RSO - Reti e Sistemi Operativi

Sara Angeretti

Gennaio 2023

Indice

Capitolo 1

Introduzione alle reti

Capitolo 1 (Kurose-Ross 7a edizione).
Paragrafi: 1.1 - 1.5 (escluso 1.3.2, "circuit switching")

1.1 Cos'è una rete

1.1.1 Introduzione ad una rete

Quando parliamo di Internet sappiamo che parliamo di una **collaborazione**, una **cooperazione** fra più server e/o più utenti, organizzati tramite una divisione di ruoli. Questa è possibile solo in caso di **compatibilità** fra le *tecnologie* e fra gli *standard* scelti.

Alcune terminologie utili.

- Parliamo di **end system** o **terminale** per indicare il punto che fa da *capo* alla comunicazione (dove comincia o dove finisce). Sono tutti gli apparati che noi conosciamo visivamente come computer di utenti o server o torri radio...

  Non sono da confondere con i **router** o gli **switch** (distinguiamo in base a tecnologie fisiche e softwares usati su di essi), che sono *intermediari, nodi* che fanno da tramite alla comunicazione. Un'altra distinzione utile fra switches e routers è la seguente: nella rete, gli **switches** si trovano dentro la parte di rete più "privata" (domestica ma non solo), mentre i **routers** si trovano all'esterno di questa parte, nella sezione di rete più pubblica. Questi non sono **end system**, ma fanno parte della sezione **core**, che indica la parte più interna e "invisibile" (all'utente che usa un host) della rete.

- Parliamo di **network apps** (**applicazioni di rete**) per indicare cosa fanno per noi nella comunicazione determinati pezzi di software.

- Parliamo di **bandwidth** (**larghezza di banda**) per quantificare con un **numero** (la **bandwidth** è un *numero*) un certo livello di prestazione della rete che vogliamo valutare. Serve per capire il punto in cui un parametro va bene o va male e perché.

Più nel dettaglio (organizzazione della struttura di una rete come schematizzato nell'immagine). Zone diverse avranno esigenze diverse e quindi soluzioni e strutture diverse.

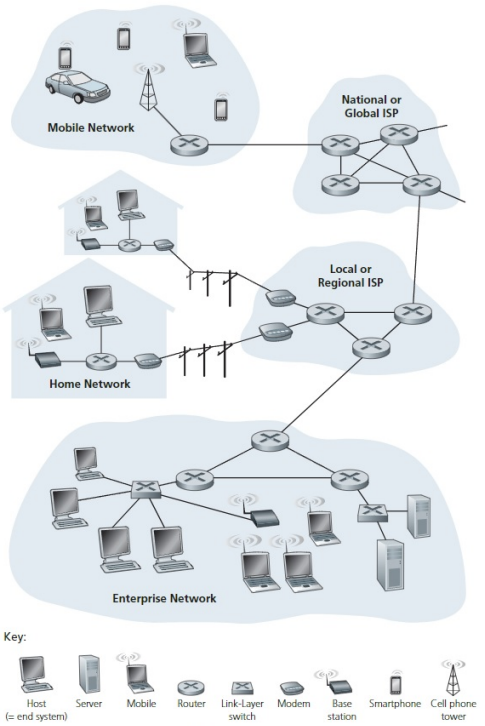


Figure 1.1 ♦ Some pieces of the Internet

Nella foto per esempio possiamo vedere in *alto* il lato cliente e in *basso* il lato fornitore.

Altra terminologia.

- Le **network edges** sono le parti terminali (**hosts**, che si dividono in *clients* e in *servers*, questi ultimi possono essere raggruppati in *data centers*).

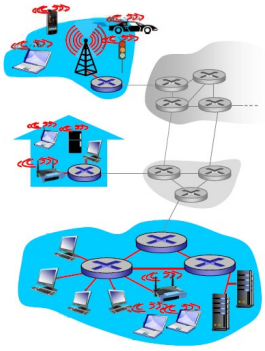
- Il **network core** è tutto ciò che sta "dietro le quinte". Verranno utilizzate tecnologie diverse da quelle delle **network edges**. Sono gli **switch** e i **router**.

- Da "tessuto connettivo" delle due aree sopracitate fungono **access network** e **physical media**. Sono composti da tutto ciò che mi serve per andare dalla parte *edge* alla parte *core*, quindi punti di *connessione* e *tratti* o *cammini* che possono essere composti da cavi fisici o connessioni senza fili.

  Esempio: per andare dal mio computer di casa (un *client edge*) al modem del fornitore (lo possiamo vedere come uno *switch*) c'è un cavo, il mio *cammino fisico*.

1.1.2 Access Network - collegamento

Una distinzione che è possibile fare per il concetto di "funzione di *collegamento fisico*" è quella delle diverse implementazioni che questa funzione ha a seconda dell'ambito in cui si trova.



Per esempio, sarà molto diverso fra:

- Residential access nets (rete privata).
- Institutional access networks (school, company, quindi reti pubbliche).
- Mobile access networks (reti mobili).

Per riuscire a fare questa distinzione, bisognerà tenere a mente:

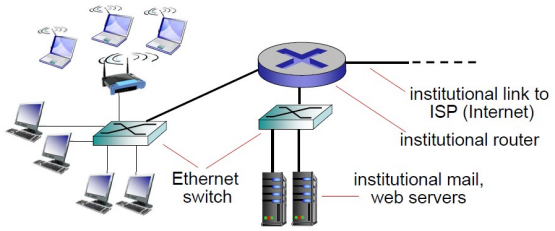
- Bandwidth (bits per second).
- Shared or dedicated network?

