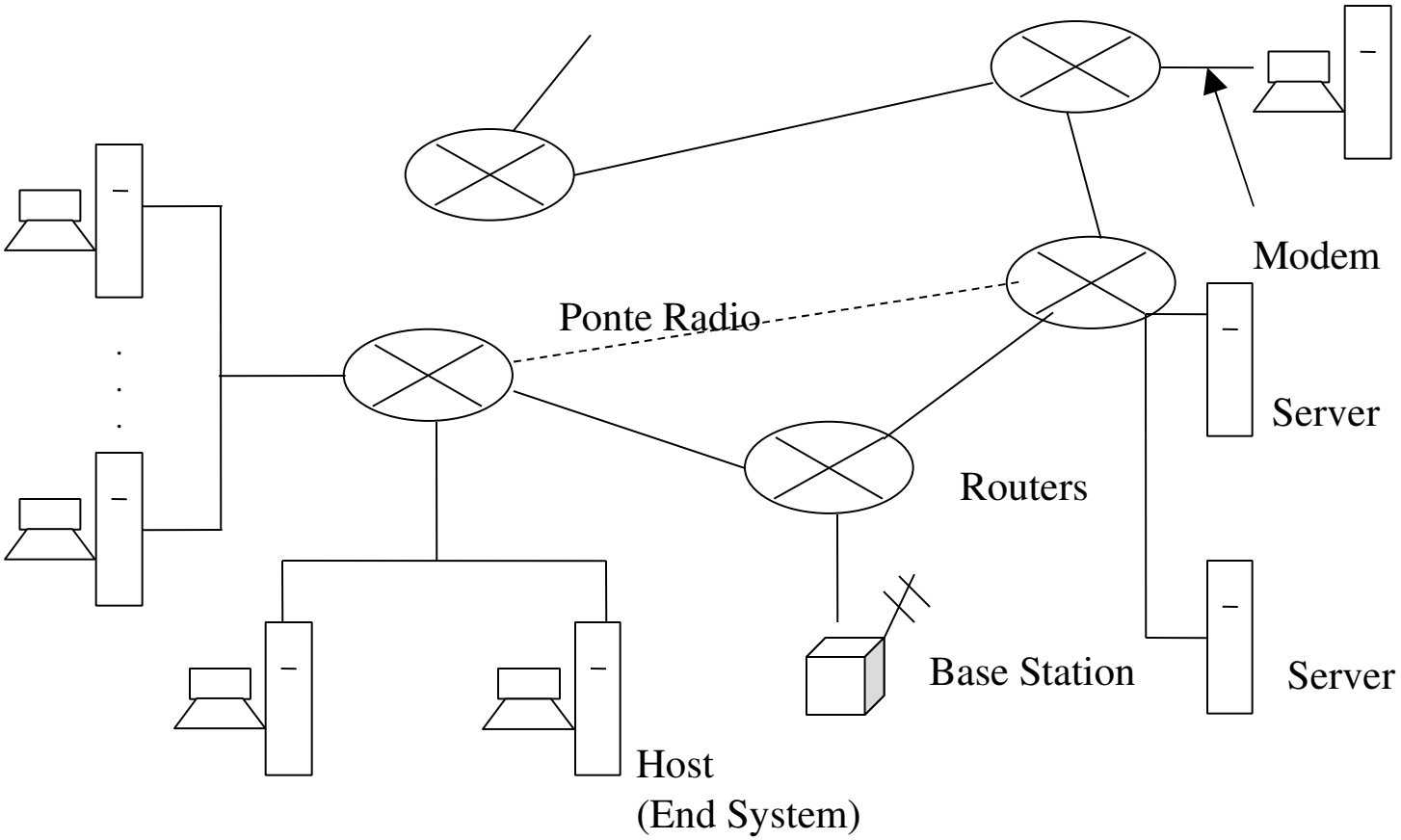
CORSO DI RETI SSIS

Laura Ricci

27 Ottobre 2005

INTENET: COMPONENTI FONDAMENTALI



INTERNET: ARCHITETTURA HARDWARE

Componenti hardware fondamentali di una rete:

- *hosts (end systems):* PCs, workstations + servers (ma anche mobile computers,..)
  Servers: hosts dedicati a fornire servizi di rete. Ad esempio server http memorizza e fornisce accesso ad un insieme di pagine we.

- *routers:*dispositivi che *instradano (routing)* i messaggi scambiati tra gli hosts. Possiedono più links di collegamento alla rete. Instradano l'informazione in arrivo da un link di ingresso su uno dei links di uscita, scelto secondo un *algoritmo di routing*

- *communication link:* supporto fisico che permette la comunicazione host/host, host/router, router/router (esempio: doppino di rame, cavo coassiale, fibre ottiche, ponti radio…)

INTERNET: APPLICAZIONI DISTRIBUITE

applicazioni distribuite: un programma *distribuito* comprende diverse componenti. Ogni componente viene eseguita su un host diverso. Le componenti *cooperano* scambiandosi *dati*

paradigma Client/Server: una componente, il *client*, in esecuzione su un host (in genere un PC) richiede un *servizio* ad un'altra componente, il *server* (in genere in esecuzione su una macchina con maggiore potenza di calcolo)

richiesta del servizio, ricezione dei risultati richiedono uno *scambio di messaggi* tra client e server

Esempi di applicazioni client/server

- web
- e-mail
- ftp (file transfer protocol)
- ssh (login remoto)
- multiplayer games

INTERNET: SUPPORTO PER LE APPLICAZIONI

La rete definisce un insieme di *supporti hardware/software (servizi*) per la comunicazione tra gli hosts

Connection Oriented Service: garantisce che:

- ogni messaggio inviato da un mittente M ad un destinatario D venga recapitato a D
- i messaggi vengono recapitati a D *nell'ordine* con cui sono stati spediti, cioè se m_1 viene spedito prima di m_2, D riceve m_1 prima di m_2.

Connectionless Service: non garantisce che un messaggio spedito verrà recapitato. Non garantisce l'ordinamento dei messaggi.

Entrambe i servizi precedenti sono *best effort:* non forniscono garanzie circa il *tempo richiesto* per recapitare un messaggio al destinatario.

INTERNET: CONNECTION ORIENTED SERVICE

Protocollo di rete: insieme di regole che definiscono il formato, l'ordine dei messagi scambiati tra due o più entità comunicanti cele operazioni eseguite in seguito all'invio/ricezione dei messaggi.

L'invio/ricezione di un messaggio prevede l'esecuzione di diversi protocolli a diversi livelli di astrazione.

Esempio: richiesta di una pagina ad un web server, protocollo *a livello applicazione:*

client: GET /public_html/file1 HTTP/1.3

host: www.unipi.it ….

server ….

