



POLITECNICO DI TORINO

Reti di calcolatori

Appunti del corso da 8 CFU del Prof. Guido Marchetto
e del prof. Claudio Casetti

A.A. 2019/20

Autore: Marco Smorti

Data: Gennaio 2020

INDICE

Prima parte

1. Concetti generali	3
2. Architetture e protocolli	24
3. Quiz	30
4. Protocolli a finestra	32
5. Strato fisico	50
6. Strato collegamento	56
7. Protocolli di accesso per reti locali (LAN)	61
8. Standard per reti locali	67
9. Esercizi CSMA	73

Seconda parte

1. Livello rete	78
2. Strato 3: instradamento	93
3. Domain Name System	99
4. Livello trasporto	104
5. NAT e indirizzamento privato	115
6. Livello applicazione – Parte 1	118
7. Esercizi di Traffic Analysis	123
8. Livello applicazione – Parte 2	126
9. Multimedia networking	132

1. Servizi e funzioni nelle reti di telecomunicazione

Il CCITT (International Telegram and Telephone Consultative Committee) della International Telecommunication Union è l'ente di standardizzazione che dal 1994 ha preso il nome di ITU-T. Le raccomandazioni sono anteposte dalla lettera I (Raccomandazione I.112, ad esempio).

Definizioni ITU-T

- **Comunicazione:** trasferimento di informazioni secondo convenzioni prestabilite.
- **Telecomunicazione:** qualsiasi trasmissione e ricezione di segnali che rappresentano segni, scrittura immagini e suono, informazioni di qualsiasi natura, attraverso cavi, radio o altri sistemi ottici ed elettromagnetici
- **Servizio di telecomunicazione:** ciò che viene offerto da un gestore pubblico o privato ai propri clienti al fine di soddisfare una specifica esigenza di telecomunicazione
- **Funzioni in una rete di telecomunicazioni:** operazioni svolte all'interno della rete al fine di offrire i servizi:
 - **Segnalazione:** scambio di informazioni che riguardano l'apertura, il controllo e la chiusura di connessioni e la gestione di una rete di telecomunicazione
 - **Commutazione:** processo di interconnessione di unità funzionali, canali di trasmissione e circuiti di telecomunicazione per il tempo necessario al trasferimento dei segnali
 - **Trasmissione:** il trasferimento di segnali da un punto a uno o più altri punti
 - **Gestione:** allacciamento nuovi utenti, sostituzione apparati, riconfigurazione per guasti, monitoraggio prestazioni e controllo apparati.

Topologie delle reti di telecomunicazione

- **Mezzo trasmissivo:** mezzo fisico in grado di trasportare segnali tra due o più punti
- **Canale:** singolo mezzo trasmissivo o concatenazione di mezzi trasmissivi
- **Banda:** nella teoria dei segnali è definita come *l'ampiezza spettrale di un segnale o di un canale* mentre nelle reti telematiche è definita come *la quantità di dati (bit) per unità di tempo (secondi)* oppure come *Bit Rate*.
- **Capacità di un mezzo trasmissivo:** massima velocità trasmissiva misurata in bit/s che dipende dalla tecnologia con cui vengono realizzati trasmettitore e ricevitore, ed è limitata superiormente dalla banda fisica del mezzo.
- **Capacità di un canale:** capacità del mezzo trasmissivo a bit rate inferiore tra quelli che lo compongono (funziona da collo di bottiglia)
- **Traffico offerto:** quantità di dati per unità di tempo che una sorgente cerca di inviare in rete
- **Traffico smaltito (throughput):** porzione di traffico offerto che riesce ad essere consegnata correttamente alla destinazione. In generale:

$$\begin{aligned} \text{Throughput} &\leq \text{capacità del canale} \\ \text{Throughput} &\leq \text{traffico offerto} \end{aligned}$$

- **Topologia di una rete di telecomunicazione:** insieme di nodi e segmenti che fornisce un collegamento tra due o più punti per permettere la telecomunicazione tra essi
- **Nodo:** punto in cui avviene la commutazione
- **Segmento:** mezzo di trasmissione o canale
- **Canale punto-punto:** due soli nodi collegati agli estremi del canale che viene utilizzato in modo paritetico
- **Canale multi-punto:** più nodi collegati ad un unico canale: un nodo master e numerosi slave

- **Canale broadcast:** unico canale di comunicazione condiviso da tutti i nodi, l'informazione è inviata da un nodo e ricevuta da tutti gli altri e se i dati trasmessi contengono l'indirizzo del nodo destinazione realizzo di fatto un punto-punto

Una topologia di rete è definita da un grafo $G = (V, A)$ dove V è l'insieme dei vertici e A quello degli archi. Qui di seguito si utilizzerà la notazione N per i vertici (nodi) e C per gli archi (canali).

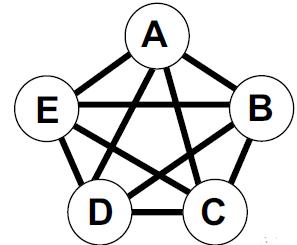
Topologia a maglia completa

$$C = \frac{N(N - 1)}{2}$$

Vantaggi: tolleranza ai guasti poiché vi sono molti percorsi tra due nodi

Svantaggio: elevato numero di canali

In questo tipo di topologia si notano molti percorsi alternativi, di cui solo un percorso diretto (cioè che attraversa un solo canale). Il percorso a minima distanza è di scelta ovvia ed è utilizzata con **pochi nodi**.



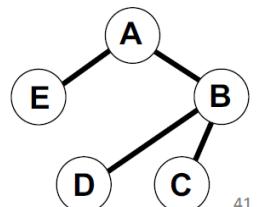
Topologia ad albero

$$C = N - 1$$

Vantaggi: basso numero di canali

Svantaggio: vulnerabilità ai guasti (solo un percorso tra due nodi)

Viene utilizzata per ridurre i costi e semplificare la stesura dei canali e vi è una singola scelta di percorso tra ogni coppia di nodi.



41

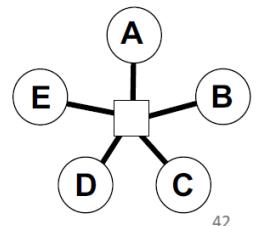
Topologia a stella attiva

$$C = N$$

Vantaggi: basso numero di canali

Svantaggio: vulnerabilità ai guasti del centro stella

Viene utilizzata per ridurre i costi e semplificare la stesura dei canali ed ogni nodo ha un'unica scelta di percorso possibile. La complessità nella scelta dei percorsi è demandata al centro stella e questa topologia viene utilizzata nelle **reti locali**, satellitari e nelle reti radio cellulari.



42

Topologia a stella passiva

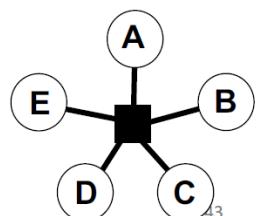
$$C = 1$$

Anche se ci sono N fili il canale è uno soltanto.

Vantaggi: basso numero di canali

Svantaggio: potenzialmente vulnerabile ai guasti del centro stella

Viene utilizzata per ridurre i costi e semplificare la stesura dei canali, ogni nodo ha un'unica scelta di percorso possibile ed esiste nella rete un unico canale (**broadcast**). Viene utilizzata nelle reti locali.



43

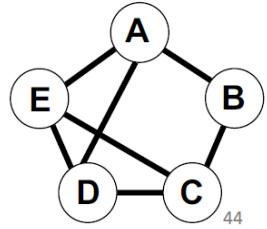
Topologia a maglia

$$N - 1 < C < \frac{N(N - 1)}{2}$$

Vantaggi: tolleranza ai guasti e numero di canali selezionabile a piacere

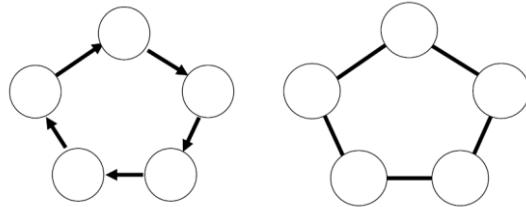
Svantaggio: topologia non regolare

L'instradamento è **complesso**, vi sono numerosi percorsi alternativi ma è la più utilizzata in **internet e telefonia**.



Topologia ad anello

La topologia ad anello può essere *unidirezionale* o *bidirezionale* come segue:



$$C = \frac{N}{2}$$

$$C = N$$

La topologia ad anello viene molto utilizzata in reti locali e metropolitane per costruire topologie magliate (SDH reti di dorsale ottica) ed esistono *uno o due* percorsi possibili per ogni coppia di nodi (senso orario o antiorario).

In caso di guasto l'anello *bidirezionale* assicura la sopravvivenza della rete a capacità dimezzata. L'anello bidirezionale offre un **instradamento alternativo in caso di guasto** (ed è anche il caso più semplice).

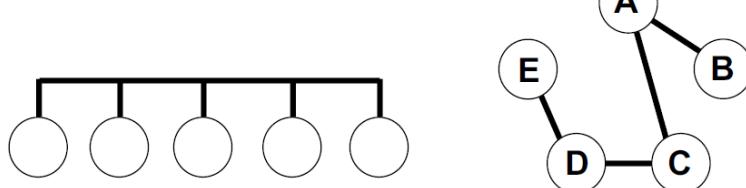
Topologia a bus

Bus attivo (caso particolare di albero): $C = N - 1$

Bus passivo: $C = 1$

Esiste una sola scelta possibile di percorso tra ogni coppia di nodi, viene utilizzata in **reti locali** e metropolitane.

Bus passivo: canale su cui sono agganciate le stazioni. I nodi hanno prestazioni che dipendono dalla posizione (mentre nella topologia a stella ciò non accade).



Topologia fisica e logica

Topologia fisica: tiene conto dei mezzi trasmissivi (si rappresentano, quindi i canali ed i nodi come fatto fino ad ora)

Topologia logica: definisce l'interconnessione tra nodi mediante canali



Topologie e prestazioni

La scelta della topologia ha ricadute sulle prestazioni di una rete. A pari capacità disponibili sui canali, la quantità di traffico smaltibile da una rete dipende: *dalla media della distanza* tra ogni coppia di nodi della rete, pesata dalla quantità di traffico scambiata tra i due nodi secondo la formula

$$\bar{D} = \frac{\sum_i \sum_j d_{ij} \cdot x_{ij}}{\sum_i \sum_j x_{ij}}$$

Avendo definito opportunamente una matrice delle distanze dove ogni coppia di nodi viene identificata con un intero che identifica il numero di nodi distanza. La matrice di traffico (x) riassume le informazioni che i due nodi si scambiano. Il traffico viene misurato in bit/s e tipicamente un nodo non scambia informazioni con sé stesso, perciò $x_{ii} = 0$.

Traffico uniforme: ogni elemento della matrice di traffico (x_{ij}) è uguale agli altri.

Esempi

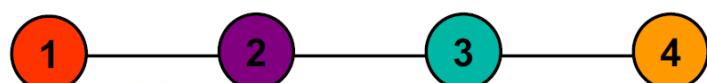
Si effettua il confronto tra alcune topologie, con pari numero di nodi (4) e quasi a pari numero di canali. Il traffico si suppone uniforme: ogni coppia di nodi scambia t bit/s ed il traffico totale generato è **12t**.

La matrice di traffico avrà dunque la forma:

$$X = \begin{bmatrix} x_{11} & x_{12} & x_{13} & x_{14} \\ x_{21} & x_{22} & x_{23} & x_{24} \\ x_{31} & x_{32} & x_{33} & x_{34} \\ x_{41} & x_{42} & x_{43} & x_{44} \end{bmatrix} = \begin{bmatrix} 0 & t & t & t \\ t & 0 & t & t \\ t & t & 0 & t \\ t & t & t & 0 \end{bmatrix}$$

Ogni canale unidirezionale ha la capacità di **B** bit/s. Si vuole calcolare: *distanza media, capacità di rete, carico massimo sul canale, carico massimo sul nodo*.

- **Bus attivo di 4 nodi**



$$D = \begin{bmatrix} d_{11} & d_{12} & d_{13} & d_{14} \\ d_{21} & d_{22} & d_{23} & d_{24} \\ d_{31} & d_{32} & d_{33} & d_{34} \\ d_{41} & d_{42} & d_{43} & d_{44} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 2 & 3 \\ 1 & 0 & 1 & 2 \\ 2 & 1 & 0 & 1 \\ 3 & 2 & 1 & 0 \end{bmatrix}$$

Distanza media:

$$\bar{D} = \frac{(1t + 2t + 3t) + (1t + 1t + 2t) + (2t + 1t + 1t) + (3t + 2t + 1t)}{12t} = \frac{20t}{12t} = 1,66$$

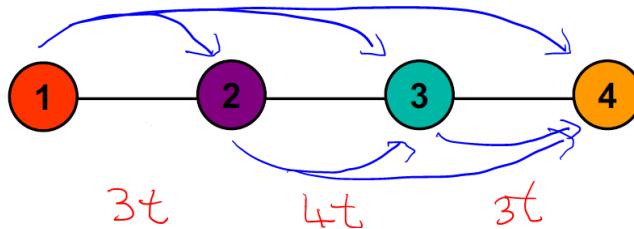
Capacità:

$$C = (n \text{ canali}) \times (\text{capacità singolo canale}) = 3 \times 2B = 6B$$

La capacità di ogni singolo canale è $2B$ perché è bidirezionale.

Carico massimo su canale:

Considero soltanto il traffico da sinistra verso destra:



Conto il numero di linee che vedo “passare” su ogni canale. Ottengo quindi un sistema (considero traffico unidirezionale, quindi capacità B):

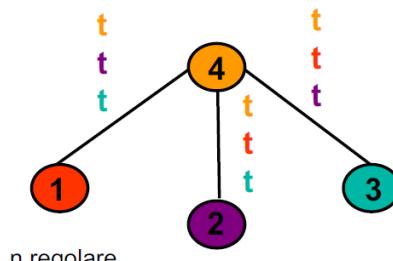
$$\begin{cases} 3t \leq B \\ 4t \leq B \rightarrow \text{prendo la più stringente} \rightarrow 4t \leq B \rightarrow t \leq \frac{B}{4} \\ 3t \leq B \end{cases}$$

Carico massimo su nodo:

Guardo il disegno sopra e vedo che sia il nodo 2 che il nodo 3 si ritrovano: da sinistra $3t$ e da destra $4t$, perciò il carico massimo sul nodo sarà **7t**. Sostituisco all’equazione sopra

$$7t = 7 \cdot \frac{B}{4} = \frac{7}{4}B$$

- Albero



Capacità: $3 \times 2B = 6B$

Distanza media:

$$D = \begin{bmatrix} d_{11} & d_{12} & d_{13} & d_{14} \\ d_{21} & d_{22} & d_{23} & d_{24} \\ d_{31} & d_{32} & d_{33} & d_{34} \\ d_{41} & d_{42} & d_{43} & d_{44} \end{bmatrix} = \begin{bmatrix} 0 & 2 & 2 & 1 \\ 2 & 0 & 2 & 1 \\ 2 & 2 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

$$\bar{D} = \frac{[(2+2+1) + (2+2+1) + (2+2+1) + (1+1+1)]t}{12t} = \frac{18}{12} = 1.5$$

Carico massimo sul canale:

Per il traffico in uscita dal nodo centrale il carico massimo sul canale vale $3t$ (uniforme), perciò si avrà:

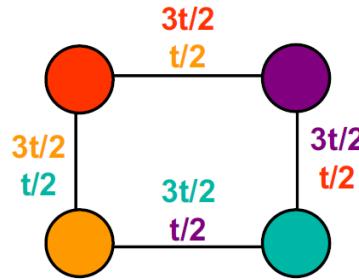
$$3t \leq B \rightarrow t \leq \frac{B}{3}$$

Carico massimo sul nodo:

Il nodo centrale (4) riceve $3t$ da tutti gli altri nodi, perciò avrà da gestire $9t$, dunque:

$$9t = 9 \cdot \frac{B}{3} = 3B$$

- **Anello**



$$D = \begin{bmatrix} d_{11} & d_{12} & d_{13} & d_{14} \\ d_{21} & d_{22} & d_{23} & d_{24} \\ d_{31} & d_{32} & d_{33} & d_{34} \\ d_{41} & d_{42} & d_{43} & d_{44} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 2 & 1 \\ 1 & 0 & 1 & 2 \\ 2 & 1 & 0 & 1 \\ 1 & 2 & 1 & 0 \end{bmatrix}$$

Per la matrice delle distanze si è scelto il percorso più breve.

Capacità: $4 \times 2B = 8B$

Distanza media:

$$\bar{D} = \frac{[(1+2+1)+(1+1+2)+(2+1+1)+(1+2+1)]t}{12t} = \frac{16}{12} = 1.33$$

Carico massimo sul canale:

Per il traffico in uscita in senso orario il carico massimo su un canale vale $2t$, perciò si avrà:

$$2t \leq B \rightarrow t \leq \frac{B}{2}$$

Carico massimo sul nodo:

Ogni nodo vede $4t$ di traffico, perciò:

$$4t = 4 \cdot \frac{B}{2} = 2B$$

Ogni nodo gestisce fino a $2B$ di traffico.

Servizi nelle reti di comunicazioni

Rispetto ai servizi offerti, le reti di telecomunicazione possono essere:

- **Dedicate** se offrono un solo servizio come radio e televisione
- **Integrate** se offrono una molteplicità di servizi come internet

Classificazione dei servizi di telecomunicazione

I servizi di telecomunicazione si dividono in:

- **Servizi portanti**: che forniscono la possibilità di **trasmettere il segnale** tra interfacce utente-rete (come il collegamento ADSL)
- **Teleservizi**: forniscono la **completa possibilità di comunicazione** tra utenti, includendo le funzioni degli apparati di utente, secondo protocolli concordati da gestori pubblici o privati (come la telefonia). I teleservizi a loro volta si dividono in:
 - **Servizi di base** che forniscono all'utente un servizio di telecomunicazione con le minime funzionalità richieste dal servizio stesso (come la telefonia di base e la televisione, o la posta elettronica)
 - **Servizi supplementari** che forniscono all'utente un servizio con funzionalità aggiuntive rispetto al corrispondente servizio di base (ad esempio per la telefonia: l'avviso di chiamata, richiamo su occupato, trasferimento di chiamata, etc..) quindi modifica o integra un servizio di base.

Flussi di informazione nei teleservizi

Il flusso di informazioni può essere: **bidirezionale simmetrico**, **bidirezionale asimmetrico** o **unidirezionale**. Oppure: **punto – punto**, **punto – multipunto**, **multipunto – multipunto**.

Classificazione dei (tele)servizi

- **Diffusivi**
 - **Senza controllo di presentazione all'utente**: diffusione di un flusso di informazioni da una sorgente a un numero qualsiasi di ricevitori autorizzati collegati alla rete. Gli utenti **non possono** controllare l'inizio e l'ordine della presentazione delle informazioni (TV, pay TV, radio).
 - **Con controllo di presentazione all'utente**: diffusione di un flusso di informazioni da una sorgente a un numero qualsiasi di ricevitori autorizzati collegati alla rete dove però gli utenti **possono** controllare l'inizio e l'ordine della presentazione delle informazioni (televideo, on-demand).
- **Interattivi**
 - **Conversazionali**: sono servizi di comunicazione con trasferimento di informazioni da estremo a estremo in **tempo reale** come ad esempio il video e il suono nella telefonìa (o videoconferenza) oppure i dati (trasferimento di file, chat e giochi multiplayer)
 - **Di messaggistica**: sono servizi di comunicazione da estremo a estremo tramite apparati per la memorizzazione di informazioni. Non sono quindi in tempo reale e possono essere: video e suono (segreteria telefonica, fotografie, videoclip) e dati (e-mail, sms).
 - **Di consultazione e reperimento**: sono servizi in cui l'informazione è immagazzinata in centri di informazione di uso pubblico (web, video streaming, teledidattica).

Modalità di realizzazione dei teleservizi

La terminologia è derivata dal mondo di internet e si differenziano per le modalità di comportamento degli applicativi utente.

- **Modello client-server:** prevede due ruoli ben distinti quali
 - **Client:** inizia l'interazione con il server richiedendo un servizio (es. richiesta di una pagina www). Il client è attivato nel momento in cui l'utente richiede un servizio e **si disattivano** quando tale servizio è terminato.
 - **Server:** fornisce al client il servizio (invia la pagina www richiesta) ed è attivato al momento dell'accensione dell'elaboratore su cui sono stati installati e sono **sempre disponibili** e in attesa di richieste da parte dei client.

L'informazione è contenuta prevalentemente nel server. La maggior parte delle applicazioni in internet seguono questo modello.

- **Modello peer-to-peer:** è pensato principalmente per l'**interazione tra gruppi di utenti** e tutti gli applicativi utenti sono **paritetici**, cioè mettono a disposizione informazioni **condivise**. Si ha la creazione di reti overlay (reti logiche) tra utenti. Realmente non sono P2P ma l'entità in comunicazione agisce simultaneamente come client e come server.

Trasmissione nelle reti di telecomunicazioni

L'informazione può essere trasmessa in maniera **analogica** o **numerica** (digitale). L'informazione analogica viene trasferita per mezzo di un segnale elettrico continuo, limitato e di infiniti possibili valori, mentre quella numerica è trasferita per mezzo di un segnale elettrico discontinuo, limitato e con un numero finito di valori.

Nelle reti telematiche l'informazione è trasferita in forma **digitale** usando segnali analogici e digitali. Il primo tratto dell'informazione è analogico, ma viene poi convertito in numerico tramite il processo di numerizzazione (che comprende campionamento e quantizzazione).

Tipi di trasmissione

- **Parallela:** l'informazione è trasferita su più canali che collegano i due nodi. Vi è quindi una molteplicità di canali che può essere utilizzata contemporaneamente. È necessario un **clock della rete** perché è necessario conoscere l'istante di tempo in cui andare ad ascoltare il segnale.
- **Seriale:** l'informazione viene trasferita un bit alla volta su un unico canale in maniera serializzata. Anche in questo caso è necessario individuare l'**inizio della trasmissione**.
- **Asincrona:** ogni byte di informazioni è trasmesso separatamente dagli altri. Il clock di ricezione è nominale ed uguale a quello di trasmissione e la prima parte del segnale non trasmette dati ma sincronizza inviando un segnale (di "avviso di inizio"). È necessaria tale operazione ogni volta.
- **Sincrona:** La temporizzazione è sempre presente e le informazioni sono strutturate in **trame** (o slot). Il trasmettitore e il ricevitore sincronizzano i loro clock prima della trasmissione e li mantengono sincronizzati per tutta la durata della trama.

Modi di trasferimento nelle reti di telecomunicazioni

- **Condivisione di canale:** tramite **multiplazione** e **accesso multiplo**
- **Condivisione di nodo:** tramite **commutazione**.

Condivisione di canale

La condivisione di canale può avvenire tramite:

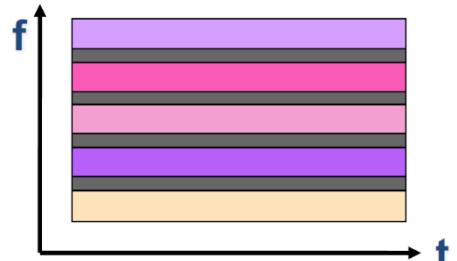
- **Multiplazione:** se tutti i flussi sono disponibili in un unico punto. Dato un canale si ha un unico ingresso ed un'unica uscita.
- **Accesso multiplo:** se i flussi accedono al canale da punti differenti.

Il canale è una risorsa che si può utilizzare in tempi diversi e quando la si utilizza si ha l'esclusività di tale canale, oppure siccome è possibile trasmettere segnali a diverse frequenze, lo si può utilizzare in contemporanea.

Multiplazione di frequenza

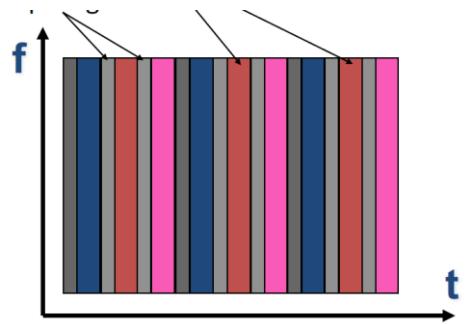
Il canale viene diviso in frequenze e quando si riceve un segnale lo si preleva alla frequenza desiderata. Le frequenze sono sempre separate da porzioni di bande di guardia, in cui non vi è segnale. Quest'ultima evita lo sconfinamento dovuto all'effetto doppler di un ricevitore in movimento. La banda di frequenza è proporzionale alla velocità di trasmissione.

Questo tipo di multiplazione (ma anche quello di tempo) ha bisogno di un controllore che gestisca le frequenze.



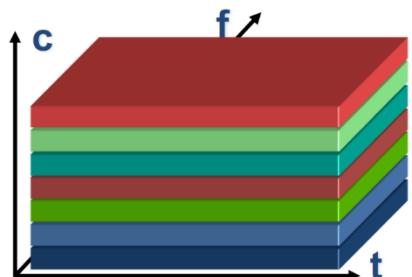
Multiplazione di tempo

Il canale è condivisibile nel dominio del tempo. Una sorgente riceve l'uso del canale solo per un tempo definito. Avrà di nuovo il canale solo dopo un certo lasso di tempo. Si ha quindi una grossa fetta di banda, ma per un intervallo limitato. In questo caso sono presenti dei **tempi di guardia** dove non si trasmette nulla. Questo tipo di multiplazione è più veloce della precedente.



Multiplazione di codice

Veniva utilizzata nel 3G. Soluzione di condivisione del canale dove ogni utente può utilizzare il canale senza limiti di tempo e a qualsiasi banda, mascherando l'informazione per non farla ricevere ad altre stazioni. Non è ottima perché i codici che si possono utilizzare sono finiti e limitati. Immaginando uno spazio a n dimensioni, ho n codici disponibili, perciò il numero è molto limitato. Questo perché utilizzando la tecnica della multiplazione di codice vengono utilizzati **segnali ortogonali** (si fa prodotto tra bit di informazione e codice, che equivale a un prodotto scalare tra vettori). Il vantaggio è la resistenza ai disturbi.



Multiplicazione di spazio

Le reti permettono di sfruttrare la diversità spaziale del sistema per far coesistere **più flussi di informazione in punti diversi**. Il territorio viene diviso in celle (cioè l'area intorno ad una antenna) e quando un ricevitore si sposta, non sente più l'antenna da cui partiva ma si aggancia ad un'altra. In questo modo viene utilizzata la stessa frequenza ma in più punti diversi senza collisioni.

