

## Slide introduzione al corso

RICEVIMENTO: venerdì 14:00 - 16:00

**Rete:** interconnessione di dispositivi in grado di scambiarsi informazioni, quali sistemi, terminali, router, switch e modem

I sistemi terminali sono chiamati host:

- Macchina dell'utente finale
- Server

Dispositivi di interconnessione:

- Router
- Switch

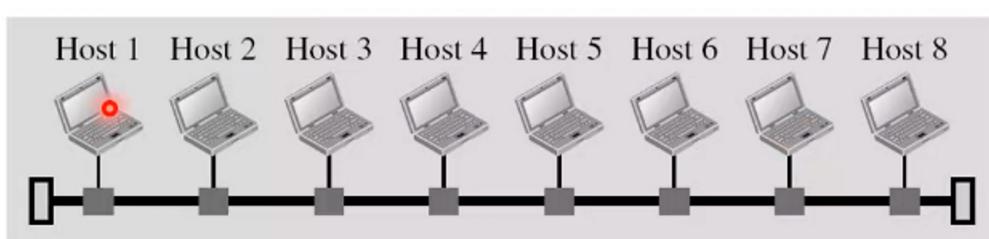
Collegamenti(link): mezzi trasmissivi (cablati, wireless)

### LAN

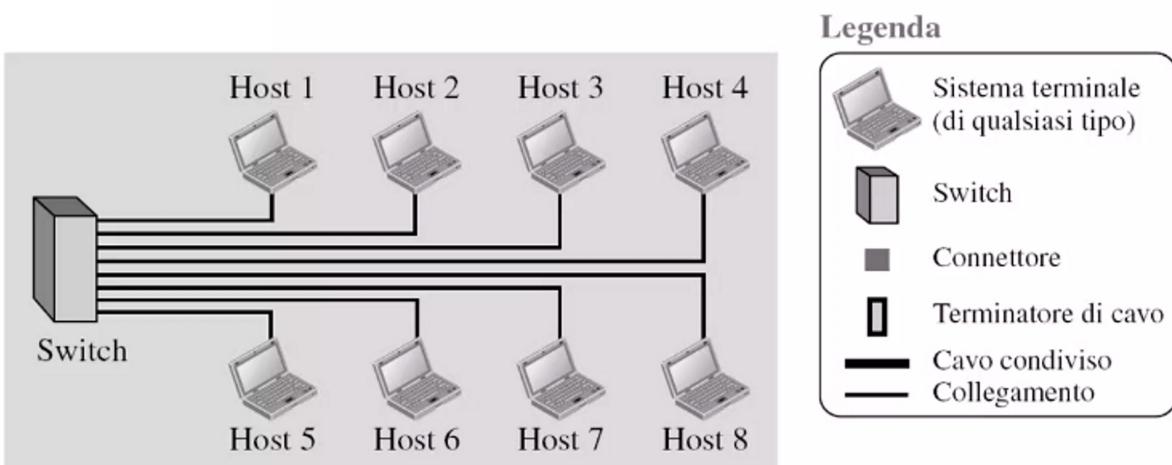
Reti di computer circoscritte ad un'area limitata, sono di proprietà di un'organizzazione privata

Hanno un'estensione che arriva fino a qualche km

Vengono usate per connettere sistemi terminali personali tramite cavo in rame o wireless principalmente



a. LAN con cavo condiviso (obsoleta)



b. LAN con switch (moderna)

Tecnologia a bus decisamente obsoleta e orrida, adesso si usano degli switch

### WAN (Wide Area Network)

Una Wan (o rete geografica) è una rete il cui compito è interconnettere LAN di singoli host separati da distanze geografiche

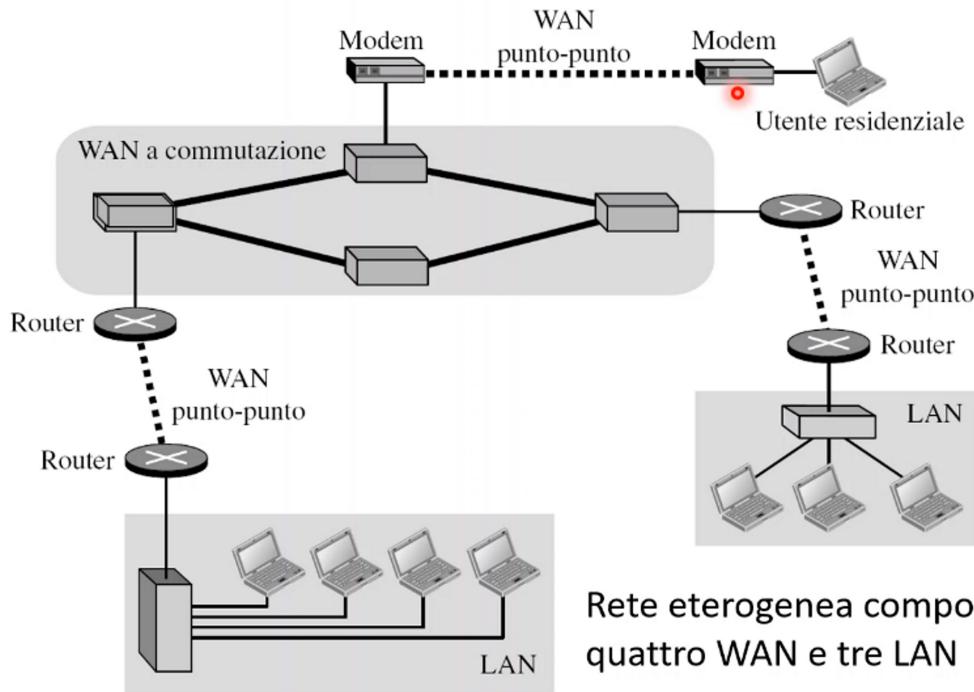
WAN punto a punto: collega due dispositivi tramite un mezzo trasmissivo

### WAN a commutazione

Collega più di due punti di terminazione (es dorsali internet)

Elementi di commutazione: elaboratori specializzati utilizzati per connettere fra loro due o più linee di trasmissione.

Si collegano in genere due reti locali (LAN) con una rete WAN punto a punto (es uffici della stessa organizzazione)



## Commutazione

Una internet è data dall'interconnessione di reti, composte da link e dispositivi capaci di scambiarsi informazioni

Dispositivi involved:

- Sistemi terminali (host)
- Dispositivi di interconnessione che si trovano nel percorso tra sorgente e destinazione

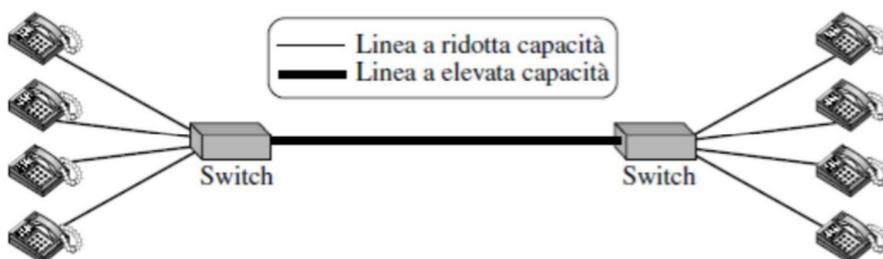
Tecniche di commutazione: (modalità con cui viene determinato il percorso sorgente-destinatario)

- Circuit switched network- reti a commutazione di circuito
- Packet-switched network - reti a commutazione di pacchetto

### Commutazione di circuito

- Instaura un cammino dedicato tra i due dispositivi che vogliono comunicare
- Sul percorso vengono dedicate risorse alla comunicazione (canale logico o circuito)
- Le risorse allocate sono garantite per tutta la durata della comunicazione

<b>Vantaggi</b>	- Performance (garantita) es. rete telefonica fissa tradizionale
<b>Svantaggi</b>	- Necessaria una fase di instaurazione della comunicazione (setup) - Le risorse rimangono inattive se non utilizzate



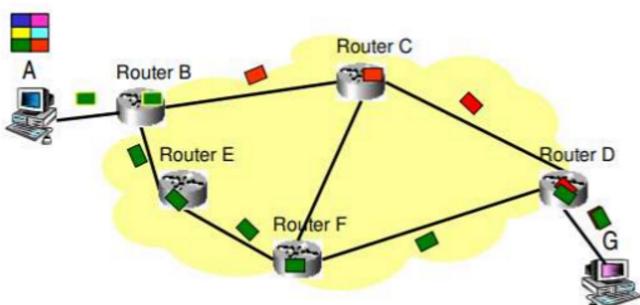
### Commutazione di pacchetto

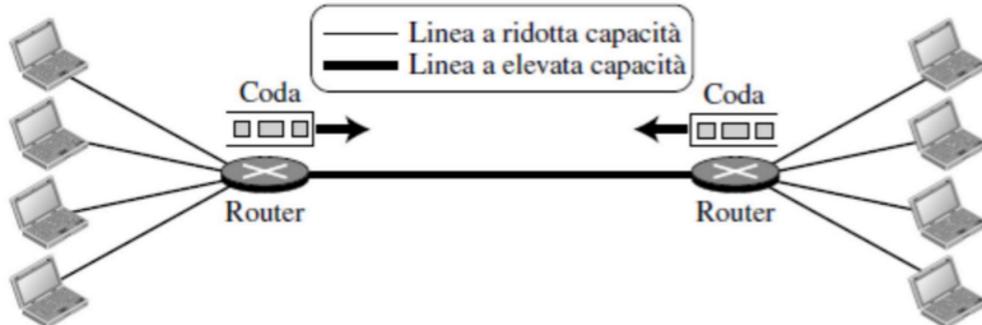
Sfrutta le risorse della rete in modo diverso.

**Il flusso di dati punto punto viene suddiviso in pacchetti.**

I pacchetti degli utenti A e G condividono le risorse di rete

I pacchetti sono indipendenti l'uno dall'altro della stessa comunicazione, possono seguire lo stesso percorso o percorsi diversi, sia pacchetti della stessa comunicazione che quelle diverse.





Il router riceve un pacchetto e lo inoltra sul collegamento a elevata capacità, se il collegamento è pieno i pacchetti vengono messi in un buffer.

L'introduzione del buffer crea congestione e potenziale ritardo nell'arrivo dei pacchetti.

Il buffer ha capacità limitata quindi in casi gravi di congestione vengono persi dei pacchetti

*Congestione: accodamento dei pacchetti, attesa per l'utilizzo del collegamento*

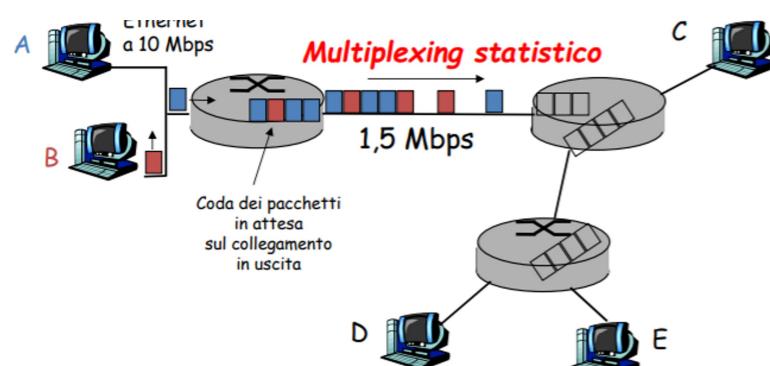
### Trasmissione store and forward

Un router aspetta che gli arrivi tutto il pacchetto poi lo analizza e lo spedisce

Aspetta che gli arrivi tutto il pacchetto prima di poter trasmettere -> *ritardo di store and forward*

Attesa dei pacchetti in code di output (buffer) -> *ritardi di coda*

I buffer hanno dimensione finita -> *perdita di pacchetti*



In questo caso ho una connessione ethernet a 10 mbps e un collegamento a 1.5, quindi al router arriveranno un tot di pacchetti che non riuscirà a instradare immediatamente, ma dovrà metterli nel buffer.

La sequenza dei pacchetti non segue un ordine prefissato: *condivisione di risorse su richiesta* (multiplexing statistico delle risorse)

### Circuit vs packet switching

- **esempio**

- N=35 utenti condividono un link 1 Mbps
- Ogni utente genera 100kbps quando è “attivo”
- Ogni utente è attivo 10% del tempo

- **Circuit Switching**

Con la commutazione di circuito, occorre riservare 100kbps per ogni utente, in ogni istante. Il link di output può quindi supportare simultaneamente al massimo  $1\text{Mbps}/100\text{kbps} = 10$  utenti

- **Packet Switching**

- 10 o meno utenti simultaneamente attivi -> banda richiesta  $\leq 1$  Mbps, nessun ritardo
- Più di 10 utenti attivi simultaneamente -> frequenza aggregata di arrivo dei dati supera la capacità del collegamento in uscita e quindi si ha ritardo di accodamento
- N.B. la probabilità che ci siano 10 o meno utenti attivi contemporaneamente è 0.9996
- con alta probabilità la tecnica del packet switching supporta tutti i 35 utenti senza introdurre alcun ritardo!

## Circuit Switching

- **Vantaggi**
  - Prestazioni garantite
  - Overhead limitato
  - Tecnologie di switching efficienti
- **Svantaggi**
  - E' richiesta la segnalazione per instaurare il circuito (configurare le tabelle di switching)
  - Sottoutilizzo delle risorse in presenza di traffico a raffica (burst) e rate di traffic variabile

## Packet Switching

- **Vantaggi**
  - Risorse trasmissive usate solo se necessario
  - Segnalazione non richiesta
- **Svantaggi**
  - Overhead
  - Tecnologie di inoltro non efficienti (necessità di selezionare l'uscita per ogni pacchetto)
  - Tempo di elaborazione ai router (routing table lookup)
  - Accodamento ai router

26

## Internet

Una internet è costituita da due o più reti interconnesse, la internet più famosa è **Internet**, composta da migliaia di reti interconnesse, ogni rete connessa a Internet deve usare IP (internet protocol) e rispettare certe convenzioni su nomi e indirizzi

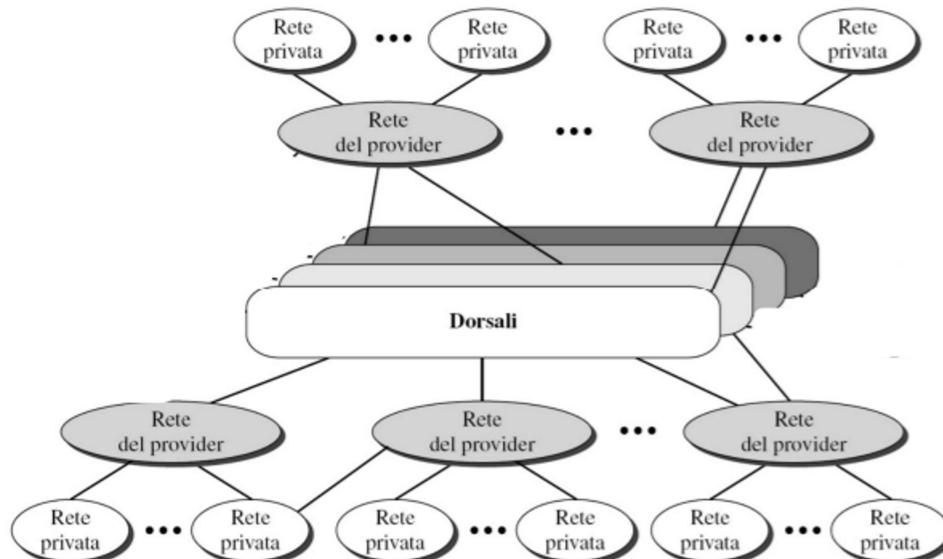
Infrastruttura che fornisce servizi di comunicazione alle applicazioni

I tipi di servizi che possono essere utilizzati sono:

- Senza connessione: senza garanzia di consegna
- Orientati alla connessione: garantiti in integrità, completezza e ordine

### Entità software:

Applicazioni, protocolli (regolamentano le trasmissioni), interfacce (come uno strat dello stack si interfaccia con l'altro)



Tante reti private interconnesse tra loro tramite reti via via più grandi. Le dorsali sono poche, ad alta capacità, interconnesse tra loro

Peering point: accordo tra due ISP di accettare e inoltrare il traffico che ricevono dall'altro

- IXP Internet eXchange Point: punto d'incontro per il peering tra due o più ISP

Il collegamento che connette l'utente al primo router di internet è detto rete di accesso:

- Accesso via rete telefonica (dial-up, dsl, fibra ottica)
- Accesso tramite reti wireless
- Collegamento diretto

## Metriche

Misura delle prestazioni di rete

- Ampiezza di banda e bitrate
- Throughput
- Latenza
- Perdita di pacchetti

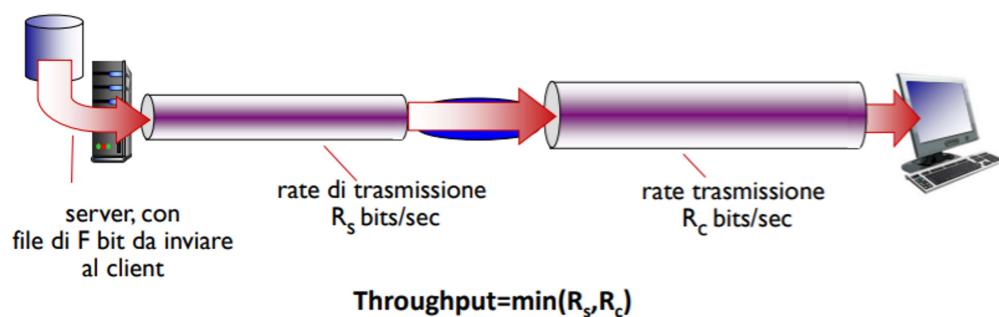
La banda/ampiezza di banda è la larghezza di un intervallo di frequenze. Un segnale è una composizione di più frequenze, l'intervallo che si può utilizzare è la banda. (Hz)

La velocità di trasmissione è la quantità di bit che possono essere trasmessi nell'unità di tempo su un certo collegamento. (bits/secondo o bps)

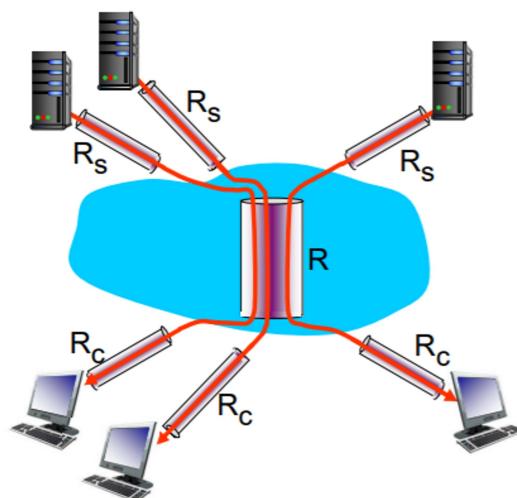
Il bit rate dipende dalla larghezza di banda e dalla tecnica trasmissiva usata

Throughput: quantità di dati utili che possono essere trasmessi tra un nodo sorgente e un nodo destinazione in un certo intervallo di tempo.

Si intende da un nodo terminale all'altro, il bitrate è la capacità trasmissiva di un collegamento, nel mio percorso posso avere più collegamenti con più bitrate diversi.



Non sono solo nel collegamento, il collegamento è condiviso per più comunicazioni



**End-to-end throughput** per connessione:  $\min(R_c, R_s, R / 10)$  (con 10 collegamenti alla rete sopra)  
 Il throughput di questo collegamento non può essere più grande del minimo tra questi valori.  
 In rete mando sia i dati del mio file che i dati relativi ai protocolli per viaggiare in internet.

- Latenza: tempo richiesto affinché un messaggio arrivi a destinazione dal momento in cui il primo bit parte dalla sorgente

$$\text{Latenza} = \text{ritardo di propagazione} + \text{ritardo di trasmissione} + \text{ritardo di accodamento} + \text{ritardo di elaborazione}$$

<Lez 3 20/09/2021>

Come si verificano ritardi e perdite?