Content-Length: 6821

Content-type: text/HTML …

INTERNET: CONNECTION ORIENTED SERVICE

Connection-Oriented Service:

- client e server si scambiano alcuni messaggi di controllo prima di scambiarsi i dati veri e propri (*get* message + *contenuto* della pagina web).

  client: chiedo una connessione

  server: connessione accettata

- lo scambio di questi messaggi consente di allocare strutture dati nel mittente/destinatario necessarie per *controllare* la trasmissione

- I messaggi di controllo scambiati sono definiti da un protocollo a *livello trasporto,* eseguito in modo trasparente rispetto alla applicazione

INTERNET: CONNECTION ORIENTED SERVICE

Connection Oriented Protocol:

Strutture dati necessarie per gestire la connessione= buffers+ variabili di stato

Protocollo *orientato* alla connessione = i partner della comunicazione stabiliscono una *connessione non stretta*

Connessione non stretta

- solo i due hosts che hanno stabilito la connessione *sono consapevoli* della connessione, cioè mantengono le strutture dati per gestirla

- i routers non registrano le connessioni stabilite tra gli hosts.

INTERNET: CONNECTION ORIENTED SERVICE

Connection Oriented Service, servizi forniti

- *Trasferimento di dati Affidabile:* Basato sull'utilizzo di messaggi di *ack* + identificatori di sequenza associati ai messaggi
- *Controllo del flusso:* consente al mittente di regolare la velocità di trasmissione dei messaggi in modo da non riempire il buffer del destinatario
- *Controllo della congestione:* consente di diminuire la frequenza di invio dei messaggi in modo da evitare la congestione dei routers intermedi

Servizio fornito ornito dal protocollo TCP (Trasmission Control Protocol)

Utilizzato da protocollo HTTP, FTP, remote login, e-mail

INTERNET: CONNECTIONLESS SERVICE

Connectionless Service: nessuna garanzia sulla affidabilità della trasmissione

$\Rightarrow$

i pacchetti possono:

- essere persi durante la trasmissione
- arrivare fuori ordine
- essere sovrascritti ad altri pacchetti presenti nel buffer del destinatario

(non esiste un meccanismo di controllo del flusso)

servizio minimale fornito dal protocollo UDP (User Datagram Protocol) a livello trasporto

Utilizzato per la spedizione di dati multimediali (video-conferenza, audio on-demand,…)

TECNICHE DI INSTRADAMENTO DEI MESSAGGI

- Come avviene la comunicazione tra gli end systems mediante i routers intermedi?

- Tecniche realizzate:

  - Commutazione di circuito
  - Commutazione di Pacchetto
    - Datagram Networks (Datagram analogo a Telegram)
    - Virtual Circuits

- Internet: Rete a commutazione di pacchetto, utilizza datagrams
- ATM: Rete a commutazione di pacchetto, utilizza virtual circuits

COMMUTAZIONE DI CIRCUITO

Commutazione di circuito: se HA vuole comunicare con HB

- si *stabilisce un cammino* P tra HA ed HB
- si *riserva un insieme di risorse* lungo P (buffers, banda dei links) prima di iniziare la comunicazione C
- le risorse rimangono riservate a C per tutta la sua durata

$\Rightarrow$

Poichê le risorse sono dedicate a quella trasmissione si può garantire una velocità di trasmissione stabilita

Nota: i routers intermedi tengono traccia della connessione tra gli end system
si realizza una connessione stretta tra HA ed HB ≠ da connection oriented service offerto da TCP.

COMMUTAZIONE DI CIRCUITO

Tecniche per l'implementazione di un insieme di circuiti sullo stesso link

- *FDM* Frequency Division Multiplexing
  circuiti diversi utilizzano frequenze diverse

- *TDM* Time Division Multiplexing
  Tempo diviso in frames
  Ogni frame temporale diviso in slots
  circuiti diversi utilizzano slot temporali diversi all'interno dello stesso frame

Vantaggi vs. svantaggi: garanzia del servizio vs. utilizzazione limitata delle risorse della rete

COMMUTAZIONE DI PACCHETTO

I messaggi scambiati tra gli hosts vengono divisi in pacchetti.

I pacchetti vengono spediti sui links di comunicazione ed instradati dai *routers (packet switches)*

Le risorse utilizzate per inoltrare un pacchetto non vengono riservate, ma vengono allocate *on demand*

$\Rightarrow$

se le risorse richieste non sono disponibili un pacchetto può essere *bloccato per un tempo non prevedibile a priori*

$\Rightarrow$

non si può garantire un livello minimo di servizio

COMMUTAZIONE DI PACCHETTO

- instradamento di un pacchetto all'interno di un router:
  - il router deve riceve *un intero* pacchetto da un *link di ingresso* prima di spedirlo su un *link di uscita (store and forward)*
  - esiste un insieme di buffers associati ai links di ingresso/uscita
  - se il link su cui deve essere instradato il pacchetto risulta occupato, il pacchetto deve essere memorizzato nella coda associata al link di uscita (*queueing delay*)
  - Perdita di pacchetti = può avvenire quando il buffer associato al link su cui il pacchetto deve essere instradato è pieno. Viene scartato il pacchetto in arrivo o uno dei pacchetti memorizzati nel buffer.

COMMUTAZIONE DI MESSAGGIO

commutazione di messaggio = caso particolare della tecnica della commutazione di pacchetto

- Il massaggio spedito da un host non viene diviso in segmenti
- ogni pacchetto contiene un intero messaggio

Packet switching $\Rightarrow$ trasmissione pipeline

Message switching $\Rightarrow$ trasmissione sequenziale

Packet switching richiede una latenza minore nella spedizione dei messaggi

COMMUTAZIONE DI PACCHETTO: ROUTING

Routing = instradamento dei messaggi nella rete

Approcci per il routing nelle reti a commutazione di pacchetto:

- *datagram networks:* il routing viene effettuato in base *all'indirizzo del destinatario* contenuto nel pacchetto

- *virtual circuit networks:* viene costruito un *circuito virtuale* tra mittente e destinatario. I messaggi scambiati vengono instradati su questo circuito

Internet: *Packet Switched Datagram Network*

ATM: *Packed Switched Virtual Circuit Network*

COMMUTAZIONE DI PACCHETTO: DATAGRAM NETWORKS

- Datagram Networks forniscono un servizio analogo al servizio postale (datagram= telegram)