1.2 Alcuni tipi di reti

1.2.1 Residential access nets - reti private

Al centro di questa immagine abbiamo un altro esempio di **router**, fin qui definito uno *strumento tecnologico intermediario* (quindi *core*) nelle reti pubbliche. Ovviamente non c'è mai un solo pc (una *edge*) per un solo fornitore: abbiamo un punto intermedio (*switch* o *WiFi*) che raccoglie le informazioni/connessioni da più pc e le porta al fornitore telefonico (esterno al grafico).

1.2.2 Enterprise access nets - Ethernet



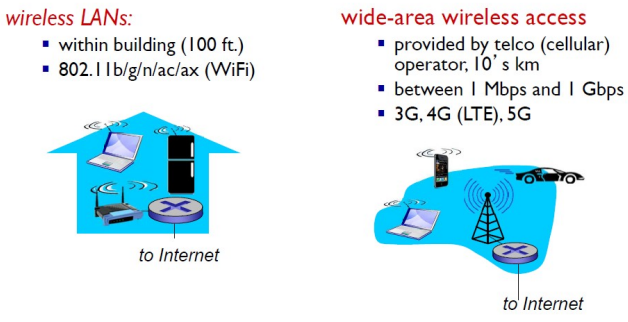
Prima tecnologia degna di presentazione. Storicamente una delle più importanti tecnologie di rete, semplice ma interessante da vedere.

Nel tempo è stata arricchita nelle prestazioni (aumento della larghezza di banda: 10Mbps, 100Mbps, 1Gbps, 10Gbps...), questa sua capacità di raggiungere prestazioni via via più elevate l'ha resa adatta all'uso anche in ambienti aziendali (non solo domestici).

Ethernet è tipicamente su cavo, sfrutta diversi *ethernet-switch* (a livello datalink, cioè switch, diremo qualcosa di più preciso). Vedremo un capitolo dedicato, che inquadra due livelli: quello *fisico* (che tipo di cavo) e quello *datalink* (come identifico chi è collegato grazie al mio cavo?).

1.2.3 Wireless access nets - WiFi

Se Ethernet era un esempio di connessione con il cavo, sappiamo che ci sono anche connessioni senza cavo.

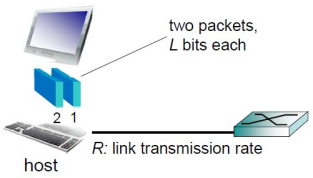


Nell'immagine abbiamo l'esempio di **WiFi** a *sinistra* (tipicamente privata) e di **4G** a *destra* (tipicamente risorsa delle aziende). Sono entrambi sistemi ad onde radio che collegano gli end-system alla base del fornitore anche nota come "access point (punto di accesso)" tramite una connessione che non prevede cavi.

1.3 Frammentazione della comunicazione

1.3.1 Cosa fa l'host - pacchetti di dati

Concetto oggi magari banale, ma non tempo fa. In origine la rete dedicava un canale costantemente e in esclusiva a mittente-destinatario, l'intero tempo di uso della rete era loro e esclusivo. Il che vuol dire che finché non finivano di inviare le informazioni, la rete non si liberava. La rivoluzione di 30 e più anni fa è che ora posso andare a *sezionare* o *frammentare* la comunicazione in **pacchetti** di alcuni bit di lunghezza (circa 1Kb, 8000 bits, è più o meno una misura standard).



Ma di cosa si occupa l'host (**network edges** o *parti terminali* di una rete) di una rete?

- Accetta il messaggio dell'applicazione.
- Rompe l'informazione in blocchi più piccoli, noti come pacchetti, di lunghezza L bit.
- Trasmette il pacchetto nella rete alla velocità di trasmissione R, espressa in bit per secondo.

  La *velocità di trasmissione del collegamento* è nota anche come *capacità del collegamento* o come *larghezza di banda del collegamento* (*bandwidth*). Soprattutto a livello fisico, si incrocia con il valore della lunghezza del pacchetto.

Vediamo qui la prima *formula*, la prima *quantificazione numerica* di quanto una rete impiega per trasmettere un determinato pacchetto.

$$delay = \frac{L(bits)}{R(bits/sec)}$$

Il tempo necessario al trasferimento di un pacchetto di L bit di lunghezza sulla connessione è il ritardo (*delay*) di trasmissione del pacchetto in questione.

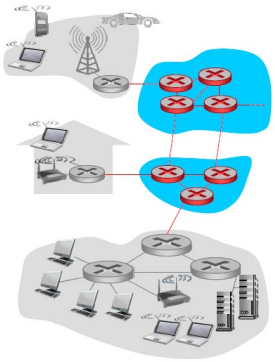
Ma come colleghiamo i due concetti? Come arriviamo a parlare di *delay* partendo dal tempo di trasmissione di un pacchetto?
Tanto per cominciare dobbiamo avere ben chiaro il concetto di *tempo di trasmissione* (**tTX**). Facciamo l'esempio seguente: abbiamo un pacchetto di dati di $L = 1000$ bit su una rete di $R = 1000$ bps. Quanto tempo ci vorrà per trasmettere quel pacchetto su quella rete? Ovviamente $tTX = 1sec$.
Tutto questo però sulla carta.
In realtà il vero tempo di trasmissione di un pacchetto è visto più come un **ritardo** (*delay*): mettiamo di avere l'*istante in cui il* ***mittente inizia***, l'*istante*

in cui il ***destinatario finisce*** e la distanza tra questi due momenti. Il tempo quindi tra cui il mittente inizia a trasmettere e il momento in cui il destinatario *realmente* ha ricevuto il pacchetto completo, è visto come un *delay*, un "ritardo" fra la disponibilità del dato in trasmissione e disponibilità in ricezione.

