Reti

Marini Mattia

1^o semestre 2^o anno

Indice

1 Introduzione ad Internet

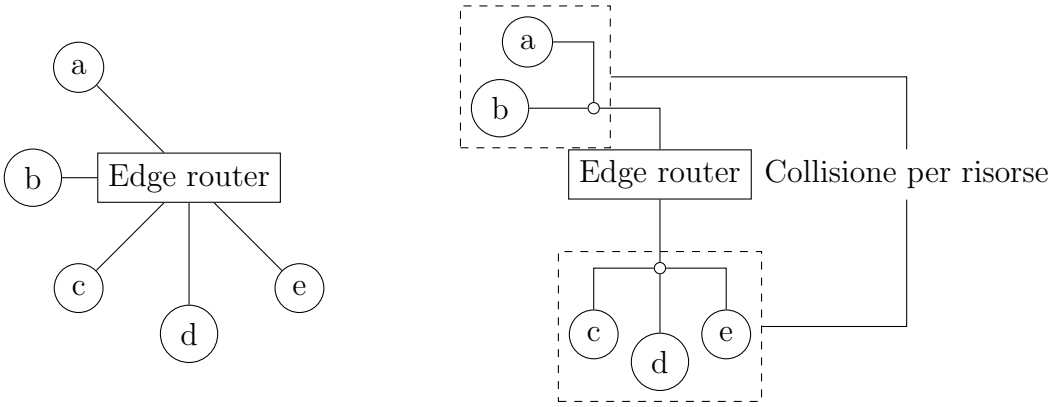
1.0.0 Terminologia

- Host / sistemi terminali / end systems sistemi (coputer/server) interconnessi
- Clien: host che invia richieste ad un server, attendendo risposta
- Server: host che fornisce un determinato servizio da un terminale. I server si trovano nella periferia della rete
- Modem: dispositivo che converte i bytes in tensioni di corrente che viaggiano sui cavi
- Router: dispositivo che si occupa di decidere dove i pacchetti vengono spediti. Poco importante per i router domestici, di più per i router che si trovano al centro della rete
- Protocollo: standard che definisce il formato e l'ordine secondo il quale vengono scambiati messaggi tra due o più entità di comunicazione, così come le azioni intraprese quando i dati vengono inviati/ricevuti. *TCP, IP, HTTP, Skype, Ethernet* sono tutti protocolli
- Internet: rete delle reti, ossia una rete che connette le diverse reti private
- Servizio affidabile: servizio che deve garantire che i dati arrivino tutti da sorgente a destinazione, ad esempio quando richiediamo una pagina web
- Servizio best effort: servizio che non garantisce necessariamente che i dati arrivino tutti da sorgente a destinazione, ad esempio un servizio di *streaming*
- Organismi che gestiscono standard internet:
  - RFC: REquest for comments
  - IETF: Internet Engineering Task Force

  queste organizzazioni fanno si che due schede di rete di produttori diversi possano comunicare senza problemi
- Architettura peer to peer: un end system si collega ad un altro end system senza passare per un server
- Architettura client/server: il client riceve un serviizo da un server. Google, ad esempio, disloca i server in diversi *data center*, nella periferia della rete
- Modem dial-up
  - Lento (8/56 kbit/s)
  - Non è possibile navigare e telefonare contemporaneamente
- DSL
  - Spesso è *asimmetrica(ADSL)*, ossia vi sono diverse velocità in download e upload

- Fino a 1/5 mbit/s in upstream(upload)
- Fino a 10/20 mbit/s in downstream(download)

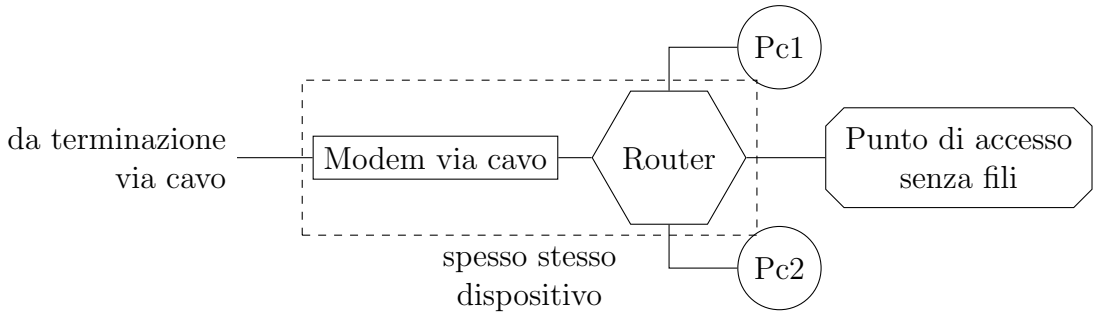
- LAN, Local Area Network è la rete che collega tutti i dispositivi all'interno di aziende/università. Tutti i dispositivi sono collegati all'edge router



- Reti wireless possono essere principalmente di due tipi:
  - LAN wireless: il WiFi. In particolare la versioni sono 802.11 a/b/g/n/ac, con corrispettive velocità di 11, 54, 100+ Mbit/s
  - Rete di accesso wireless geografica, ad esempio GPS, rete 4g/5g
  - Velocità da qualche Mbit fino a 1 Gbit al secondo

