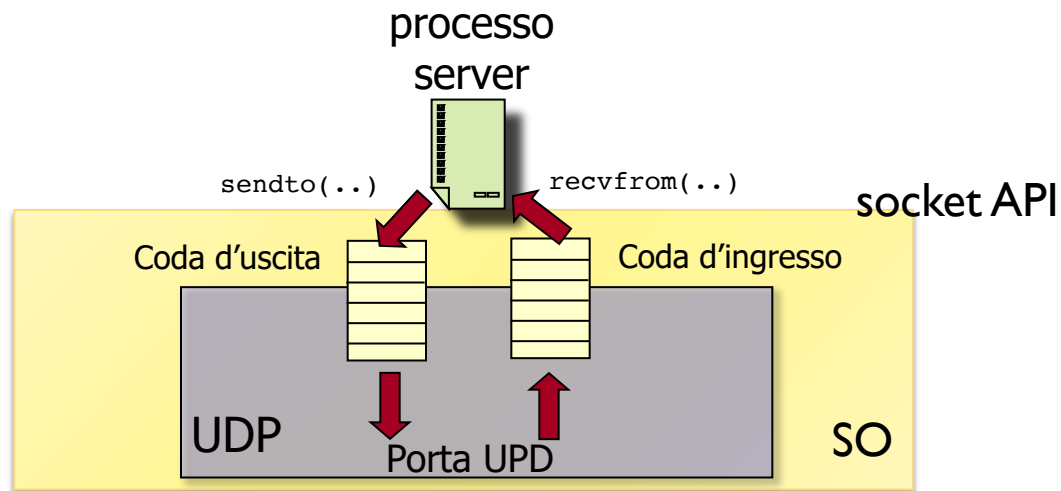


UDP

- ▶ Ogni operazione di output corrisponde all'invio di un pacchetto UDP che comporta l'invio di un datagram IP
 - ▶ UDP non garantisce affidabilità di consegna del pacchetto
 - ▶ Richiede meno overhead di una connessione TCP (header/connessione/ack)
 - ▶ Il destinatario del pacchetto è un processo server UDP in ascolto su una porta nota al mittente
 - ▶ Le porte UDP sono indipendenti da quelle TCP
 - ▶ La coppia IP-Porta identifica un socket UDP



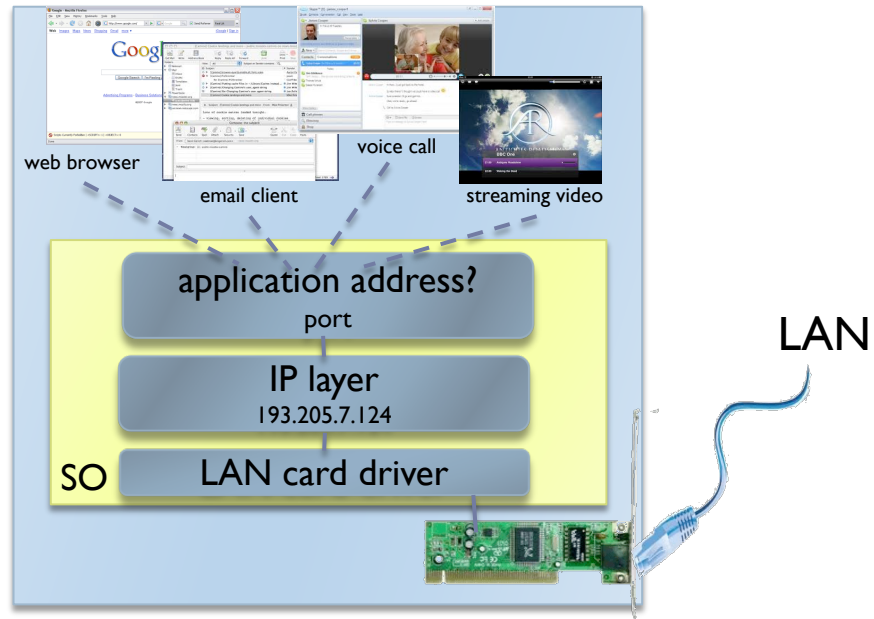
UDP server



- ▶ Il sistema operativo associa una coda d'ingresso (d'uscita) ad ogni socket UDP creato
 - ▶ Quando arriva un pacchetto il software di rete al livello UDP cerca la coda relativa alla porta destinazione
 - ▶ Se la coda non esiste (nessun processo è in ascolto), viene generato un messaggio ICMP Destination Unreachable (Port)

TCP/IP vs modello ISO-OSI

- ▶ TCP/IP è una **descrizione di protocolli già esistenti** ed è quindi meno generale e flessibile
- ▶ I protocolli TCP/IP erano già largamente usati in ambito accademico quando fu proposto il modello OSI
- ▶ Il modello e i protocolli OSI hanno dei difetti
 - ▶ Il livello sessione è poco utile e quello presentazione difficile da standardizzare
 - ▶ I livelli rete e datalink sono molto complessi e richiedono una strutturazione in sottolivelli
 - ▶ Lo standard è complesso e i protocolli difficili da implementare
- ▶ TCP/IP non distingue bene fra servizi, interfacce e protocolli



Reti di Calcolatori

Application Layer

Application Layer

- ▶ Programmi applicativi che utilizzano i servizi di rete
 - ▶ Sono spesso basati sul **modello client/server**
 - ▶ I protocolli applicativi permettono di standardizzare la comunicazione in modo da rendere interoperabili programmi sviluppati indipendentemente
 - ▶ specificano il **formato dei messaggi di richiesta e risposta** che sono scambiati fra gli agenti software coinvolti
 - ▶ definiscono il **meccanismo di trasporto** utilizzato (TCP o UDP)
- ▶ Il servizio di trasporto utilizzato dipende dal tipo di applicazione
 - ▶ **UDP** è adatto per le applicazioni che richiedono bassa latenza (ritardi) e overhead (dati in più da trasmettere) ma possono tollerare la perdita di informazioni (es. trasmissione video o audio)
 - ▶ **TCP** si usa quando è necessario garantire che tutti i dati trasmessi arrivino tollerando eventuali ritardi (es. trasferimento pagine Web)
- ▶ nessuno dei due protocolli fornisce garanzie di temporizzazione

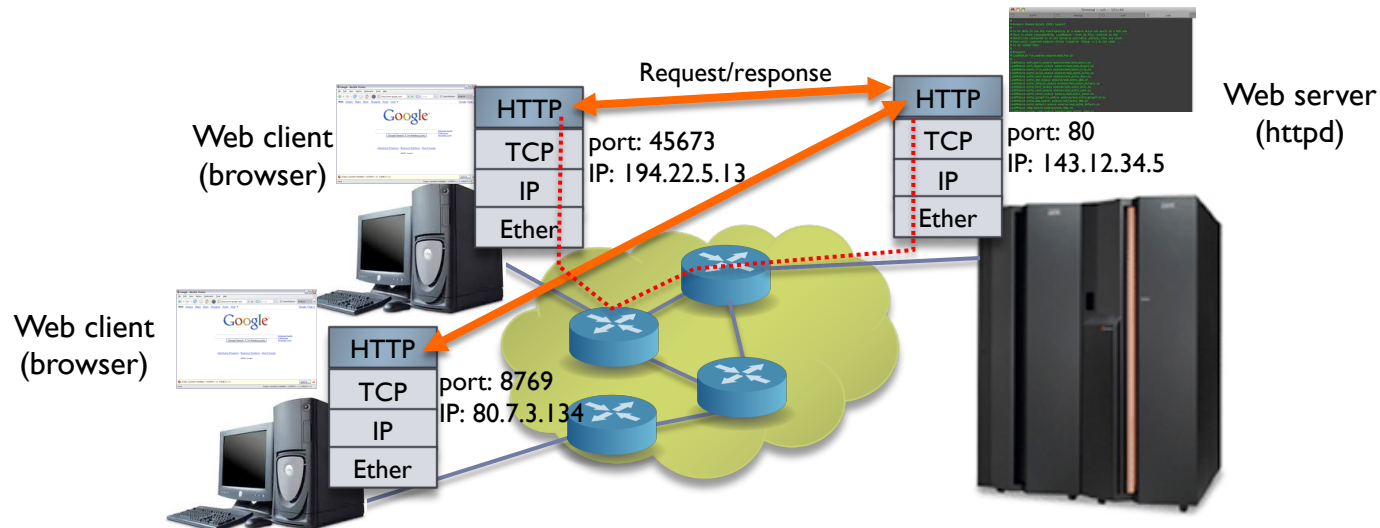
Applicazioni di rete

- ▶ Esistono numerose applicazioni e protocolli applicativi
 - ▶ DNS (Domain Name System) 
Risoluzione dei nomi degli host
 - ▶ SMTP (Simple Mail Transfer Protocol) e email 
sendmail o postfix
 - ▶ POP3 (Post Office Protocol) e IMAP (Interactive Mail Access Protocol)
Accesso remoto alle caselle di posta elettronica
 - ▶ FTP (File Transfer Protocol)
Trasferimento file remoti
 - ▶ SNMP (Simple Network Management Protocol)
Gestione di apparati di rete
 - ▶ HTTP (HyperText Transfer Protocol) e il WWW 
Server web e browsers
 - ▶ telnet, rlogin, rcp, ssh, sftp, nfs, lpd, bittorrent, applicazioni P2P (Peer to Peer), applicazioni per streaming video/audio, ...

Architetture delle applicazioni di rete 1

► Client-Server

- Il server è un applicativo sempre attivo in esecuzione su un host
 - L'indirizzo IP (nome DNS) dell'host server deve essere noto
 - Il server è in ascolto su una porta (TCP o UDP) nota (standard)
- I client sono applicativi in esecuzione su altri host della rete
 - I client sono saltuariamente attivi
 - Non è richiesto di conoscere a priori l'indirizzo IP e la porta usata dal client



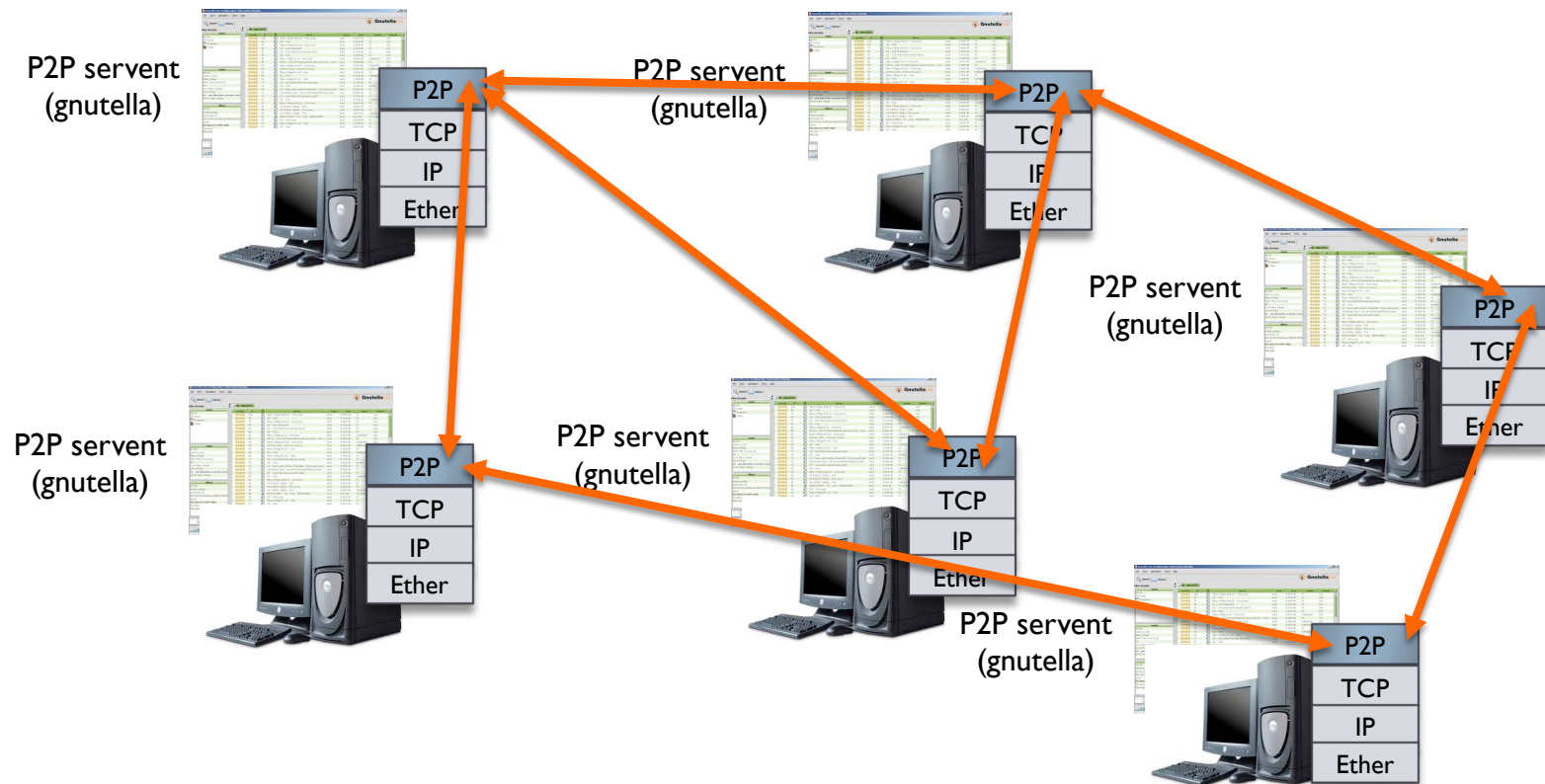
Architetture delle applicazioni di rete 2

► Peer to Peer (P2P)

- Tutti i nodi (**peers**) hanno in linea di principio la stessa importanza
- Ogni nodo è client e server (**servent**)
- Ogni peer condivide delle risorse (dati/potenza di calcolo)
- Possono comunque essere presenti server centralizzati o nodi con funzionalità diverse rispetto agli altri (supernodi)
- Il numero di nodi può essere dell'ordine delle centinaia di migliaia
- I nodi sono dinamici e autonomi
 - un nodo può entrare o uscire dalla rete in ogni momento
- gli indirizzi dei nodi non sono noti a priori e quindi devono essere definiti dei meccanismi di discovery
 - su una LAN si possono usare richieste broadcast (UDP)
 - su Internet occorre un server che indicizzi i peer (o parte di essi)

Rete P2P

- ▶ I peer sono connessi da una rete a livello applicativo
 - ▶ Un peer mantiene più connessioni TCP con altri peer
 - ▶ La rete di connessioni è usata per la gestione della rete (es. propagare le ricerche)

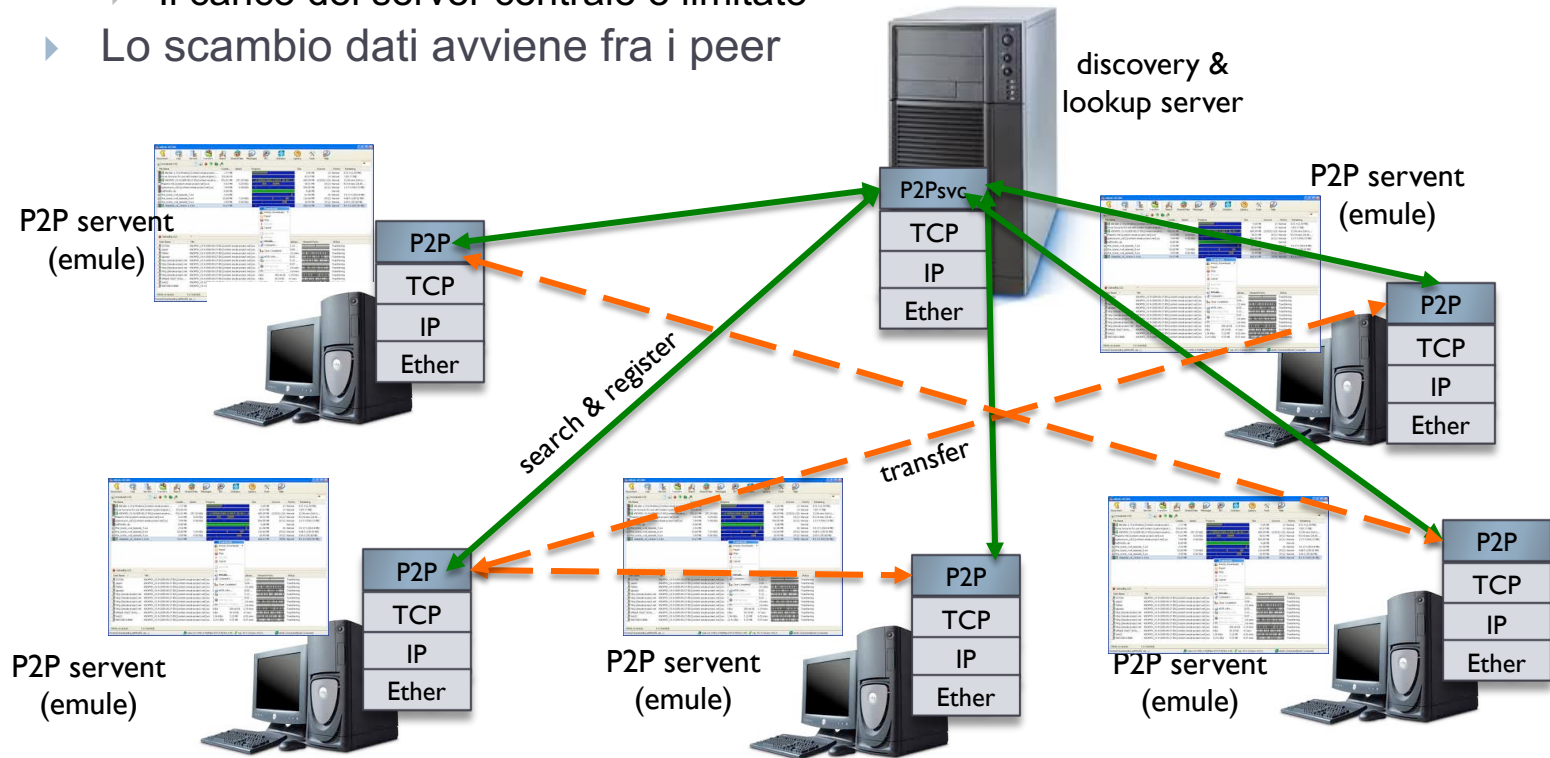


Reti P2P: Pro & Cons

- ▶ Il vantaggio delle reti P2P pure (senza server centralizzati) è la scabilità
 - ▶ tutti i peer si dividono il carico
 - ▶ ogni peer aumenta la domanda ma anche la capacità di servizio
- ▶ La gestione delle reti P2P pure è complessa
 - ▶ gestione delle ricerche in modo distribuito (mancanza di un indice centrale): *chi ha la risorsa che serve?*
 - ▶ La ricerca con flooding (propagazione a tutti i peer) crea un traffico elevato
 - ▶ gestione della rete in modo robusto: *come si trovano i peer a cui connettersi? Come si garantisce l'integrità della rete rispetto alla scomparsa di peer?*
- ▶ In genere le reti P2P pure non sono efficienti
 - ▶ es. Gnutella

Architetture ibride

- ▶ E' possibile combinare l'architettura P2P con quella client server (es. bittorrent, e-mule, skype)
 - ▶ Un server centrale fornisce alcuni servizi (indice per le ricerche, discovery dei peer)
 - ▶ Il carico del server centrale è limitato
 - ▶ Lo scambio dati avviene fra i peer



Domain Name System (DNS)

- ▶ E' un servizio che permette di convertire i nomi (DNS) degli host su Internet nei corrispondenti indirizzi IP
 - ▶ Il nome DNS (è una stringa di caratteri) più facile da ricordare (e che dà più informazioni) rispetto ad un numero a 32 bit



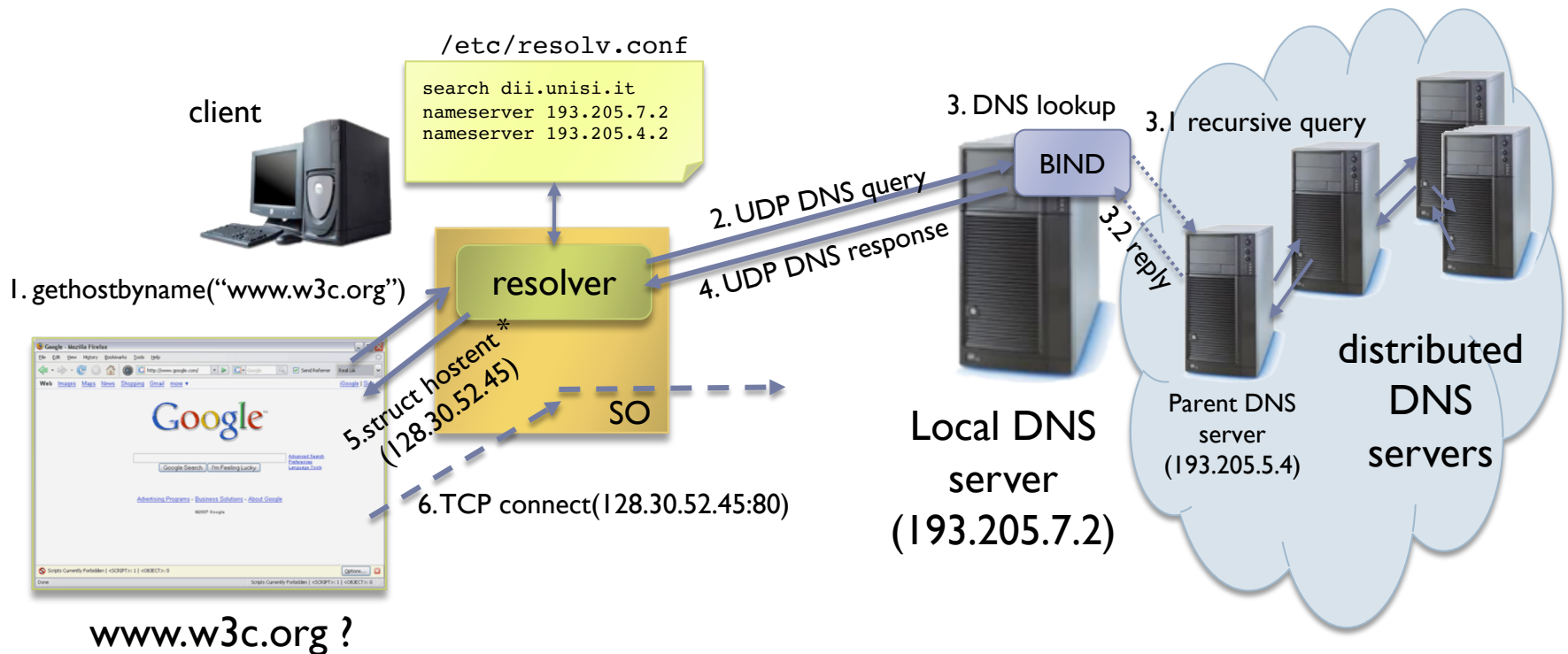
- ▶ Senza il DNS la rete Internet funzionerebbe ugualmente ma non sarebbe usabile facilmente dagli utenti
- ▶ Il DNS è un **servizio critico** per il funzionamento di Internet
- ▶ Per scambiare dati due host usano gli indirizzi IP: prima di poter contattare un host l'indirizzo DNS deve essere trasformato nel suo IP corrispondente (**DNS lookup**)

Struttura del DNS

- ▶ Il DNS è un'applicazione client/server
 - ▶ è un **database distribuito** gestito da un insieme di **server DNS** organizzati in **modo gerarchico**
 - ▶ definisce un **protocollo applicativo** con cui è possibile interrogare il database per mezzo di pacchetti **UDP** (porta 53)
 - ▶ Il client (**resolver**) è un servizio del Sistema Operativo
 - ▶ Occorre configurare il resolver specificando **l'indirizzo IP del server DNS** (Name server) del dominio locale (anche più di uno)
 - ▶ In UNIX è specificato nel file di sistema `/etc/resolv.conf`
 - ▶ Esiste una **API C** del resolver: `gethostbyname()`, `gethostbyaddr()`
 - ▶ Quando un'applicazione deve convertire un nome DNS in indirizzo IP, utilizza il resolver che con il protocollo DNS contatta il server DNS
 - ▶ **nslookup** è un programma client per interrogare il DNS

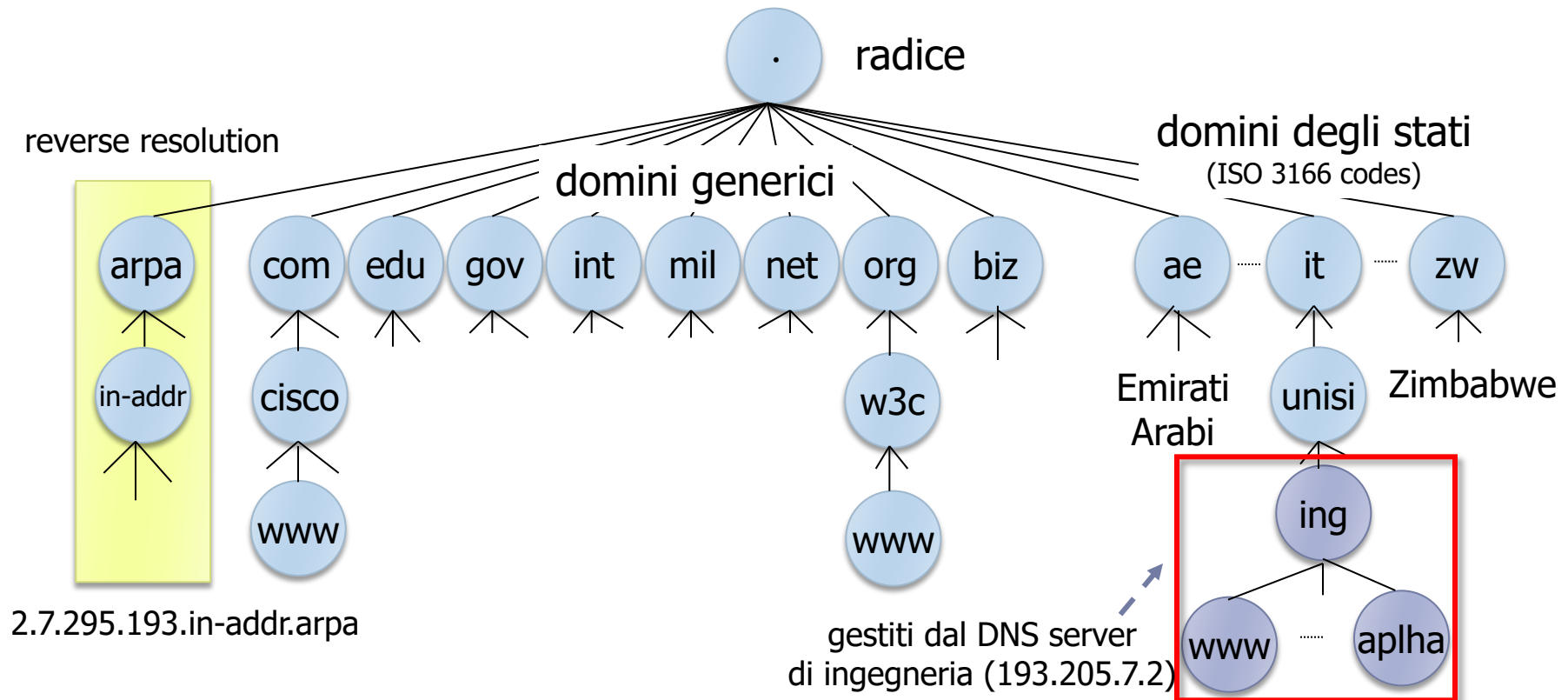
Risoluzione dei nomi DNS

- ▶ Il DNS è usato da altri protocolli applicativi quali ad esempio HTTP, SMTP
 - ▶ La richiesta DNS introduce di fatto un ritardo



Domain Name Space

- ▶ Lo **spazio dei nomi** è organizzato in modo gerarchico
 - ▶ è un albero di nomi di dominio (in parte definito da standard)
 - ▶ ciascun nodo o foglia ha associati zero o più **resource records**



I nomi di dominio

- ▶ Un nome di dominio è una stringa composta da una o più parti (**label**)
 - ▶ le label sono concatenate col carattere .
 - ▶ ciascuna label è lunga al massimo 63 caratteri e il nome completo non può superare i 255 caratteri (byte)
 - ▶ I caratteri ammessi sono un sottoinsieme della codifica ASCII (**LDH** – Letters Digit Hyphen ma non possono iniziare col carattere -)
 - ▶ I nomi DNS sono case-insensitive
- ▶ Un nome di dominio assoluto (**fully qualified domain name - FQDN**) termina con un punto (es. **www.sun.com.**)
 - ▶ I nomi che non terminano con un punto sono relativi (es. **alpha** è relativo al dominio in cui si trova l'host ovvero **ing.unisi.it.**)
- ▶ la label più a destra identifica il **top-level domain** (es. **.com**)
- ▶ un **hostname** è un nome DNS a cui è associato almeno un IP

Host & mail server aliasing

► Host aliasing

- Un host può avere uno a più sinonimi (alias)
- Il nome DNS principale è detto canonico

www.w3c.org **canonical name** = dolph.w3c.org.
Name: dolph.w3c.org
Address: 128.30.52.45

► Mail server aliasing

- Il DNS permette una risoluzione dei nomi specifica per i domini di posta elettronica (record di tipo **MX – Mail eXchange**)
- Il nome del dominio può essere associato al server di posta elettronica per rendere più leggibili gli indirizzi email

► set **type=MX**
► unisi.it
unisi.it **mail exchanger** = 10 antispam.unisi.it.
unisi.it **mail exchanger** = 10 antispam2.unisi.it.

Associazioni “uno-a-molti”

- ▶ Il DNS può essere utilizzato per distribuire il carico fra più server con IP diversi
 - ▶ Ad un hostname canonico vengono associati più indirizzi IP
 - ▶ Il server DNS risolve il nome fornendo l'insieme di indirizzi ma ruota l'ordinamento ad ogni risposta
 - ▶ In genere il resolver sceglie il primo indirizzo della lista per cui le richieste sono distribuite a rotazione fra i vari host

www.google.com canonical name = www.l.google.com.

Name: www.l.google.com

Address: 173.194.35.146

Name: www.l.google.com

Address: 173.194.35.147

Name: www.l.google.com

Address: 173.194.35.148

Name: www.l.google.com

Address: 173.194.35.144

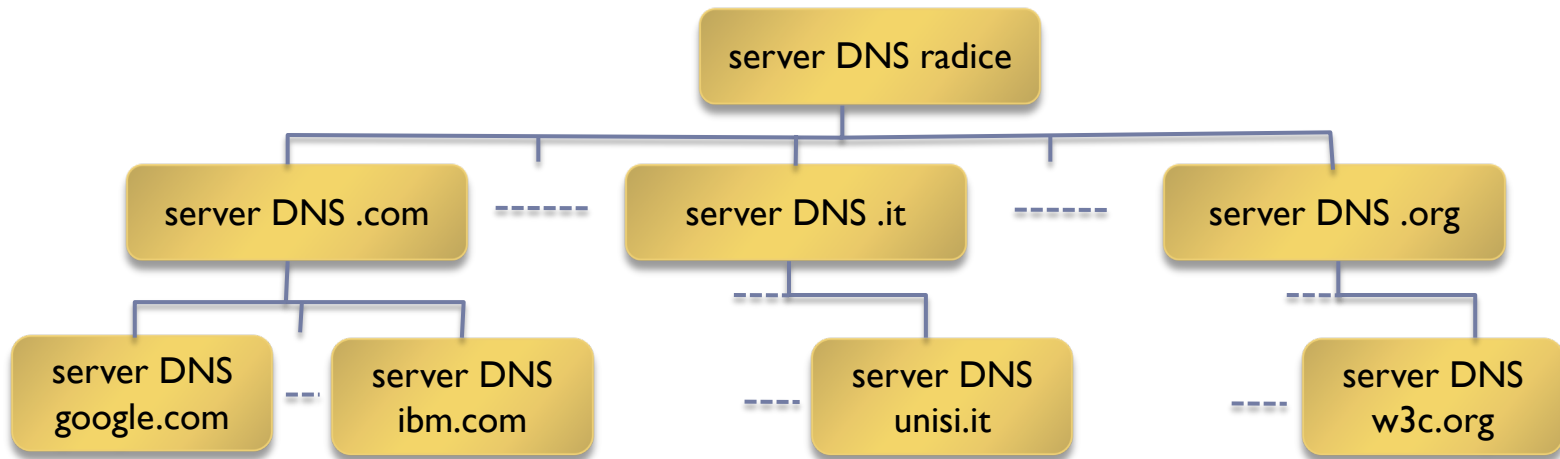
Name: www.l.google.com

Address: 173.194.35.145

Il DNS come database distribuito

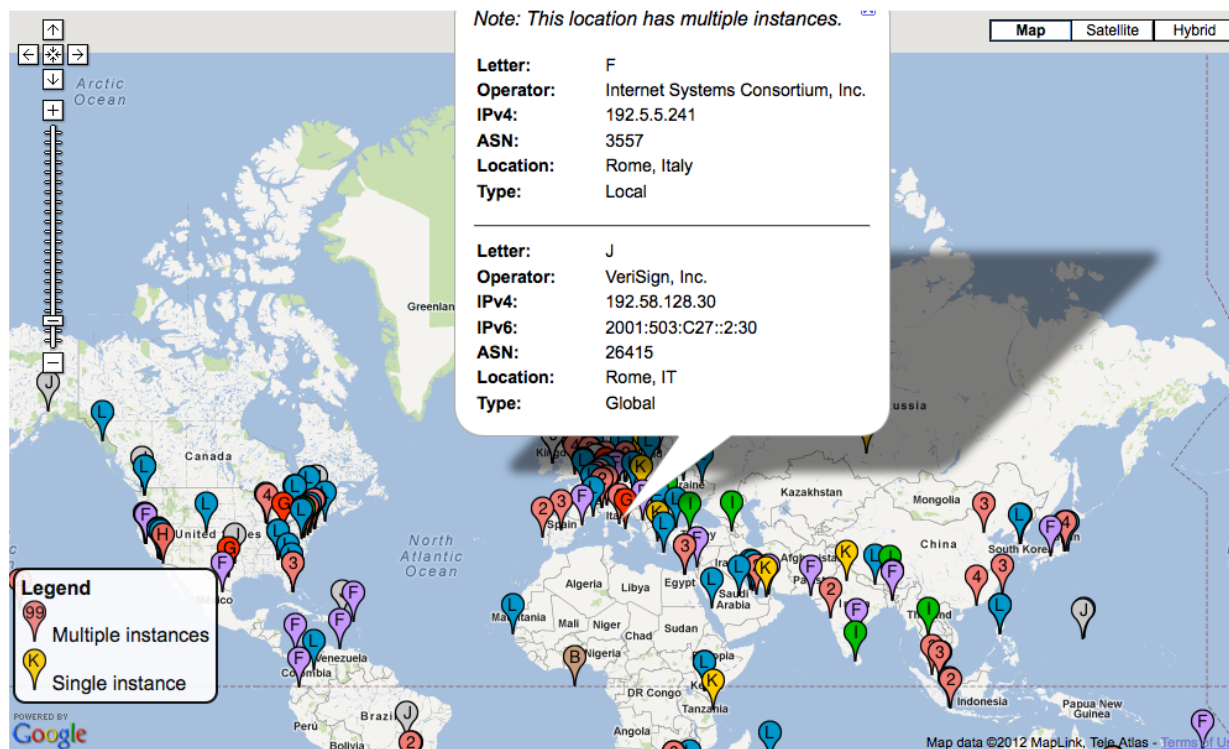
- ▶ Il servizio DNS è realizzato con un gran numero di server DNS distribuiti sulla rete
- ▶ Un approccio centralizzato per il DNS non funzionerebbe
 - ▶ Il server centrale sarebbe un **collo di bottiglia** sia per eventuali guasti che per la gestione del traffico richiesto (ogni client su rete genera numerose richieste DNS)
 - ▶ L'accesso al server centrale da punti diversi della rete potrebbe richiedere latenze anche lunghe
 - ▶ Tenere aggiornato il database sarebbe complesso perché il numero di record è molto elevato. Inoltre i record sono amministrati da numerosi soggetti eterogenei (aziende, organizzazioni, università,..)
- ▶ I server DNS sono organizzati in modo gerarchico e sono distribuiti nel mondo
 - ▶ ciascun server DNS ha competenza solo per un sottoinsieme di hostname

Struttura gerarchica dei DNS server



- ▶ **Server radice** (root servers)
 - ▶ Esistono 13 server radice etichettati da A a M
- ▶ **Server top-level domain** (TLD)
 - ▶ 20 domini generici e 248 livelli radici per le nazioni (dato 2009)
- ▶ **Server di competenza** (authoritative server)
 - ▶ server che gestisce effettivamente i record per una zona
 - ▶ I nomi di dominio devono essere registrati presso una **registration authority** fornendo i riferimenti ai name server di competenza

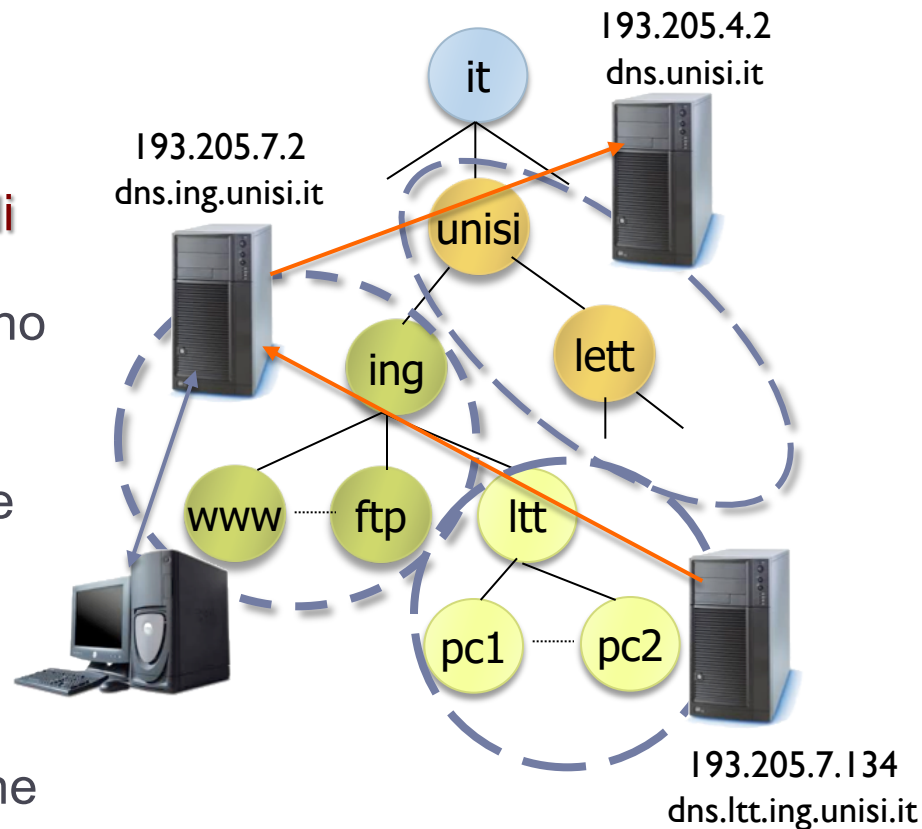
Root servers (www.root-servers.org)



- ▶ I 13 root server (A. root-servers.net – M.root-servers.net) hanno repliche anche in zone geografiche diverse
- ▶ Il **root zone file** contiene la lista dei nomi e indirizzi IP per i server di competenza dei top level domains (circa 200kb)
- ▶ Il root server fornisce il riferimento al server TLD competente per un dato dominio

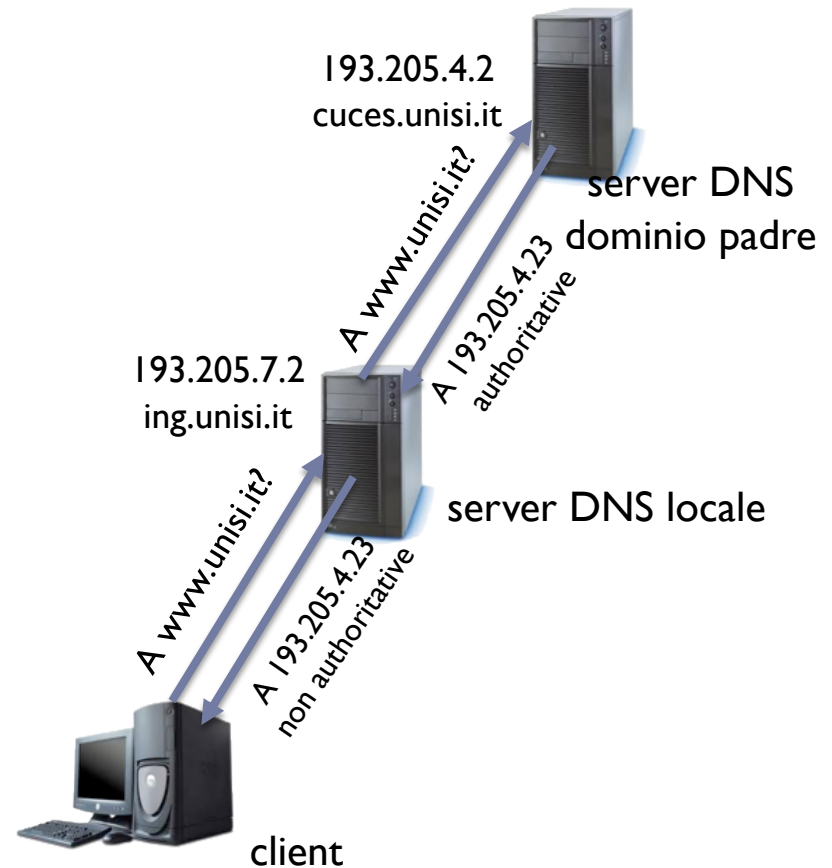
Zone & Authoritative Name Servers

- ▶ Lo spazio dei nomi di dominio è diviso in **zone**
 - ▶ Una zona corrisponde ad un sottoalbero della gerarchia dei nomi
 - ▶ Ogni zona ha un **server DNS di competenza primario** ed eventuali server secondari (sono repliche sincronizzate col primario)
 - ▶ Un authoritative server fornisce risposte che derivano da una sorgente diretta (es. amministratore di rete)
 - ▶ Un server DNS può essere **authoritative server** per più zone



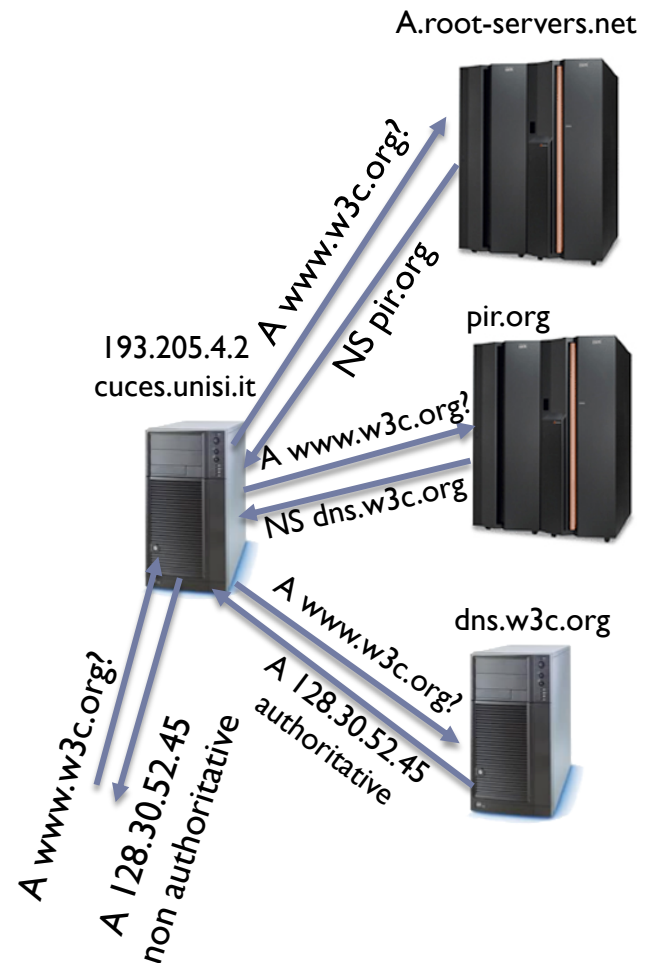
Interrogazione ricorsiva

- ▶ Il resolver chiede la risoluzione di un nome al server DNS locale con **modalità ricorsiva**
 - ▶ Se il nome è locale si ottiene un authoritative record
 - ▶ Se il nome appartiene ad una zona non gestita direttamente dal server locale si innesca un'interrogazione ricorsiva (o iterativa) ad altri nameserver
 - ▶ Il risultato è il RR richiesto o un errore
 - ▶ La modalità ricorsiva viene definita da un bit di flag nella richiesta



Interrogazione iterativa

- ▶ In un'**interrogazione iterativa** il client richiede al server la migliore risposta che può dare
 - ▶ Se il nome richiesto non è noto il server fornisce un riferimento al name server di competenza per un livello più basso dello spazio dei nomi
 - ▶ Al primo passo si contatta un root server che fornisce il riferimento al server TLD di competenza (es. .org)
 - ▶ Si contatta il server TLD che fornisce il riferimento al name server del dominio specifico (es. w3c.org)
 - ▶ Il name server invia la query al name server di dominio che o risponde con un RR con l'IP o fornisce un ulteriore riferimento ad un name server di sottodominio



Esempi con nslookup

```
> www.ing.unisi.it
Server: 193.205.7.2
Address: 193.205.7.2#53
```

```
www.ing.unisi.it canonical name = firewall.ing.unisi.it.
Name: firewall.ing.unisi.it
Address: 193.205.7.2
```

Authoritative
answer

```
> www.unisi.it
Server: 193.205.7.2
Address: 193.205.7.2#53
```

```
Non-authoritative answer:
www.unisi.it canonical name = pico.unisi.it.
Name: pico.unisi.it
Address: 193.205.4.23
```

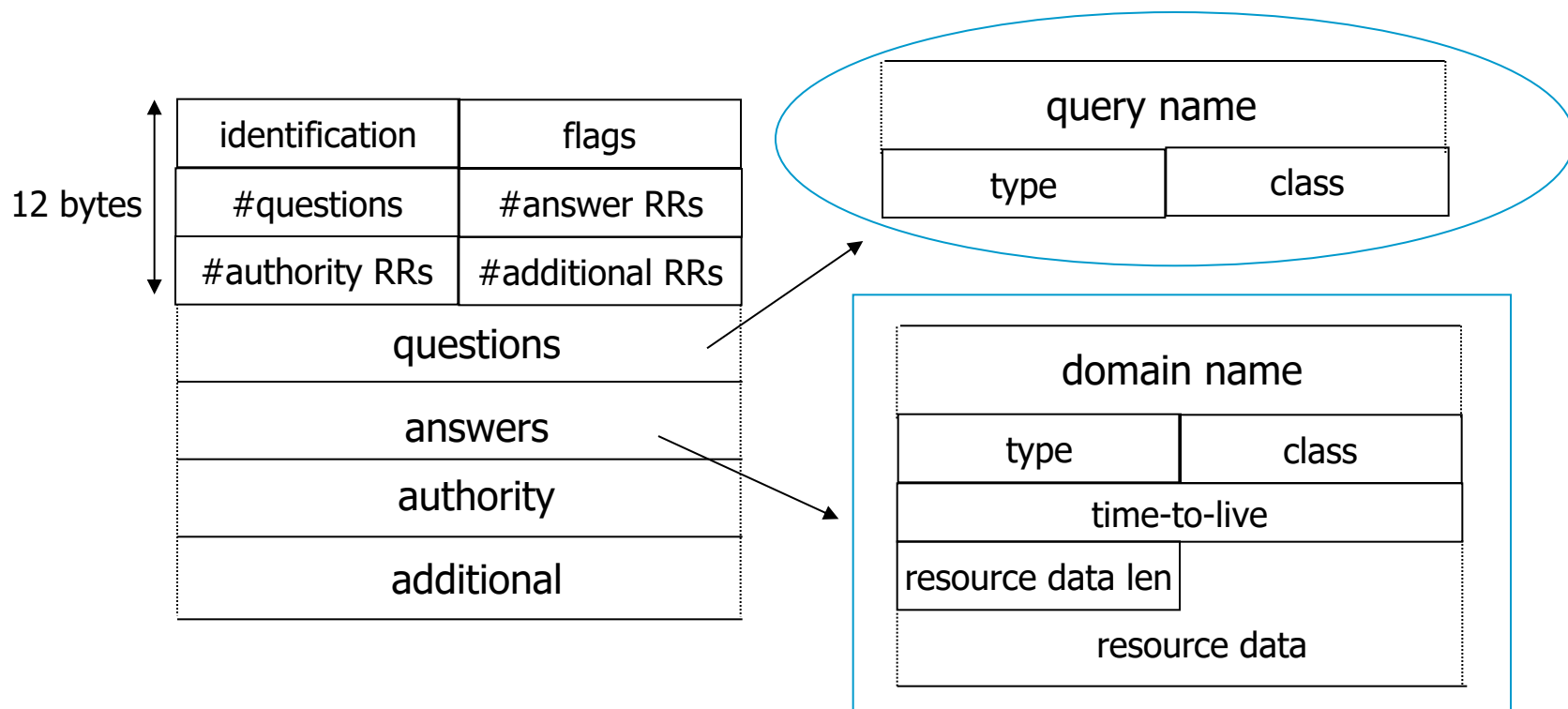
Non Authoritative
answer
ottenuta con
interrogazione ricorsiva

Cache DNS

- ▶ Il caching DNS permette di ridurre
 - ▶ la latenza nelle risposte
 - ▶ il numero di messaggi necessari ad ottenere la risposta
- ▶ I server DNS e il resolver memorizzano in una cache locale le risposte ricevute
 - ▶ Se una nuova richiesta è relativa un RR presente in cache la risposta può essere inoltrata senza ulteriori interrogazioni (**HIT**)
 - ▶ I RR nella cache vengono comunque invalidati dopo che è scaduto il loro **Time To Live** (TTL) per gestire il fatto che le associazioni IP-hostname possono essere dinamiche
 - ▶ I RR in cache comprendono anche gli indirizzi NS dei server TLD o di dominio in modo che le interrogazioni iterative non hanno necessità di contattare i relativi server DNS

Formato dei messaggi DNS

- ▶ Il messaggio di richiesta/risposta è inviato/ricevuto come pacchetto UDP
- ▶ Il messaggio può contenere una o più richieste o risposte sotto forma di **Resource Record** (RR)



Tipi di Resource Records (RRs)

▶ A

- ▶ *Nome*: hostname - *Valore*: Indirizzo IP

▶ NS

- ▶ *Nome*: dominio – *Valore*: hostname del server di competenza

▶ PTR

- ▶ *Nome*: indirizzo IP – *Valore*: hostname

▶ CNAME

- ▶ *Nome*: alias – *Valore*: nome canonico

▶ HINFO

- ▶ *Nome*: hostname – *Valore*: Informazioni sull'host (Tipo, Sistema Operativo, ...)