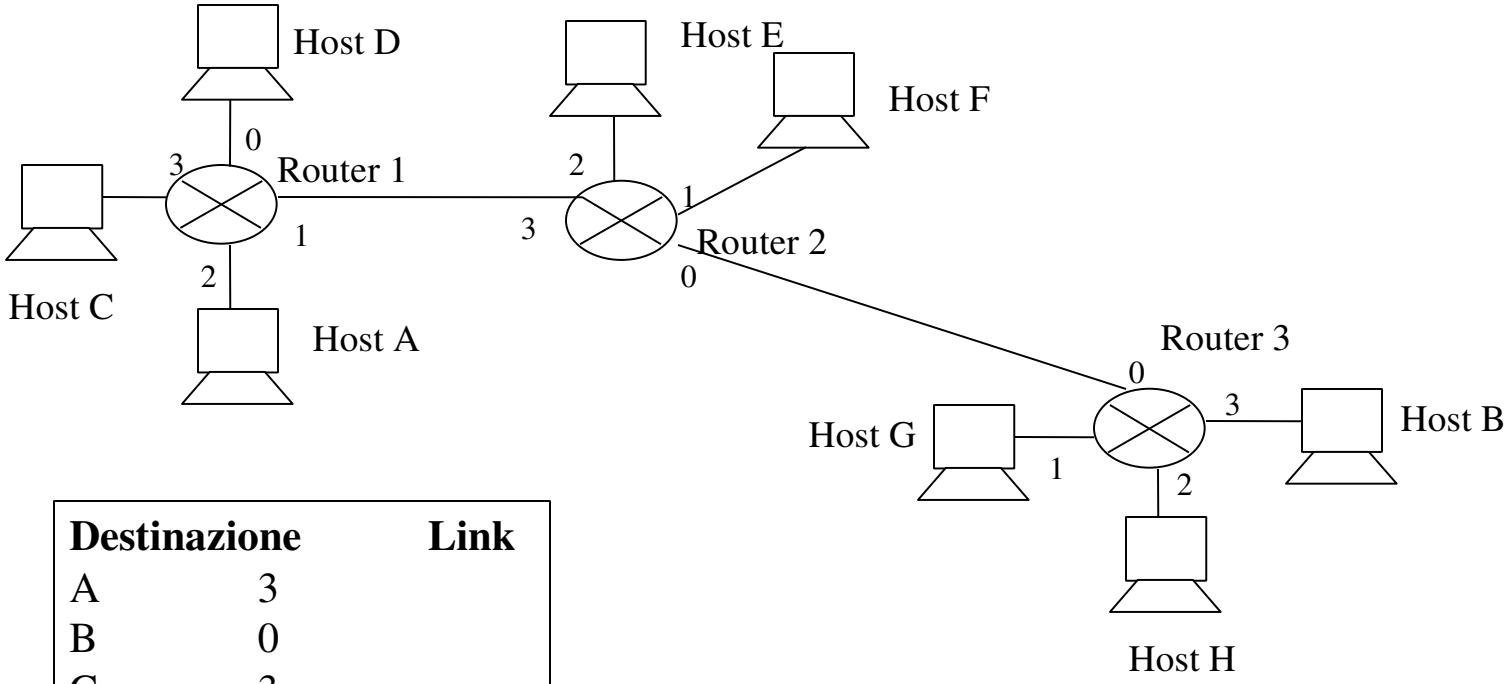
- Ogni pacchetto contiene un *header* in cui è contenuto l'indirizzo del destinatario

- L'indirizzo possiede una struttura gerarchica (es: indirizzo sottorete + indirizzo host)

- Ogni router sceglie il link di uscita su cui instradare un pacchetto in arrivo in base all'indirizzo contenuto nell'header del pacchetto (tabelle di routing)

- I routers non mantengono informazioni circa le connessioni effettuate tra hosts

COMMUTAZIONE DI PACCHETTO
DATAGRAM NETWORKS



Destinazione	Link
A	3
B	0
C	3
D	3
E	2
F	1
G	0
H	0

Router 2:
Tabella di Routing

COMMUTAZIONE DI PACCHETTO
DATAGRAM NETWORKS

Caratteristiche principali reti a commutazione di pacchetto

- non si stabilisce alcuna connessione (a livello di routers) tra mittente e destinatario (nota bene: questo non vieta che si stabilisca una trasmissione connection oriented tra gli host a livello trasporto!!)

- quando un host invia un pacchetto non ha alcun modo di sapere ne se sarà recapitato ne se il destinatario è operativo

- ogni pacchetto viene inoltrato in modo indipendente dai pacchetti precedenti

  $\Rightarrow$

  due pacchetti scambiati tra gli stessi hosts possono *segure percorsi diversi* (esempio: le tabelle di routing vengono modificate dinamicamente, sono previsti più percorsi per la stessa destinazione)

- tecnica tollerante ai guasti (ARPANET)

COMMUTAZIONE DI PACCHETTO
VIRTUAL CIRCUIT NETWORKS

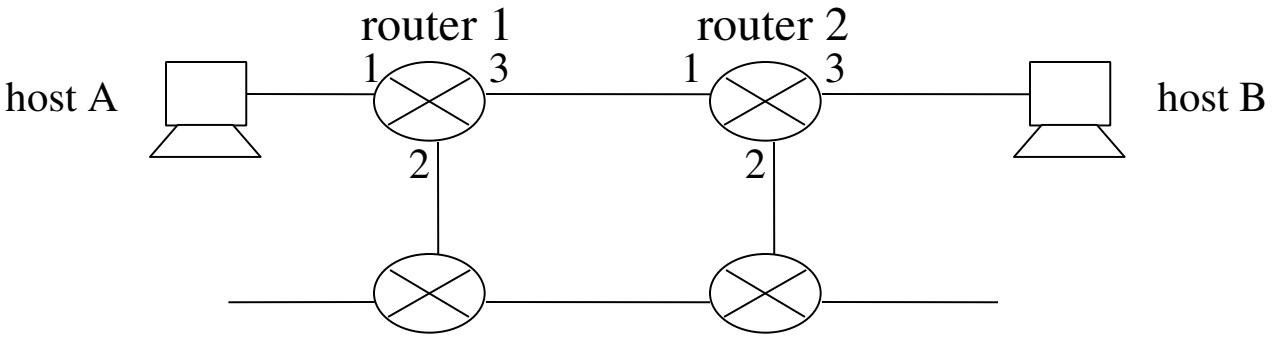
Prima dell'invio dei dati viene creato un *circuito virtuale* tra mittente e destinatario

Circuito virtuale definito da

- un insieme di links e routers che definiscono il cammino P tra mittente e destinatario
- per ogni link di P un numero intero che lo identifica
- per ogni switch, una tabella che consente di instradare correttamente i messaggi sul circuito virtuale

Ogni router deve memorizzare un insieme di informazioni per supportare la connessione tra una coppia di hosts

COMMUTAZIONE DI PACCHETTO
VIRTUAL CIRCUIT NETWORKS



comunicazione tra Host A ed Host B:
cammino scelto host A – router1 – router2 – host B
circuito virtuale 12,22,32