I pacchetti si accodano nei buffer dei router

Il tasso di arrivo dei pacchetti sul collegamento eccede la capacità del collegamento in uscita di evaderli; i pacchetti si accodano, in attesa del loro turno

- Pacchetti in attesa di essere trasmessi **ritardo**
- Pacchetti accodati **ritardo**
- Buffer liberi (disponibili): se non ci sono buffer liberi i pacchetti in arrivo vengono scartati **perdita**

**Cause di ritardo:**

Ritardo di elaborazione del nodo	Ritardo di accodamento	Ritardo di trasmissione (L/R)	Ritardo di propagazione (d/s)
<ul style="list-style-type: none"> <li>- <b>Controllo errori sui bit</b></li> <li>- Determinazione del canale di uscita</li> </ul>	<ul style="list-style-type: none"> <li>- <b>Attesa di trasmissione</b></li> <li>- Dipende da intensità e tipo di traffico (che influiscono sul numero di pacchetti in coda nel buffer)</li> </ul>	<p><b>Tempo impiegato per trasmettere un pacchetto su un link.</b></p> <p>R = rate di trasmissione del collegamento (bps)  L = lunghezza del pacchetto (bit)</p>	<p><b>Tempo impiegato da 1 bit per essere propagato da un nodo all'altro</b></p> <p>d = lunghezza del collegamento fisico (m)  s = velocità di propagazione del mezzo (<math>3-2 \times 10^8</math> m/sec)</p>

**Ritardo end-to-end:**

N-1 router tra origine e destinazione

$$\text{Ritardo(nodo)} = \text{ritardo(el)} + \text{Ritardo (tr)} + \text{Ritardo (pr)}$$

$$\text{Ritardo(end-to-end)} = N * \text{Ritardo(nodo)}$$

$$\text{Ritardo}_{\text{nodo}} = \text{Ritardo}_{\text{el}} + \text{Ritardo}_{\text{tr}} + \text{Ritardo}_{\text{pr}}$$

$$\text{Ritardo}_{\text{end-to-end}} = N * \text{Ritardo}_{\text{nodo}}$$

nel caso generale di ritardi eterogenei -> sommatoria

$$\text{Ritardo}_{\text{end-to-end}} = \text{Ritardo}_{\text{nodo1}} + \text{Ritardo}_{\text{nodo2}} + \dots + \text{Ritardo}_{\text{nodoN}}$$

Traceroute: comando che traccia un pacchetto dal tuo computer all'host mostrando il numero di salti per raggiungerlo e al ritardo dal mittente per ogni passaggio

# Lez 3 - 20/09/2021

lunedì 20 settembre 2021 11:17

## Protocollo

Insieme di regole che permettono a due entità di comunicare.

Il modello usato nei modelli di comunicazioni è un modello stratificato:

Una serie di passi. Ad ogni passo viene eseguito un determinato compito su un messaggio che viene integrato e trasferito ad un altro agente seguendo specifiche regole di esecuzione

Per ogni strato contenitore del messaggio e indirizzo

Stratificare per semplificare sistemi complessi.

La struttura esplicita permette l'identificazione delle relazioni tra gli elementi di un sistema complesso

- Modello di riferimento stratificato
- Suddivisione di funzioni e attori

La modularizzazione facilita la manutenzione e l'aggiornamento dei dati

- Un modulo (livello/strato) svolge un insieme delimitato di compiti e nel sistema appare come una black box (input/output)
- Ogni livello offre servizi allo stato superiore
  - o Implementa azioni all'interno del livello stesso
  - o Utilizza servizi del livello inferiore

Separazione tra servizi offerti (interfaccia) e implementazione: il cambiamento dell'implementazione di un servizio in un livello è sostanzialmente trasparente per il resto del sistema.

## Principi base:

Separation of Concern: separazione delle responsabilità

Information hiding: nascondere le informazioni che non sono indispensabili per il committente.

- **Modello stratificato**
  - costituito da sistemi di consumatori/produttori
  - sistemi organizzati in strati funzionali (livelli)
  - ogni strato fornisce servizi allo strato superiore e usa i servizi di quello inferiore
  - ogni strato scambia informazioni direttamente solo con gli strati adiacenti
  - in ogni comunicazione i due strati omologhi svolgono funzioni reciproche
  - esistono sistemi (intermedi) che implementano solo alcune funzioni
- **Requisiti**
  - efficiente: minimizzare lo sforzo globale di consegnare le lettere (non la singola!)
  - efficace: consegnare la maggior quota possibile di lettere

Strati omologhi svolgono funzioni reciproche.

I livelli possono evolvere con la tecnologia senza dover cambiare gli altri strati, perché sono indipendenti

## Vantaggi del modello stratificato:

- Scomponi il problema in sottoproblemi più semplici da trattare -> il singolo strato è più semplice del sistema nel suo complesso (semplifica progettazione, implementazione e manutenzione del software)
  - Rende i vari livelli indipendenti
- I servizi forniti dagli strati inferiori possono essere usati da più entità negli strati adiacenti

superiori

I livelli diversi possono essere sviluppati da soggetti diversi

Ogni livello logico di astrazione è realizzato in un apposito strato; un livello viene creato quando si rende necessario un diverso grado di astrazione

Ogni strato svolge una sola e ben definita funzione

Il flusso dati attraverso le interfacce di ogni strato deve essere minimizzato

Il numero degli strati deve essere minimizzato compatibilmente con la loro complessità

### **Modello ISO/OSI.**

Modello di riferimento per i sistemi di comunicazione in rete

Obiettivo:

Definire uno standard per cui definendo delle regole comuni fosse possibile far parlare qualsiasi dispositivo tra una rete e l'altra.

### **Definizione di uno standard APERTO:**

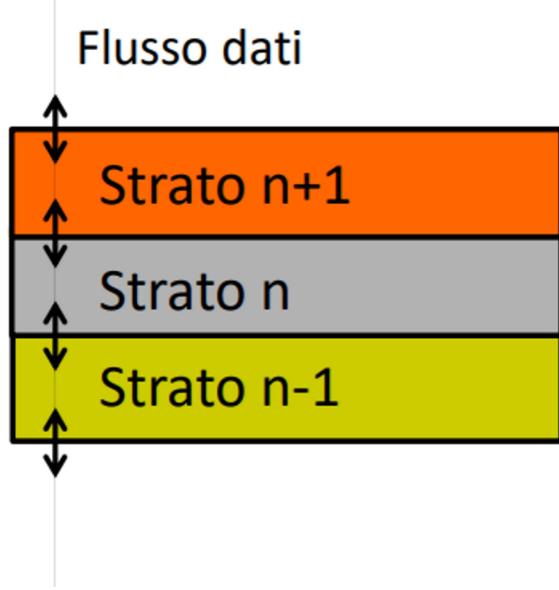
- I dettagli del protocollo sono disponibili pubblicamente
- I cambiamenti sono gestiti da un'organizzazione la cui partecipazione è aperta al pubblico

Un sistema che implementa protocolli aperti è un **sistema aperto**

L'organizzazione che ha definito questo standard è la ISO, che ha definito il modello di riferimento Open Systems Interconnection(OSI). Diventato standard internazionale nel 1983.

Modello stratificato, ogni strato ha una propria funzionalità.

La comunicazione tra i vari livelli è assicurata da chiamate standard; ogni livello è tenuto a rispondere in maniera corretta alle chiamate che gli competono e che verranno generate dai due livelli a esso adiacenti



Strato n offre un servizio a n+1 e chiede un servizio a n-1. Il flusso dati va in verticale. Lo strato n-esimo di una entità comunica con lo strato ennesimo dell'altra entità (es destinazione )

#### **Strato:**

Modulo definito attraverso servizi, protocolli e interfacce che lo caratterizzano

#### **Servizio:**

Insieme di primitive (operazione che un o strato fornisce aduno strato soprastante)

#### **Interfaccia:**

Insieme di regole che governano il formato e il significato dei dati che vengono scambiati tra due strati adiacenti della stessa entità

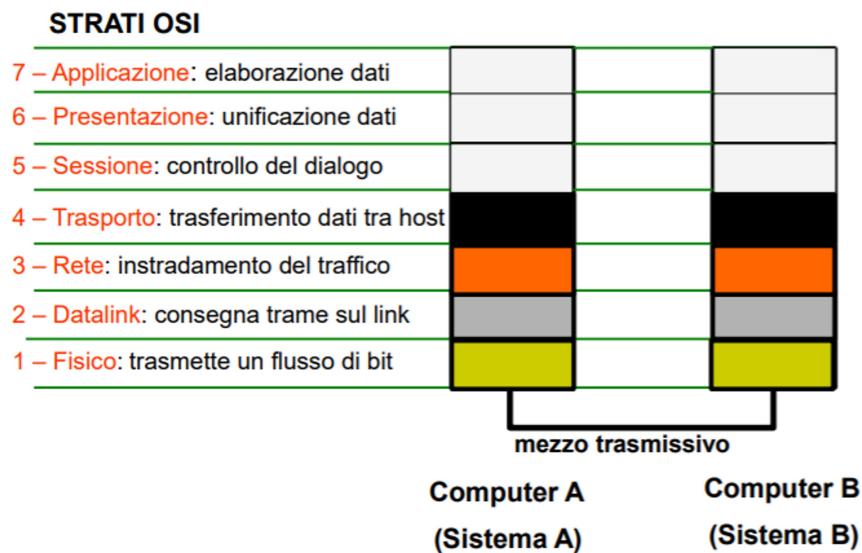
## Protocollo:

Insieme di regole che:

- Permettono a due entità omologhe (stesso trato) uno scambio efficace ed efficiente delle informazioni
- Definiscono il formato e l'ordine dei messaggi inviati e ricevuti tra entità omologhe della rete e le azioni che vengono fatte per la trasmissione e ricezione dei messaggi
- *Efficace*: un sistema che riesce a raggiungere lo scopo prefissato con la maggior frequenza possibile
- *Efficiente*: un sistema che riesce a raggiungere lo scopo prefissato con il minor sforzo possibile

Specifica la sintassi, la semantica e le azioni da intraprendere dopo la ricezione di un msg

**ISO/OSI è stratificato in 7 livelli**



Tipicamente i primi 3 livelli sono intermedi (1-2-3)

# Lez 4 - 22/09/2021

mercoledì 22 settembre 2021 11:04

Da slide 24 intro 3

I nodi intermedi non implementano tutti gli strati, implementano funzioni chiamate end-to-end.

I servizi trasporto, rete e collegamento possono offrire allo strato adiacente e superiore un servizio *connection oriented* e un servizio *connectionless*

**Connection oriented** stabilisce una connessione, instrada i dati e chiude la connessione

**Connectionless** instrada i dati senza stabilire una connessione.

Flusso dell'informazione

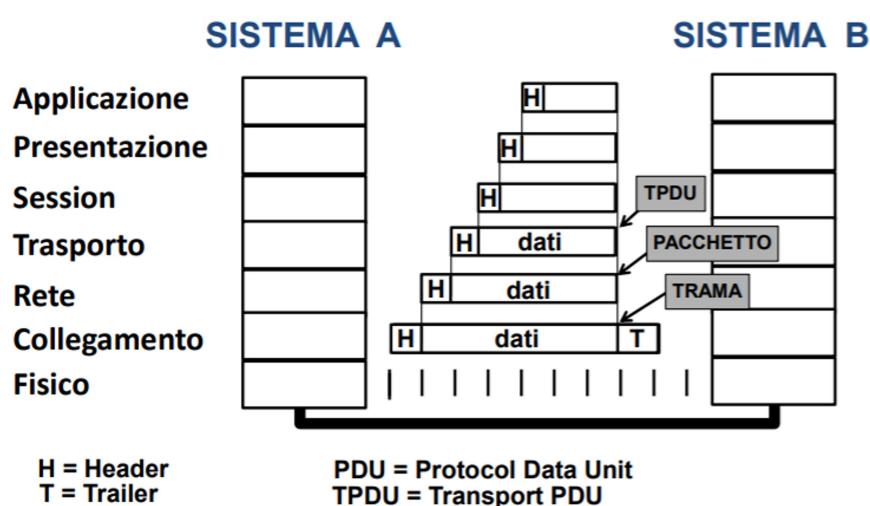
## *incapsulamento*

Tipicamente l'informazione, a livello di dati, ha origine al livello applicativo, l'informazione discende i livelli fino al canale fisico. Ogni livello aggiunge delle informazioni, un'intestazione che racchiude le informazioni di protocollo che servono per implementare i protocolli di quel livello.

Per i dati rievuti si segue il cammino inverso, si guarda l'intestazione che era stata aggiunta dall'altro nodo a livello omologo

Ogni livello esegue una operazione di incapsulamento sui dati già incapsulati a livello precedente, al nodo opposto vengono de-capsulati per estrarre i dati.

- Header
  - Qualificazione del pacchetto dati per questo livello
- DATA
  - Payload proveniente dal livello superiore
- Trailer
  - generalmente usato in funzione di trattamento dell'errore (rivelazione, correzione)



La TPDU è l'insieme dei dati e header del livello di trasporto

### Stack TCP/IP

Il modello TCPI/IP è molto simile, ma su 5 livelli (senza sessione e presentazione)



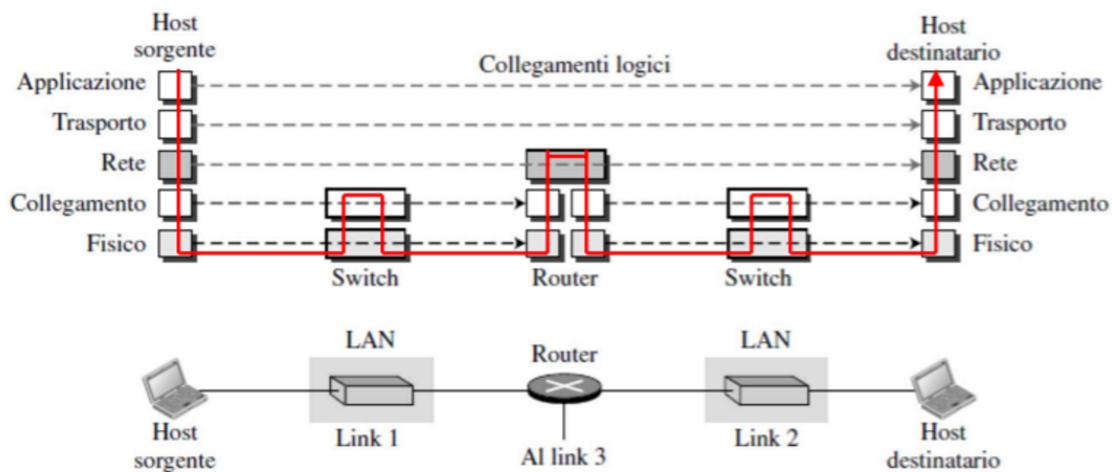
Applicazione: supporta le applicazioni di rete

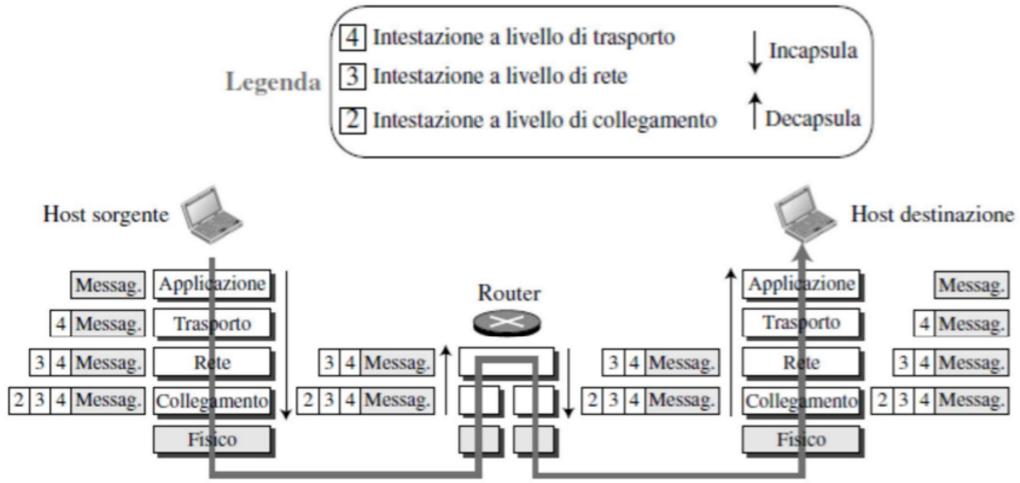
Trasporto: trasferimento dati end-to-end (tcp, udp)

Rete: instradamento dei dati tra host e destinazione (IP), protocolli di routing

Link: trasferimento dei dati in frame attraverso il collegamento che unisce degli elementi di rete vicini (ethernet, ppp, ... )

Fisico: trasferimento die bit tramite il mezzo trasmittivo





ISO/OSI usato come modello di riferimento, TCPI/IP standard de facto

### Livello Applicativo URI-HTTP

Abbiamo bisogno di far comunicare processi che stanno su host diversi, abbiamo bisogno di regole per farlo senza essere legati alle tecnologie specifiche dei nostri host

Due o più processi girano su ciascuno host e si scambiano messaggi.

Si ha un collegamento logico tra i due processi.

Definir i tipi di messagi scambiati allivello applicazione (richiesta risposta)

Sintassi dei vari tipi di messaggio

La semantica dei campi

Le regole per determinare quando e come un proesso invia o risponde ai messaggi

Ci sono due paradigmi fondamentali:

Client-Server: numero limitato di processi server che offrono un servizio e sono in esecuzione, in attesa di ricevere richieste

Client: programma che richiede un servizio

Peer to Peer: peer che possono offrire servizi e inviare richieste

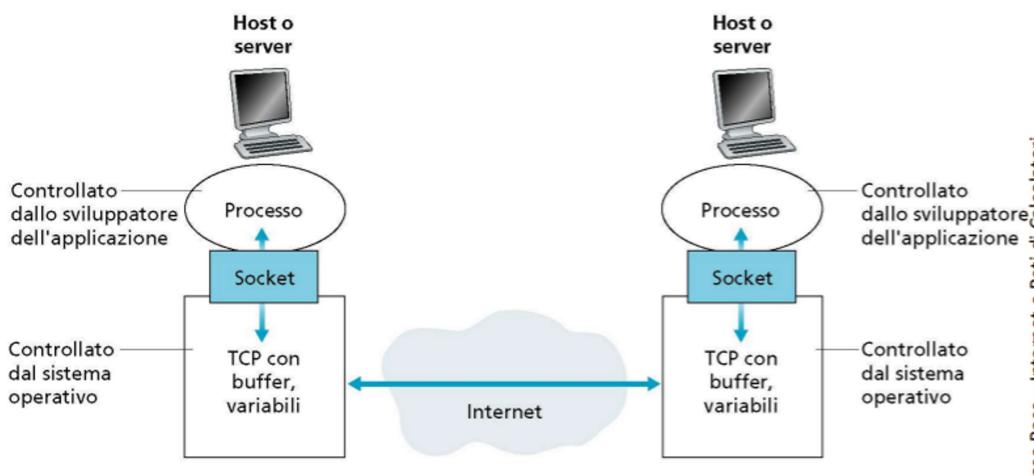
### Terminologia

API: application programming interface

Regole con il quale il livello applicativo si interfaccia con quello di trasporto, utilizzando i suoi servizi

Interfaccia socket:

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



Quello che fa il proesso quando impacchetta il messaggio, vuole che venga recapitato all'altro socket

Il client impacchetta il messaggio e ha un suo identificativo locale di socket, il destinatario sarà la socket corrispondente del processo destinatario.

È una struttura dati che viene utilizzata dal programma applicativo, endpoint della comunicazione.

Per ricevere messaggi, un processo deve avere un identificatore -> IP a 32 bit.

Per identificare il processo non è sufficiente avere solo l'indirizzo IP.

L'identificativo include l'indirizzo IP e il numero di porta associato al processo

I principali protocolli offerti da TCP/IP sono:

- UDP
- TCP

***Servizio TCP***

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

***Servizio UDP***

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

# Lez 5 - 27/09/2021

lunedì 27 settembre 2021 11:16

Requisiti più importanti

Throughput: tasso al quale il processo mittente può inviare i bit al processo ricevente, importante per i trasporti multimediali.

Ritardi: nella commutazione di pacchetto abbiamo tante componenti di ritardo, non costante. Lo stack TCP/IP non ci garantisce che i dati vengano consegnati (protocollo best effort)

Ci sono applicazioni che possono tollerare perdite di dati e alcune che non possono. Possono: streaming, telefonata, non possono: browser

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

Mappati i requisiti posso assegnare i protocolli

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

Che cos'è il WEB?

Ad oggi enorme collezione di informazioni nella quale le risorse sono distribuite e collegate l'una all'altra, tramite gli hyperlink, collegamenti. Qualsiasi server web nel mondo può aggiungere risorse.

Un sito web è un insieme di risorse web.