▶ MX

- ▶ *Nome*: hostname email – *Valore*: nome canonico del server email

DNS database file

```
@      IN      SOA      alpha.dii.unisi.it.
      postmaster.alpha.dii.unisi.it. (
                                2000072502; serial
                                86400 ; refresh
                                1800 ; retry
                                2592000 ; expire
                                4320000 ; default ttl )

@      IN      NS       alpha.dii.unisi.it.
@      IN      NS       cuces.unisi.it.

alpha      IN  A         193.205.7.2
           IN  HINFO     Pentium II Linux
mailsrv    IN  CNAME     alpha
www        IN  CNAME     alpha
ing.unisi.it IN  MX       0 alpha
```

Il servizio di posta elettronica @

- ▶ L'email rappresenta un sistema di comunicazione asincrona
 - ▶ L'invio da parte dell'utente mittente e la ricezione da parte dell'utente destinatario dei messaggi non sono sincronizzati
 - ▶ Il sistema di gestione dei messaggi deve prevedere un meccanismo di memorizzazione dei messaggi inviati ma ancora non letti
- ▶ L'infrastruttura del servizio email prevede
 - ▶ La definizione di un sistema di indirizzamento per le “caselle postali” (**mailbox**) a cui sono inviati i messaggi
 - ▶ Un sistema di agenti software per inviare, ricevere, e consegnare i messaggi (agenti utente e agenti di trasferimento) con il relativo protocollo di comunicazione (**Simple Mail Transfer Protocol SMTP**)
 - ▶ Uno standard per il formato dei messaggi (busta)

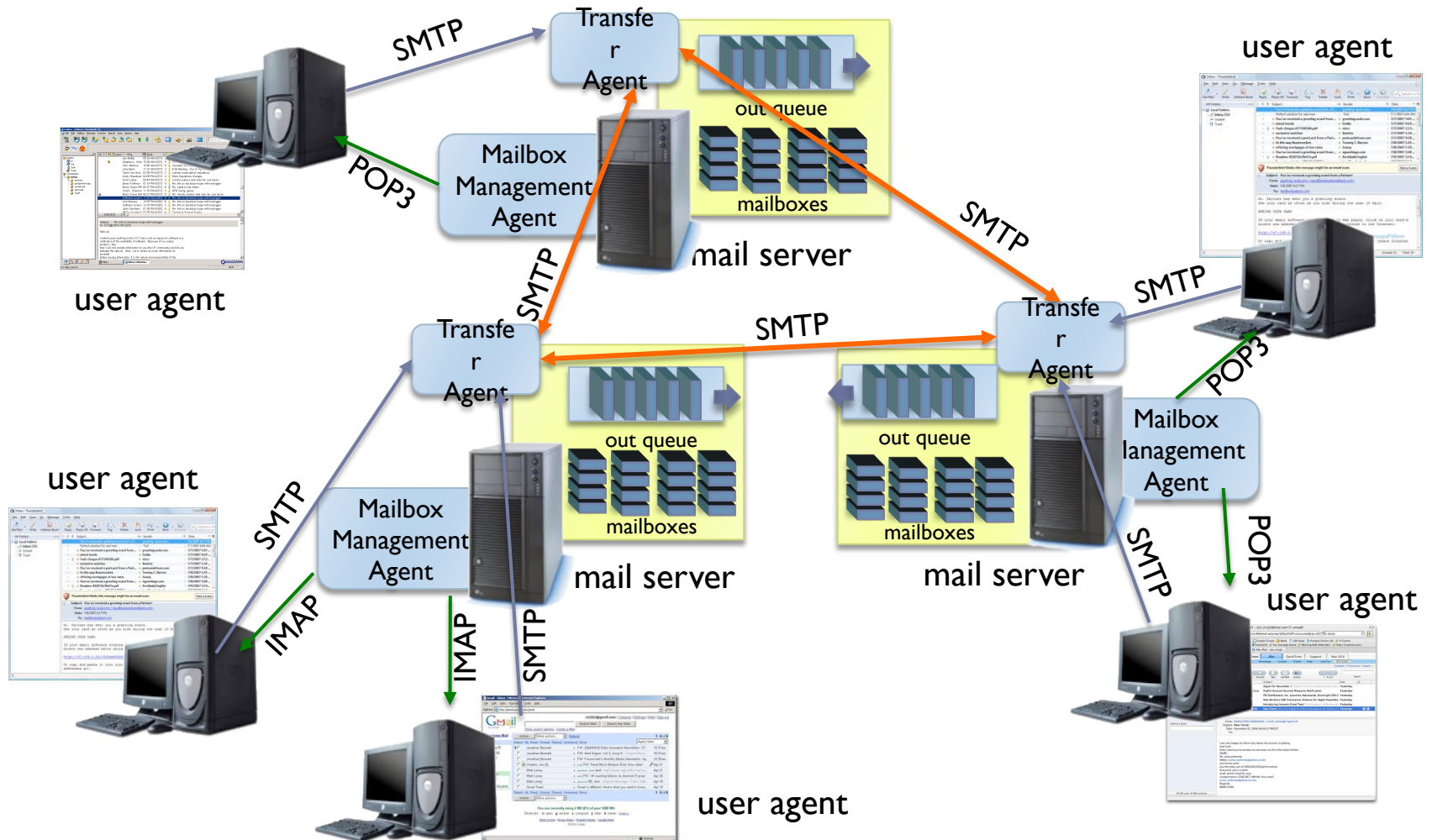
Indirizzi email @

- ▶ Un indirizzo email individua una **mailbox** su un host nella rete Internet
 - ▶ La mailbox è un file di testo in una directory specifica nel server (es. in Unix è normalmente `/var/spool/mail/utente`)
 - ▶ I messaggi di posta in arrivo sono accodati (append) al file di mailbox
 - ▶ L'utente può accedere alla posta localmente leggendo il file mailbox o attraverso protocolli per l'accesso remoto alla mailbox (POP3 o IMAP)
 - ▶ Un indirizzo email ha la struttura

utente@host.dominio

- ▶ **utente** è l'identificativo della mailbox (si possono avere alias)
- ▶ **host.dominio** è risolto dal DNS con una query di tipo MX

Infrastruttura @ email



Agenti @ email

▶ Mail User Agent (MUA)

- ▶ Programmi per leggere, comporre i messaggi e gestire le mailbox
 - ▶ eudora, thunderbird, outlook, pine, mac mail, web mail,...
- ▶ Utilizzano un server SMTP per la spedizione dei messaggi (eventualmente con autenticazione)
- ▶ Gestiscono di mailbox remote interagendo con il server usando il protocollo POP3 o IMAP

▶ Message Transfer Agent (MTA)

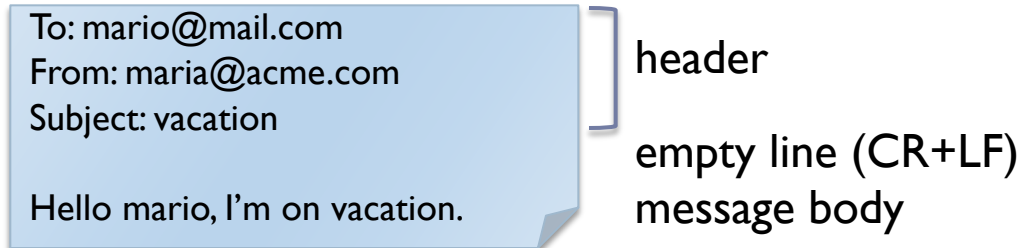
- ▶ Gestiscono il trasferimento dei messaggi dal mittente alla destinazione e la ricezione dei messaggi sui server di posta
 - ▶ Inviare un messaggio corrisponde a trasferire un file di testo
- ▶ Sono servizi di sistema che usano il protocollo SMTP su una connessione TCP (porta server 25) per trasferire i messaggi
 - ▶ sendmail, postfix, ...

Consegna di un messaggio

- ▶ Generalmente la consegna di un messaggio è diretta
 - ▶ L'user agent trasferisce il messaggio sul suo MTA (SMTP server)
 - ▶ Il MTA eventualmente accoda il messaggio nella coda di uscita
 - ▶ Il MTA mittente estrae l'hostname dall'indirizzo email del destinatario e apre una connessione TCP sulla porta 25 di tale host contattando il MTA destinatario
 - ▶ Utilizzando il protocollo SMTP il messaggio è trasferito fra i due MTA
 - ▶ L'MTA destinatario accoda il messaggio nella mailbox dell'utente specificato
- ▶ Gli MTA possono anche funzionare da **Relay Agent**
 - ▶ Il messaggio può transitare da MTA intermedi che accettano l'inoltro di messaggi verso altri MTA
 - ▶ In genere la funzionalità Relay è disabilitata perché può essere usata per inviare SPAM

Formato dei messaggi @ email

- ▶ Il formato dei messaggi è definito in RFC 821 - RFC 822
 - ▶ Un messaggio è un testo con codifica dei caratteri ASCII a 7 bit, ovvero utilizza solo i codici da 0 a 127
 - ▶ Il messaggio è strutturato in due parti separate da una linea vuota



- ▶ L'intestazione contiene meta-informazioni che descrivono proprietà del messaggio
 - ▶ è un insieme di linee di testo con codifica ASCII con la struttura
campo: **valore**
 - ▶ Le intestazioni possibili e la loro semantica sono definite e standardizzate in RFC 822, 2045 e 2046 (estensioni MIME)

Intestazione dei messaggi @ email

- ▶ Le intestazioni sono inserite dall'user agent e/o dai MTA
 - ▶ Alcune intestazioni sono informative, alcune servono per la consegna, altre per l'interpretazione del corpo del messaggio
 - ▶ Alcuni esempi
 - ▶ **To:** indirizzo DNS del destinatario (o destinatari)
 - ▶ **Cc:** copia in carta carbone (indirizzi "per conoscenza")
 - ▶ **Bcc:** copia in carta carbone con indirizzi invisibili ai riceventi
 - ▶ **From:** indirizzo di chi ha scritto il messaggio (necessario)
 - ▶ **Sender:** indirizzo di chi ha inviato il messaggio (opzionale)
 - ▶ **Subject:** oggetto del messaggio
 - ▶ **Date:** data e ora in cui il messaggio è stato inviato
 - ▶ **Reply-to:** indirizzo al quale inviare le risposte (reply) se diverso da From:
 - ▶ **Received:** indica l'agente che ha ricevuto il messaggio insieme a un timestamp. Viene inserito da ogni MTA durante il trasferimento
 - ▶ **Message-Id:** numero unico che individua il messaggio sul MTA origine

Un esempio di intestazione

Return-Path: <BWerner@computer.org>
Delivered-To: maggini@dii.unisi.it
Received: from firewall.dii.unisi.it (firewall.dii.unisi.it [10.0.0.1])
by alpha.dii.unisi.it (Postfix) with ESMTP id B763218337
for <maggini@dii.unisi.it>; Sat, 19 May 2001 03:02:57 +0200 (CEST)
Received: from sendmailout.computer.org (unknown [206.99.234.2])
by firewall.dii.unisi.it (Postfix) with ESMTP id 6A9053C0BC
for <maggini@dii.unisi.it>; Sat, 19 May 2001 02:48:10 +0200 (CEST)
Received: from cray.computer.org ([63.84.223.121])
by sendmailout.computer.org (Build 101 8.9.3/NT-8.9.3) with ESMTP
id UAA01113; Fri, 18 May 2001 20:03:57 -0400
Subject: ICDAR'01: Your Author Kit
To: ld47@csgz.edu.cn, maggini@ing.unisi.it
Cc: Karl.Tombre@loria.fr, haralick@gc.cuny.edu
X-Mailer: Lotus Notes Release 5.0.4 June 8, 2000
Message-ID: <OF1A96792E.5B9B2F1A-ON88256A50.00814155@computer.org>
From: BWerner@computer.org
Date: Fri, 18 May 2001 17:03:57 -0700
X-MIMETrack: Serialize by Router on Cray/IEEE Computer Society (Release
5.0.6a |January 17, 2001) at 05/18/2001 05:11:59 PM
MIME-Version: 1.0
Content-type: multipart/mixed;
Boundary="0__=88256A50008141558f9e8a93df938690918c88256A5000814155"
Content-Disposition: inline

Codifica dei caratteri su ASCII 7 bit 1

- ▶ Il messaggio email può contenere solo caratteri con **codifica NVT ASCII a 7 bit**
 - ▶ sono trasmessi su 8 bit con il bit più significativo a 0 (storicamente era utilizzato per il bit di controllo parità)
 - ▶ per trasmettere caratteri non NVT ASCII occorre utilizzare una **codifica** opportuna
 - ▶ trasmissione di dati binari (attachment) come ad esempio immagini, documenti, file compressi
 - ▶ trasmissione di testi che contengono caratteri non ASCII base (ASCII esteso per il set di caratteri con codici fra 128 e 255) o set di caratteri diversi (Unicode, Japanese, Chinese, ...)
 - ▶ Nell'intestazione del messaggio si deve poter indicare
 - ▶ Il **tipo di dati** contenuto e il **metodo di codifica su ASCII 7 bit** usato
 - ▶ Nel caso di testi, il set di caratteri utilizzato (**Character Encoding - charset**)

Character Encoding & ASCII esteso

- ▶ Un **Character Encoding** è l'assegnazione di un codice binario ad ogni carattere di un alfabeto
 - ▶ il **charset** è un'informazione fondamentale da rendere nota nello scambio dati per questo è sempre prevista una modalità per associare un **meta-dato che definisce il charset utilizzato**
 - ▶ se i dati non sono associati al charset appropriato non si è in grado di visualizzarli correttamente
 - ▶ **ISO 8859** descrive estensioni al set ASCII a 7 bit per linguaggi che utilizzano caratteri non presenti nella codifica base per l'Inglese
 - ▶ permette ulteriori 96 caratteri stampabili oltre a quelli ASCII standard 7 bit
 - ▶ ne esistono versioni diverse che codificano set di caratteri diversi, es.
 - ISO-8859-1 Latin-1 (copertura completa dell'Italiano)
 - ISO-8859-10 Latin-6 (linguaggi nordici)

ASCII e ASCII esteso

ASCII (1977/1986)

	_0	_1	_2	_3	_4	_5	_6	_7	_8	_9	_A	_B	_C	_D	_E	_F
0_0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1_16	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2_32	SP	!	"	#	\$	%	&	'	(	)	*	+	,	-	.	/
3_48	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4_64	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5_80	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6_96	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7_112	p	q	r	s	t	u	v	w	x	y	z	{		}	~	DEL

- ▶ Set esteso di caratteri
 - ▶ Codici > 127
 - ▶ Es. è E8 (1110 1000)
 - ▶ E 8

ISO/IEC 8859-1

Hex	_0	_1	_2	_3	_4	_5	_6	_7	_8	_9	_A	_B	_C	_D	_E	_F
0_																
1_																
2_	SP	!	"	#	\$	%	&	'	(	)	*	+	,	-	.	/
3_	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4_	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5_	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6_	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7_	p	q	r	s	t	u	v	w	x	y	z	{		}	~	
8_																
9_																
A_		ì	í	î	ï	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û
B_	°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿
C_	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
D_	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
E_	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
F_	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

Codifica dei caratteri su ASCII 7 bit 2

- ▶ Una stringa nell'intestazione può essere codificata come

`=?charset?encoding?encoded-text?='`

- ▶ **charset** è una stringa standard che descrive il character encoding
- ▶ **encoding** è il tipo di codifica usata per mappare i caratteri non ASCII a 7 bit ovvero quelli con valore nel byte > 127
 - ▶ La codifica **quoted-printable (Q)** codifica i caratteri con codice > 127 con `=` seguito dai due caratteri delle cifre esadecimali del codice del carattere (0–9, A–F). Ad esempio `è` si codifica con `=E8`.
 - ▶ La codifica **base64 (B)** codifica 3 caratteri consecutivi (24 bit) come 4 cifre in base 64 (ogni cifra corrisponde a 6 bit) codificate utilizzando 64 caratteri ASCII a 7 bit
- ▶ **encoded-text** è il testo codificato

Subject: `=?ISO-8859-1?Q?perch=E9_=E8_una_prova?='`

Encoding Base64

Index	Binary	Char	Index	Binary	Char	Index	Binary	Char	Index	Binary	Char
0	000000	A	16	010000	Q	32	100000	g	48	110000	w
1	000001	B	17	010001	R	33	100001	h	49	110001	x
2	000010	C	18	010010	S	34	100010	i	50	110010	y
3	000011	D	19	010011	T	35	100011	j	51	110011	z
4	000100	E	20	010100	U	36	100100	k	52	110100	0
5	000101	F	21	010101	V	37	100101	l	53	110101	1
6	000110	G	22	010110	W	38	100110	m	54	110110	2
7	000111	H	23	010111	X	39	100111	n	55	110111	3
8	001000	I	24	011000	Y	40	101000	o	56	111000	4
9	001001	J	25	011001	Z	41	101001	p	57	111001	5
10	001010	K	26	011010	a	42	101010	q	58	111010	6
11	001011	L	27	011011	b	43	101011	r	59	111011	7
12	001100	M	28	011100	c	44	101100	s	60	111100	8
13	001101	N	29	011101	d	45	101101	t	61	111101	9
14	001110	O	30	011110	e	46	101110	u	62	111110	+
15	001111	P	31	011111	f	47	101111	v	63	111111	/
Padding		=									

JVBERi0xLjMKJcfsj6IKNSAwIG9iago8PC9MZW
5ndGggNiAwIFlvRmlsdGVyIC9GbGF0ZURlY29k

Codifica: 4 caratteri ASCII (4 byte)

01001010 01010110 01000010 01000101

J V B E
001001 010101 000001 000100
00100101 01010000 01000100

Dati: 3 byte - %PD in ASCII
(è l'inizio di un file PDF)

- ▶ Cifre in base 64 e loro codifica con caratteri ASCII
- ▶ 3 byte → 4 byte (4 caratteri ASCII)
 - ▶ +33% di dimensione



- ▶ Lo standard Unicode ha lo scopo di associare un codice univoco ad “ogni” carattere (www.unicode.org)
 - ▶ l’associazione è indipendente dalla lingua, piattaforma, applicativo
 - ▶ Definisce la codifica di tutti i caratteri utilizzati nelle principali lingue scritte compresi caratteri di punteggiatura, simboli matematici, simboli tecnici, ecc.
 - ▶ La versione 6.0 include 109.449 caratteri
 - ▶ supporta tre differenti codifiche
 - ▶ **UTF-8** prevede che tutti i caratteri siano codificati con una sequenza di byte di lunghezza variabile (da 1 a 4). Ha il vantaggio che codifica direttamente su 8 bit il set ASCII ed è la codifica più usata nel Web
 - ▶ **UTF-16** codifica su 16 bit (2 byte) i caratteri più usati, mentre gli altri necessitano di 2 unità da 16 bit
 - ▶ **UTF-32** è la codifica a lunghezza fissa. Ogni carattere Unicode è codificato su 32 bit

UTF-8

- ▶ Prevede una codifica con lunghezza variabile da 1 a 4 byte per i caratteri fino a U+10FFFF (RFC 3629)
 - ▶ Non codifica tutti i caratteri ([code points](#)) in Unicode, ma tutti quelli di maggiore interesse
 - ▶ i codici a 1 byte codificano i caratteri ASCII a 7 bit
 - ▶ i successivi 1920 caratteri sono codificati con 2 byte e comprendono tutti i caratteri latini, greci, cirillici, arabi, copti....
 - ▶ 3 byte sono necessari per codificare tutti i caratteri del [Basic Multilingual Plane](#)

bits	Last code point	byte 1	byte 2	byte 3	byte 4
7	U+007F	0xxxxxxx			
11	U+07FF	110xxxxx	10xxxxxx		
16	U+FFFF	1110xxxx	10xxxxxx	10xxxxxx	
21	U+1FFFFF	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx

€ U+20AC code: 00100000 10101100 UTF-8: 11100010 1000010 10101100

MIME

- ▶ **Multipurpose Internet Mail Extensions** (RFC 2045-2046)
 - ▶ supporto per codifiche dei testi non ASCII 7 bit
 - ▶ possibilità di aggiungere allegati (attachment) ai messaggi email
 - ▶ gestione di messaggi con più parti (multipart encoding)
 - ▶ uso di codifica non ASCII nell'intestazione
- ▶ Aggiunge dei campi di intestazione per definire la struttura del corpo del messaggio (è gestita dall'agente utente) e il tipo di contenuto
 - ▶ Mime-Version:
 - ▶ Content-Type:
 - ▶ Content-Transfer-Encoding:
 - ▶ Content-ID:
 - ▶ Content-Description:
 - ▶ Content-Disposition:

MIME Content-Type

► Content-Type

- Definisce il tipo di dati (**Internet Media Type**) come tipo/sottotipo
 - **Content-Type:** text/plain ; charset=ISO-8859-1
 - testo non formattato con codifica dei caratteri ISO-8859-1
 - **Content-Type:** text/html
 - un testo con comandi di formattazione HTML
 - **Content-Type:** image/jpeg
 - un'immagine in formato jpeg
 - **Content-Type:** multipart/mixed ; boundary="start-separator"
 - il contenuto è organizzato in più parti di tipo diverso che iniziano con la sequenza --start-separator
 - ciascuna parte è descritta da una propria intestazione con la stessa modalità
- L'informazione sul Content-Type è fondamentale per selezionare il modulo di visualizzazione corretto da parte dell'agente utente

MIME Content-Transfer-Encoding

- ▶ Permette di indicare se è stata applicata qualche codifica al tipo dati specificato in Content-Type
- ▶ indica se c'è stata una codifica binary-to-text specificando quale
 - ▶ **7bit**
 - è il valore di default che prevede la presenza solo di caratteri ASCII a 7 bit organizzati in linee di massimo 998 byte terminate con CR LF
 - ▶ **quoted-printable**
 - usa la codifica quoted-printable per usare solo caratteri ASCII a 7bit. E' adatta per testi in cui i caratteri con codice > 127 sono "pochi" dato che questi vengono codificati con 3 byte (es. =A3)
 - ▶ **base64**
 - usa la codifica base64 che è adatta per dati non testuali (i byte con valori > 127 e < 127 sono "equiprobabili")
 - ▶ **8bit**
 - caratteri codificati su 8 bit con linee di massimo 998 byte terminate con CR LF
 - ▶ **binary**
 - qualsiasi sequenza di byte

Esempio MIME (attachment)

MIME-Version: 1.0
Content-type: multipart/mixed;
 Boundary="0__=88256A50008141558f9e8a93df938690918c88256A5000814155"
Content-Disposition: inline

--0__=88256A50008141558f9e8a93df938690918c88256A5000814155
Content-type: text/plain; charset=Windows-1252
Content-transfer-encoding: quoted-printable

Dear Author:

Congratulations! We have been notified that your paper has been accepted

--0__=88256A50008141558f9e8a93df938690918c88256A5000814155
Content-type: application/pdf;
 name="=?Windows-1252?Q?icdar01=5Fcopyright=5Fform.pdf?="
Content-Disposition: attachment; filename="=?Windows-1252?Q?icdar01=5Fcopyright=5Fform.pdf?="
Content-transfer-encoding: base64

JVBERi0xLjIgDSXi48/TDQogDTEwIDAgb2JqDTw8DS9MZW5ndGggMTEgMCBSDS9GaWx0ZXIgL0Zs
YXRlRGVjb2RlIA0+Pg1zdHJlYW0NCKiJrFfZctvIFf0C/UPXvESuUBAaO+YplEx5WDNjuSi6nFTp

SMTP

- ▶ **Simple Mail Transfer Protocol** (RFC 821 - RFC 5321)
 - ▶ Il protocollo originale è del 1984, l'**Extended SMTP** del 2008
 - ▶ Definisce il protocollo di comunicazione fra due agenti per il trasferimento di un messaggio email
 - ▶ Utilizza TCP sulla porta server 25
 - ▶ Realizza una **modalità push** (il messaggio è trasferito dal mittente verso il ricevente quando è disponibile)
 - ▶ Prevede una sequenza di comandi (inviati in ASCII) necessaria per il trasferimento dei messaggi
 - ▶ HELO <host> - “Saluta” il server
 - ▶ MAIL From: <indirizzo> - Indica il mittente del messaggio
 - ▶ RCPT To: <indirizzo> - Indica il destinatario (recipient)
 - ▶ DATA - Invio corpo del messaggio terminato da un . su una linea
 - ▶ QUIT - Chiude la connessione

SMTP session & transactions

- ▶ Un **sessione** SMTP prevede la seguente procedura
 - ▶ Un SMTP client (**initiating agent**) apre una connessione TCP verso il server SMTP
 - ▶ Viene aperta una sessione con la negoziazione di parametri
 - ▶ il client invia il comando **HELO** (**EHLO**) al server e attende una lista di opzioni (es. 250 **8BITMIME**) che poi vengono attivate o meno
 - ▶ Il client effettua una serie di **transazioni** per inviare uno o più messaggi
 - ▶ La transazione consiste di tre sequenze comando/risposta
 - **MAIL** - definisce l'indirizzo del mittente
 - **RCPT** - definisce il destinatario
 - **DATA** - invia il testo del messaggio (header/CR+LF/corpo)
 - ▶ In seguito ad ogni comando il server invia una stringa di risultato con un codice che indica se l'operazione ha avuto successo o meno

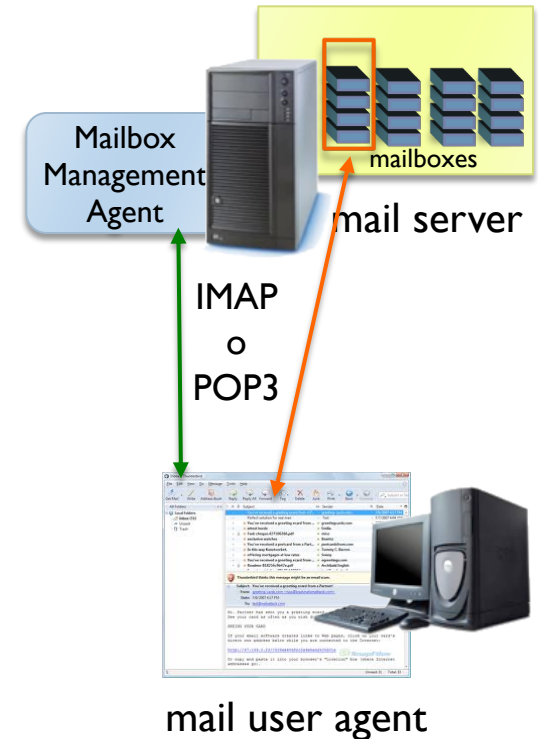
Un esempio di dialogo SMTP

```
maggini@mcculloch.ing.unifi.it... Connecting to alpha.dii.unisi.it. via relay...
220 alpha.dii.unisi.it ESMTP Postfix (Postfix-19991231) (Linux-Mandrake)
>>> EHLO ultra3.dii.unisi.it
250-alpha.dii.unisi.it
250-PIPELINING
250-SIZE 10240000
250-ETRN
250 8BITMIME
>>> MAIL From:<maggini@ultra3.dii.unisi.it> SIZE=20
250 Ok
>>> RCPT To:<maggini@mcculloch.ing.unifi.it>
250 Ok
>>> DATA
354 End data with <CR><LF>.<CR><LF>
>>> .
250 Ok: queued as BE37A18337
maggini@mcculloch.ing.unifi.it... Sent (Ok: queued as BE37A18337)
Closing connection to alpha.dii.unisi.it.
>>> QUIT
221 Bye
```

Post Office Protocol 3

Internet Mail Access Protocol

- ▶ La/Le mailbox sono in remoto
 - ▶ Per l'accesso alle mailbox si usa un'applicazione di tipo client/server
 - ▶ I protocolli per l'accesso alle mailbox usano una **modalità pull**
- ▶ **POP3** è semplice e ha funzionalità limitate
 - ▶ apre una connessione TCP sulla porta 110
 - ▶ autentica l'agente con username/password in chiaro (comandi `user`, `pass`)
 - ▶ effettua una transazione per recuperare i messaggi eventualmente marcandoli per cancellazione (comandi `list`, `retr`, `dele`)
 - ▶ applica l'aggiornamento sulla mailbox al momento del quit
- ▶ **IMAP** permette la gestione di cartelle remote





HTTP e il World Wide Web

Il World Wide Web

- ▶ Nato nel 1989 al CERN di Ginevra come mezzo per scambiare informazioni (Tim Berners-Lee, Robert Cailliau)
 - ▶ Sistema di **documenti ipertestuali e risorse** distribuite sulla rete Internet collegate fra loro con **hyperlink**
 - ▶ Pensato come sistema di gestione dell'informazione che sfrutta le infrastrutture di rete per facilitare la pubblicazione, distribuzione, condivisione e accesso alle informazioni
 - ▶ La struttura a ragnatela (**web**) definita dagli hyperlink è descritta dal grafo del Web (**webgraph**)
 - ▶ i nodi del grafo sono le pagine (risorse) Web
 - ▶ gli archi diretti sono gli hyperlink presenti nelle pagine
 - ▶ Il **World Wide Web Consortium** (www.w3c.org) è l'organizzazione che sovrintende allo sviluppo degli standard per il Web



IL Web come applicazione

- ▶ Il Web è nato come servizio disponibile sulla rete Internet
 - ▶ E' reso disponibile da un insieme di applicazioni server (i [server Web](#))
 - ▶ E' accessibile con un'applicazione client con interfaccia grafica (il [browser Web](#))
 - ▶ Si basa sulla tecnologia degli ipertesti realizzata con un linguaggio pensato per questa applicazione ([HTML](#) - [HyperText Markup Language](#))
 - ▶ Dalla prima proposta il linguaggio HTML si è evoluto per gestire la crescente complessità richiesta nella creazione dei documenti (attualmente è in uso la versione 5)
 - ▶ Definisce una modalità di indirizzamento delle risorse con l'[Uniform Resource Locator](#) ([URL](#))
 - ▶ Utilizza un protocollo applicativo pensato per questa applicazione [HyperText Transfer Protocol](#) ([HTTP](#))
 - ▶ E' diventato uno strumento diffuso per l'accesso a risorse su rete Internet

Evoluzione del Web

► Web 1.0

- L'idea originaria del Web è come collezione di documenti statici
- In teoria chiunque avrebbe potuto essere autore e pubblicare pagine sul Web
 - Gli strumenti per la creazione di pagine HTML si sono evoluti dagli editor di testo ad applicazioni WYSIWYG (WhatYouSeelsWhatYouGet)

► Web 1.x

- Le tecnologie del Web si sono evolute progressivamente per sfruttare le potenzialità di questo paradigma di accesso all'informazione
 - Possibilità di pubblicare informazioni in vari formati (immagini, video, audio,...)
 - Estensione dei protocolli per includere nuove funzionalità (HTTP1.1, supporto per i servizi interattivi con i Form HTML e l'architettura CGI)

Il Web “dinamico”

- ▶ Il Web assume una natura dinamica/interattiva
 - ▶ **Generazione dinamica delle pagine** Web Server-Side
 - ▶ Sviluppo di linguaggi di **script HTML embedded** (PHP, ASP)
 - ▶ Sviluppo di nuove tecnologie che trasformano i server Web da archivi di documenti ad **application server** (Servlet engines, JSP Java Server Pages)
 - ▶ **Programmazione e scripting Client-Side** in pagine Web interattive
 - ▶ Sviluppo della tecnologia **Java** e delle **applet**
 - ▶ Supporto per l'esecuzione di linguaggi di scripting da parte dei browser Web (**javascript**)
 - ▶ Possibilità di sviluppare “pagine Web” che in realtà sono applicazioni in esecuzione nel browser e che assumono sempre più l'aspetto di applicazioni native
 - ▶ Il Web si trasforma in una **piattaforma applicativa** per sviluppare servizi

- ▶ Le tecnologie alla base del Web sono divenute il supporto per la diffusione di **piattaforme di servizi distribuiti**
 - ▶ I **Web Services** utilizzano gli standard del Web (HTTP, URI, XML)
- ▶ Gli obiettivi del Web computing sono
 - ▶ Fornire l'accesso a servizi software (applicazioni) resi disponibili su rete Internet
 - ▶ Dare la possibilità di combinare servizi esistenti (**API di librerie di servizi**) per costruire applicazioni complesse (**mashup**)
 - ▶ es. www.programmableweb.com
 - ▶ Eventualmente pagare il software in base all'uso
 - ▶ I servizi accessibili agli utenti tramite pagine Web sono resi disponibili anche come API
 - ▶ es. API di google maps (<http://developers.google.com/maps/>)

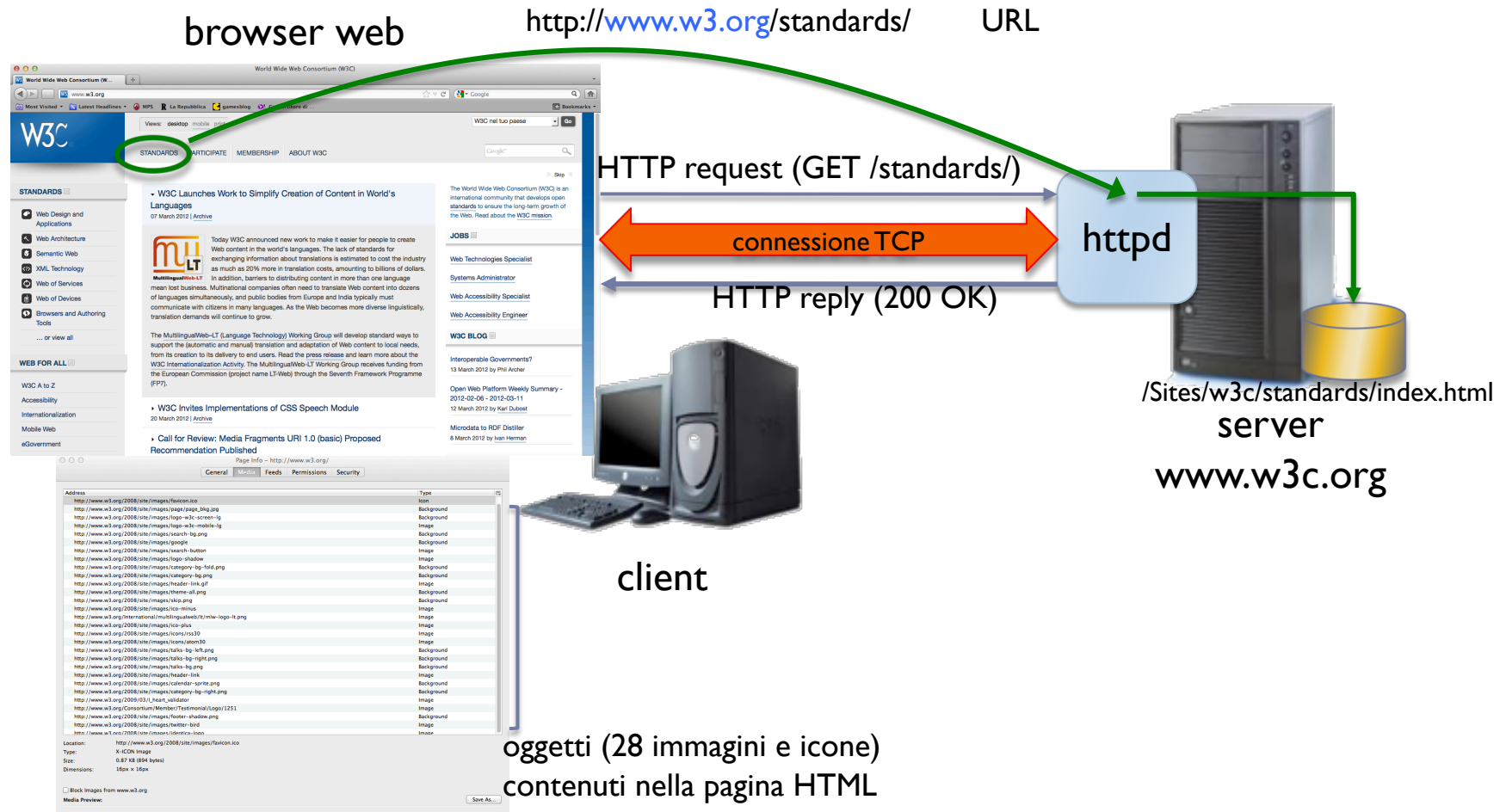
Web 2.0 e 2.x

- ▶ Il numero di versione indica una **modifica nel modo di usare il Web** e nei contenuti, metodologie e strumenti
 - ▶ Non esiste una versione del Web come quella dei programmi!
 - ▶ L'infrastruttura del Web è sempre la stessa (Internet e risorse collegate con hyperlink o riferite da URL)
 - ▶ Quello che cambia è l'esperienza che gli utenti hanno nel Web
 - ▶ Il Web è divenuto una piattaforma per lo sviluppo e la distribuzione di servizi e applicazioni
 - ▶ L'idea di Web 2.0 è legata ad **ruolo centrale di ogni singolo utente**
 - ▶ Sviluppo di reti sociali, tagging collettivo, forum, blog, wiki
 - ▶ Il Web 2.0 è più basato sui **dati** che sulle pagine (spostamento da HTML a XML e uso di javascript/Ajax lato client)
 - ▶ Es. **Content Syndication** - redistribuzione dei contenuti Web per mezzo di canali/feed (RSS - RDF Site Summary)

HTTP

- ▶ **HyperText Transfer Protocol - HTTP** - è il protocollo utilizzato per scambiare risorse sul Web
 - ▶ E' basato sul modello client/server (browser web/server web)
 - ▶ Utilizza **TCP** come protocollo di trasporto
 - ▶ Il server Web è in ascolto su una porta TCP (80 per default)
 - ▶ Il client apre una connessione TCP verso il server Web utilizzando le informazioni presenti negli hyperlink/URL
 - ▶ Il client invia la richiesta per la risorsa (la richiesta è un messaggio di testo ASCII)
 - ▶ Il server risponde inviando la risorsa richiesta o con un messaggio di errore
 - ▶ Non ci sono vincoli sul tipo di dati che possono essere trasferiti
 - ▶ testo, testo formattato HTML, dati binari (immagini, dati compressi, video,...)
 - ▶ in generale una **pagina web** può essere composta di più oggetti (il file base HTML, immagini, ecc.) che sono trasferiti con richieste distinte

Trasferimento HTTP



Uniform Resource Locator (URL)

- ▶ Un **URL (Uniform Resource Locator)** individua univocamente una risorsa su web
 - ▶ Permette di descrivere in modo “uniforme” la posizione di una risorsa sul Web rispetto al protocollo con cui si può accedervi
 - ▶ Contiene esplicitamente il riferimento all’host che la rende disponibile attraverso il suo nome DNS e opzionalmente alla porta
 - ▶ Non è un riferimento trasparente rispetto a spostamento/rimozione della risorsa (**broken links**)

protocol://host.domain:port/resource-path?query_string#fragment

http

nome DNS del server

percorso della risorsa & parametri (GET)

https

o indirizzo IP

ftp

mailto

URI, URL & URN

▶ URI (Uniform Resource Identifier)

- ▶ Necessità di identificare univocamente una risorsa sul Web
- ▶ Ogni risorsa è identificata da una stringa di caratteri opportunamente formattata

▶ URL (Uniform Resource Locator)

- ▶ Rappresenta un indirizzo preciso che può essere utilizzato immediatamente per accedere ad una risorsa
- ▶ Non è invariante rispetto alla rilocalizzazione della risorsa

▶ URN (Uniform Resource Names)

- ▶ Nome assegnato stabilmente (permanentemente) ad una risorsa
- ▶ E' indipendente dalla locazione fisica della risorsa
- ▶ Deve essere associato ad un URL attraverso un servizio di naming
 - un esempio è il [Digital Object Identifier](http://dx.doi.org/) (DOI) - www.doi.org
 - <http://dx.doi.org/10.1145/990301.990302>

Formato dei messaggi HTTP

- ▶ I messaggi HTTP di richiesta/risposta sono composti di due parti separate da una linea vuota
 - ▶ **Intestazione (header)**: è un insieme di linee di testo con codifica ASCII e struttura standard
 - ▶ **Corpo del messaggio (body)**: può contenere qualsiasi tipo di dato o essere anche vuoto

Linea Iniziale (diversa per richiesta e risposta)

Header1: valore1

Header2: valore2

Header3: valore3

intestazione

linea vuota

Corpo del messaggio

(contenuto del file, risultato di una interrogazione; può essere lungo molte linee e anche essere binario)

Richieste HTTP

- ▶ La linea iniziale di una richiesta è formata da 3 parti
 - ▶ Metodo (**GET**, **POST**, HEAD, PUT, DELETE, TRACE, **OPTIONS**, CONNECT, PATCH)
 - ▶ Indica la modalità della richiesta. Solo GET/POST sono obbligatori
 - ▶ Percorso locale della risorsa
 - ▶ E' la parte finale dell'URL dopo il nome dell'host
 - ▶ Versione di HTTP usata (HTTP/1.1 o HTTP/1.0)

```
GET /index.html HTTP/1.1
Host: univac.dii.unisi.it
User-Agent: Mozilla/5.0 (Windows; U; Win98; en-US; m18) Gecko/20010131
Netscape6/6.01
Accept: */*
Accept-Language: en
Accept-Encoding: gzip, deflate, compress, identity
Keep-Alive: 300
Connection: keep-alive
```


Metodi HTTP

- ▶ I metodi definiscono l'operazione da effettuare sulla risorsa richiesta. I principali sono
 - ▶ **GET**
Richiede il trasferimento di una risorsa. Se è seguita dall'intestazione **If-Modified-Since** il server invia i dati solo se sono stati modificati dopo la data specificata (gestione cache del browser)
 - ▶ **HEAD**
Richiede solo le intestazioni relative alla risorsa. Serve per verificare le caratteristiche della risorsa senza trasferirla
 - ▶ **POST**
Utilizzato per inviare dati da elaborare al server. L'intestazione è seguita da un corpo della richiesta che contiene i dati. Il tipo e la dimensione dei dati è indicata dagli header MIME **Content-Type:** e **Content-Length:**

Risposta HTTP

- ▶ La prima linea dell'intestazione riporta il codice dell'esito della richiesta (successo/errore)
- ▶ Le intestazioni riportano informazioni sul server e sul contenuto del corpo della risposta (la risorsa)

header
informazioni sul server
&
sulla risorsa (formato)

risorsa
pagina base HTML

```
HTTP/1.1 200 OK
Date: Wed, 06 Jun 2001 22:44:40 GMT
Server: Apache/1.3.20 (Win32)
Last-Modified: Wed, 06 Jun 2001 22:32:26 GMT
ETag: "0-64-3b1eaf7a"
Accept-Ranges: bytes
Content-Length: 100
Connection: close
Content-Type: text/html
```

```
<HTML>
<HEAD>
<TITLE>Prova</TITLE>
</HEAD>
<BODY>
<H2>File HTML di Prova</H1>
</BODY>
```

Codici di stato 1xx e 2xx

- ▶ La linea di stato riporta un **codice numerico** e una **stringa esplicativa**
 - ▶ Il contenuto del corpo della risposta dipende dal codice
 - ▶ E' la risorsa richiesta se il codice indica successo
 - ▶ E' vuoto o contiene una pagina di errore se il codice indica insuccesso
- ▶ I codici sono divisi in gruppi omogenei in base alla prima cifra
 - ▶ **1xx Informational**
 - Indica che la risposta contiene solo l'intestazione
 - **100 Continue**
Indica che il server ha ricevuto l'intestazione della richiesta e che il client può procedere ad inviare il corpo. E' usato per richieste di tipo POST che devono trasferire molti dati. Il client richiede questo tipo di conferma inserendo la linea di header **Expect: 100-continue** nella richiesta.
 - ▶ **2xx Success**
 - Indica che la richiesta è stata ricevuta, interpretata ed eseguita correttamente
 - **200 OK**
La risposta standard per richieste HTTP eseguite con successo

Codici di stato 3xx

► 3xx Redirection

- Indica che il client deve eseguire un'ulteriore operazione per accedere alla risorsa
 - **301 Moved Permanently**
 - Questa e le richieste future devono essere redirette all'URI indicato nel campo **Location:** dell'intestazione
 - **303 See Other**
 - La risposta alla richiesta può essere trovata con una richiesta GET all'URI indicato nel campo **Location:** dell'intestazione. L'URI fornito non sostituisce quello richiesto

HTTP/1.1 303 See Other

Location: http://www.otherserver.net/otherresource

► **304 Not Modified**

- Indica che la risorsa non è stata modificata dall'ultima richiesta. E' utilizzata in risposta ad una richiesta che contiene il campo **If-Modified-Since:** (gestione cache/proxy)

Codici di stato 4xx & 5xx

▶ 4xx Client Error

- ▶ Indicano casi in cui l'errore è attribuibile al client
 - ▶ **403 Forbidden**
 - La richiesta riguarda una richiesta per cui non si hanno permessi (es. Directory listing disabilitato)
 - ▶ **404 Not Found**
 - La richiesta fa riferimento ad una risorsa che al momento non esiste