1.1 Struttura reti

1.1.0 Struttura reti domestiche



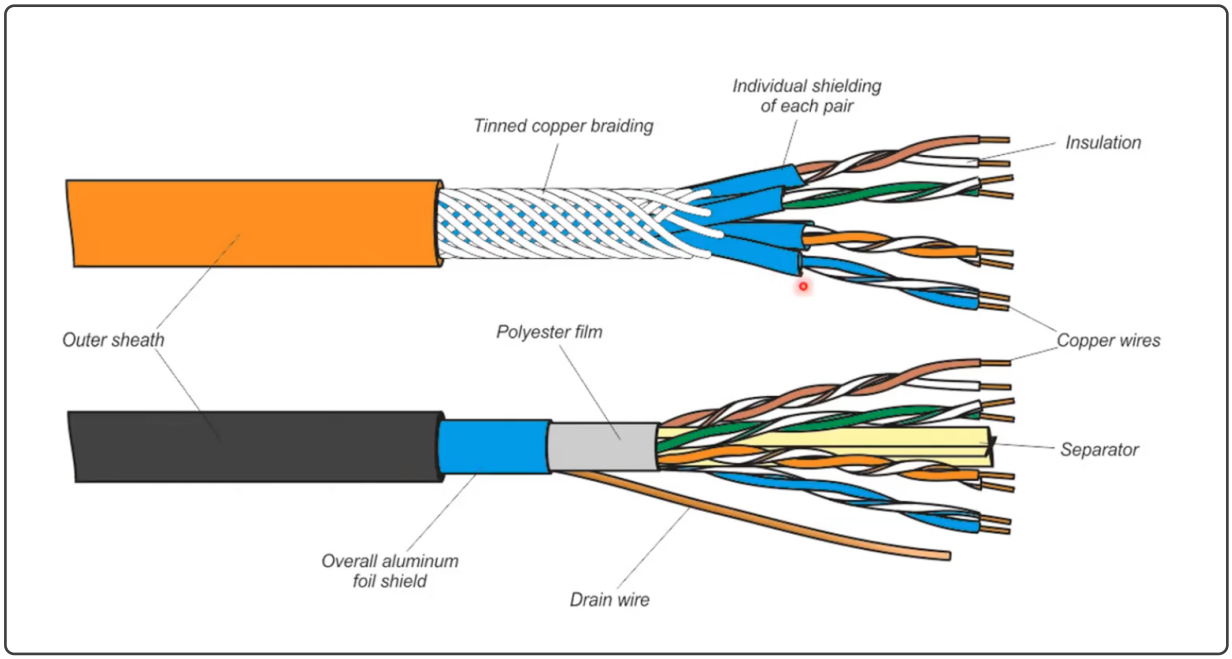
1.1.0 Mezzi trasmissivi

I mezzi sui quali veengono trasmessi i bit possono essere

- Mezzi guidati:
  - RAme
  - Fibra ottica
  - Cavo coassiale
- Mezzi a onda libera
  - I segnali si propagano nell'atmosfera o nello spazio, (es wifi, gps)

1.1.0 Il cavo

Tendenzialmente si usa la tecnologia a doppino intrecciato o TP. Questo è utilizzato per i cavi eternet



I cavi vengono deniminati come segue

X / Y TP

- X è la schermatura dell'intero cavo
  - U : unshielded
  - F : foiled (di solito, una lamina di alluminio)
  - S : maglia metallica intrecciata (di solito, rame placcato alluminio)
  - SF : entrambe
- Y è la schermatura di ogni doppino
  - U: unshielded
  - F : shielded

Esempi: U/UTP F/UTP S/FTP?

1.2 Il nucleo della rete (core)

Per trasferire dati da un end system ad un altro end system vi sono principalmente due metodi: la commutazione di circuito e la commutazione di pacchetto

1.2.0 Commutazione di circuito

In questo metodo, le risorse fisiche vengono riservate ai due end systems fino a quando non è finita la transizione di dati. Di fatto il circuito fisico che trasmette i bit viene riservato fino che la comunicazione non termina

- Risrse del circuito sono garantite
- Risorse non condivise con nessuno
- Necssario invio di segnali hand-shake per impostare la connssione (pensa a quando si fa una telefonata)

Nello scenario in cui una tratta debba essere divida da più utilizzatori, bisogna capire come spartire le risorse. Vi sono 3 approcci:

- Ripartizione bitrate: viene suddivisa la banda dedicata a ciascuna connessione (ho 10 Mbit/s, ne do 5 a testa)
- Divisione di frequenza (FDM): ogni utente può utilizzare una data frequenza per quanto tempo gli pare. Ad esempio un canale radio, possiede una frequenza per quanto vuole
- Divisione di tempo (TDM): ogni utente può utilizzare tutte le frequenze disponibili per un tempo limitato e definito a priori

Esercizio 1: *Calcolo tempo con TDM*

Quanto tempo occore per trasmettere 640.000 bit dall'host A all'host B su una rete a commutazione di circuito?

- Tutti i collegamenti presentano bit-rate di 1536 kbit/s
- Ciascun collegament utilizza TDM con 24 slot/secondo
- Si impiegano 500 ms per stabilire la connessione

Risposta:

- 24 slot/secondo quindi ogni utente utilizza 1/24 della potenza della rete. Ogni utente ha banda
$$\frac{1536}{24} = 64\frac{kb}{s}$$
- Per inviare i dati ho bisogno di
$$\frac{640000}{64} = 10s$$
- Per stabilire la connesione servono 500 ms quindi il tempo totale è
$$\text{Tempo totale} = \text{Tempo connessione} + \text{Tempo spedizione} = 0.5 + 10 = 10.5s$$

1.2.0 Commutazione di pacchetto

Le risorse possono prendere strade diverse. Per via di congestioni si potrebbe dover aspettare che sia disponibile una linea.

- Il traffico di dati viene suddiviso in pacchetti
- Ciascun pacchetto utilizza interamente il canale
- Le risorse vengono usare a seconda della necessità (*on demand*)

Questo per certi versi è più efficace che la commutazione di circuito, però possono esserci problemi di contesa per le risorse:

- La richiesta di risorse può eccdere il quantitativo disponibile
- Può esserci una congestione. Ciò significa che arriva troppo traffico in arrivo che non riesce ad essere smaltito sufficientemente rapidamente
- Ci deve essere store and forward. I pacchetti vengono memorizzati mentre attendono di essere spediti
- Ogni pacchetto deve essere ricevuto completamente prima di essere rispedito per le seguenti ragioni:
  - Per fare correzione di errori
  - Per capire dove va instradato il [1]pacchetto

La commutazione di pacchetto è l'opzione più utilizzata in quanto:

- E' ottima per spedire dati a raffica in maniera veloce, in quanto ogni host ha a disposizione tute le risorse
- Permette di emulare un comportamento *circuit like*, fornendo garanzie sulla larghezza di banda. Tuttavia questo problema è molto complesso e non ancora risolto

Tuttavia bisogna è necessario dire che:

- La commutazione di pacchetto potrebbe dare problemi di congestione, comportando ritardi e perdite di pacchetti