Pagina web formata da:

- File HTML di base
- Diversi oggetti referenziati (altre pagine, immagini, file)
- Ciascun oggetto è indirizzabile tramite **URL** (uniform resource locator)

## URI, URL E URN

URI (Uniform Resource Identifier)

Forma generale per identificare una risorsa. "a uniform resource identifier is a compact string of characters for identifying an abstract or physical resource"

Uniform: si ha una sintassi dell'identificatore

Resource: qualsiasi cosa abbia un'identità

Identifier: informazioni che permettono di distinguere un oggetto da un altro

Due tipi di URI: URL e URN

URL:

Sottoinsieme di uri che identifica le risorse attraverso un meccanismo di accesso

URN: uniform resource name

Identificativi che vogliono essere globalmente unici, persistente anche quando la risorsa non è più accessibile

**URL:**

**<schema>://<user>:<password>@<host>:<port>/<path>**

Schema: meccanismo di accesso a questa risorsa (https, http, ftp)

User e password: deprecato, nella sintassi della url c'è, opzionale

Host: nome di dominio o indirizzo IP

Porta: numero di porta del server, alcuni protocolli lo hanno di default (opzionale)

Path: percorso. Identifica, dentro il sito a una risorsa specifica con un certo identificativo, consiste in una sequenza di segmenti separati da '/'.

*HTTP URL*

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

: query: stringa di informazioni che deve essere interpretata dal server

Le url di distinguono in absolute e relative:

- Absolute: identifica una risorsa indipendentemente dal contesto in cui è usata
- Relative: informazioni che possono identificare una risorsa in base ad un'altra URL

Le url relative non viaggiano sulla rete, sono interpretate dal browser in base al documento di partenza. (file locali)(?)

Se cambi parte del percorso/host non lo trovi più :p

### **HTTP (Hypertext Transfer Protocol)**

Usato dal 1990 come protocollo di trasferimento per il WWW

Protocollo generico stateless che può essere usato per ipertesto, nome di server, oggetti distribuiti, estensibile tramite i suoi metodi.

Usato per tipizzazione e negoziazione dei formati di rappresentazione dei dati



### Chrome su S.O. Ubuntu

Il protocollo http è di tipo richiesta/risposta

Protocollo stateless:

Le coppie r/r sono indipendente, ogni richiesta viene eseguita indipendentemente da quelle che l'hanno preceduta

Usa il protocollo di trasporto TCP (connection-oriented)

Il client richiede l'instaurazione di una connessione TCP, i processi si scambiano messaggi attraverso le rispettive socket

Connessione non persistente

Viene stabilita una connessione tcp separata per recuperare ciascuna URL

Connessione persistente

A meno che non sia indicato il client assume che il server tenga la connessione aperta

Dopo la chiusura il client non deve più inviar richieste su quella connessione

### Pipelining

Server per migliorare ulteriormente le prestazioni

Consiste nell'invio da parte del client di molteplici richieste senza aspettare la ricezione di ciascuna risposta

Il server DEVE inviare le risposte nello stesso ordine in cui sono state ricevute le richieste

Il client non può inviare in pipeline richieste che usano metodi http non idem-potenti

Non è molto usato

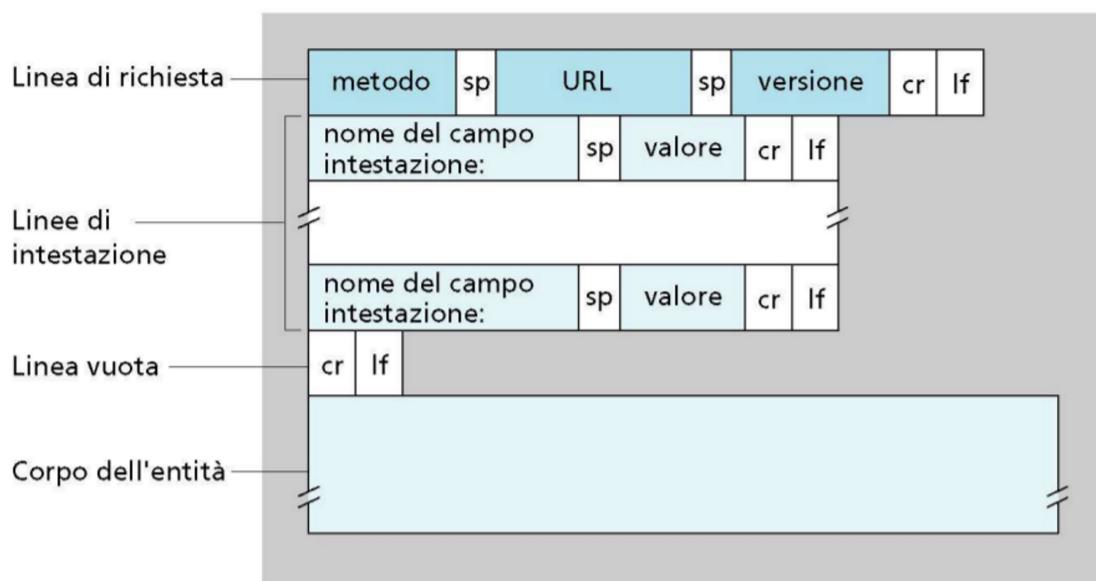
## Messaggi HTTP

Può essere di tipo **request** o **response**

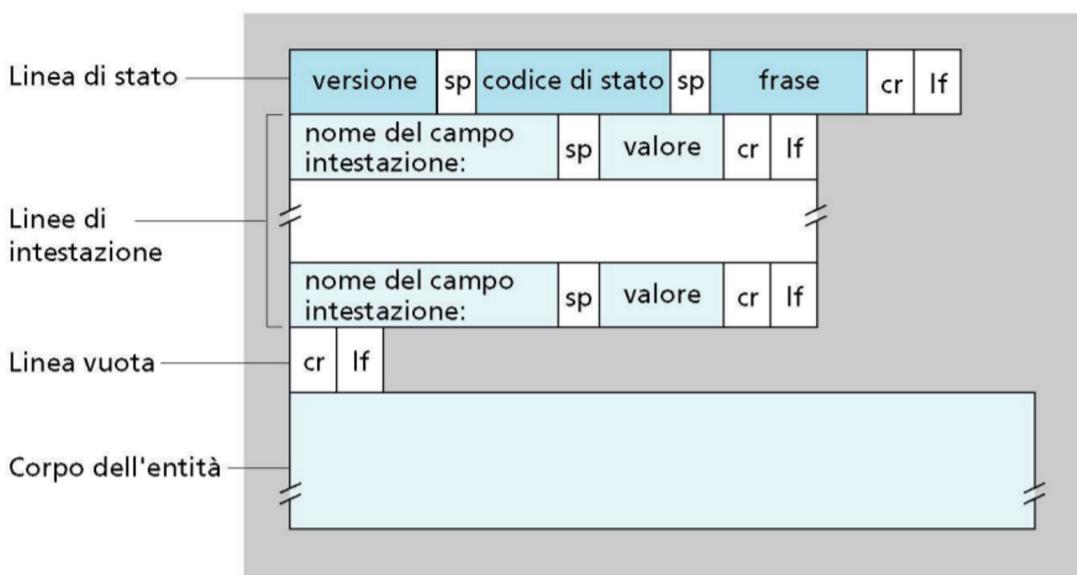
La riga iniziale distingue la richiesta dalla risposta (start - line)

Seguono una serie di HEADER e poi il corpo del messaggio (non sempre)

## HTTP Request message



## HTTP Response message



## Metodi principali del protocollo HTTP

### Header

Ci sono tante tipologie di header, anche i valori via via sono stati estesi.

### Insieme di coppie nome-valore

- General header: relativi alla trasmissione es data, codifica, connessione (connection close, connection keepAlive)
- Entità: contenuto del corpo del messaggio: tipo, lunghezza, data di scadenza...
- Response header: caratterizza il messaggio di risposta: informazioni sul server, autorizzazione richiesta...
- Request Header: intestazioni specifiche per la richiesta: tipo di software, chi sta facendo la richiesta, caratteristiche del client, di autorizzazione.

Chunked: modo per far mandare ai server degli aggiornamenti successivi, mantenersi la connessione aperta per mandare un po' di info ora e un po' modo, escamotage per tenere aperta per un po' la connessione.

### General Headers

```
Date: Tue, 15 Nov 1994 08:12:31 GMT  
Connection: close  
Transfer-Encoding: chunked
```

### Request Headers

```
Accept: text/plain; q=0.5, text/html,  
        image/png, application/json  
Accept-Charset: iso-8859-5,  
                unicode-1-1;q=0.8  
Accept-Encoding: compress, gzip
```

'q' indica una preferenza default: 1

Accept charset: set di caratteri accettabile per la risposta

Accept encoding: tipo di codifica accettabile per la risposta

## Status Line

```
Status-Line  
HTTP-Version SP  
Status-Code SP  
Reason-Phrase CRLF
```

Esempio: HTTP/1.1 200 OK

Status line: prima riga di un messaggio di risposta

Status code: intero a 3 cifre che indica il risultato dell'operazione richiesta

Reasonphrase: descrizione testuale dello status code

Tipi di status code:

```

1xx: Informational - Request received, continuing process
2xx: Success - The action was successfully received,
understood, and accepted
3xx: Redirection - Further action must be taken in order
to complete the request
4xx: Client Error - The request contains bad syntax or
cannot be fulfilled
5xx: Server Error - The server failed to fulfill an
apparently valid request

```

"100" - Continue	"101" - Switching Protocols
"200" - OK	"201" - Created
"202" - Accepted	"203" - Non-Authoritative Information
"204" - No Content	"205" - Reset Content
"206" - Partial Content	
"300" - Multiple Choices	"301" - Moved Permanently
"302" - Moved Temporarily	"303" - See Other
"304" - Not Modified	"305" - Use Proxy
"400" - Bad Request	"401" - Unauthorized
"402" - Payment Required	"403" - Forbidden
"404" - Not Found	"405" - Method Not Allowed
"406" - Not Acceptable	"407" - Proxy Authentication Required
"408" - Request Time-out	"409" - Conflict
"410" - Gone	"411" - Length Required
"412" - Precondition Failed	"413" - Request Entity Too Large
"414" - Request-URI Too Large	"415" - Unsupported Media Type
"500" - Internal Server Error	"501" - Not Implemented
"502" - Bad Gateway	"503" - Service Unavailable
"504" - Gateway Time-out	"505" - HTTP Version not supported

<http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html>

101 - switching protocols: succede quando per eseguire una richiesta devo cambiare il protocollo  
 201 - con i metodi http possiamo chiedere al server di creare una nuova risorsa, se l'operazione va a buon fine il server risponde con un 201. usato nella API REST (usare http per far parlare due programmi)

### Response headers

Informazioni che ci danno informazioni sulla risposta. Ad es programmi e librerie usate dal server per la risposta

**Location:** usato per ridirezionare il ricevente a una URI differente dalla request URI per completare la richiesta o accedere alla risorsa

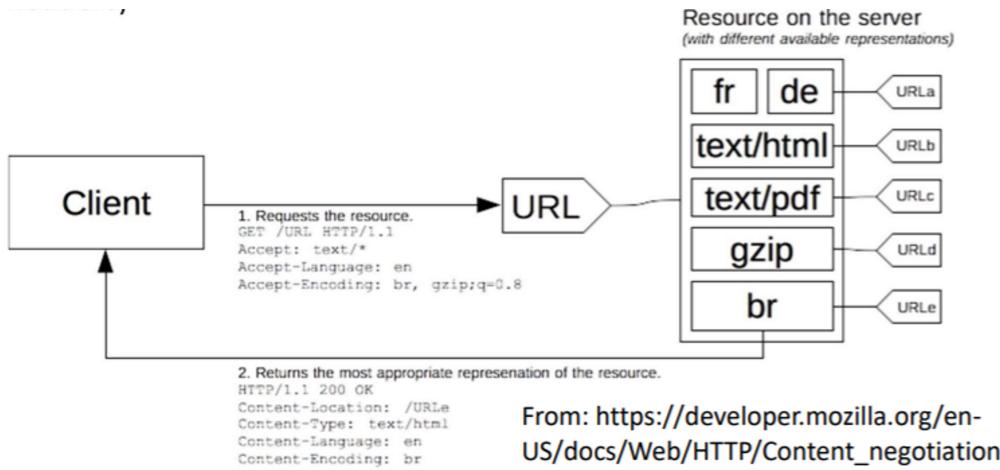
### Entity headers

```

Content-Base
  URI assoluta da usare per risolvere le URL relative
  contenute nell'entity body
Content-Encoding
  codifica dell'entity body (es: gzip)
Content-Language
  lingua dell'entity body (es: en, it)
Content-Type
  tipo dell'entity body (es: text/html)
Expires
  (utile per caching) validità temporale del contenuto
Last-Modified
  (utile per caching) data dell'ultima modifica sul server

```

### Content negotiation



Il client invia una prima richiesta al server, il server può rappresentare la risposta in vari modi (lingue, formati), ha pronte le risorse in varie *rappresentazioni* il client dice che accetta text, en, br e gzip con preferenza minore. Risponde con URL-e, gli rimanda il contenuto e descrive cosa ha mandato.

Content negotiation: meccanismo per selezionare la rappresentazione appropriata quando viene servita una richiesta

### Options:

Option è una request URI e il server risponde quali operazioni sono disponibili su quella request URI

```

OPTIONS http://192.168.11.66/manual/index.html
        HTTP/1.1
host: 192.168.11.66
Connection: close

HTTP/1.1 200 OK
Date: Sun, 14 May 2000 19:52:12 GMT
Server: Apache/1.3.9 (Unix)  (Red Hat/Linux)
Content-Length: 0
Allow: GET, HEAD, OPTIONS, TRACE
Connection: close

```

### GET - REQUEST

Metodo che richiede il trasferimento di una risorsa identificata da una URL o operazioni associate all'URL stessa

Sono possibili conditional get (header "if-...")

Header: if-Modified-Since, If-Unmodified-Since (date) . . .

Partial get: Header:Range

### GET - RESPONSE

```

HTTP/1.1 200 OK
Date: Sun, 14 May 2000 19:57:13 GMT
Server: Apache/1.3.9 (Unix) (Red Hat/Linux)
Last-Modified: Tue, 21 Sep 1999 14:46:36 GMT
ETag: "f2fc-799-37e79a4c"
Accept-Ranges: bytes
Content-Length: 1945
Connection: close
Content-Type: text/html

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2
Final//EN">
<HTML> ...

```

Get condizionale:

```

GET http://192.168.11.66 HTTP/1.1
Host: 192.168.11.66
If-Modified-Since: Tue, 21 Sep 1999 14:46:36 GMT

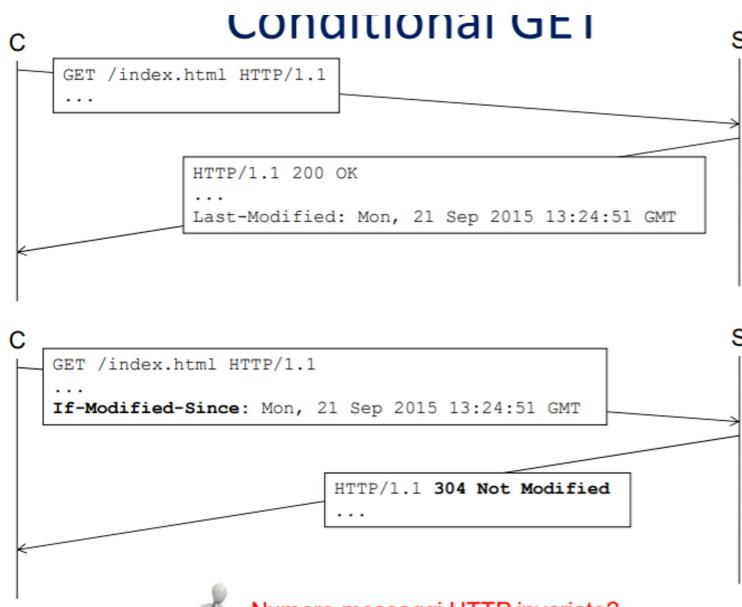
```

```

HTTP/1.1 304 Not Modified
Date: Wed, 22 Sep 1999 15:06:36 GMT
Server: Apache/1.3.9 (Unix) (Red Hat/Linux)

```

Vuole la risorsa se è stata modificata dal giorno indicato, il server risponde con codice 304 - not modified



Mando comunque la risposta col messaggio HTTP, ma non mandando la risorsa risparmio.

Request method - HEAD

```

HEAD http://192.168.11.66 HTTP/1.1
host: 192.168.11.66
Connection: close

```

```
HTTP/1.1 200 OK
Date: Sun, 14 May 2000 20:02:41 GMT
Server: Apache/1.3.9 (Unix) (Red Hat/Linux)
Last-Modified: Tue, 21 Sep 1999 14:46:36 GMT
ETag: "f2fc-799-37e79a4c"
Accept-Ranges: bytes
Content-Length: 1945
Connection: close
Content-Type: text/html
```

Fa una richiesta analoga alla get, ma in risposta voglio solo l'intestazione a quella che sarebbe stata la risposta della get non voglio il corpo. Utile per controllare lo stato dei documenti

#### Request method - POST

Con POST il client può mandare nel corpo del messaggio informazioni al server (compilare un form, postare un messaggio, scrittura su database)

#### Request method - PUT & DELETE

Con la **put** il client può chiedere di modificare una risorsa, specifica nella request URI l'identificativo della risorsa. (recupero una risorsa con GET)

Con la **delete** il client chiede di cancellare una risorsa identificata dalla request URI

#### Metodi safe e idempotenti

Metodi che non hanno effetti collaterali, non vanno a modificare la risorsa: GET, HEAD, OPTIONS, TRACE

I metodi idempotenti possono modificare la risorsa, ma se fai la richiesta più volte l'effetto non cambia. Se fai una put la risorsa viene creata, se la rifai uguale non succede nulla: Get, HEAD, PUT, DELETE, OPTIONS, TRACE

#### Pipeline

La post non è un metodo idempotente

#### Web caching

Soddisfare il cliente senza contattare il server

Memorizzare copie temporanee di risorse Web e riservarle al client per ridurre l'uso di risorse e diminuire il tempo di risposta

User Agent Cache: il browser mantiene una copia delle risorse visitate dall'utente

Proxy Cache: dispositivo intermedio, intercetta il traffico e mette in cache le risposte. La richiesta quando ho un proxy nel mezzo viene prima presa dal proxy che controlla se ha quella risorsa, altrimenti la chiede al server, che la manda al proxy che la inoltra e si mantiene una copia.

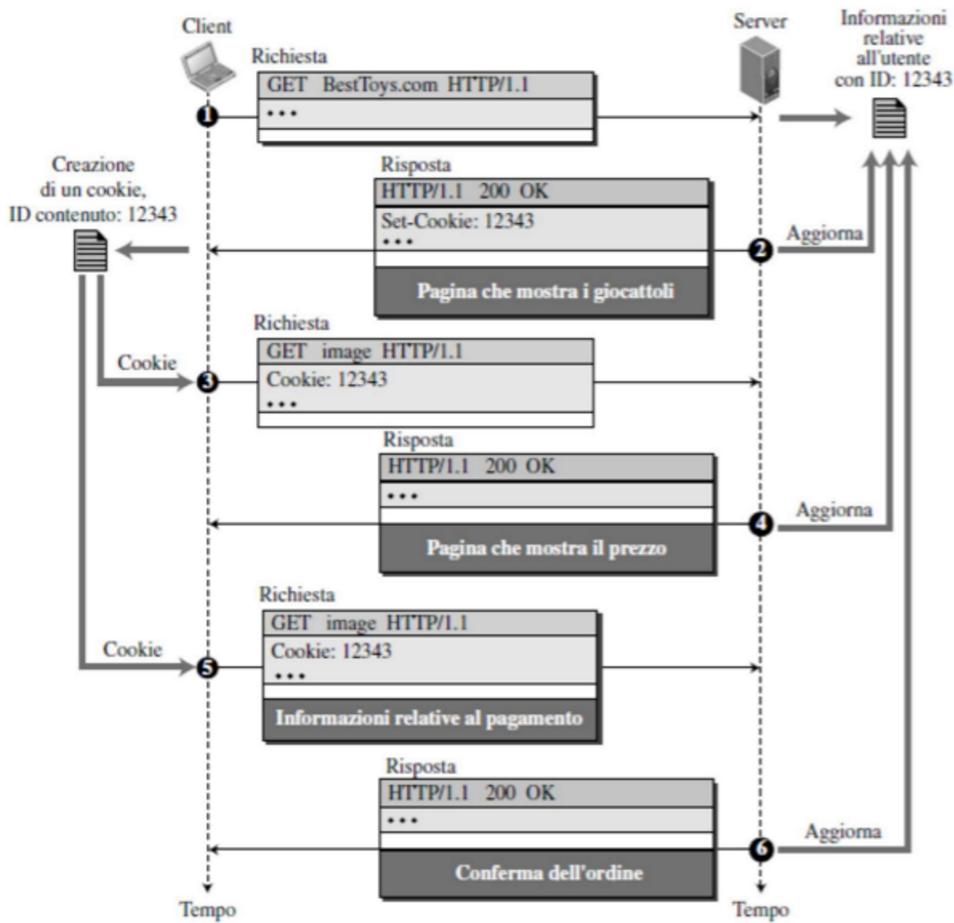
L'uso del proxy non è trasparente, non inoltri te la richiesta al proxy. Il server origine non sa chi a fatto la richiesta e risponde al proxy.