▶ 5xx Server Error

- ▶ Il server non riesce a rispondere ad una richiesta valida
 - ▶ **500 Internal Server Error**
 - Messaggio di errore generico (es. la risorsa richiede l'esecuzione di un programma che fallisce)
 - ▶ **503 Service Unavailable**
 - Il server non è al momento disponibile a rispondere alla richiesta per manutenzione o carico eccessivo

Campi dell'intestazione

- ▶ I campi dell'intestazione definiscono dei parametri operativi di una transazione HTTP
 - ▶ Sono linee di testo codificate come coppie **Nome:** **valore**
 - ▶ Le linee sono terminate dalla sequenza di caratteri CR LF
 - ▶ Campi lunghi possono essere spezzati su più linee; nelle linee dalla seconda in poi il primo carattere deve essere uno spazio o tab
 - ▶ Un insieme base di campi è stato standardizzato (RFC 2616) e devono essere implementati
 - ▶ Possono comunque essere utilizzati altri campi definiti (e usati) da applicazioni particolari
 - ▶ Lo standard non impone vincoli sulla dimensione dei singoli campi e sul loro numero
 - ▶ In pratica invece i server web/user agent possono avere limiti sulla dimensione massima (in byte) dell'intestazione e/o sul numero di campi

Esempi di campi della richiesta 1

► **User-Agent :**

- Una stringa che identifica l'user-agent (il client/browser)
- Specifica il nome e la versione del prodotto e del motore di layout
- Può essere usata dal server per ottimizzare il formato della risorsa (es. layout della pagina) per una serie di user-agents predefiniti

User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_7_3)
AppleWebKit/534.54.16 (KHTML, like Gecko) Version/5.1.4
Safari/534.54.16

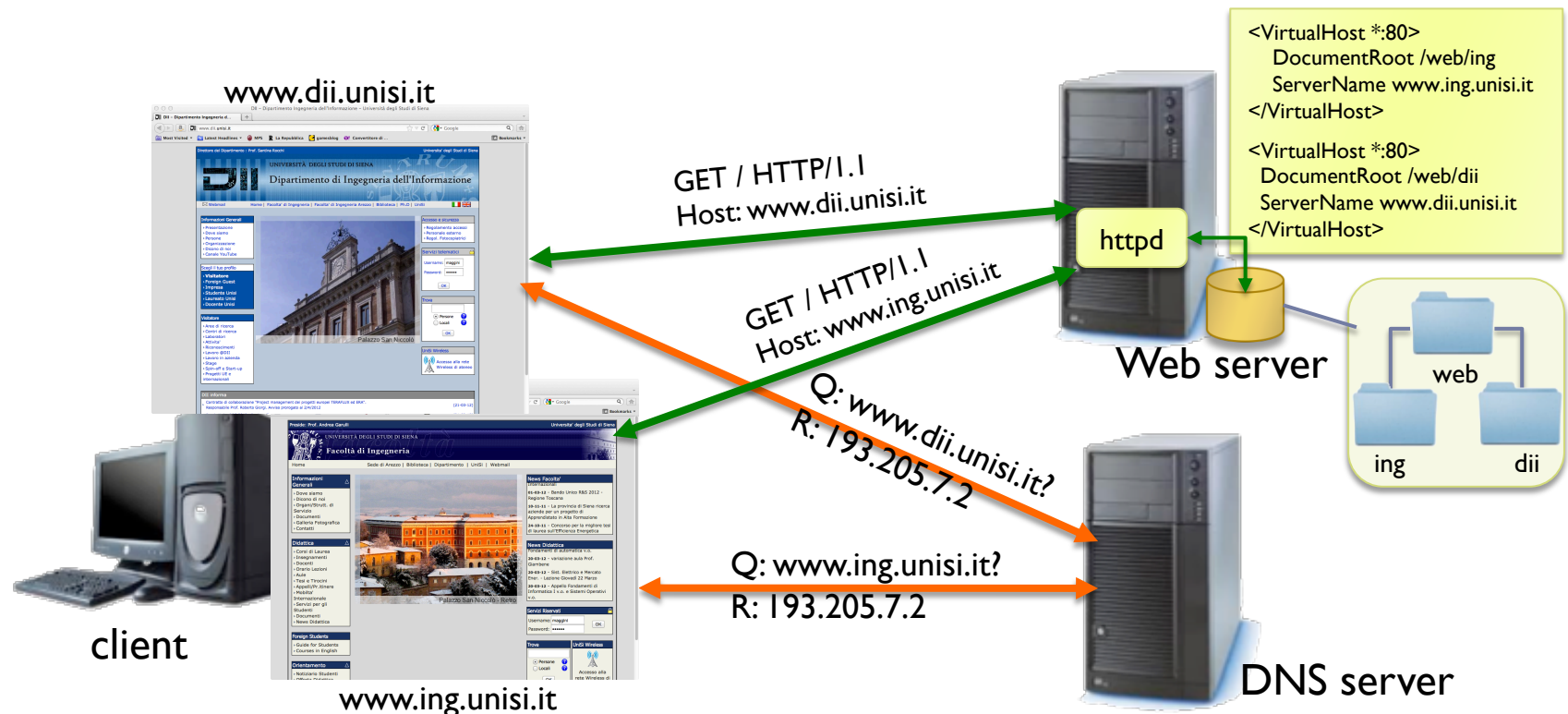
► **Host :**

- E' un campo obbligatorio per HTTP/1.1 e riporta l'hostname a cui è diretta la richiesta (è l'hostname specificato nell'URL)
- Permette la gestione del **virtual hosting** (più siti Web sullo stesso server con un unico IP)

Host: www.ing.unisi.it

(Name-based) Virtual hosting

- ▶ Il virtual hosting è una tecnica per ospitare più nomi di dominio su un singolo server
 - ▶ Permette di creare server Web condivisi (**shared web hosting**)
 - ▶ Il Web server utilizza il campo Host dell'intestazione per decidere quale sito web (**virtual host**) utilizzare



Esempi di campi della richiesta 2

► **If-Modified-Since:**

- Serve per gestire il trasferimento condizionale alla ultima data di modifica della risorsa
- Se la data di modifica sul server non è successiva a quella indicata, questo genera una risposta **304 Not modified** e non la trasferisce
- E' stato introdotto per supportare la gestione della cache sia a livello di browser che di proxy

If-Modified-Since: Sat, 24 Mar 2012 20:38:22 GMT

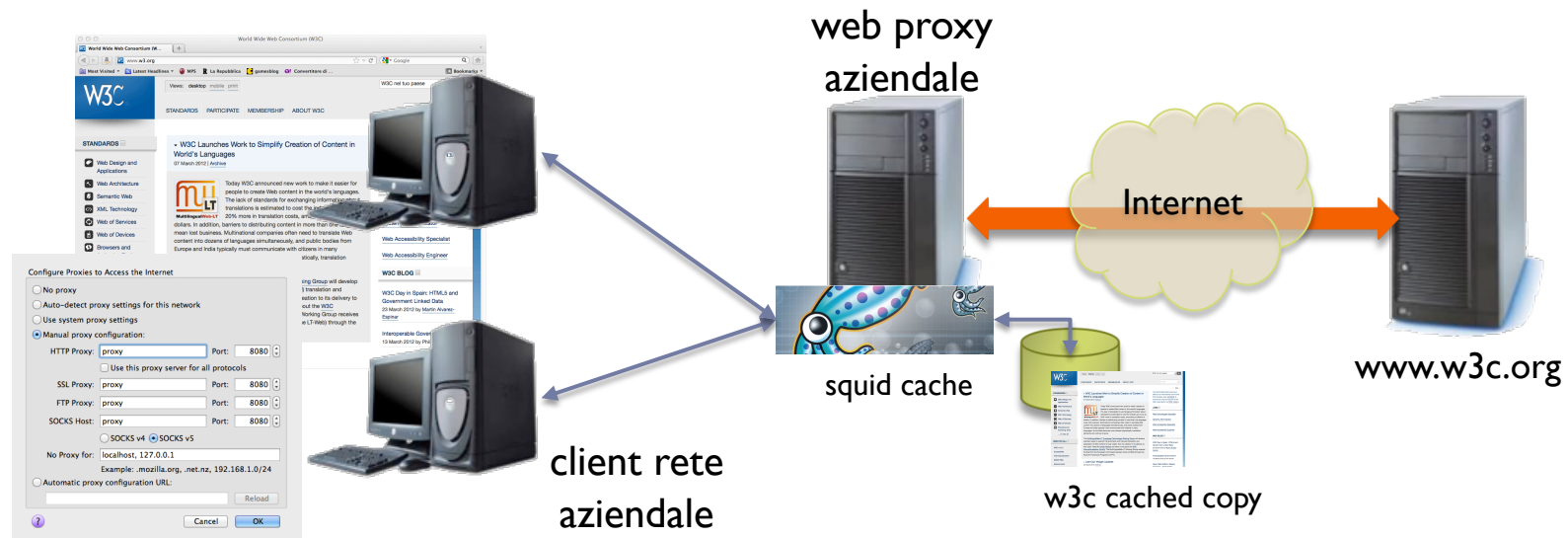
► **Cache-Control:**

- Permette di specificare delle direttive che devono essere applicate nella catena richiesta-risposta
- E' possibile specificare la vita massima della risorsa (**max-age=[seconds]**), se la risorsa non è cachabile (**no-cache**), se la risorsa può essere cachata solo nel browser e non in proxy condivisi (**private**), ECC..

Cache-Control: max-age=3600, private

Web Proxy

- ▶ Un **Web proxy** è un intermediario per le richieste HTTP da un client verso i server Web
 - ▶ Riceve le richieste dal client, le propaga al server destinazione, recupera la risposta, e infine inoltra la risposta al client
 - ▶ L'uso del proxy Web è configurabile a livello di SO e/o di browser
 - ▶ Permette la gestione di **caching condiviso**, controllo di accesso e può essere l'unico canale per l'accesso al Web da una rete privata aziendale
 - ▶ Riduce l'uso della banda per l'accesso a risorse di interesse comune



Esempi di campi della richiesta 3

▶ **Accept:**

- ▶ Indica una lista di tipi MIME che sono compatibili con l'user-agent
- ▶ Si possono indicare tutti i tipi con la wildcard */*
- ▶ In genere i browser possono gestire comunque i vari formati con plugin, applicativi esterni o permettendo di salvare il file
 - ▶ Il comportamento dei browser è configurabile con l'associazione dei tipi
- ▶ Il server può decidere di scegliere il formato della risposta in base ai tipi indicati dal browser

Accept: `text/html`

▶ **Accept-Language:**

- ▶ Permette all'user-agent di specificare una lista di lingue per cui ha preferenza
- ▶ Il server può inviare una versione localizzata della risorsa in base alla lingua indicata
- ▶ Dipende dalla lingua in cui è installato il browser

Accept-Language: `it, en`

Esempi di campi della risposta

▶ **Age:**

- ▶ Il tempo in secondi trascorso dalla risorsa in una cache

Age: 12

▶ **Content-Language:**

- ▶ La lingua in cui è la risorsa (se il server la specifica)

Content-Language: it

▶ **Content-Type: - Content-Length:**

- ▶ Il tipo MIME e la lunghezza in byte della risorsa
- ▶ Il tipo MIME permette al browser di interpretare in modo corretto il formato della risorsa (per default in genere assume che sia **text/plain**)
- ▶ Se la lunghezza della risorsa non è nota (contenuto generato dinamicamente) il server Web non specifica il campo **Content-Length** e utilizza una codifica “a pezzi” specificando **Content-Encoding: chunked**

Content-Type: image/gif

Content-Length: 4578

Caratteristiche HTTP

- ▶ Il protocollo HTTP è **senza stato** (**stateless protocol**)
 - ▶ Il server risponde alle richieste del client senza memorizzare nessuna informazione di stato sul client
 - ▶ Se il client invia più richieste queste sono di fatto indipendenti rispetto al meccanismo di HTTP
 - ▶ Se il client invia più volte la stessa richiesta il server risponde inviando lo stesso oggetto anche se non è stato modificato
- ▶ Più richieste possono essere gestite sulla stessa connessione persistente (HTTP/1.1 keep-alive)
 - ▶ Permette di gestire una sequenza di richieste verso lo stesso server utilizzando la stessa connessione TCP
 - ▶ Può essere disabilitata per mezzo del campo di intestazione **Connection: close** e viene comunque chiusa dopo un timeout dall'ultima richiesta (15s per Apache 2.0)
 - ▶ Riduce l'uso delle risorse

HTML (HyperText Markup Language)

- ▶ HTML è un linguaggio per descrivere la struttura delle pagine Web
 - ▶ Utilizza **Markup Tags** (annotazioni) per inserire comandi di formattazione di un testo
 - ▶ I Tag HTML sono keyword delimitate dai caratteri `< >`
 - ▶ In generale è previsto un tag di apertura e un tag di chiusura

`<B>Testo in grassetto</B>`

Testo in grassetto

- ▶ Una coppia di tag individua un **elemento** della pagina
- ▶ Il browser usa i tag per effettuare il rendering grafico della pagina
- ▶ Per ogni tag possono essere specificati degli attributi che permettono di definire o modificare le proprietà del tag stesso
 - Per ogni tag sono predefiniti una serie di attributi con la relativa semantica e valori ammessi

`<B style="color:red">Testo in grassetto rosso</B>`

Testo in grassetto rosso

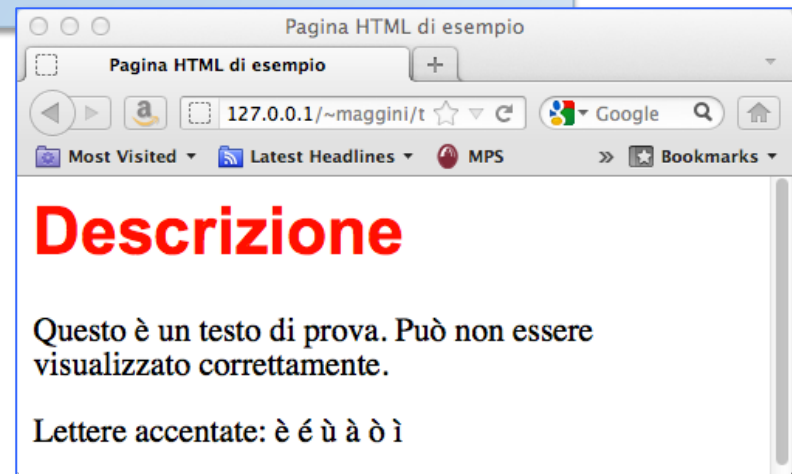
Struttura di un documento HTML

- ▶ HTML prevede la definizione di due parti del documento racchiuse fra i tag `<html>...</html>`
- ▶ **Intestazione** `<head>...</head>`
 - ▶ Permette di specificare informazioni sul documento
 - Il titolo `<title>Titolo</title>` mostrato sulla barra del browser
 - Specifiche di stile con il tag `<style type="text/css">...</style>`
 - Codice di script usati lato client nella pagina `<script>..</script>`
 - Il riferimento a risorse esterne come fogli di stile (l'attributo `rel` specifica il tipo di relazione, `type` il tipo MIME, `href` l'URL della risorsa)
`<link rel="stylesheet" type="text/css" href="site.css" />`
 - Metadati associati al documento (keywords, author, intestazioni HTTP, charset) `<meta charset="UTF-8" />`
- ▶ **Corpo** `<body>...</body>`
 - ▶ Racchiude il contenuto testuale del documento insieme agli elementi (tag) necessari per la formattazione e l'identificazione delle parti

Esempio di pagina Web 1

```
<html>
<head>
  <title>Pagina HTML di esempio</title>
  <style type="text/css">
    h1 {color:red;font-family:arial}
  <meta charset= "UTF-8" />
</head>
<body>
  <h1>Descrizione</h1>
  <P>Questo è un testo di prova. Può non essere visualizzato
  correttamente.</P>
  <P>Lettere accentate: è é ù à ò ì</P>
</body>
</html>
```

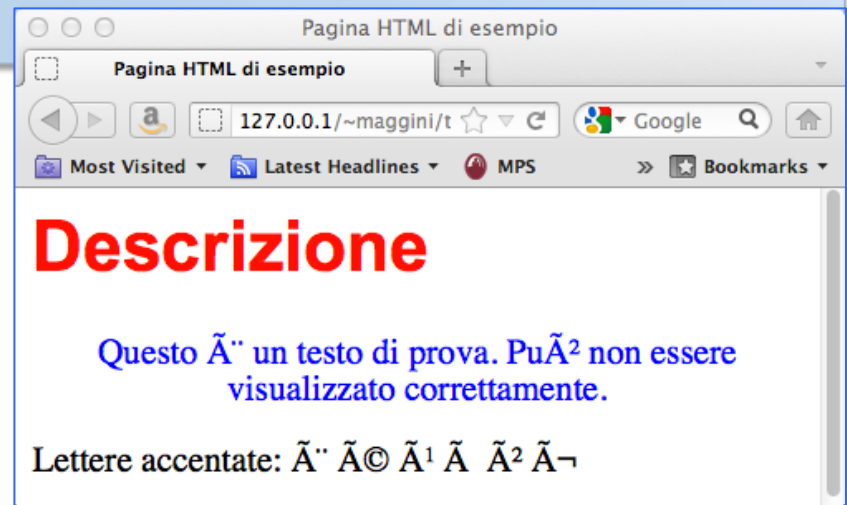
- ▶ l'intestazione definisce
 - ▶ Il titolo
 - ▶ lo stile di default (colore, font) per il tag **h1** (titolo di livello 1)
 - ▶ Il tipo di contenuto compreso la codifica dei caratteri usata (UTF-8)



Esempio di pagina Web 2

```
<html>
<head>
  <title>Pagina HTML di esempio</title>
  <style type="text/css">
    h1 {color:red;font-family:arial}
    p.nice {color: blue; text-align:center}
  <meta charset="iso-8859-1" />
</head>
<body>
  <h1>Descrizione</h1>
  <P class="nice">Questo è un testo di prova. Può non essere visualizzato
  correttamente.</P>
  <P>Lettere accentate: è é ù à ò ì</P>
</body>
</html>
```

- ▶ E' stato inserito un charset errato (il testo è stato inserito con UTF-8)
 - ▶ indicando iso-8859-1 che è una codifica a lunghezza fissa su 8 bit si vede che i caratteri accentati hanno una codifica su 2 byte
- ▶ E' stato aggiunto uno stile per la classe di paragrafi "nice"

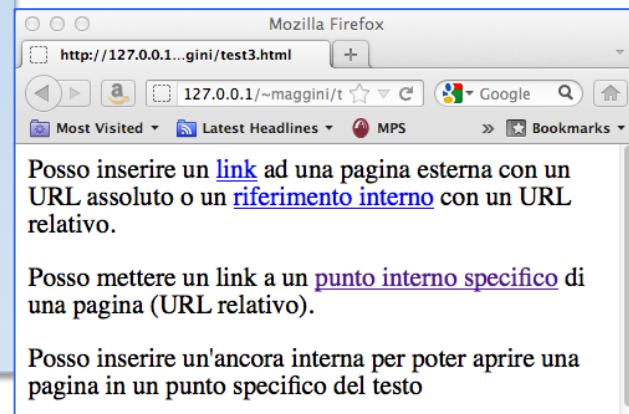


Ancore

▶ Ancora `<a>..</a>`

- ▶ Permette di inserire un link nella pagina o un riferimento ad una posizione della pagina
 - ▶ Se specifica un riferimento, l'URL riferito è indicato come valore dell'attributo `href` (l'URL può essere assoluto - un URL completo o che inizia per / - o relativo alla pagina corrente)
 - Il testo fra i tag è il link evidenziato sulla pagina Web
 - ▶ Se marca una posizione interna nella pagina, l'attributo `name` definisce l'etichetta da usare nell'URL per puntare a questa posizione

```
<body>
<p>Posso inserire un
<a href="http://www.myserver.it/mydir/mypage.html">link</a>
ad una pagina esterna con un URL assoluto o un
<a href=" ../localpage.html">riferimento interno</a>
con un URL relativo. </p>
<p>Posso mettere un link a un
<a href="test3.html#sec1">punto interno specifico</a>
di una pagina (URL relativo). </p>
<p> <a name="sec1" />Posso inserire un'ancora
interna per poter aprire
una pagina in un punto specifico del testo </p>
</body>
```



Tag principali: immagini

- ▶ Inserimento di immagini `<img />`
 - ▶ Specifica il riferimento URL con il campo `src`
 - ▶ Con gli attributi si possono specificare le dimensioni (`width,height`) e una descrizione testuale (`alt`)
 - ▶ Con l'attributo `style` si possono specificare opzioni di visualizzazione (`bordo, allineamento`)

```
<body>  
<h3 style="text-align:center">Un coniglio</h3>  
  
<br />  
</body>
```



Tag principali: formattazione del testo

- ▶ HTML prevede tag per definire l'aspetto con cui è visualizzato il testo interno all'elemento
 - ▶ Ad esempio `<b>` grassetto, `<em>` italico, `<h1>` intestazioni di livello 1-6
 - ▶ Alcuni sono sconsigliati dalla specifica HTML 4.01 perché è preferibile utilizzare l'attributo `style` (`<font>`, `<center>`)
 - ▶ Alcuni permettono di definire l'organizzazione del testo in paragrafi `<p>`, in liste ordinate `<ol>` e non ordinate `<ul>` di elementi `<li>`, o di inserire un'interruzione di linea `<br />`

```
<h4>Stili del testo</h4>
<ul>
  <li><b style="font-family:arial">Grassetto</b></li>
  <li><em style="font-family:courier">Italico</em></li>
  <li><strong style="font-family:times">Strong</strong></li>
</ul>
```

Stili del testo

- **Grassetto**
- *Italico*
- **Strong**

Attributi standard

- ▶ HTML definisce alcuni attributi validi per tutti gli elementi
 - ▶ **class**
 - ▶ permette di assegnare un elemento ad una classe. E' utile per definire uno stile comune ad un sottoinsieme di elementi di un certo tipo (riferito con *.nome*)
 - ▶ **id**
 - ▶ permette di assegnare un identificatore specifico ad un dato elemento (riferito con *#id*)
 - ▶ **style**
 - ▶ permette di definire le proprietà di stile per l'elemento usando il formato dei CSS (*cascading style sheet*). Le proprietà sono ereditate da tutti gli elementi annidati.

```
<head>
<style type="text/css">
  p.blu {font-family:arial;color:blue;text-align:center}
  p.red {font-family:courier;color:red}
  #obj3 {font: italic 18pt cursive;color:green}
</style>
</head>
<body>
<p class="blu" id="obj1">Un paragrafo di classe blu</p>
<p class="red" id="obj2">Un paragrafo di classe rossa</p>
<p id="obj3">Il paragrafo obj3</p>
<p class="blu" id="obj4">Un altro paragrafo di classe blu</p>
</body>
```

Un paragrafo di classe blu

Un paragrafo di classe rossa

Il paragrafo obj3

Un altro paragrafo di classe blu

Impaginazione con tabelle

- ▶ L'HTML ignora i caratteri di formattazione come spazi, tab e return
 - ▶ Per disporre i contenuti con un layout preciso si possono usare tabelle `<table>...</table>`
 - ▶ Le righe della tabella sono definite col tag `<tr>..</tr>` e le celle all'interno della riga con l'elemento `<td>..</td>`
 - ▶ Con l'attributo `colspan` è possibile creare una cella che occupa più colonne e con `rowspan` una cella che occupa più righe
 - ▶ Gli attributi `cellspacing` e `cellpadding` permettono di definire la spaziatura orizzontale e verticale delle celle

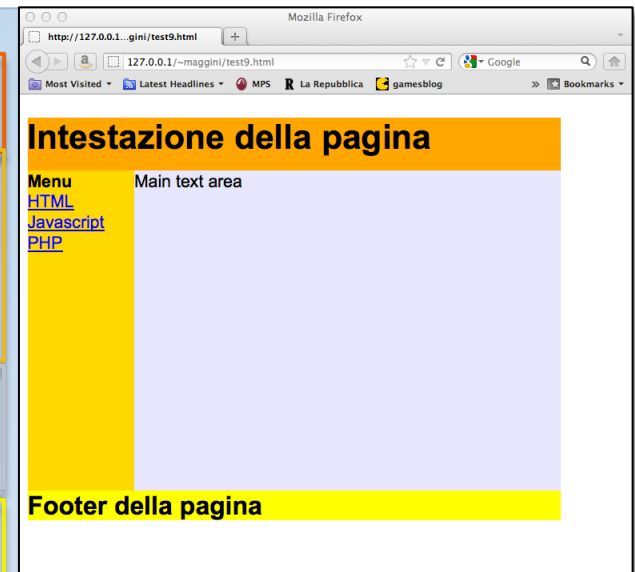
```
<head>
<style type="text/css">
  tr.head {color: #667788;background-color:#B0C4DE}
  tr.data {color: black;background-color:#FFE4E1}
</style>
</head>
<body>
<table style="text-align:center">
<tr class="head"><td>Nome e Cognome</td><td>Matricola</td><td>Voto</td></tr>
<tr class="data">
<td style="text-align:left">Mario Rossi</td><td>012345</td><td>18</td></tr>
<tr class="data">
<td style="text-align:left">Maria Bianchi</td><td>054321</td><td>21</td></tr>
<tr class="data">
<td style="text-align:left">Tim Barners-Lee</td><td>000001</td><td>30</td></tr>
<tr class="data">
<td colspan=2 style="text-align:right">Media</td><td>23</td></tr>
</table>
</body>
```

Nome e Cognome	Matricola	Voto
Mario Rossi	012345	18
Maria Bianchi	054321	21
Tim Barners-Lee	000001	30
Media		23

div e layout

- ▶ L'elemento `<div>` rappresenta un blocco di altri elementi
 - ▶ Può essere usato per definire il layout
 - ▶ Permette di ottenere gli stessi risultati dell'uso di tabelle
 - ▶ Le regole di design delle pagine Web suggeriscono di usare questa soluzione invece di quella basata su `<table>`

```
<div style="width:500px;font-family:arial;">  
  <div id="header" style="background-color:orange;height:50px">  
    <h1>Intestazione della pagina</h1>  
  </div>  
  <div id="menu" style="background-color:gold;  
    width:100px;height:300px;float:left">  
    <b>Menu</b><br />  
    <a href="html.html">HTML</a><br />  
    <a href="javascript.html">Javascript</a><br />  
    <a href="PHP.html">PHP</a>  
  </div>  
  <div id="content" style="background-color:lavender;  
    width:400px;height:300px;float:left">  
    Main text area  
  </div>  
  <div id="footer" style="background-color:yellow;clear:both">  
    <h2>Footer della pagina</h2>  
  </div>  
</div>
```



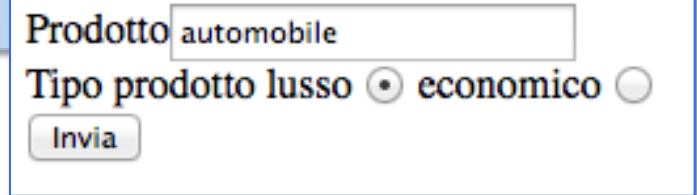
Passaggio di parametri in HTTP

- ▶ I servizi Web interattivi richiedono che il client possa inviare dati al server
 - ▶ In questo caso la risorsa richiesta al server Web è un'applicazione Web che viene eseguita per generare un contenuto Web dinamico
 - ▶ Il risultato dell'elaborazione è in genere una pagina Web che dipende dai parametri inviati ed eventualmente una modifica di stato sul server (es. memorizzazione di dati in un database)
- ▶ I dati sono inviati nella richiesta HTTP e occorre definire:
 - ▶ Come sono codificati i dati in modo che il server possa interpretarli correttamente
 - ▶ Come sono rappresentati i dati nella richiesta HTTP
 - ▶ Come avviene la gestione a livello server dei dati (come vengono resi disponibili allo script o applicazione che deve fare l'elaborazione)

Form HTML

- ▶ Un modo per gestire l'inserimento in una pagina Web di dati da inviare al server è l'uso di **Form HTML**
 - ▶ Un form è un oggetto della pagina Web individuato dalla coppia di tag **<FORM>..</FORM>**
 - ▶ All'interno del form sono specificati gli elementi **<INPUT>** che permettono l'inserimento di dati da parte dell'utente

```
<FORM action="http://www.mmnect.com/cgi-bin/query.cgi" method=GET>  
  Prodotto<INPUT name="prodotto" size=20 /><BR />  
  Tipo prodotto lusso<INPUT name="tipo" type=radio value="lusso" />  
  economico<INPUT name="tipo" type=radio value="economico" /><BR />  
  <INPUT type=submit value="Invia" />  
</FORM>
```



Prodotto

Tipo prodotto lusso ☒ economico ☐

Elementi <INPUT> e codifica dei dati

- ▶ Ad ogni elemento di INPUT è associato l'attributo **name** che definisce il **nome** di variabile a cui è associato il valore inserito per l'elemento
 - ▶ I parametri sono codificati come coppie **nome="valore"**
 - ▶ Il valore è una stringa di caratteri opportunamente codificata

```
<INPUT type="text" name="user">
```

pippo123

user="pippo123"

```
admin<INPUT type="radio" name="role" value="admin">
```

```
root<INPUT type="radio" name="role" value="root">
```

```
normal<INPUT type="radio" name="role" value="normal" checked="true">
```

admin ☐ root ☐ normal ☒

role="normal"

Tipi di Elementi INPUT

- ▶ Utilizzando l'attributo **type** è possibile variare il tipo di campo di input
 - ▶ **type=checkbox** definisce una scelta on/off
 - ▶ si può omettere l'attributo **value** che è la stringa "on" se il checkbox è selezionato

Interests


```
<INPUT type="checkbox" name="movie">movies<br />
```

```
<INPUT type="checkbox" name="music">music<br />
```

```
<INPUT type="checkbox" name="arts">arts<br />
```

Interests

☒ movies

☐ music

☒ arts

movie=on&arts=on

- ▶ **type=submit** inserisce un bottone che attiva l'invio dei dati con una richiesta HTTP alla risorsa specificata nell'attributo action del tag <FORM>
- ▶ L'attributo value può essere utilizzato per specificare l'etichetta sul bottone
- ▶ Se si inserisce l'attributo **name** viene creata la variabile corrispondente

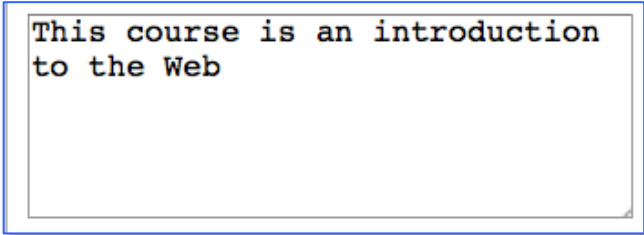
```
<INPUT type="submit" value="Invia" >
```

Invia

Altri oggetti interattivi 1

- ▶ I form HTML possono contenere altri oggetti che permettono all'utente di valorizzare variabili da inviare al server
 - ▶ Il tag `<textarea>` definisce un campo di testo multiriga di lunghezza non limitata

```
<textarea cols=30 rows=5 name="abstract"></textarea>
```



This course is an introduction
to the Web

`abstract=This+course+is+an+introduction+to+the+Web`

- ▶ Può essere inserito un testo più lungo dello spazio visualizzato. Nel caso compare una scrollbar
- ▶ Il valore assegnato alla variabile usa una codifica di tipo URL (`url-encoded`)

Altri oggetti interattivi 2

- ▶ Il tag `<select>` è utilizzato per creare menù a tendina
 - ▶ Le opzioni sono inserite col tag `<option>` che specifica la stringa da visualizzare
 - ▶ Se non è specificato esplicitamente con l'attributo `value`, il valore è la stringa contenuta nel tag `<option>` selezionato

```
<select name="genere">  
  <option>Heavy Metal</option>  
  <option>Death Metal</option>  
  <option>Gothic Metal</option>  
  <option>Hard Rock</option>  
  <option>Progressive Rock</option>  
</select>
```



`genere=Gothic+Metal`

- ▶ Con l'attributo `multiple` si può definire una lista con scelte multiple

```
<select name="genere" multiple=true size=5>
```

`genere=Heavy+Metal&genere=Gothic+Metal`



Invio dei dati con HTTP - GET

- ▶ I dati del form possono essere inviati con una richiesta HTTP di tipo **GET**
 - ▶ I parametri sono accodati all'URL della risorsa che li riceve usando in genere il carattere **?** come separatore
 - ▶ Questo metodo permette di associare la richiesta con i parametri ad un URL (può essere usato come hyperlink, generato in automatico,...)
 - ▶ Esiste un limite al numero dei caratteri e inoltre l'URL deve essere composto da caratteri ASCII
 - ▶ I caratteri non ASCII e i caratteri speciali (?,=,&,...) sono codificati con la sequenza % seguito dalle due cifre esadecimali (es %E8 per è)
 - ▶ Lo spazio è in genere codificato con il carattere + invece che con %20
 - ▶ Il metodo GET non dovrebbe essere usato per passare informazione sensibile (es. password) perché è in chiaro nell'URL e può essere memorizzato nei proxy

Esempio di form e richiesta GET

```
<form action="/ecom/buy.php" method="get">  
  Prodotto <input type="text" name="prodotto" size="20"><br/>  
  Acquisto <input type="checkbox" name="compra"><br />  
  <input type="submit" value="Invia">  
</form>
```

Prodotto

Acquisto ☒

GET /ecom/buy.php?prodotto=computer+più+2000&compra=on HTTP/1.1

.....

- ▶ L'attivazione del form con il bottone **submit** genera una richiesta HTTP
 - ▶ L'attributo **action** specifica l'URL della risorsa a cui inviare i dati del form (nell'esempio un URL relativo)
 - ▶ L'attributo **method** indica che la richiesta è di tipo GET e che di conseguenza il client deve accodare i parametri all'URL nella forma URL encoded

Invio dei dati con HTTP - POST

- ▶ I dati del form possono essere inviati con una richiesta HTTP di tipo POST
 - ▶ I dati sono inseriti nel corpo della richiesta HTTP
 - ▶ Non si ha un riferimento URL alla richiesta con i parametri valorizzati
 - ▶ E' più robusto e sicuro del metodo GET e non ha limitazione sulla dimensione dei dati inviati

```
<form action="/ecom/buy.php" method="post">  
  Prodotto <input type="text" name="prodotto" size="20"><br/>  
  Acquisto <input type="checkbox" name="compra"><br />  
  <input type="submit" value="Invia">  
</form>
```

Prodotto	computer più 2000
Acquisto	☒
Invia	

submit action

```
POST /ecom/buy.php HTTP/1.1  
host: www.mycommerce.it  
Content-type: application/x-www-form-urlencoded  
Content-Length: 38  
  
prodotto=computer+pi%F9+2000&compra=on
```


Encoding multipart/form-data

- ▶ L'attributo `enctype` permette di specificare il tipo di codifica con cui sono inviati i parametri di un form con metodo POST
 - ▶ Se non specificato la codifica è `application/x-www-form-urlencoded`
 - ▶ Se il form contiene un campo di input con `type="file"`, che permette di selezionare un file e inviarne il contenuto, occorre usare la codifica `multipart/form-data`
 - ▶ Utilizza il formato MIME multipart
 - ▶ Il messaggio contiene una serie di parti corrispondenti a ciascun elemento di input
 - ▶ Ciascuna parte ha un campo `Content-Type`: opzionale che per default è `text/plain`
 - ▶ Per ciascuna parte l'header `Content-Disposition`: specifica `form-data` e il nome del campo

Esempio multipart/form-data

```
<form action="http://myserver.it/archivia.php"
      enctype="multipart/form-data" method="post">

  Nome<input type="text" name="nome"><br />
  File<input type="file" name="file"><br />
  <input type="submit" value="Invia">
</form>
```



```
Content-type: multipart/form-data; boundary=-----63341846741
Content-Length: 339
```

intestazione HTTP

```
-----63341846741
Content-Disposition: form-data; name="nome"
```

marco

```
-----63341846741
Content-Disposition: form-data; name="file"; filename="pixel.gif"
Content-Type: image/gif
```

corpo della
richiesta HTTP

GIF89a

dati binari

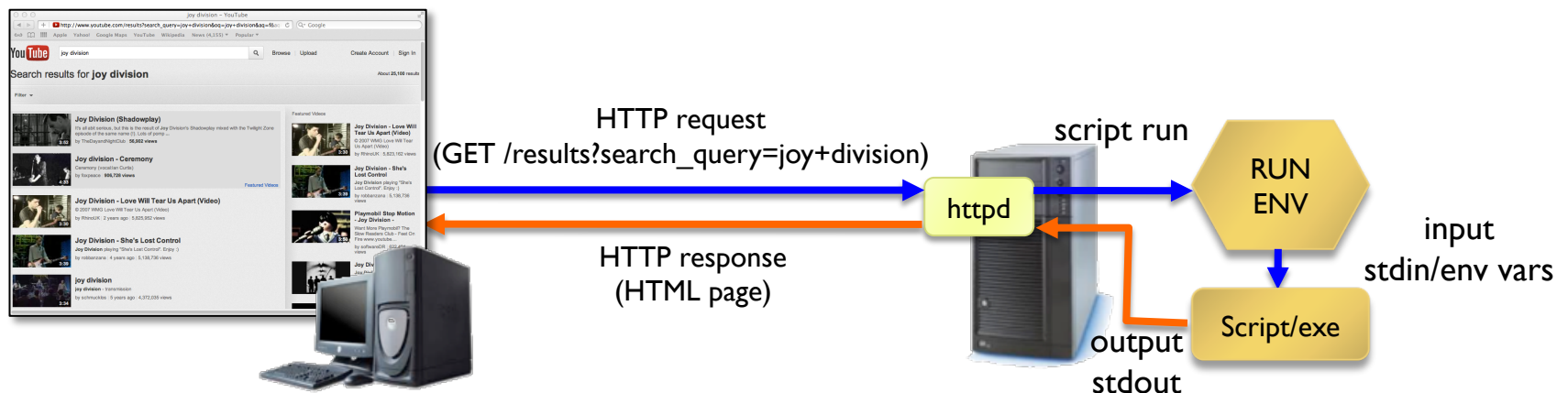
```
-----63341846741
```

Gestione di richieste “dinamiche”

- ▶ La risorsa che esegue l'azione individuata dall'URL specificato in un form deve eseguire **un'elaborazione lato server**
 - ▶ La risorsa è un **programma** (eseguibile o script interpretato) che deve essere eseguito dal server
 - ▶ I servizi sono riferiti e richiesti con la stessa modalità delle altre risorse statiche (URL e richiesta HTTP)
 - ▶ La richiesta HTTP a un servizio contiene (di norma) parametri, che servono all'elaborazione, codificati col metodo GET o POST
 - ▶ L'esecuzione del servizio produce dati che sono inviati dal server nella risposta HTTP
 - ▶ In genere è una pagina HTML, ma si possono generare altri tipi di dato come file pdf, immagini, ecc.. indicando il Content-Type opportuno
 - ▶ Il client **non “vede” cosa è eseguito** ma solo il risultato dell'elaborazione
 - ▶ Dal lato server è importante definire una modalità “portabile” con cui il programmatore può scrivere un servizio

Supporto del server Web per i servizi

- ▶ Un server Web generico deve poter supportare l'esecuzione di applicazioni collegate ad un URL
 - ▶ Deve capire in base alla risorsa richiesta se questa è un file già disponibile nel filesystem locale (es. pagina HTML, immagine) o se è un programma che deve essere eseguito
 - ▶ Deve fornire un supporto standard con cui interfacciarsi con l'esecuzione dell'applicazione "esterna" (lanciare l'applicazione, rendere disponibili i parametri, prelevare l'output per trasmetterlo al client)
 - ▶ L'insieme di informazioni che il server Web rende disponibili all'applicazione (es. parametri della richiesta HTTP) deve essere standard



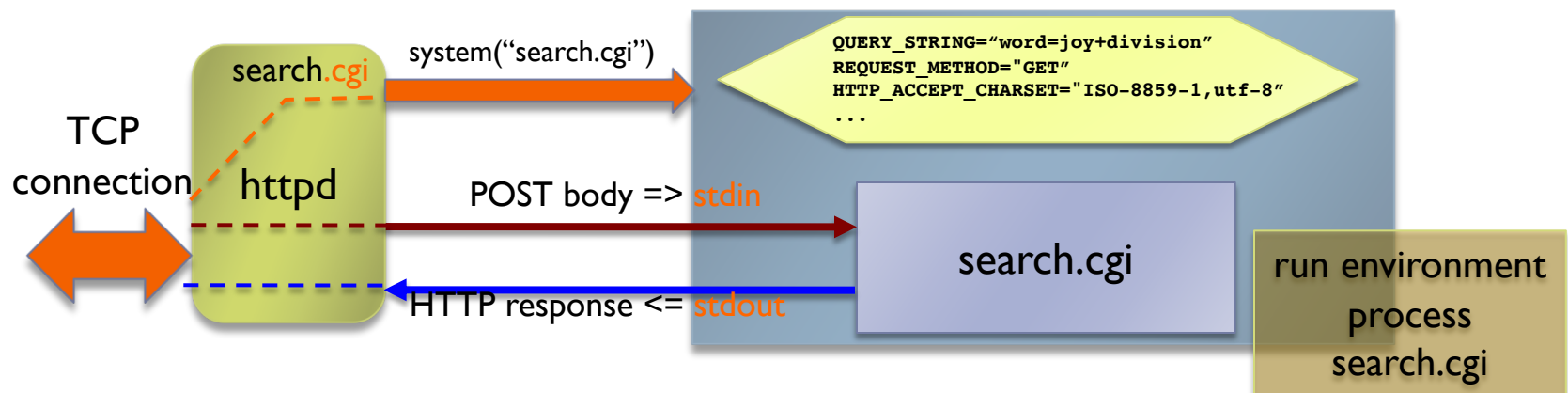
Gestione di eseguibili in un Web server

- ▶ Sono disponibili varie soluzioni per gestire l'esecuzione di eseguibili “esterni” ad un server Web
 - ▶ La possibilità di eseguire programmi di qualsiasi tipo rende il Web server flessibile
 - ▶ gli sviluppatori possono scrivere le applicazioni Web con il linguaggio e gli strumenti preferiti (C, C++, Java, perl, python,...)
 - ▶ Esiste un problema di efficienza per gestire un alto carico di richieste
 - ▶ Diventa critica la modalità con cui è realizzata l'esecuzione del programma esterno e come sono passati i dati della richiesta HTTP
 - ▶ Ad esempio lo standard **CGI** (**Common Gateway Interface**) indica una modalità di interazione fra un server Web e script eseguibili
 - ▶ Non è una soluzione particolarmente efficiente perché prevede l'esecuzione di un processo esterno **per ogni richiesta** e la procedura per l'esecuzione può essere più costosa di quella dell'effettiva generazione della risposta

CGI in un Web server



- ▶ La risorsa richiesta è identificata dal server web come un programma in base all'**estensione del nome del file** o alla **directory**
- ▶ Il server HTTP predispone l'esecuzione del programma CGI definendo l'ambiente di esecuzione assegnando i valori dei parametri che caratterizzano la connessione col client a **variabili di ambiente**
- ▶ Il server HTTP reindirizza l'input e l'output della connessione TCP del client sullo **stdin e stdout** del programma CGI
- ▶ Il server esegue l'applicazione che invia sul suo **stdout** la risposta HTTP (intestazione e corpo del messaggio) e chiude la connessione quando il programma CGI termina



Apache mod_cgi

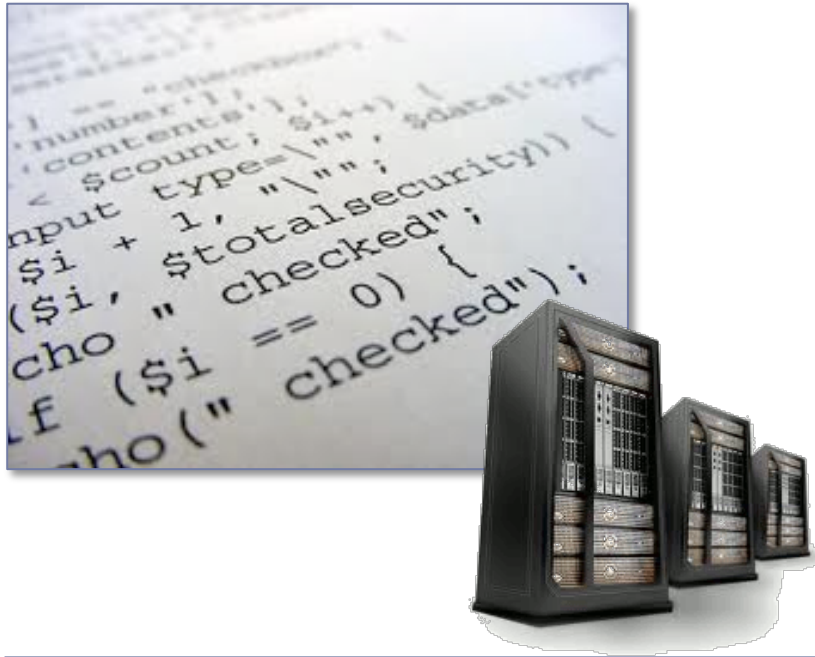
- ▶ Si può associare la gestione come CGI associando i file al gestore di cgi-script nel file di configurazione del server Web (`httpd.conf`)
 - ▶ in base all'estensione del nome del file (esempio `.cgi` `.pl` - perl)
`AddHandler cgi-script cgi pl`
 - ▶ in base alla directory in cui si trova il file
`ScriptAlias /cgi-bin/ /web/cgi-bin/`
- ▶ Per motivi di sicurezza è necessario abilitare l'esecuzione di script cgi per ogni directory (l'abilitazione è ereditata dalle sottodirectory)

```
<Directory /web/htdocs/specialdir>  
  Options +ExecCGI  
</Directory>
```

- ▶ Il programma CGI deve essere eseguibile sul SO dall'utente con cui è eseguito httpd (permesso `+x` su UNIX)

Programmazione server-side

- ▶ Sono disponibili soluzioni più efficaci ed efficienti per la programmazione server-side
 - ▶ Supporto specifico per la programmazione Web server-side
 - ▶ accesso diretto alle informazioni della richiesta HTTP con variabili
 - ▶ semplificazione della generazione dell'output (HTML embedded, manipolazione delle intestazioni HTTP)
 - ▶ Migliore interazione con il server Web
 - ▶ Ottimizzazione dell'esecuzione e del passaggio dei parametri
 - ▶ Ad esempio **moduli plug-in** per l'esecuzione degli script in Apache
 - lo script è eseguito da un modulo del server Web senza generare un processo esterno e la comunicazione usa metodi più efficienti
 - Per il PHP in `httpd.conf`
LoadModule `php5_module libexec/apache2/libphp5.so`
AddType `application/x-httpd-php .php`
 - ▶ Implementazione di server Web orientati all'esecuzione di applicazioni Web (**application server**) in uno specifico linguaggio
 - Java: Tomcat, IBM Websphere, GlassFish,... PHP: Zend PHP server



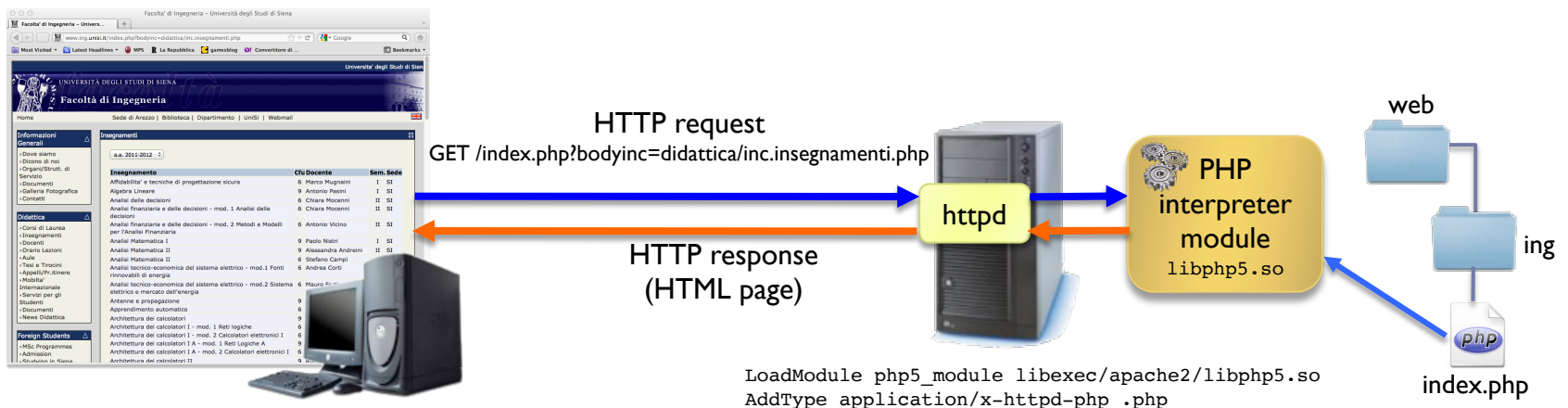
Reti di Calcolatori

Server-side programming & PHP

PHP



- ▶ Il PHP (**Personal Home Page** o **PHP: Hypertext Preprocessor**) è un linguaggio di scripting server-side utilizzato per lo sviluppo di pagine Web dinamiche
 - ▶ Il linguaggio è di tipo **HTML-embedded** ovvero le istruzioni sono inserite in un file HTML e vengono interpretate prima di essere trasmesse al client
 - ▶ E' un software **open source** disponibile su più piattaforme e compatibile con i principali server Web (es. apache, IIS)
 - ▶ Tecnologie simili sono **ASP** di Microsoft (Active Server Pages) e **JSP** (Java Server Pages)



Caratteristiche di PHP

- ▶ PHP è un linguaggio di scripting pensato per semplificare la programmazione lato server con **caratteristiche orientate al Web** ed una **ampia dotazione di librerie**
 - ▶ Può essere usato come linguaggio procedurale o ad oggetti
 - ▶ Fornisce il supporto per l'accesso ai dati inviati da form HTML
 - ▶ Permette di accedere ai valori di header della richiesta HTTP e di valorizzare header della risposta
 - ▶ Ha il supporto per la gestione dei cookie e delle sessioni
 - ▶ L'uso del PHP non è limitato alla generazione di HTML e file di testo in genere, ma sono a disposizione librerie di funzioni per la generazione di vari formati (es. immagini, pdf)
 - ▶ Sono disponibili librerie la connessione a vari DBMS (es. MySQL, PostgreSQL,..) e per la gestione di svariati protocolli di rete (es. IMAP, POP3, LDAP,..)
 - ▶ Ha il supporto per l'uso di socket, per l'elaborazione avanzata di testi (es. espressioni regolari), per l'analisi e l'elaborazione di documenti XML

HTML-embedding

- ▶ Uno script PHP è un file di testo dove il codice PHP eseguibile è inserito fra i tag `<?php ... ?>`
 - ▶ L'interprete funziona da filtro ovvero invia in uscita il testo al di fuori dei tag `<?php.. ?>` ed esegue le istruzioni al loro interno
 - ▶ Le istruzioni creano contenuto dinamico che viene inserito al posto del blocco compreso fra i tag `<?php .. ?>`

```
<html>
<body>
  <h2>Che ora &grave; per il server?</h2>
  <?php
    /*
     *  istruzioni PHP
     */
    # scrive il nome del server
    echo "<b>Server: </b>".$_SERVER['SERVER_NAME']."<br />";
    // scrive la data del server
    date_default_timezone_set("Europe/Rome");
    echo "\n<b>Data: </b>".date("F j, Y, g:i a")."\n";
  ?>
</body>
</html>
```

```
<html>
<body>
  <h2>Che ora &grave; per il server?</h2>
  <b>Server: </b>127.0.0.1<br />
  <b>Data: </b> April 8, 2012, 4:46 pm
</body>
</html>
```

Che ora è per il server?