Multiplicazione statistica

La multiplazione statistica è **una modalità** con cui si possono utilizzare tutte le precedenti multiplazioni. Infatti la multiplazione può essere **predeterminata** (cioè statica, fissa) oppure **statistica**. In quest'ultimo caso le bande vengono allocate ma non utilizzate. Nominalmente una frequenza appartiene a un certo ente X, ma quest'ultimo non trasmette. In questo modo è possibile far utilizzare la frequenza ad un altro ente Y. In questo modo si ha una sola banda ma due sorgenti, scommettendo tutto sul fatto che la probabilità che X e Y trasmettano contemporaneamente sia nulla.

Condivisione di nodo

In questo caso il nodo deve decidere cosa fare alla ricezione dei vari flussi. La **commutazione** può essere:

- Di **circuito**: se i flussi sono continui (come la telefonia)
- Di **pacchetto**: se i flussi sono intermittenti (trasmissione dati)

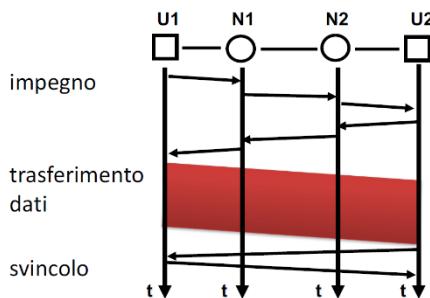
La commutazione, secondo l'**ITU-T** è: il **processo di interconnessione di unità funzionali, canali di trasmissione o circuiti di telecomunicazione per il tempo necessario per il trasferimento dei segnali**.

Più facilmente: il processo di allocazione delle risorse di rete necessarie per il trasferimento dell'informazione.

Commutazione di circuito

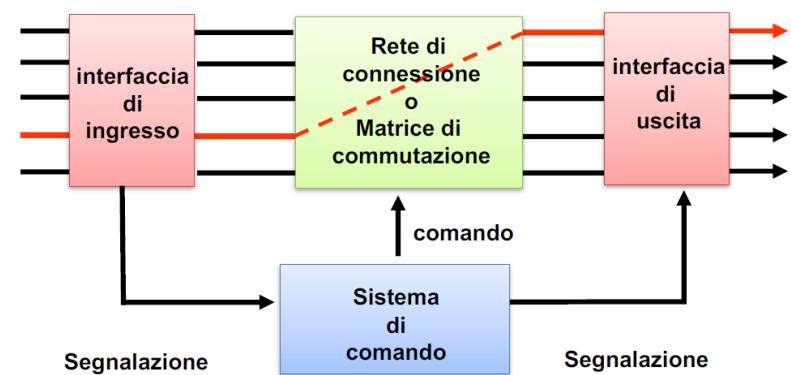
Viene utilizzata dagli albori della telefonia. La rete utilizza le risorse disponibili per allocare un circuito ad ogni richiesta di servizio. Un circuito consiste nel collegamento fisico tra i due terminali utente. Si ricordi l'operatore che molto tempo fa doveva fare lo switch dei cavi per permettere le telefonate.

Nella commutazione di circuito, quest'ultimo è di **uso esclusivo** dei due utenti per tutta la durata della comunicazione. Le risorse vengono rilasciate soltanto al **termine** della comunicazione, su indicazione degli utenti.



Quello a sinistra è il diagramma spazio-temporale. La parte relativa all'impegno è la parte in cui l'utente inizia ad accordarsi con la rete tramite ripetute **segnalazioni**. Una volta terminato l'impegno i dati non si fermano più ad ogni nodo ma scorrono diretti e ad uso esclusivo, fino allo svincolo in cui il canale viene liberato.

L'interfaccia di ingresso sente arrivare la chiamata su una linea e attiva il sistema di comando, che viene avvertito con una segnalazione (es. numero da chiamare). La segnalazione attiva una matrice di commutazione che mette in comunicazione la linea di ingresso con quella di uscita. La linea resta esclusiva.



Tra i vantaggi della commutazione di circuito troviamo:

- Banda costante garantita
- Ritardi di trasferimento costanti, poiché escludendo i ritardi iniziali, una volta stabilito il collegamento il ritardo è costante.
- Trasparenza del circuito: i nodi non possono modificare i dati o cambiare la velocità di trasmissione.
- Bassi ritardi nell'attraversamento dei nodi perché i dati **non** si fermano nei nodi intermedi una volta stabilita la connessione.

Tra gli svantaggi:

- Risorse dedicate ad una sola comunicazione
- Efficienza buona solo per sorgenti **non** intermittenti (ha senso solo per sorgenti che trasmettono continuamente dati)
- Tempo di apertura del circuito
- Nessuna conversione di formati, velocità e protocolli
- Tariffazione in base al tempo di esistenza del circuito

Commutazione di pacchetto

Questo è il paradigma di commutazione utilizzato per il traffico dati. Non vengono più allocate risorse per l'uso esclusivo di due o più utenti. Le informazioni vengono spostate su un canale che può essere usato in momenti diversi. Le risorse vengono fornite per un breve periodo di tempo all'utente che ne ha bisogno. Il funzionamento è analogo a quello del sistema postale.

L'informazione viene scritta e "imbucata", in attesa del prelievo (1° stop). Una volta prelevata raggiunge la stazione di partenza (2° stop). Una volta prelevata viene inviata alla stazione di arrivo (3° stop) e prelevata fino alla consegna del destinatario. Non viene tenuta memoria delle informazioni inviate, perciò ad ogni successivo invio bisogna ripetere il percorso.

L'informazione da trasferire è organizzata in unità dati (**PDU**) che contiene sia informazione di utente che di controllo, in altre parole l'informazione viene correlata da una intestazione.

La PDU è costituita da **PCI** (Protocol control information) che rappresenta l'Header ovvero l'intestazione e dalla **SDU** (Service data unit) ovvero l'informazione. L'unità dati è una terminologia ISO e può avere numerose altre definizioni: pacchetto, cella, datagramma, segmento, messaggio, trama (in base al livello in cui ci si trova nella pila ISO-OSI).

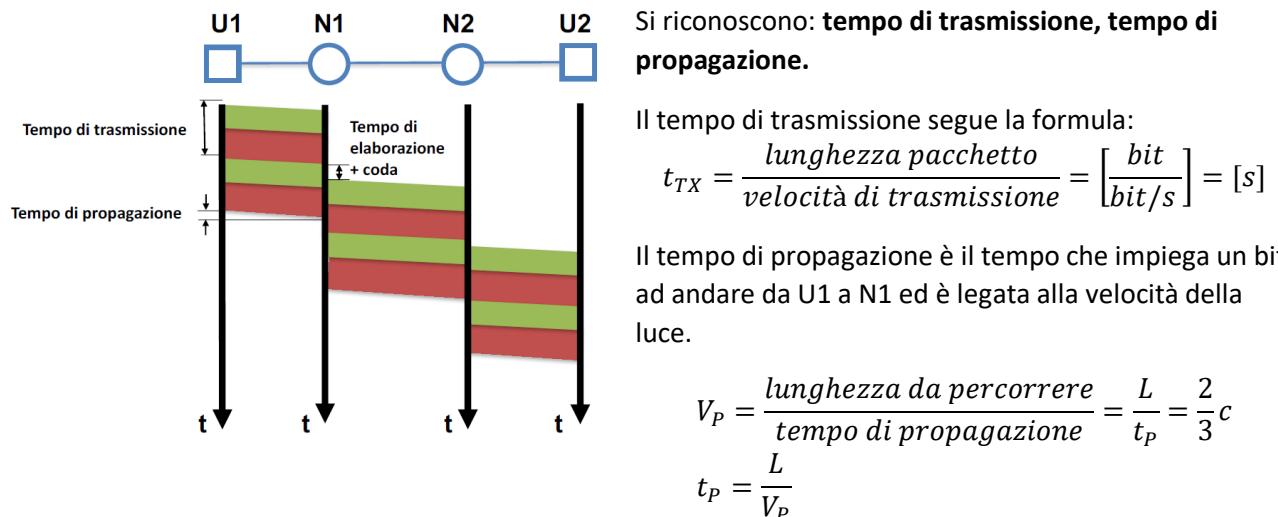
In rete a commutazione a pacchetto, i pacchetti transitano dal mittente fino al primo nodo di commutazione perché il funzionamento è tale che la fermata è necessaria perché tale nodo non sa dove il pacchetto deve andare e da dove arriva, perciò deve fermarlo e leggerne l'intestazione. Questo meccanismo si chiama **STORE AND FORWARD**. Occorre disporre quindi dell'intestazione prima di effettuare l'instradamento (che richiede del tempo) ed è anche necessaria una forma di protezione dagli errori (bit di parità) ed inoltre le diverse capacità dei mezzi trasmissivi che richiedono trasmissioni con bit rate diversi possono generare delle code.



L'elaborazione prende i dati dalla memoria e li mette sulla corretta coda di uscita, che è necessaria visto che i nodi possono viaggiare a velocità differenti. La memorizzazione può essere presente: alle uscite, agli ingressi oppure su entrambe le parti.

L'informazione di un utente può dover essere **frazionata** in molti pacchetti ed ognuno di questi pacchetti avrà una propria intestazione ed una lunghezza (che può essere fissa o variabile).

In questo caso, il diagramma spazio-temporale è notevolmente diverso dal precedente tipo di commutazione.



Dove "c" è la velocità della luce ($3 \cdot 10^8 \text{ m/s}$)

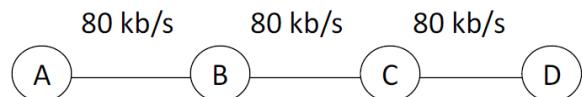
Dal grafico si deduce che i ritardi subiti da un pacchetto nel suo percorso sono molteplici:

- Ritardo su ogni **canale** in cui è trasmesso (ritardo di trasmissione e di propagazione)
- Ritardo in ogni **nodo** di commutazione (ritardo di elaborazione e di accodamento)

La dimensione dei pacchetti non è una scelta univoca, ma vi sono diverse scelte. La prima è la **possibilità di parallelizzazione (pipeline)** cioè:

Supponiamo di avere una rete composta da 4 nodi dove A trasmette un file a D attraversando i nodi B e C e facciamo le seguenti ipotesi:

- Tempo propagazione ed elaborazione trascurabile (= 0)
- Intestazioni pacchetti trascurabili
- Store & Forward ai nodi intermedi

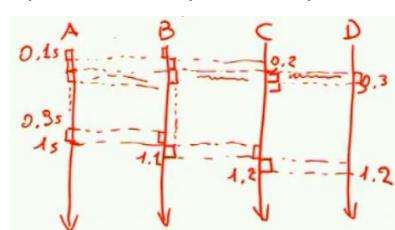


Qual è il tempo necessario del file per raggiungere la destinazione D?

Distinguo il caso in cui è un unico pacchetto da 10kb e più pacchetti da 1kb (la quantità di bit è la stessa perché sto escludendo l'intestazione)

	$t_{TX,1} = \frac{10kB}{80kb/s} = \frac{80000 \text{ bit}}{80000 \text{ bit/s}} = 1 \text{ s} \rightarrow t_{TX,1} = 1 \text{ s} + 1 \text{ s} + 1 \text{ s} = 3 \text{ s} \text{ (store&forward)}$
	$t_{TX,2} = \frac{P}{V_{TX}} = \frac{1kB}{80kb/s} = \frac{8000 \text{ bit}}{80000 \text{ bit/s}} = 0,1 \text{ s} \text{ (singolo pacchetto da 1kB)}$

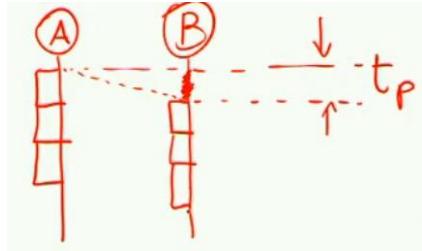
Nel secondo caso, il segmento del canale tra A e B, dopo aver fatto passare il primo pacchetto, non resta ad aspettare che esso arrivi a D, bensì trasmette subito il pacchetto successivo e così via. La differenza con prima è che adesso la rete viene utilizzata contemporaneamente in più punti diversi (ed è da qui che viene il concetto di **parallelismo**). L'ultimo pacchetto verrà inviato all'istante 0.9s e verrà trasmesso da A a B all'istante 1s.



L'ultimo pacchetto sarà quindi ricevuto da C all'istante 1.1s e da D all'istante 1.2s.

Se è presente il tempo di propagazione, il grafico rosso precedente viene semplicemente inclinato (e non è più rettilineo). Sarà quindi presente il ritardo di propagazione che si conta **una sola volta** nel treno di pacchetti (perché comunque la parallelizzazione continua a funzionare) **per ogni canale**.

Osservando la figura, si nota come il tempo di propagazione viene contato soltanto all'inizio.



$$t_{RX,B} = t_{P,AB} + 3T_{TX}$$

Ritardo di pacchettizzazione

Pacchetti brevi riducono il ritardo di pacchettizzazione (questo è importante per il traffico della voce su rete).

Percentuale di informazione di controllo

Se però i pacchetti si fanno troppo piccoli, mi troverò nella situazione in cui l'intestazione è molto più grande dell'informazione in sé. Pacchetti lunghi riducono la percentuale di informazione di controllo che, avendo una PCI di i bit e una SDU di dimensione s bit, la frazione di informazione di controllo sarà:

$$\frac{i}{s+i}$$

Che deve essere sempre piccola (ovvero pacchetto grande, visto che $s \gg i$ riduce la frazione di cui sopra).

Probabilità di errore

Se vi è un errore sul pacchetto, il ricevitore dovrà scartare l'intero pacchetto. In questo modo è necessario re-inviare l'intero pacchetto, perciò anche in questo caso è conveniente utilizzare pacchetti **brevi**.

Supponendo di avere un pacchetto di n bit, un canale con errori indipendenti e probabilità di errore sul bit uguale a p , la probabilità che un pacchetto sia **corretto** è:

$$(1-p)^n$$

Vantaggi rispetto alla commutazione di circuito

- Utilizzo **efficiente** delle risorse anche in presenza di traffico intermittente, perché utilizzo il nodo solo nel momento in cui parte il pacchetto.
- Posso controllare la **correttezza** del pacchetto in **ogni nodo**, perciò se durante l'invio si compromette viene ritrasmesso subito.
- E' possibile effettuare conversioni di velocità, formati e protocolli (es. rendere protetto il contenuto in un tratto)
- Le tariffazioni sono in funzione del traffico trasmesso (e non alla durata)

Svantaggi rispetto alla commutazione di circuito

- E' difficile ottenere garanzie di banda (cioè la velocità di trasmissione tra i nodi potrebbe variare a causa di congestioni)
- Ogni nodo deve elaborare il pacchetto
- Il ritardo di trasferimento è variabile

Modi di trasferimento

In una rete a commutazione di pacchetto i modi possono essere:

- **Datagram**
- **Circuito virtuale**

La modalità **datagram** è quella precedentemente illustrata (in cui il pacchetto effettua store and forward attraverso ogni nodo).

La modalità **circuito virtuale** ha l'obiettivo di emulare il più possibile la commutazione di circuito.

Quest'ultima ha la caratteristica di andare in ogni nodo per creare il circuito e siccome con la commutazione di pacchetto non è possibile riservare il canale (circuito) si effettua un **accordo preliminare** non vincolante tra i due interlocutori e i nodi intermedi. Si tratta di un accordo stipulato durante l'apertura della connessione tramite delle segnalazioni, dove il nodo sorgente informa tutti i nodi che riceveranno dei pacchetti con una particolare etichetta nell'intestazione (nel datagram c'è solo l'indirizzo di destinazione). In questo modo i nodi potranno immediatamente spedire il pacchetto poiché si saranno scritte questa etichetta e conosceranno a priori il nodo al quale inviare il pacchetto. Ogni nodo resta comunque condiviso. L'unica caratteristica è che tutti i pacchetti faranno lo stesso percorso per tutta la durata della connessione, finché essa non viene **chiusa** avvisando i nodi di cancellare l'etichetta perché non più necessaria.

Nella modalità **datagram** non esiste la suddivisione della comunicazione in tre fasi, visto che non c'è alcun accordo preliminare sulla fornitura del servizio e pacchetti diversi ma con identica sorgente e destinazione possono seguire percorsi diversi.

Vantaggi del circuito virtuale rispetto al datagram:

- Mantenimento della sequenza
- Minor variabilità dei ritardi
- Instradamento solo in fase di apertura della connessione

Nel datagram ogni pacchetto necessita della coppia sorgente/destinazione (identificatore globale) mentre nei circuiti virtuali è sufficiente **identificare** il circuito virtuale (anche tramite identificatori locali in ogni tratto). La coppia sorgente-destinazione viene utilizzata solo in fase di apertura del circuito.

ESERCIZIO 1

Commutazione di pacchetto - 1

Si consideri una topologia di rete lineare composta da un singolo canale di capacità pari a 500 kbit/s.



Il nodo S deve trasmettere un file di 8800 byte verso il nodo D. Si supponga che:

- la rete sia scarica e non vi siano errori di trasmissione
- il tempo di propagazione sul canale sia pari a 5 ms
- Il tempo di elaborazione nei nodi di commutazione sia trascurabile
- la dimensione massima dei pacchetti trasmessi sul canale sia di 1500 byte (si trascurino le intestazioni)

Si determini il tempo necessario a D per ricevere il file

Rete scarica: non vi sono altri pacchetti in transito. Si considera la connessione dell'esercizio come unica.

Si immagina che le intestazioni siano nulle.

1) La **divisione** viene fatta ipotizzando di dividere il file in pacchetti di dimensione massima finché si arriva al fondo (che non avrà dimensione massima).

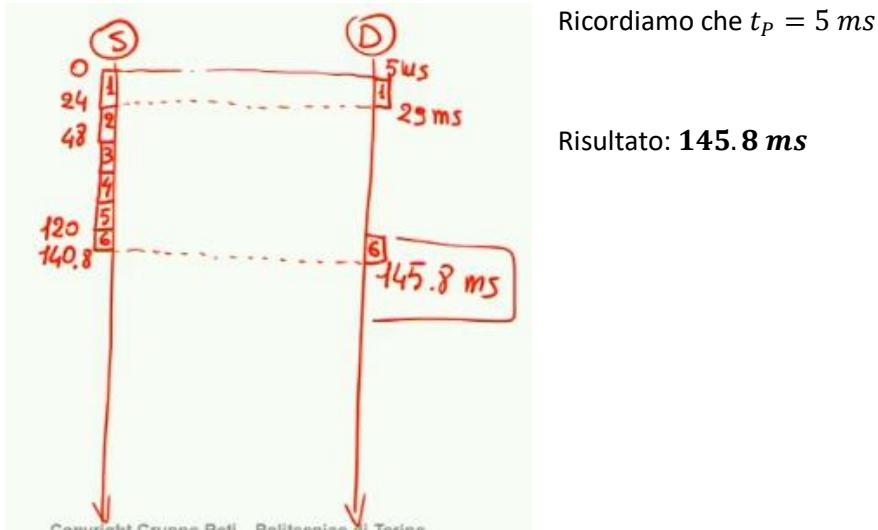
$$5 \times 1500B + 1 \times 1300B = 8800B$$

2) Tempo di trasmissione

$$T_{TX,1} = \frac{1500B \times 8}{500 \cdot 10^3} = 24 \text{ ms}$$

$$T_{TX,2} = \frac{1300B \times 8}{500 \cdot 10^3} = 20.8 \text{ ms}$$

3) Diagramma spazio temporale



ESERCIZIO 1B

Commutazione di pacchetto – 1b

Si consideri una topologia di rete lineare composta da un singolo canale di capacità pari a 500 kbit/s.



Il nodo S deve trasmettere un file di 8800 byte verso il nodo D. Si supponga che:

- la rete sia scarica e non vi siano errori di trasmissione
- il tempo di propagazione sul canale sia pari a 5 ms
- Il tempo di elaborazione nei nodi di commutazione sia trascurabile
- Ogni pacchetto riceve 100 byte di intestazione
- la dimensione massima dei pacchetti trasmessi sul canale sia di 1500 byte

Si determini il tempo necessario a D per ricevere il file

L'unica differenza è la presenza dell'intestazione nei pacchetti.

1) La **divisione** viene fatta ipotizzando di dividere il file in pacchetti di (dimensione massima-intestazione) per farci stare l'intestazione. Taglio quindi il file da blocchi di 1400B (e non più da 1500) e poi aggiungo 100B per ottenere il pacchetto e resteranno 400B alla fine (che diventeranno 500 con l'intestazione).

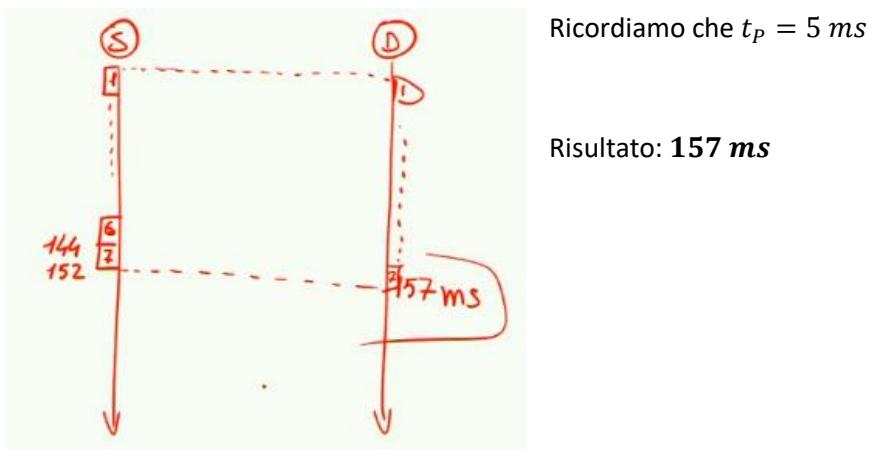
$$6 \times 1500B + 1 \times 500B = 8800B$$

2) Tempo di trasmissione

$$T_{TX,1} = \frac{1500B \times 8}{500 \cdot 10^3} = 24 \text{ ms}$$

$$T_{TX,2} = \frac{500B \times 8}{500 \cdot 10^3} = 8 \text{ ms}$$

3) Diagramma spazio temporale



ESERCIZIO 2

Commutazione di pacchetto - 2

Si consideri una topologia di rete lineare composta da due canali di capacità pari a 500 kbit/s ciascuno.



Il nodo S deve trasmettere un file di 8000 byte verso il nodo D attraverso un nodo intermedio store-and-forward. Si supponga che:

- la rete sia scarica e non vi siano errori di trasmissione
- i tempi di propagazione su ciascun canale siano pari a 30 ms per il primo e a 2 ms per il secondo canale
- Il tempo di elaborazione nei nodi di commutazione sia trascurabile
- la dimensione massima dei pacchetti trasmessi sul canale sia di 1500 byte (si trascurino le intestazioni)

Si determini il tempo necessario perché D riceva tutto il file

Il nodo N effettua store and forward.

1) La **divisione** viene fatta ipotizzando di dividere il file in pacchetti di dimensione massima

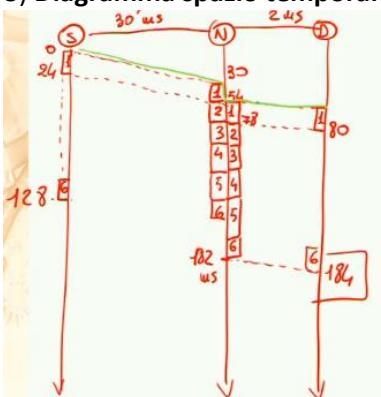
$$5 \times 1500B + 1 \times 500B = 8000B$$

2) **Tempo di trasmissione**

$$T_{TX,1} = \frac{1500B \times 8}{500 \cdot 10^3} = 24 \text{ ms}$$

$$T_{TX,2} = \frac{500B \times 8}{500 \cdot 10^3} = 8 \text{ ms}$$

3) Diagramma spazio temporale



Sul nodo N a sinistra i pacchetti che arrivano, a destra quelli che partono.

Il valore 182 ms si ottiene come:

$$128 \text{ ms} + 30 \text{ ms} (t_P) + 24 \text{ ms} (T_{TX,1}) = 182 \text{ ms}$$

Risultato: **184 ms**

Il gradino verde è da mostrare chiaramente durante l'esame.

ESERCIZIO 2A

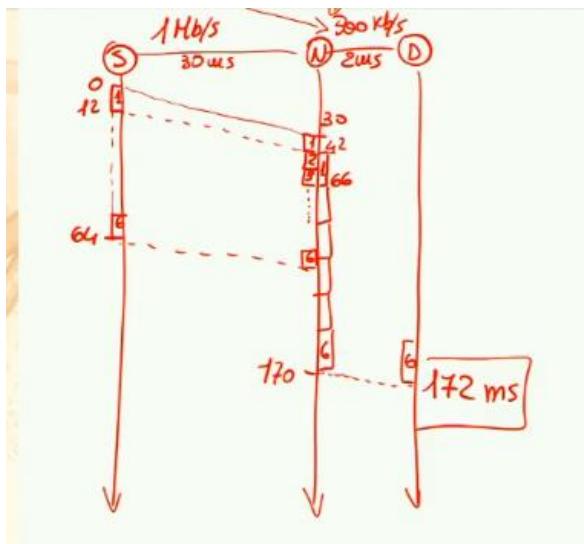
Commutazione di pacchetto – 2a

Alcune varianti degli esercizi precedenti:

1. Assumere che il primo canale abbia capacità 1Mb/s e il secondo 500 kb/s
2. Assumere che il primo canale abbia capacità 500 kb/s e il secondo 1 Mb/s
3. Il tempo di elaborazione sul nodo N sia pari a 2ms per pacchetto (con la rete dell'esercizio originale)

1. I tempi saranno dimezzati perché la velocità è il doppio, perciò

$$T_{TX,1}^1 = 12 \text{ ms} \quad T_{TX,2}^1 = 4 \text{ ms}$$



1. Il primo canale avrà dei tempi di trasmissione più veloci (da ricalcolare) mentre il secondo li avrà uguali all'esercizio precedente.

