Reti di Elaboratori

Marco Casu

Contents

1 Reti di Calcolatori e Internet

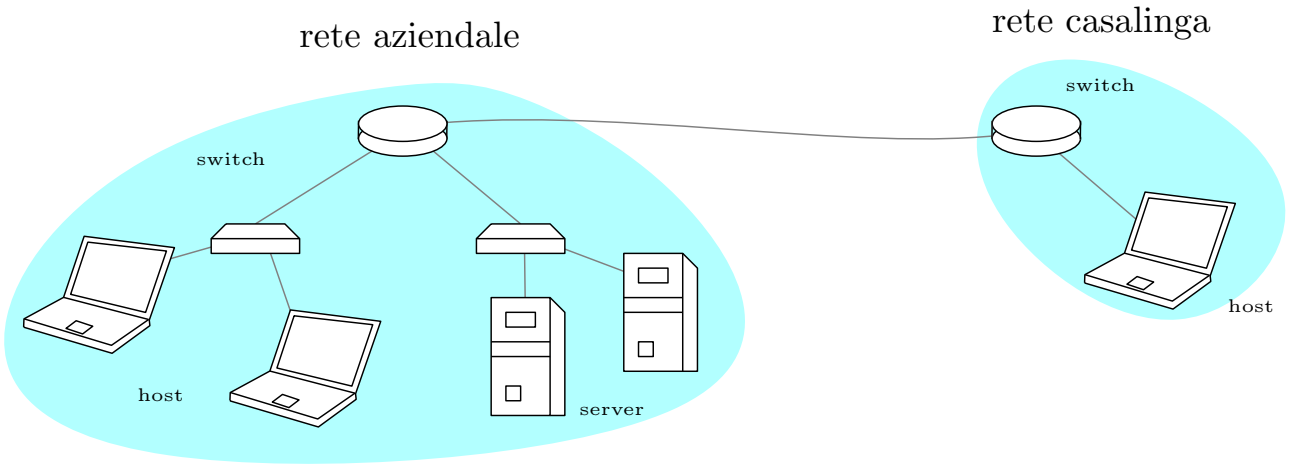
Cos'è una richiesta di rete? E soprattutto quali sono i passaggi ed il procedimento scaturito a seguito di una richiesta? Questo corso si concentrerà sull'aspetto del *networking*, ossia, su come avviene la comunicazione tramite più elaboratori.

Una *connessione* è una comunicazione aperta in cui sono coinvolte entrambe le parti in attesa di ricevere ed inviare messaggi, tramite l'apertura di un *socket* (si approfondirà in seguito). Il problema di una comunicazione di questo tipo, è il bisogno di avere la certezza che i messaggi inviati da una parte siano ricevuti correttamente dall'altra, senza il rischio di comunicare "a vuoto", per questo sono definiti degli appositi protocolli.

Host : Un dispositivo connesso alla rete in modo periferico, non funge da esclusivo tramite per la comunicazione, ed è un sistema "periferico", esegue delle *app* che forniscono servizi sulla rete.

Switch : Gli switch sono i dispositivi capaci di "instradare" i *pacchetti*.

Rete : Una collezione di dispositivi host/switch e collegamenti gestiti da un unico ente/organizzazione.



Quella che noi chiamiamo **internet**, è una "*rete di reti*", ossia l'insieme interconnesso di tutte le reti pubbliche, che si stabilisce e necessita di protocolli su tutti i livelli :

- Livello di applicazione
- Livello di trasporto
- Livello network
- Livello di collegamento

Lo scopo di internet è quello di essere un infrastruttura che fornisce i servizi alle applicazioni distribuite, è un interfaccia di programmazione e fornisce un servizio di *trasporto dei dati*.

Un **protocollo di rete**, stabilisce delle regole riguardanti lo scambio di messaggi, con le relative "azioni" specifiche da intraprendere per la ricezione di messaggi ed eventi, definiscono il *formato* e *l'ordine* dei messaggi da inviare fra le entità di rete, e le azioni intraprese sulla ricezione e trasmissione dei messaggi.

1.1 Struttura di Internet e Link

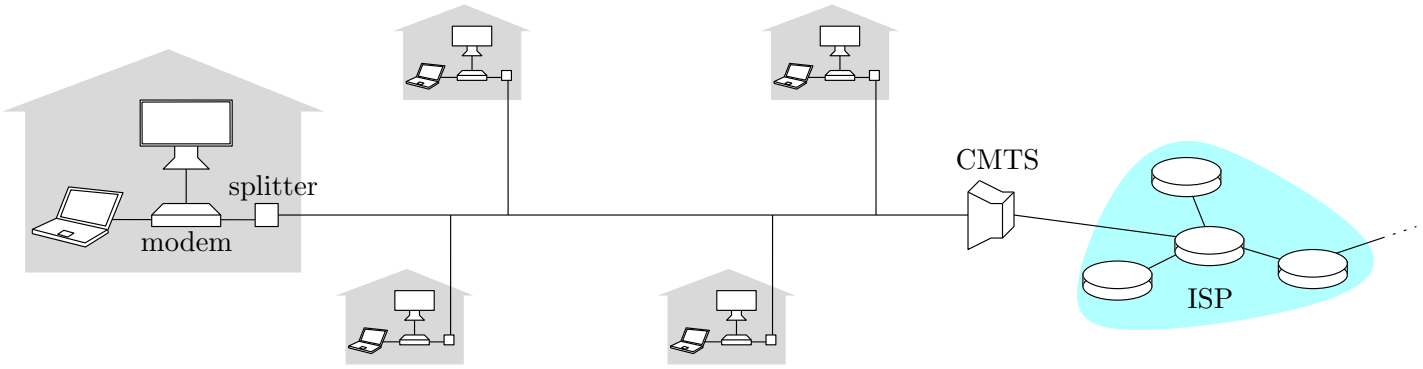
Una rete è quindi un insieme di nodi collegati tramite dei *link*, composta da dispositivi di interconnessione, che si scambiano informazioni, usualmente utilizziamo il termine *host* per i dispositivi che usufruiscono di un servizio, e *server* per i dispositivi che lo erogano.

I dispositivi di interconnessione, ricevono un segnale, lo modificano e lo ritrasmettono, sono i *router* (collegano una rete ad altre reti) e gli *switch* (collegano più dispositivi ad una rete locale). I collegamenti, o link, possono essere cablati (rame o fibra ottica) oppure wireless, senza cavi (onde elettromagnetiche).

Le reti locali come quelle casalinghe o aziendali, si collegano ad una rete regionale ISP (internet service provider) di router interconnessi detta *core* o *backbone*, che a sua volta si collega ad una simile struttura ma a livello nazionale o globale.

1.1.1 Reti Cablate e Wireless

Nell'accesso via cavo, c'è un terminale comune per più abitazioni detto **CMTS**, ossia *Cable Modem Termination System*, che viene poi diramato nelle diverse abitazioni, i dati dalla rete ed i segnali per la televisione sono trasmessi sul medesimo cavo ma a frequenze differenti. Il CMTS è direttamente collegato ad un ISP. In questo modello diverse case condividono la rete di accesso.



La **DSL** diversamente, utilizza la linea telefonica esistente, collegandosi alla DSLAM, i dati sulla linea DSL vanno su internet, la voce su DSL va sulla linea telefonica.

Una rete *domestica* c'è un *modem* (si occupa di ricevere il segnale analogico e di *modularlo* in segnale digitale) connesso ad un CMTS, collegato via cavo ad un router, collegato direttamente tramite *ethernet* agli host, oppure collegato ad un *access point WI-FI* che permette la connessione senza fili, questi utlimi tre molto spesso sono combinati in un unico device.

La connessione senza fili, o wireless connette i sistemi terminali (host) al router, tramite il già citato access point, esistono reti locali senza fili (WLAN), che hanno una copertura di circa 30 metri, e reti di accesso cellulare *wide area*, utilizzate dai dispositivi mobili e fornite da un operatore di rete cellulare, ed hanno una copertura più ampia, nell'ordine dei kilometri.

Le reti aziendali, come quelle delle università sono di una scala diverso rispetto quelle domestiche, prevedono molti più dispositivi, ed un mix di varie tecnologie di collegamento, cablate o wireless, tramite svariati switch e router.

1.1.2 Comunicazione e Classificazione delle Reti

Un host comunica tramite l'invio e la ricezioni di messaggi da un applicazione sottoforma di *pacchetti* di dati, ossia messaggi suddivisi in blocchi più piccoli, di lunghezza L bit. L'host trasmettono i pacchetti nella rete ad una *velocità di trasmissione* di R $^{bit}/_{sec}$. Il *ritardo* di trasmissione del pacchetto è il tempo necessario per trasmettere il pacchetto (L bit) nel collegamento, ed equivale a $\frac{L}{R}$ secondi.