Server: 127.0.0.1
Data: April 8, 2012, 4:46 pm

Ancora su HTML-embedding...

- ▶ Il “modo” PHP si può interrompere e riaprire in ogni momento
 - ▶ In ogni caso il flusso di uscita segue quello di esecuzione delle istruzioni ovvero il testo HTML in una parte interna ad un blocco PHP non eseguito non viene inviato in uscita

```
<h2>Che browser usi?</h2>
Client IP: <?php echo $_SERVER['REMOTE_ADDR'];?> <br />
Client Port: <?php echo $_SERVER['REMOTE_PORT'];?> <br />
Browser model:
<?php $agent = $_SERVER['HTTP_USER_AGENT'];
    if(strpos($agent,"Firefox")) { ?>
        <!-- questo HTML è nel blocco true del primo if -->
        <em style="color: orange">Firefox</em>
    <?php } else if(strpos($agent,"Chrome")) { ?>
        <!-- questo HTML è nel blocco true del secondo if -->
        <em style="color: green">Chrome</em>
    <?php } else if(strpos($agent,"Safari")) { ?>
        <!-- questo HTML è nel blocco true del terzo if -->
        <em style="color: gold">Safari</em>
    <?php } else { ?>
        <!-- questo HTML è nel blocco false del terzo if -->
        <em style="color: red">Unsupported browser</em>
    <?php } ?>

<br />User-agent string: <em><?php echo $agent;?></em>
```

Che browser usi?



```
Client IP: 127.0.0.1
Client Port: 51921
Browser model: Safari
User-agent string: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_7_3)
AppleWebKit/534.55.3 (KHTML, like Gecko) Version/5.1.5 Safari/534.55.3
```

```
<h2>Che browser usi?</h2>
Client IP: 127.0.0.1 <br />
Client Port: 51921 <br />
Browser model:
<!-- questo HTML è nel blocco true del
terzo if -->
<em style="color: gold">Safari</em>
<br />User-agent string: <em>Mozilla/5.0
(Macintosh; Intel Mac OS X 10_7_3)
AppleWebKit/534.55.3 (KHTML, like Gecko)
Version/5.1.5 Safari/534.55.3</em>
```

Configurazione di PHP

- ▶ Il file **php.ini** permette di configurare l'interprete PHP
 - ▶ Il file è commentato in dettaglio e la configurazione di default di solito è adeguata
 - ▶ La configurazione del PHP può essere visualizzata con uno script che chiama la funzione **phpinfo()**

```
<body>
<h2 style="text-align: center">
    phpinfo() on <?php echo $_SERVER['SERVER_NAME'];?>
</h2>
<?php phpinfo(); ?>
</body>
```

phpinfo() on 127.0.0.1

PHP Version 5.3.8	
	
System	Darwin edacs.local 11.3.0 Darwin Kernel Version 11.3.0: Thu Jan 12 18:47:41 PST 2012; root:xnu-1899.24.22~1/RELEASE_ARM64_T8020
Build Date	Nov 15 2011 15:30:33
Configure Command	./configure --prefix=/usr --mandir=/usr/share/man --infodir=/usr/share/info --disable-dependency-tracking --sysconfdir=/private/etc --with-apxs2=/usr/lib/apxs --enable-cli --with-config-file-path=/etc --with-libxml-dir=/usr --with-openssl=/usr --with-kerberos=/usr --with-zlib=/usr --enable-bcmath --with-bz2=/usr --enable-calendar --with-curl=/usr --enable-dba --enable-ndbm=/usr --enable-exif --enable-ftp --with-gd --with-freetype-dir=/usr --with-jpeg-dir=/usr --with-png-dir=/usr --with-ldap=/usr --with-ldap-sasl=/usr --with-ldap --with-icu-dir=/usr --with-iodbc=/usr --with-ldap --with-ldap-sasl --with-libidn=/usr --enable-mbstring --enable-mbregex --with-mysql=mysqlnd --with-mysqli=mysqlnd --without-pear --with-pdo-mysql=mysqlnd --with-mysql-socket=/usr/lib/mysql --with-readline=/usr --enable-shmop --with-snmp=/usr --enable-soap --enable-sockets --enable-sqlite3 --enable-suhosin --enable-sysmsg --enable-sysvshm --enable-xml --enable-xmlrpc --enable-zend-multibyte --enable-zip --with-pcre-regex=/usr --with-pgsql=/usr --with-pdo-pgsql=/usr
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc
Loaded Configuration File	/private/etc/php.ini

Configuration apache2handler

Apache Version	Apache/2.2.21 (Unix) mod_ssl/2.2.21 OpenSSL/0.9.8r DAV/2 PHP/5.3.8 with Suhosin-Patch		
Apache API Version	20051115		
Server Administrator	you@example.com		
Hostname/Port	edacs.local/0		
User/Group	www/70/70		
Max Requests	Per Child: 0 - Keep Alive: on - Max Per Connection: 100		
Timeouts	Connection: 300 - Keep-Alive: 5		
Virtual Server	No		
Server Root	/usr		
Loaded Modules	core prefork http_core mod_so mod_authn_file mod_authn_dbm mod_authn_anon mod_authn_dbd mod_authn_default mod_authn_host mod_authn_groupfile mod_authn_user mod_authn_dbm mod_authn_owner mod_authz_default mod_authz_basic mod_authz_digest mod_cache mod_disk_cache mod_mem_cache mod_dbd mod_dumpio mod_ext_filter mod_include mod_filter mod_substitute mod_deflate mod_log_config mod_log_forensic mod_logio mod_env mod_mime_magic mod_cern_meta mod_expires mod_headers mod_ident mod_usertrack mod_setenvif mod_version mod_proxy mod_proxy_connect mod_proxy_ftp mod_proxy_http mod_proxy_ajp mod_proxy_balancer mod_ssl mod_mime mod_dav mod_status mod_autoindex mod_asis mod_info mod_cgi mod_dav_fs mod_vhost_alias mod_negotiation mod_dir mod_imagemap mod_actions mod_speling mod_userdir mod_alias mod_rewrite mod_bonjour2 mod_php5		
Directive	Local Value	Master Value	
engine	1	1	
last_modified	0	0	
xbitack	0	0	

bz2

BZip2 Support	Enabled
Stream Wrapper support	compress.bzip2://
Stream Filter support	bzip2.decompress, bzip2.compress
BZip2 Version	1.0.6, 6-Sept-2010

calendar

Calendar support	enabled
------------------	---------

Core

PHP Version	5.3.8
-------------	-------

Directive	Local Value	Master Value
allow_call_time_pass_reference	Off	Off
allow_url_fopen	On	On
allow_url_include	Off	Off
always_populate_raw_post_data	Off	Off
arg_separator.input	&	&
arg_separator.output	&	&
asp_tags	Off	Off
auto_append_file	no value	no value
auto_globals_jit	On	On
auto_prepend_file	no value	no value

Variabili PHP

- ▶ Le variabili seguono un **meccanismo di dichiarazione implicita**
 - ▶ Sono create al momento dell'uso
 - ▶ Il tipo dipende dal dato che viene assegnato alla variabile e può quindi variare nel flusso del programma (anche se non è una buona norma...)
 - ▶ I nomi delle variabili iniziano col carattere **\$**
 - ▶ il **\$** può essere interpretato come un operatore di riferimento alla tabella dei simboli in cui sono memorizzate le variabili indicizzate per nome
 - ▶ Un nome di variabile può contenere caratteri alfanumerici (il primo carattere deve essere una lettera) e l'underscore **_**
 - **\$i**, **\$i1**, **\$_input**,
 - ▶ I nomi sono case-sensitive
 - **\$var** e **\$Var** sono variabili diverse
 - ▶ L'assegnazione copia il valore assegnato nella variabile

```
$v = 1; $v1 = $v; $v = 2;  
echo "v = ".$v."<br />v1 = ".$v1;
```

```
v = 2  
v1 = 1
```


Inizializzazione delle variabili

- ▶ Non è necessario inizializzare le variabili in PHP
 - ▶ Al primo uso assumono un valore di default per il tipo che meglio si adatta al contesto in cui sono usate
 - ▶ **false** per boolean, **0** per integer e float, la stringa vuota **""**, un array vuoto
 - ▶ Ad una variabile non inizializzata corrisponde il valore speciale **NULL**
 - ▶ In ogni caso non è una buona pratica lasciare variabili non inizializzate
 - ▶ Il PHP mette a disposizione costrutti per verificare se una variabile è stata inizializzata o se contiene valori "vuoti"
 - ▶ **bool isset(\$var)** produce **false** se **\$var** non è stata inizializzata
 - ▶ **bool empty(\$var)** produce **true** anche se **\$var** è inizializzata a **false**, **""** (la stringa vuota), **"0"** (la stringa col carattere 0), **0** o **0.0**, **array()**

```
if(!isset($_REQUEST['prodotto']))  
    echo "FORM variable prodotto is not set <br />";  
if(empty($_REQUEST['prodotto']))  
    echo "FORM variable prodotto is empty <br />";  
  
$v = 0;  
if(!isset($v))  
    echo "variable v is not set <br />";  
if(empty($v))  
    echo "variable v is empty<br />"
```

FORM variable prodotto is not set
FORM variable prodotto is empty
variable v is empty

Tipi di dato

- ▶ Il PHP supporta 8 tipi di dato primitivi - `gettype($var)`
 - ▶ Tipi scalari - `is_scalar($var)`
 - ▶ `boolean` - `is_bool($var)`
 - ▶ `integer` - `is_int($var)`, `is_numeric($var)`
 - ▶ `float` - `is_float($var)`, `is_numeric($var)`
 - ▶ `string` - `is_string($var)`
 - ▶ Tipi composti
 - ▶ `array` - `is_array($var)`
 - ▶ `object` - `is_object($var)`
 - ▶ Tipi speciali
 - ▶ `resource` - `is_resource($var)`
 - ▶ `NULL` - `is_null($var)`
 - ▶ `callable`

Le funzioni indicate permettono di verificare se una variabile contiene un valore di un dato tipo (producono **true/false**)

Variabili e tipi dato

- ▶ Il PHP non richiede di dichiarare esplicitamente il tipo di una variabile
 - ▶ Il tipo dipende dal valore assegnato
 - ▶ Il valore di una variabile viene convertito nel tipo appropriato (**type juggling**) in base al contesto in cui è usato senza variare la definizione della variabile stessa

Alla variabile `<tt>$var</tt>` viene assegnato il valore della costante `<tt>10</tt>` `<?php $var= 10; ?>` e quindi assume il tipo `<tt><?php echo gettype($var); ?></tt>` con valore `<tt><?php echo '$var = '.$var; ?></tt>`.

Se si calcola l'espressione `<tt>$var *= 1.5</tt>` `<?php $var *= 1.5; ?>` la variabile diventa di tipo `<tt><?php echo gettype($var); ?></tt>` con valore `<tt><?php echo '$var = '.$var; ?></tt>`.

Se si esegue l'assegnazione `<tt>$var+="44 cats"</tt>` `<?php $var += "44 cats"; ?>` la variabile è ora di tipo `<tt><?php echo gettype($var); ?></tt>` con valore `<tt><?php echo '$var = '.$var; ?></tt>`. La costante stringa "44 cats" viene convertita nel tipo della variabile (double) facendo una scansione da sinistra fino a che si trovano cifre.

Alla variabile `$var` viene assegnato il valore della costante 10 e quindi assume il tipo integer con valore `$var = 10`.

Se si calcola l'espressione `$var *= 1.5` la variabile diventa di tipo double con valore `$var = 15`.

Se si esegue l'assegnazione `$var+="44 cats"` la variabile è ora di tipo double con valore `$var = 59`. La costante stringa "44 cats" viene convertita nel tipo della variabile (double) facendo una scansione da sinistra fino a che si trovano cifre.

Conversione di tipo e cast

- ▶ Se una variabile viene utilizzata nella definizione di una costante stringa con *double quoted* "...\$var...", ne viene utilizzato il valore
- ▶ Se si utilizza una stringa in un contesto che richiede un numero, la stringa viene scandita da sinistra a destra fino a che si trovano cifre
 - ▶ Questa operazione implicita è particolarmente utile per l'elaborazione dei dati che provengono da form HTML che in origine sono sempre in variabili di tipo string
- ▶ Si può utilizzare il cast (**tipo**) (es. (**int**), (**float**)) per forzare la conversione di un'espressione ad un tipo dati specifico

Se si esegue l'assegnazione `<tt>$s = "2+$var"</tt>`
`<?php $s = "2+$var"; ?>` la variabile `<tt>$var</tt>` si trova nel contesto della definizione di una costante stringa con *double quotes* e ne viene usato il valore scritto come stringa di caratteri. Quindi il tipo di `<tt>$s</tt>` è `<tt><?php echo gettype($s); ?></tt>` e il suo valore è `<tt><?php echo $s; ?></tt>`.

Se si utilizza la variabile `<tt>$s</tt>` in un contesto numerico essa viene convertita con scansione delle cifre decimali da sinistra a destra. Si può vedere come avviene la conversione facendo un'assegnazione con cast `<tt>$yav = (int)$s</tt>`
`<?php $yav = (int)$s; ?>` ottenendo che `<tt>$yav</tt>` è di tipo `<tt><?php echo gettype($yav); ?></tt>` con valore `<tt><?php echo $yav; ?></tt>`. Come si può vedere la conversione non implica fare una valutazione di un'eventuale espressione contenuta nella stringa.

Se si esegue l'assegnazione `$s = "2+$var"` la variabile `$var` si trova nel contesto della definizione di una costante stringa con *double quotes* e ne viene usato il valore scritto come stringa di caratteri. Quindi il tipo di `$s` è `string` e il suo valore è `2+59`.

Se si utilizza la variabile `$s` in un contesto numerico essa viene convertita con scansione delle cifre decimali da sinistra a destra. Si può vedere come avviene la conversione facendo un'assegnazione con cast `$yav = (int)$s` ottenendo che `$yav` è di tipo `integer` con valore `2`. Come si può vedere la conversione non implica fare una valutazione di un'eventuale espressione contenuta nella stringa.

Costanti stringa

- ▶ Le stringhe sono sequenze di caratteri di 1 byte
 - ▶ Il supporto nativo permette di usare solo codifiche di caratteri su byte e non c'è supporto diretto a Unicode
 - ▶ Questo non significa che non si possono mettere codifiche Unicode nelle stringhe, ma che le operazioni sulle stringhe non vedono come un solo carattere i codici multi-byte
 - ▶ Una stringa costante può essere specificata con 4 diversi metodi
 - ▶ **double quoted** `"questa è una stringa"` – le variabili che compaiono all'interno della definizione sono espande nel loro valore e sono convertite le sequenze di escape (`\n`, `\t`,...)
 - ▶ **single quoted** `'anche questa è una stringa'` – le variabili non sono espande e solo le sequenze di escape `\'` e `\\` sono convertite nel carattere corrispondente
 - ▶ **heredoc** `<<<LABEL ... LABEL;` simile a double quoted
 - ▶ **newdoc** `<<<'LABEL' ... LABEL;` simile a single quoted

Costanti stringa: esempio

Una stringa è una sequenza di caratteri codificati su un byte. Una stringa può essere definita usando i doppi apici (*double quoted*) come in `<tt>$str = "Questa è una stringa";</tt><?php $str = "Questa è una stringa"; ?>` che si stampa in `<tt><?php echo $str; ?></tt>` ed è di lunghezza `<tt><?php echo strlen($str); ?></tt>`. Il fatto che la codifica è a 8 bit si vede cercando di stampare il carattere nella posizione 7 che dovrebbe corrispondere a 'è' ma invece risulta `<tt><?php echo $str[7]; ?></tt>` che è il primo carattere della sua codifica Unicode che è su 2 byte.

Se si usa la definizione *single quoted* eventuali variabili o sequenze di escape non vengono convertite nel loro valore. Se si assegna `<tt>$cats=44</tt><?php $cats=44 ?>` e si stampa la stringa `<tt>'$cats cats\n'</tt>` si ottiene `<tt><?php echo '$cats cats\n'; ?></tt>` mentre con `<tt>"$cats cats\n"</tt>` si stampa `<tt><?php echo "$cats cats\n"; ?></tt>`.

In alcuni casi può essere necessario delimitare le variabili (o il loro nome) con parentesi graffe {} per separarle dal testo che deve comparire subito dopo senza caratteri di separazione. Ad esempio se si definiscono le variabili `<tt>$fruit="apple"; $showmany="four";</tt><?php $fruit="apple"; $showmany="four";?>` e si stampa la stringa `<tt>"I ate $showmany $fruits"</tt>` si ottiene `<tt><?php echo "I ate $showmany $fruits";?></tt>`, mentre con `<tt>"I ate $showmany ${fruit}s"</tt>` si ha la stampa corretta `<tt><?php echo "I ate $showmany ${fruit}s";?></tt>`.

Una stringa è una sequenza di caratteri codificati su un byte. Una stringa può essere definita usando i doppi apici (*double quoted*) come in `$str = "Questa è una stringa";` che si stampa in `Questa è una stringa` ed è di lunghezza 21. Il fatto che la codifica è a 8 bit si vede cercando di stampare il carattere nella posizione 7 che dovrebbe corrispondere a 'è' ma invece risulta ` ` che è il primo carattere della sua codifica Unicode che è su 2 byte.

Se si usa la definizione *single quoted* eventuali variabili o sequenze di escape non vengono convertite nel loro valore. Se si assegna `$cats=44` e si stampa la stringa `'$cats cats\n'` si ottiene `$cats cats\n` mentre con `"$cats cats\n"` si stampa `44 cats .`

In alcuni casi può essere necessario delimitare le variabili (o il loro nome) con parentesi graffe {} per separarle dal testo che deve comparire subito dopo senza caratteri di separazione. Ad esempio se si definiscono le variabili `$fruit="apple"; $showmany="four";` e si stampa la stringa `"I ate $showmany $fruits"` si ottiene `I ate four`, mentre con `"I ate $showmany ${fruit}s"` si ha la stampa corretta `I ate four apples.`

Operatori e funzioni su stringhe

- ▶ Due stringhe possono essere concatenate con l'operatore .
- ▶ Sono disponibili numerose funzioni utili per la manipolazione di stringhe e per la ricerca di sottostringhe con espressioni regolari ([perl compatible RE](#))

```
<?php
    $s = '=?utf-8?Q?perch=C3=A9 =C3=A8 una prova (utf-8)?=';
    $parts = explode('?', $s);
?>

<head>
<title>Quoted printable decoding</title>
<meta http-equiv="Content-Type"
      content="text/html; charset=?php echo $parts[1];?>" />
</head>
<body>
    <?php echo 'Input string: <em>'.$s.'</em>'; ?><br />
    The converted string is <em>
        <?php echo quoted_printable_decode($parts[3]); ?>
    </em><br />
</body>
</html>
```

```
<html>
<head>
  <title>Quoted printable decoding</title>
  <meta http-equiv="Content-Type"
        content="text/html; charset=utf-8" />
</head>
<body>
  Input string: <em>
    =?utf-8?Q?perch=C3=A9 =C3=A8 una prova (utf-8)?=
  </em><br />
  The converted string is <em>
    perché è una prova (utf-8)</em><br />
</body>
</html>
```

```
Input string: =?utf-8?Q?perch=C3=A9 =C3=A8 una prova (utf-8)?=
The converted string is perché è una prova (utf-8)
```

Array

- ▶ In PHP gli array sono tabelle associative (map)
 - ▶ Un array crea un'associazione **key => value**
 - ▶ Gli elementi di un array possono essere di qualsiasi tipo; se sono a loro volta array si creano array multidimensionali (es. matrici)
 - ▶ La chiave può essere un integer o una string
 - ▶ Un array può essere creato col costrutto `array()` specificando come argomento (opzionale) la lista degli elementi (dalla versione 5.4 si può usare anche la definizione compatta con `[...]`)

```
$Lista_Spesa = array("Pane"=>0.5, "Latte"=>1,"Biscotti"=>3);

if(isset($_REQUEST['cosa'])&&isset($_REQUEST['quanto']))
    $Lista_Spesa[$_REQUEST['cosa']] += $_REQUEST['quanto'];

foreach ($Lista_Spesa as $cosa => $quanto)
    print "Comprare: $quanto di $cosa <br />\n";

if(isset($Lista_Spesa['Cioccolata']))
    print "<br />Cioccolata: {$Lista_Spesa['Cioccolata']} tavolette";
else
    print "<br />Cioccolata: non comprare!!";
```

127.0.0.1/~maggini/php/06-array.php?cosa=Cioccolata&quanto=2

La lista della spesa

Comprare: 0.5 di Pane
Comprare: 1 di Latte
Comprare: 3 di Biscotti
Comprare: 2 di Cioccolata

Cioccolata: 2 tavolette

Array: indici e funzioni

- ▶ Gli indici (chiavi) di un array possono essere numeri interi o stringhe
 - ▶ Possono coesistere insieme
 - ▶ Se si aggiungono valori ad un array senza specificare un valore per la chiave, viene utilizzato l'intero successivo corrispondente all'ultima chiave numerica inserita (eventualmente 0)
 - ▶ Si accede ad un elemento di un array con l'indice specificato in [..]
 - ▶ Per rimuovere una coppia **key=>value** da un array si può utilizzare il costrutto `unset($arr['key'])`
- ▶ Esiste un'ampia libreria di funzioni e costrutti per manipolare array (**count**, **sort**, **array_diff**, **array_fill**..)

```
$impegni[] = array('month'=>'Jan', 'day'=>10,  
                  'hour'=>12, 'desc'=>'Lecture');  
$impegni[] = array('month'=>'Feb', 'day'=>5, 'hour'=>14,  
                  'desc'=>'Conference');  
$impegni[] = array('month'=>'Mar', 'day'=>22, 'desc'=>'Birthday');  
  
for($i=0;$i<count($impegni);$i++) {  
    print "$i. {$impegni[$i]['month']} {$impegni[$i]['day']}";  
    if(isset($impegni[$i]['hour']))  
        print " at {$impegni[$i]['hour']}";  
    print ": <em>{$impegni[$i]['desc']}</em><br />";  
}
```

La lista degli impegni

0. Jan 10 at 12: *Lecture*
1. Feb 5 at 14: *Conference*
2. Mar 22: *Birthday*

Variabili predefinite e “superglobali”

- ▶ Il PHP definisce un ampio numero di variabili predefinite
 - ▶ Permettono di riferirsi a dati esterni (es. le variabili provenienti da un form HTML), all'ambiente interno PHP, ai messaggi di errore (es. `$php_errormsg`)
 - ▶ Le variabili superglobali sono array associativi accessibili da ogni contesto in uno script PHP
 - ▶ **`$_SERVER[ ]`**: informazioni relative al server su cui è eseguito lo script
 - ▶ **`$_ENV[ ]`**: variabili di ambiente di esecuzione dell'interprete PHP
 - ▶ **`$_COOKIE[ ]`**: informazioni sui cookies nella richiesta HTTP
 - ▶ **`$_GET[ ]`**: informazioni relative ad una richiesta con metodo GET
 - ▶ **`$_POST[ ]`**: informazioni relative ad una richiesta con metodo POST
 - ▶ **`$_REQUEST[ ]`**: è l'insieme di `$_COOKIE`, `$_GET` e `$_POST`
 - ▶ **`$_SESSION[ ]`**: contiene le variabili registrate per la sessione corrente
 - ▶ **`$_FILES[ ]`**: contiene gli oggetti passati con il metodo POST

La variabile \$_SERVER

- ▶ La variabile **\$_SERVER** permette di ottenere informazioni su
 - ▶ Le intestazioni della richiesta HTTP (**HTTP_USER_AGENT**, **HTTP_ACCEPT_LANGUAGE**, **HTTP_HOST**,...)
 - ▶ Sulla connessione col client (**REMOTE_ADDR**, **REMOTE_PORT**)
 - ▶ Sul server (**SERVER_SOFTWARE**, **DOCUMENT_ROOT**,...)

```
$vars = array('SERVER'=>$_SERVER,  
    'ENV'=>$_ENV,  
    'COOKIE'=>$_COOKIE,  
    'GET'=>$_GET,  
    'POST'=>$_POST,  
    'REQUEST'=>$_REQUEST,  
    'FILES'=>$_FILES,  
    'SESSION'=>$_SESSION);  
  
foreach($vars as $vartype=>$var)  
    echo "<a href=\"#{$_$vartype}\">\$_$vartype</a><br />";  
  
foreach($vars as $vartype=>$var) {  
    echo "<h3>Elenco \$_$vartype</h3>";  
    echo "<a name=_{$_$vartype}></a>";  
  
    foreach($var as $element=>$val)  
        echo "<b>$element</b>: $val<br />";  
}
```

[\\$_SERVER](#)
[\\$_ENV](#)
[\\$_COOKIE](#)
[\\$_GET](#)
[\\$_POST](#)
[\\$_REQUEST](#)
[\\$_FILES](#)
[\\$_SESSION](#)

Elenco \$_SERVER

HTTP_HOST: 127.0.0.1
HTTP_USER_AGENT: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.7; rv:11.0) Gecko/20100101 Firefox/11.0
HTTP_ACCEPT: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
HTTP_ACCEPT_LANGUAGE: it,en-us;q=0.7,en;q=0.3
HTTP_ACCEPT_ENCODING: gzip, deflate
HTTP_CONNECTION: keep-alive
HTTP_REFERER: http://127.0.0.1/~maggini/php/
HTTP_COOKIE: PHPSESSID=aq2klv3qsg7baqsh9ogth94k2
PATH: /usr/bin:/bin:/usr/sbin:/sbin
SERVER_SIGNATURE:
SERVER_SOFTWARE: Apache/2.2.21 (Unix) mod_ssl/2.2.21 OpenSSL/0.9.8r DAV/2 PHP/5.3.8 with Suhosin-Patch
SERVER_NAME: 127.0.0.1
SERVER_ADDR: 127.0.0.1
SERVER_PORT: 80
REMOTE_ADDR: 127.0.0.1
DOCUMENT_ROOT: /Library/WebServer/Documents
SERVER_ADMIN: you@example.com
SCRIPT_FILENAME: /Users/maggini/Sites/php/07-superglobals.php
REMOTE_PORT: 52981
GATEWAY_INTERFACE: CGI/1.1
SERVER_PROTOCOL: HTTP/1.1
REQUEST_METHOD: GET
QUERY_STRING:
REQUEST_URI: /~maggini/php/07-superglobals.php
SCRIPT_NAME: /~maggini/php/07-superglobals.php
PHP_SELF: /~maggini/php/07-superglobals.php
REQUEST_TIME: 1334241634

Accesso ai dati inviati con FORM HTML

- ▶ Gli array **`$_GET`**, **`$_POST`** e **`$_REQUEST`** permettono di accedere direttamente ai valori delle variabili inviati con un form
 - ▶ L'indice è la stringa corrispondente al nome della variabile associata all'elemento di input nel form
 - ▶ Il valore è una stringa corrispondente al valore inserito o selezionato
 - ▶ L'array in cui è inserita la variabile dipende dal metodo indicato nel form (GET o POST)
 - ▶ La variabile è sempre accessibile nell'array **`$_REQUEST`** per cui facendo riferimento ad esso si rende il codice PHP indipendente dalla scelta fatta nel form

```
<form action="07-superglobals.php" method="get">
  Prodotto<input name="prodotto" size=20><br />
  Acquisto<input type="checkbox" name="compra"><br />
  <input type="submit" value="Invia">
</form>
```

FORM HTML

Prodotto

Acquisto ☒

QUERY_STRING: prodotto=prova&compra=on
REQUEST_URI: /~maggini/php/07-superglobals.php?prodotto=prova&compra=on

Elenco **`$_GET`**

prodotto: prova
compra: on

`$_GET[ 'prodotto' ]`
`$_GET[ 'compra' ]`

Elenco **`$_REQUEST`**

prodotto: prova
compra: on

`$_REQUEST[ 'prodotto' ]`
`$_REQUEST[ 'compra' ]`

Invio dati con POST

```
<form action="07-superglobals.php" method="post">
  Prodotto<input name="prodotto" size=20><br />
  Acquisto<input type="checkbox" name="compra"><br />
  <input type="submit" value="Invia">
</form>
```

Prodotto

Acquisto ☒

FORM HTML
x-www-form-urlencoded

Elenco \$_POST

prodotto: prova `$_POST['prodotto']`
compra: on `$_POST['compra']`

Elenco \$_REQUEST

prodotto: prova `$_REQUEST['prodotto']`
compra: on `$_REQUEST['compra']`

REQUEST_METHOD: POST

QUERY_STRING:

REQUEST_URI: /~maggini/php/07-superglobals.php

CONTENT_TYPE: application/x-www-form-urlencoded

CONTENT_LENGTH: 24

```
<form action="07-superglobals.php"
  enctype="multipart/form-data" method="post">
  Nome<input type="text" name="nome"><br />
  File<input type="file" name="file"><br />
  <input type="submit" value="Invia">
</form>
```

Nome

File

FORM HTML
multipart/form-data

Elenco \$_POST

nome: marco `$_POST['nome']`

Elenco \$_REQUEST

nome: marco `$_REQUEST['nome']`

Elenco \$_FILES

file: Array `$_FILES['file']`

CONTENT_TYPE: multipart/form-data; boundary=-----168072824752491622650073

CONTENT_LENGTH: 16232

REQUEST_METHOD: POST

QUERY_STRING:

REQUEST_URI: /~maggini/php/07-superglobals.php

Upload di file

- ▶ La variabile superglobale **`$_FILES`** contiene informazioni su file inviati con form HTML attraverso l'input di tipo **`file`**
 - ▶ Ad ogni file inviato col form è associato un array **`$_FILES['nome']`**
 - ▶ L'array contiene il nome, il tipo, la dimensione, il nome del file temporaneo sul server e il codice di errore (0 per nessun errore)
 - ▶ Il file temporaneo può essere copiato nella destinazione finale con la funzione **`move_uploaded_file($tmp_file, $destination)`**

```
Name: <?php echo $_FILES['file']['name']; ?><br />
Type: <?php echo $_FILES['file']['type']; ?><br />
Size: <?php echo $_FILES['file']['size']; ?><br />
Tmp_name: <?php echo $_FILES['file']['tmp_name']; ?><br />
Error: <?php echo $_FILES['file']['error']; ?><br />

<?php
if($_FILES['file']['error']==0) {
    $upload_dir = dirname($_SERVER['SCRIPT_FILENAME'])
                ."/upload/";
    $filename = $upload_dir.basename($_FILES['file']['name']);

    if(move_uploaded_file($_FILES['file']['tmp_name'], $filename))
        echo "File uploaded successfully<br />";
    else
        echo "Error: File not uploaded!<br />";
} ?>
```

Uploaded file info

Name: Untitled.tiff
Type: image/tiff
Size: 15910
Tmp_name: /private/var/tmp/phpL3FFnY
Error: 0
File uploaded successfully

Form con selezione multipla

- ▶ L'elemento di input **<select multiple="yes">** permette all'utente di selezionare più opzioni
 - ▶ Il problema è che tutti i valori selezionati sono passati con lo stesso nome di variabile
 - ▶ La soluzione è usare la funzione *"array da elemento di form"*
<select multiple="yes" name="var[]">
 - ▶ Il PHP tratta l'elemento come array e aggiunge un elemento per ogni valore selezionato con indici 0,1,...

```
<form action="11-multiple-select.php">
<select name="genere[]" multiple="true" size=6>
  <option>Heavy Metal</option>
  <option>Death Metal</option>
  <option>Gothic Metal</option>
  <option>Power Metal</option>
  <option>Hard Rock</option>
  <option>Progressive Rock</option>
</select>
<input type=submit>
</form>
```

```
foreach($_REQUEST['genere'] as $genere) {
  echo "<h3>$genere</h3>";

  $sugg = $gruppi[$genere];
  foreach($sugg as $gruppo)
    echo "$gruppo <br />";
}
```

Suggerimenti per i generi musicali preferiti

Death Metal

Obituary
Sarcophago
Six Feet Under

Power Metal

hammerfall
Iced Earth
Helloween
Nightwish

Heavy Metal
Death Metal
Gothic Metal
Power Metal
Hard Rock
Progressive Rock

☐ 127.0.0.1/~maggini/php/11-multiple-select.php?genere[]=Death+Metal&genere[]=Power+Metal

Submit Query

Campo di visibilità delle variabili

- ▶ Il **campo di visibilità** di una variabile (**scope**) è il contesto in cui è definita
 - ▶ comprende anche file inclusi sia con la direttiva **include** che **require**
 - ▶ All'interno delle funzioni è definito un campo di visibilità locale
 - ▶ ogni variabile usata all'interno di una funzione è per default limitata all'interno del corpo della funzione
 - ▶ le variabili definite all'esterno non sono accessibili nella funzione a meno di non dichiararle **global** o di usare l'array predefinito **\$GLOBALS**

```
$pi = 3.14;  
  
function circle($r) {  
    return 2*$pi*$r;  
}  
  
function circleG($r) {  
    global $pi;  
    return 2*$pi*$r;  
}  
  
function circleGA($r) {  
    return 2*$GLOBALS['pi']*$r;  
}
```

Global variables in function scope

\$pi is local: 0 `circle(1)`

\$pi is declared global: 6.28 `circleG(1)`

\$pi is from \$GLOBALS array: 6.28 `circleGA(1)`

Operatori 1

▶ Operatori Aritmetici

- ▶ $\$a + \b , $\$a - \b , $\$a * \b , $\$a / \b , $\$a \% \b (modulo), $-\$a$, $\$a . \b (concatenazione di stringhe)

▶ Operatori di assegnazione

- ▶ $\$a = \b , $\$a += \b , $\$a -= \b , $\$a *= \b , $\$a /= \b , $\$a \% = \b , $\$a . = \b ($\$a = \$a . \$b$)

▶ Operatori di autoincremento/autodecremento

- ▶ pre $++\$a$, $--\$a$; post $\$a++$, $\$a--$

▶ Operatori di comparazione

- ▶ Producono un valore boolean (true/false)
- ▶ $\$a == \b (uguale come valore), $\$a === \b (uguale come valore e tipo), $\$a != \b e $\$a <> \b (diverso), $\$a !== \b (non identico), $\$a > \b , $\$a < \b , $\$a >= \b , $\$a <= \b
- ▶ Gli operatori $>$, $<$ con le stringhe tengono conto dell'ordinamento alfabetico ("**cane**" $>$ "**canile**" è **false** e "**cane**" $<$ "**canile**" è **true**)

Operatori 2

► Operatori logici

- `$a and $b` o `$a && $b` (AND logico), `$a or $b` o `$a || $b` (OR logico), `$a xor $b` (XOR logico), `!$a` (NOT logico)

► Operatori su array

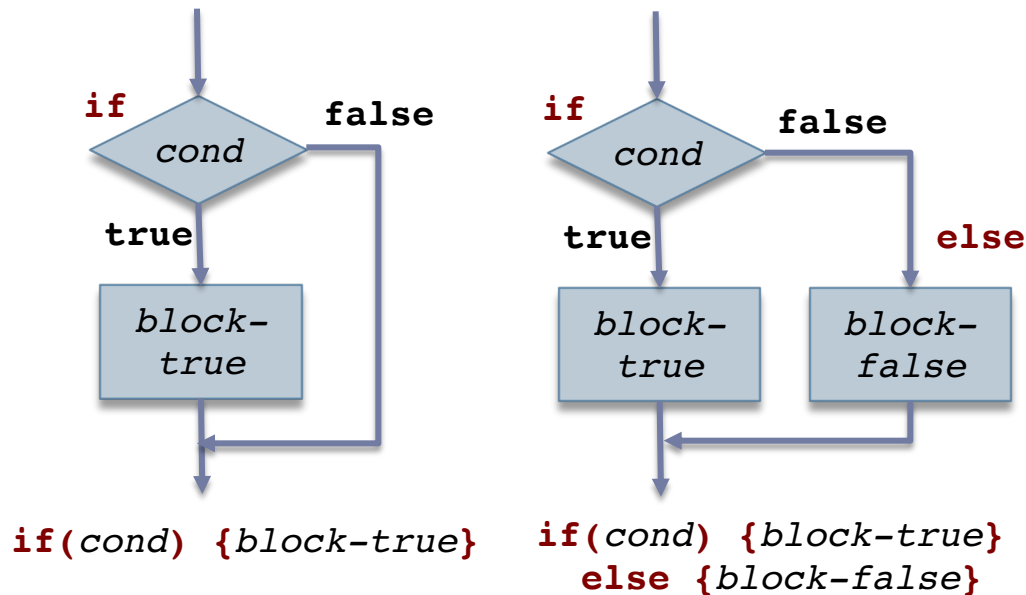
- `$a + $b` (unione), `$a == $b` (`$a` e `$b` contengono le stesse coppie chiave/valore), `$a === $b` (`$a` e `$b` contengono le stesse coppie chiave/valore nello stesso ordine e degli stessi tipi), `$a != $b` o `$a <> $b` (disuguaglianza), `$a !== $b` (non identità)

► Operatori su bit

- `$a & $b` (AND bit a bit), `$a | $b` (OR bit a bit), `$a ^ $b` (XOR bit a bit), `~$a` (NOT bit a bit), `$a << $b` (SHIFT a sinistra di `$b` bit), `$a >> $b` (SHIFT a destra di `$b` bit)

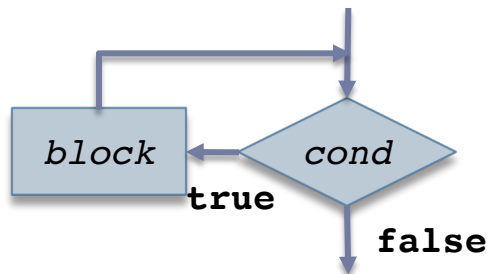
Strutture di controllo: if

- ▶ **if** (*cond*) {*block-true*} [**else** {*block-false*}]
 - ▶ *cond* è un'espressione che viene valutata come boolean
 - ▶ per **integer** (**float**) è **true** se il valore è diverso da **0** (**0.0**)
 - ▶ per **string** è **true** se la stringa non è la stringa vuota "" o "0"
 - ▶ per **array** è **true** se l'array contiene almeno un elemento

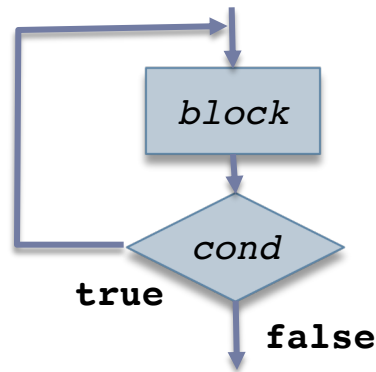


Strutture di controllo: cicli

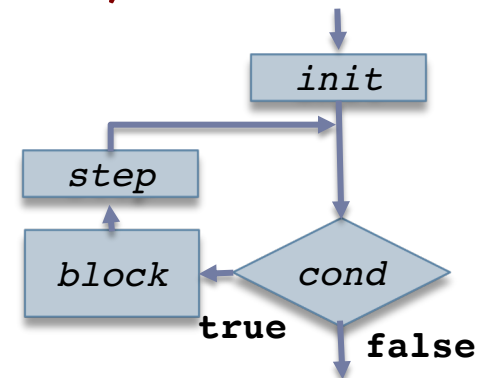
- ▶ **while** (*cond*) {*block*}
- ▶ La condizione è verificata all'inizio del ciclo per cui il blocco di istruzioni può anche non essere eseguito mai
- ▶ **do** {*block*} **while** (*cond*)
- ▶ La condizione è verificata alla fine del ciclo e il blocco di istruzioni è eseguito almeno una volta
- ▶ **for**(*init*; *cond*; *step*) {*block*}
- ▶ Le espressioni *init*, *cond* e *step* possono anche essere vuote o essere composte di più espressioni separate dal carattere ,



while(*cond*) {*block*}



do {*block*} **while**(*cond*)



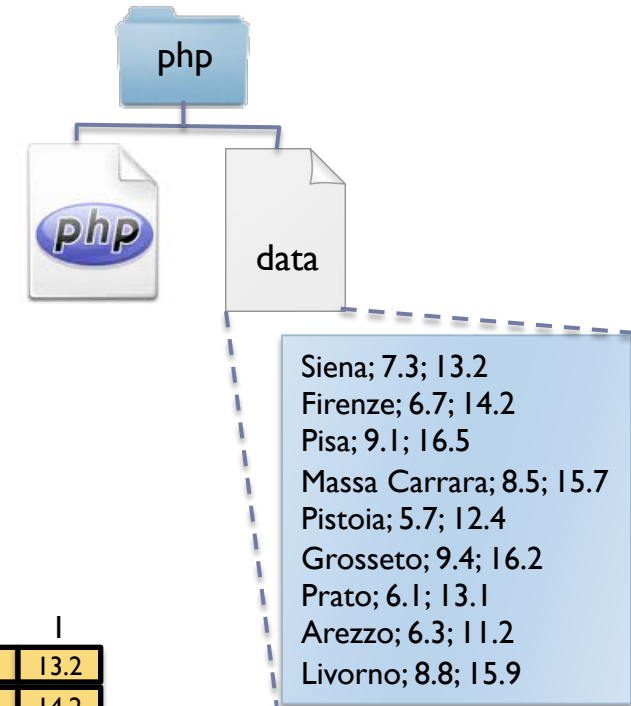
for(*init*; *cond*; *step*) {*block*}

Strutture di controllo: foreach, break

- ▶ **foreach**(\$arr as \$val) {block}
foreach(\$arr as \$key=>\$val) {block}
- ▶ funziona solo con array o oggetti
- ▶ effettua la scansione dell'array assegnando ad ogni iterazione l'elemento corrente alla variabile \$val o alla coppia (\$key,\$val)
- ▶ se si vuole modificare \$val nel ciclo occorre assegnarlo come riferimento usando l'operatore &\$val
- ▶ In un ciclo o struttura di controllo è possibile usare
 - ▶ **break;** per uscire dal blocco in cui ci si trova
 - ▶ **continue;** per saltare all'iterazione successiva del ciclo
 - ▶ in entrambi i casi si può specificare un intero *n* per indicare il livello di ciclo a cui saltare nel caso di più cicli annidati

Strutture di controllo: esempio 1

```
<?php
$file = @fopen('data','r');
if(!$file) { ?>
    <h2>Error: cannot open data file</h2>
    </body>
    </html> <?php
    die();
}
while($line=fgets($file)) {
    $data = explode(';',$line);
    if(count($data)!=3)
        continue;
    $temp[$data[0]] = array($data[1],$data[2]);
}
ksort($temp);
fclose($file);
?>
```



`$temp[ ][ ]`

		0	1
Siena	=>	7.3	13.2
Firenze	=>	6.7	14.2
Pisa	=>	9.1	16.5
Massa Carrara	=>	8.5	15.7
Pistoia	=>	5.7	12.4
Grosseto	=>	9.4	16.2
Prato	=>	6.1	13.1
Arezzo	=>	6.3	11.2
Livorno	=>	8.8	15.9

Strutture di controllo: esempio 1

```
<h2>Temperature in Toscana</h2>
<table>
<tr><th style="text-align: left">Città</th>
<th style="text-align: center">Tmin</th>
<th style="text-align: center">Tmax</th></tr>
<?php
    $minTmin = 100;
    $maxTmax = -100;

    foreach($temp as $citta=>$minmax){
        list($Tmin,$Tmax)=$minmax;
        echo "<tr><td>$citta</td><td>$Tmin</td><td>$Tmax</td></tr>";

        if($Tmin<$minTmin) {
            $minTmin = $Tmin;
            $minCitta = $citta;
        }
        if($Tmax>$maxTmax) {
            $maxTmax = $Tmax;
            $maxCitta = $citta;
        }
    }
?>
</table>
<p>La città più fredda è Pistoia ( 5.7)
    <?php echo "$minCitta ($minTmin)";?><br />
    La città più calda è Pisa ( 16.5 )
    <?php echo "$maxCitta ($maxTmax)";?></p>
```

Temperature in Toscana

Città	Tmin	Tmax
Arezzo	6.3	11.2
Firenze	6.7	14.2
Grosseto	9.4	16.2
Livorno	8.8	15.9
Massa Carrara	8.5	15.7
Pisa	9.1	16.5
Pistoia	5.7	12.4
Prato	6.1	13.1
Siena	7.3	13.2

La città più fredda è Pistoia (5.7)

La città più calda è Pisa (16.5)

Strutture di controllo: switch

- ▶ **switch(\$v) {case val: default: }**
 - ▶ Funziona anche se \$v è una stringa (i casi val sono stringhe)
 - ▶ L'esecuzione passa all'istruzione corrispondente al **case** che fa match con il valore di \$v e poi le istruzioni sono eseguite in sequenza
 - ▶ Se si vuole che ogni case sia un blocco in alternativa, occorre usare **break** come ultima istruzione per interrompere l'esecuzione in sequenza

```
switch($1) {  
  case 'it':  
    $msg="Preferisci l'Italiano con valore ";  
    break;  
  case 'en-us':  
    $msg="You prefer English American with score ";  
    break;  
  case 'en':  
    $msg="You prefer English with score ";  
    break;  
  default:  
    $msg="$1 is an unknown language specification";  
}
```

Parsing Browser's Accept-Language

Input string: *it,en-us;q=0.8,en;q=0.5,fr;q=0.3*

Preferisci l'Italiano con valore 1
You prefer English American with score 0.8
You prefer English with score 0.5
fr is an unknown language specification 0.3

inclusione di file

- ▶ **include, require, include_once, require_once**
 - ▶ Questi costrutti permettono di includere il contenuto di un file
 - ▶ Quando si inizia l'inclusione l'interprete esce dalla modalità di esecuzione e codice PHP nel file incluso deve essere fra **<?php ?>**
 - ▶ Se il nome del file incluso non contiene un *path* viene ricercato nella directory dello script corrente e nelle directory specificate nella configurazione PHP con l'opzione **include_path**
 - ▶ Il file eredita il campo di visibilità delle variabili del contesto in cui si trova la linea **include/require**
 - ▶ Variabili e funzioni definite nel file incluso hanno visibilità che dipende dal contesto in cui è incluso
 - ▶ La differenza fra **include** e **require** è che se non si trova il file **include** genera un *warning* mentre **require** un *errore fatale*
 - ▶ Le versioni con **_once** includono il file una volta sola, anche se è argomento di più costrutti che vengono eseguiti

Include: esempio 1

```
<?php
    error_reporting(E_ALL);
    ini_set("display_errors", 1);
    require '14-functions.inc';
?>
<html>
<head>
<title>Esempio di include</title>
</head>
<body>
<?php
    $content = @$_REQUEST['content'];
    if(empty($content)) { ?>
        <h2>Error: no content selected</h2>
        </body></html>
    <?php die(); }

    ?>
<h2> <?php echo include_title($content); ?></h2>
<div style="text-align: justify;
    border: 5px solid DodgerBlue;padding:5px">
<?php include_content($content); ?></div>
<em>Last modified: </em><?php echo @$last_modified ?>
</body>
</html>
```

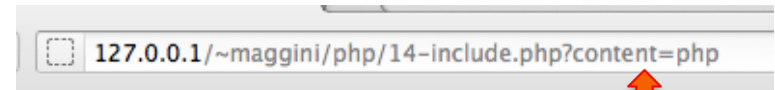
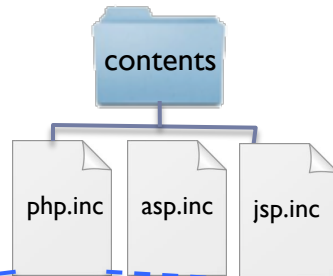
```
<?php
$contentdir = "../14-contents/";
$index = array(
    "php"=>array("titolo"=>"PHP language", "file"=>"php"),
    "asp"=>array("titolo"=>"ASP language", "file"=>"asp"),
    "jsp"=>array("titolo"=>"JSP language", "file"=>"jsp"));

function include_title($id) {
    global $index;
    if(!empty($index[$id]['titolo']))
        return $index[$id]['titolo'];
    else
        return "Contenuto non disponibile";
}

function include_content($id) {
    global $index, $contentdir, $last_modified;
    if(!empty($index[$id]['file']))
        include "{$contentdir}{$index[$id]['file']}.inc";
    else
        echo "Contenuto non disponibile";
} ?>
```

14-functions.inc

include: esempio 2



```
<?php $last_modified = "Apr 17, 2012 12:45 am"; ?>
```

`<p>PHP (acronimo ricorsivo di "PHP: Hypertext Preprocessor", preprocessore di ipertesti; originariamente acronimo di "Personal Home Page"[1]) è un linguaggio di scripting interpretato, con licenza open source e libera (ma incompatibile con la GPL), originariamente concepito per la programmazione Web ovvero la realizzazione di pagine web dinamiche.</p>`

`<p>Attualmente è utilizzato principalmente per sviluppare applicazioni web lato server ma può essere usato anche per scrivere script a riga di comando o applicazioni standalone con interfaccia grafica.</p>`

`<p>L'elaborazione di codice PHP sul server produce codice HTML da inviare al browser dell'utente che ne fa richiesta. Il vantaggio dell'uso di PHP e degli altri linguaggi Web come ASP e .NET rispetto al classico HTML derivano dalle differenze profonde che sussistono tra Web dinamico e Web statico.</p>`

PHP language

PHP (acronimo ricorsivo di "PHP: Hypertext Preprocessor", preprocessore di ipertesti; originariamente acronimo di "Personal Home Page"[1]) è un linguaggio di scripting interpretato, con licenza open source e libera (ma incompatibile con la GPL), originariamente concepito per la programmazione Web ovvero la realizzazione di pagine web dinamiche.