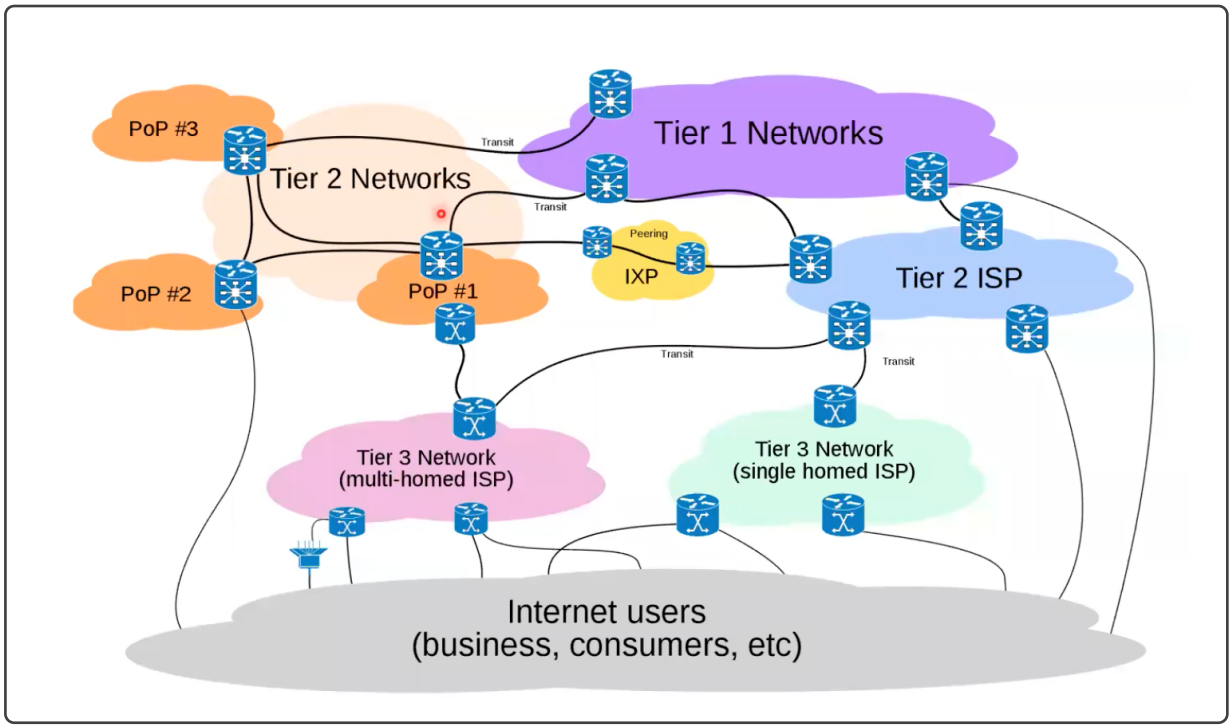
2 Struttura internet

Internet possiede una struttura gerarchica:

- ISP di livello 1
  - ISP di livello 2

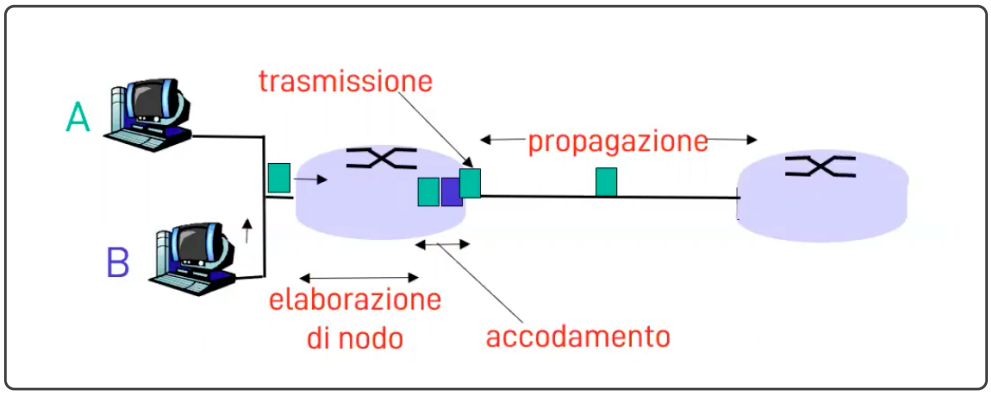
[1]In realtà, nei router moderni possono decidere dove instradare il pacchetto ancora prima che sia completamente ricevuto

* ISP di livello 3 (*last hop network*)



- IXP: internet exchange point. Struttura di una terza azienda nella quale gli ISP di livello 2 comprano spazio per router per effettuare peering con altri ISP di livello 2.
- ISP di livello 2: grandi compagnie di internet (Vodafone, Telecom...)
- PoP: point of presence: Struttura all'interno della quale gli ISP di livello 2 mettono le proprie strutture

2.1 Ritardo trasmissione commutazione di pacchetto



- Ritardo di elaborazione del pacchetto
  - Router effetta controllo correttezza dei bit
  - Router decide dove inviare i pacchetti

- Ritardo di accodamento
  - Ritardo che subiscono i pacchetti mentre attendono in coda di essere spediti
- Ritardo di trasmisssione
  - $R =$ frequenza di trasmissione in bit/s
  - $L =$ lunghezza del pacchetto in bit
  - Ritardo di trasmissione $= \frac{L}{R}$
- Ritardo di propagazione
  - $d =$ lunghezza del collegamento fisico
  - $s =$ velocità di propagazione del collegamento (m/s)
  - Ritardo di propagazione $= \frac{d}{s}$

Quindi possiamo dire che:

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

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

Un'altro fattore importante è il ritardo di accodamento. dati

- R = frequenza di trasmissione (bit/s)
- L = lunghezza del pacchetto (bit)
- A = tasso medio di arrivo dei pacchetti (pacchetti/sec) ho che:

$$\text{Intensità di traffico} = \frac{L \cdot A}{R}$$

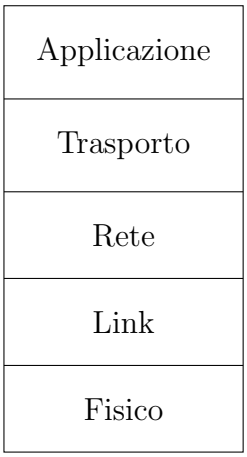
ossia il rapporto fra la "velocità" con cui entrano dati e la velocità con cui escono

2.1.0 Perdita di pacchetti e throughput

- Perdita di pacchetto. Nel momento in cui viene inviato un pacchetto ad un rounter il cui buffer è pieno, questo pacchetto viene scartato e dunque perso
- Throughput. Con throughput si intende "banda", ossia la quantità di bit che possono passare per un determinato collegamento per unità di tempo. Chiaramente vale il principio del collo di bottiglia