1.4 Rete e instradamento

1.4.1 Parte core della rete

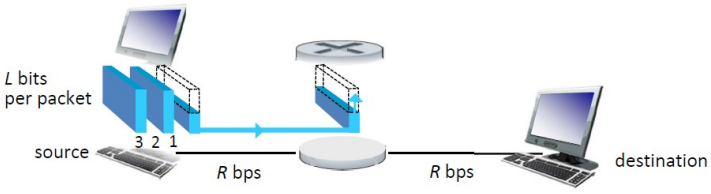
Ovviamente ogni azienda che gestisce l'infrastruttura di comunicazione può usare la sua comunicazione.
Quello che importa è che la rete, nell'immagine rappresentato come insieme di tanti punti di transito, è una maglia di router interconnessi.



Commutazione di pacchetto: gli host frammentano il livello di applicazione messaggi in pacchetti.
I pacchetti vengono inoltrati da un router al successivo, attraverso collegamenti sul percorso dalla fonte alla destinazione.
Ogni pacchetto viene trasmesso alla piena capacità della rete.

1.4.2 Store and forward.

Come precedentemente illustrato, ci vogliono $\frac{L}{R}$ secondi per trasmettere un pacchetto lungo L bit su una connessione ad una velocità (della rete) di R bps.



Il meccanismo di trasmissione ***store-and-forward***, che avviene tra *source* e *destination* tramite un punto intermedio (che prima non consideravo in quanto appartenente alla parte *core*), prevede che la trasmissione dell'**intero pacchetto** non avvenga "bit a bit" ma solamente una volta che l'**intero pacchetto** è stato inoltrato e memorizzato, prima di essere nuovamente inviato. Ovviamente:

- non stiamo parlando a livello di cavo fisico, dove ovviamente ci sarà una trasmissione "bit a bit" che ci permette quindi di avere una velocità R di trasmissione;

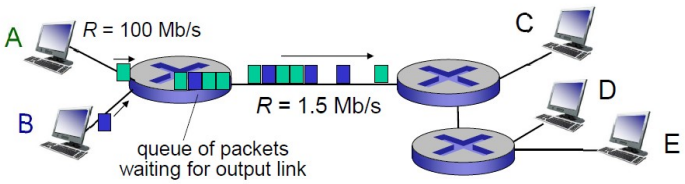
- il *nodo intermedio* deve avere uno spazio di deposito del dato per poterlo memorizzare per intero, solo quando sarà lì tutto potrà uscire dal nodo intermedio e quindi inoltrato.

- è chiaro che i tempi si allungano, in quanto dovendo ricevere tutto il pacchetto nel nodo intermedio prima di poterlo inoltrare il tempo si raddoppia (e di conseguenza si raddoppia anche il *delay*).

Perciò l'approccio *store-and-forward* ci peggiora la vita per quanto riguarda i tempi, infatti dover aspettare che tutto il pacchetto arrivi nel nodo intermedio fa aspettare di più il destinatario. Vedremo però che migliora l'organizzazione del fluire dei dati. Aiuta a comprendere questo concetto immaginare (tranne che a livello fisico) il pacchetto come un blocco rigido, ogni spostamento avviene per l'intero blocco.
In conclusione, il tempo di trasmissione da capo a capo (*end-to-end delay*) $= 2\frac{L}{R}$.

1.4.3 Packet switching: queueing delay, loss.

Vediamo una situazione un po' più concreta (reale) che ci mostra il vantaggio di avere la trasmissione a pacchetto e del condividere tratti di archi di comunicazione fra più computer. Questo vantaggio ha anche degli eventuali problemi che ora vedremo: cioè se più dispositivi condividono uno stesso canale di comunicazione, bisogna fare attenzione che non interferiscano "pestandosi i piedi" a vicenda! Questo perché ovviamente nello stesso canale fisico non possono passare due bit nello stesso istante.



Per quanto detto anche riguardo l'approccio *store and forward*, possiamo quindi affermare che sul canale **condiviso** non avvenga un'alternanza di *bit* ma di *pacchetti*. Ricordando che siamo ancora in modalità *store and forward*, notiamo che nel primo noto intermedio (condiviso da più utenti) c'è un accumulo, un salvataggio dei pacchetti di dati per intero provenienti dai due computer sulla sinistra. Perciò può capitare che si accumulino anche pacchetti di dati molto vecchi e particolarmente lenti ad essere trasmessi (velocità della coda in uscita lenta). Come detto prima il nodo intermedio avrà bisogno di una coda, uno spazio di deposito dei vari pacchetti, che potrebbe tuttavia non essere capiente a sufficienza. Questa cosa può essere vista nell'esempio rappresentato nell'immagine delle due **velocità in entrata e in uscita** del nodo intermedio. Es. tipico: casello autostradale.

N.B.: i pacchetti, provenienti dai due computer sulla sinistra nell'esempio rappresentato, sul canale condiviso non possono essere trasmessi **contemporaneamente** ("sul canale fisico non possono passare due bit contemporaneamente"), ma devono essere **alternati**. È per questo che la coda di immagazzinamento (storing) del nodo intermedio rischia di saturarsi molto velocemente.

Parliamo di **accodamento** e **perdita** (**queuing and loss**) a livello di nodo intermedio nel caso in cui le velocità di **entrata** superino per un certo periodo di tempo la velocità di **uscita** dal nodo. Una volta che il nodo intermedio ha accumulato pacchetti fino a riempire la sua **limitata** capacità fisica di memoria (*buffer*), se i due computer provano ancora a inviare pacchetti, sorge un problema. La possibile soluzione varia a seconda delle tecnologie:

- i pacchetti possono essere persi (dropped, lost), il nodo cioè "butta via" *silenziosamente* i pacchetti;

- il nodo *segnala* al trasmittente (i due computer sulla sinistra) di non avere spazio, il trasmittente aspetta accodando i pacchetti che vuole trasmettere.