Attualmente è utilizzato principalmente per sviluppare applicazioni web lato server ma può essere usato anche per scrivere script a riga di comando o applicazioni standalone con interfaccia grafica.

L'elaborazione di codice PHP sul server produce codice HTML da inviare al browser dell'utente che ne fa richiesta. Il vantaggio dell'uso di PHP e degli altri linguaggi Web come ASP e .NET rispetto al classico HTML derivano dalle differenze profonde che sussistono tra Web dinamico e Web statico.

Last modified: Apr 17, 2012 12:45 am

./contents/php.inc

Funzioni

- ▶ Il PHP permette di definire funzioni specificando
 - ▶ il nome della funzione
 - ▶ la lista dei parametri
 - ▶ è supportato il passaggio per valore (default), per riferimento (usando l'operatore &), i valori di default e un numero variabile di parametri
 - ▶ col passaggio per valore l'argomento è copiato e quindi ogni modifica non si riflette nella variabile passata come parametro
 - ▶ il codice eseguito dalla funzione
 - ▶ La funzione può produrre un valore con l'istruzione **return** *expr*;

```
function mymin($arr) {  
    if(!is_array($arr))  
        return $arr;  
  
    reset($arr);  
    $minval = current($arr);  
    foreach($arr as $value) {  
        if($value<$minval)  
            $minval = $value;  
    }  
    return $minval;  
}
```

```
$c = array("a"=>1, "b"=>2, "c"=>3, "d"=>4, "e"=>5);  
$m = mymin($c);
```

Il minimo del vettore ["a"=>1, "b"=>2, "c"=>3, "d"=>4, "e"=>5] è 1

Classi e oggetti (cenni)

- ▶ E' supportata la definizione di classi
 - ▶ Una classe descrive un insieme di oggetti caratterizzati da **proprietà** (variabili e costanti) e **metodi** (funzioni)
 - ▶ Un oggetto di una classe viene creato con l'operatore **new** che produce un **handle** (riferimento) all'oggetto creato
 - ▶ Dato un handle di un oggetto si possono eseguire i metodi su tale oggetto o accedere alle sue proprietà (se public)

```
class lamp {  
    public $power;  
    private $state="off";  
  
    function __construct($pwr)  
    {  
        $this->power=$pwr;  
    }  
    public function turnon()  
    {  
        $this->state="on";  
    }  
    public function turnoff()  
    {  
        $this->state="off";  
    }  
    public function getstate()  
    {  
        return $this->state;  
    }  
}
```

proprietà

metodi

```
$lamp75 = new lamp(75);  
echo $lamp75->getstate()."<br />";  
$lamp75->turnon();  
echo $lamp75->getstate()."<br />";  
echo $lamp75->power."<br />";  
echo $lamp75->state."<br />";
```

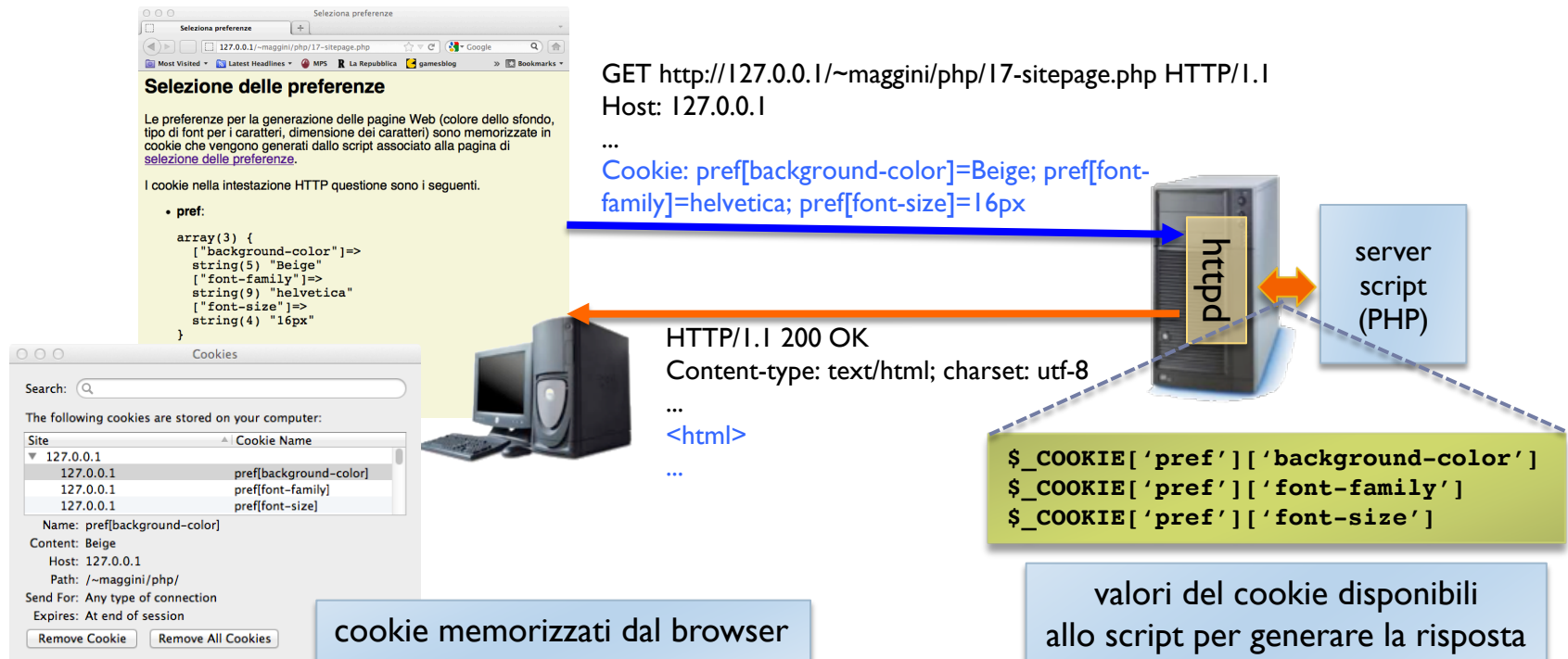
```
off  
on  
75  
Fatal error: Cannot access private property
```

Cookies

- ▶ I **cookie** sono un meccanismo per memorizzare dati su un browser da parte di un server
 - ▶ Sono un supporto per **creare uno stato nelle transazioni HTTP**, ovvero permettono di creare una storia delle azioni effettuate da un browser su un certo server Web
 - ▶ Devono essere abilitati e supportati dal browser
 - ▶ Un cookie è una variabile con associati dei dati che viene inizializzata e manipolata dal server Web e **memorizzata sul browser** in seguito a una transazione HTTP
 - ▶ Alla prima connessione il server Web invia la variabile nell'header HTTP creando il cookie sul browser (se i cookie sono abilitati)
 - ▶ Il browser memorizza la variabile in una tabella interna con il riferimento al server Web che l'ha inviata
 - ▶ Tutte le volte che il browser contatta il server Web invia nell'header della richiesta il cookie con la variabile e il suo valore attuale
 - ▶ Il server Web accede alla variabile ed eventualmente invia un aggiornamento nella risposta HTTP

Uso dei cookie

- ▶ I cookie sono dati e non codice eseguito
 - ▶ Non possono essere utilizzati per creare virus o malware
 - ▶ Possono essere utilizzati per tracciare le azioni degli utenti con possibili minacce alla privacy (tracking cookies - third-party tracking cookies)
 - ▶ L'ultima specifica è nell'RFC 6265 (2011) che aggiorna i precedenti



Uso dei cookie

► Gestione di sessioni

- Il cookie può essere utilizzato per memorizzare la storia di più richieste allo stesso sito fra loro correlate
- I cookie sono stati introdotti (1994) come meccanismo per poter gestire un carrello della spesa (shopping cart)
- Con le soluzioni attuali il cookie non contiene direttamente tutte le informazioni di stato ma solo un **ID univoco di sessione** che identifica un'area di memoria sul server ad ogni richiesta

► Personalizzazione

- Un cookie può essere utilizzato per memorizzare delle preferenze dell'utente per l'accesso ad un dato sito

► Tracking

- Possono essere utilizzati per tracciare le abitudini di navigazione degli utenti
- un cookie contenente un ID univoco viene inviato all'utente alla prima connessione al sito e da quel momento è possibile creare un log delle pagine visitate
- I **third-party cookies** sono cookie generati da siti diversi da quello visualizzato nella barra del browser per il fatto che la pagina caricata contiene un elemento proveniente dal sito del cookie (es. un banner pubblicitario)
 - Il third-party può di fatto tracciare tutti i siti visitati dall'utente che contengono i propri banner
 - Si può configurare il browser in modo che non li accetti

Cookie e HTTP

- ▶ I cookie sono inviati nelle intestazioni HTTP
 - ▶ Un server può creare o modificare un cookie su un browser inviando l'intestazione **Set-cookie:** nella quale si possono specificare (utilizzando il carattere ';' come separatore)
 - ▶ Più coppie **variabile=valore**
 - ▶ Il campo di visibilità del cookie sia come path (**path=/mydir/;**) che come dominio rispetto al server (**domain=.unisi.it;** per un cookie da www.dii.unisi.it)
 - ▶ Il tempo di vita (**expires=Sun, 22-Apr-2012 12:07:22 GMT;**) che indica il timestamp dopo il quale il browser deve eliminare il cookie
 - ▶ Le opzioni **Secure** (il cookie è inviato solo su connessioni crittate) e **HttpOnly** (il cookie è manipolabile solo via HTTP e non javascript)
 - ▶ Il client invia il cookie al server origine ad ogni richiesta con l'intestazione **Cookie:**
 - ▶ L'intestazione contiene i cookie come lista **variabile=valore;**

Cookie in PHP

- ▶ Si possono definire cookie con l'istruzione

```
setcookie($name,$val,$expire,$path,$domain,$secure,$httponly)
```

- ▶ dato che deve essere generata l'intestazione HTTP Set-Cookie l'istruzione deve essere eseguita prima di mandare in output il corpo della risposta
 - ▶ eventualmente si possono usare le funzioni di libreria per il buffering della risposta (`ob_start()`, `ob_flush()`)
- ▶ **\$name** è il nome della variabile associata al cookie e può essere usata anche per specificare un elemento di array
- ▶ **\$value** è il valore associato e viene inviato in forma *urlencoded*
- ▶ Gli altri parametri sono opzionali e si riferiscono alle opzioni possibili per la definizione di un cookie
- ▶ Uno script PHP può accedere ai cookie usando l'array associativo **\$_COOKIE**
 - ▶ Se il browser ha il cookie **'mycookie'** memorizzato la variabile **\$_COOKIE['mycookie']** è inizializzata al valore inviato dal browser

Esempio di uso dei cookie (set)

```
<form action="17-pref.php" method="post">
  Colore dello sfondo: <select name="background-color">
    <?php get_opts("background-color") ?>
  </select></br />
  Tipo di carattere: <select name="font-family">
    <?php get_opts("font-family") ?>
  </select></br />
  Dimensione del carattere: <select name="font-size">
    <?php get_opts("font-size") ?>
  </select></br />
  <input type="submit" value="Applica" /></br />
</form>
```

Seleziona le tue preferenze

Colore dello sfondo:

Tipo di carattere:

Dimensione del carattere:

[Pagina principale](#)

```
function form_get_prefs() {
    global $properties;
    $retval = false;
    foreach($properties as $prop=>$values) {
        $setprop = $_REQUEST[$prop];
        if(empty($setprop))
            continue;
        setcookie("pref[$prop]", "$setprop");
        $retval = true;
    }
    return $retval;
}
```

HTTP/1.1 200 OK

...
Set-Cookie: pref[background-color]=Beige; pref[font-family]=helvetica; pref[font-size]=16px
....

Esempio di uso dei cookie (get)

```
GET http://127.0.0.1/~maggini/php/17-sitepage.php HTTP/1.1
Host: 127.0.0.1
```

```
...
Cookie: pref[background-color]=Beige; pref[font-
family]=helvetica; pref[font-size]=16px
....
```

```
function get_style_pref() {
    global $properties;
    echo "<style>\n";
    foreach($properties as $prop=>$values) {
        $val = $_COOKIE['pref'][$prop];
        if(!empty($val))
            echo "body { $prop: $val }\n";
    }
    echo "</style>";
}
```

```
foreach($_COOKIE as $name=>$val) {
    echo "<li><b>$name</b>: <pre>";
    var_dump($val);
    echo "</pre></li>";
}
```

```
<head>
<title>Seleziona preferenze</title>
<style>
    body { background-color: Beige }
    body { font-family: helvetica }
    body { font-size: 16px }
</style>
</head>
```

Selezione delle preferenze

Le preferenze per la generazione delle pagine Web (colore dello sfondo, tipo di font per i caratteri, dimensione dei caratteri) sono memorizzate in cookie che vengono generati dallo script associato alla pagina di [selezione delle preferenze](#).

I cookie nella intestazione HTTP questione sono i seguenti.

- **pref:**

```
array(3) {
    ["background-color"]=>
    string(5) "Beige"
    ["font-family"]=>
    string(9) "helvetica"
    ["font-size"]=>
    string(4) "16px"
}
```

[Selezione preferenze](#)

Sessioni

- ▶ Il supporto PHP alle sessioni permette di mantenere dati **sul server** in una sequenza di richieste HTTP dallo stesso browser
 - ▶ Ad ogni utente (browser) è associato un identificatore univoco (**SESSION ID**)
 - ▶ Il session ID è scambiato in un cookie o nell'URL (variabile GET)
 - ▶ La sessione permette di memorizzare in modo persistente variabili nell'array **\$_SESSION[]**
 - ▶ Quando un browser accede al sito il PHP controlla automaticamente (se attivo) o su richiesta (**session_start()**) se è presente un session ID valido nella richiesta e in tal caso ripristina l'ambiente salvato in precedenza
 - ▶ Per default l'ambiente viene salvato in file in una directory sul server specificata nella configurazione

Creazione di una sessione

```
$user = trim($_REQUEST['username']);
$pass = trim($_REQUEST['password']);

if(!empty($user) && !empty($pass)) {
    if($Users[$user]==$pass) {
        session_start();
        $_SESSION['user'] = $user;

        header("Location: insidepage.php");
    } else {
        header("Location: login.php?error=yes");
    }
} else {...}
...}
```

PHPSESSID: string(26) "lgdvqlp7vn7pejihsrse4q7aq6"

```
session_start();
if(empty($_SESSION['user'])) {
    header("Location: 18-login.php");
} else {
    $_SESSION['insidecnt']++;
    $_SESSION['timestamps'][] =
        date('l jS \of F Y h:i:s A');
```

User Login

Username
Password

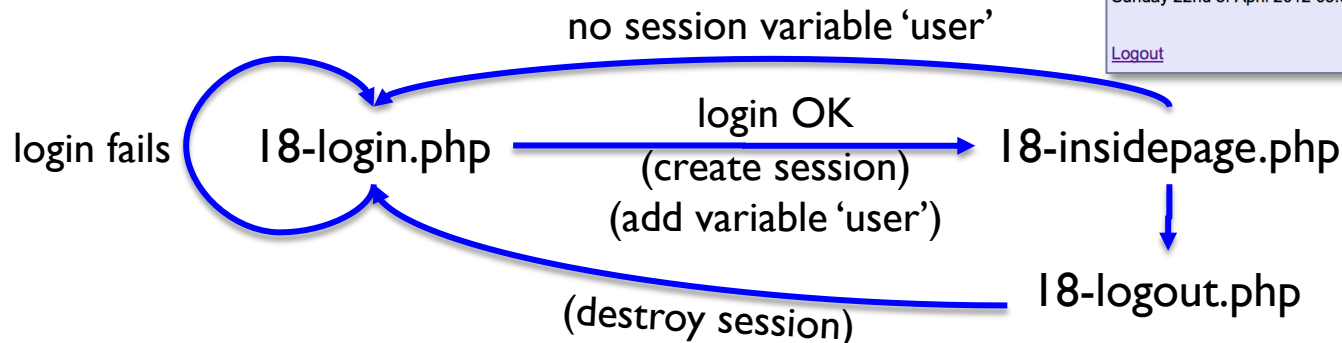
Welcome pippo

You reloaded 5 times this page in your current session

Sunday 22nd of April 2012 05:06:33 PM
Sunday 22nd of April 2012 05:06:48 PM
Sunday 22nd of April 2012 05:06:49 PM
Sunday 22nd of April 2012 05:06:50 PM
Sunday 22nd of April 2012 05:06:51 PM

[Logout](#)

SESSION
user
insidecnt
timestamps[]



Distruzione della sessione

- ▶ Per eliminare la sessione occorre
 - ▶ fare l'unset delle variabili di sessione
 - ▶ eliminare il cookie utilizzato per memorizzare il session ID
 - ▶ il nome predefinito del cookie si può ottenere con `session_name()`
 - ▶ il cookie si può eliminare specificando un timestamp di expire antecedente il timestamp attuale
 - ▶ liberare lo spazio sul server allocato per serializzare le variabili memorizzate nella sessione (`session_destroy()`)

```
session_start();
$_SESSION = array();
if (isset($_COOKIE[session_name()])) {
    setcookie(session_name(), '', time()-42000, '/');
}
session_destroy();
```

l8-logout.php

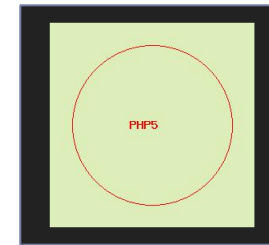
header()

- ▶ L'istruzione `header($head)` permette di inserire un campo di intestazione nella risposta HTTP
 - ▶ Deve essere eseguita prima che sia effettuato un qualsiasi output (anche uno spazio o un newline) in modo che `$head` sia correttamente nell'intestazione
 - ▶ Può essere usato per specificare varie informazioni come il tipo MIME della risposta (`Content-type:`), l'indirizzo di una redirection (`Location:`), il controllo della cache (`Cache-control:`)

```
header("Content-type: "
      . image_type_to_mime_type(IMGTYPE_JPEG));

$width = 256; $height = 256;
$txt = "PHP5";

$img = imagecreatetruecolor($width, $height);
$background = imagecolorallocate($img, 0xDD, 0xEE, 0xBB);
imagefill($img, 0, 0, $background);
$pencolor = imagecolorallocate($img, 0xFF, 0x00, 0x00);
imagearc($img, 128, 128, 200, 200, 0, 0, $pencolor);
imagestring($img, 5, 100, 120, $txt, $pencolor);
imagejpeg($img);
imagedestroy($img);
```



Content-type: image/jpeg



Content-type: text/plain
(web server default)

PHP e database

- ▶ IL PHP ha librerie per la connessione a specifici DBMS (es. MySQLi) per la gestione di dati da uno script
 - ▶ Connessione con autenticazione al DBMS utilizzando la modalità prevista dallo specifico DBMS specificando un database

```
mysqli_connect(server,user,pwd,database)
```

- ▶ Esecuzione di una query SQL

```
mysqli_query(connection, query)
```

- ▶ Accesso alle righe del risultato della query

```
mysqli_fetch_row(data), mysqli_fetch_assoc(data), ....
```

- ▶ Deallocazione delle risorse

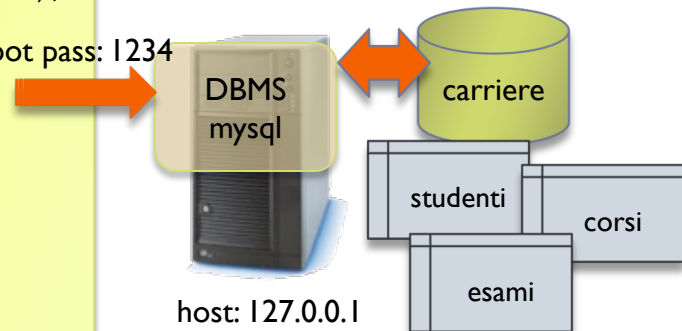
```
mysqli_free_result(data), mysqli_close(connection)
```


Esempio

```
$conn = mysqli_connect($dbmshost,$dbmsuser,$dbmypass,$dbname);
if(!$conn) {
    echo "<h1>Database connection problems</h1>"
    .mysqli_connect_error()."</body></html>";
    die();
}
$query = 'SELECT nome, cognome, titolo, voto
FROM studenti, corsi, esami
WHERE esami.matricola=studenti.matricola
      AND esami.corso=corsi.codice
ORDER BY cognome, nome, titolo';

$result = mysqli_query($conn,$query);
if(!$result) {
    echo "Database error: "
    .mysqli_error($conn)."</body></html>";
    mysqli_close($conn);
    die();
}
echo "<h2>Esami degli studenti</h2>";
echo "<table style=\"border: 2px solid gray;
      border-collapse: collapse\">\n";
while ($row = mysqli_fetch_row($result)) {
    echo "\t<tr>\n";
    foreach ($row as $col_value)
        echo "\t\t<td style=\"border: 1px solid black;\" >
            $col_value</td>\n";
    echo "\t</tr>\n";
}
echo "</table>\n";
mysqli_free_result($result);
mysqli_close($conn);
```

user: root pass: 1234



Esami degli studenti

giuseppe	mazzini	geografia	29
giuseppe	mazzini	storia	25
mario	rossi	geografia	25
mario	rossi	matematica	30
bianca	verdi	geografia	28
bianca	verdi	matematica	27



```
<script type="application/javascript">
    document.write("JS programming");
</script>
```

Reti di Calcolatori

Client-side programming & javascript

Client side programming

- ▶ Possibilità di eseguire codice sul client (browser)
 - ▶ Introdotto in origine come meccanismo per rendere più efficiente la validazione dei dati inseriti in un form
 - ▶ Il controllo di validità viene effettuato sul browser prima di inviare i dati allo script server-side
 - ▶ Si riducono la latenza nella notifica degli errori all'utente, il carico del server e il traffico di rete
 - ▶ E' diventato un supporto indispensabile per lo sviluppo di applicazioni Web 2.0
 - ▶ Gli script permettono di introdurre interattività nelle pagine HTML
 - ▶ Il linguaggio di script fornisce il supporto per accedere agli elementi della pagina HTML e a caratteristiche del browser
 - ▶ Il browser diviene un ambiente di esecuzione di applicazioni che utilizzano la pagina come output grafico e possono interagire con server Web per aggiornare parti di essa in modo dinamico

javascript come standard

- ▶ **javascript** è di fatto diventato uno standard per un linguaggio di script Web lato client
 - ▶ Supportato a partire dalla versione 2.0 del browser Netscape Navigator (1995)
 - ▶ Il nome javascript fu scelto vista la popolarità crescente del linguaggio Java senza che ci sia una vera relazione fra i due linguaggi
 - ▶ Per creare uno standard javascript 1.1 fu sottomesso alla **European Computer Manufacturers Association** (ECMA) nel 1997
 - ▶ Il Technical Committee #39 di ECMA propose uno standard per la sintassi e la semantica di un “general purpose, cross-platform, vendor-neutral scripting language” (**ECMAScript**)
 - ▶ Nel 1998 ECMAScript fu adottato come standard ISO/IEC-16262 (International Electrotechnical Commission) ed è diventato il riferimento per l'implementazione del javascript sui browser

Struttura di javascript

- ▶ Le implementazioni di javascript prevedono tre parti
 - ▶ **ECMAScript**
 - ▶ specifiche base indipendentemente dall'ambiente di esecuzione (host environment – il browser per la programmazione client-side)
 - ▶ Definisce la sintassi, i tipi dato, le istruzioni, le parole chiave del linguaggio, le parole riservate (non usabili per identificatori), gli operatori, gli oggetti predefiniti
 - ▶ supporta la codifica Unicode dei caratteri
 - ▶ **Document Object Model (DOM)**
 - ▶ supporto per la libreria DOM per la rappresentazione e elaborazione di documenti HTML e XML (supporto XML completo da level 3 DOM)
 - ▶ da DOM level 2 è stato introdotto il supporto agli eventi (DOM Events)
 - ▶ **Browser Object Model (BOM)**
 - ▶ Supporto per l'accesso alla finestra del browser (non sempre standard)
 - ▶ Aprire finestre pop-up, spostare e ridimensionare le finestre del browser, gestire la location (indirizzo della pagina), accesso ai cookie

javascript e HTML

- ▶ Il codice javascript è incluso in un file HTML col tag **<script>...</script>**
 - ▶ fra i tag si può specificare il codice dello script
 - ▶ l'attributo **type** specifica il linguaggio di scripting (default javascript)
 - ▶ l'attributo **src** permette di specificare un file esterno che contiene il codice javascript

```
<html>
<head>
  <script type="application/javascript"
    src="00-jrcode.js" ></script>
</head>
<body>
  <script type="application/javascript">
    generatePage();
  </script>
</body>
</html>
```



```
function generatePage() {
  document.write("<h2>Pagina generata da javascript</h2>");
}
```

00-jrcode.js

Dove mettere il codice javascript..

- ▶ In genere è preferibile mettere il codice principale in un file esterno e non inline
 - ▶ C'è maggiore sicurezza perché il codice non è direttamente visibile all'interno della pagina
 - ▶ E' più semplice la manutenzione perché il codice non è disperso nei file HTML
 - ▶ Permette di sfruttare il caching dei browser
 - ▶ se due pagine usano lo stesso codice, questo viene caricato solo una volta
- ▶ Il codice viene caricato ed eseguito in base alla sua posizione nella pagina HTML
 - ▶ Alcuni elementi della pagina possono non essere ancora definiti
 - ▶ Di solito si preferisce associare l'esecuzione ad eventi

ECMAScript: sintassi

- ▶ Tutto è case-sensitive
- ▶ Le variabili non hanno un tipo specifico
 - ▶ Come in PHP le variabili assumono il tipo del valore assegnato
 - ▶ Possono essere definite con l'operatore `var` (non necessario)
 - ▶ I nomi di variabile possono iniziare con `_` o `$` (non necessario)

```
var nome = "Giovanni", anni = 5  
colore = "red"
```

- ▶ I commenti sono definiti da `/* . . */` (multilinea) o `//` (singola linea)
- ▶ Il `;` alla fine delle istruzioni è opzionale (ma consigliato)
- ▶ Le parentesi graffe `{ . . . }` permettono di definire blocchi di istruzioni

Variabili javascript: esempio

```
<h2>Variabili stampate da javascript</h2>
<em>nome</em> è una variabile javascript con valore
<em id="nome"></em> e tipo <em id="tiponome"> </em><br />
<em>anni</em> è una variabile javascript con valore
<em id="anni"> </em> e tipo <em id="tipoanni"> </em><br />
<em>colore</em> è una variabile javascript con valore
<em id="colore"> </em> e tipo <em id="tipocolore"> </em><br />
```

```
<script type="application/javascript">
```

```
  var nome = "Giovanni", anni = 5;
  colore = "red";
```

```
document.getElementById("nome").innerHTML = nome;
document.getElementById("anni").innerHTML = anni;
document.getElementById("colore").innerHTML = colore;
document.getElementById("tiponome").innerHTML = typeof nome;
document.getElementById("tipoanni").innerHTML = typeof anni;
document.getElementById("tipocolore").innerHTML = typeof colore;
</script>
```

Variabili stampate da javascript

nome è una variabile javascript con valore *Giovanni* e tipo *string*
anni è una variabile javascript con valore *5* e tipo *number*
colore è una variabile javascript con valore *red* e tipo *string*

Tipi di variabile

- ▶ Le variabili possono essere associate ai seguenti tipi
 - ▶ **tipi primitivi**
 - ▶ `undefined`, `null`, `boolean`, `number`, `string`
 - ▶ l'operatore **`typeof`** permette di controllare il tipo di una variabile
 - ▶ il tipo `undefined` è associato ad una variabile definita con l'operatore `var` ma non inizializzata con un valore
 - ▶ il tipo `number` permette di rappresentare sia gli interi a 32 bit che i float a 64 bit (può essere anche `Infinity` o `NaN` se si tenta di convertire un non-numero)
 - ▶ il tipo `string` è una sequenza di caratteri Unicode
 - le stringhe costanti si possono definire sia con `""` che `''` con lo stesso significato
 - ▶ **riferimenti**
 - ▶ Sono utilizzati per accedere a oggetti

Conversioni di tipo

- ▶ I tipi primitivi sono di fatto pseudo-oggetti visto che hanno proprietà e metodi
 - ▶ le variabili string hanno la proprietà **length**
`var color = 'red'; - color.length` vale 3
 - ▶ tutti i tipi hanno il metodo di conversione in stringa **toString()**
`var anni = 5; - anni.toString(2)` vale 101 (in base 2)
 - ▶ Le conversioni da string a number si possono fare con le funzioni **parseInt(string)** e **parseFloat(string)**
 - ▶ Le conversioni vengono fatte analizzando la stringa da sinistra a destra
 - ▶ Se non ci sono cifre all'inizio della stringa le funzioni produce NaN
 - ▶ `parseInt` può avere un secondo parametro per indicare la base (2,8,16)
 - ▶ Le conversioni si possono fare anche col casting
 - ▶ **Boolean(v)** , **Number(v)** , **String(v)**

Riferimenti

- ▶ I riferimenti sono usati per accedere ad oggetti
 - ▶ ECMAScript non ha meccanismi per la definizione diretta di classi (non ha la keyword `class`)
 - ▶ Un oggetto è un'istanza di una classe
 - ▶ Un oggetto si crea con l'operatore **new**
 - ▶ `var o = new Object();`
 - ▶ Se il costruttore non ha parametri le parentesi si possono omettere
 - ▶ **Object** è la classe base di tutti gli oggetti
 - ▶ Tutti i metodi e proprietà presenti in questa classe sono ereditati da tutte le altre
 - proprietà: `constructor`, `prototype`
 - metodi: `hasOwnProperty('name')`, `isPrototypeOf(obj)`, `propertyIsEnumerable('name')`, `toString()`, `valueOf()`

```
var obj = new Object();  
objId.innerHTML= obj.toString();
```

obj è un riferimento ad un oggetto *[object Object]*

La classe String

- ▶ La classe **String** è la rappresentazione a oggetti del tipo primitivo string

```
var objString = new String("Javascript language");
```

- ▶ la proprietà **length** fornisce la lunghezza in caratteri (Unicode)
- ▶ ha numerosi metodi utili per la manipolazione di stringhe
 - ▶ `charAt(pos)`, `charCodeAt(pos)` – accesso ai caratteri
 - ▶ `indexOf(subs)`, `lastIndexOf(subs)` – ricerca di sottostringhe
 - ▶ `substring(start,end)` – estrae una sottostringa
 - ▶ `toUpperCase()`, `toLowerCase()` – convertono la Stringa in maiuscolo o minuscolo
- ▶ L'operatore `+` permette di concatenare stringhe (per il tipo primitivo string)
 - ▶ `var s1 = new String('java'), s2 = new String('javascript');`
 - ▶ `s3 = s1+" e "+s2+ "` sono parenti alla lontana!"

Operatori 1

- ▶ Sono in larga parte gli stessi degli altri linguaggi
 - ▶ **Aritmetici**
 - ▶ binari: `+`, `-`, `*`, `/`, `%`
 - Se uno degli operandi non è `numeric` viene convertito (a parte per `+`)
 - `var v = 10*"5"; -v` è il numero 50
 - ▶ `+` su stringhe – concatenazione
 - conversione automatica se necessario di variabili non stringa in stringa
 - `var v = 10+"5"; -v` è la stringa `"105"`
 - ▶ Autoincremento/decremento pre o postfisso (`++`, `--`)
 - ▶ **Operatori bit a bit**
 - ▶ `not` (`~`), `and` (`&`), `or` (`|`), `xor` (`^`), `left shift` (`<<`), `signed right shift` (`>>`), `unsigned right shift` (`>>>`)
 - ▶ **Operatori booleani**
 - ▶ `NOT` (`!`), `AND` (`&&`), `OR` (`||`)

Operatori 2

▶ Operatori relazionali

- ▶ confronto (<,<=,>,>=)
 - ▶ Si applicano sia a numeric che string con eventuale conversione di tipo
- ▶ uguaglianza/disuguaglianza (==,===,!=,!==)

▶ Operatore condizionale

- ▶ `(cond) ? valIfTrue : valIfFalse`

▶ delete

- ▶ permette di cancellare (rendere undefined) il riferimento ad una proprietà o metodo di un oggetto definita in precedenza
 - ▶ `var o = new Object(); o.language = "Java"; delete o.language;`

▶ void

- ▶ `ritorna undefined per ogni valore`
 - ▶ è usato per evitare l'output di un valore
 - ▶ `<a href="javascript:void window.open('www.google.com');>google</a>`

Controllo di flusso

► Esecuzione condizionale

- `if (cond) true-code [else false-code]`
- `switch (exp) { case ... default: .. }`

► Cicli

- `do { loop-code } while (cond)`
- `while (cond) { loop-code }`
- `for (init; cond; post-loop) { loop-code }`
- `for (property in expression) { loop-code }`
 - E' utilizzato per enumerare le proprietà di un oggetto
 - Non tutte le proprietà sono enumerabili
 - si può verificare col metodo `propertyIsEnumerable(nome)`
- `break; continue;`

Funzioni

- ▶ Le funzioni sono definite con la keyword function

```
function nomeF(arg0, arg1, . . . . ., argN) {  
    istruzioni;  
}
```

- ▶ La funzione può fornire un valore con return val; o nessun valore (non si specifica return o si usa return;)
- ▶ Se si definiscono più funzioni usando lo stesso nome
 - ▶ non si generano errori
 - ▶ la definizione in un certo punto del codice è l'ultima incontrata
- ▶ L'oggetto arguments permette di accedere agli argomenti passati alla funzione anche se questi non sono elencati
 - ▶ arguments[0], arguments[1]...
 - ▶ il numero di argomenti è arguments.length
 - ▶ Non c'è nessun controllo sul numero di argomenti effettivamente passato

Oggetti

► Oggetti senza classi...?

- Un oggetto è una collezione non ordinata di proprietà ciascuna delle quali contiene un valore primitivo, un riferimento ad un oggetto o ad una funzione (metodo)
- Un oggetto ricorda un array associativo in cui le proprietà sono le chiavi
- ECMAScript non ha la definizione formale di classe ma introduce il concetto di *object definition* come template con cui creare le istanze di oggetti
 - L'object definition è implementata nella funzione costruttore che aggiunge all'oggetto le proprietà previste dall'interfaccia
 - Sono comunque supportate le funzionalità di un linguaggio ad oggetti
 - **Encapsulation** (l'oggetto racchiude proprietà e metodi)
 - **Aggregation** (un oggetto può contenerne un altro)
 - **Inheritance** (una classe eredita proprietà e metodi da un'altra)
 - **Polymorphism** (lo stesso metodo può gestire oggetti di classi diverse)

Array

- ▶ Un oggetto Array è una collezione ordinata di elementi
 - ▶ Si crea col costruttore che può prevedere
 - ▶ nessun argomento - `var arr = new Array()`
 - ▶ il numero iniziale di elementi – `var arr = new Array(10)`
 - ▶ la lista di elementi – `var arr = new Array("a", "b", "c")`
 - ▶ si può creare con la lista degli elementi fra [] - `var arr=["a", "b"]`
 - ▶ Gli elementi sono accessibili con l'indice numerico `0, ..., length-1` con la notazione `v[i]`
 - ▶ la proprietà `length` contiene la dimensione attuale del vettore
 - ▶ se si assegna un valore ad un elemento oltre la dimensione attuale, l'array viene redimensionato di conseguenza
 - ▶ Sono definiti numerosi metodi
 - ▶ `toString()`, `join()`, `split()`, `slice()`, `concat()`, `pop()` – preleva dalla fine dell'array, `push()` – inserisce in coda all'array, `shift()` - estrae dalla testa dell'array, `unshift()` - inserisce in testa all'array, `sort()`, ...

Proprietà e metodo degli oggetti

- ▶ Si possono aggiungere dinamicamente proprietà o metodi a un oggetto
 - ▶ si assegna un valore alla proprietà o metodo
 - ▶ si può usare il riferimento con `obj.prop` o come indice di array associativo `obj["prop"]`
 - ▶ la keyword **this** permette di riferirsi alla proprietà dell'oggetto specifico in un metodo

```
var o = new Object(); // oggetto di classe base
// proprietà aggiunte
o.style = "color: red;";
o.style += "background-color: yellow;";
o.text = "test"
// metodi
o.createBox = function() {
    return "<span style=\""+this.style+"\">"
        +this.text+"</span>";
}
```

o.createBox()

string
"test"

Costruttori di oggetti...

- ▶ Un **costruttore** è una funzione con il nome della classe
 - ▶ Per convenzione si usano nomi con iniziale maiuscola
 - ▶ Nel corpo della funzione non si creano oggetti ma si istanziano le proprietà usando `this` come riferimento
 - ▶ Il costruttore viene chiamato usando l'operatore `new` per generare un oggetto con le proprietà istanziate
 - ▶ Il costruttore può avere zero o più argomenti per inizializzare le proprietà

```
function TextBox(txt,clr,bgclr,size) {  
  // definizione delle proprietà  
  this.color = clr;  
  this.bgcolor = bgclr;  
  this.size = size;  
  this.text = txt;  
  
  var st = "color: "+clr+";";  
  st += "background-color: "+bgclr+";";  
  st += "font-size: "+size+";";  
  this.style = st;  
}
```

```
o1 = new TextBox("Tinky Winky","purple","lightgray",14);  
test = o1 instanceof TextBox
```

...e i metodi?