2.2 Stratificazione internet

Internet è strutturato in maniera stratificata. Partendo dall'alto livello per arrivare fino al basso abbiamo i seguenti livelli, ciascuno dei quali fornisce determinati servizi:



- Applicazione fornisce "API" ad ogni applicazione abbia bisogno di rete. Protocolli FTP, SMTP, HTTP...
- Trasporto trasferimento dei messaggi a livello di applicazione tra il modulo client e server. Protocolli TCP, UDP
- Rete fornisce servizi che specificano la strada che i bit (*datagrammi*) prendono. Ip, protocolli di instradamento
- Link instrada datagrammi verso determinati commutatori di pacchetto (che li spediscono lungo i cavi)
- Fisico trasmissione di bit su cavi

2.2.0 Perchè la stratificazion?

- Aiuta ad identificare bene i componenti di un sistema complesso e le inter-relazioni
- Facilita la manutenzione e l'aggiornamento del sistema
  - Modifiche ad un livello risultano trasparenti ad altri livelli

3 Livello 7: Applicazione

Fornisce alle applicazioni i mezzi per scambiare dati tramite la rete fornendo servizi quali:

- World-wide web (HTTP)
- Trasmerimento file (FTP)
- Email (POP3, IMAP, SMTP)
- Domain Name System (DNS)
- File sharing peer-to-peer (P2P)
- Terminale virtuale remoto (SSH)

L'unita di dato (*data unit*) che viene trasferito a questo livello si chiama messaggio

Per creare un'applicazione di rete dobbiamo scrivere programmi che:

- Girino su più sistemi
- Cuminichino attraverso la rete

ad esempio, il software di un server web interagisce e comunica con un browser

3.0.0 Architettura client-server

Due componenti: client e server

- Sever
  - Host sempre attivo
  - Indirizzo permanente per accedervi
  - Problemi scalabilità (se è molto utilizzato devo avere hardware potente)
- Client
  - Comunica con il server
  - Può avere un indirizzo dinamico
  - Non parla con altri client in modo diretto, ma con un server

3.0.0 Architettura P2P pura

- Non esiste server, un host, detto *peer*, si connette ad un altro peer in modo diretto
- I peer non sono necessariamente sempre attivi e possono cambiare indirizzo ip

Difficile da gestire gestire. Il server gestisce tutto mentre qui server qualche sistema diverso

- Esiste server centralizzato che gestisce "l'inizializzazione della connessione". Marco viene messo in contatto con Mattia, poi viene stabilito collegamento diretto

3.0.0 Architettura cloud computing

- Permette di elaborare aarchiviare ed elaborare dati su computer distribuiti e virtualizzati in rete
- Sicurezza: i serve sono dislocati, quindi il servizio è garantito ed i dati sono sicuri
- *Server farm*: server dislocati, per avvicinarsi all'utente

3.1 Comunicazione processi

La comunicazione può avvenire fra diversi processi sia all'interno di un computer, sia fra host diversi.

- All'interno del computer vengono utilizzati *schemi interprocesso*
- Nello scambio fra host la comunicazione avviene tramite messaggi

3.1.0 Socket

La socket è la "porta" dalla quale passano i messaggi. Ci si può interfacciare ad un socket tramite api che determinano

- Quale protocollo viene usato per trasportare i messaggi (es tcp)
- Determinati parametri, che vedremo più avanti

3.1.0 Indirizzi

- Ogni host ha un indirizzo IP *univoco* a 32 bit, quindi viene rappresentato così:

  Indirizzo IP: 128.119.245.12

  ossia 4 gruppi da 8 bit, separati da un punto. Ogni gruppo, essendo 8 bit può rappresentare un numero da 0 a 255

- Per identificare un processo bisogna conoscere il numero di porta, dato che su un server possono girare più processi diversi. Questi numeri tendenzialmente sono standardizzati:

  N. porta server HTTP: 80

3.1.0 Requisiti servizi di trasporto diverse applicazioni

In base all'applicazione che dobbiamo implementare, possiamo aver bisogno di diversi requisiti sul trasporto di dati

- Tolleranza alla perdita di dati: posso lasciarmi qualche bit per strada senza compromettere il servizio?
- Throughput: devo garantire una determinata larghezza di banda per portare il servizio?
- Sensibilità al tempo: devo garantire una latenza sufficientemente bassa?

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

3.2 Protocollid i trasporto: TCP, UDP

3.2.0 Transport Control Protocol (TCP)

- Orientato alla connessione richiede processo di connessione fra client e server
- Transporto affidabile
- Controllo di flusso, e della congestione: il processo di invio viene "strozzato" per evitare di sovraccaricare il server
- Non offre: garanzie di banda minima, tempistica e sicurezza

3.2.0 User Datagram Protocol (UDP)

- Trasferimento inaffidabile
- Non offre: setup della connessione e affidabilità. Non offre nemmeno tutto ciò che offre TCP

UDP è un'interfaccia più snella con meno overhead, in quanto non deve implementare controlli sui dati e offrendo meno features è più veloce e più semplice. Spesso utilizzata per servizi di streaming etc etc

Applicazione	Protocollo a livello applicazione	Protocollo di trasporto sottostante
Posta elettronica	SMTP [RFC 2821]	TCP
Accesso a terminali remoti	Telnet [RFC 854]	TCP
Web	HTTP [RFC 2616]	TCP
Trasferimento file	FTP [RFC 959]	TCP
Multimedia in streaming	HTTP (es. YouTube) RTP [RFC 1889]	TCP o UDP
Telefonia Internet	SIP, RTP, proprietario (es. Skype)	Tipicamente UDP