#### I cookie

Http è stateless

Per poter gestire il concetto di sessione e risorse che possono cambiare stato con l'interazione, su un protocollo stateless si possono usare vari meccanismi, il principale sono i cookie.

Servono per dare al server delle informazioni sull'identità dell'utente, sue preferenze e eventuali azioni che lui può aver fatto in passato. Non ho le informazioni di ip e porta che osso associare a un certo utente perché mi collego magari da dispositivi e reti diverse.



1. Il client fa una richiesta al server
2. Il server vede la richiesta, crea un identificativo e chiede al browser di memorizzarsi un cookie con questa informazione
3. Da questo punto in poi nella navigazione il client manderà nella richiesta oltre alle varie informazioni anche il cookie
4. Il server a questo punto sa come interpretarlo, con la richiesta e l'identificativo può aggiornarsi delle informazioni della sessione o della risorsa che sta gestendo, mantenendo il protocollo stateless.

Ci sono cookie di vario tipo:

Alcuni avvantaggiano l'utente nell'uso di servizi di vario tipo e altre di tracciamento dell'utilizzo dell'utente.

Vengono ancora molto usati in modo più normato. (accettazione dei cookie)

Per vedere i messaggi http chrome -> strumenti -> altri strumenti -> strumenti per sviluppatori

## TELNET

Protocollo per l'accesso remoto a delle macchine, servizio di terminale remoto su reti TCP/IP  
ancestor di SSH

Sta per terminal network.

È un protocollo per interazione client server di login remoto generale.

Mi permette, da una macchina di accedere a un'altra che può avere caratteristiche diverse, utilizzo dei comandi che vengono interpretati dall'altra parte.

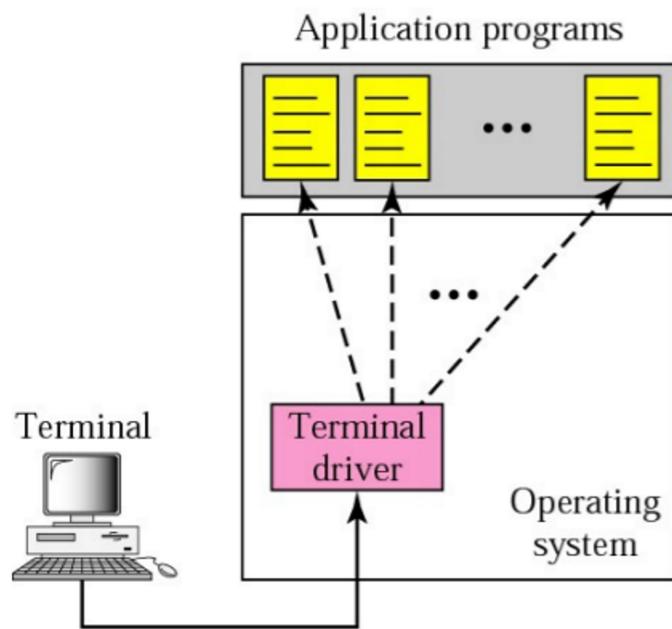
Telnet permette a un utente su una macchina di stabilire una connessione con un login remoto  
I comandi inseriti sulla macchina locale vengono inviate sulla macchina remota, quella produrrà un output che verrà mostrato sulla macchina locale

## Modello di telnet

Programma server che accetta le richieste  
Programma client che effettua le richieste

Come standard il client con telnet si connette alla porta 23 tramite connessione TCP

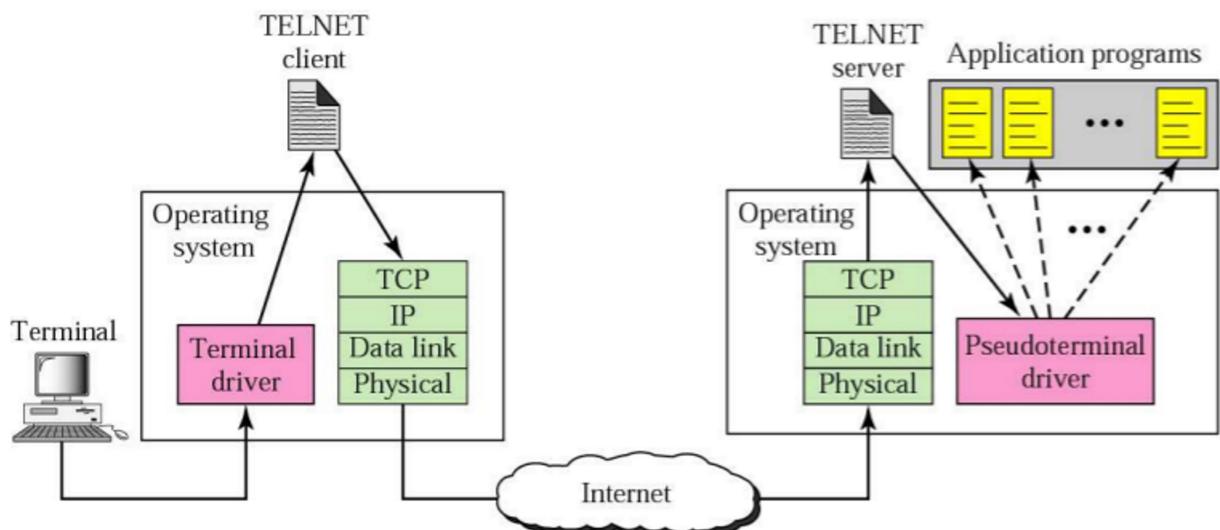
Come funziona in locale



Terminal, driver di terminale

Lato client ho il terminale dell'utente che manda il comando che viene acquisito tramite il driver di terminale e tramite il programma telnet client incapsula il messaggio e lo trasmette.

Sulla macchina in remoto il messaggio viene decapsulato e mandato al telnet server che li manda al pseudoterminal driver che interagisce con le applicazioni



Pseudoterminal driver: entry point del so che consente di trasferire caratteri a un processo come se provenissero dal terminale. Accetta i caratteri dal server telnet e li trasmette al so che li trasmette all'opportuna app

# Telnet: NVT

**Problema:** TELNET deve poter operare con il numero massimo di sistemi e quindi gestire dettagli di sistemi operativi eterogenei.

I terminali possono differire gli uni dagli altri per:

- il set e codifica di caratteri
- la larghezza della linea e lunghezza della pagina
- i tasti funzione individuati da diverse sequenze di caratteri (escape sequence)  
es. diversa combinazione di tasti per interrompere un processo (CTRL-C, ESC), caratteri ASCII diversi per terminazione righe di testo

Protocolli per implementare servizio di mail

Trasferimento di un messaggio tra un mittente e un destinatario

Servizio asincrono.

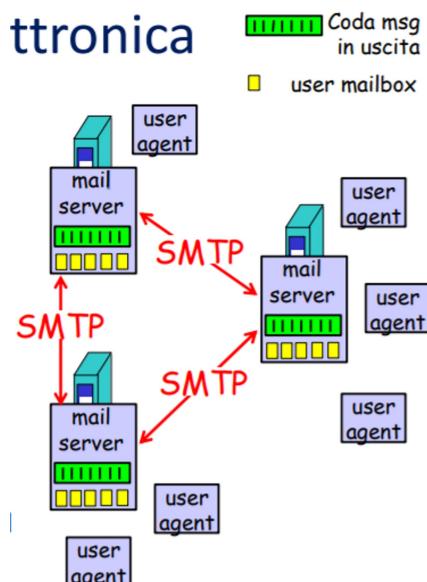
Offre delle garanzie sulla consegna dei messaggi. Messaggi di errore se un messaggio non viene consegnato

## Componenti per realizzare il servizio:

- User agent: componenti che permettono all'utente di comporre un messaggio, richiedere l'invio, riceverlo e visualizzarlo (outlook, gmail )
- Intermediari: Mail Server  
Questi mail server comprendono aree in cui vengono archiviati i messaggi in entrata (messaggi che un server ha ricevuto e che deve trasferire a altri mail server) e mailbox, (cassetta di posta), coda di messaggi in uscita

Protocollo SMTP, protocollo con cui i mail server dialogano per gestire il trasferimento di mail

## ttronica



5

Come fa un mail server a sapere dove recapitare la mail?

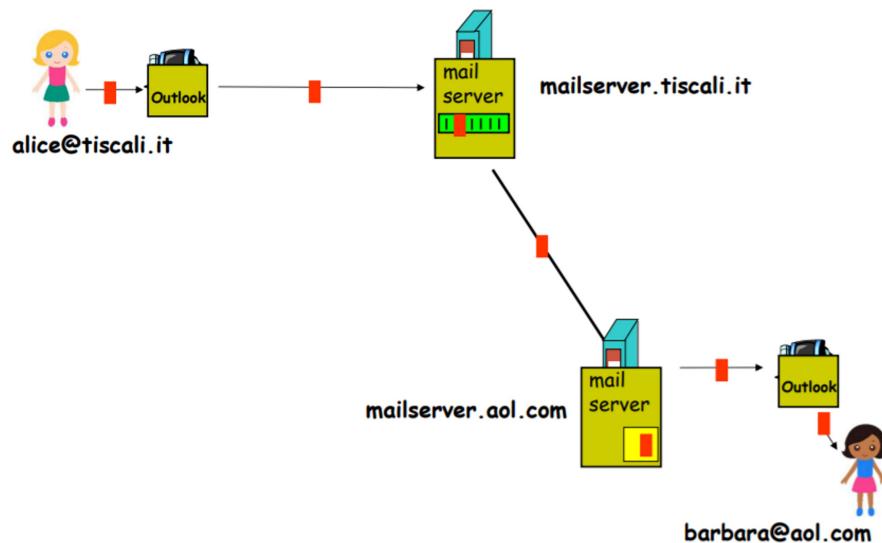
E quello di destinazione a quale casella di posta=

**Indirizzo email**

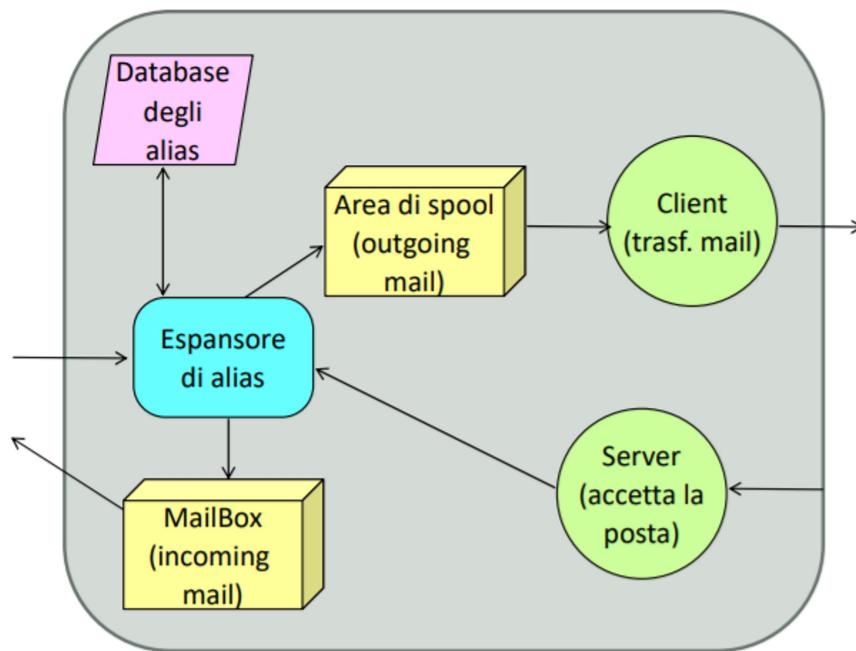
**local-part @ domain-name**

Domain-name: mail server destinazione

Local-part: cassetta di posta nel mail server



Come funziona all'interno un mail server



Il mail server è in ascolto di messaggi che gli vengono inviati da user agent o altri mail server, le mail se non sono dirette ai proprio utenti devono essere trasferiti da altri utenti quindi vengono messi in **area di spool (outgoing mail)** quindi il mail server per ciascuna mail guarda il nome di dominio e come un client chiede una connessione tcp e trasferisce al nuovo mail server, si tiene una copia finché il trasferimento non va a buon fine (o fino a scadenza messaggio).

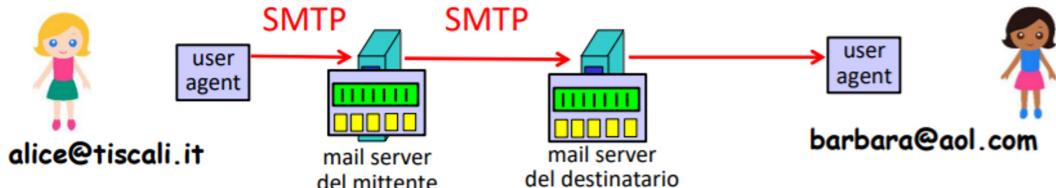
Se invece la mail che riceve è per uno dei propri utenti lo mette in **mailBox**.

Alias:

Non necessariamente c'è una corrispondenza 1 a 1 tra utente e indirizzo di mail.

Magari indirizzo `direttore@unipi.it` corrisponde a `mario.rossi@unipi.it` e quindi con le solite credenziali riceve anche le mail di direttore

Le stesse cose possono essere fatte anche con nome di dominio diversi.



- Servizio di trasferimento posta:
  - Intermediari, no consegna diretta
- **SMTP [RFC 2821]**: distribuisce/archivia nel mail server del destinatario
  - **Protocollo di tipo «push»**

Protocollo push:

Il client riceve una mail la deve trasferire e richiede una connessione tcp per delegarla all'altro  
Arriva mail -> attiva connessione -> trasferisce

### SMTP

L'obiettivo di SMTP è un trasferimento affidabile e efficiente di mail. Protocollo applicativo che usa TCP. Mail affidabile -> servizio di trasporto affidabile

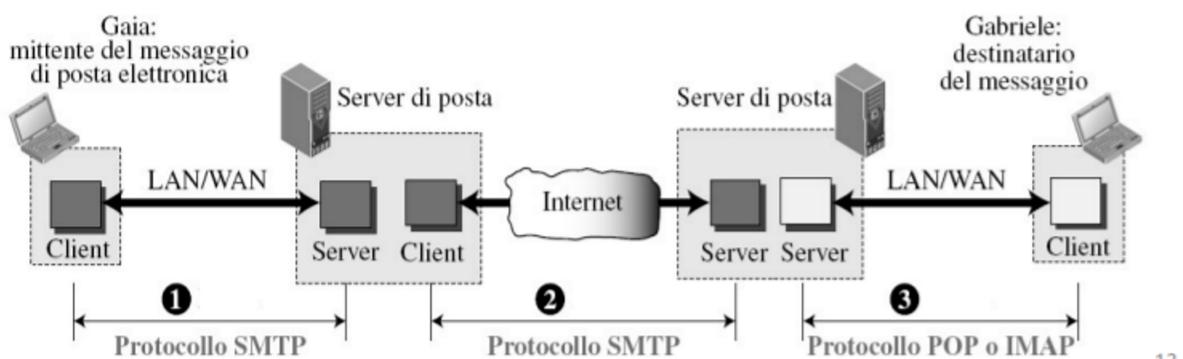
SMTP richiede solo un trasferimento di dati ordinato e affidabile.

Porta mail attraverso intermediari tra mittente e destinatario

**Quando un client SMTP vuole trasferire un messaggio, stabilisce un canale di trasmissione bidirezionale con un server SMTP. La responsabilità di un client è di trasferire la mail a un server SMTP, o comunicare un eventuale insuccesso (scambio formale di responsabilità).**

Un client SMTP determina l'indirizzo di un host appropriato che ospita un server SMTP risolvendo il nome della destinazione in un mail server destinazione

- risoluzione di nome in indirizzo IP attraverso il DNS).





.. non ho ricevuto il tuo messaggio

Possibili problemi:

- connessione con mailserver del mittente (server inesistente o irraggiungibile)
- connessione con mailserver destinatario (server inesistente o irraggiungibile)
- inserimento in mailbox destinatario (user unknown, mailbox full)  
... ma in tutti questi i casi **il mittente riceve una notifica!**

Destinatario può non ricevere (senza che mittente sia avvisato)  
solo se qualcuno (intruso, filtro antispam) rimuove messaggio



Come funziona il trasferimento di mail

Si usa protocollo TCP porta di default 25.

Fasi di trasmissione:

- Handshaking:  
client e server si "presentano"
- Trasferimento del messaggio
- Chiusura della connessione

L'interazione avviene con uno scambio di comandi/risposte ASCII



HO BISOGNO DI UN EDINTIFICATIVO A QUEL LIVELLO DI ASTRAZIONE  
E faccio un associazione poi tra il nome del livello applicativo e l'ip della macchina  
IP: usato per instradare datagrammi inrete  
Il DNS ha come obiettivo trovare associazione tra nome a livello applicativo e l'indirizzo ip  
Quando internet era "piccolo" era facile trovare associazione, tutti i nomi logici e i relativi IP erano contenuti in un file  
Periodicamente tutti gli host prelevavano una versione aggiornata del file

Adesso non è fattibile così

Si ha quindi un sistema DNS<sup>domain name system</sup> con regole e organizzazione gerarchi con una delega delle autorità

DNS è posizionato a livello applicativo -> gira sui sistemi terminali  
Si basa sul paradigma client-server e usa i servizi del livello di trasporto per mandare le richieste di risoluzione dei nomi e le risposte (associazione)  
Non interagisce con gli utenti (a meno di messaggi di errore nella risoluzione)

I servizi complessi vengono messi alle estremità della rete

1. Scrivi un url nel browser
2. Il browser chiede al DNS l'indirizzo ip di quel nome
3. Il DNS manda la traduzione
4. Con la traduzione potrà chiedere al livello di trasporto di stabilire la connessione
5. Dalla connessione farà la sua richiesta e riceverà la risposta

Quindi DNS

- Specifica sintassi dei nomi e modo per gestirli
  - Consente la conversione da nomi a indirizzi e viceversa
1. Schema di assegnazione dei nomi gerarchi basato su domini
  2. Database distribuito contenenti i nomi e i corrispondenti indirizzi con gerarchia di name server
  3. Protocollo per la distribuzione delle informazioni sui nomi tra nome e server
    - a. Gost, router, name server comunicano per **risolvere** i nomi
    - b. Utilizzando udp (porta 53) oppure TCP

### Servizi offerti dal DNS

1. Risolvere nomi in indirizzi IP
2. Mappare il fatto che un host possa avere più nomi (**host aliasing**) ([www.enterprise.com](http://www.enterprise.com) / enterprise.com)
3. Distribuzione carico: associare al nome canonico di una macchina, più indirizzi IP per bilanciare il carico

### Struttura gerarchica

Un nome è costituito da diverse parti

- Nome organizzazione, dipartimento, ufficio ecc.

Assegnazione del nome delegabile -> sistema (in buona parte) decentralizzato

Delega dell'autorità per l'assegnazione delle varie parti dello spazio dei nomi

Distribuzione responsabilità della conversione tra nomi e indirizzi

### Nomi di dominio

Spazio dei nomi -> struttura gerarchica, rappresentabile con un albero

Ogni nodo dell'albero ha un nome di dominio,

Top level: *.it*

Secondo livello: *unipi.it*

Terzo livello: *di.unipi.it*

**Dominio: sottospazione del ..**

Internet, spazio dei nomi in internet può essere suddiviso in centinaia di domini con i loro sottodomini

I nomi gerarchici delle macchine sono assegnati in base alla struttura delle organizzazioni che

ottengono l'autorità per porzioni nello spazio dei nomi

La struttura gerarchica permette autonomia nella scelta dei nomi all'interno del dominio

Es: server1.di.unipi.it != da server1.cs.cornell.edu

Top level domain

.com .edu .Mil .gov .net .org e codici geografici (.it .Uk .Fr ..)