2. Questa volta si invertono i tempi di trasmissione
3. Si somma sul nodo N un ritardo di 2 ms.

$$T_{TX,1}^2 = 24 \text{ ms} \quad T_{TX,2}^2 = 8 \text{ ms}$$

L'ultimo pacchetto arriva al nodo N all'istante

$$64 \text{ ms} + 30 \text{ ms} = 94 \text{ ms}$$

Però ripartirà molto dopo, visto che il secondo tratto è più lento (torna ad essere trasmesso a 24 ms).

$$\begin{aligned} 42 \text{ ms} + 24 \text{ ms} &= 66 \text{ ms} \text{ (1° pacchetto)} \\ 42 \text{ ms} + 128 \text{ ms} &= 170 \text{ ms} \text{ (ultimo pacchetto)} \end{aligned}$$

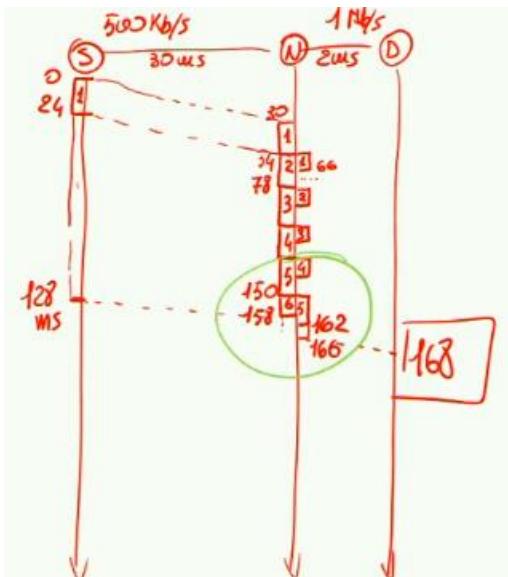
Il risultato sarà

$$170 \text{ ms} + 2 \text{ ms} = 172 \text{ ms}$$

2. Questa volta si avrà

$$T_{TX,1}^1 = 24 \text{ ms} \quad T_{TX,2}^1 = 8 \text{ ms}$$

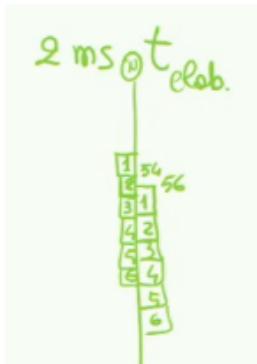
$$T_{TX,1}^2 = 12 \text{ ms} \quad T_{TX,2}^2 = 4 \text{ ms}$$



La conseguenza è che se prima la velocità maggiore dava tutti i pacchetti pronti al nodo di commutazione, adesso non sarà più così perché quando finisce la trasmissione del primo canale (che è più lento) il secondo canale avrà già finito da tempo.

Si nota che appena N avrà trasmesso il pacchetto 5, il pacchetto 6 sarà già lì (perché la trasmissione dura 8 s) quindi lo invia subito.

3. L'effetto del tempo di elaborazione è semplicemente che, quando arriva il primo pacchetto il secondo partirà con 2 ms di ritardo.



Tale tempo viene ripetuto per ogni pacchetto ricevuto. Si ottiene semplicemente uno shift di 2s verso il basso (perché mentre il secondo sconta i 2ms di elaborazione, il primo sarà in viaggio) quindi appena termina l'invio del primo pacchetto, partirà anche il secondo.

ESERCIZIO 3

Commutazione di pacchetto - 3

Si consideri una topologia di rete lineare composta da due canali di capacità pari a 500 kbit/s ciascuno.



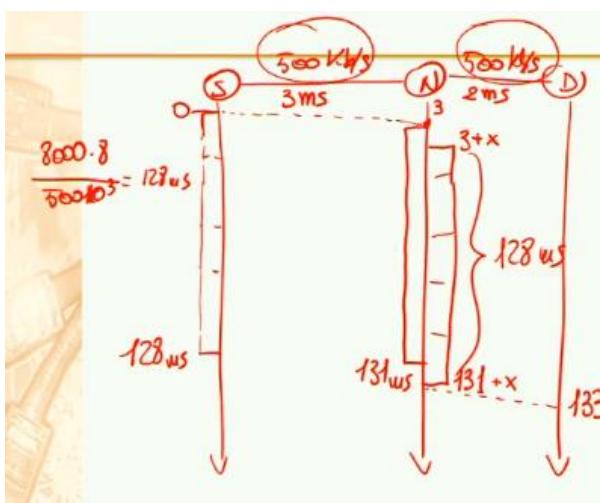
Il nodo S deve trasmettere un file di 8000 byte verso il nodo D attraverso un nodo intermedio store-and-forward. Si supponga che:

- la rete sia scarica e non vi siano errori di trasmissione
- Il tempo di elaborazione nei nodi di commutazione sia trascurabile
- i tempi di propagazione su ciascun canale siano pari a 3 ms per il primo e a 2 ms per il secondo canale
- D debba ricevere tutto il file entro 140ms

Si determini la dimensione massima dei pacchetti trasmessi sul canale (si trascurino le intestazioni)

Adesso la richiesta è quella di determinare la dimensione massima dei pacchetti, viene quindi dato il tempo di trasmissione totale.

Posso calcolare quanto tempo sul primo canale impiegano tutti i pacchetti ad essere trasmessi.



$$\frac{8000 \cdot 8}{500 \cdot 10^3} = 128 \text{ ms}$$

Il primo bit parte al tempo 0 e verrà ricevuto su N al tempo 3 ms, perciò l'ultimo sarà ricevuto a 131 ms.

Al tempo $3+x$ (dove x è la durata del pacchetto) partirà il primo pacchetto sul secondo canale, quindi la trasmissione finirà al tempo $131+x$.

L'ultimo bit sarà ricevuto al tempo $133+x$.

Ottengo l'equazione: $133 + x \leq 140 \rightarrow x \leq 7 \text{ ms}$

Per ottenere la dimensione del pacchetto sarà:

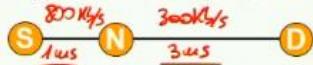
$$L = 7 \cdot 10^{-3} \cdot 500 \cdot 10^3 = 3500 \text{ bit} = 437.5 \text{ B} = 437 \text{ B}$$

$$8000 = 18 \times 437 + 134$$

ESERCIZIO 4

Commutazione di pacchetto - 4

Si consideri una topologia di rete lineare composta da due canali di capacità pari rispettivamente a 800 kb/s e 300 kb/s.



Il nodo S deve trasmettere un file di 8000 byte verso il nodo D attraverso un nodo intermedio N store-and-forward. Si supponga che:

- o la rete sia scarica e non vi siano errori di trasmissione;
- o il nodo intermedio N possa memorizzare in un buffer al massimo 3 pacchetti (indipendentemente dalle dimensioni), incluso quello in trasmissione; un pacchetto che arrivi a buffer pieno viene perso.
- o i tempi di propagazione su ciascun canale siano pari a 1 ms per il primo e a 3 ms per il secondo canale;
- o la dimensione massima dei pacchetti trasmessi sul canale sia di 1500 byte (si trascurino le intestazioni).

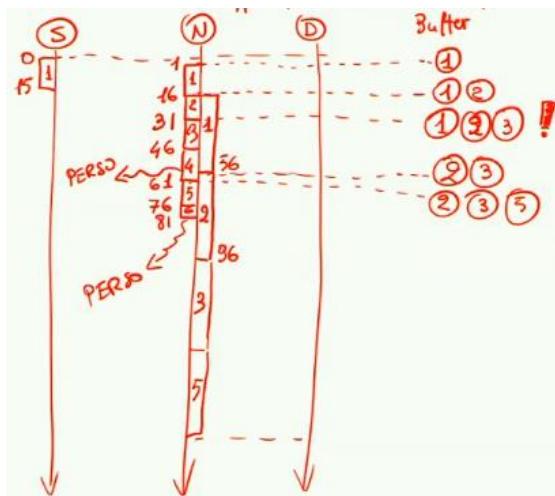
Si determini:

- il numero di pacchetti persi dalla rete
- la dimensione minima del buffer che permetterebbe di non perdere pacchetti

La complicazione sta nel secondo punto, dove il nodo ha un buffer di 3 pacchetti.

1) Stessi calcoli

$$T_{TX,1}^1 = \frac{1500 \cdot 8}{800} = 15 \text{ ms} \quad T_{TX,2}^1 = 5 \text{ ms} \quad T_{TX,1}^2 = \frac{1500 \cdot 8}{300} = 40 \text{ ms} \quad T_{TX,2}^2 = 13,3 \text{ ms}$$



Numero pacchetti persi: 2 (guarda disegno)

Dimensione minima del buffer per evitare perdite: 5 (si va a ragionamento). Se lo metto = a 4 perdo comunque il pacchetto 6.

ESERCIZIO 5

Commutazione di pacchetto - 5

Si consideri una topologia di rete lineare composta da due canali di capacità pari rispettivamente a 300 kb/s e 800 kb/s.



Il nodo S deve trasmettere un file di 8000 byte verso il nodo D attraverso un nodo intermedio N store-and-forward. Si supponga che:

- o la rete sia scarica e non vi siano errori di trasmissione;
- o il nodo intermedio N possa memorizzare in un buffer al massimo 3 pacchetti (indipendentemente dalle dimensioni), incluso quello in trasmissione; un pacchetto che arrivi a buffer pieno viene perso.
- o i tempi di propagazione su ciascun canale siano pari a 1 ms per il primo e a 3 ms per il secondo canale;
- o la dimensione massima dei pacchetti trasmessi sul canale sia di 1500 byte (si trascurino le intestazioni).

Si determini:

- il numero di pacchetti persi dalla rete
- il tempo che trascorre dall'inizio della trasmissione all'istante in cui l'ultimo bit è ricevuto da D.

Identico a prima ma le velocità dei canali sono invertiti.

In questo caso è banale perché non si riempie mai la coda al nodo N.

Tecniche di Segnalazione

Segnalazione (def. ITU-T): *scambio di informazioni che riguardano l'apertura e il controllo di connessioni e la gestione di una rete di telecomunicazione.*

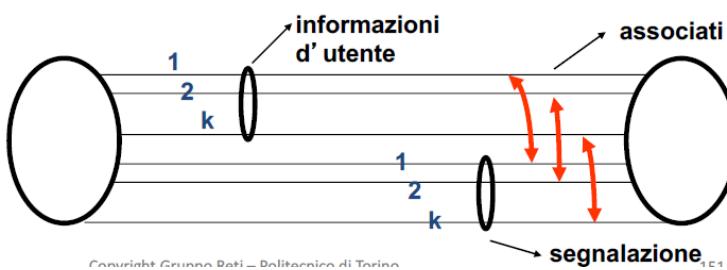
Segnalazione di utente: scambio di informazioni tra utente (terminale) e nodo

Segnalazione internodale (di rete): scambio di informazioni tra nodi

Le segnalazioni possono essere:

- **associate al canale**
 - **in banda:** dati e segnalazioni vengono inviati sulla stessa frequenza, per uguali intervallo di tempo ma in istanti di tempo differenti; in sostanza si alternano dati e segnalazioni
 - **fuori banda:** si utilizzano frequenze e intervalli di tempo diversi, pur essendo sullo stesso canale e dunque inviati allo stesso tempo (si inviano contemporaneamente dati e segnalazioni su frequenze distinte)
- **a canale comune**

Associata al canale (usata nelle reti di telefonia mobile)



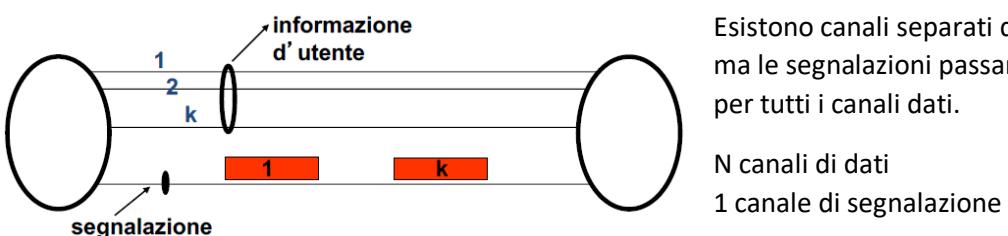
A livello logico si ha una separazione di canali (logici). Alcuni canali trasportano i dati, altri le segnalazioni. Si ricorda che il canale è sempre lo stesso ma viene "diviso" grazie alla modulazione in frequenza (un po' come la radio FM).

Vi è una corrispondenza tra:

- **canale controllante:** che porta informazioni di segnalazione
- **canale controllato:** che porta i dati

La corrispondenza dipende dal fatto che, supponendo la rete cellulare, nel caso in cui si passa da una cella (antenna) all'altra, i dati non devono andare più verso quella antenna ma verso un'altra.

A canale comune (reti con tecnologie avanzate)



Qualità di Servizio nelle reti di telecomunicazione

Offrire un servizio di rete non è solo la “quantità di gb” da scaricare (esempio), ma vuol dire offrire anche una certa qualità di tale servizio. L’operatore è tenuto a dare una certa garanzia del servizio (permetterci di accedere alla rete quando si vuole, avere una banda minima garantita).

Per il progetto di una rete devono quindi essere definite le caratteristiche con cui le informazioni vengono emesse dalle sorgenti nell’ambito di un servizio di telecomunicazione.

Quando si progetta una rete è necessario determinare le **risorse necessarie** al funzionamento di tale rete (velocità dei canali, potenza di elaborazione dei nodi di commutazione, numero di canali/nodi). Questi parametri vengono determinati in base a quali sono le **richieste di servizio** e la **qualità** del servizio da offrire. Viceversa è possibile determinare la **qualità del servizio** conoscendo già le richieste di servizio e le **risorse** disponibili.

Per fare ciò esistono modelli matematici ben definiti.

La qualità del servizio la si eroga a seconda della sorgente che si sta analizzando: possono essere presenti sorgenti numeriche a **velocità costante (Constant Bit Rate – CBR)** (come la voce, videoconferenza) e sorgenti a **velocità variabile (Variable Bit Rate – VBR)** (video in streaming, file transfer)

Le sorgenti CBR sono caratterizzabili da:

- Velocità (b/s)
- Durata (s)
- Processo di generazione delle chiamate

Le sorgenti VBR invece sono caratterizzate da:

- Velocità di picco (b/s)
- Velocità media (b/s)
 - Oppure:
- Grado di intermittenza = $\frac{\text{velocità di picco}}{\text{velocità media}}$ (più è piccolo, più si hanno picchi per poi restare a basse velocità)
- Durata
- Processi di generazione delle chiamate

Indici di qualità

Non tutte le sorgenti di informazione richiedono la stessa qualità. Gli indici che vengono maggiormente usati sono:

- Ritardo (valor medio, percentile, tempo reale)
- Velocità
- Probabilità di errore -> Qualità del canale
- Probabilità di perdita
- Probabilità di blocco

2. Architetture e protocolli

Definizione CCITT:

- **Comunicazione:** trasferimento di informazioni secondo **convenzioni prestabilite**.
- **Protocollo:** descrizione formale delle procedure adottate per assicurare la comunicazione tra due o più oggetti dello stesso livello gerarchico.

La comunicazione richiede **cooperazione**, cioè bisogna definire delle regole che stabiliscono l'interazione tra elementi di una rete e vengono chiamati **protocolli di comunicazione**.

La gerarchia tra i protocolli definisce una **architettura di rete**.

Un protocollo prevede lo scambio di messaggi definendone:

- **Tipologia:** richieste o risposte
- **Sintassi:** struttura dei messaggi
- **Semantica:** significato dei campi di bit dentro ai messaggi
- **Temporizzazione:** sequenze temporali di comandi e risposte

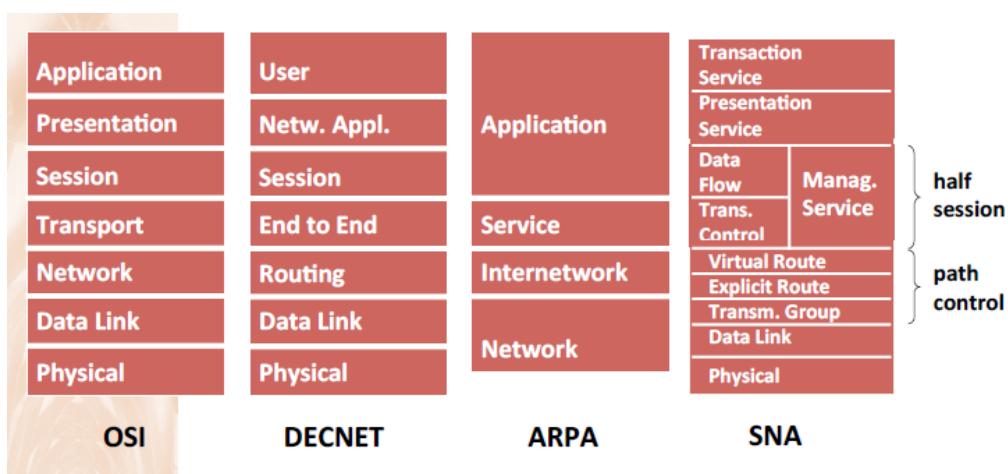
Un'architettura di rete definisce: il **processo di comunicazione**, le **relazioni** tra entità coinvolte nella comunicazione, le **funzioni necessarie** per la comunicazione e le **modalità organizzative** delle funzioni.

Per fare ciò si utilizzano delle **architetture stratificate** per la loro **semplicità** di progetto, **facilità di gestione**, **semplicità** di standardizzazione e **separazione** di funzioni.

Modello di riferimento OSI (Open System Interconnection)

Questo modello è definito da **ISO** ed i **principi fondamentali** che vengono definiti dal modello OSI sono oggi universalmente accettati. Nonostante ciò, non vuol dire che **tutte** le architetture di protocolli siano conformi al modello OSI.

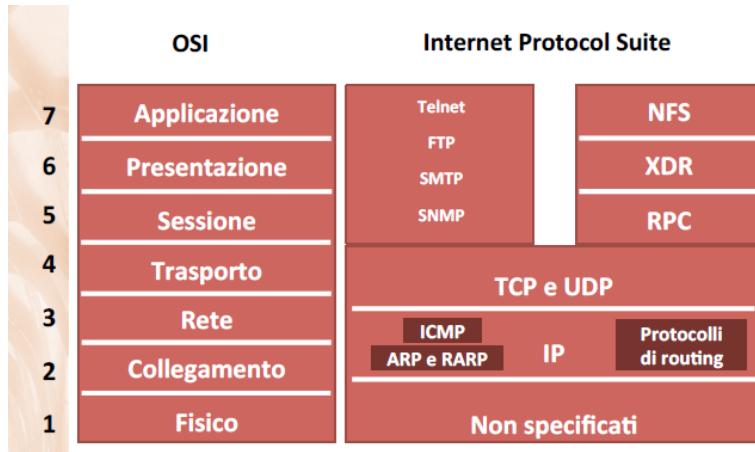
Alcune architetture hanno più/meno livelli.



Ogni strato:

- Fornisce servizi allo strato superiore
- Usa i servizi dello strato inferiore e le proprie funzioni per migliorare il servizio offerto dallo strato inferiore
- Gli utenti dello strato N, le (N+1)-entità, cooperano e comunicano usando l'N-servizio offerto dall'N-fornitore di servizio

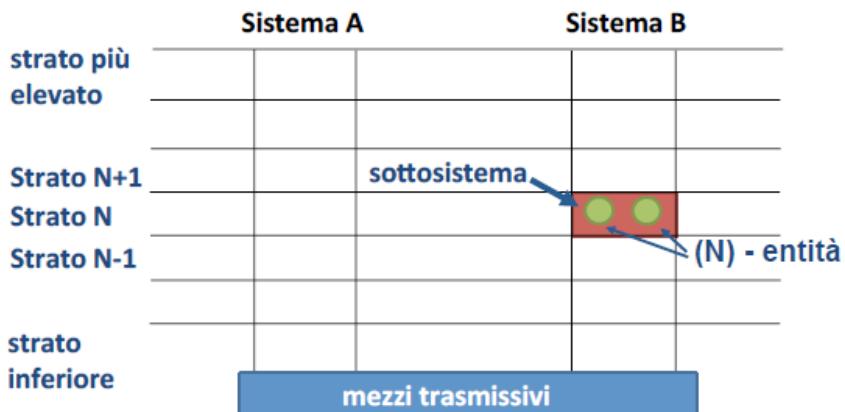
OSI ed Internet (IPS)



Architettura di rete

Una rete è composta di **sistemi** (nodi, terminali) collegati tra loro tramite mezzi trasmissivi. Un sistema possiede diversi **processi applicativi** per comunicare tra loro. Ogni sistema è composto da **sottoinsiemi**.

Ogni sottosistema realizza le funzioni proprie di uno **strato** tramite delle **entità**. Passano dallo strato più elevato (che manipola informazioni complesse) fino allo strato **elementare**.



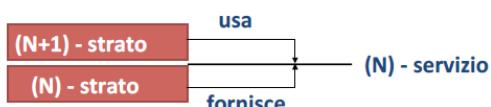
Il livello inferiore è quello col “numero” pari a 1. I successivi vengono incrementati di uno.

Le entità sono gli **elementi attivi di un sottosistema** (processi) e svolgono le funzioni che appartengono ad uno stesso strato. Ogni strato (o livello) fornisce dei servizi allo **strato superiore** ed utilizza quelli dello strato inferiore per migliorare le proprie funzioni.

In sostanza **migliora il servizio** offerto dallo strato inferiore.

Si può identificare:

- Fornitori di servizio (strati bassi)
- Utenti del servizio (strati alti)
- Punti di accesso al servizio: SAP (Service Access Point) (tra due stati intermedi) non sono altro che interfacce attraverso cui due strati si possono parlare.



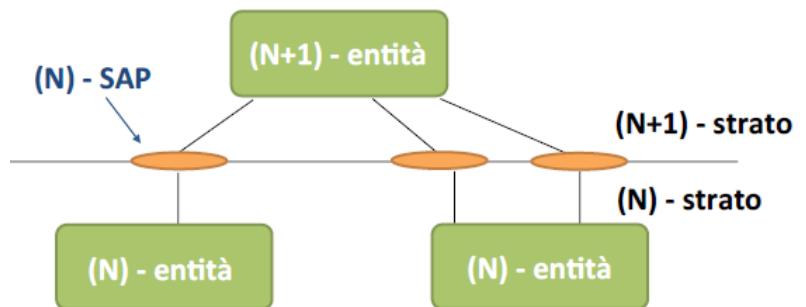
Un servizio offerto dagli strati può essere:

- **Connection-oriented (CO):** accordo preliminare (connessione) tra rete e interlocutori, poi si trasferiscono i dati e infine si rilascia la connessione (ricorda il circuito virtuale)
- **Connectionless (CL):** i dati vengono immessi in rete senza un accordo preliminare e sono trattati in modo indipendente (datagram)

Un concetto importante della comunicazione è quello della **black box**, nozione per cui uno strato sa quali servizi gli vengono offerti ma non sa in quale modo gli strati inferiori operano per fornire questi servizi. Quindi uno strato non sa quanti strati ci sono prima di lui e non sa come operano. Sa soltanto che sono presenti.

Service Access Point

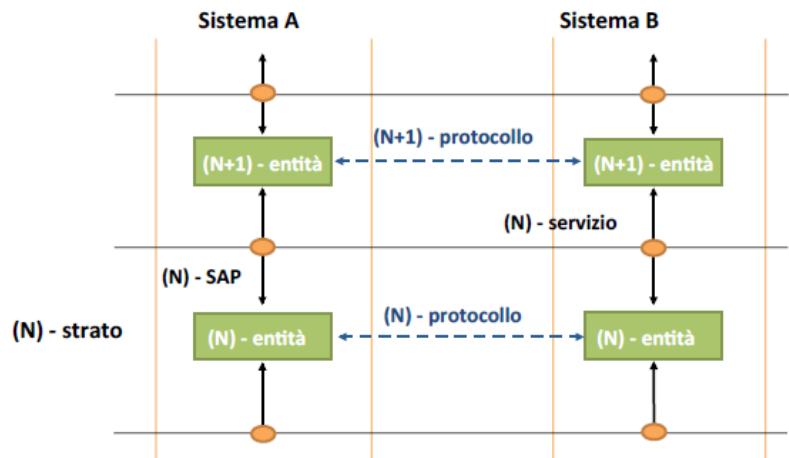
Un N-servizio è offerto ad una entità di livello superiore ($N+1$ -entità) tramite una **connessione** che passa per un punto di accesso al servizio (SAP).



Protocolli

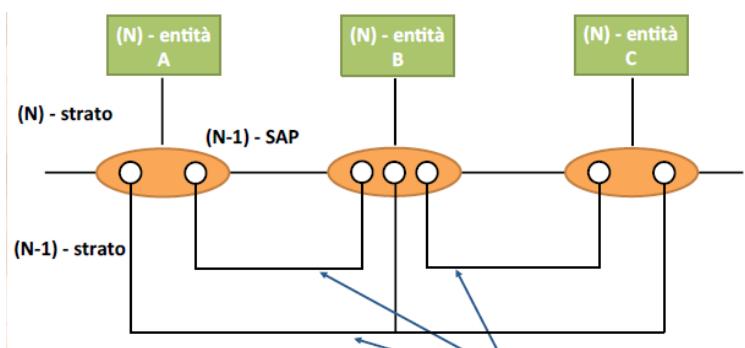
Lo scambio di informazioni tra entità omologhe di sistemi diversi avviene tramite un **protocollo**.

Entità di strato diverso **non** comunicano con protocolli.



Connessioni

Una connessione è una relazione esistente tra SAP diversi (sullo stesso strato) per lo scambio di dati tra interfacce.



Creazione PDU (pacchetto)

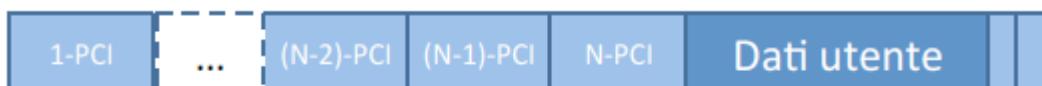
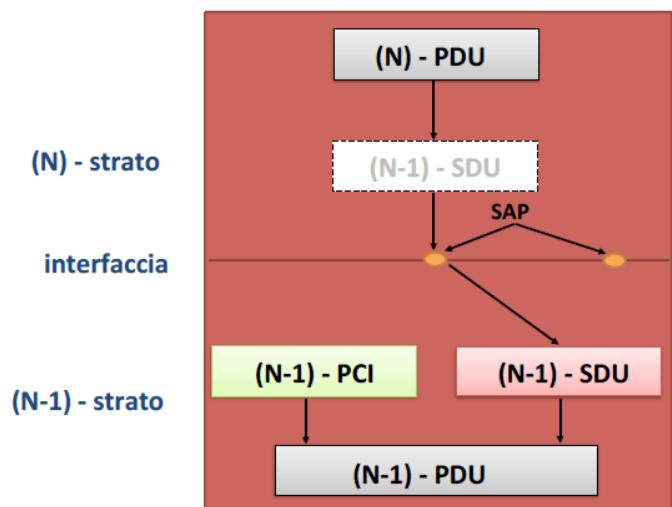
In un sistema a strati, i dati utente presenti allo strato N sono detti N-SDU (Service Data Unit). Lo strato N nel processo di trasmissione aggiunge delle proprie intestazioni (informazioni di controllo) dette N-PCI (Protocol Control Information) (Protocol Control Information)

L'unione di N-PCI e N-SDU forma una **N-PDU**.