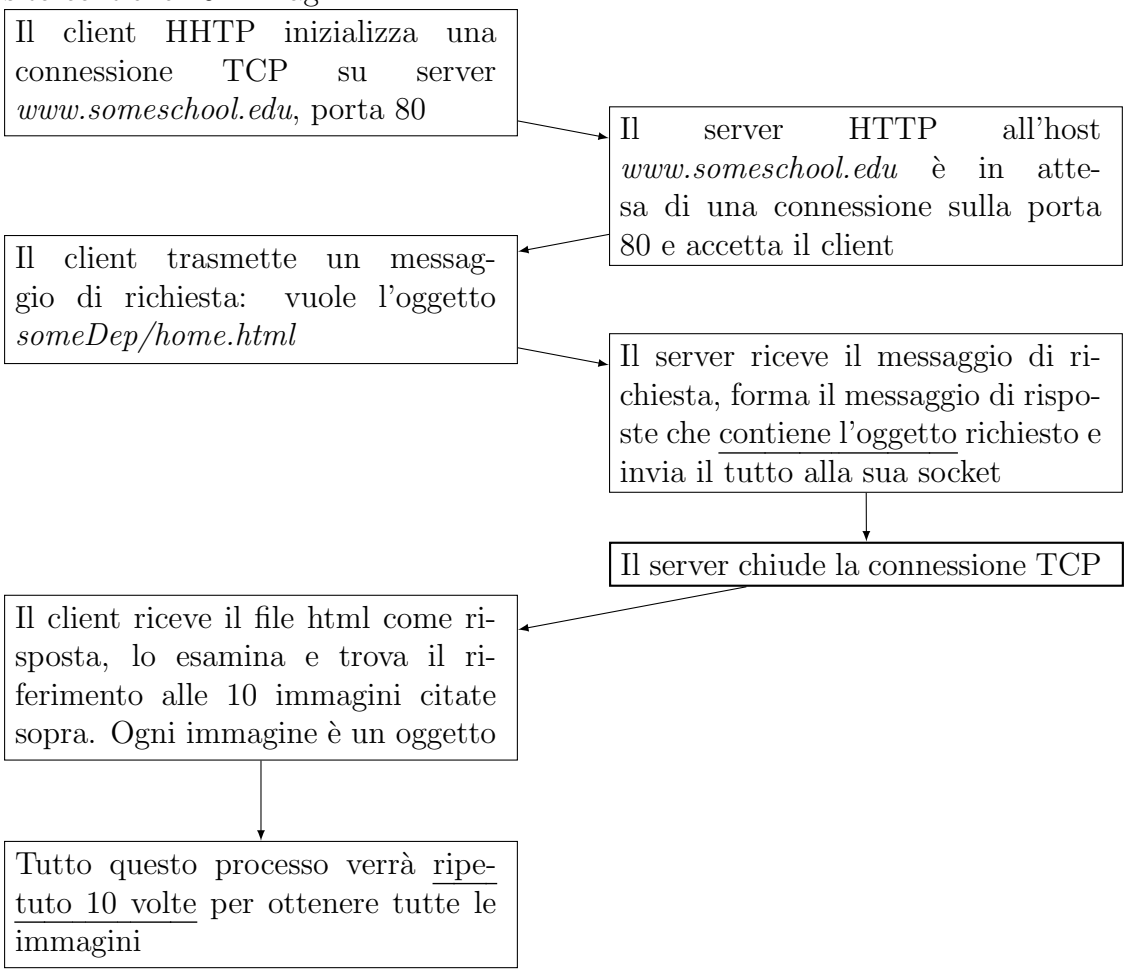
3.3 Protocollo HTTP

- Client inizializza connessione usando TCP
  - Indirizzo ip: *ip server*
  - Porta: *80*, per standard
- SErver accetta connesione
- Scambio *messaggi*
- Chiusura connessione TCP

HTTP è un protocollo stateless, ossia server non memorizza nessuna richiesta del client

3.3.0 Come avviene connessione HTTP - connessione NON persistente

: Supponiamo di volerci connettere a *www.comeschool.edu/someDep/home.html*. Questo sito contiene 10 immagini.



Svantaggio connessioni non persistenti: per ogni oggetto ho bisogno di 2 round trip time (RTT)

- Overhed sistema operativo
- Apertura connessioni TCP in parallelo → grande utilizzo memoria Crhome

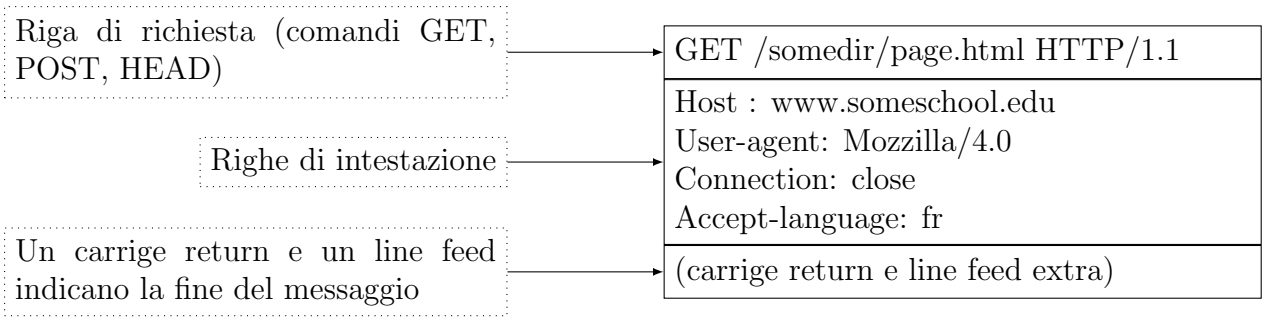
Il problema sta nel fatto che la connessione TCP viene chiusa ogni volta che prendo un oggetto. Qui sta il vantaggio delle connessioni persistenti prova