.. la proprietà prototype

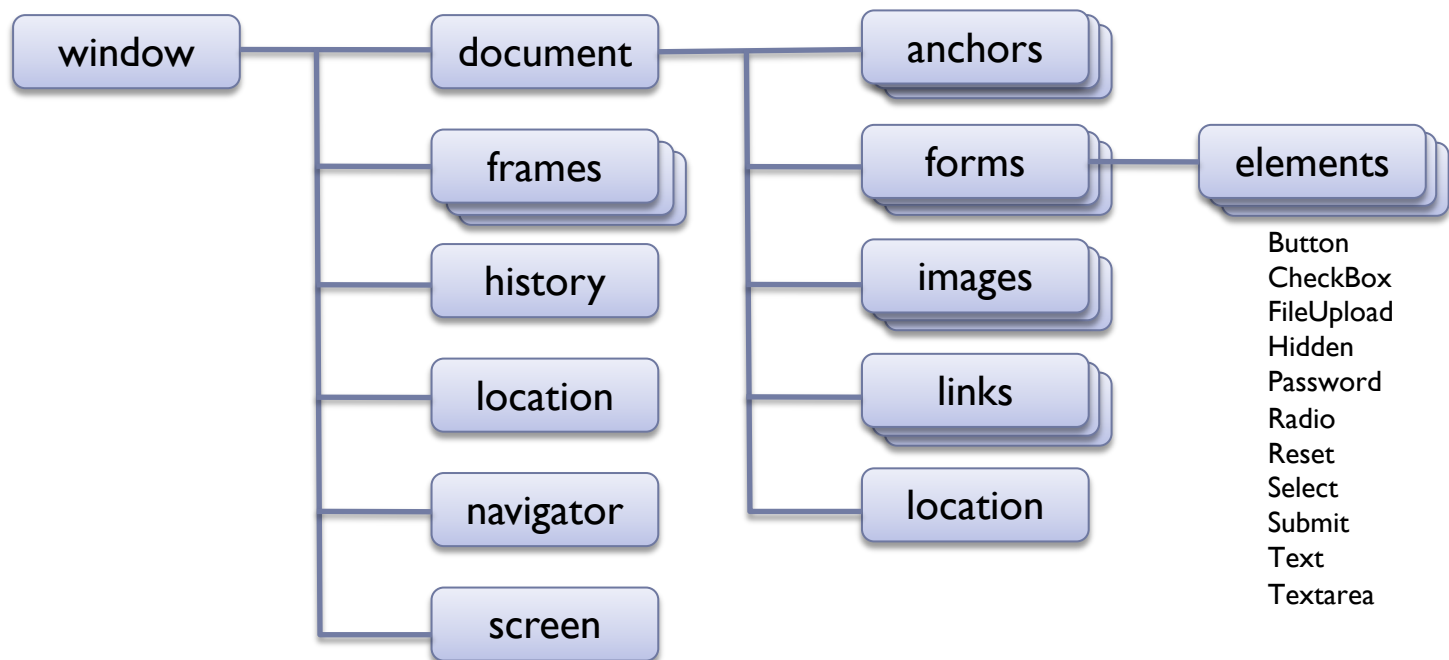
- ▶ La proprietà **prototype** di un oggetto definisce il template con cui creare nuovi oggetti con `new`
 - ▶ E' una lista di proprietà e metodi con il relativo valore
 - ▶ Gli elementi di prototype vengono aggiunti agli oggetti creati con `new` inizializzandoli con il valore predefinito
 - ▶ Se il valore è un riferimento tutti gli oggetti condividono lo stesso oggetto
 - ▶ E' utile per definire i metodi in modo che tutti gli oggetti condividano lo stesso codice

```
TextBox.prototype.createBox = function() {  
    return "<span style=\""+this.style+"\">"+this.text+"</span>";  
}
```

```
o1 = new TextBox("Tinky Winky", "purple", "lightgray", 14);  
o1Id.innerHTML = o1.createBox();
```

Il Browser Object Model

- ▶ IL BOM definisce gli oggetti per interagire con la finestra del browser indipendentemente dal contenuto
 - ▶ Il BOM contiene una serie di oggetti organizzati gerarchicamente
 - ▶ La radice è l'oggetto `window`

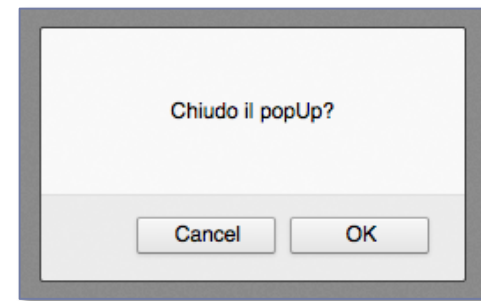


window popups & system dialogs

- ▶ E' possibile aprire nuove finestre

```
newwin = window.open("url","winName","settings")
```

- ▶ In genere il browser blocca l'apertura di finestre popup chiedendo all'utente un'abilitazione per sito
- ▶ Le settings permettono di definire le caratteristiche della finestra (height, width, top, left, resizable, scrollable, toolbar, status, location)
- ▶ Le finestre popup hanno metodi per la loro manipolazione (close, moveBy, resizeBy, moveTo,..)
- ▶ Si possono aprire **finestre di dialogo**
 - ▶ `alert("messaggio")`
 - ▶ `confirm("conferma")`
 - ▶ `prompt("richiesta")`



Intervals & timeout

► Esecuzione temporizzata di codice

- **Timeouts** - `ToutId = setTimeout(func, msec)`
 - Esecuzione di una funzione dopo un numero prefissato di millisecondi
 - Si può annullare prima che scada con `clearTimeout(ToutId)`
- **Interval** - `IntId = setInterval(func, msec)`
 - Esecuzione ripetuta di una funzione ad intervalli di un numero prefissato di millisecondi
 - Si può sospendere con `clearInterval(IntId)`

```
var TIntId;  
function startAdRotation() {  
  if(popupwin) TIntId = setInterval(changeAds, 1500);  
}  
  
function stopAdRotation() {  
  if(TIntId) clearInterval(TIntId);  
}
```

```
var ads = new Array("Eat popUP!!",  
  "Drink PoPUP!!", "Read POPup!!");  
var ad;  
function changeAds() {  
  if(!popupwin.document) return;  
  el = popupwin.document.getElementById("adId");  
  el.innerHTML=ads[ad++];  
  if(ad>ads.length) ad=0;  
}
```

Supporto al DOM

- ▶ javascript ha il supporto alle **API standard DOM**
 - ▶ Forniscono il modo consigliato per manipolare il documento HTML visualizzato nel browser e accessibile come `window.document`
 - ▶ La base è l'oggetto **Node** la cui interfaccia prevede (fra gli altri)
 - ▶ **nodeName** – String il nome del nodo
 - ▶ **nodeValue** – String il valore del nodo
 - ▶ **nodeType** – Number il tipo del nodo (es. `Node.ELEMENT_NODE`)
 - ▶ **firstChild** – Node il riferimento al primo figlio
 - ▶ **childNodes** – NodeList la lista di tutti i figli
 - ▶ **hasChildNodes()** – Boolean è true se il nodo ha figli
 - ▶ **attributes** – NamedNodeMap è la lista degli attributi
 - ▶ **appendChild(*node*)** – Node aggiunge *node* ai figli in ultima posizione
 - ▶ **removeChild(*node*)** – Node rimuove il nodo dalla lista dei figli
 - ▶ **insertBefore(*node*,*refnode*)** – Node inserisce il nodo nei figli nella posizione precedente il nodo *refnode*

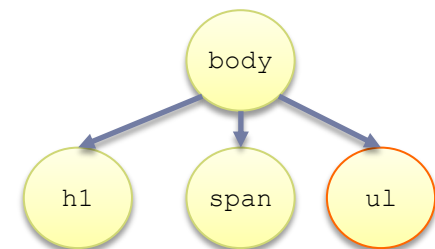
Accesso a nodi specifici

- ▶ La libreria prevede metodi per ottenere il riferimento a nodi specifici
 - ▶ permettono di ottenere nodi di interesse senza dover effettuare una visita dell'albero DOM
 - ▶ `getElementsByTagName("tag")` – `NodeList` fornisce tutti gli elementi (`Element`) col tag specificato
 - ▶ `getElementsByName("name")` – `NodeList` fornisce la lista di tutti gli elementi con il valore indicato per l'attributo `name`
 - ▶ `getElementById("id")` – `Node` fornisce l'elemento con il valore indicato per l'attributo `id`

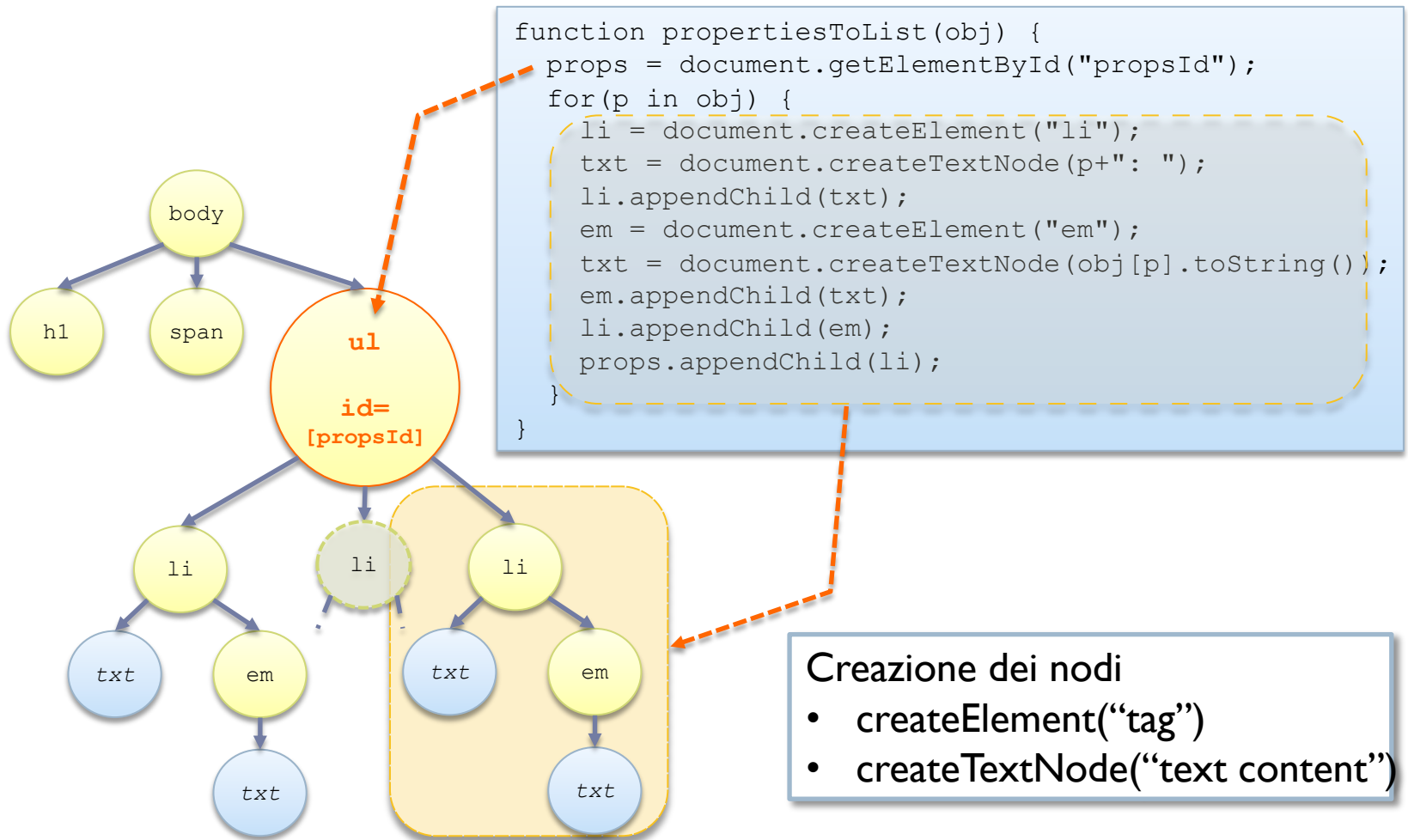
```
props = document.getElementById("propsId");
```

HTML

```
<ul id="propsId"></ul>
```



Esempio di generazione di DOM



HTML DOM

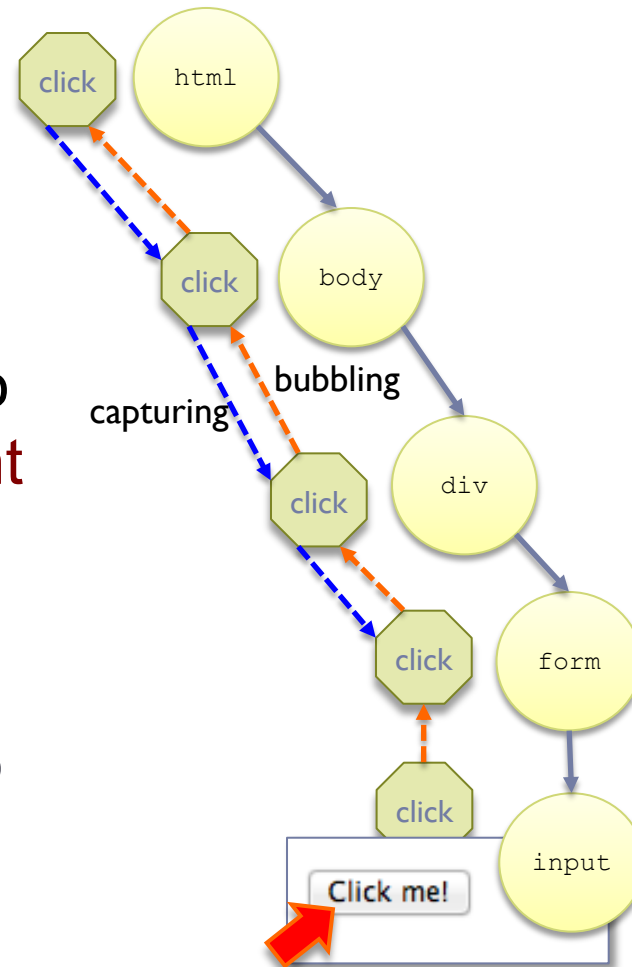
- ▶ Il supporto specifico DOM per HTML prevede la semplificazione di alcune operazioni
 - ▶ Accesso agli attributi con i metodi del Core DOM
 - ▶ `node.getAttribute("style")`
 - ▶ `node.setAttribute("style", "color: red")`
 - ▶ `node.removeAttribute("style")`
 - ▶ Accesso agli attributi con HTML DOM – gli attributi sono direttamente proprietà dell'oggetto
 - ▶ `st = node.style;`
 - ▶ `node.style = "color: red";`
 - ▶ `delete node.style;`
 - ▶ Gestione degli `Element` `<table>` con proprietà e metodi
 - ▶ `tab.rows`, `tab.deleteRow(pos)`, `tab.insertRow(pos)`, ...
`tr.cells`, ...

Eventi

- ▶ In genere l'interazione del codice col documento HTML avviene sulla base del verificarsi di **eventi**
 - ▶ Gli eventi sono generati dall'utente o dal browser
 - ▶ caricamento della pagina
 - ▶ click su un elemento
 - ▶ passaggio del cursore del mouse su un elemento
 - ▶ fuoco dell'input su un elemento
 - ▶ E' possibile associare l'esecuzione di codice al verificarsi di un particolare evento
 - ▶ Gli eventi sono associati ad un **event flow**
 - ▶ più di un elemento sulla stessa pagina può rispondere allo stesso evento
 - quando si clicca un bottone l'evento è per il bottone, il contenitore in cui si trova (es. una tabella o un form), fino ad arrivare alla pagina stessa

Propagazione e cattura degli eventi

- ▶ L'event si propaga nel DOM dall'elemento più in basso in cui è stato generato verso la radice (**event bubbling**)
- ▶ L'event si propaga nel DOM dall'elemento più in alto verso l'elemento più specifico (**event capturing**)
- ▶ Il DOM supporta entrambe le modalità
- ▶ L'event capturing viene effettuato prima



Event handlers/Listeners

- ▶ Una funzione chiamata in risposta ad un evento è detta **event handler** or **event listener**
 - ▶ una funzione che risponde all'evento click è un *onclick handler*
 - ▶ per assegnare un event handler ad un elemento della pagina HTML occorre inserire il nome della funzione nella proprietà corrispondente dell'elemento

```
<div id= "d1" onclick="myClickHandler()">..</div>
```

attributo del tag HTML

```
div1 = document.getElementById("d1");  
div1.onclick = myClickHandler;
```

proprietà dell'elemento DOM
corrispondente all'evento

```
div1 = document.getElementById("d1");  
div1.addEventListener("onclick", myClickHandler, true);
```

metodo DOM per la
registrazione di gestori
di evento (se ne possono
registrare anche più di 1)

Rimozione di un event handler

- ▶ Per rimuovere un event handler
 - ▶ Si può assegnare null alla proprietà corrispondente
 - ▶ `div1.onclick = null;`
 - ▶ Si può usare il metodo DOM di rimozione
 - ▶ `div1.removeEventListener("onclick", myClickHandler, true);`
 - ▶ Nel caso si siano assegnati più event handler allo stesso evento vanno rimossi uno alla volta
- ▶ Se si vuole cambiare l'event handler è sufficiente assegnare alla proprietà corrispondente all'event handler un nuovo valore
 - ▶ `div1.onclick = newEventHandler;`

L'oggetto evento

- ▶ Le caratteristiche dell'evento sono utili per la gestione
 - ▶ L'**Event Object** contiene le informazioni sull'evento che si è verificato
 - ▶ L'oggetto che ha causato l'evento
 - **target** – elemento che ha generato l'evento
 - ▶ Informazioni riguardanti il puntatore (mouse) al momento dell'evento
 - **button** – stato dei bottoni del mouse
 - **clientX**, **clientY** – le coordinate del mouse nella finestra
 - ▶ Informazioni sulla tastiera al momento dell'evento
 - **charCode** – codice del carattere battuto sulla tastiera
 - **ctrlKey** – indica se il tasto CTRL è premuto
 - ▶ Informazioni generali sull'evento
 - **timeStamp** – istante di tempo in cui si è verificato
 - **type** – tipo dell'evento (“click”, “mouseover”,...)
 - **eventPhase** – (0 capturing phase, 1 - at target, 2 – bubbling phase)
 - ▶ L'oggetto è disponibile come parametro dell'event handle

Tipi di evento

▶ Mouse events

- ▶ generati da azioni effettuate dall'utente col mouse
 - ▶ click, dblclick, mouseover, mousemove, mouseout, mouseup, mousedown

▶ Keyboard events

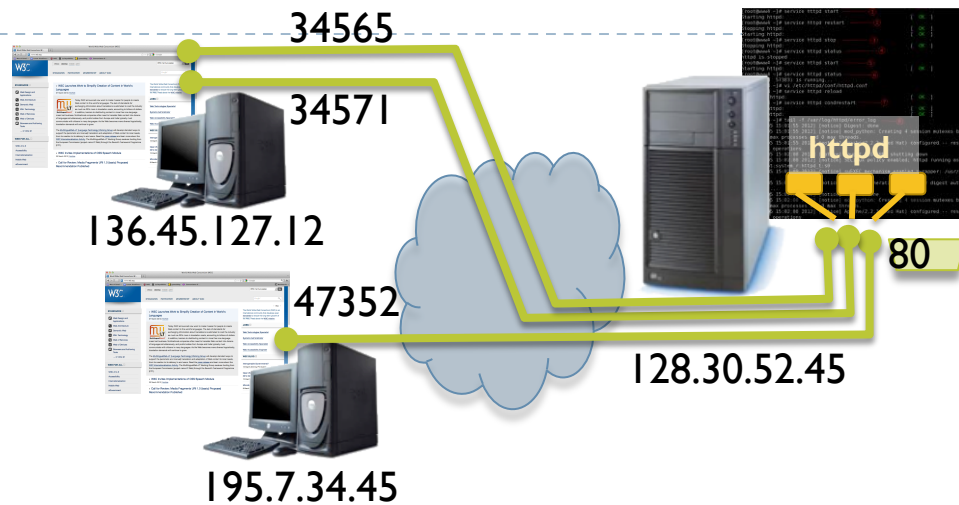
- ▶ generati dalla pressione di tasti sulla tastiera
 - ▶ keydown, keypress, keyup

▶ HTML events

- ▶ generati quando avvengono cambiamenti nella finestra del browser o si verifica interazione fra client e server
 - ▶ load, change (textbox), submit (form), focus,....

▶ Mutation events

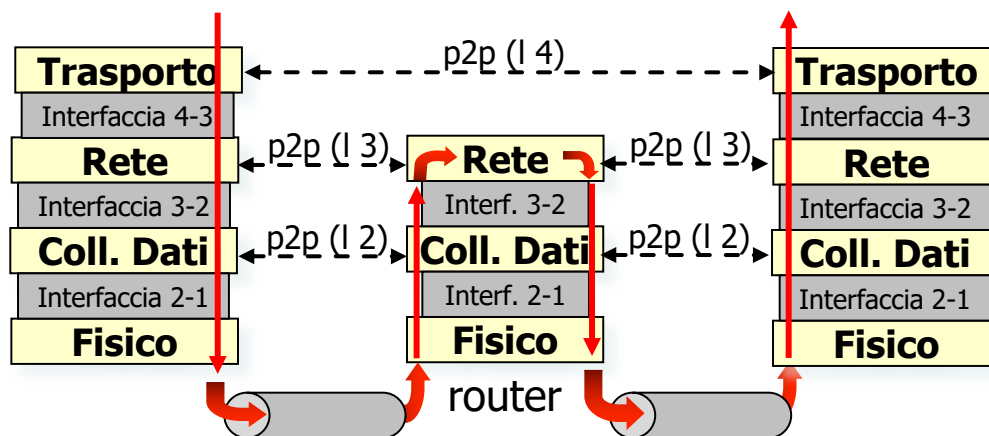
- ▶ generati quando ci sono cambiamenti nella struttura del DOM
 - ▶ DOMSubtreeModified, DOMNodeInserted,...



Reti di Calcolatori

Transport Layer & TCP/UDP

Il livello di trasporto



- ▶ Fornisce un servizio per la **comunicazione logica** fra **processi applicativi** in host su rete
 - ▶ E' una funzionalità del sistema operativo disponibile come libreria API con cui si possono implementare meccanismi di scambio dati fra applicazioni (**socket API**)
 - ▶ Presuppone un **modello client-server**
 - ▶ Gestisce l'indirizzamento verso dei processi (**indirizzo di porta**) e il multiplexing/demultiplexing dei messaggi (**indirizzo di socket**)

Servizio di trasporto

- ▶ Il livello di trasporto può offrire più modalità per lo scambio dati fra due applicazioni
 - ▶ Servizio per **stream dati (con connessione)**
 - ▶ fornisce un **canale affidabile** su cui scrivere o da cui leggere dati
 - ▶ dal punto di vista del programmatore è **come un file**
 - ▶ deve gestire il **controllo degli errori**, la **perdita di pacchetti**, i **numeri di sequenza** per l'invio del flusso dati come sequenza di datagram e il **controllo di flusso** per un collegamento attraverso una rete inaffidabile
 - ▶ deve nascondere il problema della capacità di memorizzazione della rete (un pacchetto può essere memorizzato in un router e consegnato dopo un certo ritardo)
 - ▶ Servizio **datagram (senza connessione)**
 - ▶ fornisce il meccanismo di indirizzamento verso le applicazioni (porte)
 - ▶ replica le funzionalità di consegna del servizio di consegna dei pacchetti (datagram) del livello di rete
 - ▶ può aggiungere il controllo a rilevazione di errore (con checksum)

Tipi di servizi e QoS

- ▶ Le modalità con connessione (**stream**) e senza connessione (**datagram**) sono modalità di base
 - ▶ Disponibili per lo sviluppo di applicazioni su rete Internet (TCP/UDP)
 - ▶ In generale potrebbero essere sviluppate tipologie di trasporto con funzionalità/garanzie aggiuntive, come ad esempio
 - ▶ servizio con flusso dati garantito
 - ▶ servizio con ritardo dati massimo limitato
- ▶ Un'applicazione potrebbe richiedere la tipologia di servizio di trasporto in base al tipo di esigenze richieste
 - ▶ La conoscenza del livello di **qualità del servizio** (Quality of Service) richiesto può essere utilizzata per ottimizzare l'uso delle risorse di rete
 - ▶ Le richieste possono non essere accettate se il sistema non è in grado di soddisfarle (col carico attuale)
 - ▶ Il tipo di servizi che si possono offrire dipendono anche dalle caratteristiche del livello di rete (es. gestione del ritardo massimo)

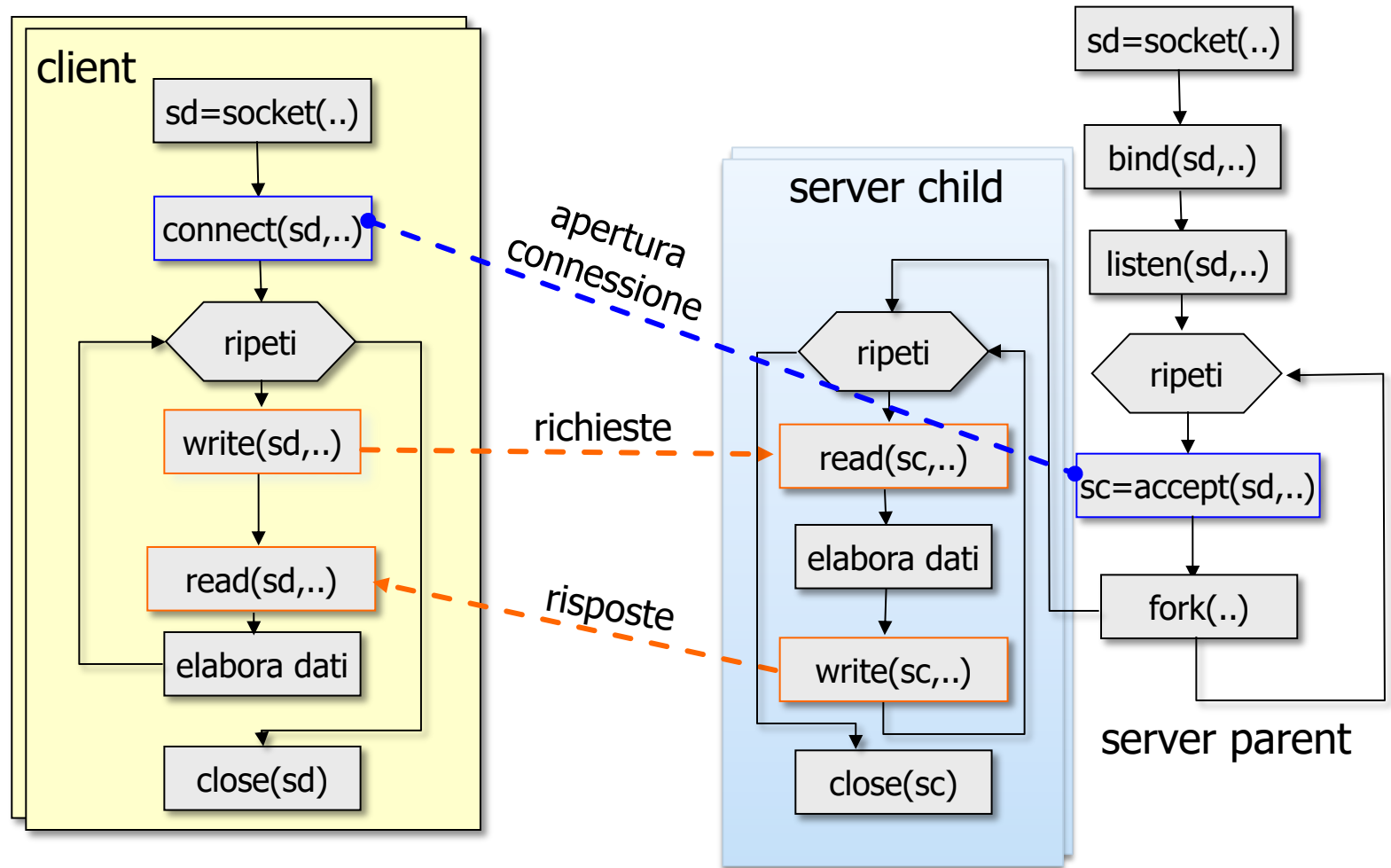
Modello client-server e API - 1

- ▶ L'interfaccia API del servizio di trasporto assume il modello client-server
 - ▶ L'agente server attiva una procedura che rimane in attesa di richieste da parte di un agente client (**LISTEN**)
 - ▶ Il server effettua una richiesta al Sistema Operativo (SO) per la creazione di un end-point (**server socket**) specificando il tipo di servizio richiesto (**stream/dgram**)
 - ▶ Il processo server specifica la **porta** su cui si metterà in ascolto
 - La porta deve essere non già in uso e nota al processo client
 - ▶ La procedura crea le strutture dati a livello di SO per la gestione della comunicazione col livello di servizio richiesto (buffer, code di richiesta di connessione, strutture per il multiplexing di più canali di comunicazione,...)
 - ▶ Per il servizio stream il server rimane in attesa di accettare una richiesta di connessione (**ACCEPT**)
 - ▶ Per il servizio datagram rimane in attesa dell'arrivo di un pacchetto datagram da un client (**RECVFROM**)

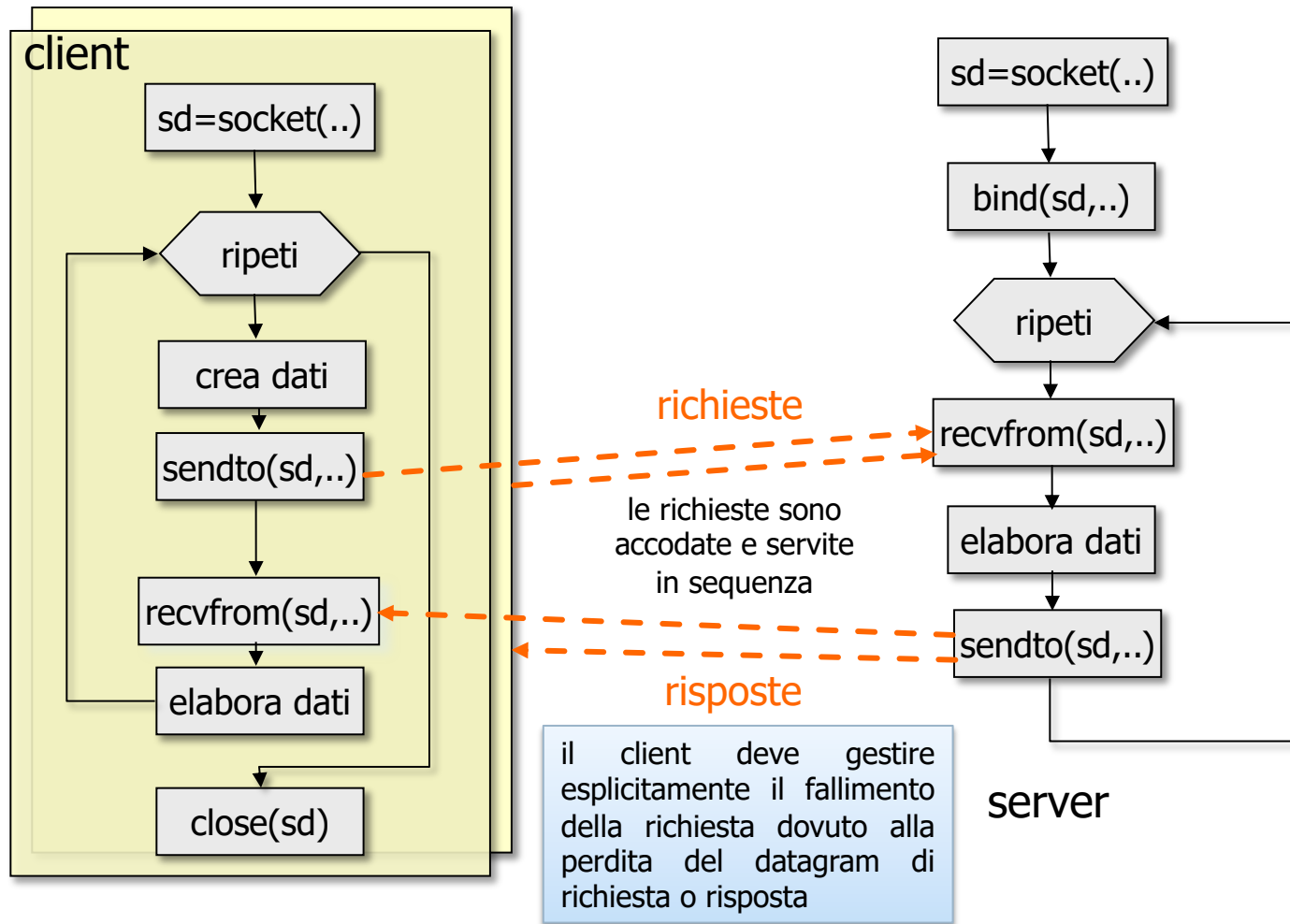
Modello client-server e API - 1

- ▶ Dopo che il server ha inizializzato l'end-point e si è messo in attesa
 - ▶ Il processo server è bloccato in attesa di un evento generato da un client
 - ▶ Richiesta di connessione/invio di un datagram
 - ▶ L'agente client invia una richiesta al server usando il suo indirizzo di rete e di trasporto (IP+porta)
 - ▶ Può essere una richiesta di connessione (**CONNECT**) o l'invio di un messaggio datagram (**SENDTO**) in base al tipo di servizio usato
- ▶ Per i servizi stream con connessione, client e server si scambiano dati leggendo/scrivendo (**SEND/RECEIVE**) sul canale (socket)
 - ▶ Al termine dello scambio la connessione viene chiusa (**DISCONNECT**)

Client-server stream



Client-server datagram



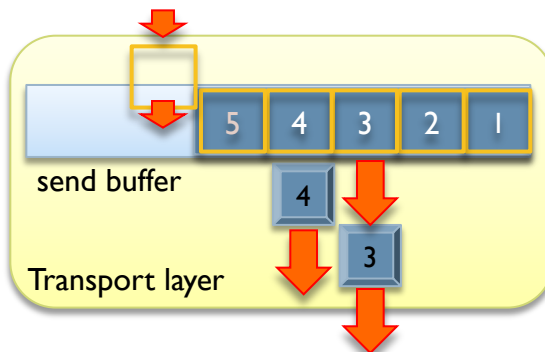
TCP e UDP

- ▶ Protocolli di trasporto definiti su rete Internet (su IP)
 - ▶ **Transmission Control Protocol (TCP)**
 - ▶ protocollo di trasporto orientato alla connessione
 - ▶ definito in RFC 793, RFC 1122 e RFC 1323
 - ▶ progettato per fornire un flusso affidabile end-to-end su una rete internet non affidabile (consegna best-effort)
 - ▶ include meccanismi per il controllo di flusso e della congestione
 - ▶ **User Datagram Protocol (UDP)**
 - ▶ protocollo senza connessione che invia messaggi in datagram IP indipendenti
 - ▶ descritto in RFC 768
- ▶ L'indirizzo di trasporto è il **numero di porta** a 16 bit
 - ▶ TCP e UDP usano due spazi di porte distinti
 - ▶ La differenza è definita da un campo dell'intestazione IP che indica il protocollo usato

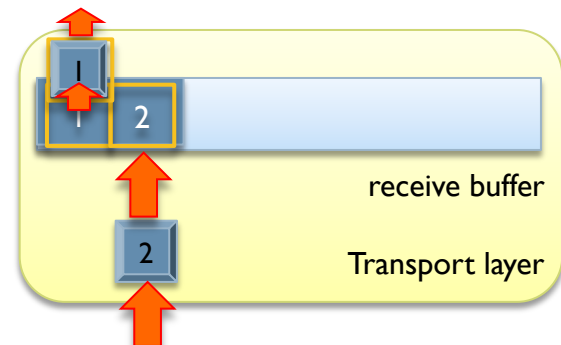
Servizio stream con TCP

- ▶ TCP implementa un trasferimento affidabile di un flusso dati
 - ▶ Il flusso è scomposto in datagram
 - ▶ I datagram sono inviati al destinatario usando un servizio di rete non affidabile
 - ▶ I datagram possono essere persi
 - ▶ I datagram posso arrivare in un ordine diverso da quello di spedizione
 - ▶ Il contenuto dei datagram si può corrompere

`write(sd, databuf, numbytes)`



`read(sd, databuf, maxbytes)`



Affidabilità del servizio

▶ Perdita dei pacchetti

- ▶ Per garantire l'arrivo di tutti i pacchetti viene adottato un meccanismo basato sul **riscontro (ack)** della ricezione
 - ▶ Il ricevente invia un ack in risposta alla corretta ricezione di un pacchetto
 - ▶ Il mittente ritrasmette il pacchetto se non riceve l'ack prima dello scadere di un timeout (**ritrasmissione** – Automatic Repeat Request)
 - Il mittente rimanda il pacchetto fino a che non è sicuro della sua corretta ricezione
 - Il meccanismo gestisce sia la perdita del pacchetto che del suo ack
 - Dopo un certo numero di ritrasmissioni fallite si genera un errore

▶ Ordinamento e duplicazione

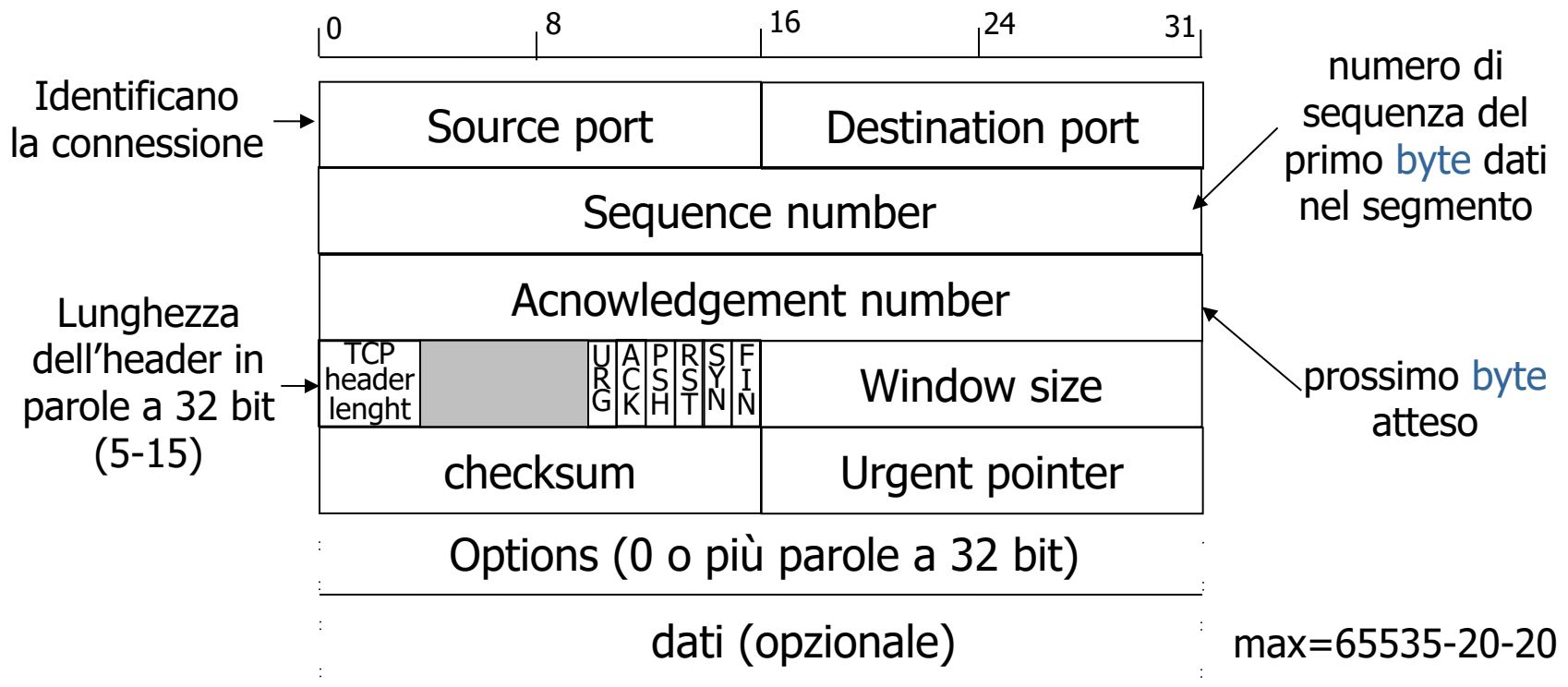
- ▶ Il protocollo prevede la **numerazione dei pacchetti** e dei relativi ack
 - ▶ Il numero d'ordine identifica univocamente i dati nella connessione e permette di riconoscere pacchetti duplicati (si scartano)

▶ Controllo di errore

- ▶ Un **codice a rilevazione di errore** permette di scartare i pacchetti corrotti (sono ritrasmessi quando scade il timeout del mittente)

I segmenti TCP

- Ogni **segmento** ha un header fisso di 20 byte più eventuali parti opzionali seguito da 0 o più byte di dati



TCP flag bits - 1

- ▶ Nel segmento TCP sono presenti 6 bit di flag
 - ▶ **URG**
 - ▶ Indica che l'Urgent Pointer specifica la posizione di dati urgenti a partire dal numero di sequenza attuale (es. pressione di CTRL-C per interrompere il programma remoto)
 - ▶ **ACK**
 - ▶ Indica se il campo Acknowledgement number è valido
 - ▶ **PSH**
 - ▶ Indica dati di tipo PUSH ovvero si richiede di consegnare subito i dati senza bufferizzarli (la funzione read esce)
 - ▶ **RST**
 - ▶ Richiesta di reinizializzazione di una connessione diventata instabile
 - ▶ Viene anche usato per rifiutare un segmento non valido o l'apertura di una connessione

TCP flag bits 2

▶ SYN

- ▶ Viene utilizzato per creare connessioni
- ▶ La richiesta di connessione è caratterizzata da SYN=1 e ACK=0
- ▶ La risposta di connessione contiene un ack e quindi ha SYN=1 e ACK=1
- ▶ Individua i segmenti TCP CONNECTION REQUEST e CONNECTION ACCEPTED

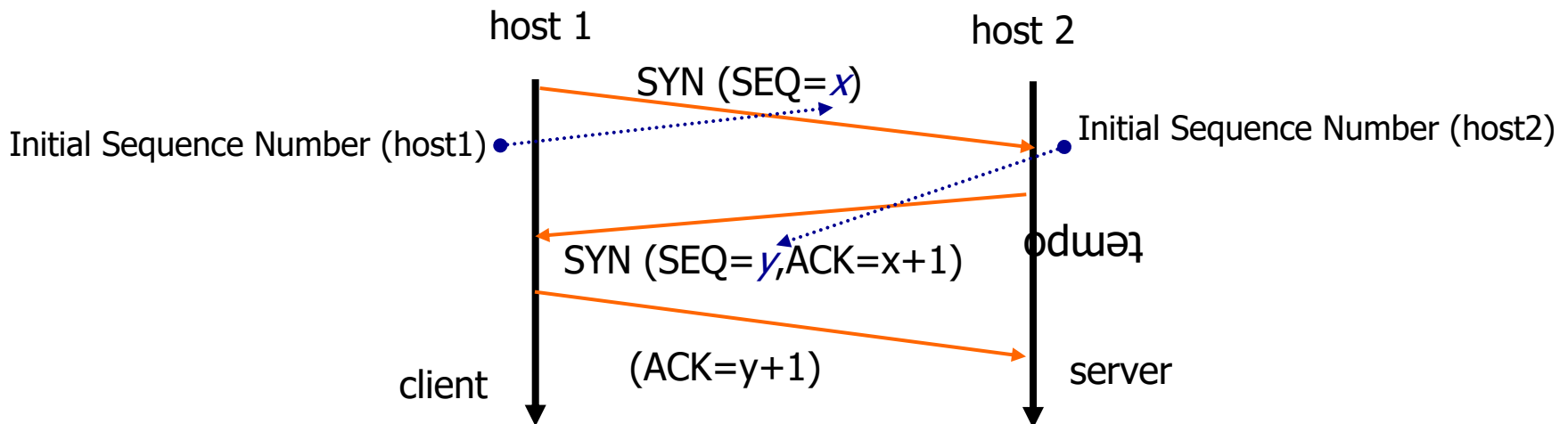
▶ FIN

- ▶ Viene utilizzato per chiudere una connessione (il mittente non ha altri dati da spedire)

- ▶ I flag ACK, SYN, FIN, RST sono usati nelle procedure di gestione della connessione
- ▶ I flag URG, PSH sono usati per richieste particolari nello scambio dati

Apertura della connessione

- ▶ Si utilizza un protocollo **3-way handshake**
 - ▶ Sincronizza la numerazione dei pacchetti (valore iniziale)
 - ▶ La specifica richiede che il numero iniziale sia casuale in modo da evitare che si possano scambiare segmenti relativi a due connessioni distinte con gli stessi parametri e aperte a breve distanza di tempo
 - ▶ I segmenti possono essere memorizzati nella rete ma in ogni caso la loro vita non può eccedere la **Maximum Segment Lifetime (MSL)** di 2 minuti
 - ▶ Il TCP ricevente invia un segmento di rifiuto della connessione (**RST**) se non esiste un processo in attesa sulla porta destinazione



Un esempio di connessione

- ▶ Connessione Telnet fra da 10.6.1.9 a 10.6.1.2 catturata con **tcpdump** *
 - ▶ porta client 4548 - porta server 23 (telnet)

```
10.6.1.9.4548 > 10.6.1.2.23: S 2115515278:2115515278 (0) win 32120
<mss 1460,nop,nop,sackOK,nop,wscale 0> (DF)
```

opzioni da negoziare
(es. **Max Segment Size MSS**)

pacchetti senza dati

```
10.6.1.2.23 > 10.6.1.9.4548: S 1220480853:1220480853 (0)
ack 2115515279 win 32120 <mss 1460,nop,nop,sackOK,nop,wscale 0>
(DF)
```

```
10.6.1.9.4548 > 10.6.1.2.23: . ack 1220480854 win 32120 (DF)
```

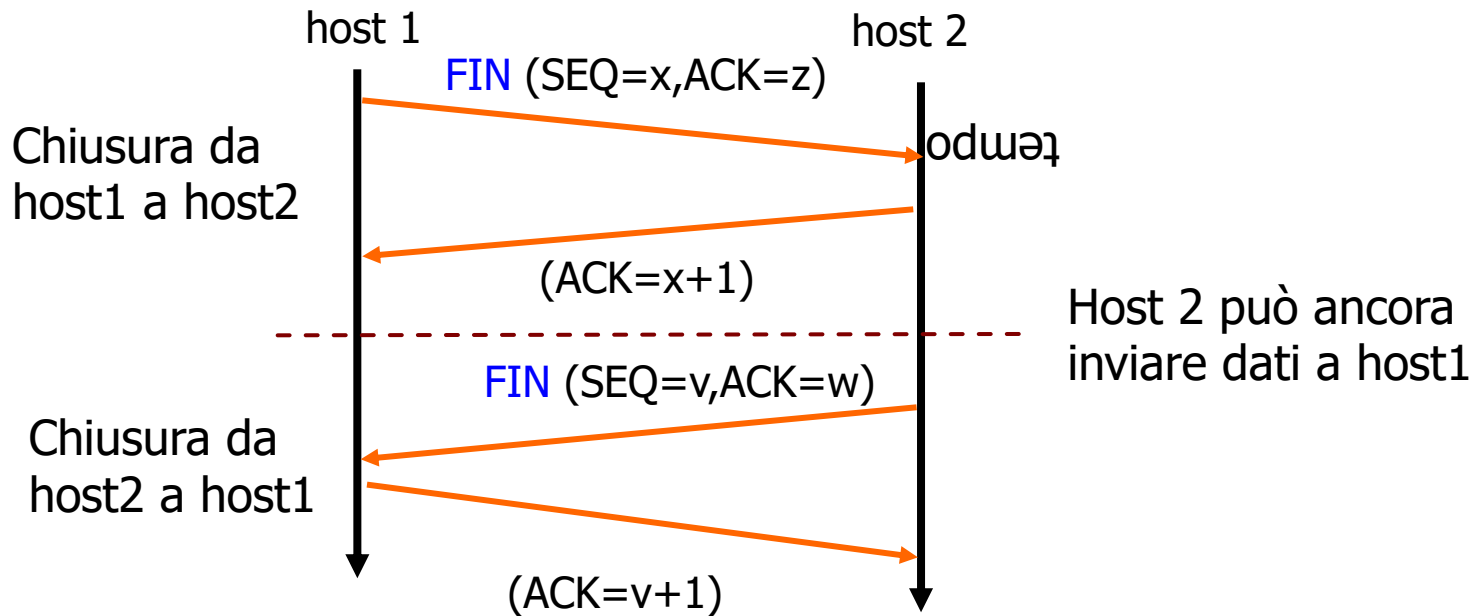
* `tcpdump -S -n -t \ (dst 10.6.1.2 and src 10.6.1.9\ ) or \ (dst 10.6.1.9 and src 10.6.1.2\ )`

Sequence Number Wrap Around

- ▶ Lo spazio dei numeri di sequenza è finito $[0, 2^{32}-1]$
 - ▶ Si ripresentano gli stessi numeri di sequenza dopo 4Gb
- ▶ Può essere un problema quando
 - ▶ Si trasferiscono flussi dati più lunghi di 4Gb
 - ▶ La velocità del trasferimento è tale che la numerazione si ripresenta prima dello scadere del Maximum Segment Lifetime
 - ▶ Per una linea a 1.2Gbps il Wrap Around si ha in circa 28s
- ▶ **Protection Against Wrapped Sequence (PAWS)**
 - ▶ Viene utilizzata l'opzione **Timestamp** che aggiunge un ulteriore campo a 32 bit che permette di individuare l'ordine con cui sono stati generati i pacchetti
 - ▶ In base al Timestamp è possibile ordinare correttamente i segmenti anche in presenza di reinizializzazione dei numeri di sequenza

Chiusura della connessione

- ▶ La connessione è full-duplex e le due direzioni devono essere chiuse indipendentemente
- ▶ Se l'ack di un messaggio FIN si perde, l'host mittente chiude comunque la connessione dopo un timeout



Un esempio di chiusura

- ▶ Chiusura Telnet da 10.6.1.9
porta client 4548 - porta server 23 (telnet)

FIN

10.6.1.9.4548 > 10.6.1.2.23: F 2115515449:2115515449(0) ack
1220480986 win 32120 (DF)

10.6.1.2.23 > 10.6.1.9.4548: . ack 2115515450 win 32120 (DF)

10.6.1.2.23 > 10.6.1.9.4548: F 1220480986:1220480986(0) ack
2115515450 win 32120 (DF)

10.6.1.9.4548 > 10.6.1.2.23: . ack 1220480987 win 32120 (DF)

Tipologie di flusso dati

- ▶ Il flusso di dati ha caratteristiche che dipendono dall'applicazione
 - ▶ **Flussi a bassa velocità**
 - ▶ I dati da trasmettere sono prodotti da un utente (pochi byte/s)
 - ▶ E' richiesta interattività, ovvero l'utente deve avere il feedback sui input con una bassa latenza
 - ▶ I segmenti inviati portano pochi dati perché non si possono usare tecniche di bufferizzazione su lunghi intervalli di tempo
 - ▶ Si devono adottare tecniche per massimizzare il payload
 - ▶ **Flussi ad alta velocità**
 - ▶ I dati sono il prodotto di un'elaborazione
 - ▶ I dati possono essere inviati alla massima velocità perché il buffer di invio è "sempre" pieno
 - ▶ Occorre regolare il flusso per evitare di saturare la capacità del ricevente (riempire il buffer di ricezione) o della rete (riempire le code dei router)

Flussi di dati interattivi

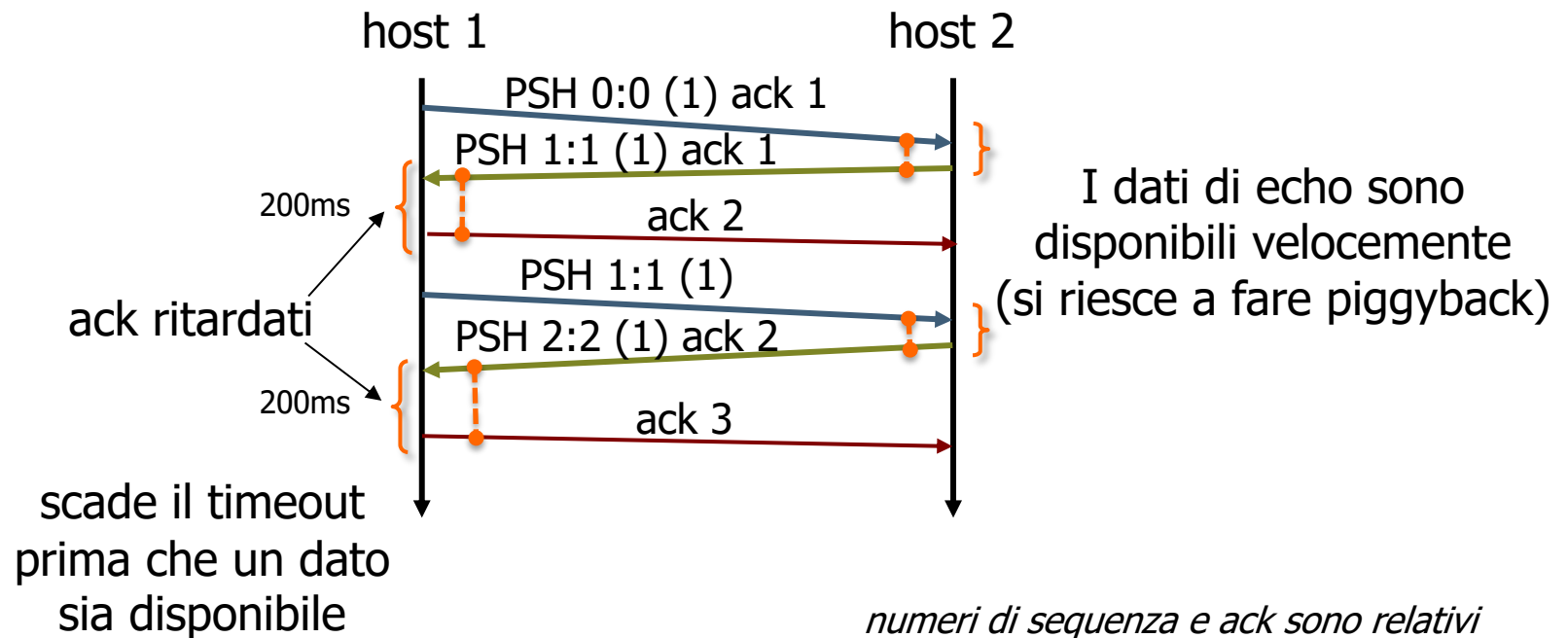
- ▶ Una connessione interattiva è caratterizzata da una **bassa frequenza dei dati** da trasmettere
 - ▶ I dati sono generati dall'azione di un utente umano
 - ▶ Sessione di terminale remoto (es. telnet o ssh)
 - ▶ L'applicazione è interattiva
 - ▶ Il sistema di trasmissione non può introdurre latenze che riducono la percezione dell'interattività
 - ▶ Non si possono accumulare i dati ma occorre inviare segmenti piccoli
- ▶ Il carico utile (**payload**) dei segmenti TCP è di pochi byte
 - ▶ Il 90% dei segmenti telnet porta circa 10 byte
 - ▶ Nel caso limite si ha un segmento per ogni carattere battuto
 - In genere si ha anche un echo del carattere battuto da parte del destinatario con un ulteriore segmento con un solo carattere

Esempio: sessione telnet

- ▶ Quando l'utente batte un tasto in una sessione telnet
 - ▶ Segmento dal client col carattere battuto
 - ▶ $20\text{IP} + 20\text{TCP} + 1\text{byte} = 41\text{byte}$
 - ▶ Segmento di ack dal server al client
 - ▶ Segmento di echo dal server
 - ▶ Segmento di ack dal client
- si possono unire...
- ▶ In totale si userebbero 162 (122) byte in 4 (3) segmenti TCP per 1 carattere!!