Mantenuti da IANA (internet assigned numbers authority)

### **Gerarchia dei name server**

Alla gerarchia dei domain name corrisponde la gerarchia dei name server

Radice: risoluzione top level

Sempre a albero poi gli altri

Grazie alla gerarchia dei name server io posso distribuire le info sui domini su più name server

Zona: regione di cui è responsabile un certo name server (non necessariamente coincide col dominio)

Il server immagazzina le informazioni relative alla propria zona, inclusi i riferimenti ai server di domini di livello inferiore

### **Server radice**

Responsabili delle informazioni di top-level domain

Centinaia di root name server in tutto il mondo

### **Server top level domain**

Mantengono le informazioni di quel livello più restituiscono le info sui name server di competenza dei sottodomini

### **Server di competenza**

Autorità per una certa zona

Memorizzare nome e ip di un insieme di host

Può effettuare traduzioni per quegli host

Per una certa zona ci possono essere server di competenza primari e secondari

P: file di zona

S: ricevono i file di zona e li offrono al servizio di traduzione

### **Local name server**

Non appartiene strettamente alla gerarchia dei server

Ogni ISP ha il suo

Le query DNS vengono prima rivolte al name server locale

Quando un programma deve trasformare un nome in ip chiama un programma detto resolver passando il nome come parametro di ingresso

Il resolver interroga il local name server che cerca il nome nelle sue tabelle e restituisce.....

# Lez 9

mercoledì 13 ottobre 2021 11:09

### Lo strato di trasporto

Realizza una comunicazione logica fra processi residenti in host system diversi

- Logico: i processi si comportano come se gli host fossero direttamente collegati

Il compito dello strato di trasporto è quello di fornire servizi a quello di applicazione

L'applicazione sceglie lo stile di trasporto:

- Sequenza di messaggi singoli UDP
- Stream di byte TCP

Il livello di trasporto utilizza servizi del livello di rete

È un servizio fondamentale, da un'idea di comunicazione diretta tra gli host, è implementato solo sui sistemi terminali

Prende i dati, li formatta in segmenti o user datagram e li passa al livello di rete

- **Servizio privo di connessione**

il livello di trasporto si occupa di mandare **messaggi** (al liv trasp destinatario), ogni messaggio viene incapsulato in un datagram e non è correlato coni messaggi prec e succ, non ha garanzia di consegna, se perdo un messaggio non lo recupero. I messaggi possono non arrivare in ordine, possono non fare lo stesso percorso. Il liv trasp li manda al livello applicativo così come gli arrivano

- **Servizio orientato alla connessione**

stabilisce una connessione logica tra client e server

- **Servizi offerti:**

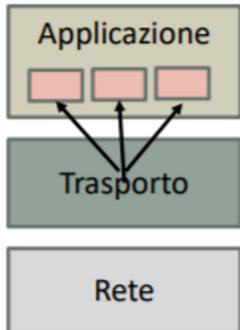
- Multiplexing/demultiplexing
  - Controllo degli errori (header + dati)
- **Il Protocollo TCP [RFC 793]**
  - Gestione della connessione
  - Consegnà affidabile (priva di errori, completezza e ordine)
  - Flow Control
  - Controllo di congestione

- **Il Protocollo UDP [RFC 768]**

- Senza connessione
- Non affidabile, consegne senza ordine
- Estensione “senza fronzoli” del servizio di consegna “host to host” di IP

### Demultiplexing

Lo stato di trasporto provvede allo smistamento dei pacchetti tra la rete e le applicazioni



### Multiplexing

Accorpamento dei flussi dati dai processi alla rete. "imbusta" i dati ricevuti con un preambolo

Le operazioni di multi e deplexing si basano sui socket address dei processi

Socket address: indirizzo ip e numero di porta

#### Demultiplexing

nell'host ricevente:

consegnare i segmenti ricevuti  
alla socket appropriata

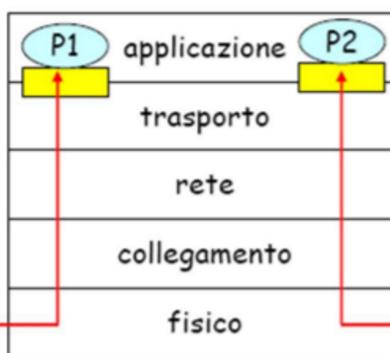
#### Multiplexing

nell'host mittente:

raccogliere i dati da varie  
socket, incapsularli con  
l'intestazione (utilizzati poi  
per il demultiplexing)

= socket

= processo



host 1

host 2

host 3

Copyright 1996-2005 J.F Kurose and K.W. Ross

### Porta:

La porta è un numero da 16 bit.

L'indirizzo IP è da 32 bit.

Il range di porte è organizzato con determinate regole:

- *System ports*: (Well Known Ports) 0 - 1023 porte assegnate dallo IANA a determinati servizi (es porta HTTP 80) identificano processi server
- *User ports (registered ports)* 1024 - 49151 porte registrate per un servizio (es porta MySQL è in questo range) assegnate da IANA
- *Dynamic Ports (Private o Ephemeral Ports)* porte per l'indirizzo del mio client, non assegnate da IANA

Il SO assegna dinamicamente le porte ai processi che ne fanno richiesta

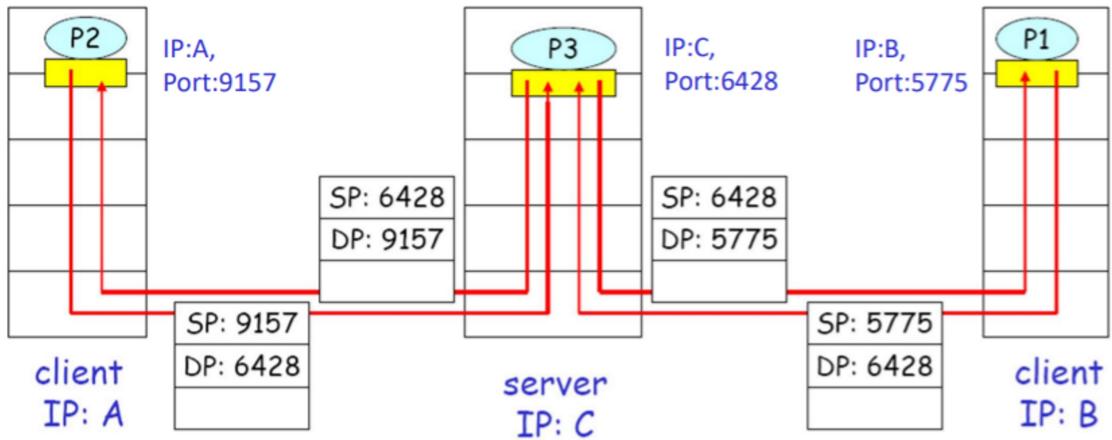
### Demultiplexing UDP

Creo una socket (DatagramSocket mySocket = new DatagramSocket())

Socket UDP identificata da coppia (IP, porta)

Lo stato si trasporta dell'host ricevente consegna il segmento UDP alla socket

I datagram con stesso IP e porta destinatario uguale vengono consegnati alla stessa porta  
(anche con mittente diverso)



### Demultiplexing orientato alla connessione (TCP)

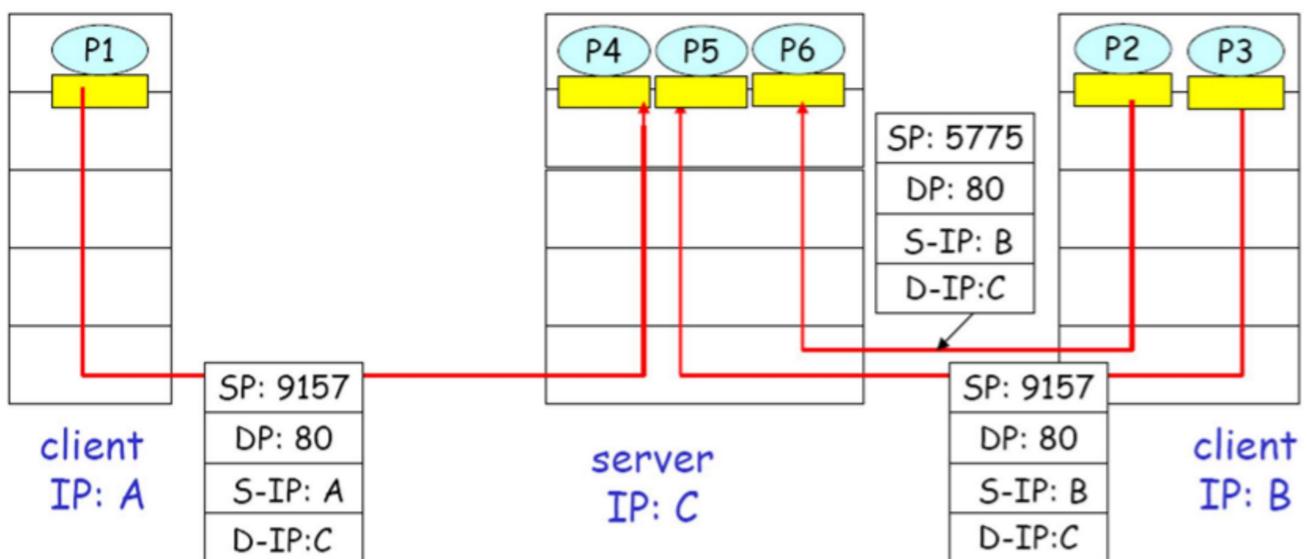
Due processi che comunicano con una connessione logica instaurata dal livello di trasporto

4 parametri:

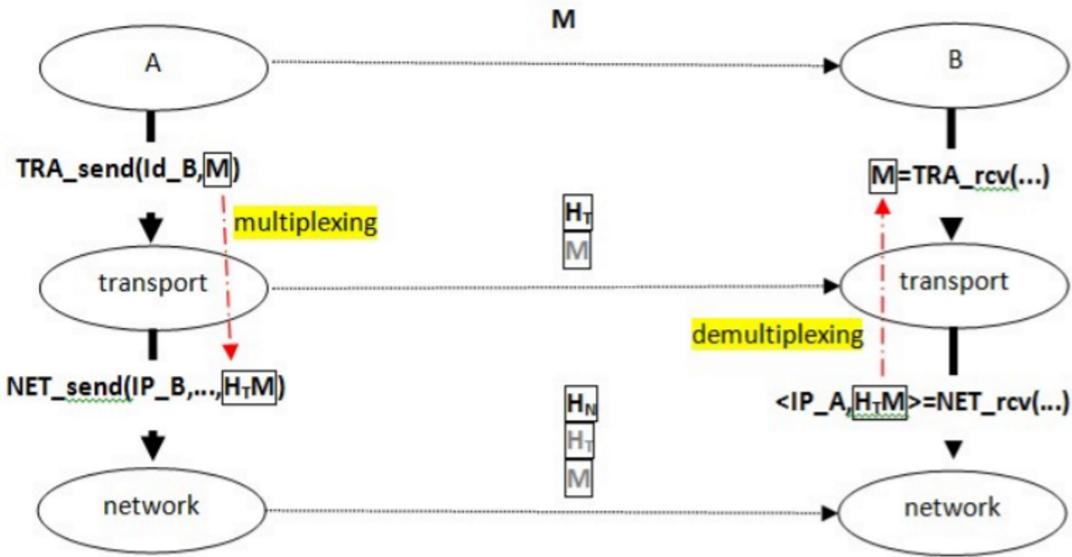
- IP origine
- Numero porta origine
- IP destinatario
- Numero porta destinatario

L'host usa i 4 parametri per inviare il segmento alla socket appropriata

Un host può supportare più socket contemporanee



# Interfaccia tra i livelli dello stack



Quello che fa il TCP è prendere un flusso di byte indefinita a priori e mandarlo al destinatario che lo manda al processo dest

### Flusso di byte ordinati ma non strutturati

Cosa c'è nel flusso di byte non interessa a TCP

Il processo applicativo poi mette in ordine i byte e ci opera riorganizza e raggruppa i byte.

I nodi intermedi non sono informati della connessione.

La connessione viene instaurata tramiti **handshake** e si preparano strutture dati per gestire la connessione

Il client è quello che per primo stabilisce la connessione, non è detto che sia lui che inizia a mandare i dati

### Connessione full duplex

I dati possono viaggiare in entrambe le direzioni, contemporaneamente

Connessione punto-punto, 1 a 1

Servizi del TCP:

- Flusso continuo di dati
- Trasferimento bidirezionale
- Multiplexing/demultiplexing

Garantisce un trasferimento dati ordinato e affidabile, offre un servizio di controllo degli errori (dati corrotti, segmenti persi, duplicati, fuori sequenza)

Meccanismi di inizio e fine trasmissione (*controllo di sessione*)

Controllo di flusso: non manda più dati di quelli su cui il processo dest riesca a operare

Controllo di congestione: ha lo scopo di recuperare situazioni di sovraccarico della rete

Per il trasferimento dati si utilizzano dei buffer, i byte che arrivano vengono messi in un buffer e quando ritiene opportuno invia i byte in rete (in un segmento), ottimizza il trasferimento



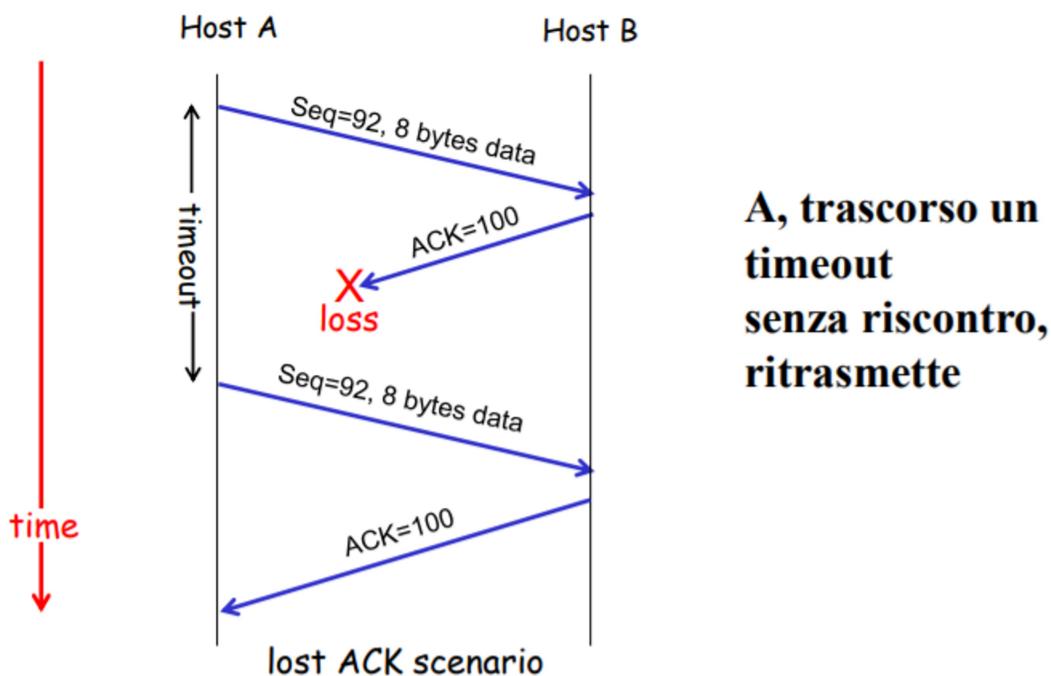
Timeout: in caso di timeout si ritrasmette il segmento che ha causato il timeout e si riavvia il timer  
 Ack duplicato: ricevo una volta un ack con numero di riscontro es. 500, lo ricevo altre 2 volte,  
 significa che ho perso il segmento successivo a quello riscontrato, senza aspettare che scada il timer,  
 eseguo una **ritrasmissione veloce** (fast retransmission)

Segmenti fuori sequenza:

I dati possono arrivare fuori sequenza ed essere temporaneamente memorizzati dall'entità TCP destinataria.

Il TCP non dice come il destinatario deve gestire i pacchetti fuori sequenza, dipende dall'implementazione

## Ritrasmissione dovuta a riscontro perso



### Lato destinatario

Se il destinatario ha dei dati da trasmettere può trasmettere dati e manda anche l'ACK di quello che ha ricevuto

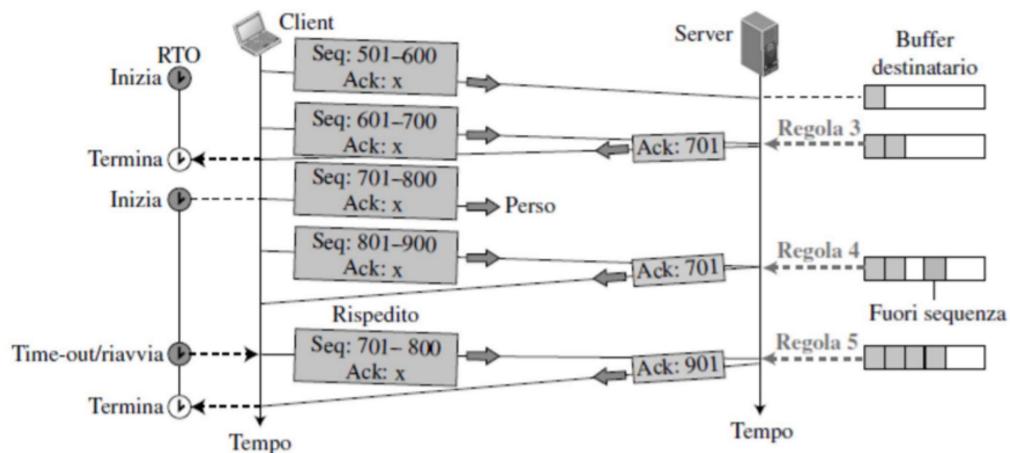
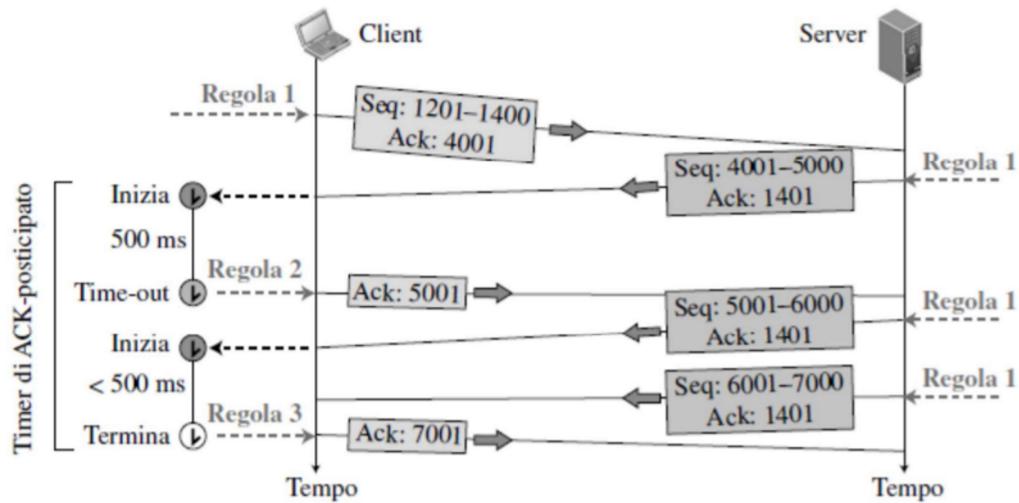
Se il destinatario deve riscontrare dei segmenti, ma non ha nulla da mandare e riceve segmenti in ordine, prima di mandare l'ack ritarda 500ms, se entro quel tempo arriva un altro segmento può poi mandare l'ack, così manda l'ack di più segmenti

Se il destinatario riceve un segmento atteso e quello precedente non è arrivato allora invia subito l'ack

Se dest riceve un segmento fuori sequenza oppure mancante o duplicato manda subito ack

# Operatività normale

(1) Tutti i segmenti contenenti dati includono ACK  
 (2) Se destinatario non ha dati da inviare e riceve segmento «in ordine» ritarda invio ACK di 500ms a meno che non riceva nuovo segmento  
 (3) Se destinatario riceve segmento atteso e precedente non è stato riscontrato allora invia immediatamente ACK  
 Se destinatario riceve (4) segmento fuori sequenza oppure (5) «mancante» oppure (6) duplicato allora invia immediatamente un segmento ACK (indicando prossimo numero atteso)



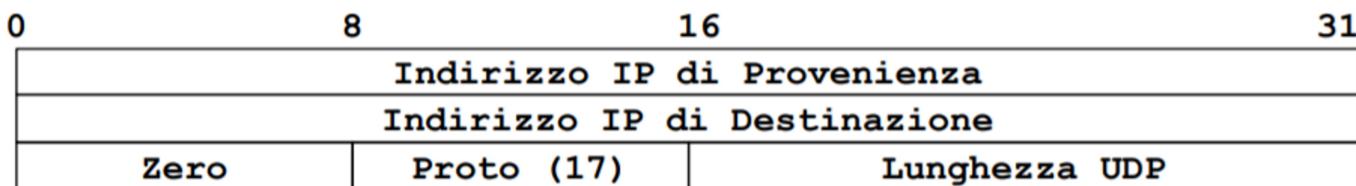
08/11/2021

lunedì 8 novembre 2021 11:23

#### Datagramma UDP: campi

- **Checksum:** checksum dell'intero datagramma, opzionale per il controllo errore end-to-end, il pacchetto corrotto viene scartato senza notifica la mittente

Pseudoheader – usato per calcolo checksum



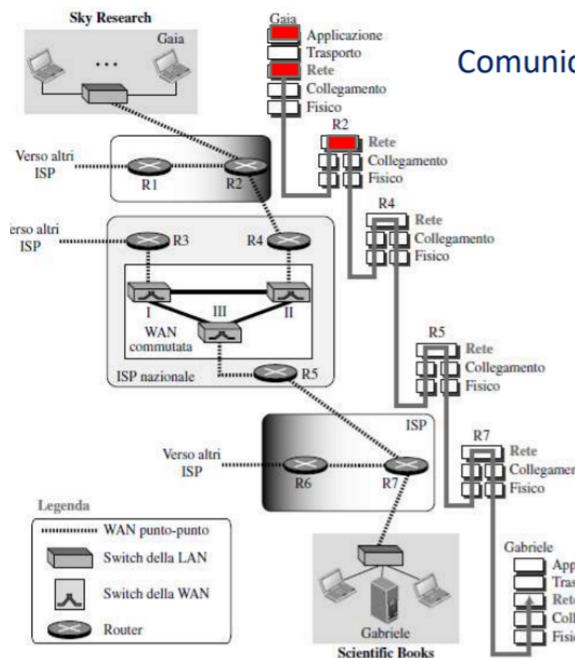
Il campo checksum della nostra intestazione viene messo a 0, si divide la nostra sequenza di bit in parole da 16 bit, si fa la somma delle parole in complemento a 1 e poi si dà il complemento a 1 del risultato di questa somma

Il mittente mette il risultato di queste operazioni nel campo checksum del datagramma UDP

Il ricevente riceve il segmento ne calcola la checksum (del segmento che include la checksum).