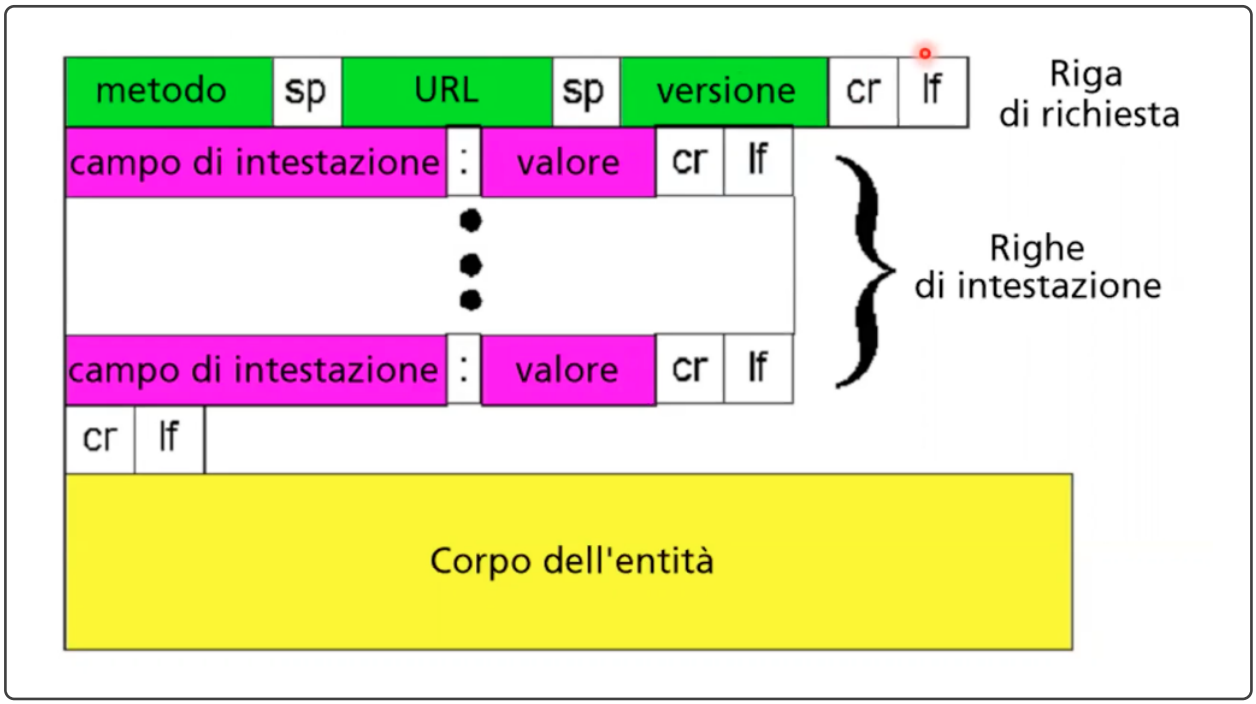
3.3.0 Messaggi HTTP

In HTTP vi sono due tipi di messaggi: richieste e risposte

3.3.0 Messaggio di richiesta

- Messaggi strutturati in maniera testuale (ASCII)

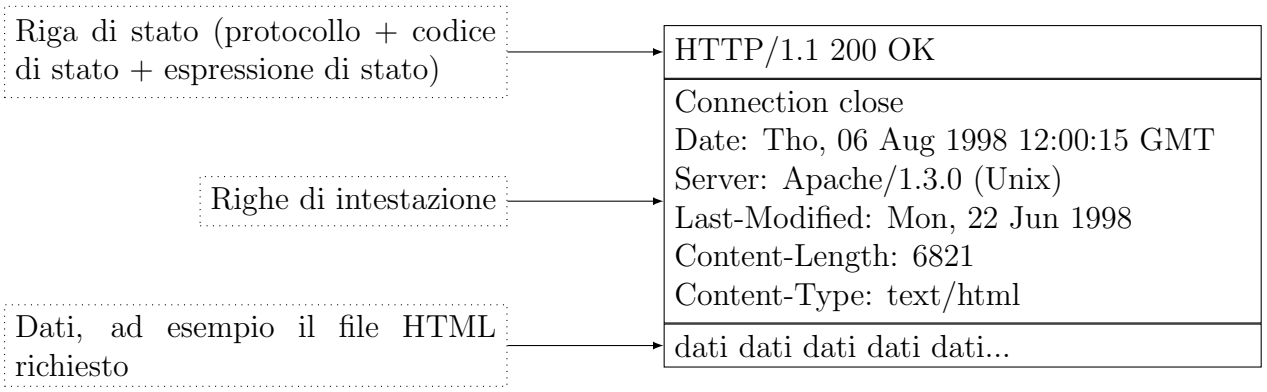




Comandi:

- GET: richiede risorsa dal server tramire URL
- POST: invia risorsa al server (es invio dati di un modula o form)
- HEAD: richiede risorsa al server, ma non non scarica l'oggetto, ma le informazioni che lo descrivono (dati che descrivono la composizione dell'oggetto, ma non l'oggetto stesso)
- PUT: include file nel corpo dell'entità (in giallo) e lo mette all'URL, sempre che vi siano i permessi
- DELETE: elimina il fine all'URL, sempre che vi siano i permessi

3.3.0 Messaggio di risposta



4 Termini

- Directory server: server di un'architettura P2P a cui ogni peer comunica le risorse che ha da offrire e di cui ha bisogno
- Super nodo o ultra peer: nodo in architettura peer to peer che funge da directory server. Determinato dinamicamente da algoritmi
- Architettura P2P ibrida: P2P con super nodi
- Scalabilità verticale o orizzontale: rispettivamente aggiungere più risorse allo stesso server o piùs erver
- Rete single homed-multi homed
- IXP: internet exchange point, serve a far comunicare isp di livello 2 direttamente fra di loro
- CDN enter deep e bring home: la prima installa server negli ISP, la seconda negli IXP
- HTTP: livello applicazione porta 80
- Struttura HTTP:
  - Richiesta : *Metodo - percorso - verione*
    - Metodo: *GET, POST, PUT, DELETE, HEAD, OPTIONS, TRACE*
  - Risposta *Versione - codice stato - Espressione stato*
    - Codice di stato = numero
    - Espressione di stato = descrizione numero
- Binary framing: livello intermedio di HTTP 2.0 che modifica la codifica dei messaggi, rendendolo incompatibile con HTTP 1.0 / HTTP1.1
- FTP: livello applicazioe porta 21 per comandi e 22 per files
- SMTP, POP3, IMAP3. Scambio messaggi fra host online, scarico messaggi senza memoria, riorganizza messaggi creando cartelle con memoria. SMTP ha porta 25