Tra l'opzione di mandare indietro il segnale di memoria piena e invece l'opzione di buttar via silenziosamente il pacchetto, **Internet** sceglie la via più semplice (cioè la seconda). Sarà quindi compito del **mittente** (i due computer) accorgersi che il pacchetto inviato non è stato ricevuto dal nodo intermedio. TCP è un esempio di standard che si occupa di questo.

Obiettivo finale: nodi diversi si occupano di problemi diversi, i nodi intermedi nella loro versione semplice si occupano solo di prendere ed inoltrare.

Nuovo problema: nel caso di più uscite da un singolo nodo intermedio, come facciamo a capire dove deve andare il pacchetto da trasmettere? Ovviamente dipende dal **destinatario**. In base alla destinazione ho una scelta di percorso. Ma non solo: dipende anche dalla **situazione** (per esempio un nodo intermedio non funziona più). È una questione abbastanza estesa, ci dedicheremo un capitolo intero nella sezione del *livello di rete (network)*. Il succo è che la scelta di quale percorso seguire in base alla situazione in cui mi trovo è dettata da specifici algoritmi. Ma Internet fa del suo meglio (*best effort*).

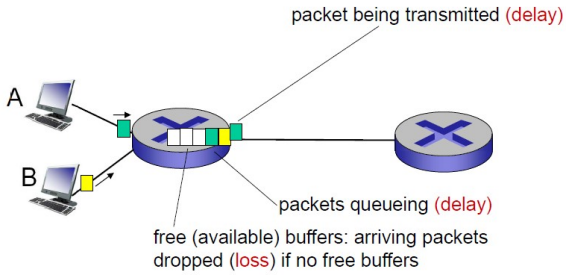
1.4.4 Routing and forwarding.

Introduciamo i termini **instradamento** (**routing**) e **inoltro** (**forwarding**).

- il **routing** è una scelta del percorso che dipende dall'*algoritmo di instradamento* o *di routing*;

- il **forwarding** è semplicemente un nodo passa l'informazione ad un altro nodo.

1.4.5 Altro sui tempi di trasmissione, *delay* e *loss*.



Come abbiamo già illustrato, i pacchetti provenienti dai computer A e B vengono trasmessi con due velocità (presumibilmente) diverse al nodo intermedio (di sinistra): qui vengono accodati per essere poi trasferiti al nodo successivo (qua a destra) tramite un cavo.

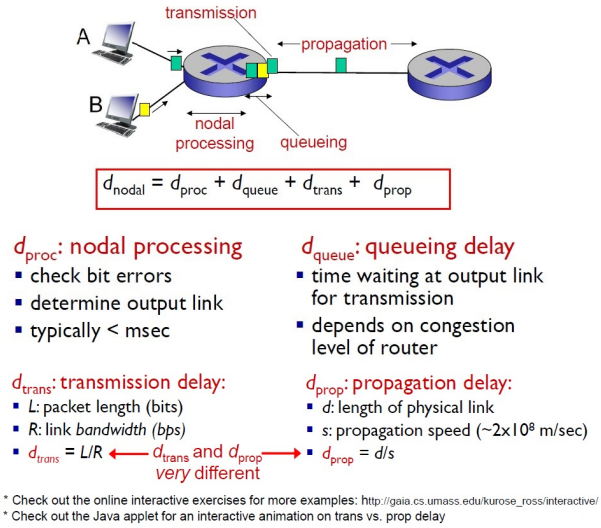
Questo cavo (fisico) abbiamo detto ha una velocità sua di trasferimento. Per spostarsi tra i due nodi intermedi, i pacchetti ci mettono un tempo fisico pari a $\frac{L}{R}$ che è il minimo tempo strettamente necessario per attraversare il cavo. Questo tempo, sommato al tempo che i pacchetti passano in coda nel primo nodo intermedio, mi dà il *ritardo* o *delay* dei pacchetti (dato sia dal *ritardo di trasferimento* dei pacchetti da nodo intermedio all'altro e dal *ritardo di accodamento* su un nodo).

Questo secondo tempo (di accodamento) non lo conosciamo ed è difficile da determinare, in quanto è *transitorio*, cioè dipende dalla situazione in cui ci troviamo e potrebbe variare in modo abbastanza *imprevedibile*.

Un'altra cosa che potrebbe variare in modo abbastanza imprevedibile è ciò che abbiamo chiamato *loss*, ovvero l'eventuale perdita di dati.

Come abbiamo precedentemente già detto, la perdita (*loss*) dei dati avviene perché la coda (*buffer*) del nodo intermedio immediatamente precedente a dove ci troviamo ha dimensione *finita* ma i pacchetti di informazione continuano ad arrivare anche quando è completamente piena. L'informazione persa può essere trasmessa di nuovo dal mittente, dal nodo intermedio oppure persa per sempre.

1.4.6 Quattro sorgenti di *delay* nella trasmissione di pacchetti.

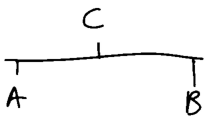


L'immagine prova a modellare quanto detto finora.
N.B.: l'immagine illustra solo il *delay*, non considera l'eventuale *loss*.

Perciò il tempo reale di attraversamento (salto) di un nodo è:

$$d_{nodal} = d_{proc} + d_{queue} + d_{trans} + d_{prop}$$

Un esempio pratico per capire quanto importante sia questo concetto della propagazione fisica è la vecchia Ethernet di una volta.
Immaginiamo di avere un cavo coassiale con tre stazioni. A livello fisico, è **condiviso** (aka tutti vedono tutti).

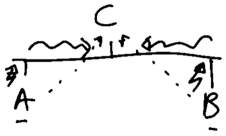


Cosa succederebbe se entrambe le stazioni A e B si mettessero a trasmettere dati alla stazione C? Come entrerebbe in gioco il tempo di propagazione in questa situazione?
Si operava così:

1. *Check* disponibilità del cavo: per evitare di interferire quando il cavo non è occupato si aspetta sia libero.
2. Trasmetto (se libero).