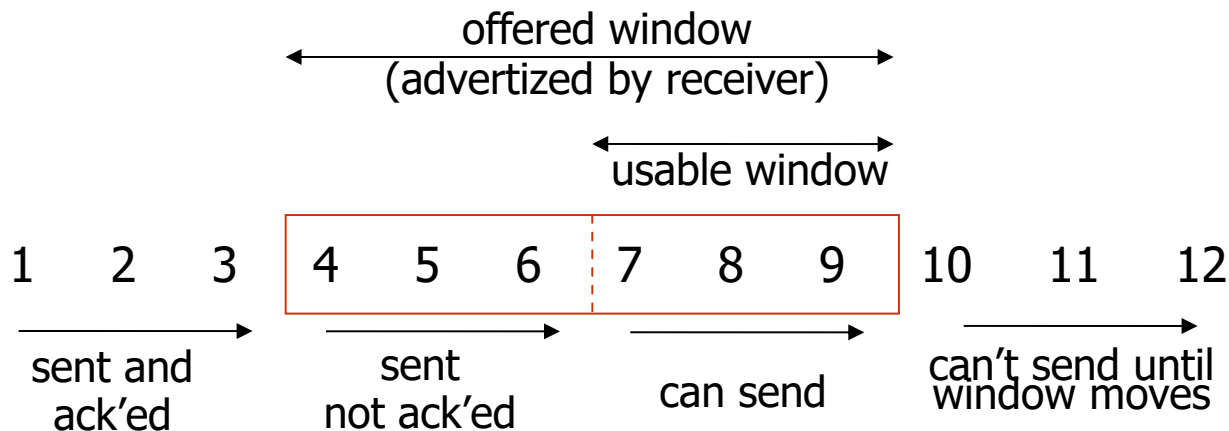
Ack ritardati

- ▶ Normalmente il TCP non invia un ack istantaneamente ma ritarda l'invio sperando di avere dati da spedire con l'ack (**ACK piggyback**)
- ▶ Molte implementazioni usano un ritardo di 200ms



Regolazione del flusso di dati

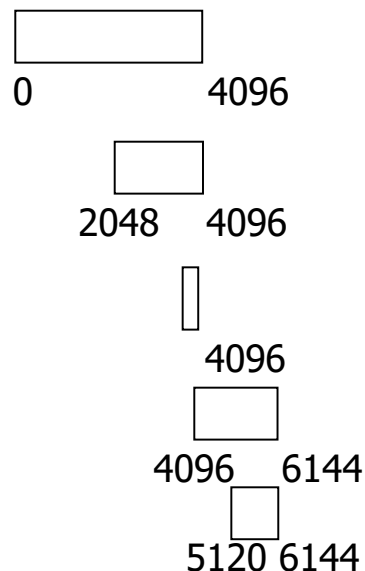
- ▶ Viene utilizzato un protocollo a **finestra scorrevole** (**sliding window**) con dimensione variabile
 - ▶ Il ricevente indica la dimensione della finestra (quantità dei dati) che può gestire in un dato momento
 - ▶ La dimensione dipende dallo spazio disponibile nel buffer di ricezione
 - ▶ L'evoluzione della dimensione dipende dalla capacità dell'applicativo destinatario di elaborare e quindi leggere i dati dal buffer di ricezione
- ▶ La finestra di dati trasmissibili ancora senza aspettare l'ack è ottenuta dall'ampiezza della finestra e dal numero dell'ultimo byte ricevuto
 - ▶ Il mittente può continuare a inviare dati senza attendere l'ack fino a che non esaurisce la finestra disponibile



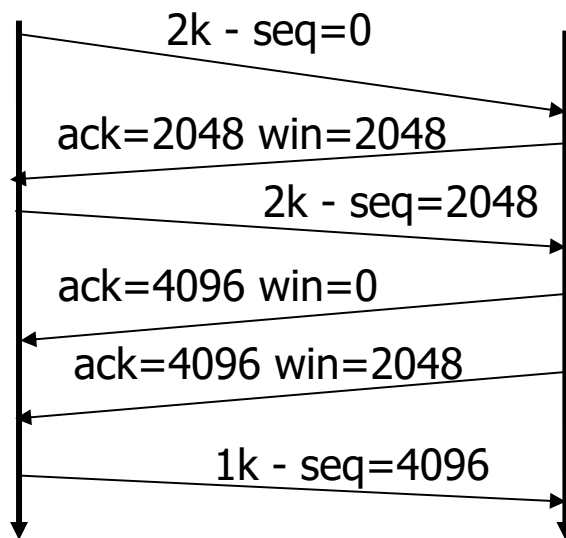
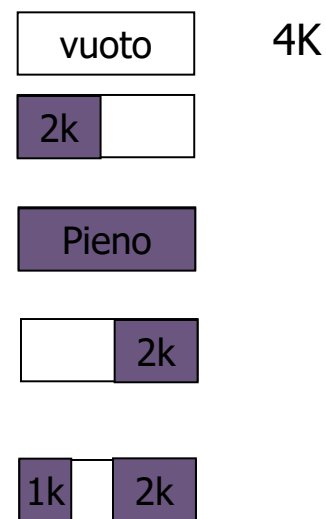
Finestra scorrevole

- ▶ Se il ricevente è saturo indica una finestra 0 e il mittente non può trasmettere dati
 - ▶ Il mittente può inviare periodicamente un segmento di un byte per forzare il destinatario a indicare il prossimo byte atteso e l'ampiezza della finestra (per non rimanere in attesa infinita se si perdono pacchetti)
 - ▶ Non conviene notificare subito la disponibilità della finestra se è piccola (**silly window syndrome**)

Finestra del mittente



buffer del ricevente



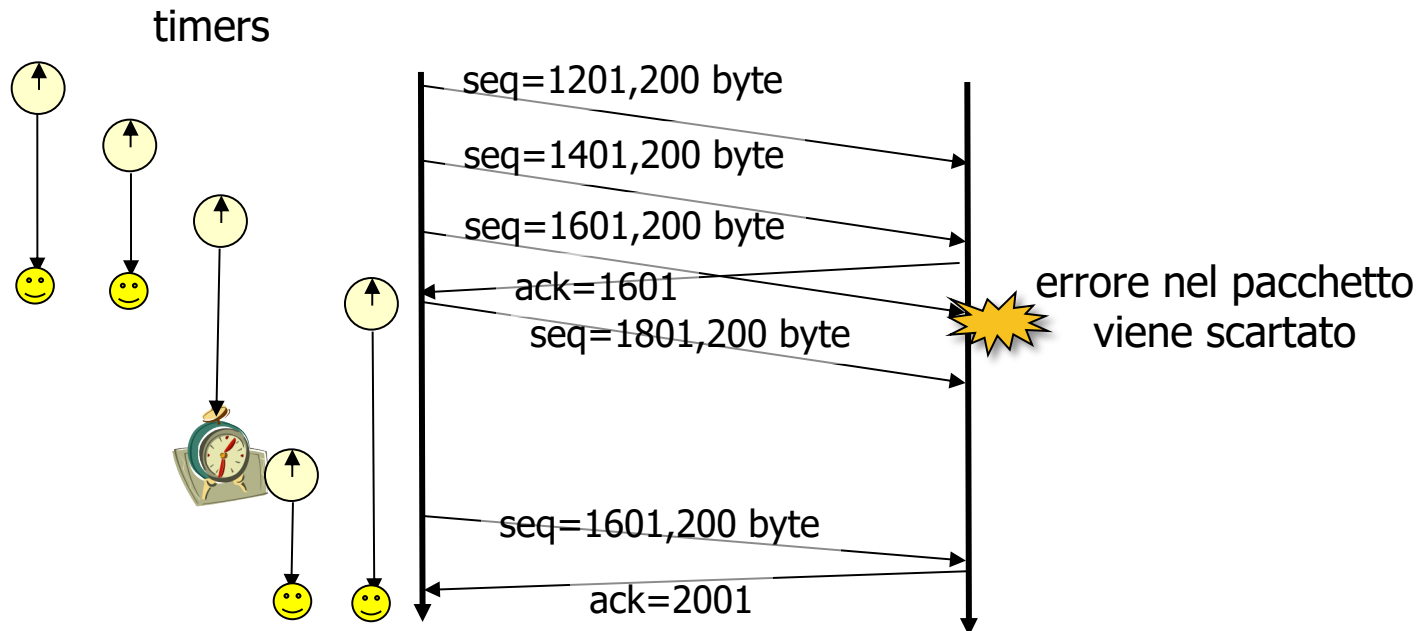
Un esempio di trasmissione

- ▶ Connessione Telnet fra da 10.6.1.9 a 10.6.1.2 catturata con tcpdump porta client 4548 - porta server 23 (telnet)
 - ▶ La finestra offerta è di circa 32k e pari alla dimensione del buffer di ricezione

```
10.6.1.9.4548 > 10.6.1.2.23: P 2115515279:2115515306(27) ack 1220480854 win 32120
10.6.1.2.23 > 10.6.1.9.4548: . ack 2115515306 win 32120
10.6.1.2.23 > 10.6.1.9.4548: P 1220480854:1220480866(12) ack 2115515306 win 32120
10.6.1.9.4548 > 10.6.1.2.23: . ack 1220480866 win 32120
10.6.1.2.23 > 10.6.1.9.4548: P 1220480866:1220480905(39) ack 2115515306 win 32120
10.6.1.9.4548 > 10.6.1.2.23: P 2115515306:2115515443(137) ack 1220480905 win 32120
10.6.1.2.23 > 10.6.1.9.4548: P 1220480905:1220480908(3) ack 2115515443 win 32120
10.6.1.9.4548 > 10.6.1.2.23: P 2115515443:2115515446(3) ack 1220480908 win 32120
10.6.1.2.23 > 10.6.1.9.4548: . ack 2115515446 win 32120
```

Gestione della perdita di segmenti

- ▶ Il mittente ha un timeout di attesa dell'ack per i pacchetti spediti
 - ▶ Se il timeout scade il pacchetto viene ritrasmesso
 - ▶ Permette di gestire la perdita di pacchetti (non arrivati o arrivati con errori)



TCP timeout e ritrasmissione

- ▶ TCP utilizza un timeout di attesa dell'ack dopo di che provvede alla ritrasmissione dei dati
 - ▶ Il problema è determinare il valore del timeout migliore (i ritardi possono essere molto variabili nel tempo sulla rete)
 - ▶ Se il timeout è troppo piccolo si fanno ritrasmissioni inutili
 - ▶ Se il timeout è troppo elevato si avranno ritardi di trasmissione
- 
- ▶ Si utilizza un algoritmo di stima del migliore timeout basato sulla misura del Round-Trip Time (RTT)

Stima del timeout

- ▶ Per ogni connessione si tiene una stima di RTT, aggiornandola per ogni pacchetto con

$$RTT_i = \alpha RTT_{i-1} + (1 - \alpha) T_{rtt}(pkt_i)$$

- ▶ Si stima poi la deviazione media

$$D_i = \alpha D_{i-1} + (1 - \alpha) |RTT_i - T_{rtt}(pkt_i)|$$

- ▶ E si sceglie

$$\text{timeout} = RTT + 4 * D$$

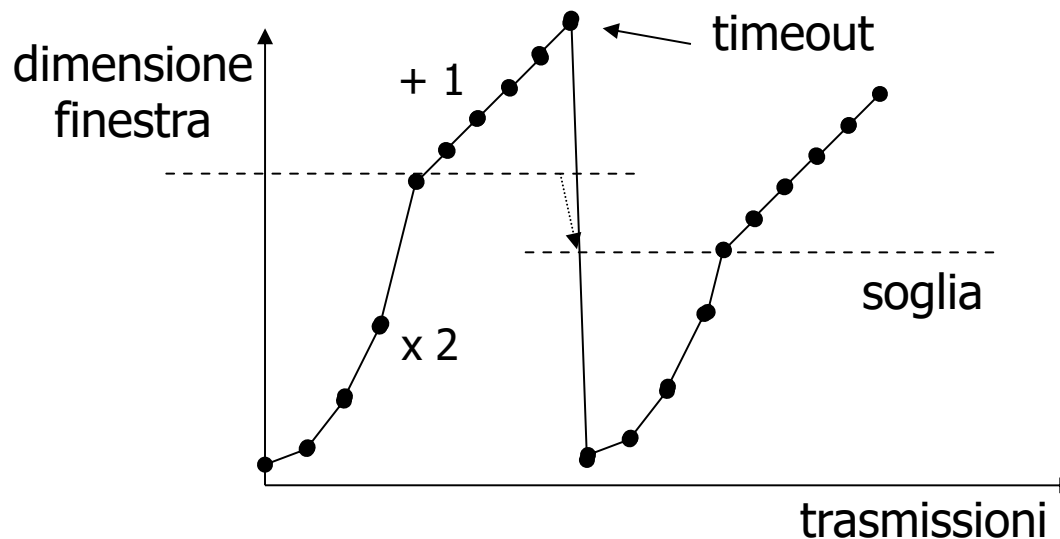
- ▶ Ci sono altre soluzioni (es. algoritmo di Karn – nella stima si usano solo i pacchetti non ritrasmessi)

Controllo di congestione

- ▶ Il TCP adatta la velocità di trasmissione alla capacità della rete
 - ▶ I pacchetti possono essere distrutti dai router se si riempiono i buffer di trasmissione/ricezione
 - ▶ Per i pacchetti distrutti nella rete non si riceve l'ack e quindi vengono ritramessi
 - ▶ La ritrasmissione è un potenziale fattore per incrementare ulteriormente la congestione
- ▶ Si utilizza una **finestra di congestione** per il mittente
 - ▶ La finestra effettiva è la minima fra quella di congestione (definita dalla rete) e quella di ricezione (definita dallo stato del buffer del ricevente)

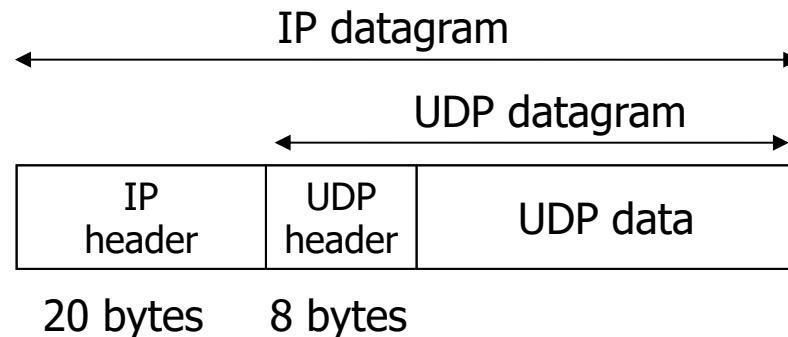
Finestra di congestione

- ▶ La dimensione della finestra di congestione è adattata dinamicamente
 - ▶ Cresce se non scade il timeout di trasmissione
 - ▶ Viene ridotta se scade il timeout di ritrasmissione

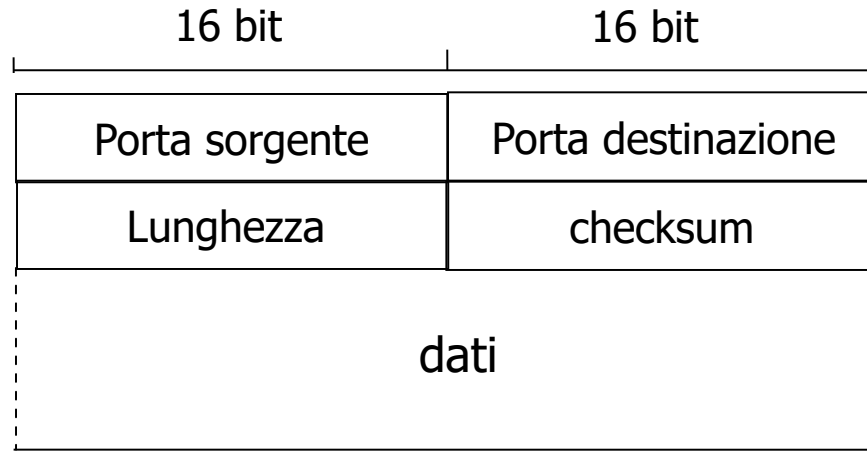


UDP

- ▶ Ogni operazione di output produce esattamente un datagram UDP che comporta l'invio di un datagram IP
 - ▶ UDP non garantisce affidabilità di consegna
 - ▶ UDP implementa il **controllo di integrità** con un codice a rilevazione di errore (checksum)
 - ▶ Se il datagram eccede la MTU (Maximum Transfer Unit) della rete fisica, esso viene frammentato dal livello di rete
 - ▶ Richiede **meno overhead** di una connessione TCP
 - ▶ Ha un header di dimensioni minori (8 byte invece di 20)
 - ▶ Non richiede la procedura di connessione
 - ▶ Non prevede l'invio di un ack



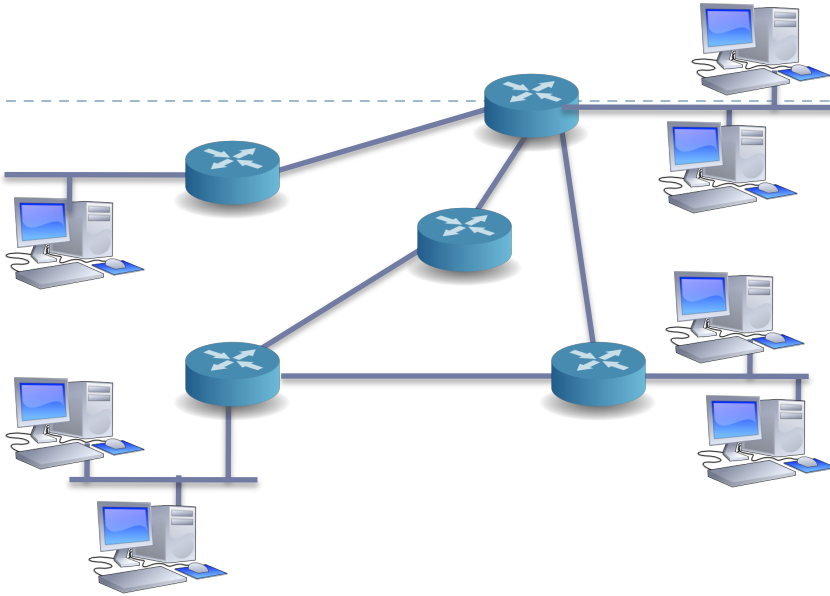
Header UDP



- ▶ La **porta destinazione** individua il processo destinatario (socket)
- ▶ La **porta sorgente** (insieme all'indirizzo IP) fornisce le informazioni necessarie ad inviare una eventuale risposta
- ▶ La **lunghezza in byte** comprende sia i dati che l'header (□8)
 - ▶ Considerando l'header IP (20 byte) un pacchetto UDP può contenere fino a 65507 byte = 65535-20-8
- ▶ Il **checksum** comprende anche uno pseudoheader che contiene le informazioni IP (indirizzi IP, tipo di protocollo, dimensione)

Uso di UDP

- ▶ Implementazione esplicita del protocollo di scambio dati
 - ▶ **Perdita della richiesta**
 - ▶ L'applicazione che invia una richiesta deve prevedere il caso in cui si verifica un errore e la risposta non arriva
 - ▶ Ritrasmissione della richiesta allo scadere di un timeout di attesa
 - ▶ **Associazione richiesta-risposta**
 - ▶ Identificativo della richiesta da inviare nella risposta
 - ▶ La rete non garantisce che le risposte arrivino nello stesso ordine delle richieste
 - ▶ **Controllo sull'invio/ricezione dei dati**
 - ▶ **Nessuna bufferizzazione.** I datagram UDP sono inviati direttamente al livello di rete
 - ▶ **Nessun controllo di flusso e di congestione.** I datagram sono inviati alla velocità con cui sono generati
 - ▶ **Nessun ack per ogni pacchetto inviato.** L'ack è implicito nella ricezione di una risposta
 - L'ack può non essere necessario per le applicazioni in cui è tollerata la perdita di dati (es. streaming in tempo reale)

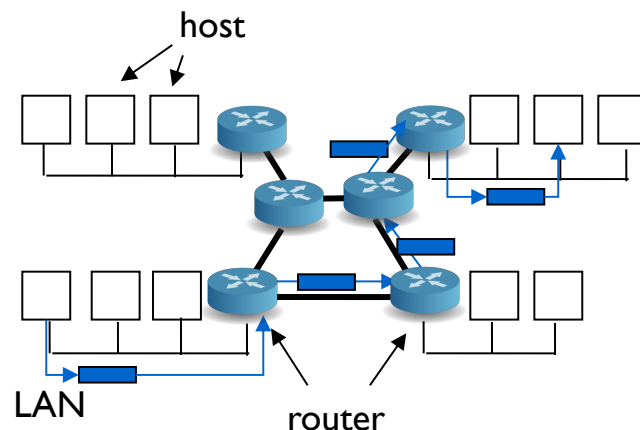


Reti di Calcolatori

Network Layer & IP

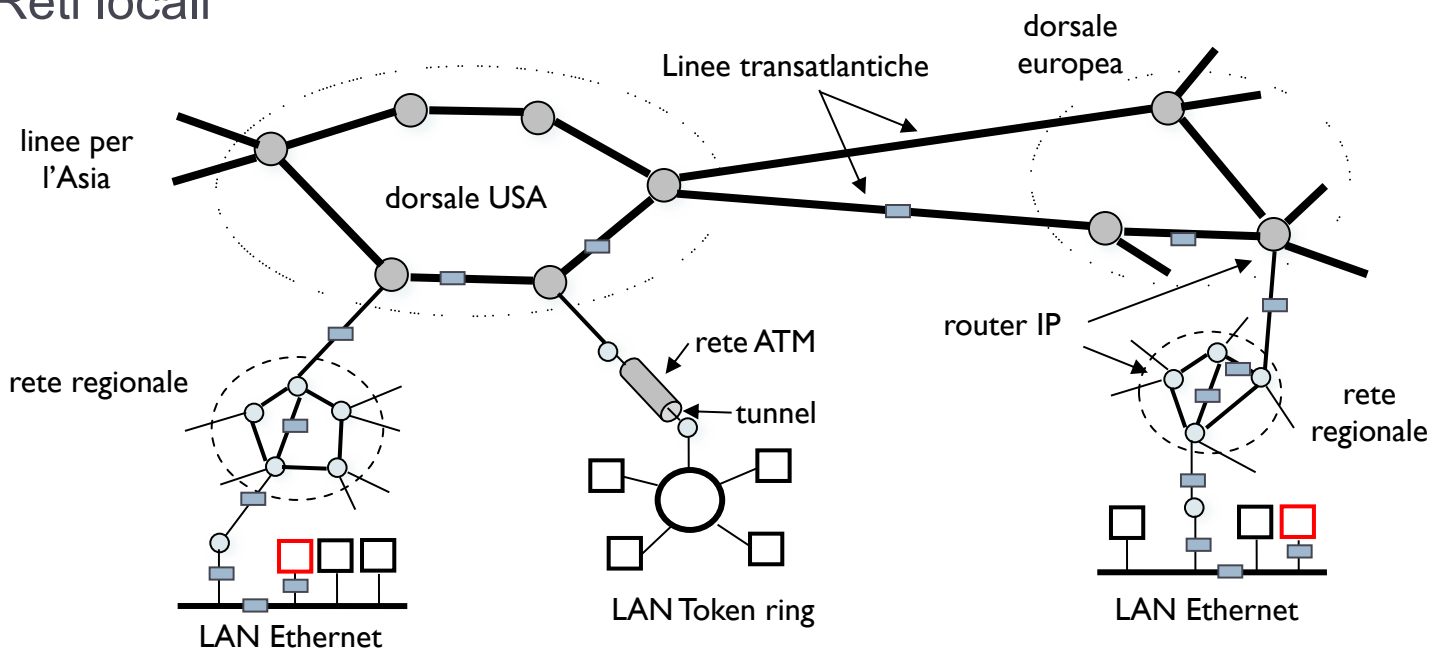
Il livello di rete

- ▶ Trasmissione **host-to-host** di pacchetti dati
- ▶ Connessione di reti fisiche basate su tecnologie anche diverse (**internetworking**)
 - ▶ Definizione di uno schema di **indirizzamento logico globale** degli host (indirizzi pubblici di rete)
 - ▶ Uso di nodi intermedi di instradamento (**router**) per connettere le reti fra loro
 - ▶ Ogni nodo router connette due o più reti fisiche
 - ▶ La rete ha una struttura a grafo
 - ▶ L'instradamento è possibile se è nota la topologia del grafo
 - ▶ I router instradano i pacchetti sulle loro linee di uscita in modo da indirizzarli verso la destinazione
 - ▶ I percorsi sono scelti in modo ottimale (es. per evitare sovraccarichi)



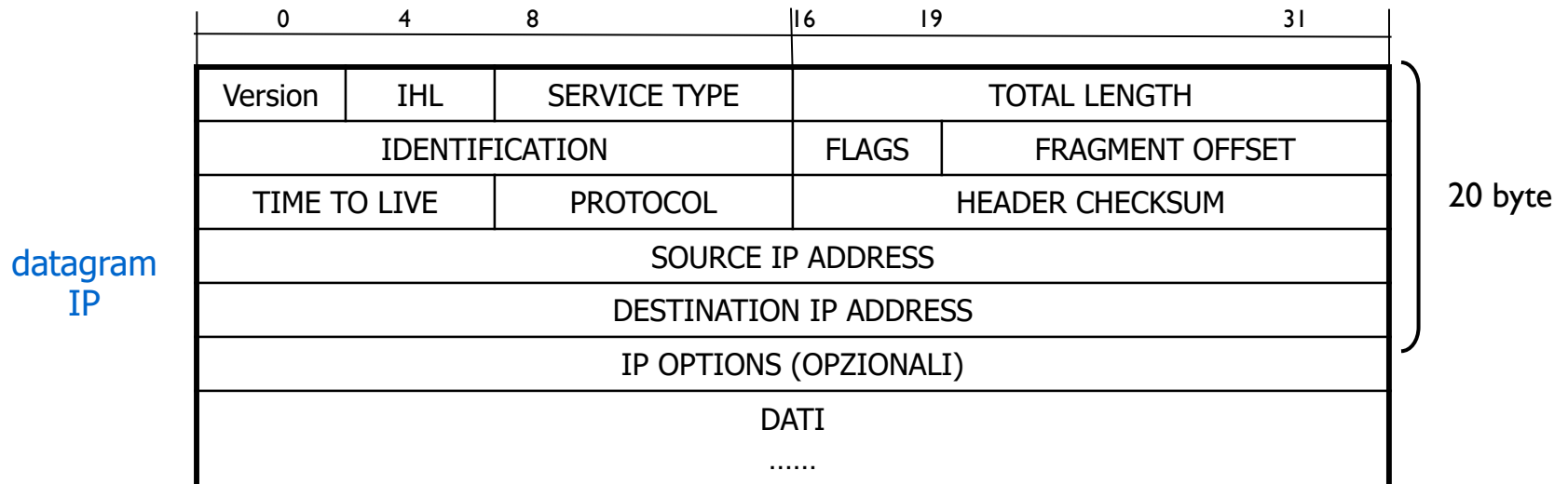
Internet

- ▶ E' una collezione di sottoreti connesse insieme
 - ▶ Backbone (dorsali) ad alta capacità con router veloci
 - ▶ Reti regionali
 - ▶ Reti locali



Internet Protocol (IP)

- ▶ Standardizzato in RFC 791 è la base di Internet
- ▶ E' un protocollo senza connessione **best-effort**
 - ▶ I datagrammi contengono l'identità della destinazione
 - ▶ Ogni datagramma viene spedito/gestito indipendentemente
 - ▶ Non è garantita la consegna



Header IP – word 1

Version	IHL	SERVICE TYPE	TOTAL LENGTH	
IDENTIFICATION			FLAGS	FRAGMENT OFFSET
TIME TO LIVE		PROTOCOL	HEADER CHECKSUM	

- ▶ **Version** permette di mantenere compatibilità fra le versioni
- ▶ **IHL** è la lunghezza del preambolo in parole da 32 bit (su 4 bit)
 - ▶ Al massimo il preambolo è $15 \times 4 = 60$ byte
 - ▶ Il campo opzionale può contenere fino a 40 byte (60-20)
- ▶ **SERVICE TYPE** individua il tipo di servizio desiderato in termini di affidabilità, velocità e ritardo
 - ▶ Può influenzare le scelte di routing
- ▶ **TOTAL LENGTH** riguarda l'intero datagram (header e dati)
 - ▶ La lunghezza massima del datagram è di $2^{16}-1=65535$ byte

Header IP – word 2

Version	IHL	SERVICE TYPE	TOTAL LENGTH	
IDENTIFICATION			FLAGS	FRAGMENT OFFSET
TIME TO LIVE		PROTOCOL	HEADER CHECKSUM	

- ▶ Gestione della **frammentazione** dei datagram
 - ▶ Può essere necessaria se si invia un datagram su reti con protocolli datalink basati su pacchetti di lunghezza massima minore (es. Ethernet con 1500 byte/pacchetto)
 - ▶ I frammenti (eccetto l'ultimo) sono multipli di 8 byte
 - ▶ **IDENTIFICATION** è l'identificatore del datagram a cui appartiene il frammento
 - ▶ **FRAGMENT OFFSET** (13 bit) identifica il numero d'ordine del frammento
 - ▶ Si possono avere un massimo di 8192 frammenti
 - ▶ **FLAGS** permettono di indicare che il pacchetto non è frammentabile (Don't Fragment) e che ci sono ulteriori frammenti (More Fragments)

Header IP – word 3

Version	IHL	SERVICE TYPE	TOTAL LENGTH	
IDENTIFICATION			FLAGS	FRAGMENT OFFSET
TIME TO LIVE		PROTOCOL	HEADER CHECKSUM	

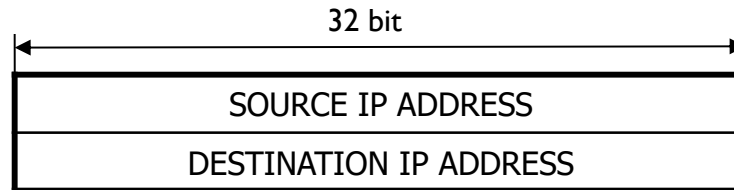
- ▶ **TIME TO LIVE** è un contatore utilizzato per limitare la vita dei pacchetti
 - ▶ E' in secondi (0-255)
 - ▶ Dovrebbe essere decrementato ad ogni salto (hop) su un router (o se rimane per più tempo in coda)
 - ▶ Quando il contatore si azzerà il pacchetto è scartato e si invia una notifica al mittente
- ▶ **PROTOCOL** individua il protocollo di trasporto (es. TCP o UDP)
- ▶ **HEADER CHECKSUM** verifica solo l'header
 - ▶ deve essere ricalcolato ad ogni hop se cambia il campo TIME TO LIVE

Header IP Options

Version	IHL	SERVICE TYPE	TOTAL LENGTH	
IDENTIFICATION			FLAGS	FRAGMENT OFFSET
TIME TO LIVE		PROTOCOL	HEADER CHECKSUM	
SOURCE IP ADDRESS				
DESTINATION IP ADDRESS				
IP OPTIONS (OPZIONALI)				

- ▶ Ogni opzione inizia con 1 byte che la identifica
 - ▶ Può essere seguito da 1 byte che ne indica la lunghezza
 - ▶ Seguono i dati relativi all'opzione attivata
 - ▶ Alcuni esempi
 - ▶ [Strict source routing](#) – specifica il percorso dalla sorgente alla destinazione con una sequenza di indirizzi IP
 - ▶ [Record route](#) – forza i router ad aggiungere i loro IP nel campo opzione del pacchetto (max 9 router)
 - ▶ [Timestamp](#) – aggiunge oltre all'IP dei router sul percorso anche un timestamp a 32 bit

Indirizzamento IP



- ▶ L'indirizzo IP a 32 bit identifica **univocamente** un dispositivo (host o router) sulla rete
 - ▶ Gli indirizzi IP **pubblici** sulla rete Internet sono assegnati dal **NIC** (Network Information Center)
 - ▶ Due dispositivi connessi alla rete Internet non possono avere lo stesso IP
- ▶ L'indirizzo può essere scomposto in due parti
 - ▶ Indirizzo di rete
 - ▶ Indirizzo di host nella rete

Classi di indirizzi IP

- In base alla divisione rete – host sono state definite classi di indirizzi standard

	0 1 2 3 4	8	16	31	
classe A	0	rete	host		1.0.0.0 127.255.255.255
classe B	1 0	rete		host	128.0.0.0 191.255.255.255
classe C	1 1 0	rete		host	192.0.0.0 223.255.255.255
classe D	1 1 1 0	indirizzo multicast			224.0.0.0 239.255.255.255
classe E	1 1 1 1	riservato per scopi futuri			240.0.0.0 247.255.255.255

Dimensioni delle reti

classe	bit nel prefisso	numero massimo di reti	bit nel suffisso	numero massimo di host per rete
A	7	128	24	16777216
B	14	16384	16	65536
C	21	2097152	8	256

- ▶ Non tutte le configurazioni sono indirizzi utilizzabili per host
 - ▶ Sono definiti indirizzi speciali
- ▶ Dimensioni delle reti
 - ▶ A grosse organizzazioni
 - ▶ B medie organizzazioni
 - ▶ C piccole organizzazioni

Indirizzi speciali

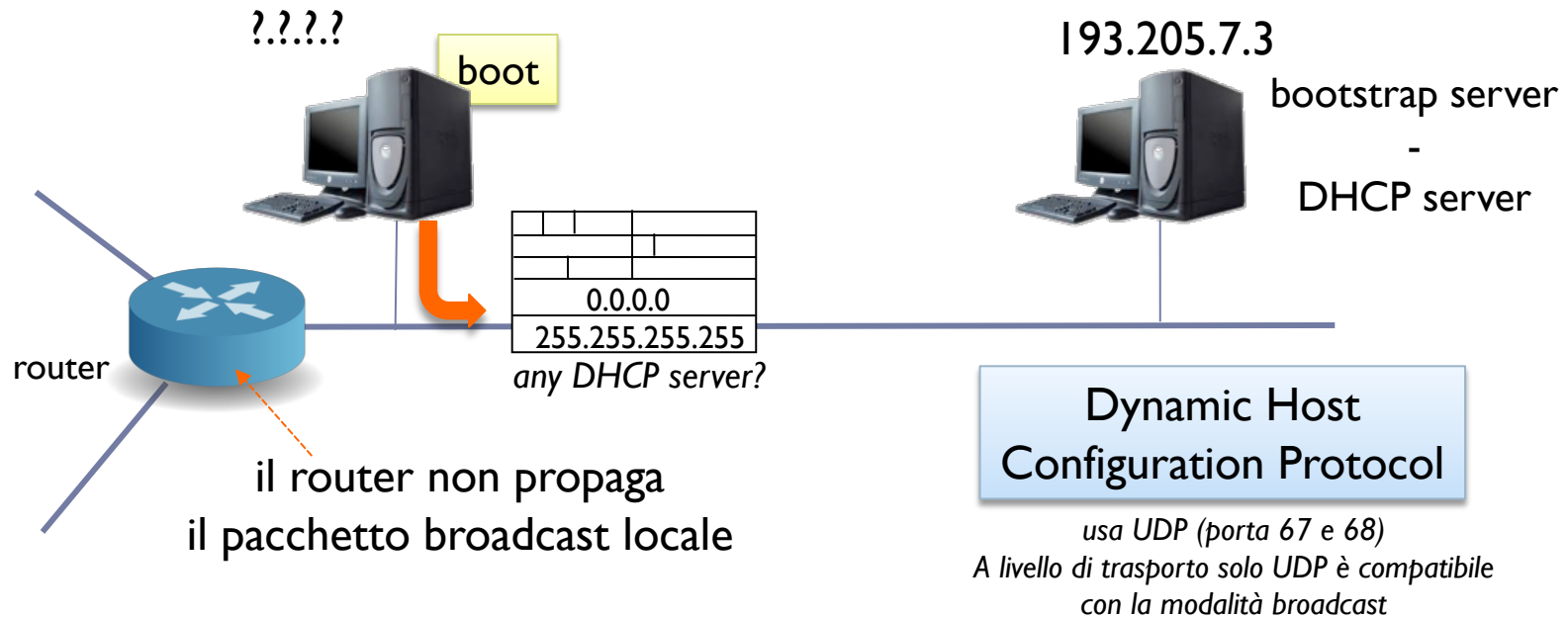
prefisso	suffisso	tipo di indirizzo	scopo
tutti 0	tutti 0	questo computer	utilizzato nel boot
rete	tutti 0	la rete	identifica una rete
rete	tutti 1	broadcast diretto	broadcast su una rete specifica
tutti 1	tutti 1	broadcast limitato	broadcast sulla rete locale
127	qualsiasi	loopback	test

- ▶ I pacchetti di **broadcast** limitato sono confinati alla sottorete dal router che non li propaga all'esterno
- ▶ Con gli indirizzi di **loopback** il pacchetto non viene immesso sulla rete
 - ▶ Il pacchetto viene smistato localmente nel modulo del livello rete (non viene passato al livello fisico)

0.0.0.0 e 255.255.255.255

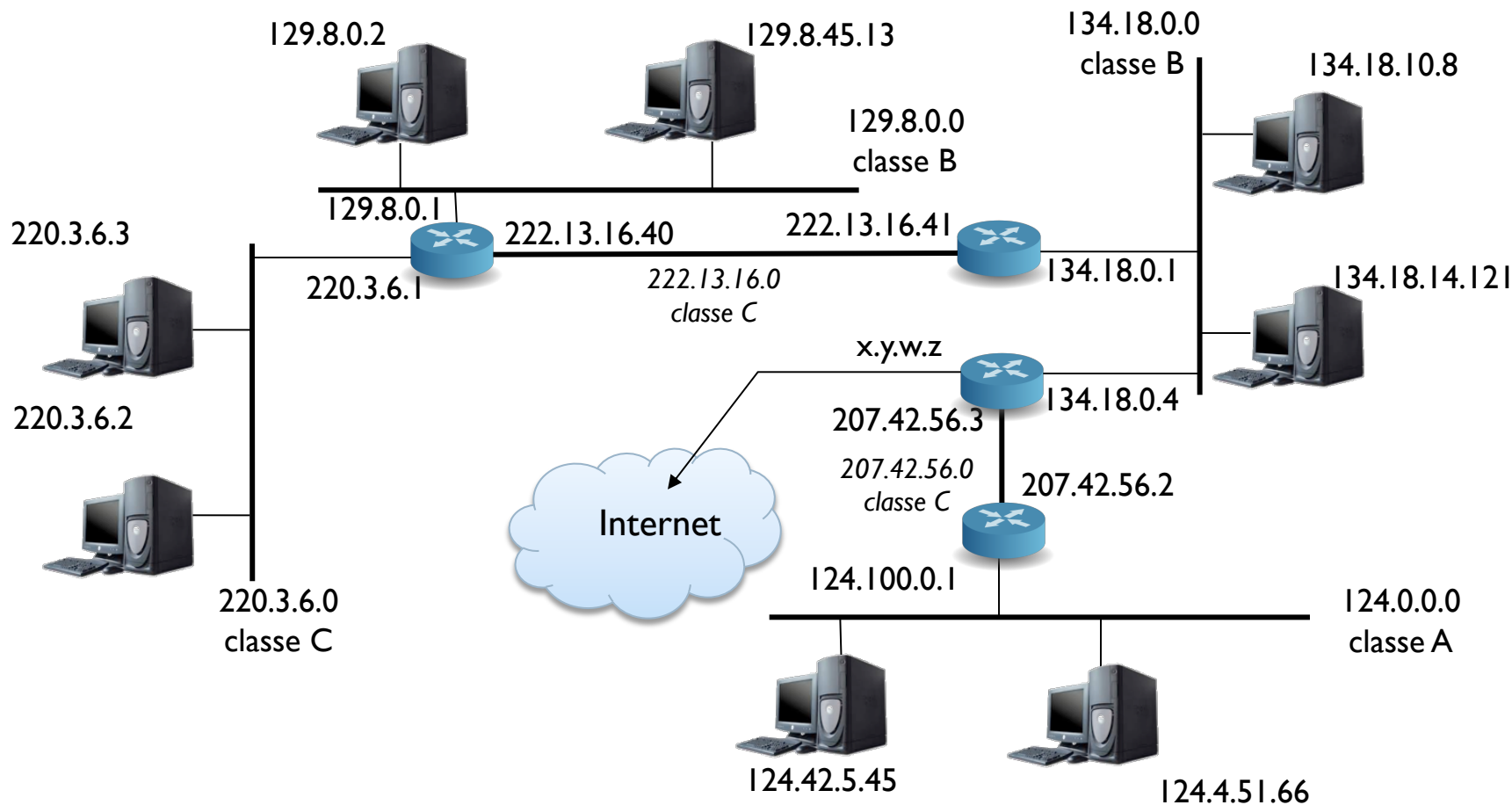
0.0.0.0 – questo host su questa rete

255.255.255.255 – broadcast limitato sulla rete corrente



- ▶ La modalità broadcast UDP può essere utilizzata per fare **discovery** di servizi su una rete locale

Un esempio di inter-rete



Indirizzi privati

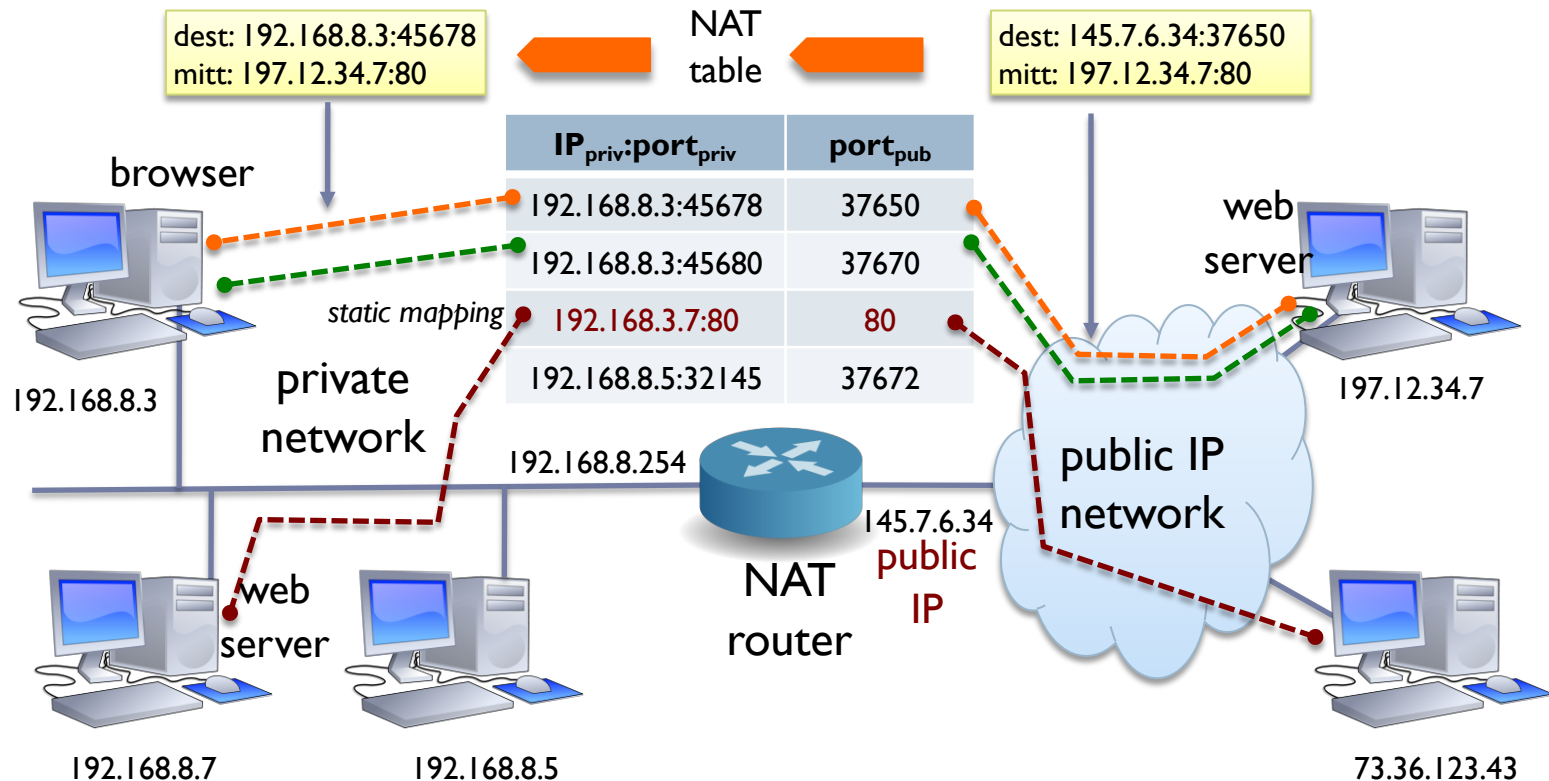
- ▶ Sono stati riservati indirizzi per reti non connesse direttamente alla rete Internet
 - ▶ Sono indirizzi locali e non pubblici
 - ▶ Devono rimanere confinati alla sottorete
 - ▶ La rete può accedere alla rete Internet pubblica per mezzo di un gateway che implementa una tecnica di **Network Address Translation (NAT)**

classe	rete	numero reti
A	10.0.0.0	1
B	Da 172.16 a 172.31	16
C	Da 192.168.0 a 192.168.255	256