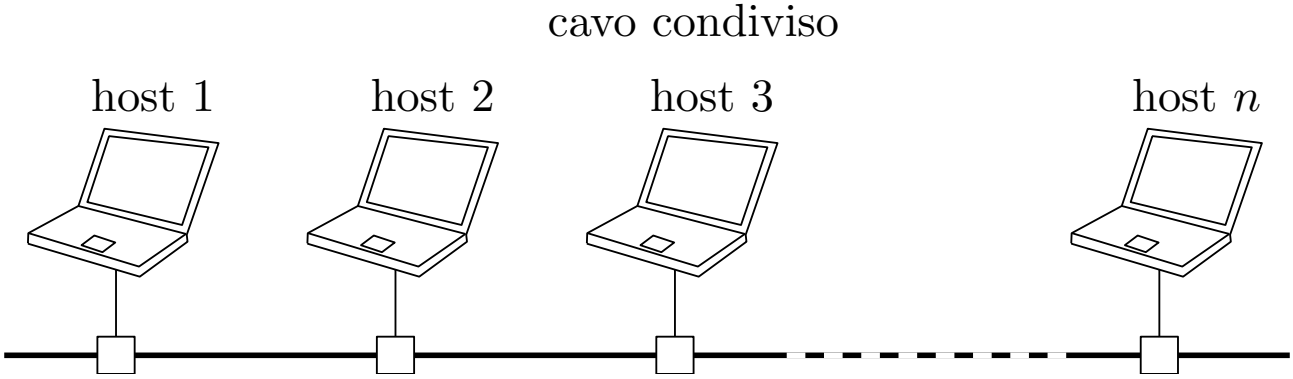
Il segnale si propaga fra trasmettitore e ricevitore, tramite *supporti guidati*, ossia i mezzi solidi (cavi), oppure *non guidati*, propagandosi liberamente nell'aria (onde elettromagnetiche).

- Il **cavo coassiale** è provvisto di due conduttori di rame concentrici, è bidirezionale, supporta più canali date diverse frequenze, ed è resistente alle interferenze.
- La **fibra ottica** ha soppiantato il cavo coassiale, è una fibra di vetro che trasporta impulsi luminosi, dove ciascun impulso rappresenta un bit, la luce rimbalza nel cavo muovendosi ad alta velocità, è necessario però considerare dei ripetitori (anche se molto distanziati), dato che la luce rimbalzando potrebbe tendere a disperdersi causando una perdita di informazioni.
- La rete **wireless** non ha un supporto guidato, il segnale è propagato nell'aria, è quindi *broadcast*, chiunque può ricevere il segnale. Tale metodo di comunicazione è *half-duplex*, ossia, la comunicazione avviene da un mittente ad un destinatario, e non è possibile comunicare fra due enti contemporaneamente, è soggetta ad effetti dovuti all'ambiente di propagazione, come interferenze, riflessione e ostruzione da parte di oggetti fisici.

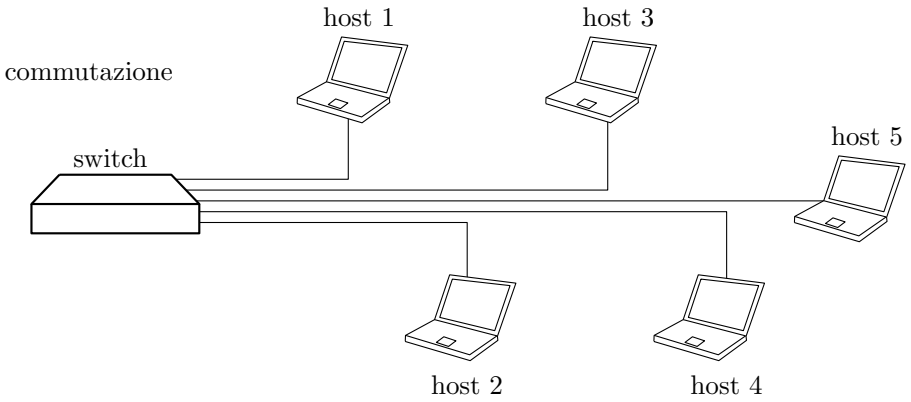
Esiste una scala di classificazione delle reti :

Scala	Tipo	Nome completo	Esempio
Distanza ravvicinata	PAN	Personal Area Network	Bluetooth
Edificio	LAN	Local Area Network	WiFi, Ethernet
Città	MAN	Metropolitan Area Network	Cablata, DSL
Paese	WAN	Wide Area Network	Grandi ISP
Pianeta	Internet	La rete di tutte le reti	L'Internet

La **LAN** è la rete locale, come una rete domestica, è una rete privata ed ogni terminale connesso ad essa è identificato da un indirizzo distinto dagli altri, può essere a *cavo condiviso* oppure a *commutazione* con uno switch.



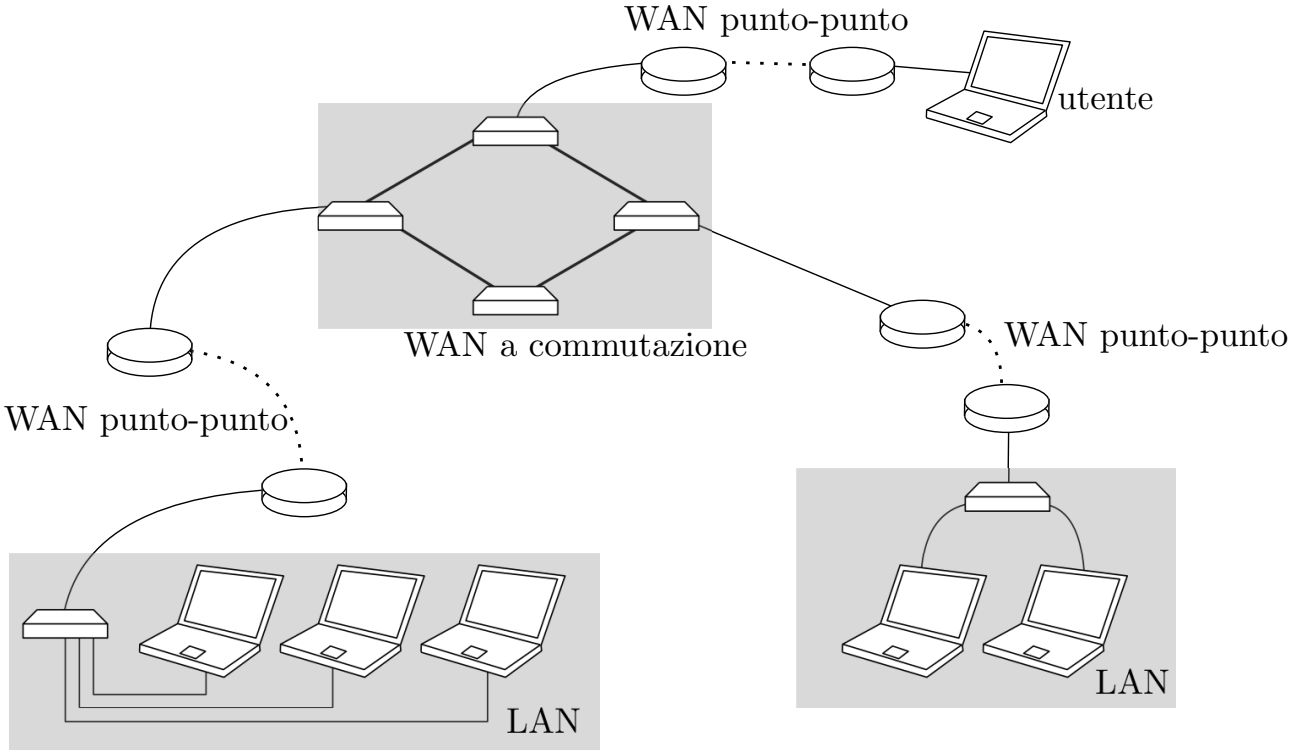
In tale modello di cavo condiviso il pacchetto inviato ad un dispositivo viene ricevuto da tutti, solo il destinatario lo elaborerà, tutti i restanti host lo ignoreranno.



Quest'ultimo a commutazione è il più utilizzato tutt'oggi, ogni dispositivo è direttamente collegato allo switch, ed esso è in grado di riconoscere gli host ed inviare i pacchetti esclusivamente al destinatario, riduce il traffico nella LAN.

Le reti **WAN** sono reti geografica, vengono interconnessi dispositivi di comunicazione, necessari a città, regioni o perfino nazioni. I dispositivi in questione sono switch, router e modem, tale rete è gestita da un grande operatore/ente di telecomunicazioni detto IPS (Internet Service Provider) che fornisce i servizi alle organizzazioni.