- TLS : protocollo che aggiunge autenticazione e sicurezza sul passaggio di dati da associare a quelli sopra. Numeri di porta differenti da protocolli senza TLS in quanto non posso comunicare poste con TSL con poste senza
- STARTLS: TLS ma su porte standard
- Server DNS : TLD (top level domain (.com)), SLD (second level domain (google)). *Roor server(13)* $\rightarrow$ *TDL server* $\rightarrow$ *Authoritative server* $\rightarrow Server ISP$
- Resolver: parte del OS che risolve nomi host
- RR: Resource record *(name, value, type, ttl)*
- Checksum: sommo parole 16 bit e faccio complemento a 1
- Trasmissione sicura:
  - Stop and wait: mando pacchetto aspetto relativa ack e ripeto
  - Go back n: pipeline con massimo n pacchetti insieme. Quando parte un pacchetto parte un timer. I pacchetti vengono ricevuti in ordine e quelli in disordine vengono scartati. Vengono inviati in ordine fino a quando non scatta timeout. In quel casi vengono rinviati da pacchetto perso
  - Selective repeat
    * Mittente: tiene traccia di quali pacchetti sono stati inviati e di quali è stato ricevuto ack. Invia ogni volta il più basso non inviato. Slitta finestra quando ho ricevuto un ack per i primi k elementi di W_T
    * Ricevitore: tiene traccia dei pacchetti arrivati. Slitta W_R ogni volta che sono arrivati i primi k pacchetti
    * Vale che $|W_T| + |W_R| \leq n_{seq}$ dove n_{seq} è massimo numero di pacchetto (pacchetti numerati da 0 a n_{seq}). Questo perchè nel peggiore dei casi (perdo tutti ack di ritorno), W_T e W_R sarebbero messe così: $\underbrace{1,2,3,4}_{W_T},\underbrace{5,6,7,8}_{W_R},$ quindi se la somma della loro dimensione fosse maggiore di n_{seq}, allora l'id del pacchetto non sarebbe più univoco e rischierei di accettare pacchetti diversi da quelli effettivamente desiderati
- TCP:
  - Init: *SYN*, chiusura: *FIN*, chiusura brusca: *RST*
  - Raggruppa in MSS *maximus segment size*, che dipende da MTU *maximum transmit unit* del livello sottostante
  - RTO: *retransmission time out*

$$\text{EstimatedRTT} = (1-\alpha)\cdot \text{EstimatedRTT} + \alpha \cdot \text{SampleRTT} \quad \text{per } \alpha = 0.125$$

$$\text{DevRTT} = (1-\beta)\cdot\text{DevRTT} + \beta \cdot \mid \text{SampleRTT} - \text{EstimatedRTT} \mid \quad \text{per } \beta = 0.25$$

$$RTO = \text{EstimatedRTT} + 4\ \text{DevRTT}$$

Se invece non vi sono misure:

$$\text{EstimatedRTT} = \text{SampleRTT} \qquad \text{DevRTT} = 1/2\ \text{SampleRTT} \qquad \text{RTO} = 1\ \text{sec}$$

- AIMD:
  - Incremento di 1 MSS finche non perdo pacchetti. Quando perdo, dimezzo CWND
  - Si ottiene fairness: banda tende a ripartizione equa
  - Slow start:
    * Incrementa ogni ack ricevuto di 1 MSS (esponenzialmente)
    * Quando raggiunge SSTHRESH (Slow Start Threshold)passa a congestion avoidance
  - Congestion avoidance
    * Aumeta CWND di $\frac{1}{\text{CWND}}$ segmenti per ogni ack ricevuto
    * Se ricevo tutti ack, allora aumento di 1 segmento ogni RTT

    i
- Fast retransmit e fast recovery
  - Al terzo ack duplicato entro in fast recovery.
    * RESTORE contiene ultimo segmento del quale si è ricevuto ack che dovrebbe essere stato trasmesso prima di entrare in fast recovery
    * Per ogni ack duplicato aumento CWND di 1
    * Appeka arriva un ack superiore o uguale a RESTORE, riprendo a trasmettere con
      · CWND = CWND/2 + 3
      · SSTRESH = CWND / 2
- Alternative a Fast retransmit:
  - CUBIC: simile a fast retransmit su RTT bassi. invece che dimezzare CWND fa CWND = 0.7 CWND
  - BRR: protocollo serer side di google, sfrutta concetto collo di bottiglia
  - QUIC: emula TCP + TLS usando UDP, ma evita di effettuare doppio handshake

4.1 Rete

- Funzione piano dati: decide come fare *forwarding* localmente, ossia come passare un pacchetto da una porta di ingresso ad una di uscita
- Funzioni piano controllo: insieme di algoritmi che determinano percorso da prendere (*routing*)
- *Terminatore di linea - protocollo data link (ethernet) - buffer di inoltro - sistema di commutazione*
- Sistemi di commutazione:
  - A memeoria: leggo e scrivo su memoria, 2 accessi a sistema

  - A bus: uno per volta sul bus condiviso
  - A matrice: più pacchetti alla volta
- HOL: *Head of line block*, quando il nodo non calcola abbastanza in fretta
- Porta di uscita:
  - FIFO or *priority* scheduling
  - Pacchetti scartati con *tail drop - priority drop - random drop*
- Frammentazione
  - Flags M *more fragments* e D *do not fragment*
  - Non usata per ragioni di sicurezza (Attacchi Ddos o overlapping fragments)
- Registrar: autorita che possono assegnare indirizzi ip e nomi id dominio
- Indirizzi ip:
  - *Subnet mask, default gateaway*
  - NAT: apparecchio che associa indirizzo ip privato ad uno pubblico
    * Rompe encapsulation livelli (modifica porte livello trasporto e ip livello rete)
    * Per comunicare a server serve port forwarding
  - Indirizzi:
    * Privati: 10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16
    * Indirizzi di rete: la parte dopo la subnet mask è tutta zero
    * Indirizzo broadcast: la parte dopo la subnet mask è tutta uni
    * Broadcast locale: tutti uni (il pacchetto non va fuori dalla rete locale)
    * Indirizzo "questo computer": tutti zeri
    * Indirizzi loopback: 127.0.0.0/8 per testare app di rete
    * Indirizzi multicast: iniziano con 1110, come broadcast
    * Link local: 169.254.0.0/16, vengono assegnati in modo random a disposivi ai quali non è possibile assegnare ip
- ARP
  - Serve per scoprire un indirizzo MAC (HADDR) associato ad un indirizzo ip (PADDR)
  - I messaggi arp vengono trasmessi in broadcast e solo il dispositivo di interesse risponde
  - Arp caching: le risponde arp vengono salvate 30 sec e sovrascritte dalle più recenti
- ICMP: *internet control message protocol*
  - Serve per recuperare informazioni e segnalare errori
  - Ping: viene sfruttato il meccanismo id *echo reply*

  - Traceroute: vengono mandati pacchetti con TTL incrementali. Ogni volta che il pacchetto scare viene inviata risposta