Se il risultato è 0 no sono stati rilevati errori, altrimenti vuol dire che c'è un qualche errore (non si va a rilevare DOVE è l'errore) il messaggio viene scartato

## Livello di rete



## Comunicazione a livello rete

1. L'entità a livello di rete riceve i segmenti dal livello di trasporto nell'host mittente, e incapsula i segmenti in datagrammi

2. I datagrammi sono inoltrati al prossimo nodo (host o router)

3. Il router esamina i campi intestazione in tutti i datagrammi IP che lo attraversano e li inoltra da un collegamento in ingresso ad un collegamento in uscita

4. Sul lato destinatario, consegna i segmenti al livello di trasporto (demux TCP o UDP)

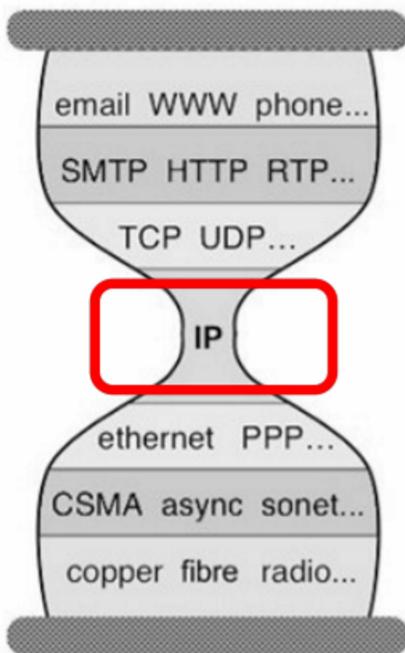
4

Il livello di rete è implementato nei livelli intermedi, mentre quello di trasporto è implementato nei punti terminali, il livello di rete implementa poche funzioni

*Il livello di rete interconnette reti eterogenee*

Al livello sottostante possiamo avere tecnologie di tipo diverso

Il modello a clessidra che rappresenta il piano dati



Abbiamo tecnologie eterogenee sia ai livelli superiori che quelli inferiori

Abbiamo poi un livello dirette relativamente semplice che sfrutta un solo protocollo(?)

In generale a livello di rete possiamo avere:

- Controllo errori
- Controllo flusso

- Controllo congestione
- Qualità del servizio (rispetto al modello originale sono stati aggiunti servizi per implementare politiche di qualità del servizio)
- Sicurezza

### IP RFC 791

Specifiche successive alle prime specifiche di protocolli connectionless

Il protocollo IP nasce come servizio best effort, non è affidabile, non prevede meccanismi di recupero degli errori, ci sono controlli di errore (send and pray)

Non prevede garanzie sulla QoS tempo di consegna e controllo di flusso, sono stati aggiunti dopo i meccanismi del genere

La funzionalità di base del protocollo IP è quello di consegnare i pacchetti **forwarding**

Siamo nel piano dati, abbiamo i dati utili (datagrammi) che arrivano al router, il router va a vedere l'intestazione, e in base all'info contenuta decide dove inoltrare il datagramma

Come decide?

Decide in base a una tabella di inoltro, in base all'indirizzo di destinazione si va a decidere l'interfaccia di inoltro (tutto fatto nel minor tempo possibile)

Come si costruisce la tabella di inoltro? **Funzione di instradamento** -> piano di controllo

**Intradamento:** processo decisionale di scelta del percorso verso una destinazione, i router implementano degli algoritmi di instradamento, con informazioni sulla rete circostante, costruzione di un suo modello della tipologia di rete e in base a questo costruire la tabella

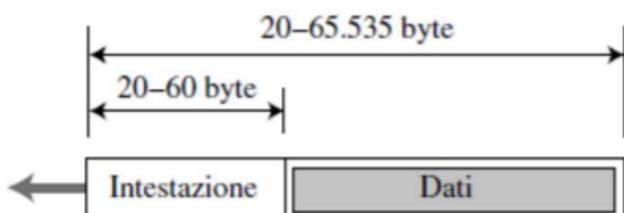
### Meccanismi IP

Schema di indirizzamento

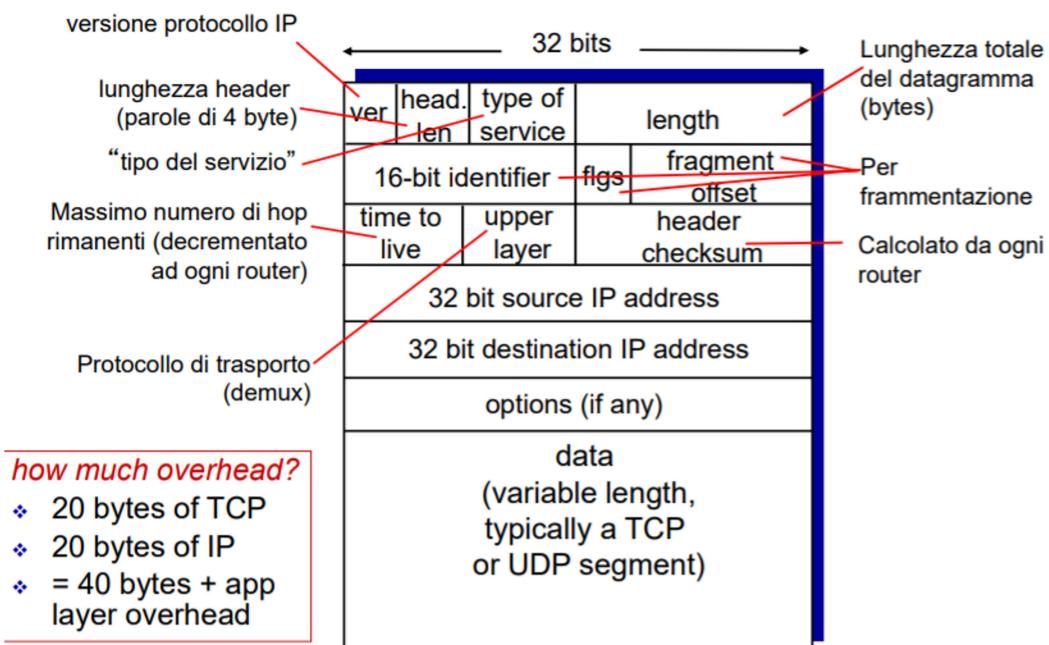
Modello del datagramma

Anche a livello di rete si fa multiplexing e demultiplexing

### Formato datagramma IP



# IP datagram format



Lunghezza totale del datagramma: max 16 bit

- **Versione: versione di IP usata 4b**
- **Hlen: lunghezza dell'header espresso in parole da 32 bit, tipicamente 0101 20 byte (4b)**
- **Tipo di servizio: 8b**
  - 6 bit per *Differentiated services*
    - I pacchetti vengono marcati in base alla classe di servizio (teleonata, controllo, multimedia streaming, dati a bassa priorità)
    - Politiche di accodamento router
  - 2 bit *Explicit Congestion Notification (ECN)*: supporto a livello di rete e trasporto per la notifica di eventi di congestione, analizzare gli eventi e sapere se ci sono eventi di congestione, lo fa da solo il livello di trasporto, con questi 2 bit i router collaborano nella notifica di congestione
- **Lunghezza del datagramma: 16 b lunghezza di tutti il datagramma in byte incluso header (max 65535 byte in pratica 1500)**
- **Identificatore, flag, offset: campi per la frammentazione**
- **TTL: 8bit viene decrementato a ogni passaggio da un router, quando arriva a 0 viene scartato, evita che i pacchetti vadano in giro all'infinito in percorsi ciclici, il valore iniziale dipende da SO**
- **Protocollo: 8bit indica quale protocollo dello stato superiore deve ricevere i dati**
- **Checksum dell'intestazione: viene calcolato il checksum della osla intestazione ad ogni router se si ottiene un errore si scarta il datagram**
- **Indirizzi sorgente e destinazione: 32bit**
- **Opzioni: multiplo di 32 bit variabile (vengono usate tipicamente per test, monitoraggio)**
- **Dati**

# 15/11/2021

lunedì 15 novembre 2021 11:19

(manca una slide qui)

Se il router riceve un datagramma la cui dimensione supera l'MTU della rete verso cui deve inoltrare quel datagramma il router grammenta il datagramma IP in due o più datagrammi più piccoli, detti frammenti

Il riassemblaggio avviene alla destinazione

Se qualche frammento non arriva a destinazione si scarta l'intero datagramma

Ogni frammento è a sua volta un datagramma IP completo, il suo percorso può essere diverso da quello degli altri frammenti del datagramma

I frammenti possono arrivare fuori ordine, coe si ricompone l'ordine?

Si usa identificatore, flag e offset

- Identificatore (16 bit) assocaito al datagramma dell'host sorgente
  - o I frammenti di quel datagramma mantendono il valore di questo campo
- Per metterli in ordino ho l'**offset** posizione relativa del frammento come multiplo di 8 byte
- Flag: identificare qual è l'ultimo frammento
  - o 0: riservato
  - o 1: *do not fragment*
    - 0: pacchetto può essere frammentato
    - 1: non deve essere frammentato
  - o 2: *more fragments*
    - 0: il pacchetto è l'ultimo
    - 1: non è l'ultimo

## Indirizzamento IP

Ogni host è connesso a internet tramite un'interfaccia d rete che è il confine fra host ed il collegamento su cui vengono inviati i datagrammi.

Ad ogni interfaccia è assegnato un **indirizzo ip**

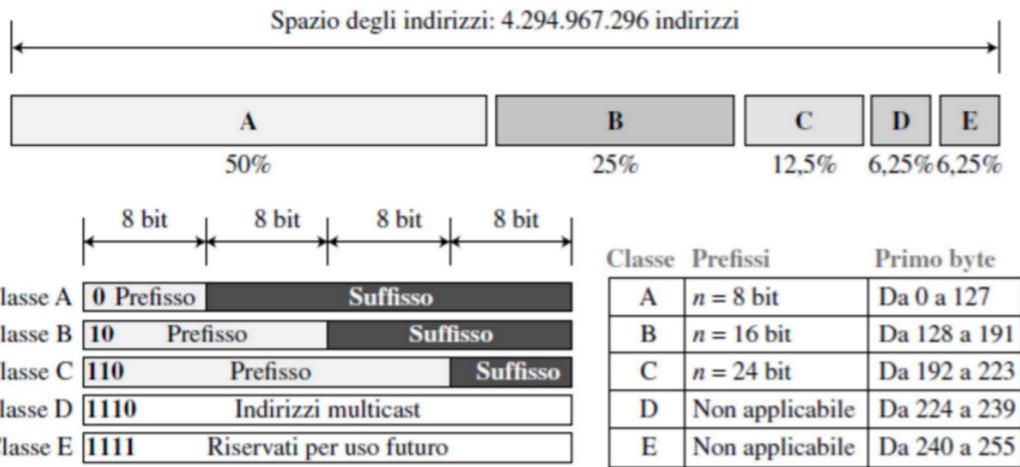
Un indirizzo ip è costituito a 32 bit ( 4 byte) rappresentati in notazione decimale puntate

Ogni host ha un indirizzo univoco diviso in 2 parti: **network ID + Host ID** identificano una rete IP su Internet e l'host su quella rete IP

**Indirizzamento classico:**

**Classfull addressing:**

Spazio di indirizzi suddiviso in 5 classi in modo statico



Dato un indirizzo IP è facile risalire all'indirizzo di rete, ma poco flessibile nell'utilizzo dello scarso range di indirizzi disponibili

### Classless addressing

Ipv6

### Classless addressing

Notazione CIDR ( Classless Intrdomain Routing):

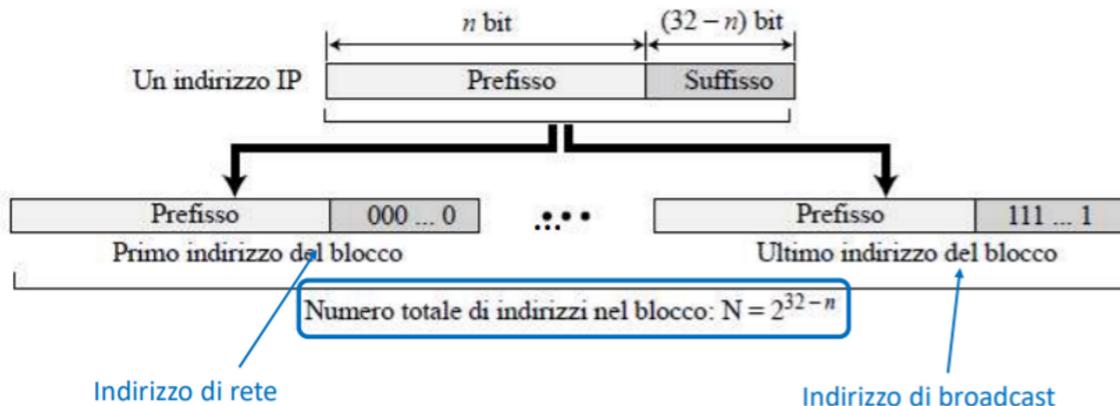


Esempi:

12.24.76.8/8

23.14.67.92/12

220.8.24.255/25

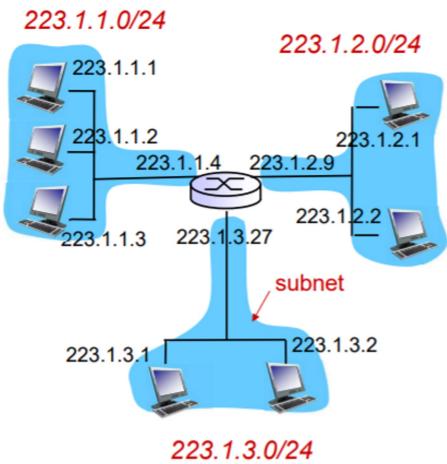


Indirizzo IP composto da un prefisso da  $n$  bit e un suffisso da  $32 - n$  bit

Numero massimo di host:

Una rete con un prefisso di rete di  $n$  bit può avere al massimo  $2^{(32 - n)} - 2$  host (tolgo indirizzo di rete e di broadcast)

Il prefisso indica l'indirizzo di rete



### Maschera di sottorete (subnet mask)

Indirizzo: a.b.c.d/n

N bit più a sinistra costituiscono la parte di indirizzo di rete

*Subnet mask:* numero composto da 32 bit in cui i primi n bit a sx sono impostati a 1 e i rimanetnti a 0, serve a distinguere quale parte di un indirizzo ip identifica la rete e quale l'host

: l'indirizzo IP **150.217.8.42** ha netmask **255.255.255.0**

Indirizzo IP	<b>150.217.8.42</b>	<b>10010110 11011001 00001000 00101010</b>
Subnet Mask	<b>255.255.255.0</b>	<b>11111111 11111111 11111111 00000000</b>
AND	<b>150.217.8.0</b>	<b>10010110 11011001 00001000 00000000</b>

La rete ha indirizzo  
**150.217.8.0/24**

### Assegnazione di blocchi di indirizzi:

Gli indirizzi ip sono gestiti da ICANN e assegnati in blocchi agli ISP, come assegnano gli ISP gli indirizzi ai clienti?

- ISP che deve suddividere un blocco in 8 blocchi di indirizzi (un blocco di 512 indirizzi per ciascuna organizzazione)

Blocco dell'ISP	<b>11001000 00010111 00010000 00000000</b>	<b>200.23.16.0/20</b>
Organizzazione 0	<b>11001000 00010111 00010000 00000000</b>	<b>200.23.16.0/23</b>
Organizzazione 1	<b>11001000 00010111 00010010 00000000</b>	<b>200.23.18.0/23</b>
Organizzazione 2	<b>11001000 00010111 00010100 00000000</b>	<b>200.23.20.0/23</b>
...	.....	....
Organizzazione 7	<b>11001000 00010111 00011110 00000000</b>	<b>200.23.30.0/23</b>

- Il numero di indirizzi N in ogni subnetwork deve essere una potenza di 2
- La lunghezza del prefisso di ogni sottorete (n) va calcolata con la formula
  - $n = 32 - \log_2 N$  dove N è il numero di indirizzi della sottorete
  - Nel nostro caso: 23 -> sottoblocco identificato con 3 bit aggiuntivi
- Si assegnano blocchi di indirizzi **contigui** (nell'esempio 000, 001, 010..)
- NB se i blocchi sono di dimensioni diverse si parte dai blocchi più grandi

## Aggregazione di indirizzi

Il CIDR permette anche di aggregare i blocchi di indirizzi per rendere più efficiente l'instradamento

### Indirizzi speciali:

- **This-host 0.0.0.0:** usato quando un host ha necessità di inviare un datagramma ma non conosce il proprio indirizzo IP (quando deve mandare il pacchetto per farselo assegnare)
- **Limited-broadcast 255.255.255.255:** usato quando un router o un host devono inviare un datagramma a tutti i dispositivi che si trovano all'interno della rete. I router bloccano la propagazione alla rete locale
- **Loopback 127.0.0.1:** il datagramma con questo indirizzo di destinazione non lascerà l'host locale
- **Indirizzi privati:** quattro blocchi riservati per indirizzi privati (riservati per reti locali)  
**10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16, 196.245.0.0/16**
- **Indirizzi multicast:** blocco **224.0.0.0/4**

### Assegnazione di un indirizzo IP

- **Configurazione manuale:** l'amministratore configura direttamente nell'host indirizzo IP, gateway/route, netmask, e indirizzo IP di almeno un server DNS
- **DHCP:** L'host ottiene il proprio indirizzo e le altre informazioni in modo automatico

### DHCP

Quando un host si aggiunge alla rete ottiene dinamicamente un indirizzo IP da un programma server in rete

Cosa offre il servizio:

Rinnovo indirizzo in uso

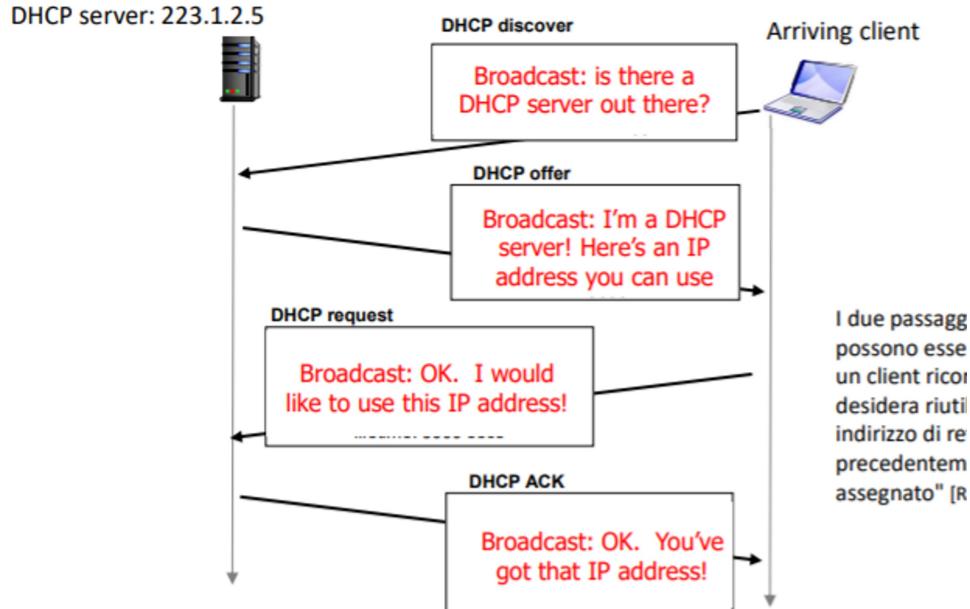
Riuso di un indirizzo precedentemente assegnato

Supporto per utenti in mobilità che si uniscono/lasciano la rete

### È un protocollo client-server

- L'host invia in broadcast un messaggio **DHCP discover (opzionale)**
- Il server DHCP risponde con un messaggio **DHCP offer (opzionale)**
- L'host richiede un indirizzo IP: messaggio **DHCP request**
- Il server DHCP risponde con un messaggio **DHCP ack (se la richiesta va a buon fine)**

Il server DHCP può essere ospitato sul router, servendo tutte le sottoreti a cui è collegato



# 17/11/2021

mercoledì 17 novembre 2021 11:13

## Inoltro

Inoltro di un pacchetto verso l'uscita verso un altro host o router che si prende in carico del datagramma o alla destinazione finale se siamo nella sottorete dell'host finale

## Inoltro diretto

Il pacchetto IP ha come destinazione un host nella propria sottorete

Si manda direttamente il datagramma al destinatario e l'indirizzo è quello del destinatario MAC

## Inoltro indiretto

Il pacchetto ha come destinazione un host di un'altra rete, si delega l'invio a qualcun altro  
L'indirizzo di dest IP è quello originale, l'indirizzo a livello di collegamento è quello del router (a cui viene delegato l'invio)

Si osserva come, in entrambi i casi, condizioni necessarie perché tutto funzioni sono che:

» esista un cammino (funzionante e) diretto, a livello data-link, tra tutti gli host che appartengono ad una stessa sottorete;

» ogni host coinvolto abbia un indirizzo IP "giusto", cioè con uguale net ID (cioè appartenga alla stessa sottorete) e con host ID univoco nella sottorete.