Ogni strato inferiore tratta la PDU dello strato superiore come se fosse una "busta chiusa" a cui aggiungere solo un'intestazione.

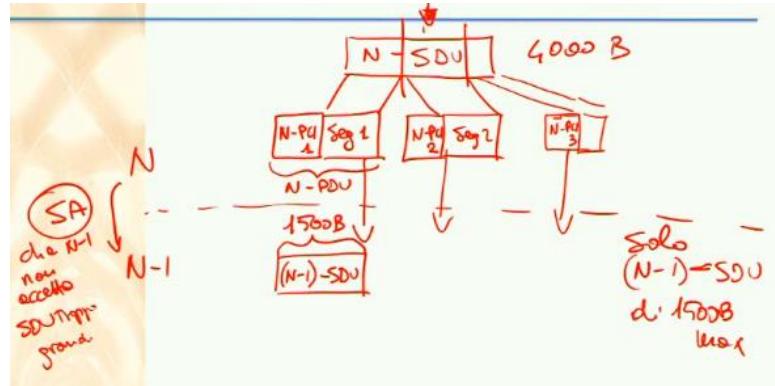
Il processo in foto si ripete strato dopo strato.

Prima della trasmissione, ai dati sono aggiunte tante intestazioni quanti sono gli strati traversati dal sistema. In ricezione avviene il **processo inverso**. Le intestazioni possono essere in **testa** e in **coda**.



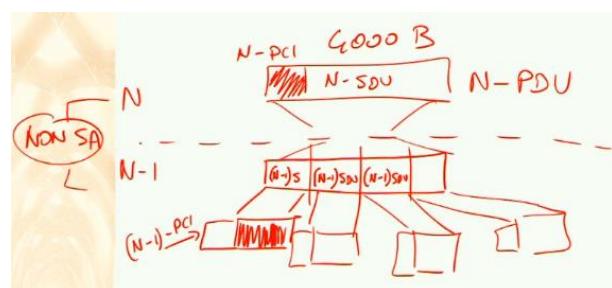
Sulle unità dati esiste la possibilità di effettuare **segmentazione (in TX)** e **concatenazione (in RX)**. Questo perché i vari strati possono avere dei limiti sulla PDU utilizzabile.

Quindi per fare ciò lo strato N deve sapere che N-1 accetta dati di una certa dimensione e dunque farà segmentazione.



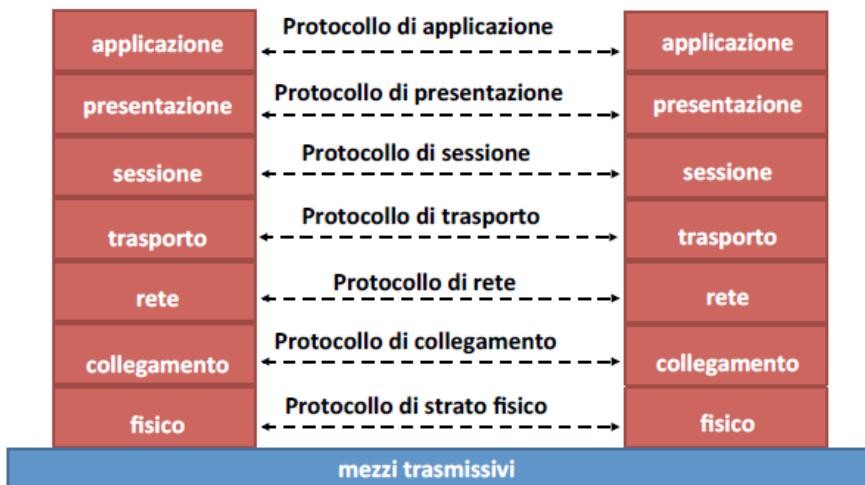
Ovviamente tale operazione aggiunge informazione di controllo (che non sono dati) inutile.

Nel caso in cui il livello N non sapesse la massima dimensione accettata dal livello N-1 si procede come segue

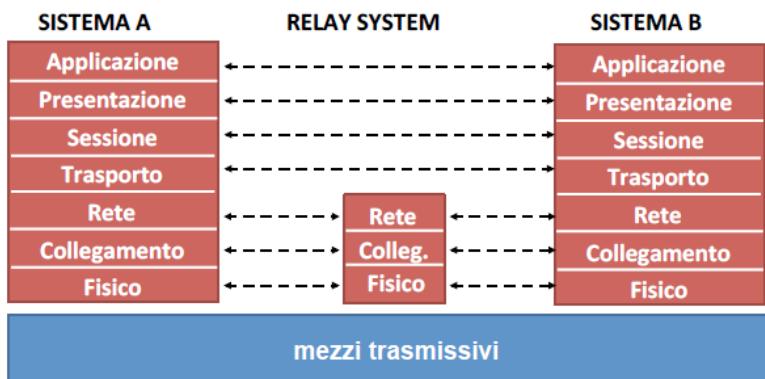


Segmentare non conviene mai, poiché si ha un maggiore overhead delle intestazioni.

Sette strati OSI



I nodi di commutazione (durante il trasferimento) non hanno bisogno di tutti i livelli, perciò è pratica comune nelle reti avere tra i nodi intermedi dei **relay system**, che sarebbe una architettura di rete parziale perché manca dei livelli superiori, ma quelli inferiori li ha tutti.



Strato 1: fisico

Fornisce i mezzi meccanici, fisici, funzionali e procedure per attivare, mantenere e disattivare le connessioni fisiche. Ha il compito di effettuare il **trasferimento** delle cifre binarie scambiate dalle entità di strato di collegamento. Le unità dati sono **bit o simboli**.

Definisce codifiche di linea, connettori e livelli di tensione (impulsi di linea).

Assicura il trasferimento dei bit da un nodo all'altro.

Strato 2: collegamento (Data-Link)

Ha lo scopo di delimitare le unità dati. Fornisce i mezzi funzionali e procedurali per il trasferimento delle unità dati tra entità di strato rete e per fronteggiare malfunzionamenti dello strato fisico.

Questo livello avrà procedure protocollo per dire dove finisce un pacchetto e inizia un altro, oltre ad avere tecniche per rilevare e recuperare errori di trasmissioni. Fa anche controllo di flusso.

Strato 3: rete (Network)

Scopo di **intradare** (ricercare percorsi sulla topologia della rete). Fornisce i mezzi per instaurare, mantenere e abbattere le connessioni di rete tra entità di strato trasporto.

Oltre ad effettuare instradamento, esegue controllo di flusso e congestione (a seconda del protocollo usato).

Strato 4: trasporto

Ha il compito di colmare le carenze di qualità di servizio delle connessioni di strato rete con funzioni di: **controllo di errore, controllo di sequenza, controllo di flusso**. Esegue anche la **multiplazione e demultiplazione** di connessioni e la **segmentazione** dei dati in pacchetti.

Strato 5: sessione

Viene spesso integrato nelle funzioni dei livelli superiori. È il livello che si occupa di sincronizzare lo scambio di dati in modo da poterlo **sospendere, riprendere e terminare ordinatamente**.

Strato 6: presentazione

Anche questo livello viene spesso integrato nelle funzioni del livello superiore. Esso risolve problemi di compatibilità cambiando il formato di rappresentazione dei dati oppure in caso **cifrare** le informazioni tramite delle chiavi condivise con l'altro strato applicazione.

Strato 7: applicazione

Questo è il livello in cui sono presenti le applicazioni. Contiene tutti i mezzi per accedere all'ambiente OSI.

3. Quiz nuovo esame

QUIZ 1

La segnalazione a canale comune

- Richiede l'uso di messaggi di segnalazione che contengano l'identificativo della connessione a cui si riferiscono.
- E' usata solamente nelle reti a commutazione di pacchetto, perché nelle reti a commutazione di circuito viene usata la tecnica di segnalazione a canale associato.
- Non può essere usata nelle reti a commutazione di cella o pacchetto perché è troppo difficile identificare le connessioni in queste reti senza associare loro un canale di segnalazione.
- Fa uso di comuni canali di trasmissione che vengono associati a ciascuna connessione in fase di apertura della connessione stessa.

- **Vera:** Definizione di segnalazione a canale comune: avere un canale separato gestito a commutazione di pacchetto e in ogni pacchetto c'è l'informazione di segnalazione che riguarda uno specifico canale controllato e dentro al pacchetto è necessario inserire l'identificativo della connessione.
- Falsa: perché a canale comune è tipicamente usata nelle reti telefoniche (commutazione di circuito).
- Falsa: non vuol dire nulla
- Falsa: definizione di canale associato

QUIZ 2

In una topologia ad albero con N nodi, il numero di canali bidirezionali è uguale a:

- N
- $N*(N-1)/2$
- N-1
- N/2

QUIZ 3

La multiplazione a divisione di codice

- è un caso particolare di sola divisione di tempo
- è un caso particolare di sola divisione di frequenza
- permette di usare multiplazione e commutazione sullo stesso mezzo fisico
- non richiede necessariamente di essere accoppiata a multiplazione di tempo o di frequenza

La terza è errata perché la commutazione è la condivisione del nodo (non del mezzo fisico) mentre la multiplazione è il modo in cui si divide il mezzo fisico.

Quiz 4

- Nel modello OSI, il livello rete fornisce servizi
 - 1. basandosi sui servizi forniti dal livello collegamento
 - 2. basandosi sui servizi forniti dal livello applicazione ~~X~~
 - 3. basandosi sui servizi forniti dal livello trasporto



Occorre ricordare dove si posiziona il livello rete. Sta sopra al livello collegamento e sotto al livello trasporto.

La risposta è la 1.

Quiz 5

- Un (N)-protocollo regola le interazioni
 - 1. tra una (N)-entità ed un (N-1) SAP
 - 2. **tra (N)-entità dello stesso sistema o di sistemi diversi**
 - 3. tra una (N)-entità e tutte le entità al di sotto di essa nella pila protocollare
 - 4. tra una (N)-entità e la (N-1)-entità

Regola interazioni soltanto tra entità dello stesso strato.

Quiz 6

- Un protocollo a finestra di tipo go-back-N prevede che:
 - 1. nel momento in cui un pacchetto è ricevuto con un errore siano ritrasmesse le ultime N unità dati
 - 2. il trasmettitore riceva i riscontri per le ultime N unità dati trasmesse prima di poter iniziare la trasmissione di altre unità dati
 - 3. **il trasmettitore non possa inviare più di N unità dati prima di aver ricevuto uno o più riscontri**
 - 4. ogni volta che scade un time-out sia ritrasmessa l'unità dati il cui ordine di sequenza è pari al numero di sequenza dell'ultima unità dati trasmessa diminuito di N

1. La ritrasmessione avviene con timeout e non con pacchetti ricevuti con errori.
2. Basta ricevere un ACK che faccia avanzare la finestra.
3. Giusta
4. Il timeout può scadere durante la trasmissione della finestra e la regola non è di ritrasmettere andando indietro, ma ritrasmettendo il contenuto della finestra.

Funzioni svolte:

- Recupero errori di trasmissione
- Controllo del flusso di trasmissione
- Controllo sequenza

4. Protocolli a finestra

I protocolli a finestra compaiono in quasi tutte le reti telematiche moderne e passate. Essi servono principalmente per recuperare gli errori di trasmissione (avere pacchetti inviati che sono soggetti a errori di trasmissione o congestione). Effettuano anche controllo del flusso di trasmissione (passo con cui le trasmissioni sono inviate) e controllano la sequenza (si assicura di consegnare i pacchetti in ordine).

Protezione dagli errori di trasmissione

La trasmissione di bit su un canale non è mai esente da errori. Un errore è una errata interpretazione del segnale al ricevitore (dopo un segnale trasmesso). A seconda del tipo di canale la probabilità di errore sul bit varia da 10^{-12} (fibra ottica) a 10^{-3} (canale radio rumoroso). Servono quindi tecniche per **rilevare** o **recuperare** (correggere) errori. Le tecniche utilizzate sono le **codifiche di canale**.

Codifica a blocco per controllo di errore



“utente” da intendere in senso OSI (l’entità di strato superiore che usa lo strato inferiore). I bit utente sono i bit che lo strato n ha passato allo strato n-1.

L’aggiunta dei bit di parità avviene, di norma, dentro l’intestazione. Il numero di bit di parità dipende da ciò che si vuole fare.

Bit di parità

- Riconosce errori in numero dispari (perché se pari, la parità rimane)
- Non corregge
- Un errore più probabile di K errori

0	1	1	0	1	0	1	0
0	1	0	0	1	0	1	1

Cerco di ottenere sempre un numero pari di “uni” (vedi immagine).

La parte bianca è la n-PDU, mentre l’unione di parte bianca e verde è la n-1-PDU.

Codice a ripetizione

- Decisione a maggioranza
- Permette di correggere gli errori
- Maggiore ridondanza

0	1	1	0	1	0	1	0
0	1	1	0	1	0	1	0
0	1	1	0	1	0	1	0

Si ripete semplicemente ogni byte per 3 volte. Il ripetitore decide a maggioranza quale dei tre bit è stato trasmesso.

Parità di riga e di colonna

- Consente la correzione di errori singoli
- Minore ridondanza della parità di riga (o colonna)
- Maggiore ridondanza del codice a ripetizione

0	1	1	0	1	0	1	0
0	1	0	0	1	0	1	0
0	0	0	1	0	1	0	1
1	1	0	0	0	1	0	1
1	1	1	0	1	0	1	0
0	0	0	0	1	0	1	0
0	1	1	1	0	1	0	1
0	1	0	0	0	1	0	0
0	0	1	0	0	1	1	1

Siccome noto che c’è un errore sulla colonna 5 e un errore sulla riga 2 posso identificare esattamente qual è l’errore e posso correggerlo.

NON esiste nessuna tecnica di codifica di canale che darà la **certezza** di non avere errore di trasmissione

In generale, si introducono **bit di parità** tra le informazioni di controllo all'interno delle PDU come il CRC (Cyclic Redundancy Check), cioè dei controlli sul pacchetto (si calcola un valore con un algoritmo) e una volta ricevuto viene rieseguito lo stesso algoritmo e confrontato il risultato. I bit di parità fanno parte dell'intestazione.

A seconda della quantità di bit di parità si possono utilizzare tecniche diverse:

- FEC (Forward Error Correction): i bit di parità (molti) vengono utilizzati per tentare di **correggere gli errori in ricezione**, senza ritrasmettere il pacchetto
- ARQ (Automatic Retransmission reQuest): i bit di parità (pochi) vengono usati per cercare di rivelare gli errori e permettere al ricevitore di chiedere la ritrasmissione della PDU.

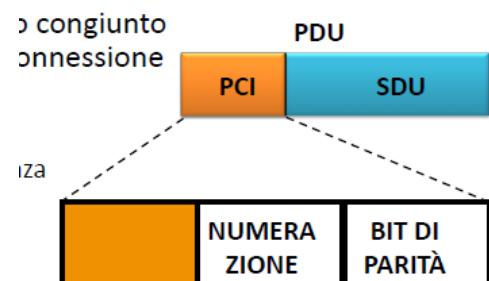
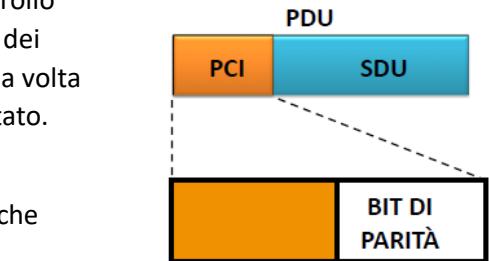
In questo corso si analizzerà ARQ.

ARQ

Si realizza attraverso un protocollo finestra.

Si effettua un controllo congiunto di **errore, sequenza e flusso**.

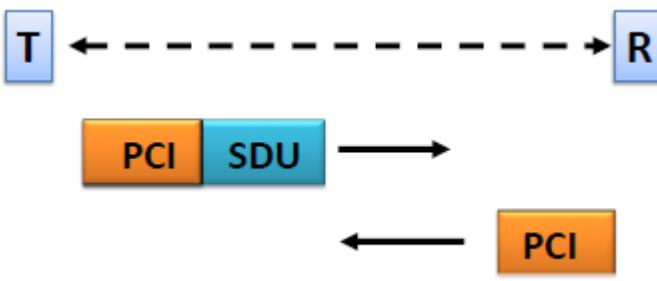
I **bit di parità** sono necessari per effettuare il controllo di errore, ma se voglio effettuare anche il controllo di sequenza devo inserire una **numerazione** (numero di sequenza).



Esistono **tre tecniche ARQ**:

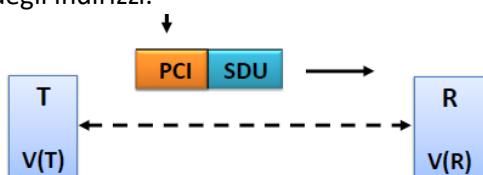
- Stop and wait (Alternating bit)
- Go back N
- Selective repeat

Le tre tecniche sono scritte in ordine di complessità crescente.

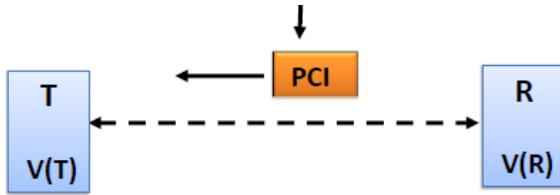


Si ipotizza un ambiente di comunicazione unidirezionale con un trasmittitore e ricevitore. È unidirezionale perché ipotizziamo che i dati utente vengano inviati solo da T a R. Il ricevitore restituirà soltanto delle informazioni di controllo (bit di ridondanza e numerazione).

Nella PCI del pacchetto dati si troverà il **bit di ridondanza/codifica canale**, il **numero di sequenza N(T)** e degli indirizzi.



Il trasmittitore ha un suo contatore $V(T)$ che corrisponde al numero di sequenza $N(T)$.



La PCI del pacchetto di riscontro viene definito ACK (Acknowledgment). Nel pacchetto di ACK si trovano i bit di ridondanza ed un numero di sequenza **N(R)** che è il **numero di sequenza atteso**. Se il trasmettitore invia il pacchetto 5, il ricevitore risponde con 6. Questo vuol dire "Ho ricevuto il pacchetto 5, mi aspetto il 6".

Stop and Wait ($W_T = 1, W_R = 1$)

Il trasmettitore fa una **copia** della PDU da inviare (per una possibile ritrasmissione). Successivamente invia la PDU e fa partire un **orologio** (timer) detto tempo di **timeout**. In questo lasso di tempo attende una risposta alla PDU che viene inviata (cioè attende l'ACK). Se il timeout scade prima dell'arrivo della conferma, ripete la trasmissione facendo ripartire il timeout.

Quando il trasmettitore riceve un ACK, quest'ultimo controlla la **correttezza** dell'ACK. Se l'ACK è errato, **viene buttato via**. Se l'ACK è corretto, si passa al controllo del **numero di sequenza**. Se l'ACK è corretto e il numero di sequenza rileva che l'ACK è effettivamente riferito all'ultima PDU inviata, allora il trasmettitore butta la copia e prosegue con la prossima PDU.

Il ricevitore quando riceve la PDU, controlla se quest'ultima è corretta (altrimenti la scarta). Successivamente controlla il numero di sequenza e, se la PDU è corretta viene inviata una **conferma di ricezione** inviando il numero della PDU **attesa** e successivamente la PDU viene consegnata ai livelli superiori.

ESEMPIO

La connessione viene configurata alla sua apertura (cioè TX e RX concordano i parametri del protocollo).

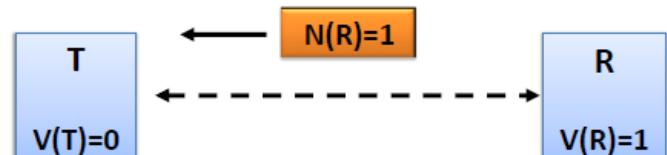
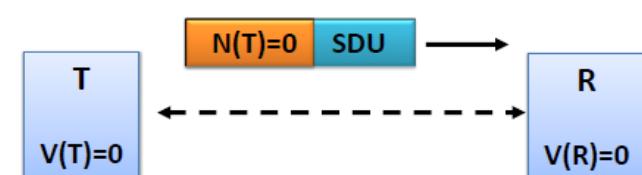
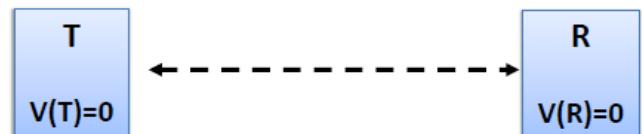
Si avrà $V(T)=0$ e $V(R)=0$.

Viene inviata una PDU con $N(T) = V(T)$ e parte il timer.

Il ricevitore riceve la PDU, controlla che sia corretto, controlla che sia in sequenza (confrontando $N(T)$ con $V(R)$). Se la PDU non fosse in corretta il pacchetto verrebbe scartato.

Se il numero di sequenza non fosse corretto, verrebbe inviato un ACK con scritto che si sta aspettando qualcos'altro.

Supponendo la PDU corretta, il ricevitore incrementa il contatore e copia tale valore nell'intestazione dell'ACK e lo invia.



Alla ricezione dell'ACK, il trasmettitore controlla che il pacchetto sia corretto e controlla che il numero di sequenza sia pari a $N(R) = V(T)+1$ (ovvero a $0+1$) e in tal caso arresta il timer e incrementa il suo contatore (predisponendosi per la prossima trasmissione).

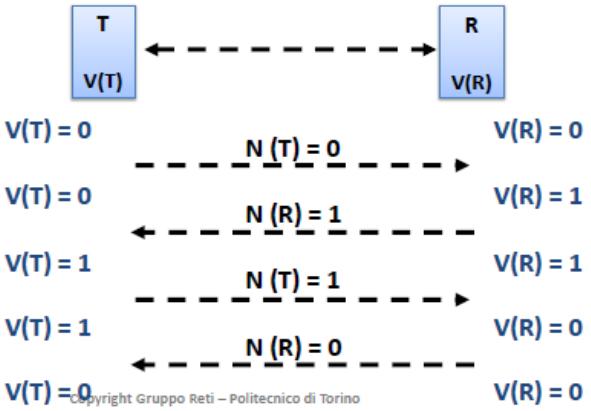
Alla fine del ciclo i contatori dei due nodi sono **uguali**.

Si ha la sicurezza che la PDU sia stata inviata correttamente? **NO, MAI**.

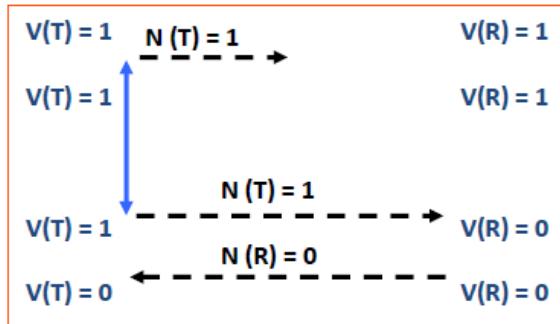
La durata del ciclo (da andata a ritorno) viene denominata **round trip time (RTT)**. La numerazione delle PDU è indispensabile e ciclica (perché il numero di bit intestazione e contatori sono finiti).

Alternating bit protocol

E' il caso in cui si utilizza **un solo bit** per la numerazione. Si incrementa sempre alternando tra 0 ed 1.

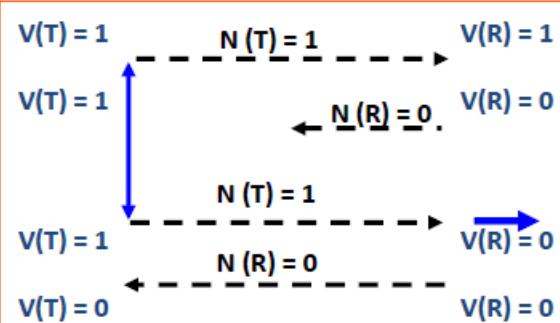


In caso di ricezione di una PDU errata (supponendo che il pacchetto inviato si perda), si aspetta il timeout e



si invia di nuovo il pacchetto. La PDU inviata (prima e dopo la freccia blu) è sempre la stessa. La regolazione del timeout è **delicata**. Questo perché il timeout dipende dal RTT (cioè dal tempo di propagazione). Non si può inserire un timeout troppo breve (perché trasmetto più volte la stessa PDU) e se lo imposto troppo lungo ho evidenti problemi di velocità. Una tecnica è quella di misurare il RTT per perfezionare il timeout.

Se ad esserci una perdita è il pacchetto di Acknowledgment (e non la PDU), il trasmettitore si accorge



semplicemente che il timeout scade (perché non riceve ACK) e allora ritrasmette la PDU. Il ricevitore attende 0, ma siccome riceve 1 scatta la PDU (dove c'è la freccia blu). Il trasmettitore riceve un'ack che ha il significato di ribadire quale PDU si aspetta il ricevitore. (in pratica si reinvia l'ACK).

Nuovamente si ha lo stesso contatore al termine.

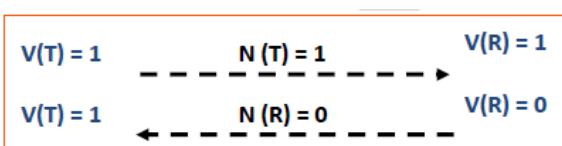
Da qui si evince che la numerazione è **indispensabile**.

Supponiamo di avere adesso un **canale non sequenziale** (topologia con più strade per andare da T ad R). Essendo presenti strade alternative, non è detto che le PDU viaggeranno tutte sullo stesso canale. In questo caso ci si ritrova in un canale non sequenziale ed è possibile che i pacchetti possano essere ricevuti in **ordine sparso**.