Ma questo è sufficiente per garantire che il segnale arrivi "pulito" a destinazione? No. Può capitare che le due stazioni comincino a trasmettere in contemporanea in assenza di un segnale di "cavo occupato". Più il tempo di trasmissione

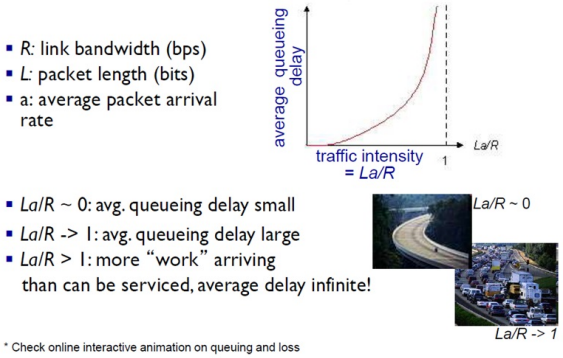
aumenta, più è facile che il segnale di una stazione incappi in segnali di altre stazioni non visti.



La soluzione fu quindi di **limitare** la lunghezza del cavo. Una lunghezza massima prestabilita di questo cavo semplificava anche i controlli (per assicurarsi che era tutto a posto).

1.4.7 Quantificazione del *delay* di trasmissione.

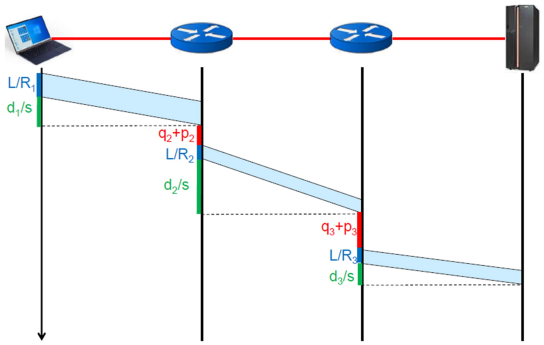
È una **stima** (molto probabilmente lo rivedremo del corso di Ricerca Operativa).



In base al valore di a abbiamo più o meno rischi di perdere dati.

1.4.8 Simulazione/modellizzazione del *delay* di trasmissione.

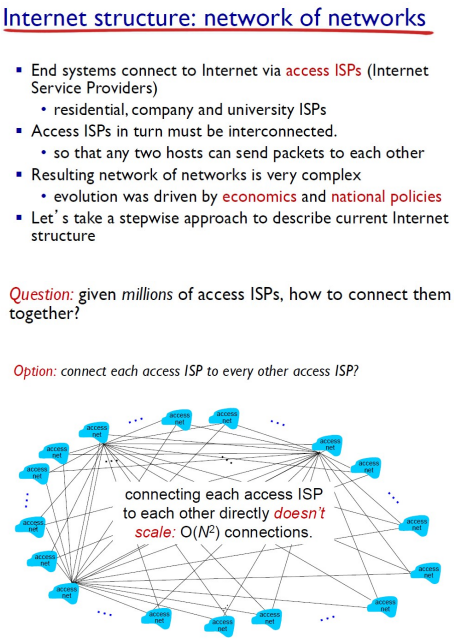
Questo diagramma riassume un movimento dinamico nel tempo a partire da un utente, attraverso due nodi intermedi (non consideriamo le code) e infine fino ad un server.



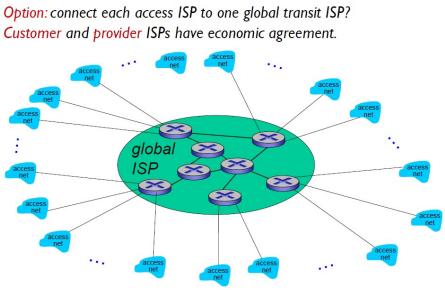
Il tempo rappresentato dalla fascia azzurra ($\frac{L}{R}$) è il tempo di **trasmissione** (numero di *bit* sulla velocità espressa in $\frac{bit}{sec}$). Qual è il tempo di **propagazione**?

Quello rappresentato in verde, ovvero la distanza fisica del cavo d fratto la velocità di propagazione s, $\frac{d}{s}$.
In rosso sono rappresentati anche il tempo di **accodamento** (*queue*) e di **elaborazione** (*processing*).
N.B.: guardando solo il tempo di propagazione dei due nodi intermedi, si vede in base a quanto alta è la fascia del tempo di propagazione che il primo nodo è più lento del secondo.
Una situazione più realistica è possibile vederla grazie al comando ***traceroute*** (provato e commentato nella sezione 1.6.1 dei miei appunti).

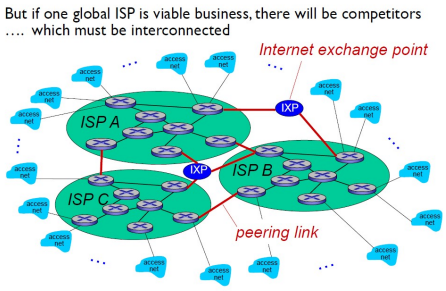
1.5 La struttura di Internet: rete di reti.



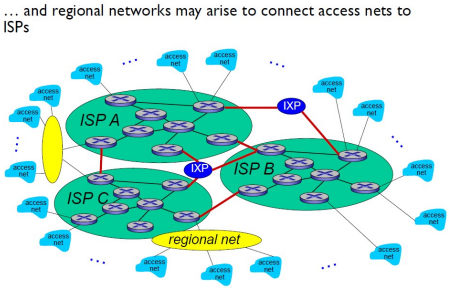
Consideriamo per esempio un insieme di utenze (tipo aziende) tutte collegate fra loro. È chiaro che se avessimo n aziende avremmo così un numero di collegamenti spropositato, dell'ordine di grandezza di $O(n^2)$. Poco pratico.