Una WAN può vedere i suoi dispositivi di comunicazione connessi punto-punto, oppure a commutazione, con più punti di terminazione (usata nelle dorsali di Internet), tutt'oggi è raro trovare LAN o WAN isolate, spesso sono connesse fra loro per formare una internetwork (internet), per mettere in comunicazione due LAN in città differenti tramite una WAN.



1.1.3 Nucleo della Rete

Si definisce nucleo della rete, quella parte composta esclusivamente da router che effettuano *commutazione* di pacchetto, ossia ricevono il pacchetto, e lo re-indirizzano tramite i collegamenti. Ogni router presenta più porte, si occupa di fare il cosiddetto **forwarding**, ossia l'azione locale di spostare i pacchetti in arrivo dal collegamento in ingresso, al collegamento in uscita, è un azione locale del router.

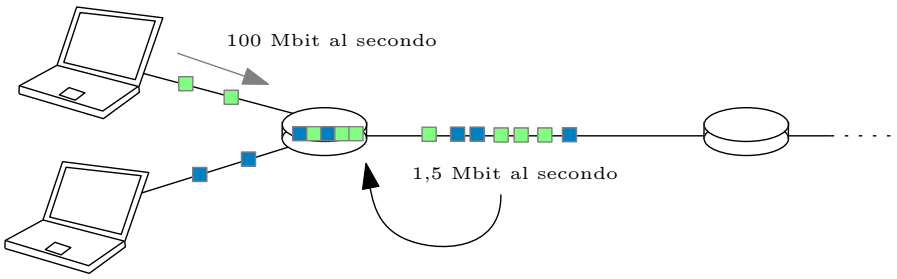
Il **routing** invece è un azione globale, si intende la determinazione dei percorsi origine-destinazione che i pacchetti dovranno prendere all'interno dell'Internet, tale decisione è presa tramite degli algoritmi di instradamento.

Con il termine *Trasmissione*, si intende l'azione che intraprende un pacchetto per essere trasferito interamente sul collegamento. Tale termine non include l'intero tragitto fino a destinazione, ma esclusivamente la (appunto) trasmissione sull'eventuale cavo, il *ritardo di trasmissione* è quindi il delay misurato in secondi per trasmettere un pacchetto, si è già specificato che tale valore è uguale a $\frac{L}{R} = \frac{\text{bit di un pacchetto}}{\text{bit per secondo}}$.

I router funzionano nella seguente maniera, detta **store and forward** : Un pacchetto deve arrivare per intero ad un router prima di essere ri-trasmesso su un nuovo collegamento.

> *Esempio* : Si devono trasferire, dal collegamento A al collegamento B, 3 pacchetti da 10 Kbit ad una velocità di 100 Mbitt per secondo, si assume che il tempo che impiega un bit per propagarsi nel collegamento è zero. In totale, il ritardo di trasmissione sarà : $\frac{10 * 3 * 10^3}{100 * 10^6} = \frac{3}{100 * 10^2} = \frac{3}{10^4} = 0,0003 \text{ sec } = 0,3 \text{ msec}$

Un problema noto durante la tramissione dei pacchetti è l'**accodamento**, avviene quando la velocità di arrivo ad un router da parte di un link è maggiore della velocità di trasmissione in uscità (dal link al router), quindi si causa un accodamento in attesa della trasmissione.



I pacchetti arrivano al router prima di essere trasmessi sul link, quindi si accoderanno in una memoria interna del router prima di essere ritrasmessi, se i pacchetti accodati diventano troppi e la memoria viene esaurita, ci sarà una *perdita* di pacchetti.

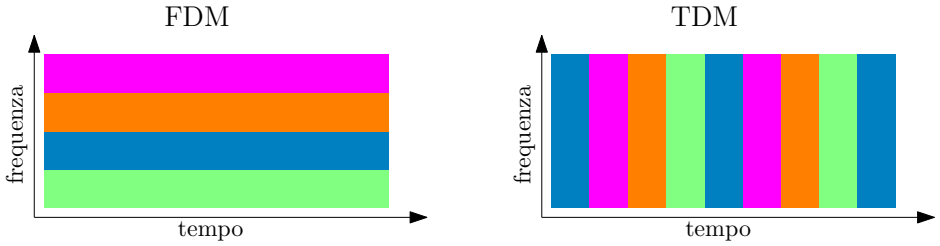
Una possibile soluzione alla perdita è la **commutazione di circuito**, ossia, far si che i diversi host non condividano lo stesso collegamento per i pacchetti, bensì, si considerano dei canali riservati per la comunicazione tra sorgente e destinazione.

Non c'è condivisione, ci saranno quindi svariati collegamenti, in numero necessario per permettere a tutti i dispositivi di poter comunicare in modo libero, ovviamente non tutti comunicano nello stesso momento, quindi i un segmento di cirucito potrebbe rimanere inutilizzato.

Un'altra alternativa è quella di condividere lo stesso circuito, riservando ad ogni comunicatore una frequenza (*FDM*), oppure un certo quanto temporale (*TDM*).

- Nel FDM, le frequenze elettromagnetiche o ottiche sono divise in bande, ogni utente può comunicare in maniera continua, ma la velocità massima di comunicazione è data dalla larghezza della banda, che è stretta in quanto condivisa.
- Nel TDM, ogni utente, a turno in un attesa circolare, può comunicare per un quanto di tempo determinato, in cui ha a disposizione la velocità massima della banda.

4 diversi comunicatori sono denotati con 4 colori differenti



Il problema è che la commutazione di circuito risulta comunque in efficiente, in quanto permette ad un numero limitato di utenti di usufruire della rete. Si preferisce quindi la condivisione delle risorse, in quanto l'accodamento e la significativa perdita di pacchetto, incombe quando un elevato numero di utenti sta usufruendo della rete.

Il fatto è che gli utenti non comunicano il 100% del tempo, supponiamo che un utente stia comunicando con probabilità p, in un sistema con n utenti. Vi è una significativa perdita di pacchetto quando k utenti condividono le risorse, qual'è la probabilità che k utenti comunichino contemporaneamente ?

Sia X la variabile aleatoria che indica il numero di utenti che comunica contemporaneamente, X è una variabile aleatoria binomiale, e la sua distribuzione vale $\mathbb{P}(X \geq k) = \sum_{i=k}^{n} \binom{n}{i} p^i (1-p)^{n-i}$.

> *Esempio* : In un sistema con 35 utenti, ogni utente comunica con probabilità uguale a 0.1. Vi è una significativa perdita di pacchetto quando 10 utenti comunicano contemporaneamente, qual'è la probabilità che ciò accada?
>
> $$\sum_{i=10}^{35} \binom{35}{i} (0.1)^i (0.9)^{35-i} \simeq 0.001$$

La condivisione del circuito risulta quindi la scelta migliore, anche se la possibile congestione in casi particolari è eccessiva, e risulta inevitabilmente una perdita di pacchetti.

1.1.4 Internet

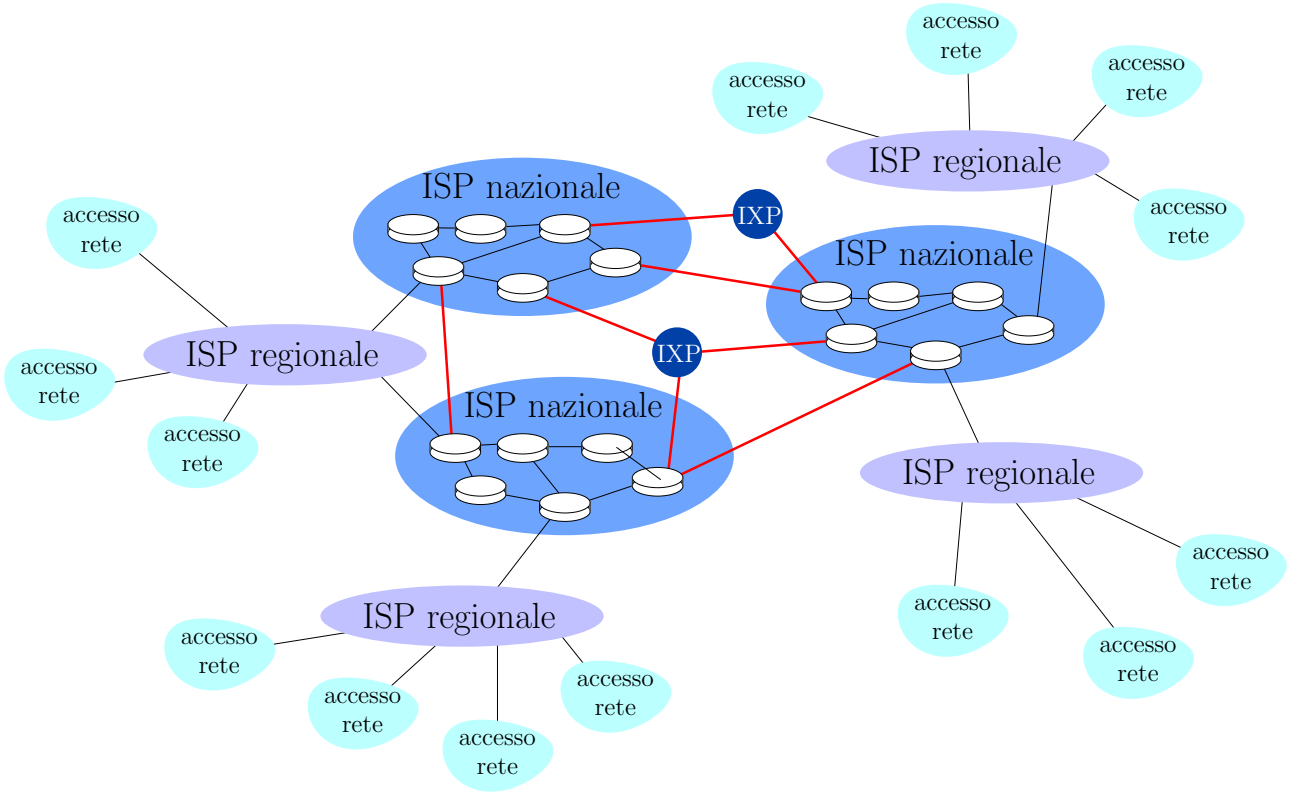
Fino ad ora, abbiamo utilizzato il termine Internet (con la iniziale maiuscola) ed internet (con la iniziale minuscola), è necessario dare una definizione più rigorosa di Internet.