*Le due condizioni insieme diventano condizione necessaria e sufficiente perché la comunicazione "funzioni".*

Manca la parte del NAT

## Protocollo ICM

Entra in azione quando si è verificato un errore

Protocollo implementato per funzioni di controllo

Può essere usato sia da host che da router per errori o per sapere se un host è raggiungibile PING

I messaggi ICMP sono incapsulati all'interno di datagrammi IP, gli host che implementano un protocollo IP devono supportare anche ICMP

Casi tipici di utilizzo:

Router o host che devono informare la sorgente che qualcosa è andato storto:

- Host non raggiungibile, porta non raggiungibile, niente in ascolto...

Ci sono regole per evitare il moltiplicarsi di questi pacchetti di controllo:

Per i frammentati vengono mandati solo relativi al primo pacchetto

Non vengono mai inviati in risposta a qualcosa che non rappresenta un host in modo univoco (0.0.0.0 localhost)

Destinazione ip broadcast o ip multicast

Non mandati in risposta a messaggi di errore ICMP, ma possono essere inviati in risposta a messaggi

## ICMP di interrogazione

### Tipi di messaggio icmp

- Messaggi di segnalazione/error
- Messaggi richiesta/risposta

#### Valori del tipo e dei codici:

Messaggi di segnalazione errori

**03:** destinazione non raggiungibile (codici da 0 a 15)

**04:** source quench (solo codice 0)

**05:** reindirizzamento (codici da 0 a 3)

**11:** tempo scaduto (codici 0 e 1)

**12:** problema ad un parametro (codici 0 e 1)

Messaggi di richiesta

**08 and 00:** richiesta e risposta echo (solo codice 0)

**13 and 14:** richiesta e risposta timestamp (solo codice 0)

Nota: si veda il sito web del volume per ulteriori spiegazioni relative ai valori dei codici.

Nell'header ICMP i campi Tipo e codice indicano il tipo e significato dei messaggi (entrambi 8 bit)

ICMP Tipo	Codice	Descrizione
0	0	risposta al messaggio di eco (a ping) - <i>echo replay</i>
3	0	rete di destinazione irraggiungibile - <i>destination network unreachable</i>
3	1	host di destinazione irraggiungibile - <i>destination host unreachable</i>
3	2	protocollo di destinazione irraggiungibile - <i>destination protocol unreachable</i>
3	3	porta di destinazione irraggiungibile - <i>destination port unreachable</i>
3	6	rete di destinazione sconosciuta - <i>destination network unknown</i>
3	7	host di destinazione sconosciuto - <i>destination host unreachable</i>
4	0	strozzamento della sorgente (controllo della congestione) - <i>source quench</i>
8	0	richiesta di echo - <i>echo request</i>
9	0	annuncio dal router - <i>router advertisement</i>
10	0	scoperta del router - <i>router discovery</i>
11	0	TTL scaduto - <i>TTL expired</i>
12	0	cattiva intestazione IP - <i>IP header bad</i>

### Ping

Il ping è un programma che ci permette di mandare messaggi di richiesta e attendere risposta  
ICMP invia una richiesta, fa un payload lo incapsula direttamente in un datagramma IP tipo 8 e codice 0, un altro host se attivo può rispondere con una risposta eco tipo 0 e codice 0  
Fornisce anche l'RTT, la richiesta viene mandata più volte per fare una statistica

### Traceroute

Traceroute individua il percorso di un datagramma dalla sorgente alla destinazione, mostra l'indirizzo IP dei router che vengono visitati lungo il percorso

Di base si imposta il TTL a un certo numero partendo da uno zero basso e via via incrementare in modo da costringere i router nel percorso a mandare il messaggio di errore di pacchetto esaurito

- Uso del protocollo UDP, si invia un datagramma ad una porta su cui è improbabile che un processo sia in ascolto
- I datagrammi sono configurati in modo da scatenare l'invio di messaggi di errore di tipo (TTL esaurito, porta non raggiungibile)

Tipicamente vengono mandati 3 messaggi per ogni TTL

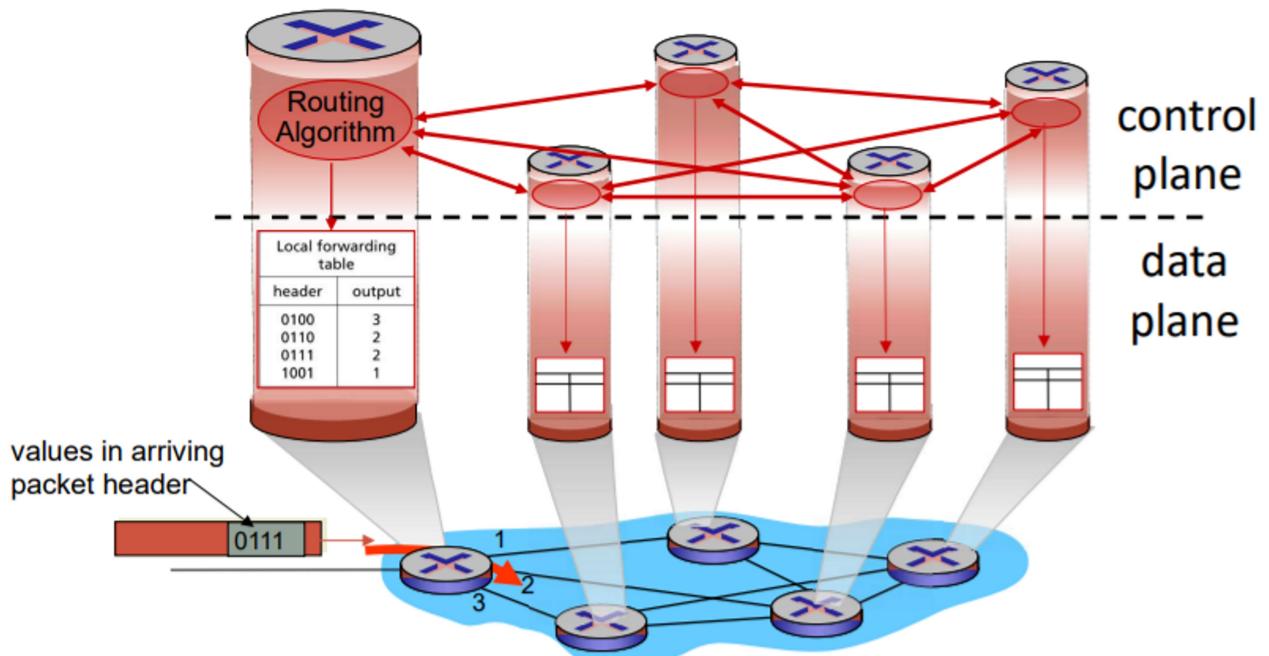
## Architettura di un router

### Forwarding: *data plane*

Azione del nodo a livello di rete (terminale host o router) per trasferire i pacchetti sull'appropriato collegamento in uscita, tramite la tabella di inoltro *data plane*

### Routing: *control plane*

Algoritmi di routing e protocolli di routing che li implementano e le comunicazioni tra i router a determinare i valori che verranno inseriti nella tabella di inoltro



routing basato su destinazione

Il router va a prendere il valore della destinazione nel pacchetto, consulta la tabella di inoltro e decide su quale uscita inoltrare il pacchetto

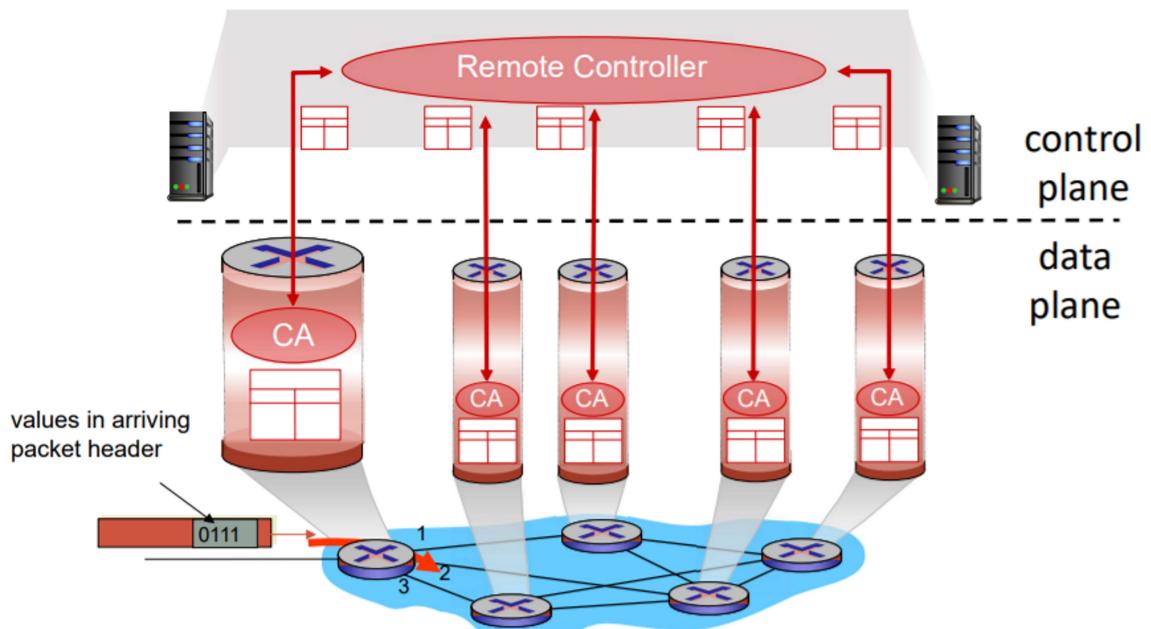
Routing decentralizzato:

Ogni router implementa algoritmi e protocolli di instradamento, costruisce la sua tabella scambiandosi messaggi con gli altri router, in base a queste informazioni costruisce la tabella di instradamento

## Software defined networking

In questo approccio, mentre nell'architettura decentralizzata ogni router implementa funzioni del piano dati e funzioni del piano di controllo, in questo approccio abbiamo che a livello di router si implementa principalmente la funzione di inoltro, mentre la funzione di controllo non è implementata sui router, ma da un software (Control agent o controller remoto?) che riceve dai router delle info sulla tipologia, stato collegamenti e traffico in base a queste info si costruisce la topologia della rete info sui punti raggiungibili e i percorsi. La funzione del control plane e dei router è lasciata al software in esecuzione su un server.

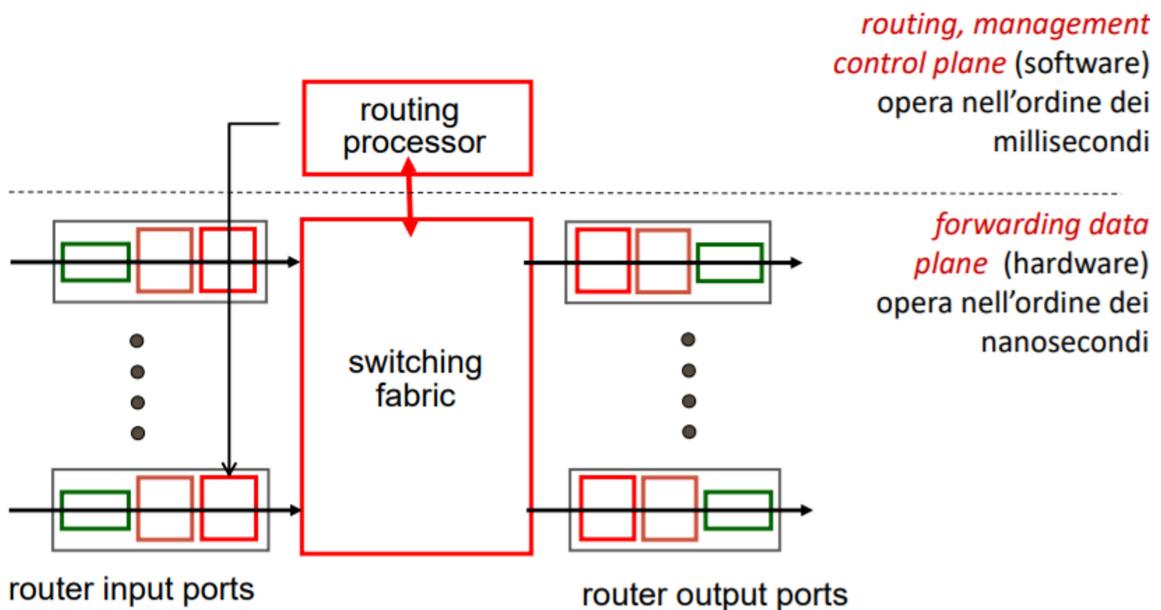
# Routing logicamente centralizzato



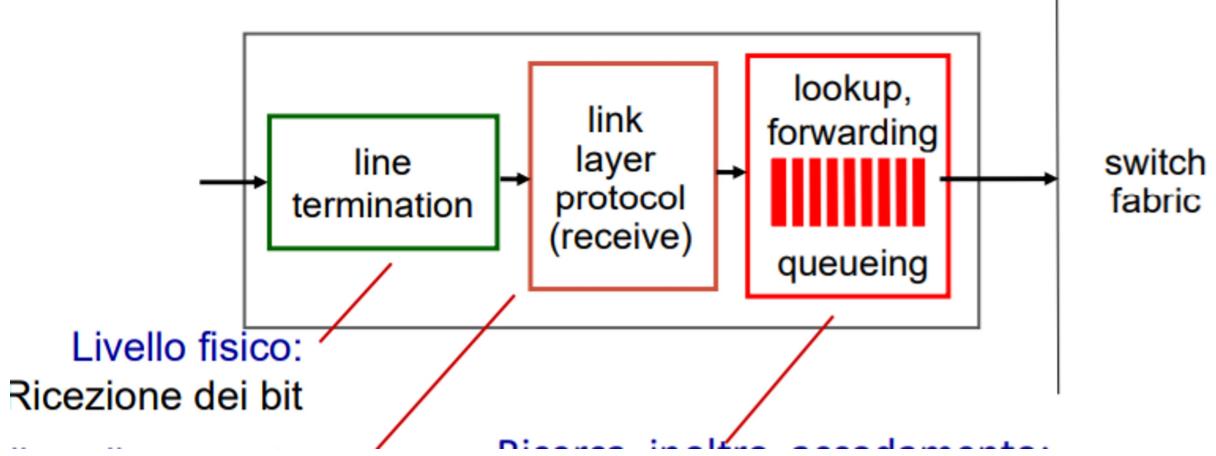
Vantaggio principale di un routing decentralizzato:  
Se c'è un cambiamento appena il router lo sa aggiorna i percorsi

## Architettura di un router

- Porte di input
- Porte di output
- Processore di routing
- Switching fabric (struttura di commutazione)



Porta di input: terminazione di linea riceve il segnale elettrico  
Link layer protocol: implementazione dei protocolli di collegamento  
Lookup forwarding queueing: decidere su quale porta inoltrare il messaggio



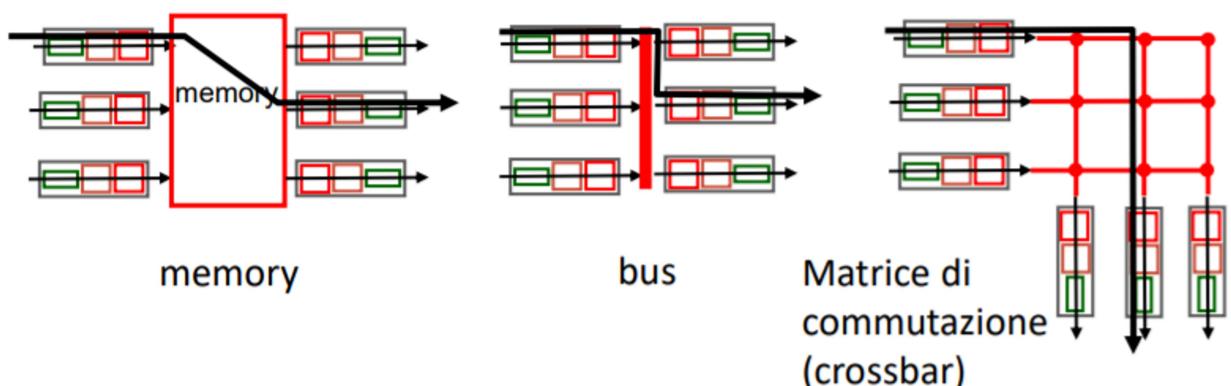
Ogni porta tiene una copia della tabella di inoltro per determinare la porta nel modo più veloce possibile

Elaborazione line rate

Se arrivano i pacchetti a una velocità maggiore di quella di inoltro si crea accodamento

Struttura di commutazione

Mette in relazione la porta di input e quella di output

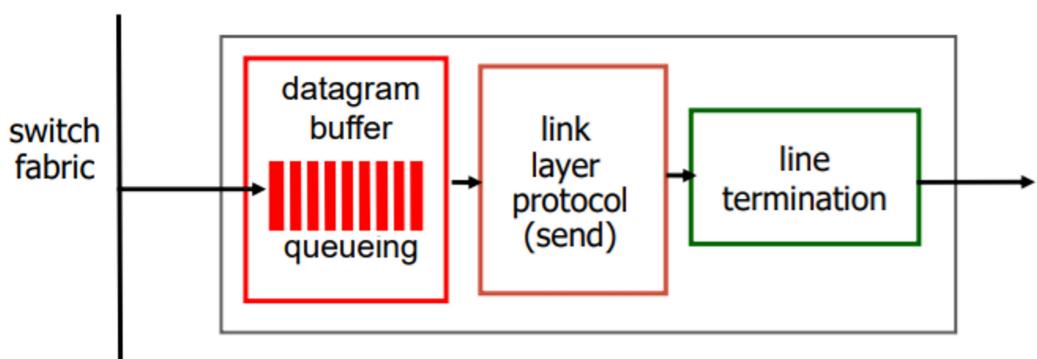


Memoria: i pacchetti vengono copiati in memoria e spostati da una porta d'input a una di output

Bus: bus condiviso per trasferire i pacchetti

Matrice di commutazione (interconnessione): implementata con dispositivi a stato solido in cui è possibile chiudere i collegamenti per creare un percorso tra p input e p output