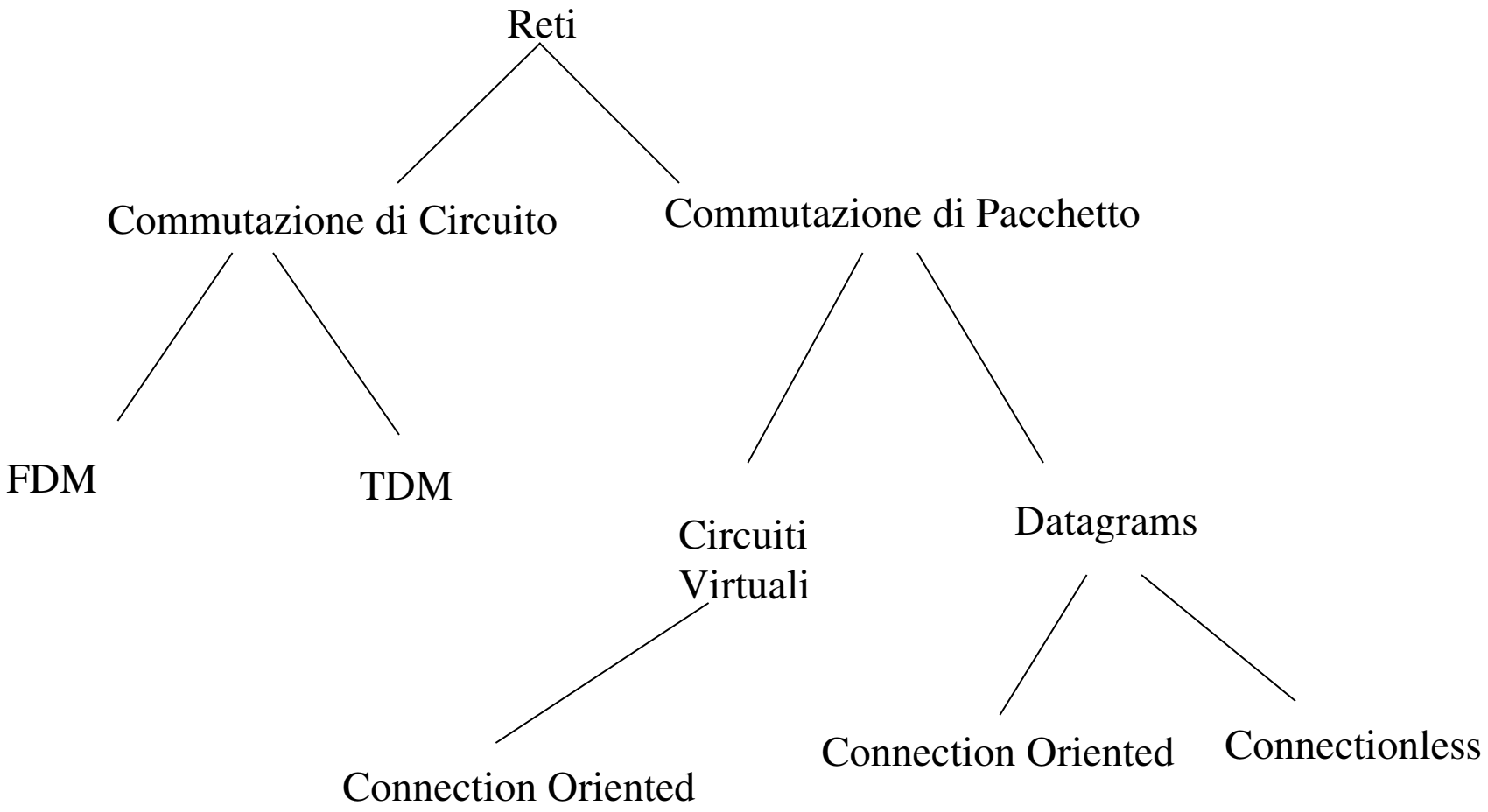
Interfaccia input	InputVCNum	Interfaccia output	Output VCNum
1	12	3	22
2	63	2	18
3	7	2	17
1	97	3	87

COMMUTAZIONE DI PACCHETTO
VIRTUAL CIRCUIT NETWORKS

Caratteristiche della tecnica dei circuiti virtuali:

- Un numero di VC diverso per ogni link del circuito virtuale
  - semplifica il protocollo necessario per stabilire il circuito virtuale: ogni link può scegliere il numero di circuito in modo autonomo
  - Consente di riutilizzare numeri già utilizzati quando una connessione viene chiusa
- *Svantaggi*
  - *overhead iniziale* per stabilire il circuito virtuale
  - in caso di guasti, occorre ricostruire il circuito virtuale
- *Vantaggi*
  - risparmio di spazio nell'header del pacchetto (numero circuito virtuale vs. indirizzo dell'host)
  - al momento della costruzione del circuito virtuale ottengo informazioni utili sullo stato della rete (esempio: operatività del destinatario)

SCHEMA RIASSUNTIVO



ARCHITETTURA DI RETE

Strutturazione a livelli della architettura di rete: consente di modularizzare la progettazione di un sistema complesso (es: una rete)

Idee generale:

- partire dai servizi base offerti dai dispositivi hardware L_0 (esempio invio di un bit su un link fisico)
- definire una sequenza di livelli L_1, …L_i,…L_n che forniscono servizi sempre *più astratti*
- il livello L_i *utilizza* i servizi definiti al livello L_{i-1}
- il livello L_i *offre* servizi al livello L_{i+1}
- I servizi offerti a livello L_i sono implementati mediante un insieme di componenti C_1, …C_i,…C_n che cooperano medante un *protocollo* P_i

ARCHITETTURA DI RETE

Per ogni livello L_i, si definisce un protocollo P_i che stabilisce le regole con cui cooperano le componenti a livello L_i

Il protocollo P_i viene eseguito in *modo distribuito* dalle componenti

Le componenti cooperano scambiandosi un insieme di messaggi (*Protocol Data Units*)

i-PDU = Protocol Data Units scambiati a livello L_i

Quando un host A invia un i-PDU all'host B, l'i-PDU viene passato dal Livello L_i al livello L_{i-1}

$\Rightarrow$

il livello L_i sfrutta i servizi del livello L_{i-1} per l'invio dell'i-PDU

ARCHITETTURA DI RETE

quando un host A invia un i-PDU all'host B, l'i-PDU viene passato dal Livello L_i al livello L_{i-1}

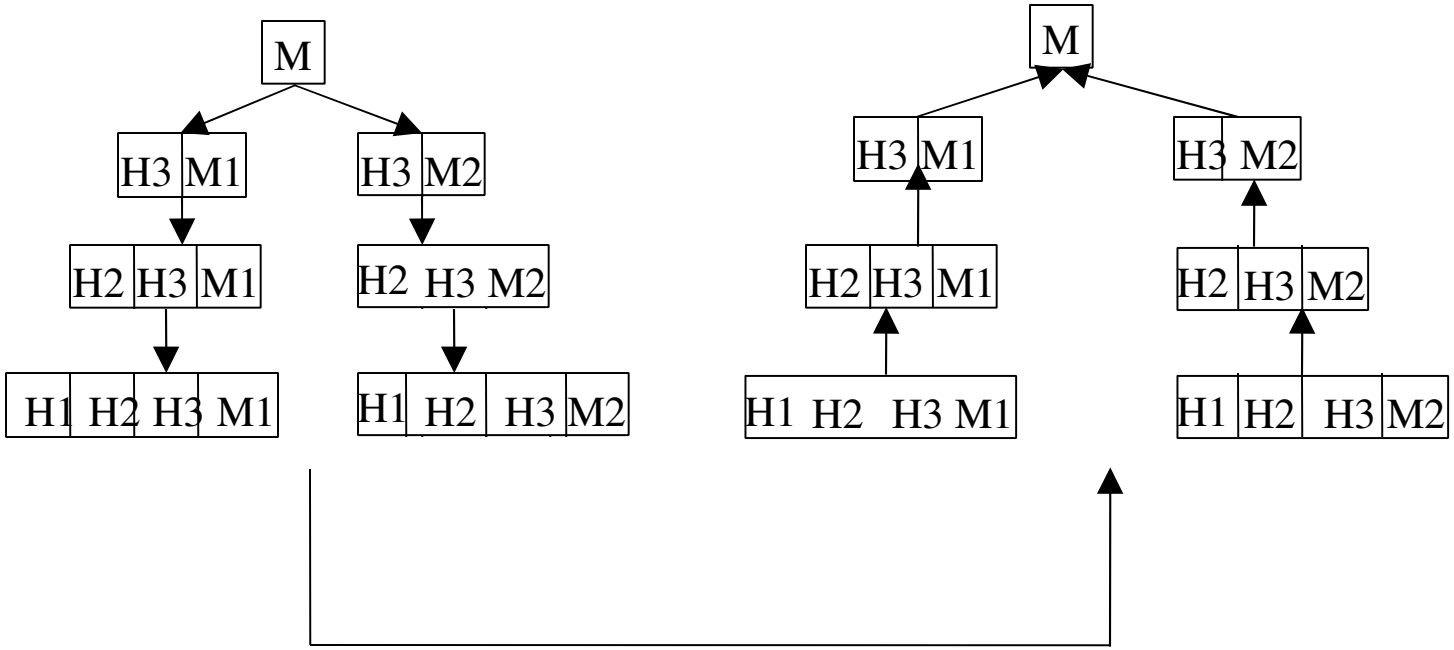
ogni livello L_i aggiunge alcune informazioni all'i+1-PDU ricevuto dal livello superiore