Abbiamo visto come gli host si connettono ad Internet tramite l'accesso agli ISP, residenziali o

aziendali. I grandi ISP sono fra loro interconnessi, da qui il nome internet (inter network), in tal modo, tutti gli host connessi ai differenti ISP possono scambiarsi pacchetti.

Ogni host connesso ad Internet, è quindi connesso ad un ISP, ed i grandi ISP sono tutti interconnessi fra loro, creando un unica grande rete globale, denominata appunto, Internet (con la iniziale maiuscola), ossia una rete delle reti.

Tale rete globale è complessa, e la sua evoluzione, nella struttura, è anche derivata da dinamice politiche ed economiche a livello nazionale, si osservi la seguente immagine.



Le icone con scritto *accesso rete* rappresentano i punti in cui un host può connettersi, ad esempio una LAN, tali punti si collegano agli ISP regionali, che a loro volte si collegano ad ISP nazionali, spesso condiviso anche da più nazioni, spesso interconnessi da degli *Internet Exchange Point* (ISP), gestiti da più enti in comune accordo.

Quindi una internet è una rete costituita da più reti interconnesse, Internet invece, è la "internet" più grande e famose, composta da migliaia di reti interconnesse.

1.2 Prestazioni della Rete

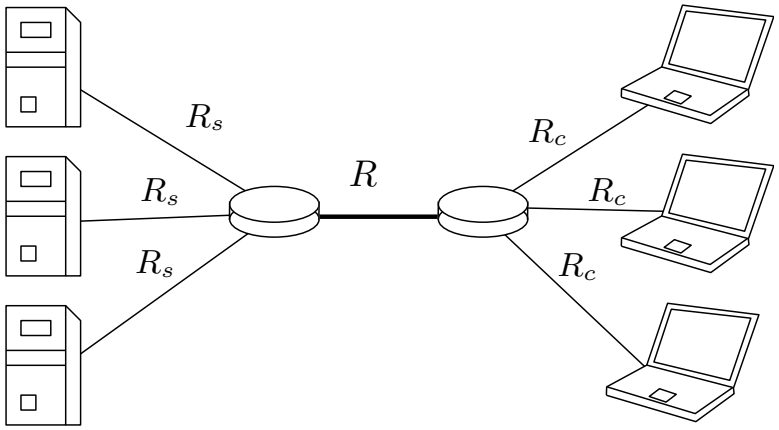
Utilizziamo il termine **ampiezza di banda** per intendere due concetti differenti, ma collegati:

- Caratterizzazione del canale di trasmissione dei dati - quantità che si misura in *hertz*, rappresenta la larghezza dell'intervallo di frequenze utilizzato dal sistema trasmissivo, ovvero l'intervallo di frequenze che un mezzo fisico consente di trasmettere senza danneggiare il segnale in maniera irrecuperabile. Maggiore è l'ampiezza di banda, maggiore è la quantità di informazione che può essere veicolata attraverso il mezzo trasmissivo.

- Caratterizzazione di un collegamento - rappresenta i bit al secondo che possono essere trasmessi in un canale di trasmissione, tale grandezza viene denotata *bit rate*.

Il bit rate dipende dalla banda e dal canale di trasmissione, è proporzionale alla banda in hertz, tale rate descrive la capacità indicativa (o potenziale), non l'effettivo numero di bit trasferiti per unità di tempo, quest'ultimo dato è detto *troughput*, ed è il numero effettivo di bit al secondo che passano attraverso un punto della rete. Il troughput è limitato dal bitrate.

Se un pacchetto deve passare per due o più link con bit rate differenti, il link con il bit rate minore condizionerà il troughput medio dell'intero tragitto, causando un collo di bottigllia. Si consideri adesso la seguente situazione :



- R_s è il bitrate dei link che collegano i server al router di sinistra.
- R_c è il bitrate dei link che collegano gli host al router di destra.
- R è il bitrate del link che collega i due router.

Il troughput medio come sarà condizionato? I server e gli host hanno un collegamento riservato, i due router invece, hanno un link unico per far comunicare i pacchetti di tutti i dispositivi periferici (in totale 6), il collo di bottiglia sarà causato dal collegamento che ha il bit rate minimo fra : (il link server-router, il link host-router, il link router-router condiviso da 6 differenti dispositvi), quindi si avrà : $\min(R_s, R_c, {}^{R}/_{n})$, dove n è il numero di dispositivi (in questo caso 6).

1.2.1 Latenza e Perdita di Pacchetti

Abbiamo visto come i pacchetti si accodano nella memoria di un router, i delay in una comunicazione di un bit che passa su un determinato nodo della rete sono i seguenti:

- d_{proc} - Elaborazione del nodo, controllo di possibili errori sul bit, solitamente ininfluente.
- d_{coda} - Tempo che il bit di un pacchetto trascorre in attesa nella coda di un router.
- d_{trans} - Ritardo di trasmissione già visto in precedenza.
- d_{prop} - Il tempo di propagazione del bit attraverso il link (cablato o wireless).

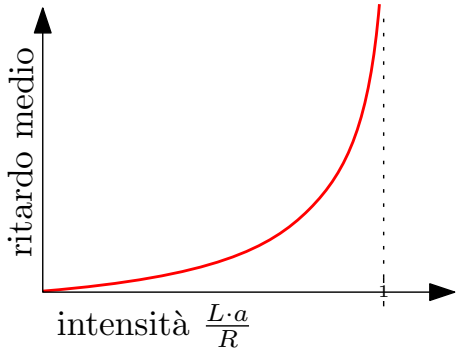
Abbiamo visto come il d_{trans} si misura in secondi tramite la formula $\frac{L}{R} = \frac{\text{bit di un pacchetto}}{\text{bit per secondo}}$, il ritardo di propagazione, ossia il d_{prop}, si misura con la formula $\frac{k}{v}$, dove k è la lunghezza in metri del collegamento fisico, e v la velocità di propagazione attraverso il collegamento (ad esempio, la luce nella fibra ottica si propaga a circa 300 000 $^{km}/_{s}$).

1.2.2 Ritardo di Accodamento

Calcolare il ritardo di accodamento è piuttosto difficile in quanto ci sono innumerevoli fattori in gioco da considerare, è possibile fare delle stime, consideriamo i seguenti parametri:

- L - lunghezza di un pacchetto in bit.
- R - velocità di trasmissione in bit al secondo.
- a - tasso medio di arrivo di pacchetti, misurato in pacchetti al secondo.

L'*intensità del traffico* è una misura adimensionale data da $\frac{L \cdot a}{R}$ ed è limitata dall'unità di tempo (in questo caso, 1 secondo).



- Se $\frac{La}{R}$ tende ad 1, il ritardo medio tende a crescere in maniera esponenziale
- Se $\frac{La}{R} \geq 1$ il ritardo medio è infinito

Come si calcola però l'effettivo ritardo di accodamento nei casi reali? Esistono dei software diagnostici come *tracerout*, che si occupano di misurare il ritardo che impiega un pacchetto per spostarsi dalla sorgente ad un nodo della rete.

Tali software fanno uso di una proprietà dei pacchetti, è possibile impostare per ogni pacchetto un "tempo di vita", ossia un numero massimo di nodi della rete (router) nella quale possono passare, quando tale limite viene superato, il pacchetto verrà automaticamente scartato.