Una collaborazione non comporterebbe solo flessibilità ma anche efficienza: con meno collegamenti posso raggiungere più destinatari.



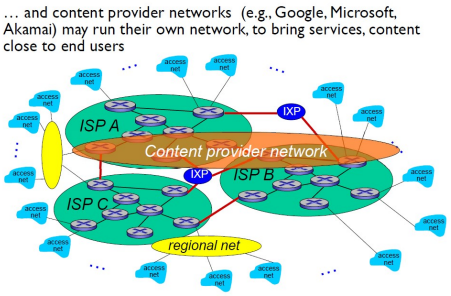
Questo porta a particolari accordi fra diversi fornitori (internet providers), ciascuno dotato dei propri clienti che *in proprio* collega fra loro. Semplice. Ma per far comunicare fra loro clienti diversi di providers diversi? Dovranno transitare da altri fornitori.



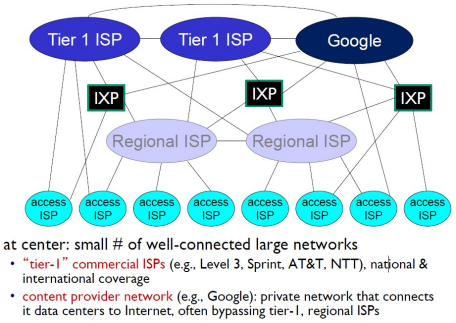
Questo è reso possibile in diversi modi:

- tramite **archi diretti**, veri e propri collegamenti fisici.

- **punti di interscambio**, livelli intermedi di collegamento condivisi (e pagati) da più aziende e che danno un punto di scambio per i dati che devono essere trasferiti.



Risultato finale:

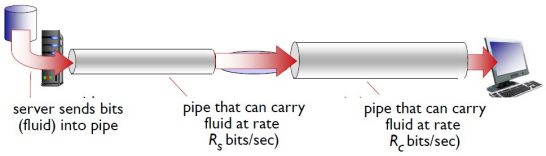


1.6 Prestazioni della rete: throughput.

1.6.1 Throughput: cos'è.

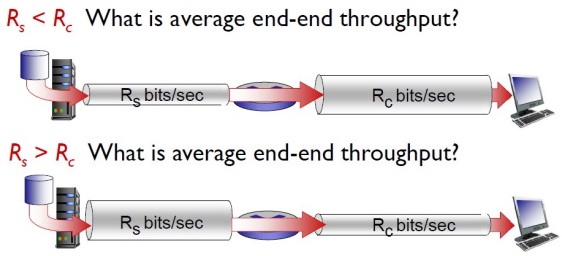
Velocità (espressa in $\frac{\text{bit}}{\text{unità di tempo}}$) a cui determinati *bit* in un determinato periodo di tempo vengono trasmessi da mittente a destinatario, ottenuta dopo tutti gli effetti del singolo viaggio.
Di due tipi:

- **Istantaneo** (*instantaneous t.*): velocità espressa in un punto preciso del percorso.
- **Medio** (*average t.*): velocità espressa in un momento preciso del percorso.



1.6.2 Throughput: visione a "collo di bottiglia".

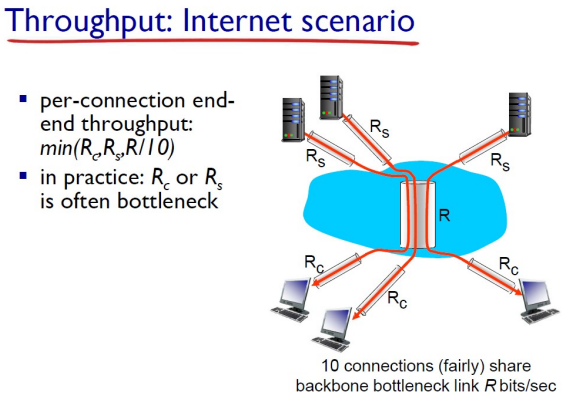
Abbiamo questa situazione: l'informazione passa da una sorgente, attraverso un noto intermedio e fino ad una destinazione. Abbiamo quindi due *throughput* diversi, uno prima (R_S) e uno dopo (R_C) il nodo intermedio.



Ovviamente la prestazione complessiva dipende dal rapporto che intercorre fra le due R.
È possibile notare che:

- se $R_S < R_C$, non ci sarà molta coda di attesa a livello di nodo intermedio essendoci a valle una velocità (R_C) consistente;
- se $R_S > R_C$, siamo nella situazione opposta ovvero tende a crearsi una coda di attesa decisamente consistente a livello di nodo (con conseguente rischio di *loss* di informazione).

A questa situazione facciamo riferimento quando parliamo di ***bottleneck link***, ovvero *collegamento a collo di bottiglia*, che è il tipo di condivisione appena descritta.



N.B.: questa situazione illustra una condivisione (la rete internet) che è **equa**, parliamo di *neutralità della rete*.

1.7 I protocolli di Internet: in breve.

Protocolli organizzati in *layers*, che messi assieme costituiscono la nostra rete (concetto base importantissimo).

1.7.1 Protocolli e layers.

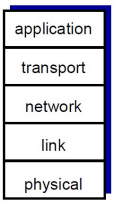
La rete è complessa, composta da molti pezzi (alcuni anche materiali) che collaborano fra loro.

- **hosts**: sono le macchine di terminale (*edge*);
- **routers**: sono le macchine interne che instradano;
- **links of various media**: sono collegamenti con vari supporti fisici (*media*) quindi cavo, fibra...
- **applications**: sono elementi software che fanno generazione del traffico (applicazioni, browser, server web...);
- **protocols**: sono standard che mettono in collaborazione correttamente i vari elementi;
- **hardware, software**: vario supporto hardware, schede...