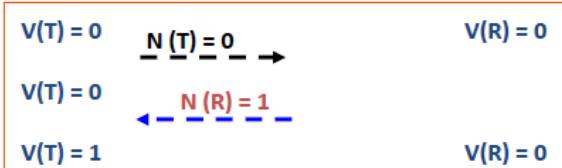
Il trasmettitore inizia a trasmettere la prima PDU, il ricevitore risponde con il primo ACK, ma questo finisce sulla strada più lunga (e quindi non arriva entro lo scadere del timeout).

Allora il trasmettitore, allo scadere del timer, invia nuovamente la stessa PDU (anche se questa è stata ricevuta correttamente) e il ricevitore, avendola già ricevuta la scarta e manda indietro un'ulteriore copia del primo ACK (mentre il primo è ancora in fase di invio).

Alla fine del primo ciclo entrambi i ricevitori hanno lo stesso contatore. Parte il secondo ciclo.



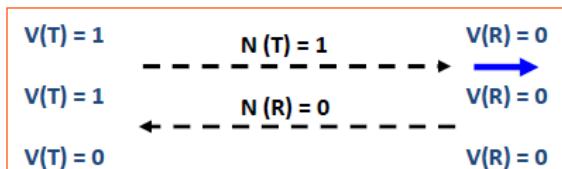
Il secondo ciclo (primo rettangolo) si svolge correttamente. Ovviamente appena arriva la conferma il contatore passa a 0 e si ritorna allo stesso contatore per entrambi.



Nel terzo ciclo, si nota che la 3°PDU con numerazione 0 si perde, e il primo ACK che era stato inviato molto tempo prima viene ricevuto (proprio come se confermasse che la PDU 0 è stata ricevuta, ma in realtà

non è così). Il trasmettitore prende per buona questa conferma e quindi il ciclo termina, ma i contatori non sono allineati.

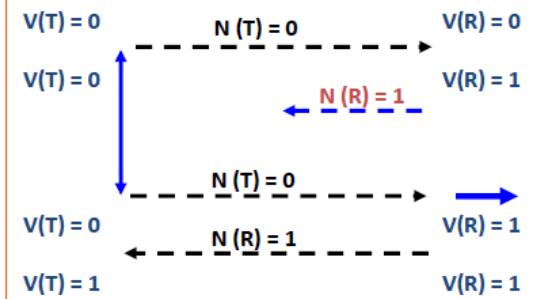
Questo perché l'ACK non dice a quale pacchetto fa riferimento tale Ack.



Al quarto ciclo, con i contatori non allineati, il trasmettitore invia la 4° PDU (considerando la terza per buona) ma al ricevitore avendo il contatore a 0 non si aspetta la PDU numerata con 1 perciò appena la riceve la scarta, anche se era una PDU buona. Invia quindi un ACK in cui continua ad aspettarsi 0. Allora i contatori tornano ad essere allineati, il ciclo continua ma il protocollo non ha funzionato perché: si è persa la 3° PDU e la 4° PDU è stata scartata.

Su un canale non sequenziale è dunque possibile che si verifichino malfunzionamenti: **perdita e/o duplicazione di PDU**.

Il responsabile di tali disguidi è la **numerazione** (poiché due numeri non sono stati sufficienti per identificare i pacchetti).

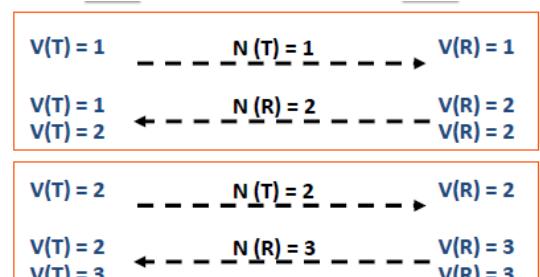


Si effettua nuovamente una trasmissione su canale non sequenziale utilizzando questa volta una **numerazione di modulo 4**.

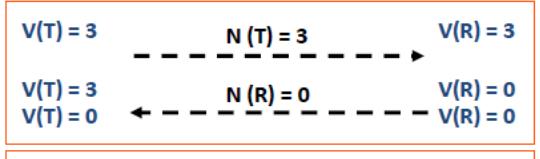
Ci si ritrova nella stessa situazione di prima (ACK ritardato nella consegna).

Il trasmettitore invia nuovamente il pacchetto e il ciclo termina correttamente (contatori allineati).

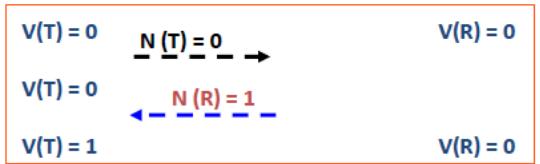
All'inizio del 2° ciclo parte la seconda PDU che arriva correttamente, idem anche per il terzo ciclo dove tutto si svolge correttamente.



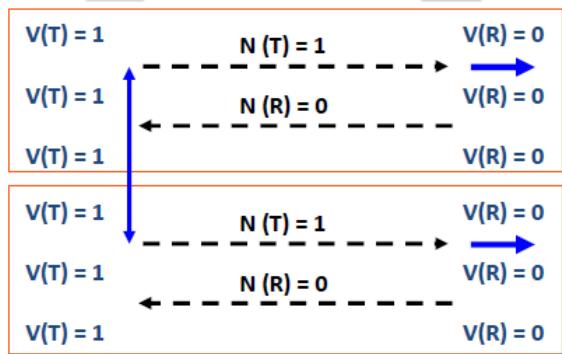
Al quarto ciclo ancora tutto procede.



Al quinto ciclo si perde proprio il pacchetto con stessa numerazione della prima PDU di cui si era perso l'ACK (che ricompare) e funziona perché conferma al trasmettitore che il pacchetto è stato ricevuto.



Nel sesto ciclo si ha che il trasmettitore invia quella che per lui è la 6°PDU, ma il ricevitore si vede arrivare una PDU con numero 1 mentre lui si aspetta la 0, perciò la scarta e invia nuovamente la conferma con lo 0.



Il trasmettitore a questo punto si vede un ACK fuori sequenza (perché il ricevitore si aspetta 0 ma lui l'ha già inviata) perciò scarta l'ACK, non cambia il contatore e finito il timeout invia nuovamente la stessa PDU.

Il ricevitore continua a scartarlo e a inviare un'ACK fuori sequenza (LOOP).

Si continuerà in loop finché non si supera un determinato numero di PDU inviate (valore in cui si riconosce che c'è un malfunzionamento).

A quel punto si blocca il protocollo e servono dei meccanismi per ripartire.

Le possibilità di malfunzionamento si possono semplicemente ridurre aumentando il numero di **bit per la numerazione** ed inserire un **tempo di vita massimo** (TTL Time To Leave) per **PDU e ACK**.

Go Back N ($W_T = n$, $W_R=1$)

Il protocollo Stop and Wait in assenza di errori permette di trasferire con successo un pacchetto per RTT (Round Trip Time) e ciò può essere poco efficiente a causa di elevati ritardi di attesa delle conferme.

Permettere la trasmissione di più di una PDU (nello stesso ciclo) prima di fermarsi in attesa delle conferme migliora le prestazioni e questo è il protocollo **Go Back N**.

La **finestra di trasmissione** W_T rappresenta *la quantità massima di PDU in sequenza che il trasmettitore è autorizzato ad inviare in rete senza averne ricevuto riscontro (ACK)*.

Vuol dire che se ho una finestra $W_T = 10$ posso inviare 10 pacchetti uno dopo l'altro senza curarmi di aver ricevuto conferma.

La dimensione è limitata dalla quantità di memoria allocata in trasmissione. La finestra di trasmissione W_T rappresenta anche il numero massimo di PDU presenti contemporaneamente sul canale o in rete (inviate da uno stesso trasmettitore).

La **finestra di ricezione** W_R rappresenta *la sequenza di PDU che il ricevitore è disposto ad accettare e memorizzare*.

La dimensione della finestra è limitata dalla quantità di memoria allocata in ricezione.

Questa finestra di ricezione serve per memorizzare i pacchetti nel caso in cui durante l'invio in sequenza qualche pacchetto viene perso (non si possono inviare subito al livello superiore ma bisogna aspettare la ritrasmissione del pacchetto perso).

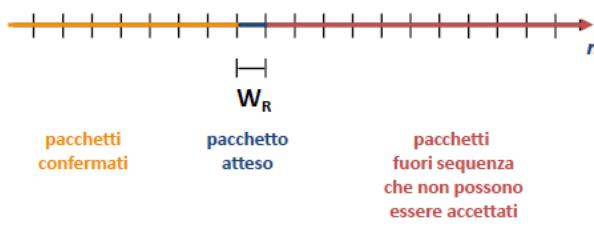
Questi due parametri sono indipendenti (il trasmettitore non è tenuto a conoscere la W_R e il ricevitore non è tenuto a conoscere la W_T).



Il termine corretto per la finestra di trasmissione è la **sliding window** (finestra scorrevole). In un istante di tempo questa finestra di trasmissione si trova posizionata in un certo sottosieme dei numeri di pacchetti inviati (presenti sull'asse). I pacchetti a sinistra della W_T sono pacchetti già inviati (e quindi sono usciti dalla finestra di trasmissione) e all'interno trovo i pacchetti trasmessi ma in attesa di conferma e anche quelli ancora da trasmettere. Alla destra della finestra di trasmissione ci sono i pacchetti che ancora non possono essere trasmessi (perché fuori dalla finestra di trasmissione). La finestra scorre alla ricezione di un ACK.

sono usciti dalla finestra di trasmissione) e all'interno trovo i pacchetti trasmessi ma in attesa di conferma e anche quelli ancora da trasmettere. Alla destra della finestra di trasmissione ci sono i pacchetti che ancora non possono essere trasmessi (perché fuori dalla finestra di trasmissione). La finestra scorre alla ricezione di un ACK.

Nel Go-back-N il ricevitore **ha sempre finestra di ricezione unitaria**.



A sinistra della W_R vi sono i pacchetti già inviati al livello superiore. Nella W_R ho il pacchetto atteso (che devo ancora ricevere e che sto aspettando). I pacchetti successivi sono pacchetti fuori sequenza, ma avendo finestra di ricezione unitaria non li potrei comunque immagazzinare (perché non ho spazio nel buffer della W_R).

Algoritmo Go Back N

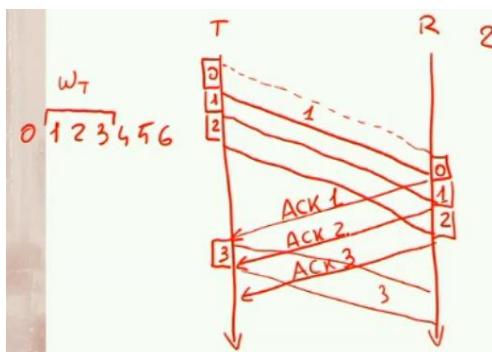
- Il trasmettitore può inviare fino a $N = W_T$ PDU, facendone di ognuno una copia.
- Attiva un solo orologio per le N PDU (che viene resettato ad ogni trasmissione di PDU)
- Si pone in attesa degli ACK
- Per ogni ACK in sequenza ricevuto, fa scorrere in avanti la finestra di tanti pacchetti quanti sono i pacchetti confermati
- Se il timeout scade prima della conferma di ricezione relativa alla PDU che ha settato il timeout, ripete la trasmissione di tutte le PDU non ancora confermate.

Al ricevitore

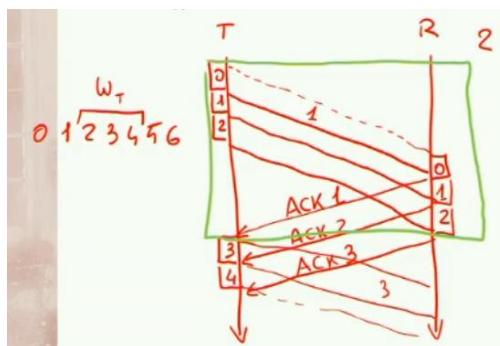
- Controlla la correttezza della PDU
- Controlla il numero di sequenza
- Se la PDU è corretta invia la conferma di ricezione
- Se la PDU contiene il primo numero di sequenza non ancora ricevuto, viene consegnata ai livelli superiori.

ESEMPIO

$W_T=3$ e $W_R=1$



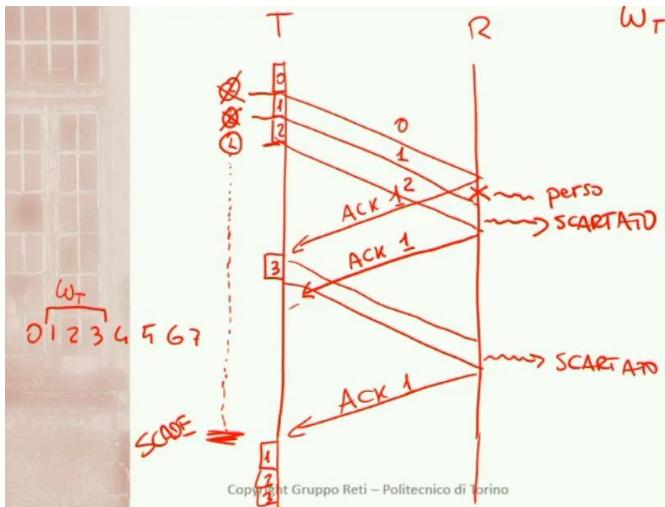
Il trasmettitore invia i primi pacchetti della finestra di trasmissione (0,1,2). Il ricevitore, appena riceve il primo pacchetto invia l'ACK relativo a quel pacchetto. Non appena l'ACK arriva al trasmettitore, la finestra di trasmissione si sposta a destra di una posizione e quindi può inviare il nuovo pacchetto rientrato nella finestra di trasmissione (il 3).



La stessa cosa accade alla ricezione del di ACK 2.

Quello in verde è il tempo del primo ciclo. Il primo ciclo si conclude quando viene abilitato l'avanzamento della trasmissione.

ESEMPIO $W_T=3$ $W_R=1$



L'orologio inizia a scorrere con l'invio del pacchetto 2 e continuerà a scorrere fino alla ricezione di ACK 3.

ACK 1 abilita il pacchetto successivo (ma il timer continua a scorrere).

Si suppone che il pacchetto 1 venga perso. Il ricevitore dunque non manda nulla.

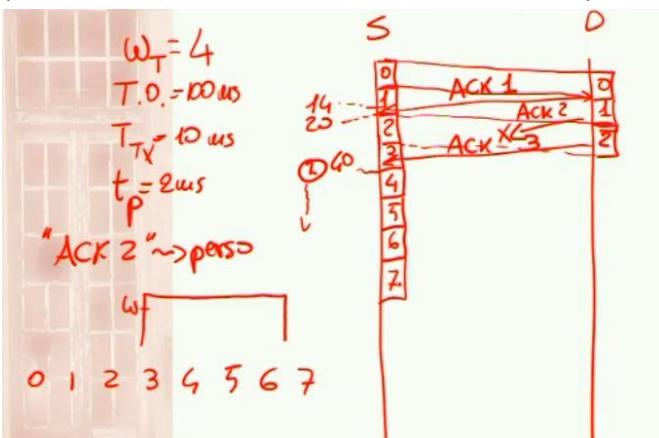
Successivamente arriva il pacchetto 2 che viene ricevuto correttamente e invia **ACK 1** perché il pacchetto 2 viene scartato (perché non c'è spazio nel buffer di ricezione).

Al trasmettitore, ACK 1 non fa avanzare la finestra e dunque resta in attesa e il timer continua a scorrere.

Il pacchetto 3 viene inviato e ricevuto ma scartato (sempre per via del buffer) e viene inviato nuovamente ACK 1.

Il timer al trasmettitore scade, perciò riparte dall'ultimo pacchetto non confermato e invierà nuovamente i pacchetti 1,2 e 3.

Cosa succede se arriva prima ACK 2 di ACK 1 ? Il trasmettitore fa diventare inutile ACK 1 perché per lui darà per scontato che avendo ricevuto ACK 2 anche il primo pacchetto sia stato consegnato. L'esempio in foto non vede alcun problema con la perdita del pacchetto ACK 2, perché ACK 3 confermerà di aver ricevuto tutto quanto e attende il pacchetto 3.



non vede alcun problema con la perdita del pacchetto ACK 2, perché ACK 3 confermerà di aver ricevuto tutto quanto e attende il pacchetto 3.

T.O = timeout

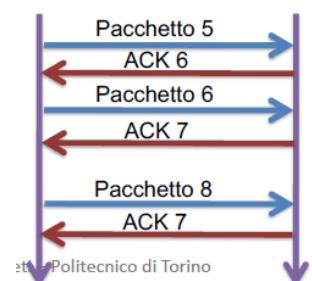
Si nota che quando il tempo di andata e ritorno (RTT) è tale per cui il primo ACK torna prima che sia completata la finestra di trasmissione, l'effetto è che il trasmettitore non si ferma mai (salvo errori).

Semantica degli ACK

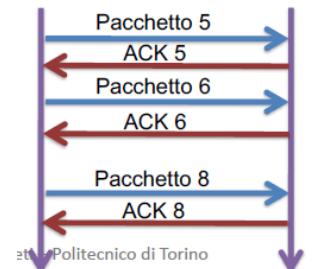
La semantica associata al pacchetto di riscontro può essere diversa da protocollo a protocollo. La semantica è quella descritta fino ad ora.

Trasmettitore e ricevitore si devono accordare preventivamente.

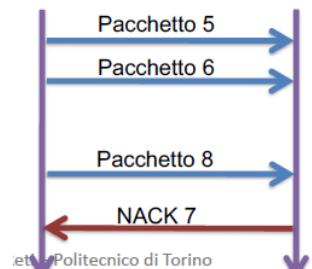
- **ACK cumulativo:** si notifica la corretta ricezione di tutti i pacchetti con numero di sequenza inferiore a quello specificato nell'ACK (UTILIZZATO IN TUTTI GLI ESERCIZI)
- **ACK(n)** significa "ho ricevuto tutto in sequenza fino ad n escluso"



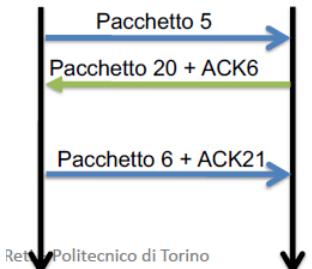
- **ACK selettivo (o individuale):** si notifica la corretta ricezione di un pacchetto particolare.
ACK(n) significa "ho ricevuto il pacchetto n, ma non ti do indicazioni sui pacchetti successivi o precedenti".



- **ACK negativo (NAK):** si notifica la richiesta di ritrasmissione di un singolo pacchetto. In questo caso è il ricevitore che ha la responsabilità di identificare pacchetti persi.
NAK(n) significa "ritrasmetti il pacchetto n" ma non da indicazioni su quali pacchetti sono stati ricevuti.



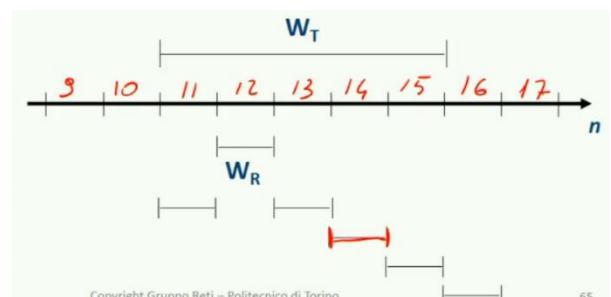
- **Piggybacking:** nel caso di flussi di informazione bidirezionali, è sovente possibile scrivere l'informazione di riscontro (ACK) nella intestazione di PDU di informazione che viaggiano nella direzione opposta.
Permette di risparmiare ACK.



Posizioni relative corrette tra W_T e W_R

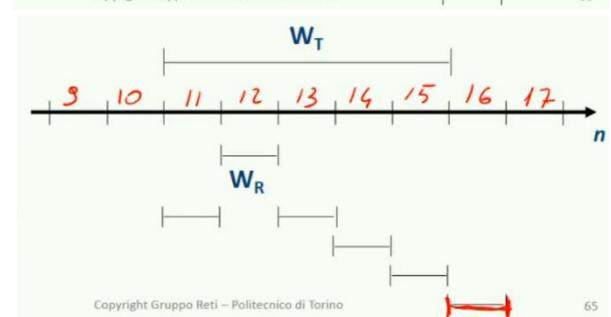
W_T e W_R possono solo trovarsi nelle seguenti posizioni reciproche. Altre posizioni danno luogo a malfunzionamenti.

La finestra del ricevitore può trovarsi, ad esempio, sulla posizione 11. Ovvero sono nella situazione in cui il ricevitore ha la possibilità di inviare il pacchetto 11.



E' possibile anche che ci si trovi nella posizione identificata in rosso.

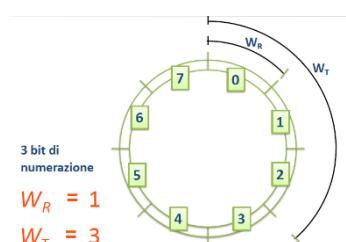
In quel caso vuol dire che il ricevitore ha ricevuto tutti i pacchetti correttamente ma gli ACK dei pacchetti sono ancora in viaggio o persi.



Numerazione PDU

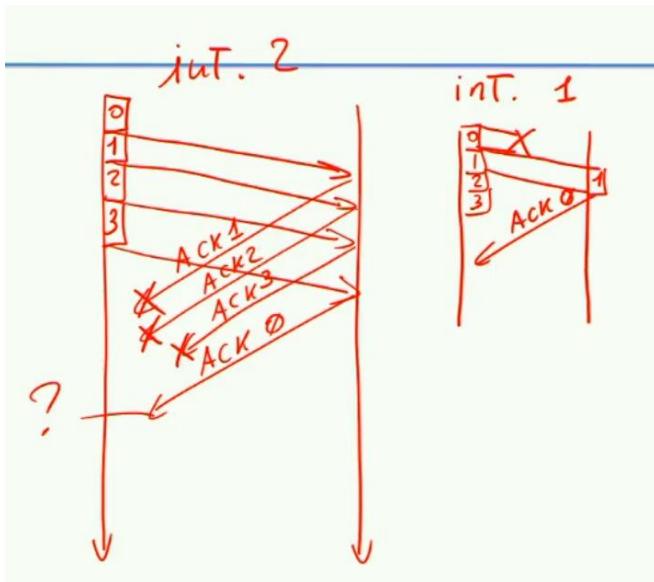
Idealmente i numeri delle PDU devono essere tutti diversi, tuttavia la numerazione delle PDU è ciclica (cioè devo scrivere il numero di sequenza in un campo di bit).

Se ho k bit di numerazione, avrò una numerazione modulo 2^k .



Non posso utilizzare una finestra di trasmissione pari alla numerazione.

ESEMPIO K=2, modulo ($2^k=4=w_T$)



Vi sono ambiguità. Ci sono varie interpretazioni. Si possono essere persi 3 ACK ma pacchetti ricevuti, oppure perso solo il primo pacchetto e quindi il ricevitore attende ancora quello.

$w_T < 2^k$ SEMPRE

Riassunto Go Back N

Il trasmettitore è significativamente più complesso rispetto al caso Stop and Wait (maggiore quantità di memoria, gestione dell'orologio e algoritmi).

Il ricevitore rimane inalterato (perché finestra pari a 1).

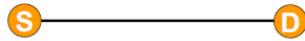
Si possono usare conferme multiple su gruppi di PDU grazie alla semantica cumulativa (orologio al ricevitore)

La finestra di trasmissione non può avere dimensioni arbitrarie: $W_T < 2^k$

ESERCIZIO 1 – STOP & WAIT

1 - Stop & Wait

Si consideri una topologia di rete lineare composta da un singolo canale di capacità pari a 1000 kbit/s.



Il nodo S deve trasmettere un file di 8000 byte verso il nodo D. Si supponga che:

- la rete sia scarica e non vi siano errori di trasmissione
- il tempo di propagazione sul canale sia pari a 10 ms
- la dimensione massima dei pacchetti trasmessi sul canale, comprendenti 40 byte di intestazione, sia di 1500 byte
- venga usato un protocollo Stop & Wait con ACK di dimensione trascurabile

Si determini il tempo necessario perché D riceva tutto il file

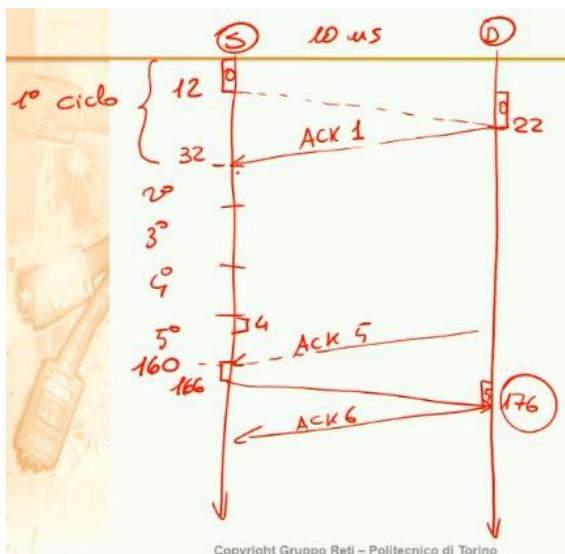
$$\frac{8000}{1460} = 6 \text{ pacchetti}$$

$$5 \times 1500 \text{ B} + 1 \times 740 \text{ B}$$

$$T_{TX,1} = 12 \text{ ms}$$

$$T_{TX,2} = 6 \text{ ms}$$

Dimensione ACK trascurabile vuol dire che hanno dimensione talmente piccola che il tempo di trasmissione è pari a 0 (ma il tempo di propagazione rimane).



La ricezione del file termina appena l'ultimo pacchetto arriva a D, ed è all'istante **176 ms**.

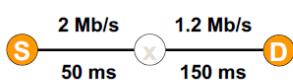
ESERCIZIO 2 – STOP & WAIT

2 - Stop & Wait

Ripetere l'esercizio 1 per la topologia in figura, composta da due canali aventi capacità e tempi di propagazione indicati.