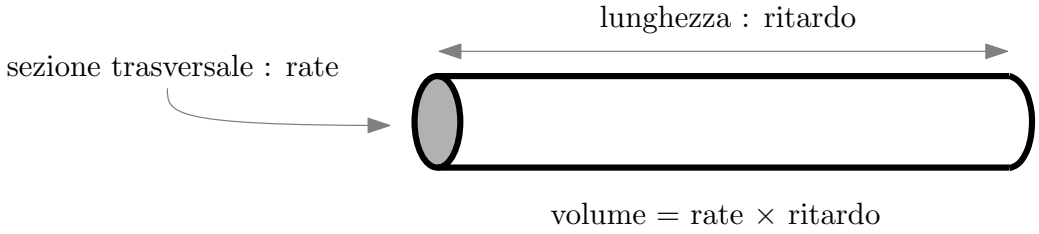
La maggiorparte dei router, quando vedono un pacchetto venire eliminato, mandano indietro al destinatario un messaggio di avvertimento, per indicare appunto che il pacchetto è stato perso.

Grazie alla combinazione di questi due meccanismi, traceroute invia dei pacchetti con un determinato tempo di vita, per poi aspettarsi un messaggio di ritorno, misurando il tempo che intercorre fra l'invio del pacchetto ad un determinato router, ed il messaggio di ritorno che avvisa dell'eliminazione del pacchetto.

Un dato importante da considerare è il **prodotto rate×ritardo** ossia il prodotto fra il bit rate ed il ritardo di propagazione di un certo collegamento. Supponiamo di avere un link con rate di R bit al secondo ed un ritardo di propagazione di x secondi, si avrà che $R \cdot x$, non sarà altro

che il *massimo numero di bit* che possono passare contemporaneamente sul collegamento.

Possiamo pensare al collegamento come un tubo che passa fra due punti, il ritardo rappresenta la lunghezza del tubo, ed il rate la sezione trasfersale, il volume del tubo è appunto tale prodotto rate×ritardo.



1.3 Introduzione ai Protocolli

Le reti sono piuttosto complesse, i protocolli sono un tentativo di rendere la struttura più organizzata, definiscono delle regole che un mittente ed un destinatario devono rispettare per comunicare, insieme al formato del messaggio.

La comunicazione in rete è suddivisa su più livelli, per questo si dice che i protocolli sono definiti a strati (verrà chiarito il concetto), con lo scopo di suddividere un compito complesso in più compiti semplici tramite la modularizzazione.

Possiamo vedere ogni livello come una *black box*, in cui un messaggio entra e viene manipolato per poi uscire e passare al livello sccessivo. Ogni livello offre utilizza i servizi del livello inferiore, e fornisce servizi al livello superiore, indipendentemente da coma sia implementato.

Quando la comunicazione è bidirezionale, un protocollo deve poter eseguire i due compiti inversi (ad esempio, il protocollo che si occupa della crittografia, deve saper cifrare il messaggio ed anche decifrarlo).

La stratificazione dei livelli e dei protocolli garantisce più semplicitià quando si tratta di manutenere o aggiornare il sistema, aumenta la riusabilità e l'eterogeneità, comporta però anche degli svantaggi, come la ridondanza delle operazione ed un calo dell'efficienza.

1.3.1 Layer di Protocollo

I 5 macro-livelli sulla quale si fonda la comunicazione sono i seguenti :

applicazione	supporto delle applicazioni di rete
trasporto	trasferimento dati fra processi
rete	instradamento dei pacchetti dalla sorgente alla destinazione
collegamento	trasferimento di dati tra elementi di rete vicini
fisico	bit sul canale fisico (cavo o wireless)

Sino ad ora i dati incapsulati che vengono comunicati sulla rete sono stati chiamati generalmente "pacchetti", vedremo che questi assumono una denominazione diversa per ogni livello. Ogni

protocollo fa parte di un livello, ed anche se esistono più protocolli per un livello, ogni pacchetto che viene trasmesso usufruisce di un solo protocollo per livello. I nomi dei protocolli citati in seguito, verranno approfonditi e caratterizzati in seguito.

1. Il livello di **applicazione** è dove risiedono le applicazioni di rete che usufruiscono dei servizi di Internet, alcuni dei protocolli presenti in questo livello sono *HTTP,SMTP,FTP,DNS*, in questo livello, i pacchetti sono chiamati **messaggi**.

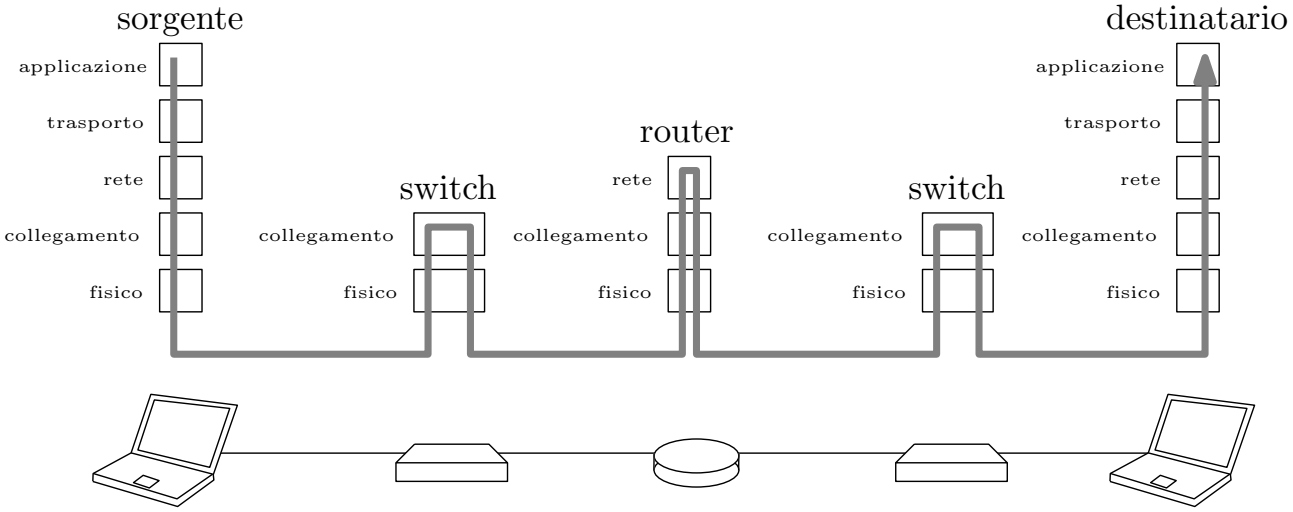
2. Il livello di **trasporto** si occupa del trasferimento dei messaggi dal livello di applicazione di un client al livello di applicazione del server, alcuni protocolli sono *TCP,UDP*, in questo livello, i pacchetti sono chiamati **segmenti**.

3. Il livello di **rete** riguarda l'instradamento dei segmenti dall'origine alla destinazione, un noto protocollo è l'*IP*, i pacchetti in questo livello sono detti **datagrammi**.

4. Il livello di **collegamento** si occupa della trasmissione dei datagrammi da un nodo della rete al nodo successivo sul percorso, alcuni protocolli sono *Ethernet,Wi-Fi e PPP*, lungo un percorso sorgente-destinazione, un datagramma può essere gestito anche da differenti protocolli, i pacchetti qui sono detti **frame**.

5. Il livello **fisico** riguarda il trasferimento dei singoli **bit**sul canale fisico, tramite elettricità nei cavi, oppure onde elettromagnetiche-

Durante la comunicazione, non tutti i sistemi intermedi richiedono che il messaggio venga processato su tutti i livelli, alcuni dispositivi richiedono solo alcuni layer, riducendo la complessità.



Lo strato di un livello ha una comunicazione logica/virtuale con lo stesso livello su un altro computer, ma i dati non sono trasferiti direttamente da uno strato all'altro, passano per tutti i livelli inferiori, un *protocollo* è quindi un insieme di regole che controllano il formato ed il significato dei pacchetti scambiati tra le entità *pari* all'interno di uno strato, un *servizio* invece è un insieme di primitive che uno strato offre a quello superiore, ossia :

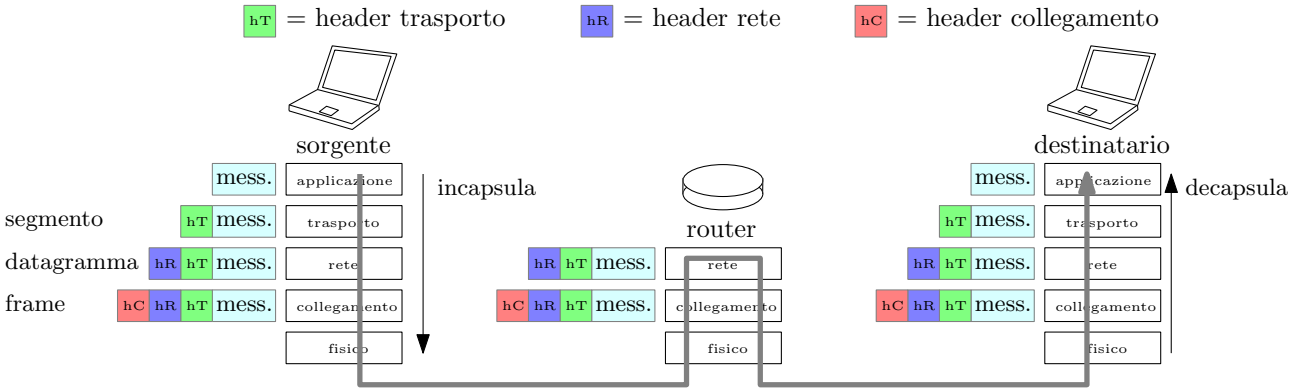
- Quali operazioni fornisce (senza dire nulla sull'implementazione).
- Posto come interfaccia fra due strati, quello inferiore fornisce il servizio, quello superiore ne usufruisce.