- DHCP: *dynamic host configuration protocol*
  - DHCP discover
  - DHCP offer
  - Client accetta richiesta
  - DHCP ack
  - Utilizzo indirizzi broadcast 255.255.255.255 perchè client non ha ancora ip
  - Transaction id inclusa in messaggio per sapere a chi è riferito broadcast
  - Usa UDP (connessione impossibile vista assenza di ip)
  - Se non esiste serve DHCP, allora si assegnano indirizzi link-local, e ogni volta si manda messaggio arp. Se nessuno risponde vuol dire che ip non è utilizzato

4.1.0 Protocolli di routing

- OSPF: sfrutta *multicast* 224.0.0.5
  - *Hello* - *exchange* - *flooding* verifica router adiacenti - comunica ai ruoter adiacenti stato rete - comunica a tutti i router ricorsivamente stato rete
  - Dijkstra
  - Gerarchia: rete *dorsale* e *reti di area*
- Dijkstra - *link state*
- Bellman ford - *distance vector*
- Count to infinity: quando un collegamento viene modificato si viene fatti rimbalzare fra due nodi all'infinito. Soluzioni:
  - Settare *max hop* (di solito 15)
  - Split horizon - X non manda a Y le rotte apprese da Y stesso
  - Poison reverse - se X passa da Y per arrivare a Z, allora X informa Y che la sua distanza da Z è ∞
- RIP: *UDP, porta 520, muticast su 224.0.0.9, processo routed, usa poisoned reverse*, simile a Distance vector. Quando un nodo non riceve aggiornamenti per 180 il link si da per rotto

4.2 BGP

- *BPG speakers* si inviano *path vectors* con destinazione, next hop e path
- Connessione client-provider, viceversa o peer to peer
- *Ingress policies* - *egress policies*, filtrano in entrata e uscita. I persorsi accettati sono salvati rispettivamente in tabelle ADJ_RIB_IN e ADJ_RIB_OUT

- Messaggi Open, Notification, KeepAlive e Update.
  - Open apre connessione. Connessione è effettivamente aperta dopo ever ricevuto keepalive
  - Notification: informa BPG speaker di un errore
  - Update: *Addittive o sottrattive (withdraw)*: aggiungono o rimuovono rotte

4.3 Livello data link

- Servizi di *consegna affidabile - controllo flusso - rilevamento e correzione degli errori*
- Collegamento *half duplex o full duplex*, rispettivamente uni/bi direzionale
- Correzione errori:
  - Bit di parità unidirezionale: sgama errori, bidirezionale permette di correggerli
  - Ridondanza e interleaving: ogni bit viene ridondato e riordinato in maniera casuale. All'arrivo i bit vengono rimessi nell'ordine originale e per ogni gruppo si sceglie il bit ripetuto di più
  -
- TDMA, FDMA, CDMA: in quest'ultimo viene assegnato un *chip*, e viene inviato lo xor fra dati e chip. Al destinatario viene ancora fatto xor. Se il dato non corrisponde allo xor giusto si ottiene solo rumore
- Slotted ALOHA: ogni trasmettitore è sincronizzato e può trasmettere solo 1 alla volta. Avendo N trasmettitori il numero massimo di pacchetti è

  $$N * p\,(p-1)^{N-1}$$

  il valore di p che massimizza l'espressione sopra è $\frac{1}{n}$ e il limite per $N \to \infty$ è :

  $$\lim_{n\to\infty} \left(1 - \frac{1}{N}\right)^{N-1} = \frac{1}{e} \approx 0.36$$

  Se ALOHA non fosse slotted, la trasmissione massima sarebbe circa $\frac{1}{2e} \approx 0.18$, ossia la metà dello slotted
- CSMA p-persistente (*carrier sense multiple acces*): quando canale occupato aspetta fino a quando non è libero e poi con probabilità p trasmette segnale
  - Finestra di vulnerabilità: periodo in cui non è certo che nodo sia libero o occupato. Tempo di propagazione + tempo necessario per rilevare collisione
  - Efficace quando tempo di propagazione è molto inferiore a tempo trasmissione
  - In reti cablate: CSMA/CD quando rilevo collisione invio subito segnale
  - In reti wireless: CSMA/CA: si procede a tentoni, riducendo p(probabilità collisione) in base a proprietà rete
- Protocolli a turni:

- Polling: *master* invia poll a *slave* e slave risponde con dati. Problema SPOF e inefficiente
- Token: onferisce abilità di trasmettere e viene passato fra host. Stessi problemi polling
- Ethernet:
  * Inizialmente *bus* e *vampirizzazione* con *terminatori*
  * Poi *HUB* con *switch*
  * Doppino incrociato con connettore *rj45*
  * No connesso no sicuro no ack
  * Protocollo MAC CSMA/CD unslotted
  * Ogni collisione si sceglie valore di backof fra 0 e $\left(2^k - 1\right) T$ con $k \leq 7$ e T = tempo per trasmettere 512 bit

◦ Domini:

- Di collisione: hub →1, switch → tanti quante le porte
- Di broadcast: mappatura 1:1 con domini di collisione

◦ Switch:

- *backward learning*, aggiorna route corrispondente a mac sorgente
- Se pacchetto ha dest === src viene scartato
- Problema dei loop, finisce in flooding infinito, risolto da STP

◦ STP: *spanning tree protocol*

- Struttura logica è spanning tree → no cicli. Struttura fisica ha loops (ridondata)
- Ogni switch è identificato da 16 bits (impostati da amministratore rete) + 48 bits MAC address
- Pacchetti di controllo *BPDU*
- Ogni porta può essere: *bloccata, designata, radice*

◦ LAN: per performance o separazione degli intenti

- Impossibile identificare gruppi → introduzione nuovo protocollo ethernet
- Campo *VLAN identifier* in header
- Se host non supporta vlan no problem: switch manipola headers
- Se switch non supporta vlan problemi: o scarta frame, o inoltra senza considerare vlan

◦ Reti wireless:

- Ad hoc (tipo bluetooth) o a infrastruttura (tipo wifi)
- *(ESS (BSS AP)(BSS AP))*
  * WLAN: *Wireless Local Area Network*

    - Extended service point
    - Basic service point
    - Acces point
- Standard wifi
    - 2.4 ghz divisi in 11 canali
    - Associazione passiva o attiva
    - Invio *beacon* con SSID, indirizzo MAC
    - Assegnazione ip sottorete tramite DHCP
    - Attenuazione del segnale viaggia come r^2
    - Non può ricevere e inviare contemporaneamente (autointerferenze)