Il nodo di commutazione 'x' opera in modalità store and forward

Il protocollo Stop&Wait opera tra sorgente e destinazione (S e D)



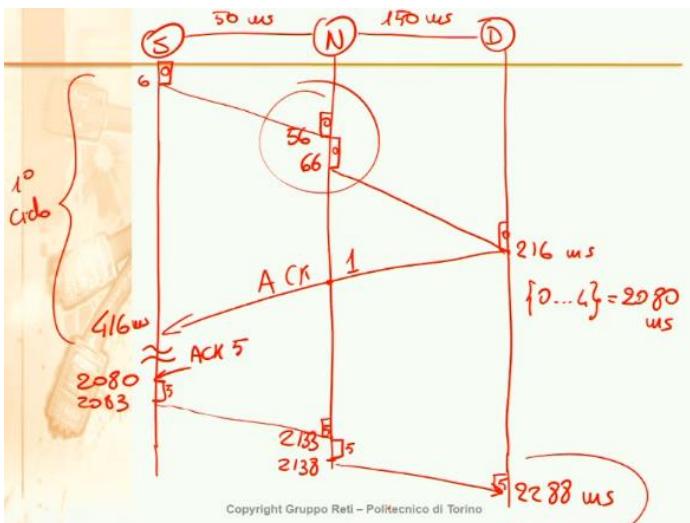
$$T_{TX,1}^{(1)} = 6 \text{ ms}$$

$$T_{TX,2}^{(1)} = 3 \text{ ms}$$

$$T_{TX,1}^{(2)} = 10 \text{ ms}$$

$$T_{TX,2}^{(2)} = 5 \text{ ms}$$

ACK ha tempo di trasmissione nullo (quindi non faccio lo store & forward al nodo N)



I cicli si ripetono per i pacchetti da 0 a 4 e l'ultimo avendo dimensione diversa ha tempi di trasmissione diversi.

Il tempo totale è **2288 ms.**

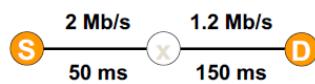
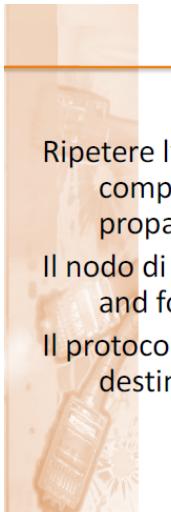
ESERCIZIO 3 – STOP & WAIT

3 - Stop & Wait

Ripetere l'esercizio 1 per la topologia in figura, composta da due canali aventi capacità e tempi di propagazione indicati.

Il nodo di commutazione 'x' opera in modalità store and forward ed ha buffer *infinito*

Il protocollo Stop&Wait opera tra nodo X e destinazione D

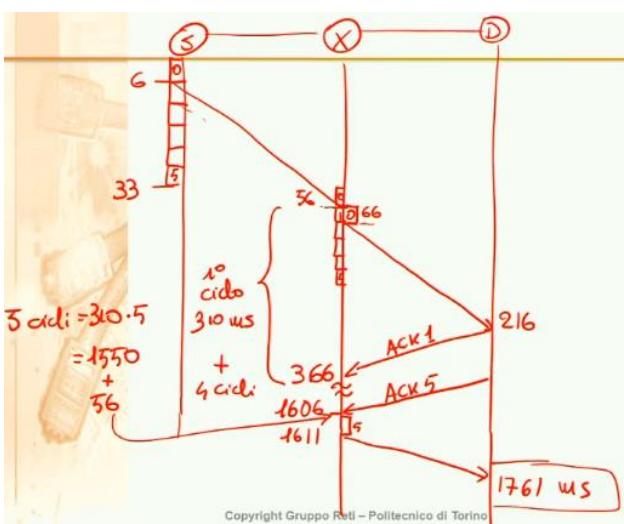


Esercizio simile al precedente ma il protocollo stop & wait opera tra nodo X e destinazione D.

Informazione necessaria è il buffer del nodo di commutazione.

Il nodo S invierà tutti i pacchetti perché non è limitato.

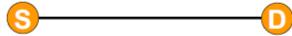
L'ACK si fermerà su X.



ESERCIZIO 1 – GO BACK N

1. Go-Back-N

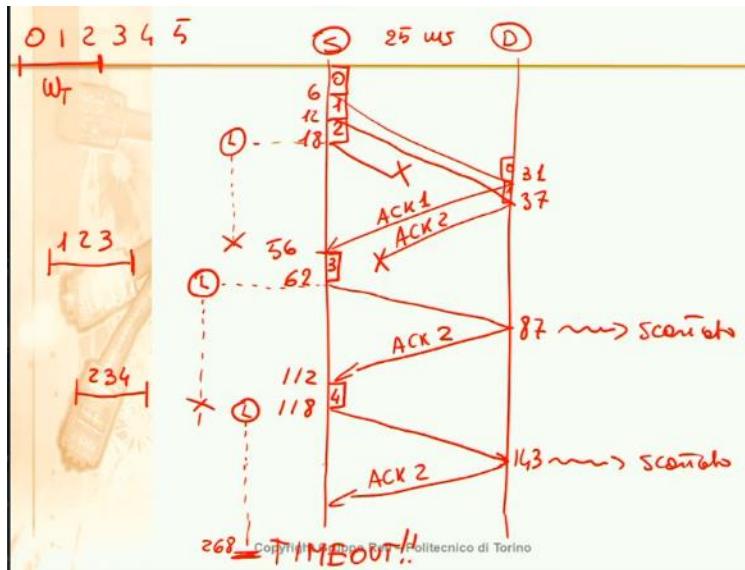
Si consideri una topologia di rete lineare composta da un singolo canale di capacità pari a 2 Mbit/s.



Il nodo S deve trasmettere un file di 8000 byte verso il nodo D. Si supponga che:

- la rete sia **scarica**
- errori di trasmissione pregiudichino la ricezione del *terzo* pacchetto e del *secondo* ACK
- il tempo di propagazione sul canale sia pari a 25 ms
- il timeout sia pari a 150 ms
- la dimensione max dei pacchetti trasmessi sul canale, comprendenti 40 byte di intestazione, sia di 1500 byte
- sia usato un protocollo Go-back-N con finestra $W_T=3$ e ACK di dimensione trascurabile

Si determini il tempo necessario perché D riceva tutto il file



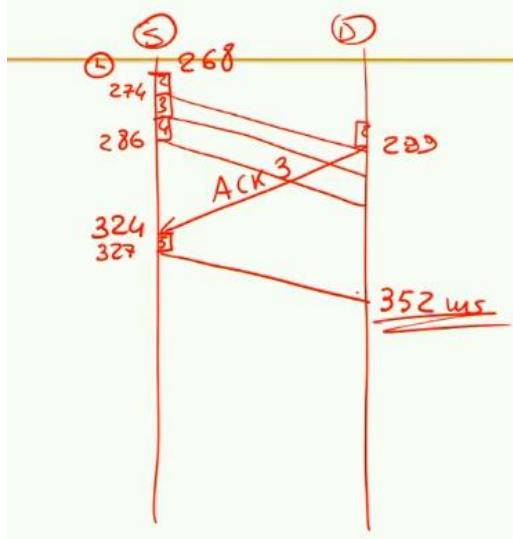
Con la ricezione di ACK1 la finestra si sposta su 123

L'ACK2 si perde e il terzo pacchetto si perde. Viene fissato un timer che si azzerà con l'invio del pacchetto 3 (possibile grazie alla ricezione di ACK1). Il pacchetto 3 viene consegnato, ma il trasmettitore ha perso il pacchetto 2 e quindi il ricevitore lo scarta e ribadisce ACK 2.

La finestra avanza nuovamente (234) ma non essendo ancora scaduto il timer, viene mandato il pacchetto numero 4. Questo invio azzerà nuovamente il timer e anche il pacchetto 4 viene ricevuto, scartato e viene inviato nuovamente ACK2.

Adesso ACK2 non fa spostare la finestra (perché già ricevuto) e il trasmettitore a questo punto aspetta che il timer scade. Una volta scaduto (al tempo 268) vi è il **timeout**. Col timeout si devono ritrasmettere **tutti i pacchetti** in finestra, perciò si riprende con l'invio del pacchetto 2.

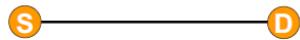
pacchetti in finestra, perciò si riprende con l'invio del pacchetto 2.
Il tempo totale sarà **352 ms**.



ESERCIZIO 2 – GO BACK N

2. Go-Back-N

Si consideri una topologia di rete lineare composta da un singolo canale di capacità pari a 2 Mbit/s.



Il nodo S deve trasmettere un file di 18000 byte verso il nodo D. Si supponga che:

- la rete sia scarica e non vi siano errori di trasmissione
- il tempo di propagazione sul canale sia pari a 25 ms
- la dimensione max dei pacchetti trasmessi sul canale, comprendenti 40 byte di intestazione, sia di 1500 byte
- Sia usato un protocollo Go-back-N con finestra $W_T=4$ e ACK di dimensione trascurabile

Si determini:

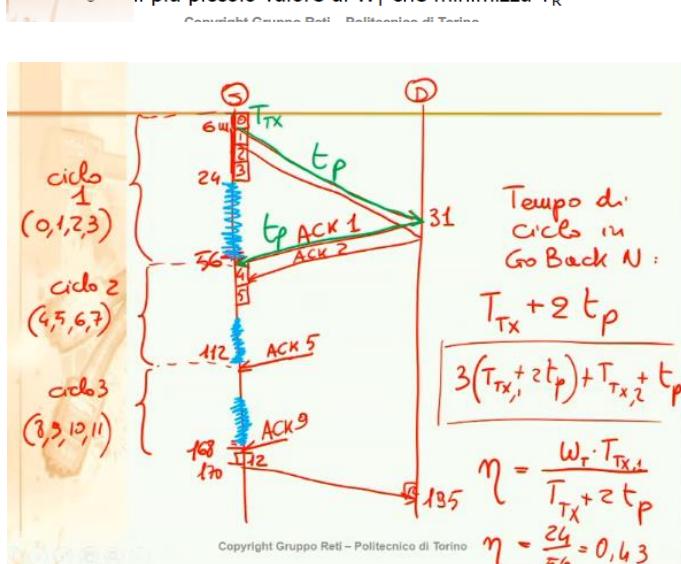
- il tempo necessario T_R perché D riceva tutto il file
- l'efficienza del protocollo
- il più piccolo valore di W_T che minimizza T_R

$$\frac{18000}{1460} = 13 \text{ pacchetti}$$

$$12 \times 1500 \text{ B} + 1 \times 520 \text{ B}$$

$$T_{TX,1} = 6 \text{ ms}$$

$$T_{TX,2} = 2 \text{ ms}$$



Mentre trasmetto pacchetto 4 sarà in transito ACK 2 e così via. Ciò che conta è il tempo necessario per ACK1 a tornare indietro. Quello determina il ciclo di trasmissione.

Tempo di ricezione: **195 ms** calcolato come $3(T_{TX,1}+2T_p)+T_{TX,2}+T_p$

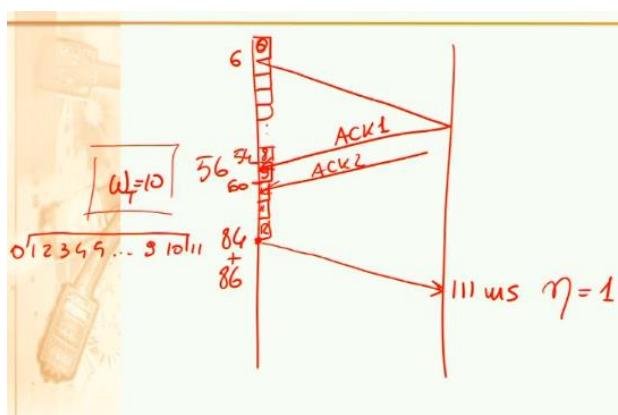
Tempo di ciclo: **56 ms** calcolato come $T_{TX,1}+2T_p$.

Efficienza del protocollo: il canale rimane utilizzato per tutto il tempo che va dal pacchetto 3 al pacchetto 4 (perché il trasmettitore aspetta di vedersi abilitata la trasmissione con i nuovi ACK).

L'efficienza è calcolata come il rapporto tra il tempo in cui il trasmettitore è stato occupato diviso il tempo di ciclo.

$$\eta = \frac{W_T \cdot T_{TX,1}}{T_{TX,1} + 2T_p} = \frac{24}{56} \text{ (leggendo il grafico)} = 0,43$$

Il più piccolo valore di W_T che minimizza T_R si traduce come: qual è il minimo valore di finestra di trasmissione che mi rende l'efficienza pari a 1?



Efficienza 1 vuol dire che il trasmettitore non si ferma mai e trasmette sempre. Bisogna trovare il valore (incrementando W_T) minimo per cui alla ricezione del primo ACK posso continuare a trasmettere.

Con $W_T=9$ mando i pacchetti da 0 a 8 e arrivo a 54 ms (< 56). Non ho ancora finestra pari 1. Se incremento ancora $W_T=10$ noto che alla ricezione del primo ACK la finestra si sposta e posso subito inviare il pacchetto successivo.

Selective repeat ($W_T=N$, $W_R=N$)

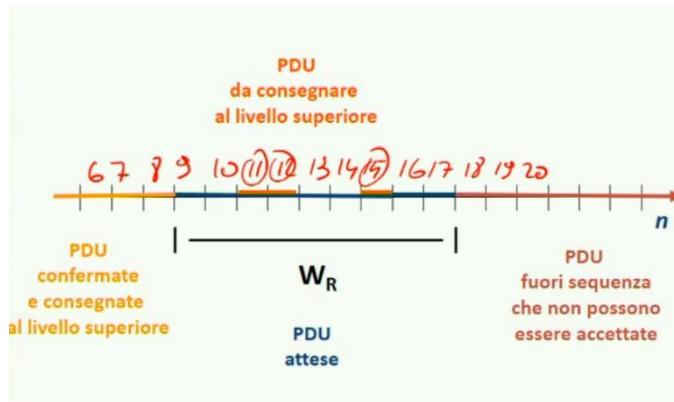
Il selective repeat è una variazione del protocollo Go Back N dove modifichiamo il ricevitore. Gli permette di immagazzinare pacchetti fuori sequenza aumentando la finestra di ricezione.

Nel protocollo Go Back N il ricevitore può accettare solo PDU in sequenza. È possibile accettare PDU corrette, ma fuori sequenza, migliorando le prestazioni: ottengo il protocollo **Selective repeat**. Questo protocollo usa finestra di trasmissione e finestra di ricezione di dimensioni **maggiori di 1** (di solito di pari dimensione).

Esistono diverse possibili implementazioni:

- Uso di ACK selettivi o cumulativi;
- Uso di timer associati alle singole PDU o alla finestra
- Comportamenti del trasmettitore e del ricevitore

Si utilizzeranno ACK cumulativi e timer associati alla finestra.



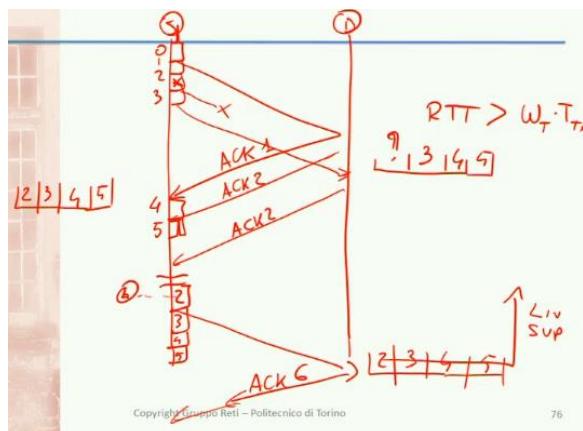
Pacchetti con PDU inferiore a 9 ricevute e consegnate ai livelli superiori. I pacchetti con numerazione superiore a 17 non possono essere accettati. Tutto ciò che è compreso tra 9 e 17 può essere accettato. Si suppone che 11, 12 e 15 siano stati già ricevuti.

Questo protocollo è molto utile nei canali **non sequenziali** poiché si ha modo di immagazzinarli nell'attesa che si completi la sequenza.

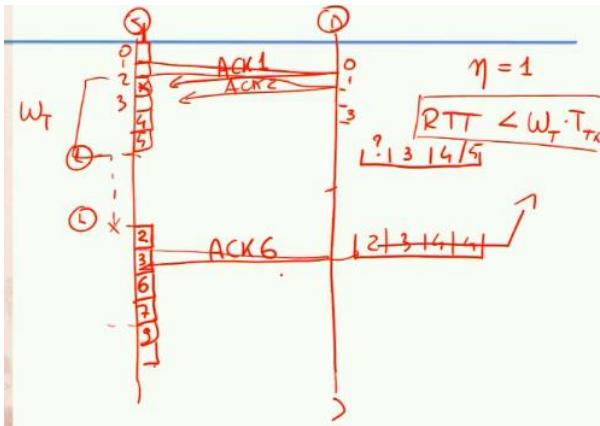
Il trasmettitore è **identico al protocollo Go-Back-N**.

Il ricevitore adesso deve:

- Ricevere una PDU
- Controllare la **correttezza** della PDU
- Controllare **numero di sequenza**
- Se la PDU è corretta ed in sequenza viene **consegnata al livello superiore** (eventualmente a tutte le altre PDU ricevute in sequenza)
- Se la PDU è corretta ma **non in sequenza**: se è dentro la finestra di ricezione la **memorizza**, altrimenti la scarta.
- Invia comunque un **ACK relativo** all'ultima PDU **ricevuta in sequenza**.



In caso di perdita singola, questa versione del protocollo si comporta come il go back N in termini di velocità di trasmissione (throughput) e occupazione del canale.



Si ottengono vantaggi rispetto al go back N se $RTT < \text{tempo trasmissione della finestra}$ (implica efficienza 1), poiché il nuovo ACK permette di spostare in avanti la finestra prima di completarne la ritrasmissione, oppure si hanno vantaggi anche in presenza di perdite ripetute sui dati poiché si memorizza in ricezione ed è sufficiente che una sola copia di ogni pacchetto sia arrivata al ricevitore.

Guardando il disegno, si nota che pur perdendo il pacchetto 2, ma avendo ricevuto il resto, appena il pacchetto viene ritrasmesso, il trasmettitore non

ritrasmette tutta la sua finestra, perché il ricevitore risponderà con ACK 6.

Modificando il comportamento del trasmettitore, e vincolandolo a ritrasmettere solo il primo pacchetto (perso) nella finestra si riduce l'occupazione del canale (si recupera 1 pacchetto perso ogni RTT)

Migliori prestazioni si hanno adottando ack selettivi

Nel protocollo selective repeat è presente una relazione stretta tra la dimensione dello spazio di numerazione e la dimensione delle finestre:

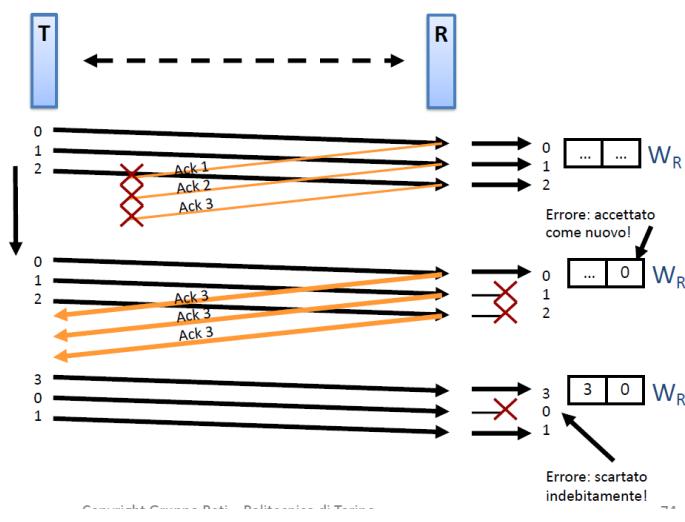
$$W_T + W_R \leq 2^k$$

Nel Go back N vale $W_T < 2^k$, che sostituendo sopra si ha $W_T + 1 \leq 2^k$ (che vuol dire $W_R=1$ e eliminando l'1 si elimina l'uguale e si arriva alla forma del Go Back N).

L'espressione completa è quindi quella indicata in grassetto.

ESEMPIO – VIOLAZIONE DELLA RELAZIONE

$W_T=3$, $W_R=2$, $k=2$ con ipotesi timer su finestra



Il trasmettitore invia 3 pacchetti, si suppone vengano ricevuti. Se un pacchetto è in ordine, transita direttamente al livello superiore e non si ferma nella finestra di ricezione W_R .

Il ricevitore risponde con 3 ack che vengono persi.

Il risultato è che il trasmettitore, non vedendo ACK, attende il termine del timer (**timeout**) e ritrasmette tutto ciò che ha in finestra (cioè i pacchetti 012).

Il ricevitore ha un posto libero per il pacchetto 3 (che sta attendendo) più un altro eventuale posto per un nuovo pacchetto fuori sequenza successivo al 3 (quinto pacchetto che è

numerato 0). Quando arriva il pacchetto 0 il ricevitore pensa che sia fuori sequenza e viene accettato **come se fosse nuovo**. I pacchetti 1 e 2 sono entrambi fuori sequenza e non entrano nella finestra, perciò vengono scartati. Si invia nuovamente ACK 3 perché continua ad essere quello il pacchetto atteso in sequenza. Il trasmettitore, ricevendo ACK 3, può spostare in avanti la sua sequenza e invierà i pacchetti 301. Il 3 verrà accettato perché riempie il buco lasciato in finestra, ma il nuovo pacchetto 0 (cioè il quinto pacchetto) verrà scartato poiché già presente nella finestra del ricevitore.

Non rispettare la relazione finestra/numerazione ha avuto come conseguenza: 1 pacchetto erroneamente accettato due volte ed 1 pacchetto erroneamente scartato.

Efficienza e Throughput

In generale, l'efficienza è data dal rapporto:

$$\eta = \frac{\text{tempo di attività}}{\text{tempo di ciclo}}$$

L'efficienza di un protocollo Stop & Wait ideale senza errori è pari a $\eta = \frac{T_{TX}}{T_{TX} + 2t_p} = \frac{1}{1+2a} < 1$ che tiene conto del tempo necessario a trasmettere il pacchetto, tempo per ricevere completamente il pacchetto e il tempo per ricevere correttamente l'ACK (tempo di ciclo).

Il parametro $a = \frac{t_p}{T_{TX}}$ serve per identificare qual è la relazione tra la velocità del canale e la distanza tra trasmettitore e ricevitore. Questo parametro non sarà mai 0 (tempo di trasmissione infinito), perciò la relazione è strettamente minore.

Nel caso del Go-Back-N se la durata della finestra è tale da proseguire oltre l'istante di tempo in cui ritorna il primo ACK, il risultato è che non ci si ferma mai e l'efficienza è 1.

$$\eta = \begin{cases} 1 & W_T \geq 1 + 2a \\ \frac{W_T}{1 + 2a} & W_T < 1 + 2a \end{cases}$$

Viceversa, se l'efficienza è data dal rapporto di cui sopra. Tale rapporto deriva da:

$$\frac{(W_T \cdot T_{TX})}{T_{TX} + 2t_p}$$

Raccogliendo e semplificando per il tempo di trasmissione si ottiene l'espressione di cui sopra.

Queste formule permettono di calcolare il **throughput** (traffico smaltito).

$$\theta = \eta C$$

Ad esempio, un canale con capacità $C=100$ Mb/s e con efficienza 0.3, avrà throughput di 30Mbps.

$$\theta = \eta C \propto \frac{W_T}{a}$$

Questo vuol dire che connessioni corte ottengono throughput più elevati. È possibile regolare il throughput agendo sulla **dimensione della finestra** e sul **Round Trip Time (RTT)**.

ESERCIZIO 1- SELECTIVE REPEAT

1. Selective Repeat

Si consideri una topologia di rete lineare composta da un singolo canale di capacità pari a 2 Mbit/s.



Il nodo S deve trasmettere un file di 18000 byte verso il nodo D usando un protocollo Selective Repeat con $W_T=4$ e $W_R=3$ (dim. trasc. ACK). Si supponga che:

- il tempo di propagazione sul canale sia pari a 2 ms
- la dimensione max dei pacchetti trasmessi sul canale, comprendenti 40 byte di intestazione, sia di 1500 byte
- Si verifichi un errore che pregiudichi la ricezione del **sesto, settimo e ottavo** pacchetto della sequenza (timeout=60ms)

Si determini il tempo T_R perché D riceva tutto il file

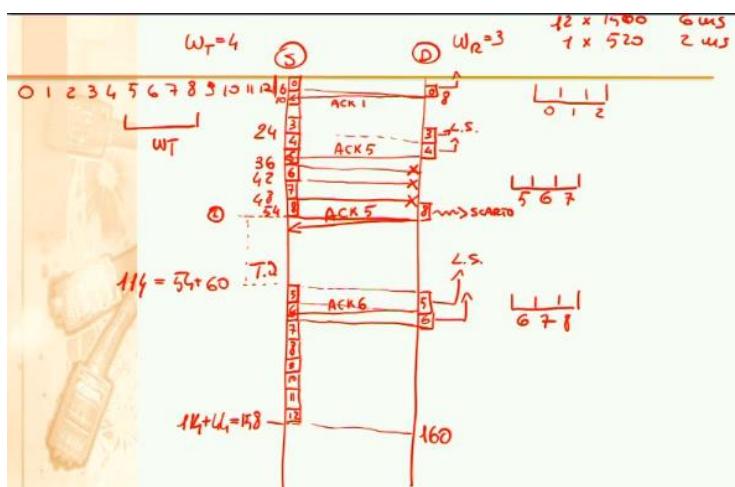
$$\frac{18000}{1460} = 13 \text{ pacchetti}$$

$$12 \times 1500 \text{ B} + 1 \times 520 \text{ B}$$

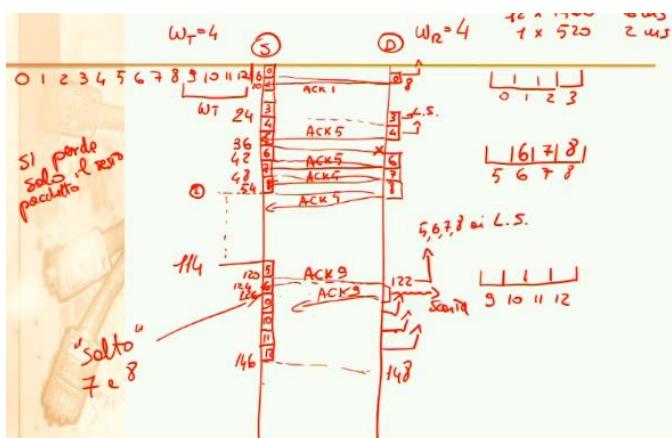
$$T_{TX,1} = 6 \text{ ms}$$

$$T_{TX,2} = 2 \text{ ms}$$

$$W_T = 4, W_R = 3$$



$W_T = 4, W_R = 4$ e solo perdita sesto pacchetto



I pacchetti 567 vengono persi, il pacchetto 8 viene scartato e allora parte il timer. Dopo 60ms (al tempo 114) si ha **timeout**. L'intera finestra viene ritrasmessa (5678) e non essendoci più errori si riprende la trasmissione senza interruzioni.