1.3.2 Incapsulamento e Multiplexing

Un pacchetto passando di livello in livello viene *processato*, vengono aggiunte informazioni, la sorgente effettua *l'incapsulamento*, prende il pacchetto dal livello superiore, e aggiunge un "intestazione".

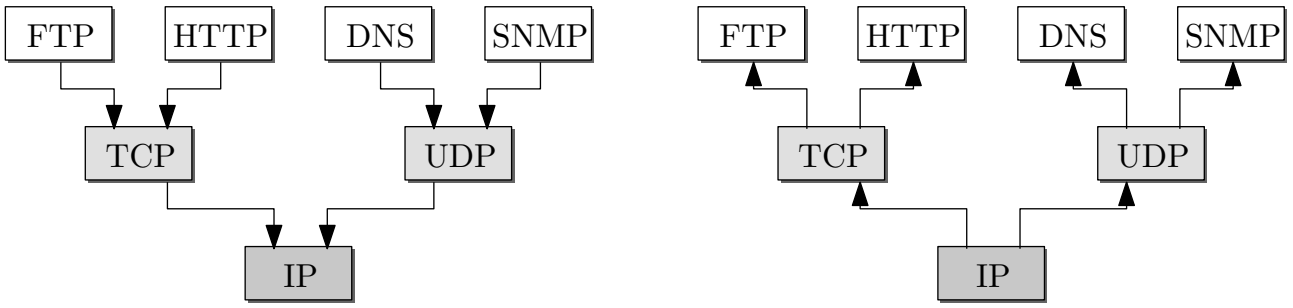
Un *messaggio* passa dal livello di applicazione al livello di trasporto, qua gli verrà aggiunto un "header trasporto" e diventerà un *segmento* (incapsulato), verrà poi passato al livello di rete, e con l'aggiunta di un "header rete" diventerà un *datagramma* (un altro incapsulamento di un messaggio già incapsulato).

Così il processo si ripete nei vari livelli rendendo il messaggio una matrioska, il destinatario che dovrà leggerlo, effettuerà il decapsulamento ai livelli di riferimento (quando il datagramma arriva al destinatario, il livello di rete si occuperà di leggere le informazioni contenute nell'header di rete).



Si è già accennato al fatto che, ad ogni livello, possono esistere più protocolli per l'incapsulamento dei messaggi, è necessario fare *multiplexing* alla sorgente e *demultiplexing* alla destinazione.

- **Multiplexing** - Un protocollo può incapsulare un pacchetto che proviene da più protocolli dal livello superiore (ad esempio, a livello di trasporto, il protocollo *TCP* può incapsulare sia pacchetti che sono stati incapsulati tramite *HTTP*, che incapsulati tramite *FTP*).
- **Demultiplexing** - Un protocollo può decapsulare pacchetti a più protocolli del livello superiore (ad esempio, se al livello di datagramma arriva un frame, il protocollo deve poterlo decapsulare sia in un segmento *TCP*, sia in un segmento *UDP*).



Per poter effettuare il multiplexing, all'interno dell'header sono contenute le informazioni sul tipo di protocollo da usare. Per far si che tale modello protocollare funzioni, è necessario che ogni sorgente e destinazioni, prevedano degli *indirizzi* ad ogni livello.

1.4 Introduzione alla Sicurezza

Esiste un modello referenziale per descrivere gli strati/livelli della comunicazione chiamato **modello OSI**, che ai 5 precedenti aggiunge 2 livelli interposti fra applicazione e trasporto, ossia *presentazione* (si occupa della crittografia) e *sessione* (si occupa della sincronizzazione).

Definiamo **campo della sicurezza**, l'ambito che si occupa di capire come le reti potrebbero subire attacchi, e quali potrebbero essere eventuali difese, Internet in origine, alla sua nascita, non è stato pensato per essere "sicuro", in quanto *Arpanet* era una rete confidenziale condivisa fra utenti che si fidano fra loro, ed ogni livello dello stack è soggetto ad attacchi.

1.4.1 Attacchi alla Rete

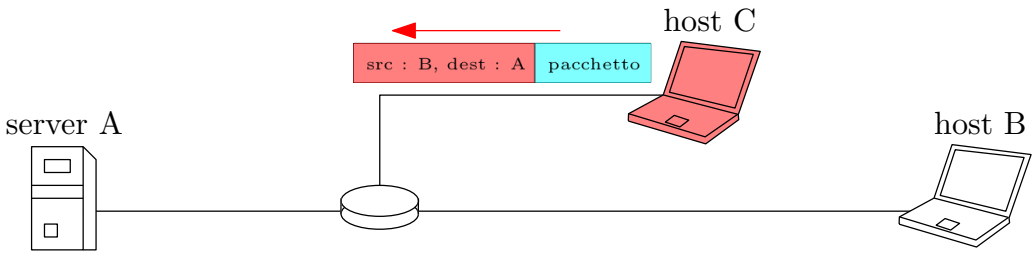
Un **malware** è un software malevolo che può entrare in una rete, ed infettare gli host tramite un meccanismo "autoreplicante" mediante la ricezione/esecuzione di oggetti, come allegati di posta elettronica o file eseguibili.

Uno *spyware* è un tipo di malware che ha lo scopo di registrare gli input dell'host infetto, e le sue tracce sulla rete, come i siti web visitati. L'host infetto può diventare parte di una *botnet*[1], utilizzata per scopi malevoli.

Uno degli attacchi tipici che possono arrivare da una botnet è l'attacco **DOS** (Denial of Service), tale attacco viene eseguito sovraccaricando un servizio (bersaglio), inviando un numero elevato di pacchetti in modo da creare traffico sulla rete e rendere il servizio non disponibile.

Un altro attacco tipico è il **packet sniffing**, ossia l'intercettazione di pacchetti da parte di un terzo utente alla quale tali pacchetti non erano destinati. Viene spesso perpetrato tramite mezzi di trasmissione fisici come il cavo Ethernet o Wireless. L'utente malevolo legge e registra i pacchetti, che possono contenere informazioni confidenziali.

L'**IP spoofing** è un attacco che ha lo scopo di inviare pacchetti ad un destinatario, fingendosi una sorgente fasulla, ovvero utilizzando un indirizzo di origine falso, convincendo il destinatario ad avviare una comunicazione, facendogli credere di star comunicando con un utente fidato.



Ci sono diverse forme di divesa agli attacchi presentati, l'**autenticazione** (dimostrare che il destinatario è effettivamente chi dichiara di essere), la **confidenzialità** (cifrare i messaggi tramite la crittografia), il **controllo integrità** (associare delle firme digitali ad un messaggio per prevenire o riconoscere eventuali manomissioni), le **restrizioni di accesso** (VPNs protette da password), e l'utilizzio di un **firewall** (programmi che si trovano nel nucleo dell'access point, con lo scopo di filtrare i pacchetti e riconoscere eventuali attacchi DOS).

[1] Una botnet è una rete di computer infettati da malware che vengono controllati da un criminale informatico.

2 Livello di Applicazione

Le applicazioni di rete sono il motivo per la quale esiste Internet, è possibile creare un applicazione che si interfacci con la rete in modo facile, possiamo ignorare la complessità dei livelli inferiori, e preoccuparci solo di ciò che è gestito dal livello applicativo.

I due paradigmi di comunicazione a livello applicativo sono

- **Client-Server** : Il server è un host sempre attivo, con un indirizzo IP fisso, ed eroga il servizio sulla rete. Il client è l'host che usufruisce del servizio contattando il server, diversi client non comunicano direttamente fra loro.
- **Peer-to-Peer** : Non esiste un server sempre attivo, qualsiasi utente in rete detto peer può comunicare direttamente con gli altri, più peer ci sono, più incrementano le capacità del servizio, i peer sono connessi con indirizzi IP variabili e la loro gestione è complessa.