Tutti questi elementi hanno esigenze diverse ma devono collaborare per un buon funzionamento complessivo.
Come è possibile fare ciò? Organizzandoli per ***ambiti*** o ***livelli***.
Questa divisione deve essere standardizzata, cioè serve avere un accordo sulla terminologia e le definizioni. Questa *compartimentazione*, questa *modularità*, semplifica ogni valutazione o aggiornamento senza doversi coordinare con il *resto* del sistema (solo con una o due parti, ovvero quelle immediatamente adiacenti a quella dove mi trovo).

1.7.2 Internet come una cooperazione.

Tutto quello di cui abbiamo parlato finora dipende da una cooperazione fra **tecnologie** e **standard** (non prodotti, i prodotti implementano gli standard, es. Chrome implementa HTTP), gli ultimi sono divisi per **funzionalità**. I vari standard collaborano fra loro secondo una struttura verticale, basandosi sulla *adiacenza* o *contiguità*: un livello può quindi comunicare esclusivamente con il livello immediatamente precedente o immediatamente successivo.



- **Livello applicativo: supporta applicazioni di rete.** Es.: HTTP.

  Sono tutto il nostro software utente. Non le vedremo in dettaglio, ci interessa sapere che l'applicazione che vuole scambiare dati con un server web deve farlo rispettando un protocollo (HTTP).

  Client + server: sono a livello applicativo entrambi implementazioni di HTTP.

- **Livello di trasporto: trasferisce dati fra processi.** Es.: TCP, UDP.

  Il livello trasporto si occupa di "far vedere la rete all'applicazione". Quest'ultima se ne frega dei pacchetti dei dati. Il trasporto non fa vedere il pacchetto ma internamente è obbligato a frazionare i dati da trasferire in pacchetti. Perciò il compito del livello di trasposto è di collaborare con i livelli immediatamente adiacenti, uno sta sopra l'altro sta sotto.

  Due esempi di protocollo (quelli citati) sono TCP, dove c'è garanzia di consegna, e UDP, che invece butta in giro pacchetti dimenticandosene (non completamente inutile, per condivisioni di dati come per esempio multimediali va benissimo UDP).

- **Livello di rete: routing di dati (mittente→destinatario).** Es.: IP.

  Il livello di rete mette in collaborazione i nodi fra loro; quando c'è collaborazione, è il livello di rete a decidere l'instradamento. Non solo, si occupa anche di identificare mittente e destinatario.

  Perciò *livello di rete = identificazione + instradamento.*

- **Livello di data link: trasferimento di dati tra elementi di rete adiacenti.** Es.: Ethernet.

  Gestisce il trasferimento di dati fra nodi adiacenti. Il livello di data link si concentra sul singolo arco e ignora cosa gli sta intorno, organizza il salto dei dati mettendosi d'accordo con chi "sta sopra" e chi "sta sotto": sotto dettatura dell'instradamento, fa un singolo salto sapendo già su quale livello fisico deve andare.

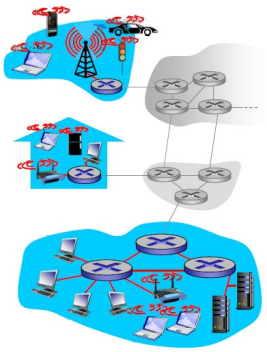
- **Livello fisico: i bit sul cavo fisico.**

  Un altro livello presentato superficialmente. Ne abbiamo parlato in dettaglio nelle sezioni 1.3-1.6.

  Il livello fisico ha rilevanza per quanto riguarda la sicurezza: è chiaro che un livello fisico via segnale radio sarà più fragile parlando di sicurezza rispetto ad un collegamento via cavo (ci sono corsi universitari appositi, ma per fare un esempio è per questo che quando si parla di connessioni wireless si vede anche la loro crittografia, perché sono connessioni poco sicure).

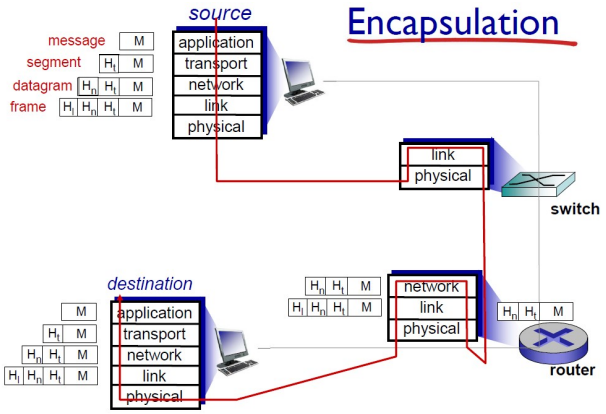
1.7.3 Esempio pratico di trasferimento di un dato.

Quest'immagine rappresenta l'output di un comando che cerca di farmi vedere cosa accade quando cerco di mandare un dato a destinazione. Il comando è *traceroute (tracert)*. Come funziona? Chiede la destinazione che si vuole raggiungere e fa vedere ogni salto intermedio di comunicazione.



Nell'esempio raffigurato, presumibilmente si è cercato di far partire il dato dalla sede degli autori del libro ("umass") verso una macchina per loro dall'altra parte dell'oceano (in Francia, ".fr"). I salti compiuti sono almeno 19 (infatti abbiamo 19 righe): abbiamo un mittente, un destinatario e 17 nodi intermedi. È possibile vedere ad ogni step l'*identità tecnica* (IP) delle macchine attraversate e l'*identità più umana* (lo vedremo col sistema *vbns*). Poi vengono fatti esperimenti di comunicazione (tre volte per ogni nodo intermedio) e vengono registrati i tempi per ognuno dei messaggi inviati.
Nell'esempio raffigurato, è possibile notare un considerevole aumento del tempo di transito (*delay*) che potrebbe corrispondere al salto al di là dell'oceano.