In questo caso il selective repeat si è comportato esattamente come il Go-Back-N.

Come prima, la finestra viene ritrasmessa, ma appena si riceve il pacchetto 5, la finestra di ricezione viene spostata e si viene avvisati con un ACK9. Perciò si saltano i pacchetti 7 e 8.

5. Strato Fisico

Mezzi trasmissivi

I mezzi trasmissivi sono principalmente 3:

- **Elettrici** (come doppini non schermati o cavo coassiale)
- **Ottici** (fibra ottica)
- **Radio**

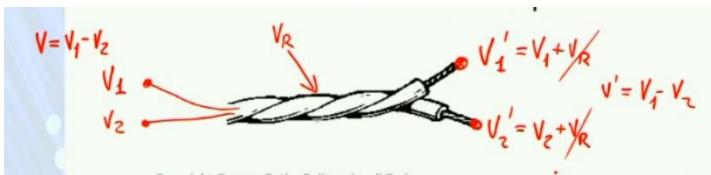
In generale i mezzi trasmissivi hanno caratteristiche legate alle caratteristiche fisiche (resistenza, capacità parassite, impedenze) ma anche aspetti meccanici (come la resistenza alla trazione, flessibilità) ed è anche legato alla facilità di collegamento dei ricetrasmettitori.

Mezzi elettrici

I mezzi elettrici in generale devono avere resistenze, capacità parassite ed impedenze **basse**. Devono avere una buona resistenza alla trazione, flessibilità e facilità di collegamento dei ricetrasmettitori.

Le caratteristiche dei mezzi elettrici dipendono da: geometria, numero di conduttori a distanza reciproca, tipo di isolante e tipo di schermatura.

Il Doppino



Viene anche detto coppia (pair) ed è il mezzo trasmissivo classico della **telefonia**. È composto da due file di rame ritorti (binati, twisted) per **ridurre le interferenze** elettromagnetiche utilizzando tecniche

trasmissive differenziali. Siccome vi è applicato del rumore, tale rumore essendo circa pari su entrambi i cavi, viene cancellato facendone la differenza.

I costi sono ridotti e l'installazione è semplice.

La qualità del doppino dipende dalla qualità della venatura del doppino (che deve essere fatta attentamente).

Il connettore RJ45 (del doppino) contiene 8 cavi accoppiati a due a due. Delle 4 coppie una viene utilizzata per trasmettere, una per ricevere e altre due rimangono inutilizzate (vengono utilizzate per trasportare l'alimentazione PoE).

La qualità dei doppini viene suddivisa in categorie.

Il cavo coassiale: è un sistema trasmissivo composto da un connettore centrale e una o più calze di schermo. Ha una maggiore schermatura dai disturbi esterni, ma costi elevati e difficoltà maggiore di installazione. Oggi non viene più utilizzata per le reti locali ma solo su antenne/parabole.

La fibra ottica è composta da un minuscolo e flessibile filo di vetro costituito da due parti con indici di rifrazione diversi. Per la legge di Snell, il raggio luminoso (generato da un LED) introdotto nella fibra entro un angolo di accettazione rimane confinato nel core.

Ha una totale immunità da disturbi elettromagnetici e un'alta capacità trasmissiva (fino a decine di Terabit/s). Possiede una bassa attenuazione (0.1 dB/km) in base alla lunghezza d'onda.

La fibra ha però la difficoltà di essere collegata tra loro ed ha un ridotto raggio di curvatura. Soffre anche le vibrazioni.

Posa di cavi sottomarini: avviene interrando i cavi sul fondo del mare e richiede cavi con amplificatori ottici ridondanti ogni 30/50Km.

Il canale radio (mobile) (detto anche **etero**) è un canale che vede la propagazione del segnale mediante l'uso di antenne. Se almeno uno (tra ricevitore e trasmettitore) è in movimento, viene detto mobile. La qualità della trasmissione dipende dal rapporto tra **potenza ricevuta** e **potenza trasmessa**. Viene regolata dall'equazione di Friis:

$$\frac{P_R}{P_T} = G_T G_R \frac{\lambda^2}{(4\pi D)^2}$$

Sapendo che $f = \frac{c}{\lambda}$

Quindi più aumenta la frequenza, meno strada fa il segnale (perché è inversamente proporzionale alla lunghezza d'onda).

La potenza ricevuta è soggetta ad attenuazione dovuta ai fenomeni atmosferici e all'interferenza da sorgenti alla stessa frequenza. La presenza di ostacoli determina **riflessioni e rifrazioni**, che producono copie del segnale trasmesso disturbando la ricezione: **fading** (ostacolo in movimento) e **shadowing** (ostacolo fisso).

La trasmissione su mezzi fisici

La trasmissione di informazioni a distanza si realizza **associando** bit o simboli di informazione a segnali diversi. Il ricevitore deve **decidere** quale bit o simbolo il segnale ricevuto rappresenta. Vengono utilizzate tecniche diverse a seconda del mezzo fisico utilizzato (codifiche di linea per mezzi elettrici e ottici; modulazioni digitali per mezzi radio, ottici ed elettrici).

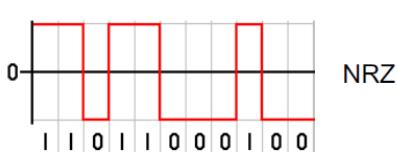
Codifiche di linea

Le codifiche di linea sono tecniche per rappresentare informazioni numeriche (digitali) mediante segnali numerici su mezzi elettrici e ottici. Le codifiche che si usano fanno uso della tensione come modo di rappresentare il segnale. (tensione variabile applicata al mezzo elettrico che viene rilevata dal ricevitore).

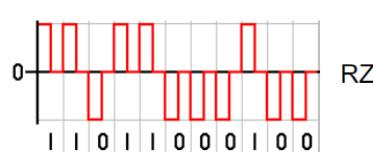
Le **codifiche unipolari** sono molto semplici ed utilizzano un livello di tensione per "0" ed uno per "1". Trasmettitore e ricevitore devono sincronizzarsi sul momento in cui andare a leggere tali valori e a quale **tempo di simbolo**.

Il tempo di simbolo rappresenta l'intervallo di tempo durante il quale il segnale scelto per un simbolo va mantenuto. Proprio per questo, lunghe trasmissioni dello stesso bit possono portare a una perdita del sincronismo.

Le **codifiche polari** utilizzano due livelli di tensione con polarità diverse (riducendo la componente continua). Esistono tre possibili varianti: NRZ (Non return-to-zero) dove si utilizza per lo 0 una tensione negativa; RZ dove si ha una transizione su tensione nulla tra due bit consecutivi; Bifase dove ogni bit è rappresentato da due livelli di tensione di polarità inversa.

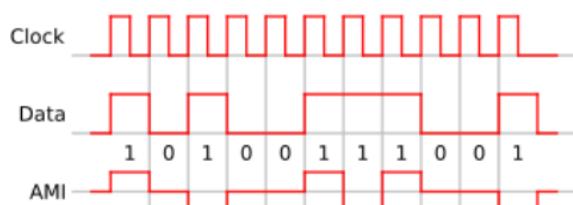


NRZ



RZ

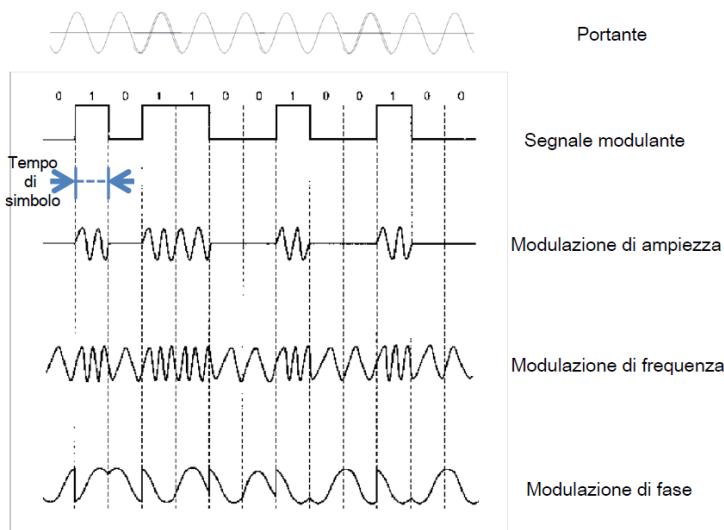
Le bifase sono migliori per il sincronismo, ma RZ e bifase richiedono bit rate elevati pari al doppio del bit rate che voglio segnalare.



Le **codifiche bipolari** utilizzano una tensione nulla per rappresentare "0" e due polarità opposte per "1", utilizzate in alternativa. Vengono chiamate anche AMI e permettono l'uso di simboli ternari (-1, 0, +1) come nella codifica 8B6T (8 bit

codificati con 6 simboli ternari). Così facendo posso rappresentare i bit con delle codifiche che richiedono meno cambi di stato del numero di bit.

Si può generale con le **codifiche nBmB**: si tratta di codifiche in cui simboli di n bit sono rappresentati da simboli di m bit, con $n < m$ (es. 4B5B, 8B10B, 64B66B...). Sono molto popolari perché richiedono meno banda di codifiche polari, permettono il controllo sulla scelta delle parole di codice, limitando quelle con troppi 0 e 1 consecutivi, limita la componente continua e fornisce caratteri speciali per delimitazione pacchetti, trasmissione in idle o padding.



Nei mezzi in cui la trasmissione richiede la percorrenza di un mezzo come l'etere non è possibile utilizzare le tecniche descritte in precedenza ma tramite **modulazioni digitali**.

Le modulazioni digitali sono tecniche adatte per segnali ondulatori e l'informazione è impressa su di un segnale sinusoidale portante variandone l'ampiezza, la frequenza, la fase o loro combinazioni. Tali variazioni sono codificate in un segnale **modulante**. Le modulazioni digitali vengono chiamate **Shift Keying** (ASK, FSK, PSK).

La sigla è di solito preceduta da un numero che indica il numero di simboli del segnale modulante, come ad esempio:

- 8-PSK -> modulazione di fase a 8 simboli (ogni simbolo può rappresentare 3 bit)
- 64-QAM -> modulazione di fase/ampiezza a 64 simboli (6 bit per simbolo)

Modulazioni con alto numero di simboli (= maggior bit rate) si possono usare solo se il canale non distorce eccessivamente il segnale ricevuto.

Le Reti di Accesso e Trasporto

Un utente si collega ad una rete sfruttando la rete di accesso e di trasporto. La rete di accesso comprende apparati e mezzi trasmissivi che collegano l'utente col nodo di accesso del gestore di servizi di TLC. La rete di trasporto comprende apparati e mezzi trasmissivi appartenenti ad uno o più gestori di servizi di TLC e destinati al transito di dati tra due nodi di accesso.

Le reti di accesso possiedono diverse classificazioni: **residenziali** (quelle che arrivano in casa), **mobile** (quelle degli smartphone, chiavette, etc..) e **istituzionali** (come quelle del Politecnico).

Le reti di trasporto possono essere: **metropolitano**, rete che raccoglie in tutta la città le reti di accesso e le trasporta verso le reti di trasporto **nazionali**, ovvero che si diramano lungo tutto il territorio nazionale (lungo le autostrade, per esempio) e **sovranazionali** (presenti sui fondali degli oceani).

Reti di accesso

Reti che servono per arrivare all'utenza residenziale, dette "Ultimo miglio", ovvero l'ultima tratta di rete. Le principali tecnologie nelle reti di accesso sono: **DSL, PON, HFC, Accesso Broadband Mobile**. Tecnologie secondarie oppure obsolete sono: reti via radio WI-MAX, reti ISDN e reti satellitari.

DSL e ADSL

	ADSL	ADSL2	ADSL 2+
Downstream	6 Mb/s	8 Mb/s	24 Mb/s
Upstream	1,5 Mb/s	3.5 Mb/s	3.5 Mb/s

DSL (Digital Subscriber Line) è una famiglia di tecnologie che fornisce l'accesso dati. La più diffusa è l'**ADSL (Asymmetric DSL)**

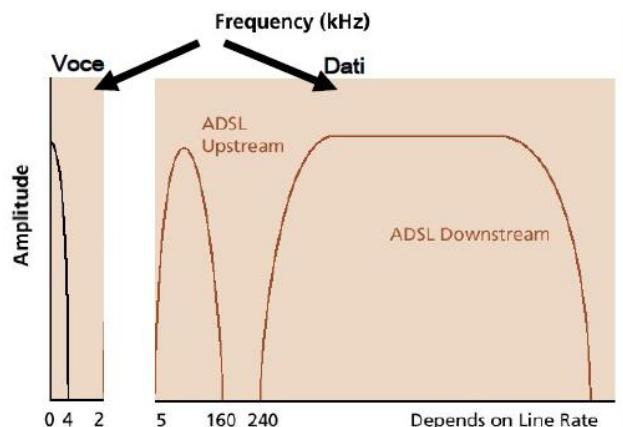
ovvero una velocità maggiore in downstream (o downlink) che in upstream (o uplink). Le velocità teoriche delle ADSL dipendono dalla distanza utente-centrale e dalla versione di tale tecnologia.

L'accesso alla DSL viene fornito dal **modem**: **Modulatore e DEModulatore**. Si utilizza per effettuare trasmissioni digitali su cavi della rete telefonica pubblica. Trasformano il segnale da digitale ad analogico e viceversa. Rendono il segnale idoneo alla trasmissione su rete pubblica in tecnologia analogica su bande adiacenti a quella fonica.

Il MODEM ADSL non va più in banda sonora ma utilizza frequenze più elevate.

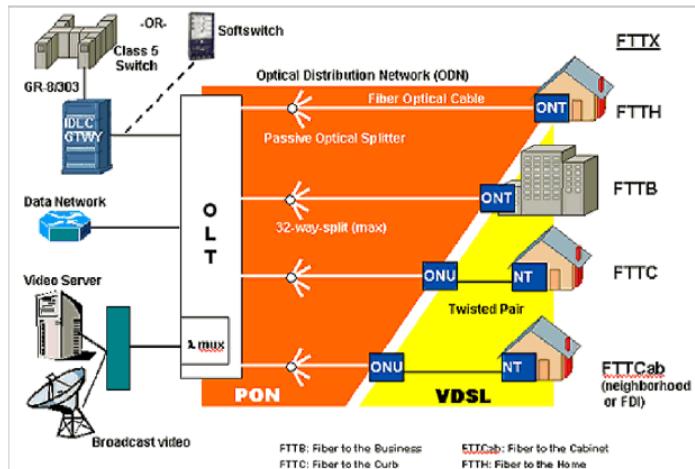
Quando si è sviluppata la tecnologia DSL, l'esigenza era quella di non installare nuove reti di accesso per la città, ma utilizzare quella che tutti già a casa hanno tramite il cavo di rame. Il segnale telefonico è molto limitato in banda, ma il cavo di rame può portare segnali con bande molto alte. Sfruttando questa idea l'accesso ADSL è stato creato portando il MODEM a casa degli utenti, che trasmetteva a frequenze elevate senza interferire con le chiamate telefoniche. POTS Splitter: filtro che "filtrava" telefonate e dati. Le telefonate vanno verso un public switch, i dati vanno verso il DSLAM (DSL Access Multiplexer).

L'armadio trasmette su un cavo (dall'armadio di strada alla centrale di zona) ad anello, raccogliendo i dati che sono stati convogliati verso il DSLAM. Se la distanza tra casa e "armadio" è contenuta, oggi si utilizzano le VDSL (Very-High-Rate DSL) che permette di aumentare di molto i bit rate e utilizza bande fino a 12 MHz.



La distanza è una delle caratteristiche cruciali perché il segnale si degrada col quadrato della distanza.

Le **passive optical networks (PON)** sono un complemento dell'architettura per la connettività di ultimo miglio in fibra ottica. La **ONU** (Optical Network Unit) sta sopra il tradizionale armadio di rete, che contiene le terminazioni in fibra ottica per portare tutti i dati dall'armadio sull'anello precedente (della DSLAM).



Le diverse combinazioni di dove si trova l'ONU sono mostrate in figura ed hanno degli effetti diversi. E' importante perché nella PON ho ordini di grandezza maggiori rispetto alla VDSL. In sostanza la PON sostituisce (FTTH) o integra (FTTCab) la connettività VDSL. Più la tratta in rame è breve, meno attenuazione si ha.

Le **reti di accesso HFC** sono più diffuse negli stati uniti e nord europa, sfruttando la diffusione della TV via cavo, cioè in casa degli utenti arriva, oltre al cavo rame, anche un cavo coassiale. L'idea è simile all'idea ADSL, moltiplicando sia per la televisione che per la connessione e l'utente ha un **cable modem** perché utilizza un cavo invece del doppino.

Reti mobili a banda larga

In genere sostituiscono o affiancano la connettività ADSL.

Il principio della rete cellulare si basa sulla copertura capillare del territorio mediante antenne di portata limitata. Oltre alla copertura, le reti cellulari devono gestire la mobilità degli utenti che si distinguono in **roaming** (rintracciabilità dell'utente sul territorio) e **handover** (continuità della connessione nel passaggio da una cella all'altra).

Architettura di una rete cellulare (LTE)

L'antenna è la parte più evidente che si nota di una architettura di rete cellulare. eNodeB è l'apparato dell'operatore che riceve i segnali, li demodula e li trasforma in segnali elettromagnetici. La parte composta da antenna ed eNodeB fa parte dalla rete di accesso.

La rete di trasporto è composta da una serie di dispositivi tra cui un database che si ricorda degli ID degli utenti, ed altre informazioni (sull'abbonamento, etc..) e viene detto HSS. Altri apparati come l'MME gestiscono le segnalazioni per far passare i dati e farli andare verso l'antenna corretta.

Le tecnologie cellulari vengono distinte in base alle generazioni di tali reti. Ad oggi ne esistono **5**. Molte di queste tecnologie vengono tutt'ora utilizzate perché molte volte la tecnologia più moderna può non essere disponibile in alcune zone, questo perché le vecchie tecnologie utilizzano modulazioni più robuste che gli permettono di resistere in zone dove il segnale è più debole.

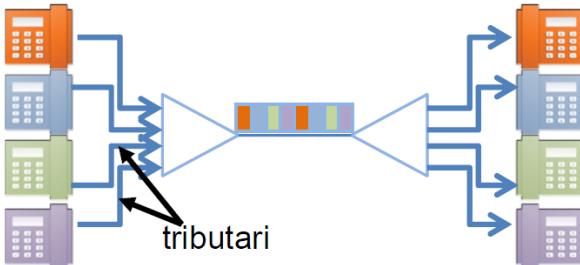
Riguardo la telefonia gli operatori continuano ad utilizzare la tecnologia **GSM** perché è già presente sul territorio, funziona bene e non occupa la parte dati che oggi frutta molto agli operatori.

In alcuni casi il servizio voce viene offerto anche come Voice-over-LTE con una qualità audio maggiore.

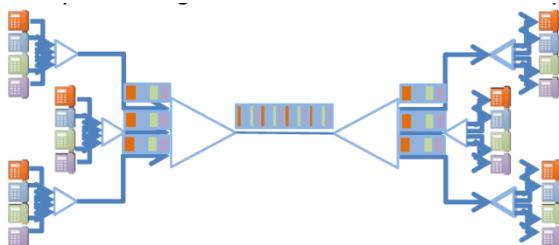
Reti di trasporto

La rete di trasporto crea l'interconnessione tra reti di accesso. Come nodi, ha commutatori telefonici e dati (router) in una topologia magliata e gerarchica. I nodi sono connessi da linee ad alta velocità (solitamente in fibra ottica). La rete di trasporto è gestita da più operatori telefonici o dati (**ISP**) in competizione. Alcuni operatori sono proprietari delle infrastrutture, mentre altri affittano l'infrastruttura ed offrono solo il servizio di trasporto.

La trasmissione sulle reti di trasporto è **interamente digitale** e si è evoluta dalla rete telefonica tradizionale. È strutturata secondo principi di **multiplazione gerarchica a divisione di tempo**.



Se ognuno dei telefoni in foto trasmette a 64kb/s (PCM), il cavo deve andare almeno 4 volte superiore per multipliare nel tempo.



Tale struttura si può iterare per ottenere la struttura gerarchica di cui prima.

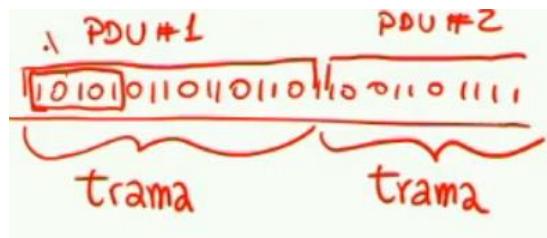
In questo modo si possono inserire tanti tributari su linee sempre più veloci e possono essere trasportate dove si vuole. Dall'altro lato si effettua la demultiplicazione per inviare ogni dato all'utente corretto. Le velocità saranno quindi multipli di 64kb/s.

I sistemi di multiplazione gerarchica si sono evoluti a partire dalla Plesiochronous Digital Hierarchy (PDH): era pensata per canali vocali numerici a 64Kb/s (e non dati); non usava Store-and-Forward: occorre una stretta sincronizzazione end-to-end tra TX e RX; in generale nello strato fisico non si fa mai store and forward; tale sistema era quasi-sincrono (plesio-synchronous) con orologi solo nominalmente entro margini di tolleranza; era possibile l'inserzione/rimozione dinamica di bit lungo il percorso; per mantenere il **sincronismo** si utilizzava la tecnica del **bit stuffing** ovvero, nel caso in cui si riscontrava che qualcuno dei tributari era in ritardo, si aggiungeva un bit "fasullo" che veniva poi ricordato e rimosso successivamente. Questo perché le varie sorgenti non erano sincronizzate tra loro; le velocità di trasmissione erano limitate; gli standard erano diversi in USA/Europa/Giappone.

Negli anni 70/80 ha preso piede la rete **SONET/SDH** (due standard diversi ma compatibili). La PDH usava segnali elettrici e non in fibra ottica, mentre SONET/SDH è pensato per reti in fibra ottica e che sono misto di voce e dati ed è una gerarchia **fortemente sincrona**. La multiplazione funziona sempre con uno schema a divisione **di tempo** (TDM). Prevede delle vere e proprie trame con bit di segnalazione aggiunte.

6. Lo Strato collegamento

Utilizza lo strato fisico e offre un servizio allo strato superiore (Rete). Questo livello interpreta i bit e capisce ciò che viene trasportato al livello inferiore. La prima funzione è la **delimitazione di trama** (PDU, pacchetto), cioè delimita le varie PDU che vengono inviate utilizzando delle regole (ad esempio sequenze prefissate di bit che indicano l'inizio di una trama, altri che ne indicano la lunghezza oppure possono avere lunghezza fissa).



- Un'altra funzione è la **multiplazione**, cioè utilizza degli identificatori che permettono al ricevitore di distinguere a quale entità consegnare i dati (perché ogni livello superiore può avere diversi protocolli).
- L'**indirizzamento locale** serve, nel caso in cui il canale non sia punto-punto, perché lo strato deve dire chi è che riceverà i dati (es. topologia a stella), locale perché ha validità locale e non globale.
- Rilevazione errore aggiungendo CRC, bit di parità o FEC
- Può fare controllo di flusso, sequenza ed errore se ci sono protocolli a finestra che lo fanno
- Lo strato collegamento può utilizzare protocolli ad accesso multiplo, cioè protocolli che regolamentano la trasmissione nel caso in cui ci sono più trasmettitori che possono parlare contemporaneamente.
- Controllo di flusso sull'interfaccia verso i livelli superiori perché in caso di presenza di errori lo strato deve poter far rallentare il livello superiore.

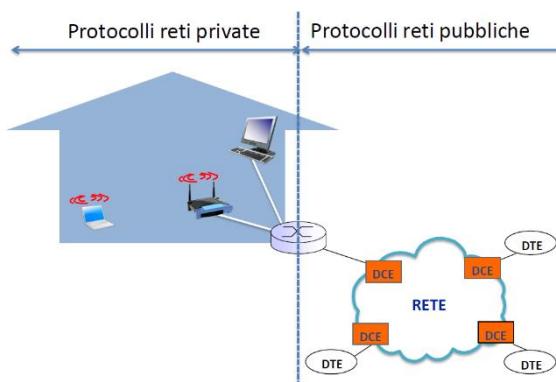
Protocolli di strato collegamento

I protocolli di questo livello derivano dal protocollo SDLC (Synchronous Data Link Control) utilizzato nell'architettura IBM SNA. L'ANSI ha standardizzato SDLC come ADCCP (Advanced Data Communication Control Procedure) e l'ISO come HDLC (High-level Data Link Control). Il CCITT (poi ITU) ha derivato da prima HDLC LAP (Link Access Procedure), e poi LAPB (Link Access Procedure Balanced). Nelle reti locali esiste inoltre un sottostrato (MAC) per risolvere il problema di accesso multiplo.

Alla stessa famiglia di protocolli che derivano da HDLC appartengono: LAPB (Link Access Procedure Balanced-ISDN e X.25), LAPD (Link Access Procedure D-Channel-ISDN), LAPF (Link Access Procedure toFrame Mode BearerService –Frame Relay), LLC 802.2 (LogicalLink Control-LAN), PPP (Point-to-PointProtocol-ADSL), LAPDm(LAP for the mobile Dchannel-GSM).

LLC 802.2 è l'unico che è ancora utilizzato in forma molto più evoluta.

I protocolli di strato 2 si utilizzano principalmente nelle reti pubbliche di accesso (da casa fino alla sede del gestore), si usano nelle reti **locali** ed esistono anche nelle reti pubbliche di trasporto (**ATM**). Le funzioni di strato 2 si dividono in due sottostrati.



Nelle reti pubbliche vi è il **DTE** (Data Terminal Equipment, che sono gli apparati utente) e il **DCE** (Data Circuit-terminating Equipment, che sono gli apparati del gestore). L'insieme di questi due elementi forma le **reti di accesso**. I protocolli di reti private sono differenti da quelli di reti pubbliche.