informazioni aggiunte dal livello L_i = *Header* H_i

ogni header contiene le informazioni necessarie per *implementare il servizio* offerto dal livello L_i

Esempio: il livello L_{i-1} garantisce ad L_i che un i-PDU venga recapitato in maniera corretta al destinatario. L'header H_{i-1} contiene informazioni che consentono di implementare la spedizione sicura.

ARCHITETTURA DI RETE



Mittente

Destinatario

ARCHITETTURA DI RETE

Funzioni che possono essere svolte da un livello Li:

- *Definizione di connessioni:* scambio di alcuni messaggi prima della comunicazione vera e propria

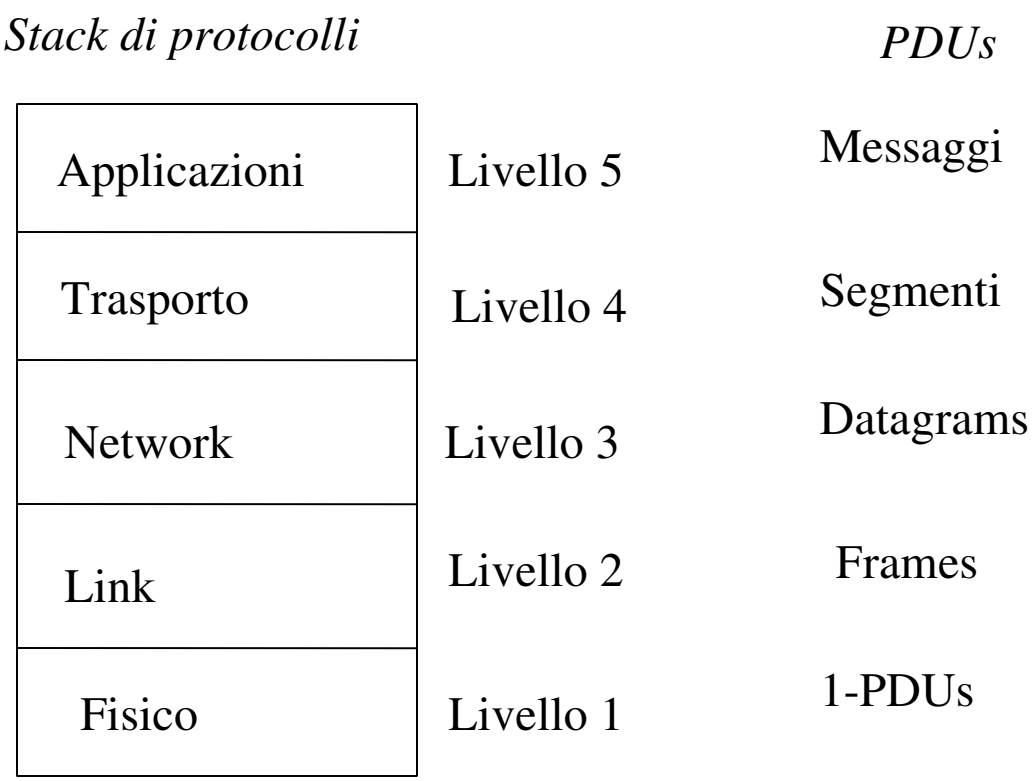
- *Definizion*e di connessioni *sicure*

- *Controllo del flusso:* regola il flusso di PDU tra mittente e destinatario in modo da evitare l'overflow dei buffers

- *Segmentazione e riassemblaggio dei messaggi:* partizionamento di un messaggio (mittente) e successiva ricomposizione del messaggio (destinatario)

- *Multiplexing*

ARCHITETTURA DI RETE: INTERNET



LIVELLO DELLE APPLICAZIONI

Livello delle applicazioni: è possibile:

- utilizzare una applicazione nota e quindi il protocollo associato
  - HTTP (web)
  - FTP (file transfer protocol)
  - SSH (secure shell)
  - SMTP (posta elettronica)
- creare nuove applicazioni
  - Definizione di componenti (C, JAVA,..)
  - Comunicazione tra componenti mediante sockets

LIVELLO TRASPORTO

Livello di trasporto: consente di definire un canale logico tra due componenti del livello applicazioni

Servizi forniti:

TCP (Trasmission Control Protocol): Definisce un canale logico su cui può essere inviato un flusso di bytes. Il canale è affidabile (garantita la consegna ordinata dei messaggi spediti)

UDP: (*User Datagram Protocol*). Definisce canali logici non affidabili

Protocolli del livello trasporto : end-to-end protocols.

LIVELLO NETWORK

- Protocollo IP = *Internet Protocol*

- *Servizio fornito:* routing dei messaggi dal mittente al destinatario

- Internet definisce più algoritmi di routing
  - link state routing
  - distance vector routing

- Livello trasporto solo negli end systems

- Routers eseguono solo protocollo IP

LIVELLO LINK

Definisce servizi per inviare un pacchetto tra due nodi della rete (routers o end systems)

Il servizio fornito dipende dalle caratteristiche fisiche del link specifico

Esempi: Ethernet per reti locali, PPP (Point to Point Protocol) per collegamento telefonico PC Provider

Servizi offerti: invio di un *frame* da un nodo all'altro, affidabilità della trasmissione (diverso da affidabilità di TCP)

Lo stesso datagram può essere trasportato mediante diversi link layer protocols quando attraversa link diversi