Porta output



Coda per eventuali pacchetti in attesa se arrivano a una velocità maggiore di quella di uscita (vincolo della tecnologia usata per trasmettere)

I router possono avere più code per la gestione dei pacchetti in uscita, è possibile "accelerare" il flusso di traffico andando a rallentare gli altri (traffic shaping) dai una priorità al flusso di traffico che deve essere inviato con meno attesa e rallentati gli altri, dai una priorità a una determinata coda

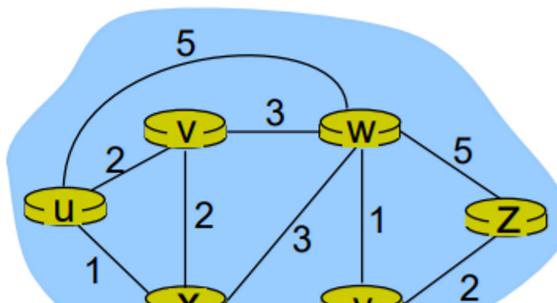
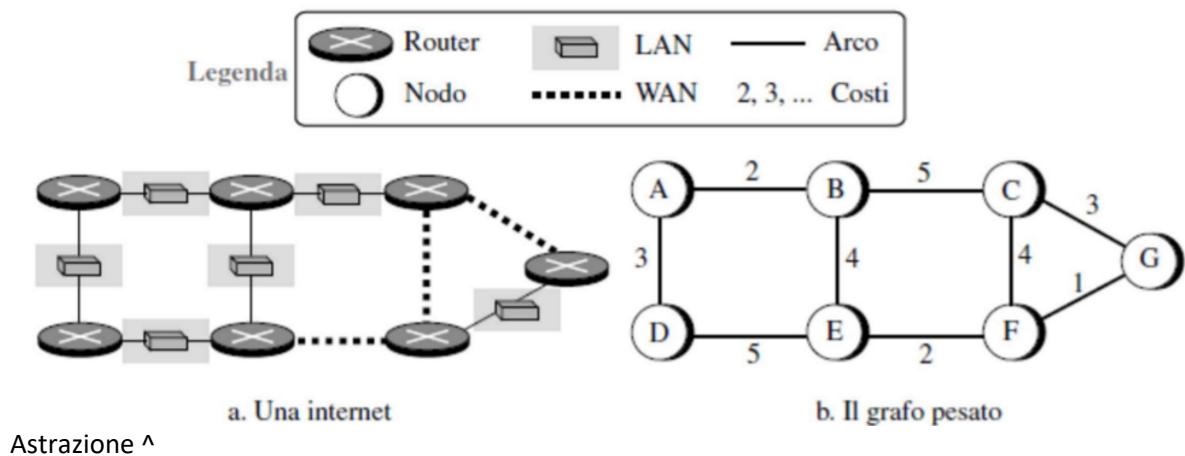
Ritardi e perdite:

Accodamento e buffer overflow (perdita) nei buffer di I/O

### Routing:

Unicast: un datagramma è destinato a una sola destinazione

Multicast: il datagramma è destinato a un gruppo di host



$c(x,x')$  = costo del link  $(x,x')$   
e.g.,  $c(w,z) = 5$

**key question:** qual è il cammino a costo minimo tra u e z ?

**routing algorithm:** algoritmo che calcola il cammino a costo minimo

### Instradamento statico:

L'amministratore configura manualmente la tabella di inoltro. Usato per reti piccole la cui topologia non varia molto

### Routing dinamico:

Protocolli specifici che provvedono automaticamente a inserire nella tabella le entry relative ai possibili percorsi

### Algoritmi di instradamento

- Globali: ciascun router riceve informazioni su tutta la topologia della rete, lancia l'algoritmo e ottiene la tabella di inoltro (link state) -> architettura centralizzata
- Decentralizzato: nessun nodo conosce la topologia di tutta la rete, comincia a calcolare i percorsi andando a ricevere le info dai nodi collegati direttamente, dopo un tot di aggiornamenti riesce a costruirsi percorsi più precisi

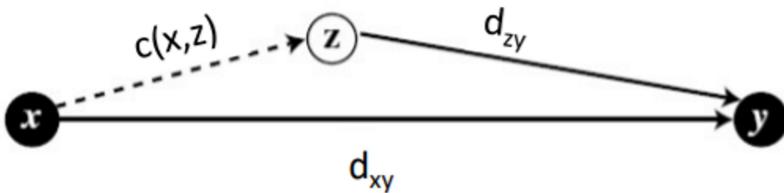
### Distance vector

Su un nodo riceve parte delle informazioni da uno o più nodi direttamente connessi (vicini) effettua calcoli e comunica i risultati ai vicini

- Iterativo

- Asincrono: i nodi non sono sincronizzati sulla topologia della rete
- Per il calcolo del percorso minimo si basa sull'equazione di bellman ford, ci detta quali sono le info che un nodo deve mandare agli altri epr aggiornare la sua tabella

$$d_{xy} = \min (d_{xy}, (c(x,z) + d_{zy}))$$



D: distanza tra x e y, è una metrica, tra x e y ci possono essere altri nodi fisici

C(x, v) costo tra nodo sorgente x e il vicino v (noto)

Il costo è il costo del collegamento fisico, la distanza è il costo di un percorso, vado ad astrarre il collegamento fisico da x a y

### Bellman-Ford equation

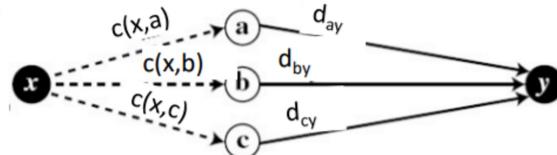
$d_{xy}$  := costo del cammino a costo minimo da x a y

allora

$$d_{xy} = \min_v \{ c(x,v) + d_{vy} \}$$

|                   | Costo dal vicino v alla destinazione y  
 |                   | costo da x al vicino v

min su tutti i vicini v di x

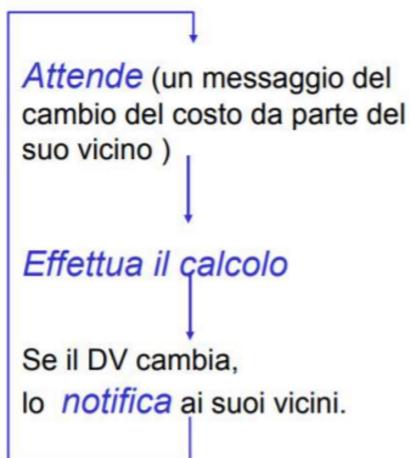


Come funziona l'algoritmo:

- Ciascun nodo x inizia con una stima delle distanze verso tutti i nodi in N
  - $D_{xy}$  = stima del costo minimo da x a y **per ogni**  $y \in N$
  - Il nodo x mantiene quindi le seguenti informazioni:
    - Conosce il costo verso ciascun vicino v:  $c(x,v)$
    - x mantiene il suo vettore distanza  $D_x = [D_{xy} : \text{per ogni } y \in N]$
    - Riceve i vettori distanza dei suoi vicini. Per ogni vicino v, x mantiene
- $D_v = [D_{vy} : \text{per ogni } y \in N]$

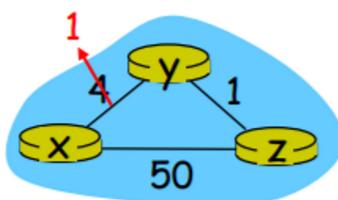
- Idea di base:
- Ogni nodo invia una copia del proprio vettore distanza a ciascuno dei suoi vicini.
- Quando un nodo  $x$  riceve un nuovo vettore distanza, DV, da qualcuno dei suoi vicini  $v$ , lo salva e usa l'equazione di Bellman-Ford per aggiornare il proprio vettore distanza
 
$$D_{xy} = \min \{ c(x,v) + D_{vy} \} \text{ per ogni } y \in N$$
- Finché tutti i nodi continuano a cambiare i propri DV in maniera asincrona, ciascuna stima dei costi  $D_{xy}$  converge all'effettivo costo del percorso a costo minimo  $d_{xy}$

Ciascun nodo:

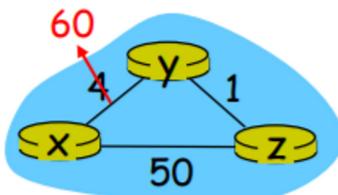


Esempio slide 24

I costi dei collegamenti possono cambiare (algoritmi di namici)



Il costo da 4 diventa 1, il nodo rileva il cambiamento, ricalcola il vettore distanza e lo manda ai vicini, e i vicini a loro volta aggiornano le loro se gli cambia qualcosa



Il costo del collegamento aumenta -> può portare a dei problemi  
 Y vede che il collegamento aumenta la distanza, ma z gli aveva detto che raggiungeva x con costo 5, y quindi y pensa che con un costo di 6 può raggiungere x, y lo manda a z che aggiorna di nuovo e così via per un po', può crearsi un loop infinito (problema conteggio verso l'infinito)

- Split-horizon with poisoned reverse

Se un nodo x inoltra V i pacchetti destinati a Z allora x invia a v che la distanza verso Z è infinito, le rotte ricevute tramite un'interfaccia devono essere pubblicate indietro a quell'interfaccia con una metrica non raggiungibile

### Algoritmo link state

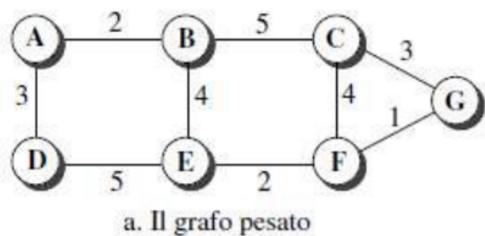
Algoritmo globale

La topologia di rete e tutti i costi dei collegamenti sono noti a tutti i nodi attraverso il "link-state-broadcast"

Tutti i nodi dispongono delle stesse info

Calcola il cammino a costo minimo da un nodo a tutti gli altri nodi della rete, si usa l'algoritmo di dijkstra

Crea una tabella di inoltro per quel nodo



	A	B	C	D	E	F	G
A	0	2	$\infty$	3	$\infty$	$\infty$	$\infty$
B	2	0	5	$\infty$	4	$\infty$	$\infty$
C	$\infty$	5	0	$\infty$	$\infty$	4	3
D	3	$\infty$	$\infty$	0	5	$\infty$	$\infty$
E	$\infty$	4	$\infty$	5	0	2	$\infty$
F	$\infty$	$\infty$	4	$\infty$	2	0	1
G	$\infty$	$\infty$	3	$\infty$	$\infty$	1	0

b. Il link-state database

## Notazione

- $c(x,y)$ : costo del collegamento dal nodo x al nodo y
  - $\infty$  se non sono adiacenti.
- $D(v)$ : costo del cammino dal nodo origine alla destinazione v per l'iterazione corrente
- $p(v)$ : immediato predecessore di v lungo il cammino
- $N'$ : sottoinsieme di nodi per cui il cammino a costo minimo dall'origine è definitivamente noto

Algoritmo di dijkstra

```

1 Initialization of node u:
2 N' = {u}
3 Per tutti i nodi v
4   se v è adiacente a u
5     allora D(v) = c(u,v)
6   altrimenti D(v) = ∞
7
8 Loop
9   trova w non in N' tale che D(w) sia minimo
10  aggiungi w a N'
11  aggiorna D(v) per tutti v adiacenti a w e non in N' :
12    D(v) = min( D(v), D(w) + c(w,v) )
13  /* il nuovo costo verso v è il vecchio costo verso v oppure il
costo del cammino minimo noto verso w più il costo da w a v */
14 Finché tutti i nodi sono in N'

```

#### Sistema autonomo

Gruppo di router sotto lo stesso controllo amministrativo

Dentro un sistema autonomo, l'amministratore sceglie quale protocollo di routing usare (LS o DV) ci sono delle ISP che possono decidere di partizionare in più SA la stessa rete

## Instradamento gerarchico

I router sono organizzati in sistemi autonomi (AS: gruppi di router sotto lo stesso controllo amministrativo)

La rete è un'interconnessione di AS

- Instradamento dentro un sistema autonomo: **Interior Gateway Protocol (IGP)**
- Protocollo di routing tra sistemi autonomi: **Exterior Gatewat Protocol (EGP)**

All'interno di un AS l'amministrazione sceglie che protocollo usare, tra AS si usa sempre EGP

## Tipi di AS:

- **Stub**: collegato solo a un altro AS
- **Multihomed**: collegato a più di un altro AS (ma trasporta - come stub - solo traffico di cui è origine o destinazione) non trasportano traffico di altri AS, ma solo traffico originato da loro e uscente (es. azienda che usa più ISP)
- **Transito** inoltro di traffico da un AS a un altro

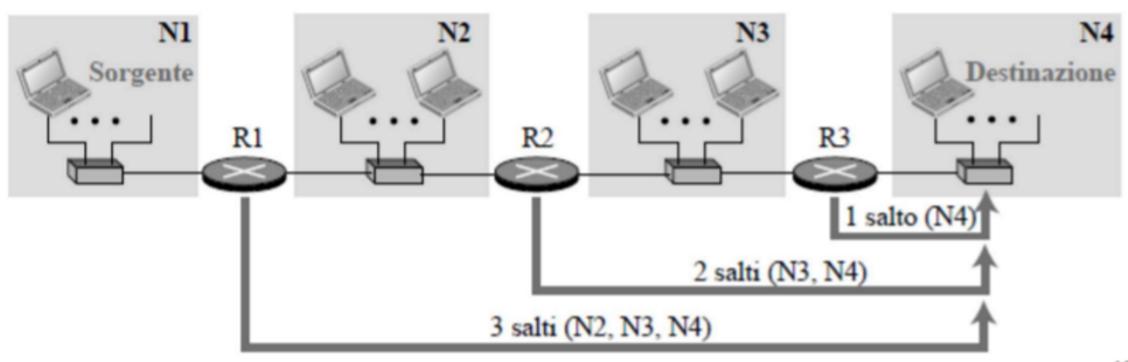
Ogni AS ha un identificativo associato (assegnato da IANA) (ASN)

## Protocolli INTRA-AS

### RIP

Routing information protocol, implementa l'algoritmo DV con poisoned Reeverse.

Metrica per la distanza: nel RIP si usa il numero di sottoreti attraversate



# Esempio

Tabella d'inoltro per R1

Rete di destinazione	Prossimo router	Costo (in hop)
N1	—	1
N2	—	1
N3	R2	2
N4	R2	3

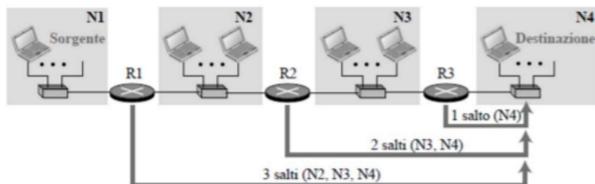


Tabella d'inoltro per R3

Rete di destinazione	Prossimo router	Costo (in hop)
N1	R2	3
N2	R2	2
N3	—	1
N4	—	1

Tabella d'inoltro per R2

Rete di destinazione	Prossimo router	Costo (in hop)
N1	R1	2
N2	—	1
N3	—	1
N4	R3	2

47

Domanda possibile ^

RIP è un processo *demone* che usa UDP sta in ascolto sulla porta 520.

Ogni 30 sec si scambiano i distance vector (oppure se cambia la tabella di inoltro)

## Algoritmo

Un nodo invia la tabella di inoltro ai propri vicini (periodi di circa 30 secondi)

Nodo R riceve advertisement da vicino V con  $D_V[y]$ :

$$D_R[y] = 1 + D_V[y]$$

Aggiunge un hop ( $c(x,v)$ ) e considera come next-hop V

Il nuovo percorso viene inserito in tabella se:

- se è un percorso non presente nella tabella e quindi viene aggiunto
- se  $D_V[y] + 1 < D_{\text{Old}}[y]$  costo ricevuto inferiore a quello del vecchio percorso
- Se V è nextHop del vecchio e nuovo percorso, ma il costo è cambiato (diminuito o incrementato)

$D_V[y]$ : distance vector di y -> distanza di v da y

4

## OSPF (Open Shortest Path First)

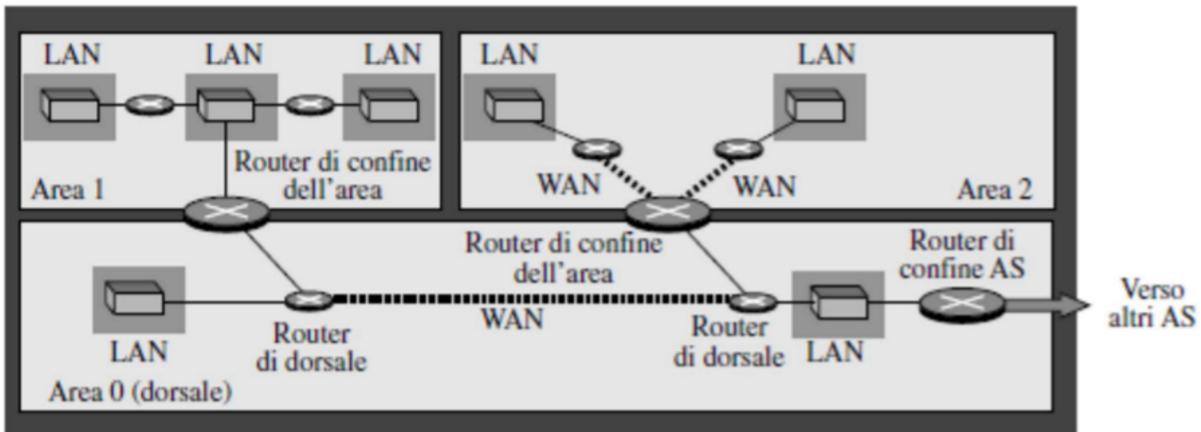
Algoritmo: Link State

Metrica decisa dall'amministratore (latenza, affidabilità, banda, numero di hop..)

RIP viene usato per reti medio-piccole, OSPF può essere usato per reti più grandi.

L'AS può essere *suddiviso in aree* (per ridurre flooding di Link State Packet), una delle aree fa da dorsale. (i pacchetti link state vengono mandati in broadcast)

## Sistema autonomo (AS)

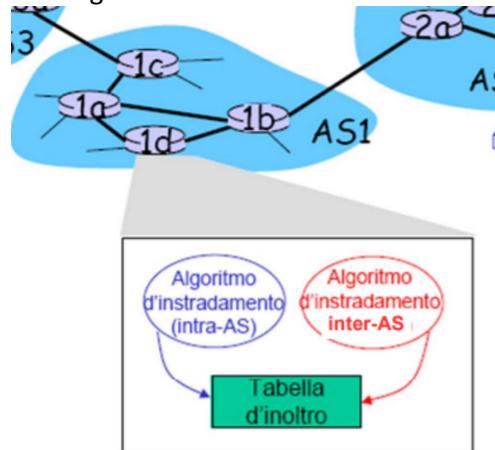


Si spezza la comunicazione broadcast, i messaggi vengono mandati attraverso le aree, per arrivare alle altre aree devono passare dalla backbone

Usa IP (porta 89)

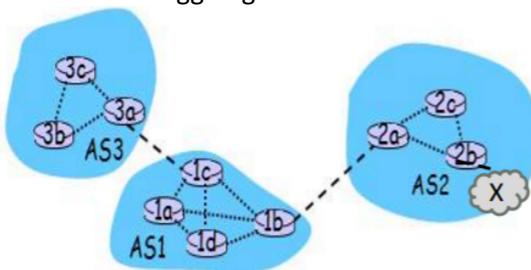
## Routing INTER-AS

Routing TRA sistemi autonomi



Gateway router si scambiano informazioni sulla raggiungibilità delle reti esterne

## Esempio



- AS1 scopre (grazie a INTER-AS routing protocol) che una sottorete X è raggiungibile via AS2 (a cui AS1 è collegato mediante il gateway 1b)
- AS1 propaga (con INTER-AS routing protocol) tale informazione al suo interno
- Un router R (es. 1d in figura) di AS1 riceve l'informazione “rete X raggiungibile via AS2”:  
Aggiorna (se necessario) la tabella di inoltro

## BGP

Protocollo per gestire comunicazioni inter-as, è un protocollo unico.

Andiamo a distinguere le sessioni **esterne** e **interne**

Quando vengono "annunciate" delle rotte, vengono annunciate il più possibile con indirizzi aggregati

Pubblicizzare un prefisso significa impegnarsi a instradare pacchetti destinati a reti in quel prefisso

Un gateway riceve info sulla sessione eBGP da altri AS e distribuisce informazioni ai router interni con sessioni iBGP

In BGP il percorso migliore non è necessariamente quello a costo minimo perché si parla di diverse amministrazioni (magari vodafone da tim non ci vuole passare)