I processi in host diversi comunicano fra loro tramite i messaggi, un **socket** è il canale virtuale che si occupa di connettere 2 processi a livello applicativo, sia il server che il client per comunicare, aprono un socket, può essere visto come un'entrata per i messaggi.

Per ricevere i messaggi, un processo deve avere un identificatore univoco nella rete, ogni host ha un indirizzo IP univoco a 32 bit, l'identificatore di un processo è composto dall'indirizzo IP del dispositivo, più un *numero di porta*.

Per inviare un messaggio HTTP al server web *gaia.cs.umass.edu*:

- Indirizzo IP : 128.119.245.12
- Numero di porta : 80

2.1 Definizione di Protocollo

Si è già accennato alle funzioni che deve svolgere un protocollo, non è altro che un insieme di regole che definiscono:

1. I tipi di messaggi scambiati.
2. La sintassi dei messaggi.
3. La semantica del messaggio.
4. Le regole per quando e come i processi devono inviare e rispondere ai messaggi.

I protocolli possono essere aperti e reperibili a chiunque, come HTTP, oppure chiusi e proprietari, come il protocollo che utilizza *Skype*.

2.1.1 Parentesi sul Livello di Trasporto

Facciamo adesso degli accenni al livello di trasporto, di quali servizi necessità un app a questo livello? In base alle caratteristiche o alle funzione, un applicazione potrebbe prediligere alcuni servizi piuttosto che altri:

- **integrità dei dati** : Alcune applicazioni di rete, come quelle che devono trasferire file, richiedono che i dati ricevuti siano al 100% quelli inviati, e non ammettono perdite di pacchetti.
- **garanzie temporali** : Alcune applicazioni necessitano che il ritardo fra l'invio e la ricezione dei pacchetti sia basso, un esempio sono i videogiochi online interattivi.
- **troughput** : Alcune applicazioni richiedono una minima quantità di troughput per essere efficaci, alcune sono "elastiche" e si adattano a qualsiasi velocità effettiva.
- **sicurezza** : Alcune applicazioni richiedono la crittografia dei dati inviati, e la protezione da eventuali manomissioni.

Applicazione	Integrità dei dati	Troughput	Sensibile al ritardo
Trasferimento dati	senza perdite	elastico	no
E-mail	senza perdite	elastico	no
Documenti web	senza perdite	elastico	no
Audio/video in tempo reale	tollerante a perdite	audio : 5Kbps-1Mbps video : 10Kmbs-5Mbps	10 millisecondi
Audio/video in streaming	tollerante a perdite	audio : 5Kbps-1Mbps video : 10Kmbs-5Mbps	qualche secondo
Videogiochi interattivi	tollerante a perdite	1 Kbps o più	10 millisecondi
Messaggistica	senza perdite	elastico	dipende

I due principali protocolli di trasporto sono **UDP** e **TCP**.

- Il servizio **TCP** offre un *traporto dei dati affidabile* fra i due processi, offre *controllo del flusso*, facendo attenzione alla velocità con la quale il mittente manda i pacchetti al destinatario, evitando di ingolfarlo. Offre *controllo della congestione*, limitando il mittente se la rete è sovraccarica, *non* prevede un *timing o troughput minimo garantito*, e non prevede nemmeno *sicurezza*.
- Il servizio **UDP** non garantisce nulla, si occupa eslcusivamente di inviare i pacchetti, può quindi risultare più rapido, e può risultare una "base" per costruire sopra il proprio protocollo, gestendo le proprietà precedentemente elencate a livello di applicazione.

Applicazione	Protocollo a livello applicativo	Protocollo a livello di trasporto
Trasferimento dati	FTP	TCP
E-mail	SMTP	TCP
Documenti web	HTTP	TCP
Telefonia Internet	SIP, RTP o proprietario	TCP o UDP
Audio/video in streaming	HTTP o DASH	TCP
Videogiochi interattivi	WOW, FPS o proprietario	TCP o UDP

Entrambi i protocolli non forniscono alcuna protezione o crittografia, esiste quindi un livello intermedio fra trasporto e applicazione, noto come **Transport Security Layer (TLS)**, e fornisce delle connessioni TCP crittografate, con garanzia rispetto l'integrità dei dati, e l'autenticazione dell'end-point. Le applicazioni di rete si interfacciano con delle librerie TLS, queste ultime si interfacciano con TCP, il testo in chiaro inviato al socket TLS verrà crittografato.

2.2 Web e HTTP

Una pagina web, come quelle che siamo abituati e vedere da sempre, è composta da **oggetti**, archiviati, ossia contenuti nella memoria fisica di un server web, alla quale accediamo dai nostri dispositivi quando vogliamo connetterci ad un sito. Tali oggetti possono essere dei file *HTML*, delle immagini, degli applet Java, dei file audio e altro ancora.

Una pagina web consiste in un file HTML di base, che include più oggetti "referenziati" nel file, ogni oggetto è indirizzato da un *URL*, ossia una dicitura che indica il nome dell'host che funge da server web, ed il percorso del file.

`www.someschool.edu/someDept/pic.jpg`

host name percorso file

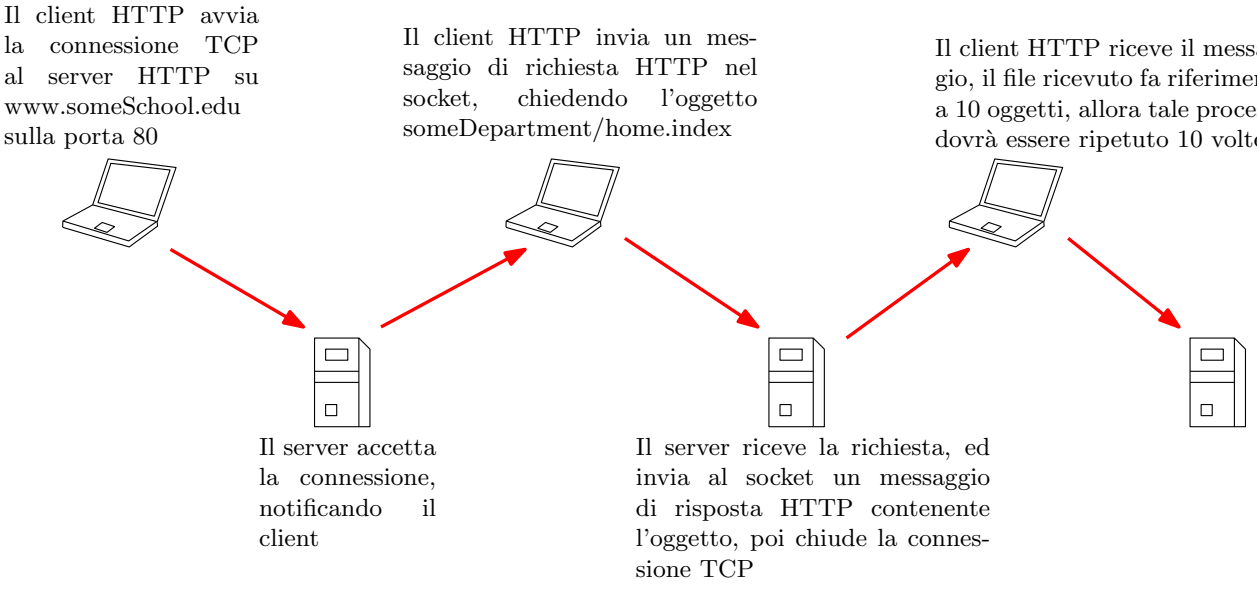
HTTP (hyper text transfer protocol) è un protocollo a livello di applicazione che segue un modello client-server, in sostanza, il client richiede un oggetto (tramite un browser), ed il server risponde, fornendoglielo.

Tale protocollo funziona con il protocollo di trasporto TCP, il client avvia una connessione con il server, il cui processo che si occupa di comunicare è *aperto* sul numero di porta 80. Il server accetta la connessione TCP dal client, ed essi possono scambiarsi dei messaggi HTTP, per poi chiudere la connessione.

Un fatto importante, è che il protocollo HTTP non salva lo stato della comunicazione, non conserva alcuna informazione sulle richieste passate del client.

Le connessioni HTTP sono di due tipi:

- **non persistenti** - La connessione TCP viene aperta, viene scambiato al massimo un oggetto, per poi chiudere la connessione.
- **persistenti** - La connession TCP viene aperta, permette a più oggetti di essere inviati sulla singola connessione, per poi venire chiusa.