LIVELLO FISICO

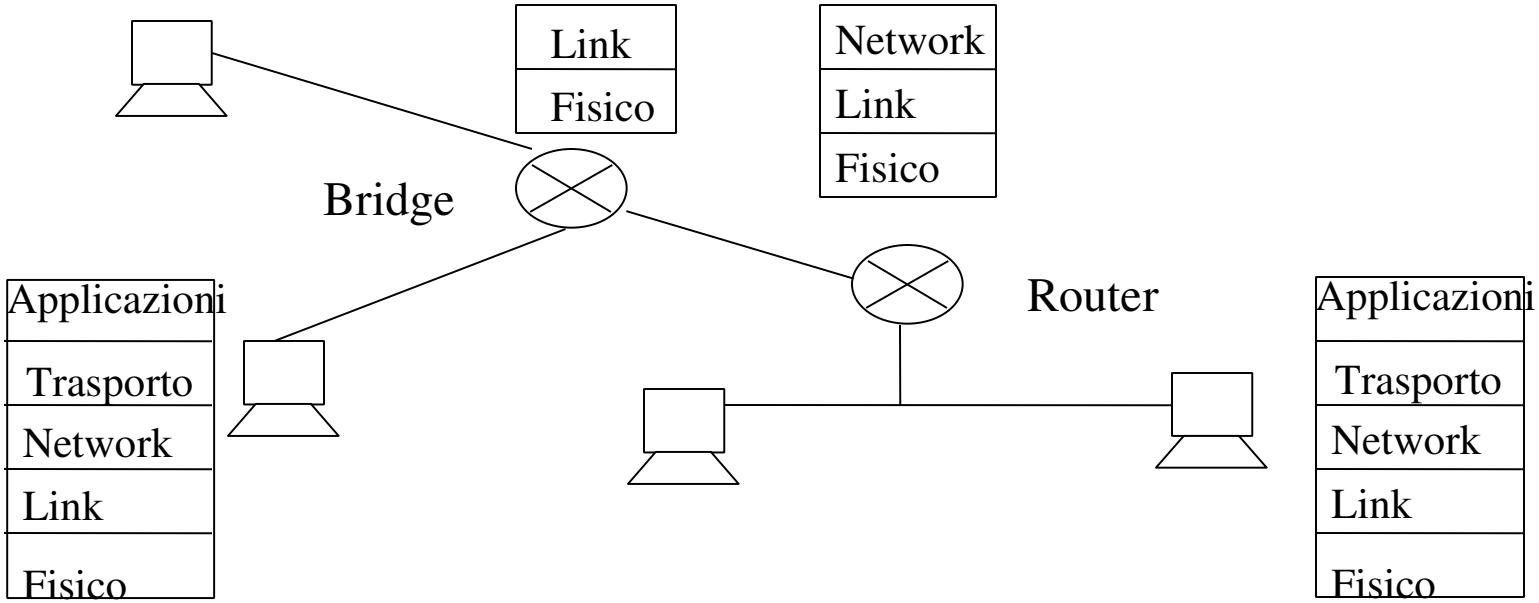
- servizi offerti: invio del singolo bit tra due nodi (routers o end systems) della rete

- i servizi forniti dipendono dal collegamento fisico che collega i due links

- lo stesso protocollo a livello link può utilizzare protocolli diversi a livello fisico (ad esempio Ethernet definisce protocolli diversi per collegamenti su fibra ottica, doppino di rame,…)

ARCHITETTURA DI RETE

Diversi nodo della rete utilizzano diversi livelli dello stack



LIVELLO DELLE APPLICAZIONI

Applicazione di Rete = e' composta da diverse componenti (programmi) che sono in esecuzione su nodi diversi della rete

Esempio: *Web* è una applicazione di rete che consente ad un *client* di ottenere documenti da un *server.* Utilizza:

- *diverse componenti:* web browsers (es: Mozilla, Opera, Netscape Navigator), web servers (Apache, Microsoft Servers,…),
- una *rappresentazione standard* dei documenti (HTML) ed un protocollo standard per la comunicazione tra le componenti (HTTP)

Protocollo a livello applicazione =HTTP (solo una parte della applicazione)

Altri esempi: posta elettronica utilizza il protocollo SMTP (Simple Mail Transfer Protocol)

LIVELLO DELLE APPLICAZIONI

Applicazioni distribuite:

Ogni componente di una applicazione distribuita è incapsulata in un *processo*

Processo = Programma in esecuzione su un nodo della rete

Comunicazione tra processi= Se i processi sono in esecuzione sullo stesso host avviene mediante meccanismi offerti dal sistema operativo

Se i processi sono distribuiti (sono in esecuzione su nodi diversi) la comunicazione avviene mediante *scambio di messaggi.*

Protocollo a livello applicazione= Definisce il formato dei messaggi scambiati tra i processi e l'ordine con cui vengono scambiati i messaggi

LIVELLO DELLE APPLICAZIONI

Scambio di messaggi tra processi

- punto a punto: tra una sola coppia di host (*unicast*)
- multipunto: tra un insieme di hosts (*multicast*)

Socket = Meccanismo attraverso il quale avviene lo scambio dei messaggi. E' l'interfaccia tra il livello applicazioni ed il livello di trasporto

Identificazione (Indirizzamento dei processi) Indirizzo IP dell'host su cui e' in esecuzione il processo + numero di porta

Indirizzo IP= identifica *univocamente* una interfaccia di *rete di un host*

IDENTIFICAZIONE DEI SERVIZI

Per identificare un processo con cui si vuole comunicare, occorre identificare

- la rete all'interno della quale si trova l'host su cui e' in esecuzione il processol'host all'interno della rete
- Il processo in esecuzione sull'host

Identificazione dell'host = definita dal protocollo IP (Internet Protocol)
Faremo riferimento ad IPv4 (IP versione 4). Ipv6 (versione 6) si basa sugli stessi principi

Identificazione del processo = utilizza il concetto di *porta*

Porta = Intero da 0 a 65535

INDIRIZZI IP: STRUTTURA

Ipotesi semplificativa: Un host ha una unica interfaccia di rete.

- Ogni host su una rete IPv4 è identificato da un numero rappresentato su 4 bytes

  indirizzo IP dell'host (assegnato da IANA).

(se un host, ad esempio un router, presenta più di una interfaccia sulla rete ⇒ più indirizzi IP per lo stesso host, uno per ogni interfaccia)

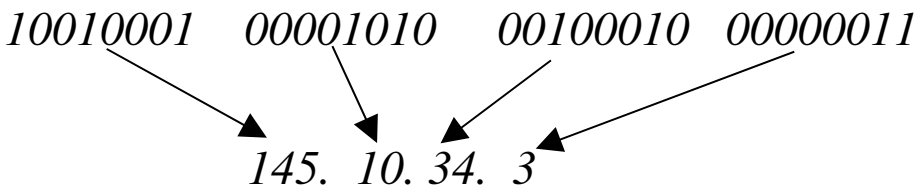
- indirizzi disponibili in IPv4, $2^{32} = 4.294.967.296$ (specifica IPV4 Settembre 1981)

- alcuni indirizzi sono *riservati:* ad esempio indirizzi di *loopback,* individuano il computer su cui l'applicazione è in esecuzione

- Indirizzi IP del mittente e del destinatario inseriti nel pacchetto IP

INDIRIZZI IP

Esempio:



Ognuno dei 4 *bytes,* viene interpretato come un numero decimale *senza segno*

$\Rightarrow$

Ogni valore varia da 0 a 255 (massimo numero rappresentabile con 8 bits)

Evoluzione: in Ipv6 indirizzo IP comprende 16 bytes

INDIRIZZI IP:
ASSEGNAZIONI STATICA VS. DINAMICA

Assegnazione degli indirizzi IP:

- *Statica:* ad un host viene assegnato un indirizzo IP fisso (utilizzata ad esempio per i servers)
- *Dinamica:* l'indirizzo IP di un host può variare *dinamicamente.*

Assegnazione dinamica:

- utilizzata per hosts che si collegano in modo temporaneo alla rete (es collegamento telefonico). Viene assegnato un nuovo indirizzo IP al momento del collegamento
- utilizza *servers DHCP* (Dynamic Host Configuaration Protocol)
- dal punto di vista del programmatore di rete: tenere presente che l'indirizzo IP di un host può cambiare in relazione ad esecuzioni diverse della stessa applicazione, ed anche durante una stessa esecuzione della applicazione

INDIRIZZI IP

Classfull Addressing: un indirizzo IP strutturato seocndo unagerarchia a due livelli

Network Prefix + Host Number

Network Prefix : identifica la rete in cui si trova l'host

Host Number : identifica il perticolare host su quella rete

- tutti gli host di una rete presentano *lo stesso prefisso*
- due hosts in reti diversi possono avere lo stesso host number
- è possibile specificare network prefixes di lunghezza diversa, in questo modo si possono individuare reti di dimensioni diverse:

  maggiore la dimensione del network prefix, minore la il numero di hosts sulla rete)

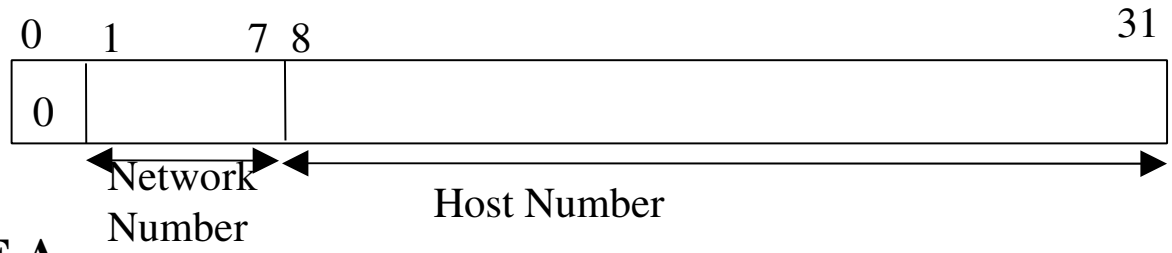
CLASSI INDIRIZZI IP

- IPv4 definisce 5 classi di indirizzi IP (A,B,C,D,E)
  - Classi A,B,C caratterizzate da diverse lunghezze dei network prefixes (quindi da diverso numero di hosts per rete)..
  - Classe D utilizzata per *indirizzi di multicast* (la vedremo meglio in seguito).
  - Classe E riservata
- La classe determina il confine tra il network prefix e l'host number
- Ogni classe è individuata dalla *configurazione dei primi bits* dell'indirizzo IP (esempio indirizzi in classe A hanno il primo bit dell'indirizzo =0)

$\Rightarrow$

questo semplifica le operazioni effettuate dal router

STRUTTURA INDIRIZZI IP: CLASSFULL ADDRESSING



CLASSE A

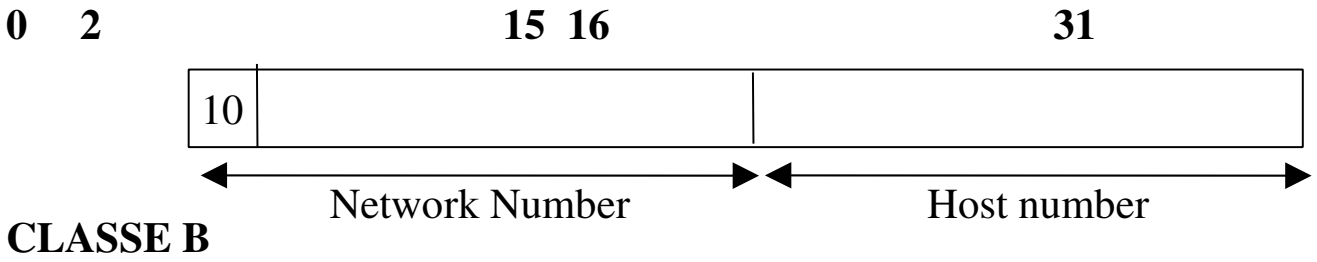
· numero di reti = $2^7 - 2$

• (0.0.0.0 riservato, 127.xxx.xxx.xxx *indirizzi di loopback)* = indirizzo che individua il *computer locale* su cui è in esecuzione l'applicazione a cui corrisponde il nome *localhost*)

· n umero di hosts = $2^{24} - 2$ (circa 16 milioni di hosts, configurazioni di tutti 0 e tutti 1 sono riservate, tutti 1 = broadcast sulla rete)

· indirizzi utilizzabili: da 1.xxx.xxx.xxx a 126.xxx.xxx.xxx

STRUTTURA INDIRIZZI IP: CLASSFULL ADDRESSING



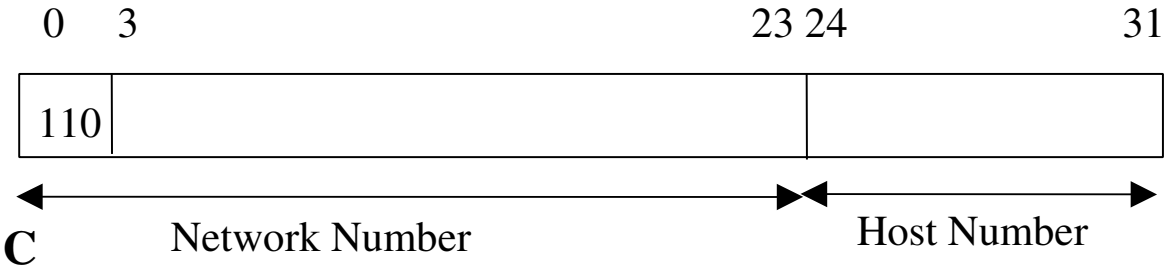
CLASSE B

· **n**umero di reti = 2^{14} (circa 16.000)

· numero di hosts = $2^{16} - 2$ (circa 65000 hosts per rete, configurazioni di tutti 0 e tutti 1 sono riservate)

· indirizzi utilizzabili = da 128.0.xxx.xxx a 191.255.xxx.xxx

STRUTTURA INDIRIZZI IP: CLASSFULL ADDRESSING



CLASSE C

· **n**umero di reti = 2^{21}

· numero di hosts = $2^{8} - 2$ (254 hosts, configurazioni di tutti 0 e tutti 1 sono riservate.)

· Indirizzi utilizzabili: 192.0.0.xxx a 233.255.255.xxx

DOMAIN NAME SERVER

Problema: gli indirizzi IP semplificano l'eleborazione effettuata dai routers, ma risultano poco leggibili per gli utenti della rete

Soluzione: assegnare un *nome simbolico unico* ad ogni host della rete si utilizza uno spazio *di nomi gerarchico*
esempio: fujih0.cli.di.unipi.it (host fuji presente nell'aula H alla postazione 0, nel dominio cli.di.unipi.it)

- livelli della gerarchia separati dal punto.
- nomi interpretati da destra a sinistra (diverso dalle gerarchia di LINUX)alla radice della gerarchia:
- un dominio per ogni paesi + *big six* (edu,com,gov,mil,org,net)
- la corrispondenza tra nomi ed indirizzi IP gestita da un servizio di nomi

DNS = Domain Name System

INDIRIZZAMENTO

Su ogni host possono essere attivi contemporaneamente più *servizi* (es: e-mail, ftp, http,…)

Ogni servizio viene incapsulato in un diverso processo

L'indirizzamento di un processo avviene mediante il concetto di *porta*

Porta = intero compreso tra 1 e 65535. Non *è un dispositivo fisico*, ma una *astrazione* per individuare i singoli servizi (processi).