Ma perché <u>tre</u> misurazioni? Perché noi sappiamo che la rete *fa del suo meglio* (best-effort); per avere un'indicazione approssimativa ma affidabile si fanno tre misurazioni, anche per evitare situazioni transitorie e non ottimali che potrebbero dare una misurazione non accurata.



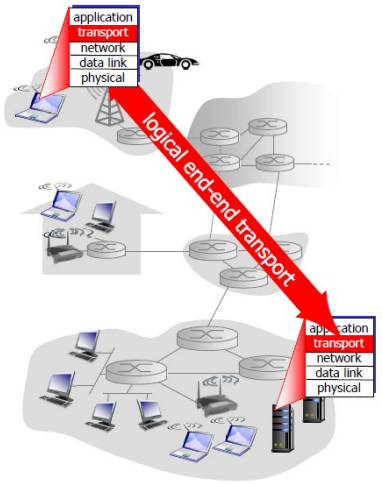
Capitolo 2

Strato di trasporto

Capitolo 3 (Kurose-Ross 7a edizione).
Paragrafi: 3.1 - 3.7 (escluso 3.4.1, "Building a Reliable Data Transfer Protocol", 3.5.2, "TCP Segment Structure", 3.5.3 "Round-Trip Time Estimation and Timeout")

2.1 Servizi e protocolli del livello di Trasporto.

Questo livello si occupa di fornire **comunicazione logica** tra i processi di applicazioni in esecuzione su *hosts* diversi.
Cosa succede nel *lato mittente*: si occupa di spezzare i messaggi delle applicazioni in ***segmenti*** che passa poi al livello di rete.
Cosa succede nel *lato destinatario*: si occupa di riassemblare i segmenti in messaggi che manda poi al livello applicativo.



C'è più di un tipo di protocollo per il livello di trasporto, Internet per esempio usa sia **TCP** che **UDP**.

2.1.1 Livello di trasporto vs Livello di rete.

Il livello di *rete* si occupa della comunicazione logica fra *hosts* (saltando di nodo in nodo).
Il livello di *trasporto* si occupa della comunicazione logica fra *processi* (si appoggia sul livello di rete per mandare i singoli pezzetti).
Il bello del lavorare su più *layers* è proprio l'indipendenza delle singole operazioni (cioè "ognuno si fa gli affari suoi") che però collaborano per supportare ciò che sta sopra. Per esempio: il livello di trasporto ha sopra quello di applicazione e deve lavorare per distinguere fra le varie applicazioni che ci sono sulla macchina chi è quella coinvolta, il livello di rete che sta sopra al livello di trasporto non si deve preoccupare di tutto ciò, ma solo di rapportarsi col livello di trasporto per prendere il pezzo di dato con cui fare il suo lavoro.

2.1.2 I protocolli Internet del livello di trasporto.

Abbiamo parlato dei protocolli Internet per il livello di trasporto, **TCP** e **UDP**. Principali differenze:

- ***TCP***: è affidabile, garantisce (finché può) la consegna dei dati. Si occupa di:
  - controllo del flusso (eventuali attese, dimensioni dei pacchetti, etc.);
- ***UTP***: è

Capitolo 3

Strato di rete

3.1 Rappresentazione dell'informazione

- Sistemi numerici
- Rappresentazione dei numeri interi con e senza segno
- Rappresentazione dei numeri in virgola fissa e mobile
- Rappresentazione dell'informatica non numerica

Capitolo 4

Introduzione ai sistemi operativi

4.1 Rappresentazione dell'informazione

- Sistemi numerici
- Rappresentazione dei numeri interi con e senza segno
- Rappresentazione dei numeri in virgola fissa e mobile
- Rappresentazione dell'informatica non numerica

Capitolo 5

Processi e thread

5.1 Rappresentazione dell'informazione

- Sistemi numerici
- Rappresentazione dei numeri interi con e senza segno
- Rappresentazione dei numeri in virgola fissa e mobile
- Rappresentazione dell'informatica non numerica

Capitolo 6

Scheduling della CPU

6.1 Rappresentazione dell'informazione

- Sistemi numerici
- Rappresentazione dei numeri interi con e senza segno
- Rappresentazione dei numeri in virgola fissa e mobile
- Rappresentazione dell'informatica non numerica

Capitolo 7

Livello di data link e LAN

7.1 Rappresentazione dell'informazione

- Sistemi numerici
- Rappresentazione dei numeri interi con e senza segno
- Rappresentazione dei numeri in virgola fissa e mobile
- Rappresentazione dell'informatica non numerica

Capitolo 8

Reti wireless

8.1 Rappresentazione dell'informazione

- Sistemi numerici
- Rappresentazione dei numeri interi con e senza segno
- Rappresentazione dei numeri in virgola fissa e mobile
- Rappresentazione dell'informatica non numerica

Capitolo 9

Gestione della memoria

9.1 Rappresentazione dell'informazione

- Sistemi numerici
- Rappresentazione dei numeri interi con e senza segno
- Rappresentazione dei numeri in virgola fissa e mobile
- Rappresentazione dell'informatica non numerica

Capitolo 10

File system

10.1 Rappresentazione dell'informazione

- Sistemi numerici
- Rappresentazione dei numeri interi con e senza segno
- Rappresentazione dei numeri in virgola fissa e mobile
- Rappresentazione dell'informatica non numerica

Capitolo 11

Macchine virtuali

11.1 Rappresentazione dell'informazione

- Sistemi numerici
- Rappresentazione dei numeri interi con e senza segno
- Rappresentazione dei numeri in virgola fissa e mobile
- Rappresentazione dell'informatica non numerica