Definiamo come **Round Trip Time** (RTT), il tempo che un pacchetto impiega per andare dal client al server, per poi ritornare al client. Il tempo di risposta HTTP per un oggetto equivale al tempo necessario per l'invio della richiesta di connessione TCP, l'invio della richiesta HTTP, e la trasmissione del file, il tempo di risposta di una richiesta HTTP non persistente è quindi:

$$2 \cdot RTT + d_{trans}$$

2.2.1 Formato del Messaggio

Un messaggio HTTP può essere una richiesta oppure una risposta, il messaggio è di tipo testuale, ossia ASCII, ed è composto da una riga di richiesta, dove si dichiara il tipo del messaggio con i comandi `GET,POST` e `HEAD`, da un *header*, e dal *corpo* del messaggio, contente le informazioni da inviare.

Ogni riga deve contenere alla fine i due caratteri `\r\n` per indicare che essa termina, l'header viene chiuso da una riga contenente esclusivamente i caratteri `\r\n`, un messaggio di richiesta non presenta un corpo, si veda il seguente esempio:

```
GET /index.html HTTP/1.1 \r\n      // riga di richiesta
Host: www-net.cs.umass.edu\r\n     // inizio header
User-Agent: Firefox/3.6.10\r\n
Accept: text/html,application/xhtml+xml\r\n
Accept-Language: en-us,en;q=0.5\r\n
Accept-Encoding: gzip,deflate\r\n
Accept-Charset: ISO-8859-1,utf-8;q=0.7\r\n
Keep-Alive: 115\r\n
Connection: keep-alive\r\n
\r\n     // fine header
```

L'header contiene le seguenti intestazioni:

Intestazione	Descrizione
User-agent	Indica il programma client utilizzato
Accept	Indica il formato dei contenuti che il client è in grado di accettare
Accept-charset	Famiglia di caratteri che il client è in grado di gestire
Accept-encoding	Schema di codifica supportato dal client
Accept-language	Linguaggio preferito dal client
Authorization	Indica le credenziali possedute dal client
Host	Host e numero di porta del client
Date	Data e ora del messaggio
Upgrade	Specifica il protocollo di comunicazione preferito
Cookie	Comunica il cookie al server (verrà spiegato successivamente)
If-Modified-Since	Invia il documento solo se è più recente della data specificata

Un messaggio di richiesta di tipo `GET` può includere dei parametri da passare al server, da inserire nell'url dopo il carattere `?`, un messaggio `HEAD` richiede al server esclusivamente

l'header, senza il corpo del messaggio. Una richiesta `PUT` sostituisce il file esistente nell'URL specificato con il contenuto del corpo del messaggio. Vediamo ora un esempio di messaggio di risposta, esso include una *riga di stato*, l'header ed il corpo:

```
HTTP/1.1 200 OK\r\n
Date:  Sun, 26 Sep 2010 20:09:20 GMT\r\n
Server:  Apache/2.0.52 (CentOS)\r\n
Last-Modified:  Tue, 30 Oct 2007 17:00:02 GMT\r\n
ETag:  "17dc6-a5c-bf716880"\r\n
Accept-Ranges:  bytes\r\n
Content-Length:  2652\r\n
Keep-Alive:  timeout=10, max=100\r\n
Connection:  Keep-Alive\r\n
Content-Type:  text/html; charset=ISO-8859-1\r\n
\r\n
data data data data data...      // Il corpo, ossia il file richiesto
```

Il codice di stato che appare nella prima riga fornisce informazioni sulla risposta:

- `200 - OK` : la richiesta ha avuto successo.
- `301 - Moved Permanently` : L'oggetto richiesto è stato trasferito nella locazione specificata nell'header.
- `400 - Bad Request` : Il messaggio di richiesta non è stato compreso dal server.
- `404 - Not Found` : L'oggetto richiesto non si trova sul server.
- `505 - HTTP Version Not Supported` : Il server non supporta la versione di protocollo HTTP.

Le possibili voci dell'header sono le seguenti :

Intestazione	Descrizione
Date	Data corrente
Upgrade	Specifica il protocollo di comunicazione preferito
Server	Indica il programma server utilizzato
Set-Cookie	Il server richiede al client di memorizzare un cookie
Content-Encoding	Specifica lo schema di codifica
Content-Language	Specifica la lingua del documento
Content-Lenght	Specifica la lunghezza del documento
Content-Type	Specifica la tipologia del documento
Location	Chiede al client di inviare la richiesta ad un altro sito
Last-modified	Fornisce data ed ora dell'ultima modifica del documento

2.2.2 Cookie

Il problema di HTTP è che non mantiene alcuna informazione sulle richieste passate, risulta infattibile comunicare con esclusivamente richieste *stateless*, esiste un modo per mantenere lo stato della comunicazione, tramite i *cookie*, sono dei file che vengono inclusi nell'header dei messaggi, conservati sulla macchina dell'host e gestiti dal browser dell'utente.

I cookie permettono al sito che riceve la richiesta di identificare l'utente, il server può chiudere una "sessione" con l'intestazione `Set-Cookie`, rimuovendo tutti i cookie dell'host.

2.2.3 Web Cache

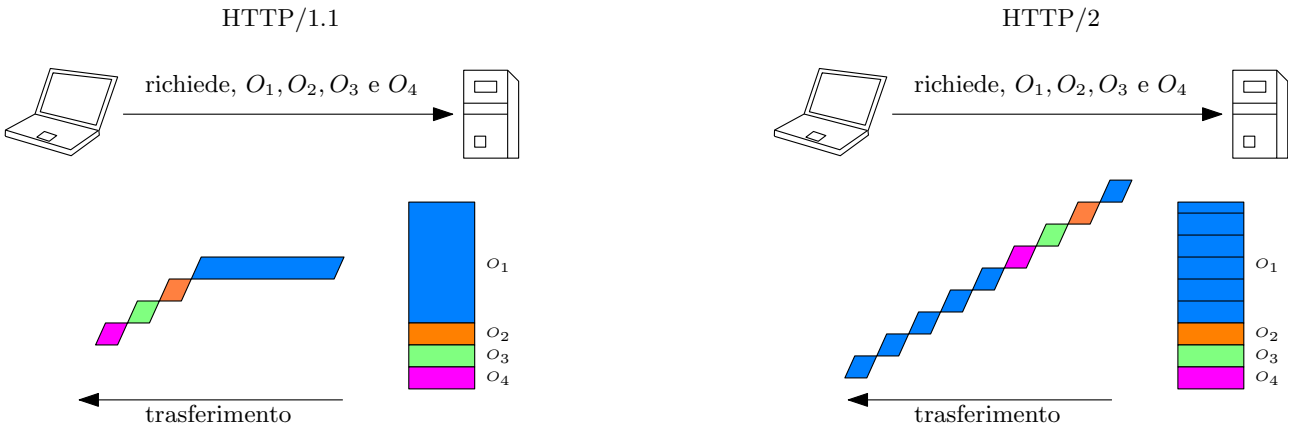
Abbiamo visto che nell'header di un messaggio di risposta HTTP ci sono informazioni riguardanti la data di ultima modifica del documento richiesto. L'utente può configurare un **proxy server**, esso fungerà da cache, il browser potrà fare richiesta ad esso, e se presenta già il file richiesto, non sarà necessario fare richiesta al server di destinazione, se non presenta, una volta ottenuto l'oggetto, esso verrà memorizzato nella cache, rendendo l'accesso più rapido.

Generalmente tali cache sono installate dagli ISP, servono a ridurre il traffico nei link di accesso ad una rete locale e minimizzare il tempo di risposta. Abbiamo visto l'intestazione `if-modified-since` nelle richieste HTTP, essa serve per fare delle richieste `GET` condizionate, richiedendo al server di origine il documento esclusivamente se è stato aggiornato rispetto alla versione già presente sulla cache.

2.2.4 HTTP/2 ed HTTP/3

Abbiamo visto come HTTP/1.1 permette di aprire una singola connessione TCP per lo scambio di più oggetti, utilizza però una politica First Come First Serve, ossia, se vengono richiesti x oggetti secondo uno specifico ordine, verrà forniti nello stesso ordine con la quale sono stati richiesti.

Se il primo oggetto richiesto è molto più grande degli altri oggetti, la sua trasmissione potrebbe rallentare il caricamento di una pagina, HTTP/2 quindi introduce la permutazione dell'ordine di trasmissione secondo delle priorità specificate dal client, più oggetti vengono poi divisi in *frame* della stessa dimensione, e la trasmissione avviene interlacciata.



La creazione di HTTP/3 ha lo scopo di ridurre il tempo di attesa per le richieste multi-oggetto, HTTP/2 funziona su una singola connessione TCP, che è stato pensato per gestire un singolo

flusso di informazioni. La perdita di pacchetti blocca tutte le trasmissioni di oggetti, inoltre, TCP non incorpora alcun livello di sicurezza.

HTTP/3 aggiunge sicurezza e controllo degli errori, e controlla la congestione per oggetto, utilizzando come protocollo di trasporto UDP.

2.3 SMTP e Posta Elettronica