NAT

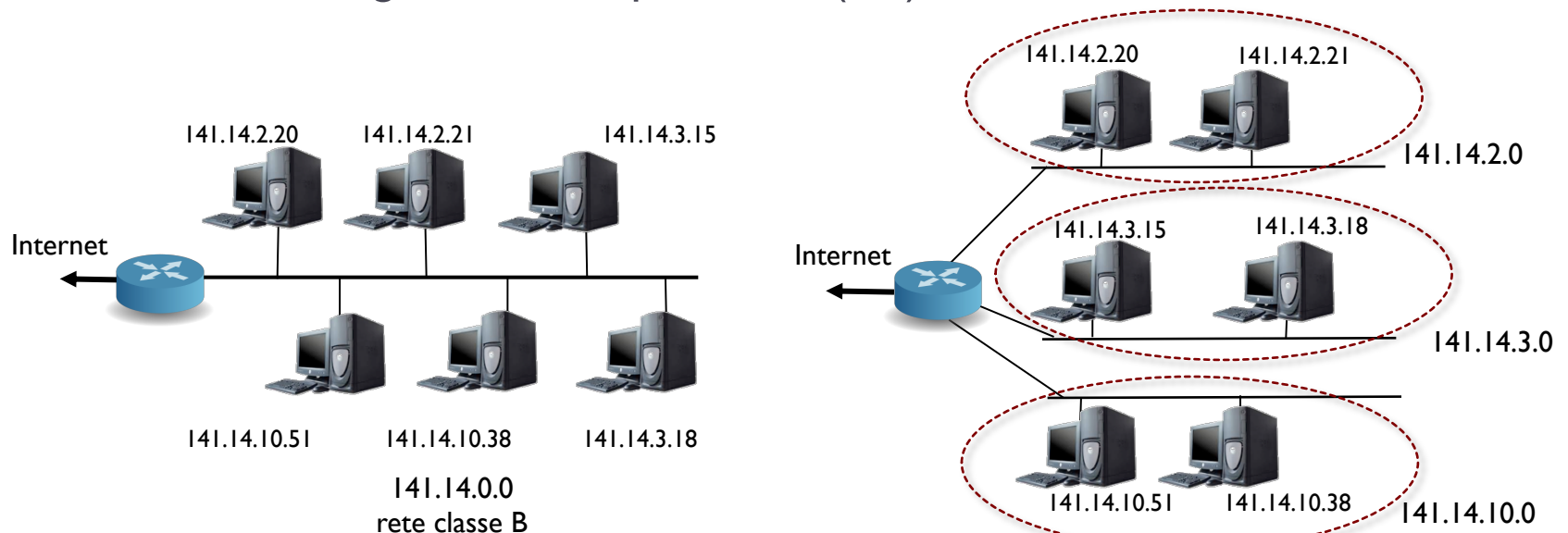
- ▶ Permette l'accesso alla rete pubblica da una rete privata
 - ▶ Il mapping è **one-to-many** ovvero tutti gli host della rete privata accedono alla rete pubblica condividendo un unico IP pubblico
 - ▶ Il mapping richiede di modificare le intestazioni sia a livello di rete che a livello di trasporto (TCP/UDP)
 - ▶ Il meccanismo NAT permette la comunicazione in automatico attraverso un router quando la comunicazione origina dall'interno
 - Le coppie $IP_{priv}:port_{priv}$ sono mappate in coppie $IP_{pub}:port_{pub}$ dal router NAT
 - Il router mantiene una tabella NAT che permette di instradare all'host/porta della rete privata i pacchetti ricevuti dall'esterno
 - ▶ E' possibile anche configurare manualmente le tabelle del NAT per rendere pubblici dei servizi in esecuzione su host della rete privata (**port forwarding**)
 - Ad esempio si può associare la porta 80 del router alla porta 80 di un host interno per rendere accessibile dalla rete pubblica il server Web

Rete privata con NAT



Subnetting

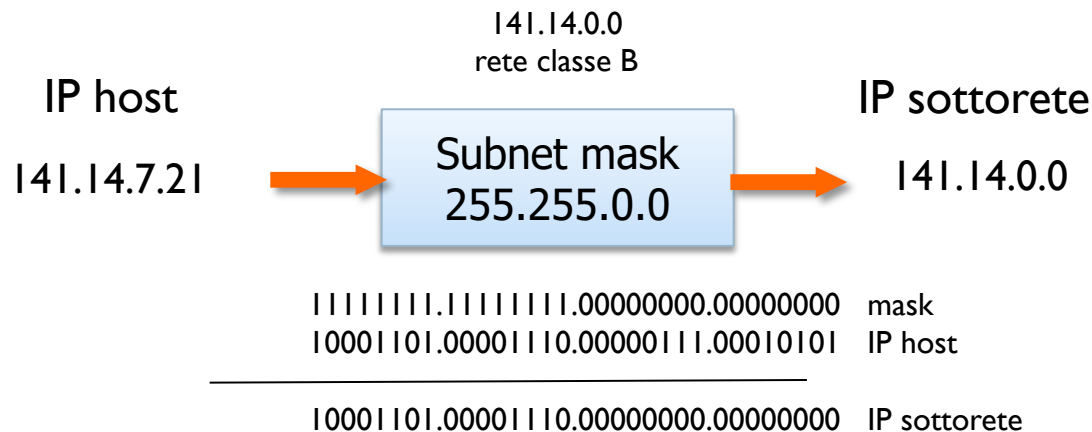
- ▶ Le classi standard definiscono una gerarchia a 2 livelli
 - ▶ Rete
 - ▶ Host
- ▶ Può essere utile suddividere la rete in sottoreti per gestire separatamente un numero minore di host
 - ▶ Si crea una gerarchia a più livelli (>2)



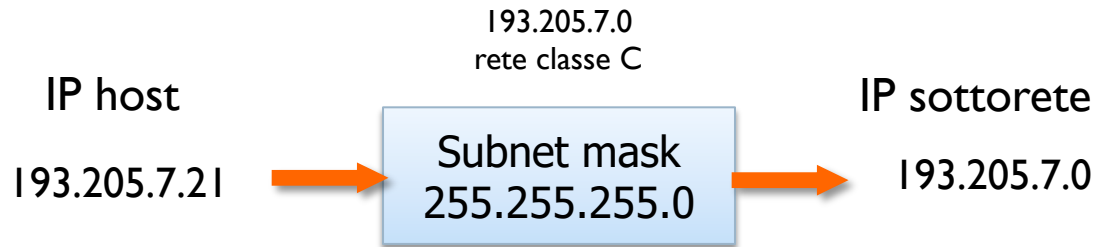
Subnet Mask

- ▶ Si utilizza un maschera di bit a 1 nelle posizioni corrispondenti alla parte di indirizzo riferita alla rete
- ▶ Permette di verificare se un indirizzo destinazione appartiene alla propria sottorete calcolando l'AND bit a bit

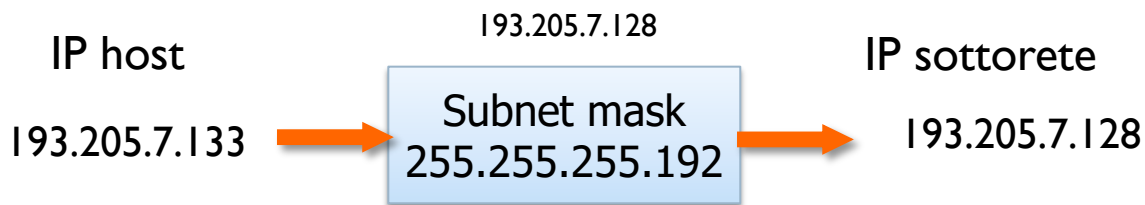
```
if((address & subnet_mask) == mynetwork_address)
{ .. stessa sottorete .. }
```



Subnet mask per classe C



- ▶ Si può suddividere una rete di classe C in sottoreti
 - ▶ Ad esempio sottoreti di 62 host (6 bit) – 4 sottoreti possibili



11111111.11111111.11111111.11000000	mask
11000001.11001101.00000111.10000101	IP host
11000001.00001110.00000111.10000000	IP sottorete

Subnet mask per sottoreti

- ▶ Le subnet mask per le classi standard sono
 - ▶ classe A 255.0.0.0
 - ▶ classe B 255.255.0.0
 - ▶ classe C 255.255.255.0
- ▶ Se si creano sottoreti, la sottorete è individuata da bit aggiuntivi nella parte prevista per l'host

n bit	bit netmask	valore decimale	numero sottoreti	indirizzi sottorete	range indirizzi
1	10000000	128	2	128	0-127, 128-255
2	11000000	192	4	64	0-63, 64-127, 128-191, 192-255
3	11100000	224	8	32	0-31, 32-63, ..., 192-223, 224-255
4	11110000	240	16	16	0-15, 16-31, ..., 224-239, 240-255
5	11111000	248	32	8	0-7, 8-15, ..., 240-247, 248-255
6	11111100	252	64	4	0-3, 4-7, ..., 248-251, 252-255
7	11111110	254	128	2	0-1, 2-3, ..., 252-253, 254-255
8	11111111	255	256	1	0, 1, 2, ..., 253, 254, 255

Configurare la scheda di rete

- ▶ Su Unix il comando **ifconfig** configura la scheda di rete
 - ▶ La abilita/disabilita (up/down)
 - ▶ Definisce i parametri hardware (es. irq, io_addr)
 - ▶ Definisce i parametri della connessione di rete (indirizzo, netmask,..)
 - ▶ Abilita funzionalità particolare (modo promiscuo – promisc)

```
[host]# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:06:29:55:15:4C
          inet addr:193.205.7.146  Bcast:193.205.7.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST MTU:1500  Metric:1
          RX packets:24559513 errors:0 dropped:0 overruns:0 frame:0
          TX packets:32527633 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          Interrupt:16 Base address:0x2200

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:3924  Metric:1
          RX packets:3803246 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3803246 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
```

Configurazione IP

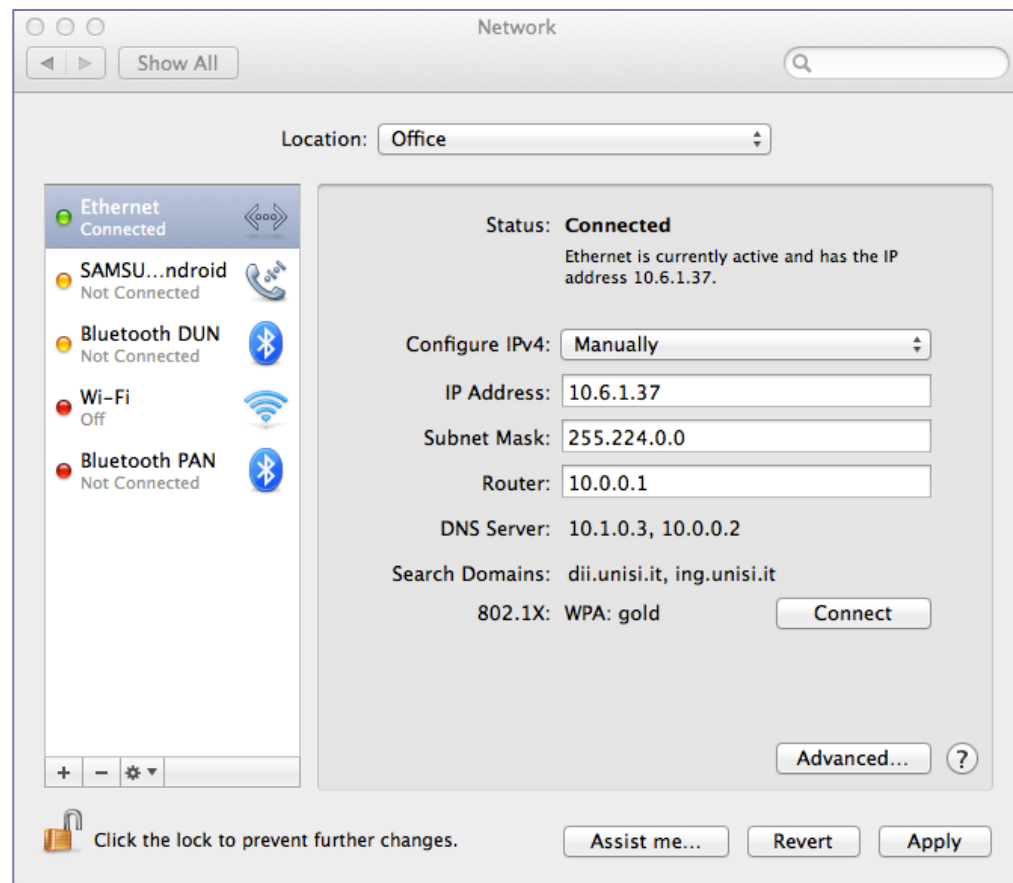
Statistiche d'uso

Configurazione HW

Max Transfer Unit

Configurazione da pannello

- ▶ Il pannello delle preferenze per la rete permette di configurare l'indirizzo
 - ▶ **configurazione manuale**
 - ▶ IP
 - ▶ netmask
 - ▶ default gateway (router)
 - ▶ DNS server
 - ▶ **configurazione automatica**
 - ▶ uso di DHCP server
- ▶ La configurazione si può effettuare per ogni dispositivo di rete

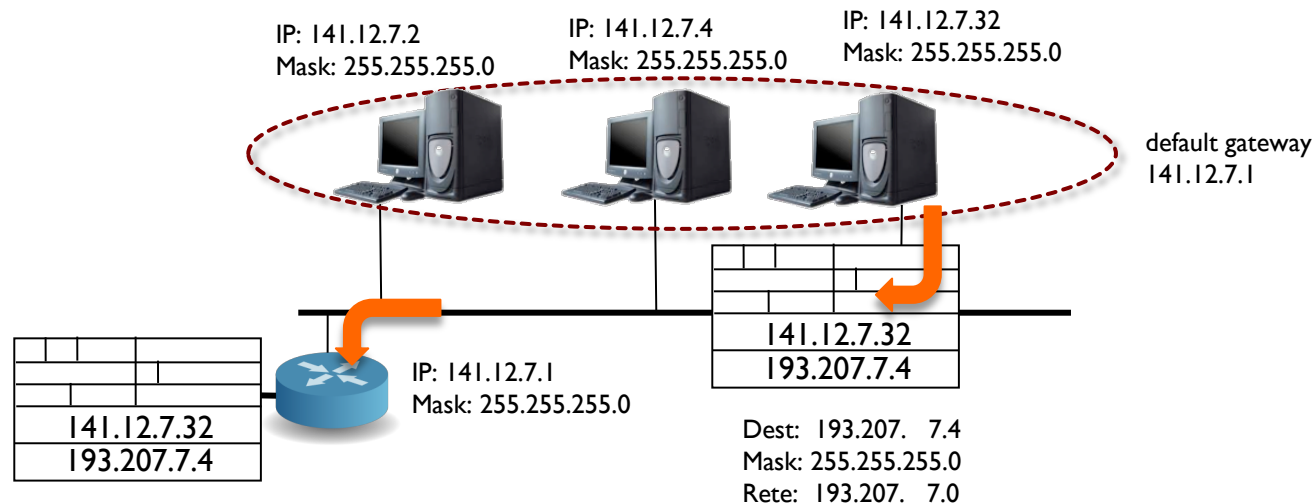


Subnet mask e instradamento

- La subnet mask permette di verificare se l'indirizzo destinatario di un pacchetto IP appartiene alla stessa sottorete del mittente

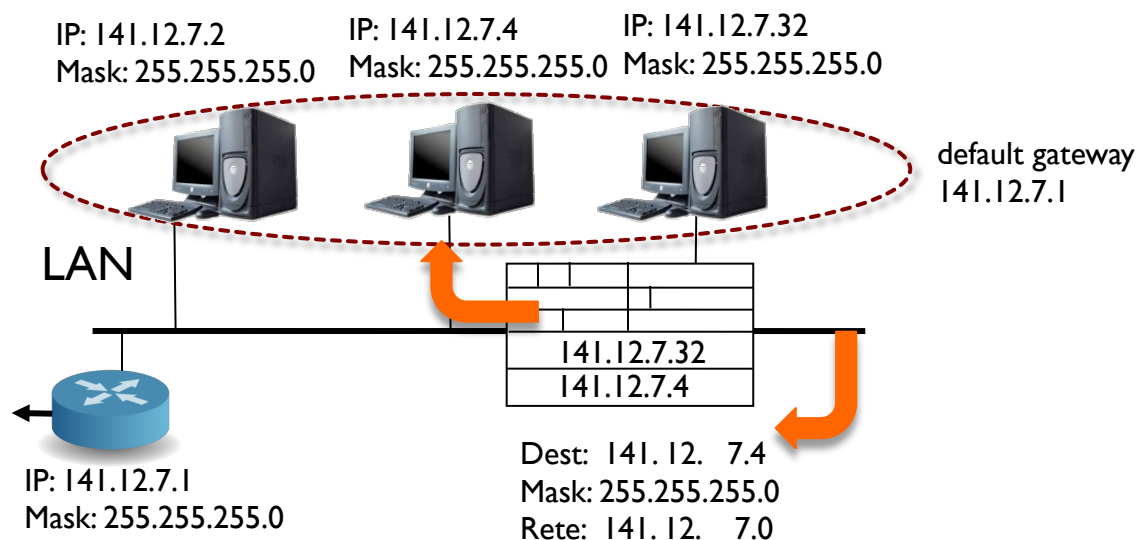
Se l'host destinatario non è nella stessa sottorete il pacchetto deve essere instradato verso il router (gateway)

```
if(!((address & subnet_mask) == mynetwork_address))  
{.. send to default gateway (router) .. }
```



Consegna diretta

- ▶ Il destinatario è un host sulla stessa sottorete LAN del mittente
- ▶ Si verifica se l'indirizzo della rete è lo stesso usando la netmask
 - ▶ Occorre associare all'indirizzo IP del destinatario il suo **indirizzo fisico**
 - ▶ Esempio: **conversione IP** → **Indirizzo ethernet** (indirizzo MAC a 48 bit)
 - ▶ Si usa il protocollo **ARP** (Address Resolution Protocol)



Protocolli ARP e RARP

- ▶ L'associazione IP-Indirizzo fisico può essere
 - ▶ **Statica**
 - ▶ Si compila una tabella con le associazioni che deve essere memorizzata su tutti i dispositivi della sottorete
 - ▶ Richiede di aggiornare periodicamente le tabelle
 - ▶ E' poco flessibile
 - ▶ **Dinamica**
 - ▶ Permette di ricavare l'associazione inoltrando una richiesta su rete



Address Resolution Protocol

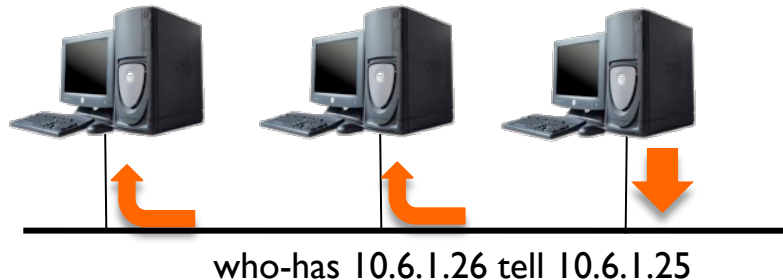


Reverse Address Resolution Protocol

Protocollo ARP

- ▶ La richiesta è inoltrata in broadcast sulla sottorete
- ▶ La macchina che ha l'indirizzo IP risponde col suo indirizzo MAC (**Media Access Control**)
- ▶ nel caso della rete LAN Ethernet è un indirizzo a 48 bit

IP: 10.6.1.26 IP: 10.6.1.5 IP: 10.6.1.25
Mask: 255.255.255.0 Mask: 255.255.255.0 Mask: 255.255.255.0



IP Source: 10.6.1.25
IP Dest: 10.6.1.26
MAC Source: 00:E0:98:2D:09:97
MAC Dest: FF:FF:FF:FF:FF:FF



IP Source: 10.6.1.26
IP Dest: 10.6.1.25
MAC Source: 00:00:39:7C:83:DB
MAC Dest: 00:E0:98:2D:09:97

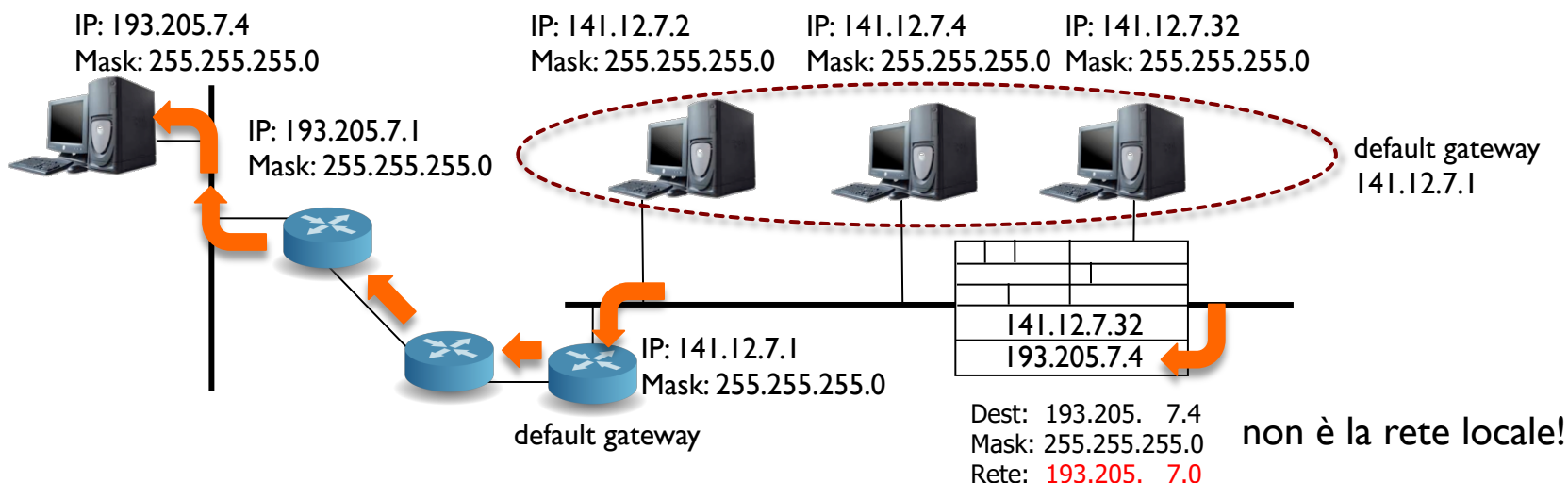
Cache ARP

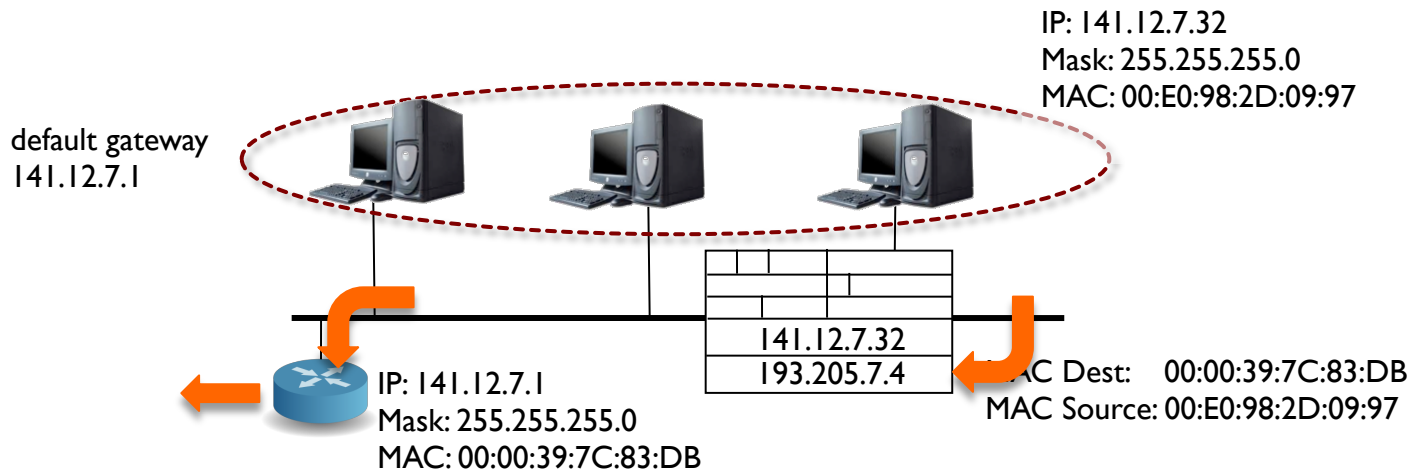
- ▶ Per evitare di risolvere l'IP in indirizzo MAC per ogni pacchetto si mantiene una **cache locale**
- ▶ Su Unix si può gestire la cache col comando **arp**
 - ▶ Consultare la cache
 - ▶ Aggiungere o cancellare elementi

```
[host]# arp -n
Address      HWtype  HWaddress      Flags Mask  Iface
193.205.7.1  ether   00:00:0C:3A:99:C6  C           eth0
193.205.7.2  ether   00:10:5A:2E:5A:CE  C           eth0
```

Consegna indiretta

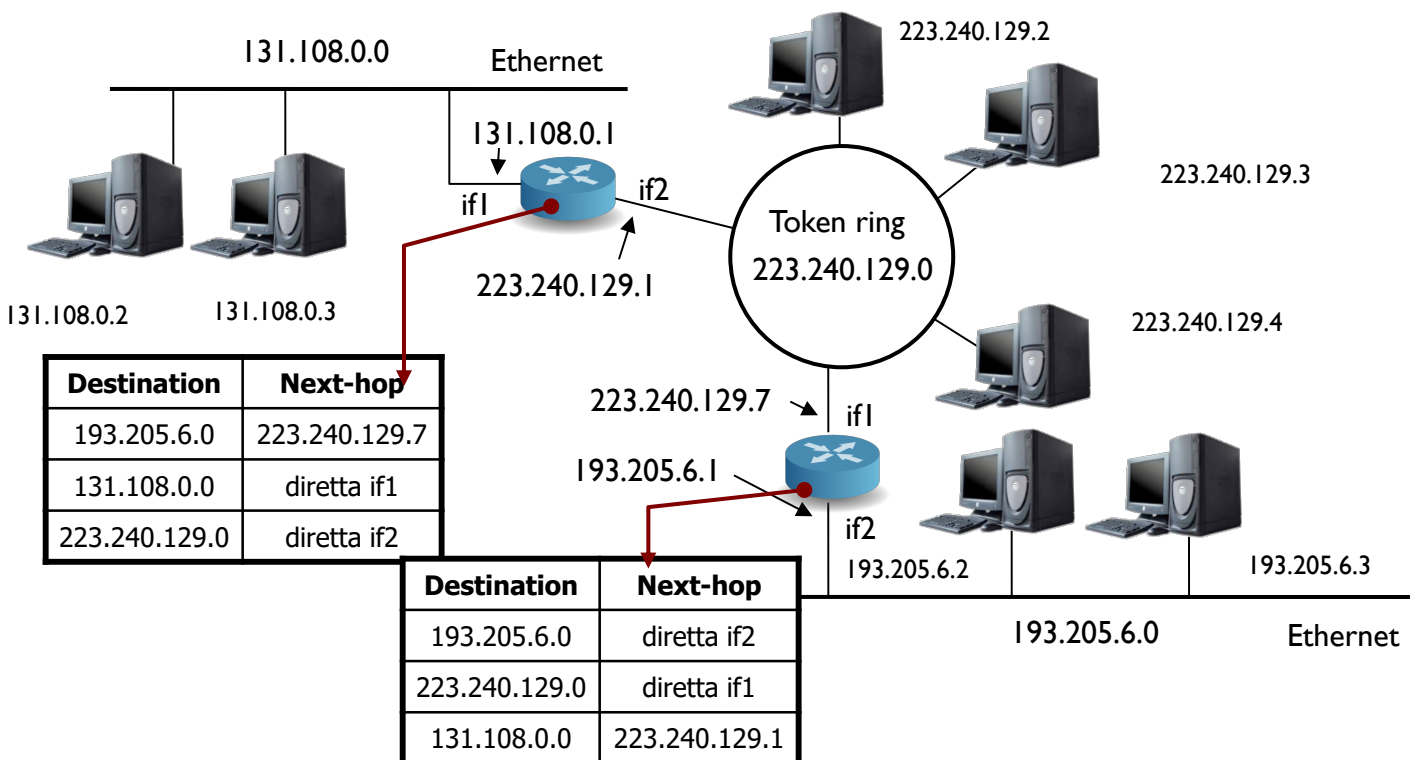
- ▶ La destinazione finale è un host connesso ad una rete fisica LAN diversa da quella in cui si trova il mittente
 - ▶ Il mittente verifica che la destinazione non è sulla sua sottorete e usa le informazioni di routing (**tabella di routing**)





Instradamento next-hop

- ▶ Il router utilizza le tabelle di routing per individuare il percorso verso la destinazione finale
 - ▶ Non si memorizza l'intero percorso ma solo il prossimo “salto” (hop)
 - ▶ Le entry della tabella sono sottoreti (l'instradamento è in genere per gruppi di IP)



Routing di default

- ▶ Il routing di default permette di instradare pacchetti verso reti che non sono gestite localmente
 - ▶ I pacchetti sono indirizzati verso un router “di confine” della rete
 - ▶ Tutti i pacchetti con IP non appartenente alle reti note sono inviati dal default router

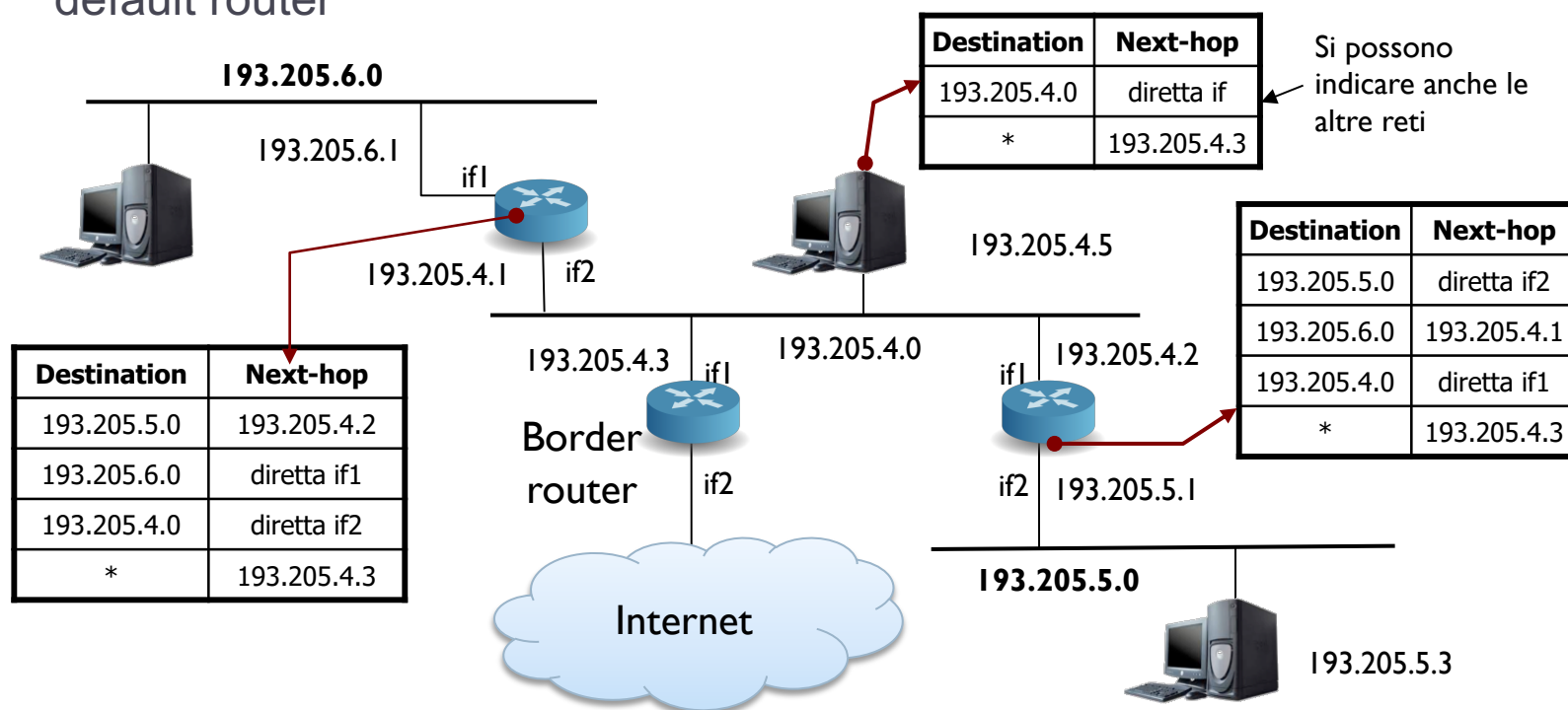


Tabella di routing

- ▶ La tabella di routing definisce il next-hop per un pacchetto da instradare
 - ▶ Le entry (linee) sono scandite in sequenza
 - ▶ Si applica la prima regola verificata (`Ipdest & Genmask`) == Destination
 - ▶ L'ultima entry è l'instradamento di default che è verificato da ogni IP

[host]# route -n									
Kernel IP routing table									
Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface		
193.205.7.146	0.0.0.0	255.255.255.255	UH	0	0	0	eth0		
193.205.7.0	0.0.0.0	255.255.255.0	U	0	0	0	eth0		
127.0.0.0	0.0.0.0	255.0.0.0	U	0	0	0	lo		
0.0.0.0	193.205.7.1	0.0.0.0	UG	0	0	0	eth0		

Connezione diretta (nessun gateway)

distanza dalla destinazione (in hops)

Default route

Next-hop

U = up (connessione attiva)
G = gateway
H = host specifico
D/M = aggiunto/modificato da ICMP

interfaccia usata

Routing

- ▶ Spedizione di un pacchetto con IP destinazione IPdest
 - ▶ Si cerca IPdest nelle tabelle di routing (diretta/host/rete/default)
 - ▶ Si ricava il prossimo “hop” ovvero l’indirizzo IP del prossimo nodo sul percorso (e l’interfaccia da usare)
 - ▶ Si spedisce il datagram al prossimo nodo
- ▶ Nel pacchetto inviato l’unico indirizzo che compare è quello dell’ultima destinazione
- ▶ Si devono definire le tabelle di routing
 - ▶ **Instradamento statico** (costruzione manuale – piccole reti)
 - ▶ **Instradamento dinamico** (aggiornamento periodico in base a informazioni scambiate fra i router con protocolli RIP, OSPF o BGP)

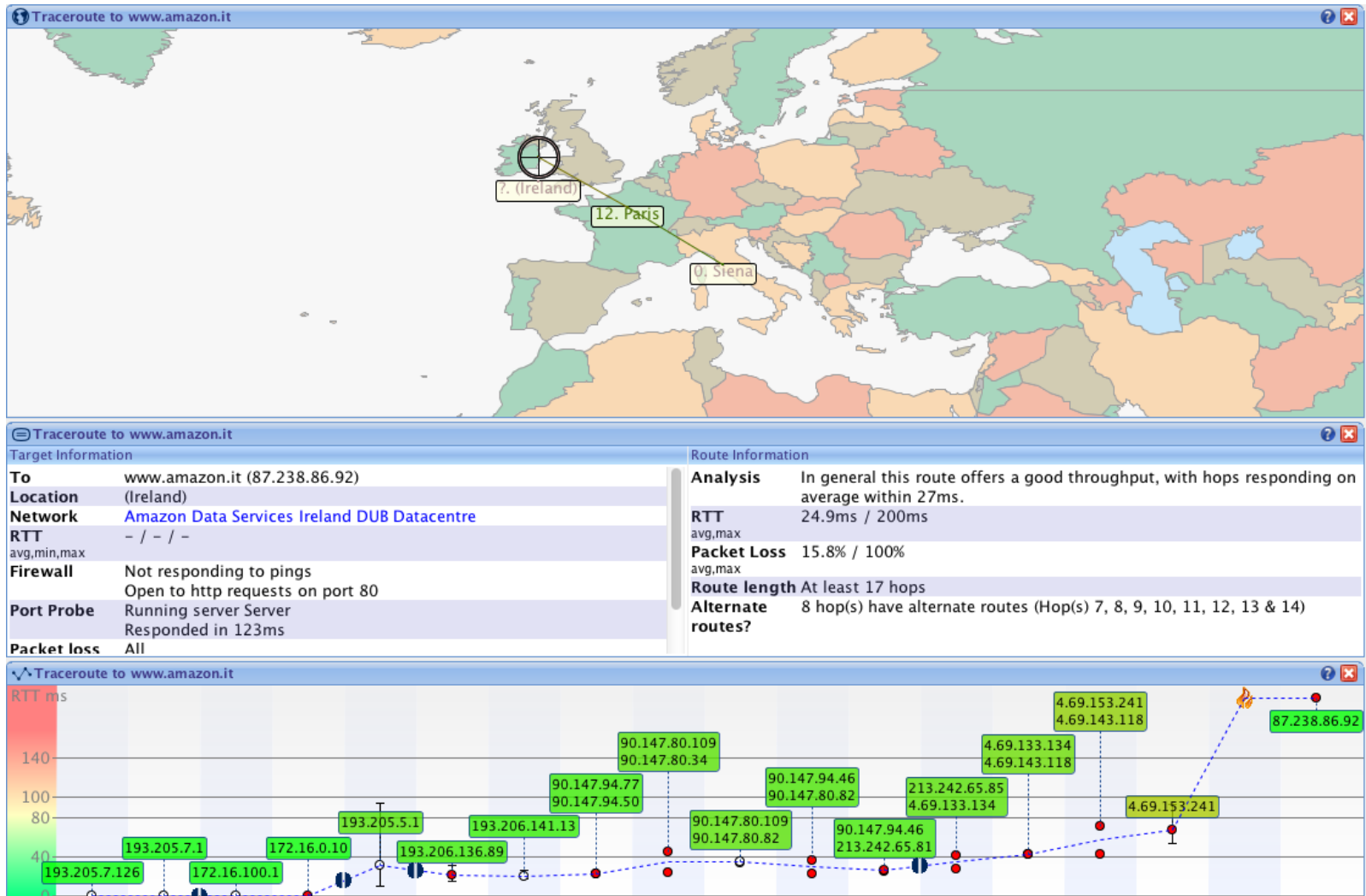
Il comando traceroute

- ▶ Traccia il percorso dei pacchetti verso una data destinazione mostrando i nodi attraversati
- ▶ Utilizza il campo **TIME TO LIVE** utilizzando il messaggio di notifica ICMP **TIME_EXCEEDED** inviato dai gateway quando il pacchetto scade
- ▶ Per default sono inviati 3 pacchetti “sonda” per ogni TTL e viene mostrato il tempo di round-trip per ciascuno di essi
- ▶ Il messaggio di notifica è atteso per un tempo massimo (5 s) altrimenti si stampa “*” per il relativo pacchetto sonda
- ▶ Il numero massimo di hop è 30 (lo stesso delle connessioni TCP)
- ▶ Non tutti i gateway inviano il messaggio **TIME_EXCEEDED**
 - ▶ Compare “*” in tal caso

Un esempio di route

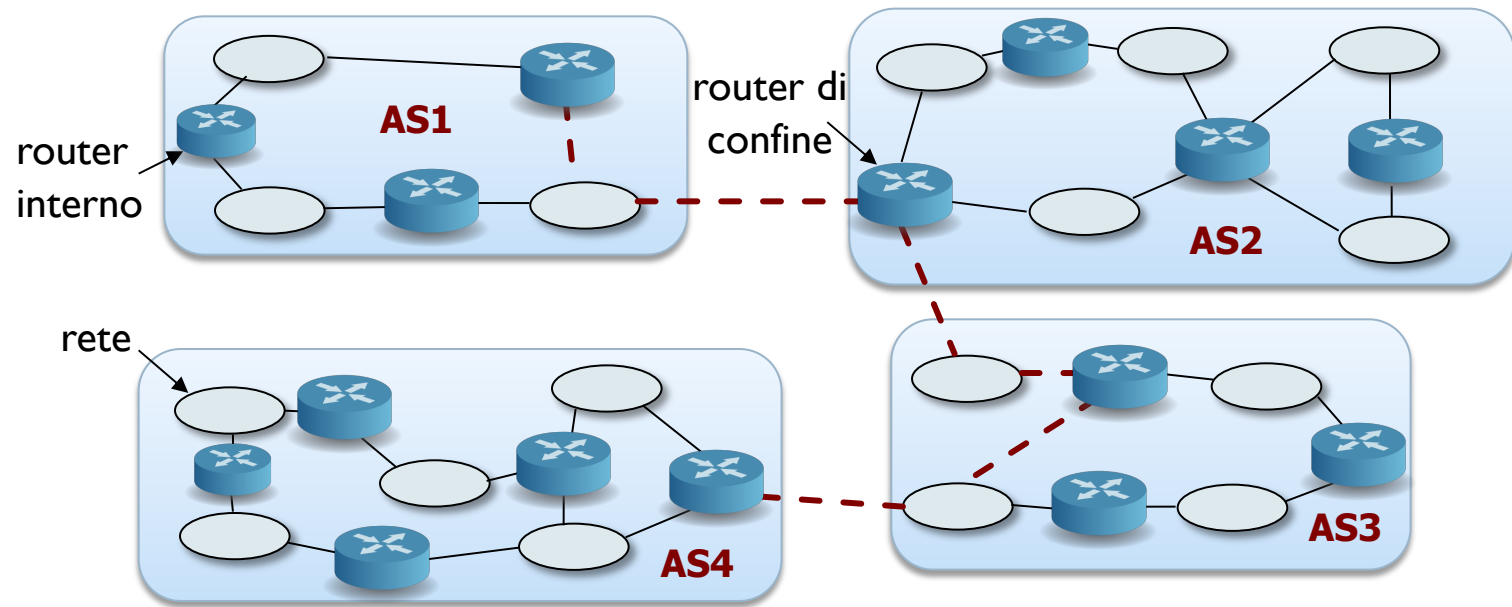
```
[host]# traceroute www.virgilio.it
traceroute to www.virgilio.it (212.48.10.150), 30 hops max, 38 byte packets
 1  ringe (193.205.7.1)  2.685 ms  1.665 ms  1.662 ms
 2  cuces-ing.link.unisi.it (193.205.5.101)  70.407 ms  32.951 ms  70.588 ms
 3  rcuces.unisi.it (193.205.4.1)  43.928 ms  130.064 ms  125.421 ms
 4  rc-unisi.fi.garr.net (193.206.136.89)  125.552 ms  119.365 ms  103.817 ms
 5  bo-fi.garr.net (193.206.134.93)  70.092 ms  38.814 ms  20.001 ms
 6  rtg-rt-1.bo.garr.net (193.206.134.198)  19.653 ms  105.629 ms  14.991 ms
 7  rm-bo-g.garr.net (193.206.134.49)  32.389 ms  60.269 ms  25.866 ms
 8  cw-nap.inroma.roma.it (194.242.224.3)  45.573 ms  54.386 ms  88.611 ms
 9  s3-0-r2-ROM2.cwitaly.net (195.94.140.173)  37.144 ms  27.868 ms  42.471 ms
10  194.79.199.35 (194.79.199.35)  26.232 ms  25.961 ms  33.669 ms
11  pos0-1-0-r5-MIL3.cwitaly.net (194.79.192.5)  49.560 ms  33.136 ms  81.481 ms
12  fe0-0-r3-MIL1.cwitaly.net (213.233.14.75)  164.763 ms  71.242 ms  107.814 ms
13  * matrix-mi.ar1.mil.cwitaly.net (195.250.255.102)  45.349 ms  79.267 ms
14  212.48.4.241 (212.48.4.241)  71.192 ms  59.393 ms  43.822 ms
15  212.48.4.174 (212.48.4.174)  59.138 ms  83.495 ms  39.530 ms
16  * * *
17  * * *
```

Visual route



Sistemi autonomi

- ▶ La dimensione di Internet rende impossibile gestire centralmente tutti router
 - ▶ La rete è divisa in **sistemi autonomi** (**Autonomous Systems**)
 - ▶ Il routing è **esterno** o **interno** se esce o rimane confinato in un AS



Costruzione delle tabelle di routing

- ▶ La costruzione delle tabelle di routing deve garantire
 - ▶ **Robustezza** rispetto ai guasti
 - ▶ **Imparzialità** rispetto ai mittenti
 - ▶ **Ottimalità** rispetto ad un data **metrica** (minimo ritardo, massimo throughput della rete, minimo numero di hop)
 - ▶ Ad ogni passo di instradamento è associata una metrica
 - ▶ La metrica del percorso è la somma delle metriche associate alle reti attraversate
- ▶ Gli algoritmi per il calcolo delle tabelle di routing possono essere
 - ▶ **Non adattivi** - calcolano i percorsi off-line (routing statico)
 - ▶ **Adattivi** - modificano le tabelle di routing in base a variazioni della topologia e del traffico

Limiti dell'indirizzamento IPv4

- ▶ Gli indirizzi IP (o meglio le sottoreti) si stanno esaurendo
 - ▶ Esistono circa 2.000.000.000 di indirizzi “usabili” per host
 - ▶ L'organizzazione in classi porta ad uno spreco
 - ▶ Acquisire una rete di classe B implica acquisire 65536 indirizzi
 - ▶ Una rete di classe C è troppo piccola per molte organizzazioni
 - ▶ Le reti di classe A sono troppo grandi
- ▶ Le tabelle di routing sono molto grosse
 - ▶ i router devono conoscere tutte le reti
 - ▶ il problema sono le reti di classe C (sono 2.097.152)
 - ▶ **CIDR** (Classless Interdomain Routing – RFC 1519) prevede di allocare le reti di classe C in blocchi di reti contigue
 - ▶ Sono definite regole di allocazione “geografica” per intervalli di indirizzi
 - ▶ es. 194.0.0.0 – 195.255.255.255 sono per l'Europa

Il protocollo ICMP

- ▶ I messaggi **ICMP** (**I**nternet **C**ontrol **M**essage **P**rotocol – RFC 792) servono per la gestione di funzionalità del livello di rete
 - ▶ **Segnalazione di errore** (inviati al mittente)
 - ▶ **Destination Unreachable** – Il pacchetto non può essere consegnato
 - ▶ **Time exceeded** – TTL ha raggiunto il valore 0 oppure non si sono ricevute tutte le parti di un pacchetto frammentato entro un tempo limite
 - ▶ **Parameter Problem** – Errore nei parametri dell'header
 - ▶
 - ▶ **Messaggi di richiesta**
 - ▶ **Echo request/Echo reply** – Richiesta/risposta di eco (verifica se un host risponde)
 - ▶ **Timestamp request/Timestamp reply** – Richiesta/risposta con timestamp
 - ▶

Destination unreachable

► Messaggio di errore

- Inviato (eventualmente) da
 - Router che non riesce ad instradare un pacchetto
 - Host che non riesce a consegnarlo
- Un codice indica il motivo (host/rete non raggiungibile, porta TCP non aperta,...)
- Al messaggio è allegato l'inizio del pacchetto per capire a chi notificare l'errore (porta del mittente)

Tipo: 3	Codice: 0-15	checksum
0 (non usato)		
Header e primi 8 byte del datagram IP ricevuto		

Il comando ping

- ▶ Invia pacchetti ICMP ECHO REQUEST ad un host o gateway
- ▶ Attende le risposte ICMP ECHO REPLY
- ▶ Il pacchetto contiene un timestamp per misurare il tempo di round-trip

```
[host]# ping -v cuces.unisi.it
PING cuces.unisi.it (193.205.4.2): 56 data bytes
64 bytes from 193.205.4.2: icmp_seq=0 ttl=252 time=28.6 ms
64 bytes from 193.205.4.2: icmp_seq=1 ttl=252 time=45.1 ms
64 bytes from 193.205.4.2: icmp_seq=2 ttl=252 time=61.8 ms
64 bytes from 193.205.4.2: icmp_seq=3 ttl=252 time=62.8 ms
64 bytes from 193.205.4.2: icmp_seq=4 ttl=252 time=78.0 ms

--- cuces.unisi.it ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max = 28.6/55.2/78.0 ms
```