Porte comprese tra 1 e 1023 riservati per particolari servizi. Linux :solo i programmi in esecuzione su root possono ricevere dati da queste porte. Chiunque può inviare dati a queste porte.

- porta 7 *echo*
- porta 22 *ssh*
- Porta 80 *HTTP*

APPLICAZIONI DI RETE

Applicazione	Protocollo a livello applicazioni	Protocollo a livello trasporto
Posta Elettronica	SMTP	TCP
Login Remota	SSH	TCP
Web	HTTP	TCP
Trasferimento File	FTP	TCP
File Server Remoto	NFS	UDP o TCP
Multimedia	Proprietario	UDP o TCP
Telefonia	Proprietario	UDP
DNS (Domain Name Server)	….	…..

WORLD WIDE WEB: URLS

World Wide Web: insieme di clients e servers che cooperano mediante il protocollo HTTP.

Clients = programmi dotati di una interfaccia grafica (browser)(Netscape, Explorer, Mozilla, Opera,…) che consentono di reperire informazioni memorizzate su un server web

Informazioni localizzate mediante una *URL (Uniform Resource Locator)*

Esempio: URL *http://www.cs.princeton.edu/index.html* individua un *oggetto* all'interno del web

URL è composto di due parti: nome dell'host http://www.cs.princeton.edu + *pathname* che individiua l'oggetto all'interno dell'host

WORLD WIDE WEB: CARATTERISTICHE

Accesso ad un oggetto individuato da una URL, ad esempio
http://www.cs.princeton.edu/index.html

- apertura di una connessione TCP dal browser web al server web di nome http://www.cs.princeton.edu (*porta 80 di default*)
- invio di un messsaggio di richiesta sulla connessione aperta
- il server web ricerca l'oggetto individuato *dal path index.html* ed invia il contenuto del file al browser
- Il server chiude la connessione con il client (*connessione non persistente*)

WORLD WIDE WEB: CARATTERISTICHE

- l'oggetto O inviato dal server S al client *(index.html)* può contenere *collegamenti ipertestuali*, cioè URL che puntano ad altri file, allocati su S stesso o su altri servers

- quando l'utente seleziona un collegamento ipertestuale, il browser apre una nuova connessione TCP per recuperare l'oggetto individuato

- viene creata una nuova connessione anche nel caso in cui gli oggetti riferiti siano memorizzati su S stesso (cooneessione non persistente)

Connessione non persistente: per ogni connessione client/server una sola richiesta ed una sola risposta

WORLD WIDE WEB: PROTOCOLLO HTTP

Protocollo HTTP: insieme di regole che definiscono formati e ordine dei messaggi
scambiati tra un browser ed un server web.

Caratteristiche protocollo HTTP:

- protocollo a livello applicazione
- Utilizza i servizi TCP
- *stateless protocol:* non memorizza informazioni relative ai clients serviti
- tipo di connessione:
  - HTTP 1.0 connessioni *non persistente :* viene aperta una nuova connessione per ogni *richiesta/risposta* $\Rightarrow$ inefficiente
    (2 RTT, uno per la connessione, uno per richiesta/risposta)
  - HTTP 1.1 *connessioni persistenti:* una sola connessione per più messaggi di richiesta/risposta sulla stessa connessione TCP

WORLD WIDE WEB: PROTOCOLLO HTTP

Connessioni persistenti (HTTP 1.1): il server lascia aperta la connessione TCP dopo aver inviato la risposta al client, in attesa di ulteriori richieste

Vantaggi

- minor overhead nella creazione di connessioni
- miglior utilizzazione del meccanismo TCP di controllo *di congestione*

Problemi

- nel server: decidere il momento in cui chiudere una connessione

  $\Rightarrow$

  uso di *timeouts* = il server chiude una connessione dopo che è trascorso un intervallo di tempo in cui non ha ricevuto richieste dal client

WORLD WIDE WEB: PROTOCOLLO HTTP

Struttura generale dei messaggi HTTP:

REQUEST LINE <CRLF>
MESSAGE HEADER <CRLF><CRLF>
MESSAGE BODY <CRLF>

CRLF = Carriage Return + Line Feed

Messaggio di richiesta HTTP (dal client al server):body vuoto

GET index.html HTTP/1.1 <CRLF>
Host: http://www.cs.princeton.edu <CRLF>
Connection: close <CRLF>
User Agent: Mozilla/4.0 <CRLF>
Accept Language:it <CRLF>
<CRLF><CRLF>

WORLD WIDE WEB: PROTOCOLLO HTTP

Messaggio di risposta: un esempio

HTTP/1.1 202 accepted <CRLF>
Connection: close <CRLF>
Date:…. <CRLF>
Server: Apache/1.3.0 (Unix) <CRLF>
Last Modified: Mon 24 Oct 2005<CRLF>
Content Length: 6821<CRLF>
Content Type: text/HTML<CRLF><CRLF>
dati
<CRLF>

Il corpo del messaggio contiene la pagina HTML

AUTENTICAZIONE DEI CLIENTS

Meccanismo soft di autenticazione mediante *username* + *password*

client: invia la richiesta

server: risponde con un messaggio di richiesta autenticazione

client: visualizza la richiesta di autenticazione ed attende i dati dall'utente. Rispedisce al server username+password e le memorizza (*caching*)

server: autentica l'utente

client: rispedisce username+password in tutte le richieste successive utilizzando quelle memorizzate nella cache

IL MECCANISMO DEI COOKIES

Cookies: utilizzati da un server per registrare gli utenti serviti (introduzione di stato nel protocollo)

Per ogni client servito: generazione di un *identificatore unico*

invio dell'identificatore al client (esempio *set-ccokie: 12657* nell'header del messaggio)

il client memorizza *in un file locale l'identificatore ricevuto.*

l'identificatore viene utilizzato successivamente quando il client invia nuove richieste al server (esempio cookie: 12657)

Cookie: può contenere anche informazioni più complesse

IL MECCANISMO DEI COOKIES

Utilizzazione

- autenticare un utente senza richiedere *username+password* ogni volta che l'utente si collega al server

- ricordare le preferenze dell'utente (es: pagine visitate,…) per riproporle successivamente

MECCANISMI DI CACHING: CONDITIONAL GET

il client memorizza gli oggetti ricevuti dal server in una *cache*

ad ogni oggetto viene associata la data dell'ultima modifica
(es: 15/10/2005 15:06:29)

successivamente, il client richiede nuovamente l'oggetto mediante una *Conditional GET*

GET index.html HTTP/1.1 <CRLF>
Host: http://www.cs.princeton.edu <CRLF>
User Agent: Mozilla/4.0
if-modified-since: 15/10/2005 15:06:29

Il server invia l'oggetto solo se è stato modificato dopo la data inviata