Caratteristiche comuni

Ogni trama ha un formato generico

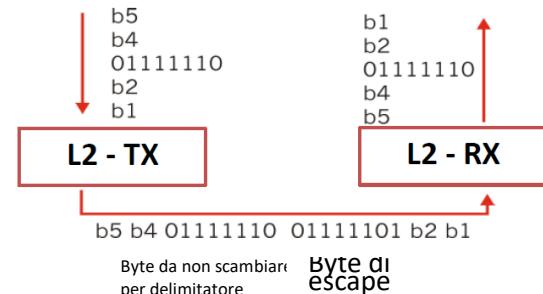
01111110	indirizzo	controllo	dati	CRC	01111110
8	8	8/16	>=0	16	8

La trama inizia e finisce con dei bit ben definiti (sia all'inizio che alla fine). Questi protocolli sono orientati al bit (la lunghezza è multiplo di 1 bit) o al byte (lunghezza multiplo di 8 bit). Il flag di delimitazione 01111110 non può comparire in nessun altro punto della trama per garantire la trasparenza dei dati. I dati, arrivando dallo strato 3 non sanno della presenza di questo limite, perciò è possibile che nel campo dati compaia la sequenza "vietata". In questi casi si utilizzano delle tecniche **bit/byte stuffing**.

Se la trama è organizzata in multipli di byte si utilizza il **byte stuffing**.

Bit stuffing: il trasmettitore inserisce un bit '0' dopo una sequenza di 5 bit '1' consecutivi, eccetto che nel flag. Il ricevitore, dopo aver estratto la SDU, vi elimina ogni bit '0' che segue 5 bit '1' consecutivi.

Byte stuffing: Il trasmettitore inserisce un byte (invece che un bit) di **escape** 01111101 prima di ogni byte uguale al delimitatore (01111110 o 01111101). Il ricevitore scarta il primo 01111101 della coppia. In questo modo col byte di escape si avvisa che il byte che segue **non** è un delimitatore, ma fa parte della trama.



Principali protocolli reti pubbliche analizzati: PPP, ATM.

PPP (Point-to-Point Protocol)

Protocollo standardizzato da RFC (Request For Comments) utilizzato nei collegamenti su linea telefonica o ADSL tra host di utenza residenziale e provider Internet. Si usa anche su connessioni SONET/SDH. Il collegamento punto-punto cablato è più facile da gestire rispetto a collegamenti radio o a canali broadcast in generale. Il protocollo PP è suddiviso in tre sotto-protocollo: Incapsulamento, Link Connection Protocol (LCP), Network Control Protocol (NCP).

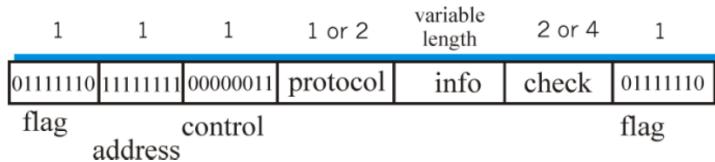
L'obiettivo del protocollo è:

- Eseguire la **delimitazione delle PDU**
- Utilizza il **byte stuffing** per garantire trasparenza del contenuto
- **Riconosce** ma non corregge gli **errori** (non usa protocollo a finestra)
- Esegue multiplazione di più protocolli di strato rete (superiore)
- Controlla l'attività sul collegamento
- Negozia l'indirizzo di livello rete, cioè l'indirizzo globale che permette a un utente di essere riconosciuto in tutto il mondo, tipicamente IP. L'indirizzo viene assegnato al router di casa tramite protocollo PPP.

PPP non fa:

- Correzione errori
- Controllo di flusso
- Mantenimento della sequenza (perché non serve) perché per forza è in sequenza.
- Non gestisce multi-punto

La struttura di una trama PPP è

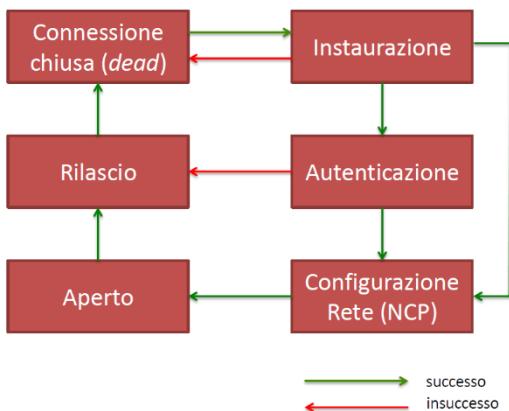


I campi address e control sono fissi perché non vengono più utilizzati.

Il campo flag è il delimitatore (per il "framing") spiegato prima. Address e Control non hanno significato (mantiene compatibilità con HDLC). Il campo Protocol indica il protocollo di livello superiore cui destinare i dati (simile a IEEE 802.2 LLC).

Info contiene i dati del livello superiore e Check identifica gli errori (ma non corregge).

Il protocollo PPP-LCP crea e abbatte il collegamento PPP, negoziando le opzioni (ogni qualvolta si riaccende il router). Inizia nello stato di DEAD e una volta stabilito il collegamento PPP e fatta l'autenticazione, è configurata la tipologia di connessione.



Il PPP è una macchina a stati. Parte dello stato DEAD, successivamente passa nello stato instaurazione in cui si chiedono le autorizzazioni a stabilire il collegamento. Successivamente vi è la fase di autenticazioni per le credenziali di accesso (già inserite nel modem) e poi c'è la fase di configurazione della rete in cui parte il protocollo NCP (per esempio IP). Se tutto ciò va a buon fine, si arriva allo stato Aperto.

ATM – Asynchronous Transfer Mode

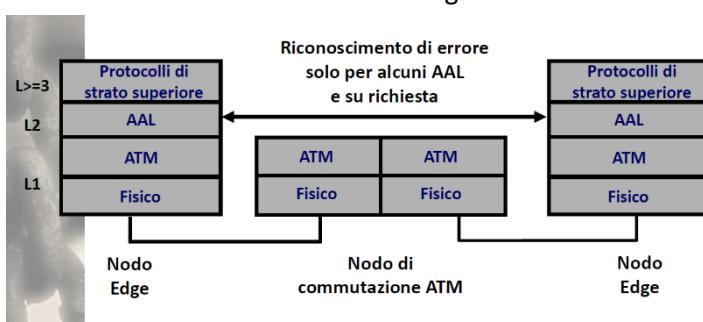
Standard nato nell'ambito di una evoluzione di ISDN chiamata B-ISDN. Si tratta di una rete a pacchetto con servizio a circuito virtuale su scala geografica.

E' una tecnologia che è stata riadattata. Le PDU vengono dette **celle** e hanno dimensione fissa pari a **53 byte** (di cui 48 byte di dati). Grazie a ciò non servono i delimitatori ma basta contare in multipli di 53 byte. In questo caso però è necessario trasmettere **continuamente** altrimenti si perdono dei byte.

ATM si posiziona allo strato due (sopra lo strato fisico) e lo strato 3 è identificato da "Livelli ATM". Il sottostrato AAL è

lo strato di adattamento che serve a gestire i livelli alti che hanno pacchetti di dimensione variabile con

ATM che li ha di dimensione fissa. Lo strato AAL serve a fare **segmentazione**. Lo strato ATM si utilizza per trasportare i dati tra **due nodi di commutazioni**. I due nodi possono essere l'armadio (di strada) e un nodo che sta in sede dell'operatore. Il protocollo ATM raccoglie i dati degli utenti verso la sede, infatti hanno solo 2 livelli.



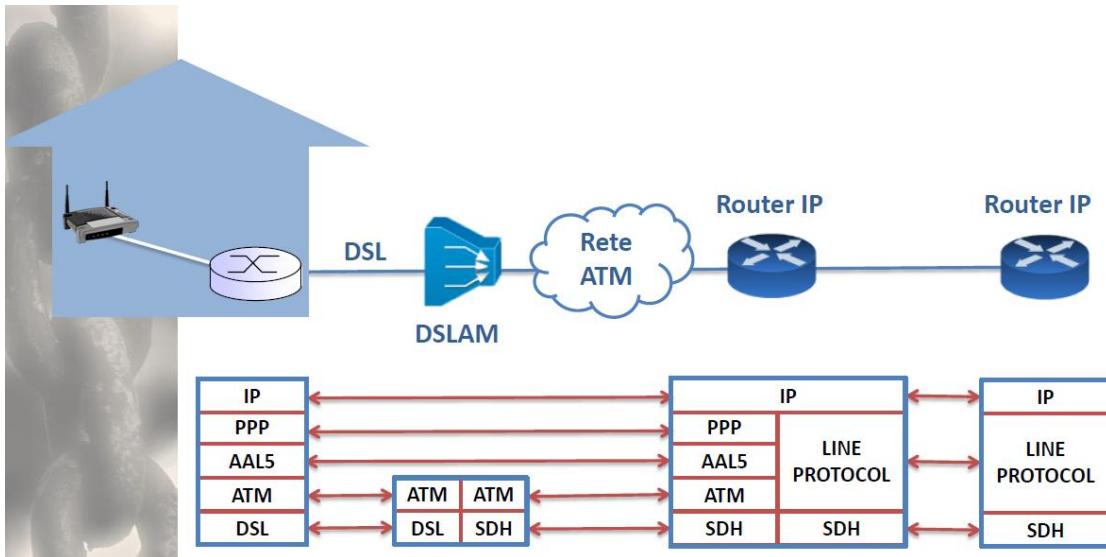
Il formato di una cella ATM è il seguente:

- Intestazione(5 byte) + dati(48 byte)
- Cella di dimensione fissa per ridurre la complessità dei commutatori
- Cella di piccole dimensioni
- Bassa latenza (di Store & Forward)
- Basso ritardo di pacchettizzazione della voce

Commutazione di pacchetto a circuito virtuale: i nodi sanno che guardando l'etichetta dovranno inviare in un nodo piuttosto che in un altro i dati. Visto che tutti i pacchetti ATM vanno sempre verso la sede dell'operatore, tutti i pacchetti andranno verso la sede e grazie all'etichetta si conoscerà il destinatario. Tale etichetta è la parte della dicitura VPI (Virtual Path Identifier)/VCI (Virtual Circuit Identifier). E' lì che il commutatore legge per capire su quale uscita far andare la cella che è appena arrivata. Il campo HEC indica il rilevamento degli errori ma essendo su fibra ottica, tali errori sono molto rari. Il campo CLP indica la priorità delle celle.

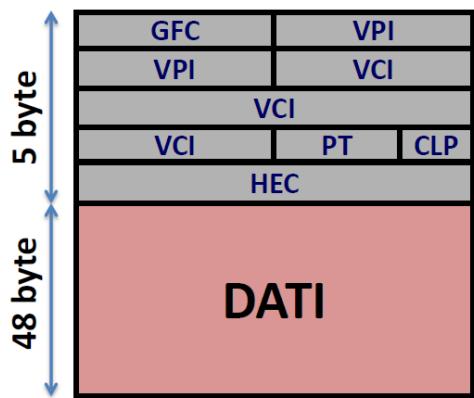
AAL5 è il più semplice tra tutti gli AAL e gestisce la segmentazione di PDU di strato 3 e il loro trasporto nelle celle ADM. Viene divisa la trama in multipli di 48 byte e se necessario viene aggiunto del padding. La cella che trasporta l'ultimo segmento del pacchetto di livello 3 ha il campo PT nell'intestazione ATM a 1. Questa è una violazione del principio di separazione tra strati.

ATM e PPP oggi



Da casa fino al DSLAM (armadio) vi è il protocollo a livello fisico che deriva da DSL. Ci sono poi, soprattutto nei modem di ultimi generazione, dei circuiti ATM che vengono trasportati fino al modem. Al DSLAM inizia la parte dell'operatore e al livello fisico SDH (perché fibre ottiche), al livello 2 vi è ATM e il protocollo PPP è incapsulato dentro ATM (tramite la segmentazione) e il protocollo PPP fa comunicare l'operatore con l'utente (perché è questo il protocollo che negozia con l'operatore). Il PPP non viene toccato dal DSLAM e dall'ATM.

In altre parole ATM trasporta PPP senza saperne nulla e chi riceve riassegna il protocollo ai livelli superiori.

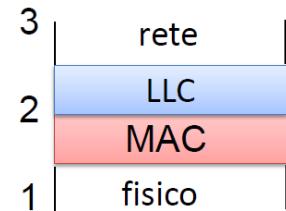


Strato 2 nelle reti private (locali)

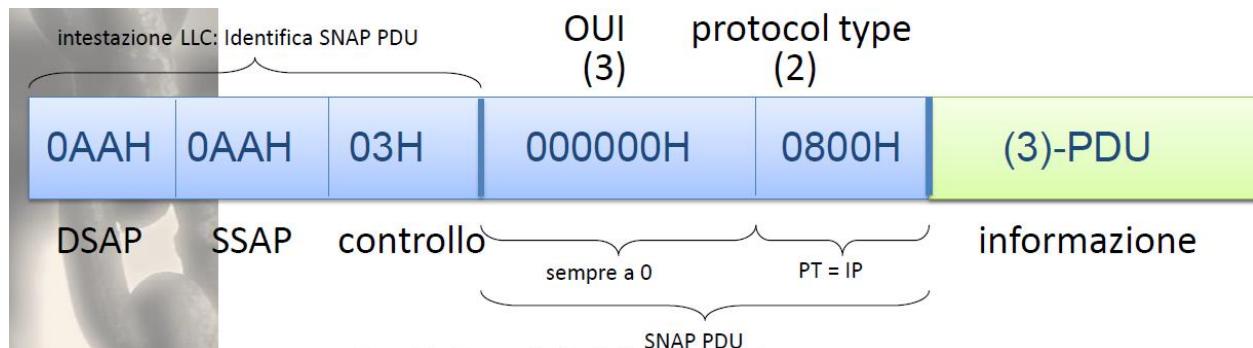
Il livello 2 è diviso in due sottolivelli: **LLC** (Logical Link Control) derivato da HDLC; **MAC** (Medium Access Control).

Il protocollo LLC permette di fare multiplazione per i livelli superiori. Dentro casa, non essendoci PPP a fare multiplazione, LLC sostituisce proprio quest'ultimo.

Il protocollo LLC essendo derivato da HDLC, possiede molti campi che oggi non sono più utilizzati perché molte funzioni sono state spostate in altri livelli.

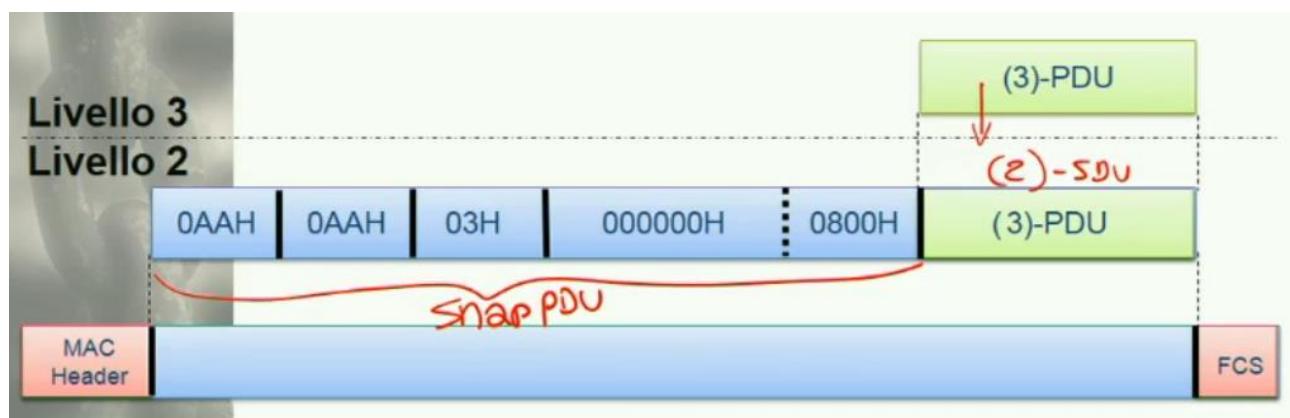


Il formato originale contiene: 16 bit (destinazione, sorgente) ma tali valori sono stati acquisiti da MAC perciò adesso questi bit hanno un altro scopo: identificare qual è il protocollo di livello superiore trasportato nel campo dati. Essendo 16 bit nel tempo diventati molto pochi, questi bit sono stati estesi, creando un'estensione nel campo intestazione: **SNAP PDU**.



Sono stati introdotti in modo da avere bit aggiuntivi per identificare i protocolli dello strato superiore (tutta la parte in blu).

In sostanza si hanno 8 byte che stanno ad indicare quei pochi protocolli che si utilizzano allo strato 3 (perché molti oggi sono andati in disuso).



La PDU del livello 3 diventa una 2 SDU, viene aggiunta la Snap PDU e poi passata al MAC.

7. Protocolli di accesso per reti locali (LAN)

LAN (Local Area Network) sono reti di piccola estensione geografica (generalmente una casa). In generale, se il mezzo trasmissivo è condiviso, può trasmettere solo **un nodo alla volta**. I protocolli dovranno risolvere questo problema.

Si utilizza un mezzo trasmissivo condiviso perché nelle reti locali il traffico non è costante ma **impulsivo**. Se si utilizzasse un canale dedicato per ciascuno di questi dispositivi sarebbe mal utilizzato, visto che molto spesso non comunicano contemporaneamente.

La trasmissione è **broadcast**, cioè tutti sentono i dati inviati ma li riceve una sola persona. Le topologie tradizionalmente utilizzate in reti locali sono: bus, anello, stella, bus monodirezionale.

Multiplazione ed accesso multiplo

Il problema principale è la condivisione di un canale. Nella multiplazione il problema è concentrato, cioè non vi sono accessi lungo il canale di trasmissione (ma solo a inizio e fine).

In questo caso di accesso multiplo il problema è **distribuito** perché i flussi **accedono al canale da punti differenti, distanti**.

Per risolvere il problema di accesso multiplo:

Condivisione “rigida” del canale

- si divide canale in piccole porzioni
- si allocano porzioni canale in modo esclusivo

Tre possibili soluzioni

- Time Division
- Frequency Division
- Code Division

Il problema è l'esclusività: se si limitano le frequenze a un utente a un certo punto si termineranno le frequenze e inoltre tale utente non sempre trasmette. Bisogna quindi cercare di trovare una soluzione comune e di condivisione. Quando la frequenza è inutilizzata, l'utente che deve trasmettere utilizzerà tale frequenza, avvisando però gli altri nodi tramite una comunicazione di allocazione ai nodi.

Per utilizzare l'accesso multiplo nelle reti locali serve un dispositivo che sia presente in rete e che gestisce chi e quando deve trasmettere. Questo dispositivo deve rispondere a una serie di problemi (deve essere centralizzato o replicato in maniera distribuita, che protocollo di accesso utilizza per comunicare chi può trasmettere?)

Gli umani usano protocolli ad accesso multiplo molto spesso:

- Moderatore che decide chi parla
- Allocazione su alzata di mano (prenotazione)
- Accesso libero
- Accesso libero, ma educato (se qualcuno parla tacco)
- Passaggio ciclico di testimone

Perciò tali modi possono essere utilizzati anche nelle reti.

I protocolli LAN tentano di simulare i comportamenti umani in modo da avere:

- Capacità e traffico smaltito (throughput)
- Equità
- Ritardo (accesso, propagazione, consegna)
- Numero di nodi (stazioni), lunghezza della rete, topologia, facilità di realizzazione, robustezza

I protocolli per reti locali sono stati storicamente centinaia, ma oggi esistono tre famiglie principali:

- **a contesa o accesso casuale (Ethernet, WiFi)**
- ad accesso ordinato (Token Ring, Token Bus, FDDI)
- a slot con prenotazione (DQDB)

Gli ultimi due sono in disuso.

Il token ring funzionava tramite un token (pacchetto) che girava sulla configurazione ad anello e permetteva al trasmettitore di trasmettere.

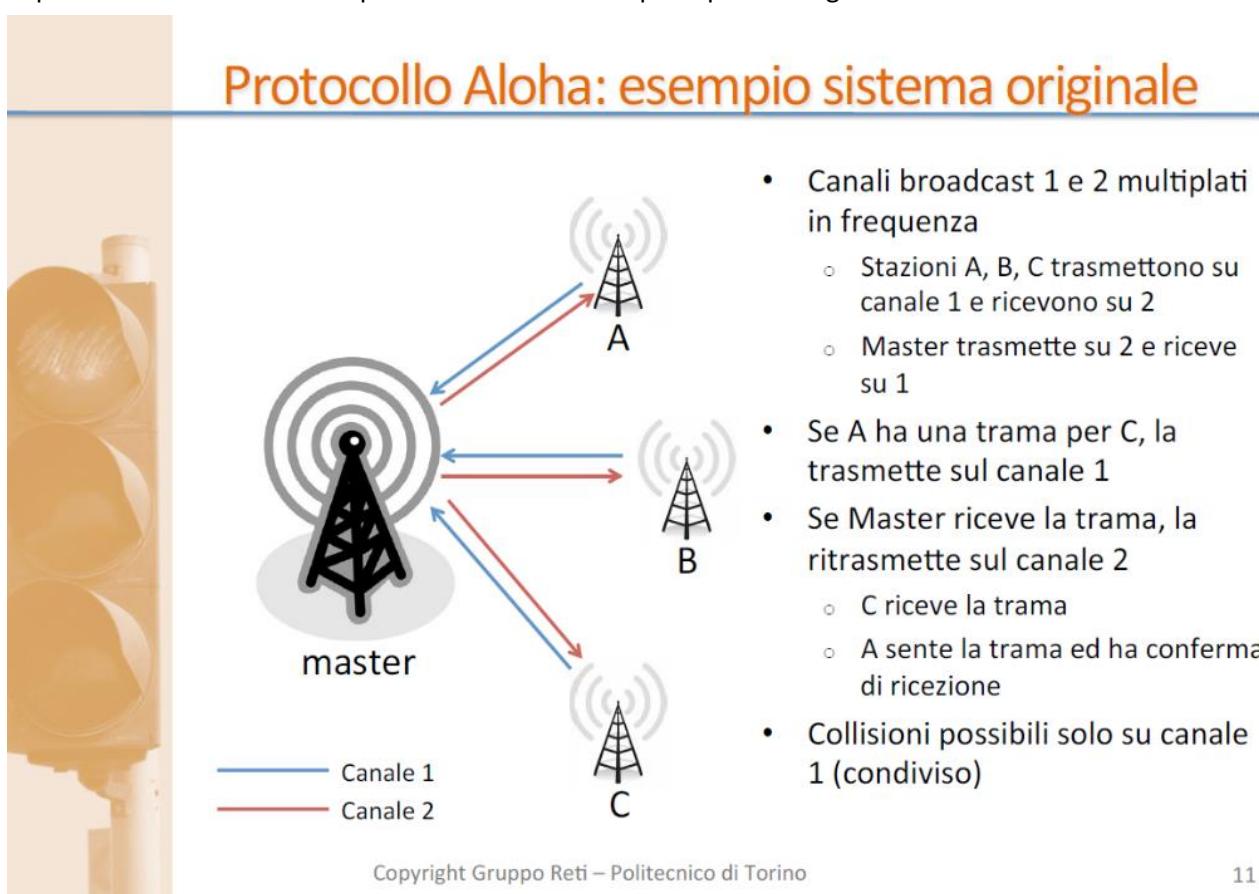
Protocolli ad accesso casuale

Quando un nodo deve trasmettere, trasmette la trama occupando tutto il canale (velocità massima che gli consente il canale), senza doversi coordinare con gli altri nodi.

Se due o più nodi trasmettono contemporaneamente si ha **collisione** (evento negativo che presuppone la perdita dei pacchetti che hanno colliso).

I protocolli MAC ad accesso casuale possono specificare come **rendere meno probabile, riconoscere e recuperare** a fronte di collisione.

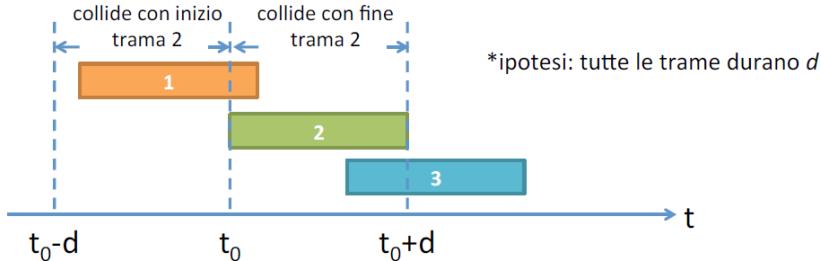
L'idea la ebbe Norm Abramson nel 1970 con il protocollo **Aloha** e l'idea era di realizzare una commutazione di pacchetto su onde radio. Il protocollo funziona col principio che segue:



Il protocollo Aloha è semplice perché non richiede sincronizzazione. La trasmissione può iniziare in qualsiasi istante e la conferma di ricezione è su un canale separato senza collisioni. Stop&Wait in caso di collisione (e mancata conferma di ricezione) ritrasmettendo la trama dopo un timeout.

- Probabilità di collisione elevata:

- trama verde collide con altre trame trasmesse in $[t_0-d, t_0+d]^*$



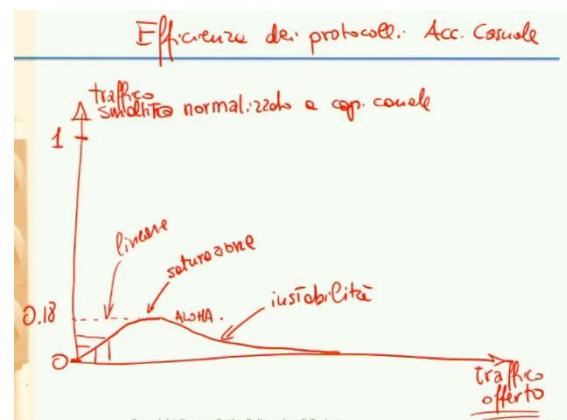
La finestra di vulnerabilità dura **2d**.

In caso di collisione, si esegue un **backoff** ovvero le stazioni che hanno colpito aspettano un tempo casuale prima di ritrasmettere la **PDU**. Il tempo casuale serve per **rompere il determinismo**. Se le due stazioni aspettassero lo stesso tempo, si avrebbe di nuovo collisione con probabilità 1. In caso di ulteriore collisione si raddoppia il **tempo massimo di attesa casuale (backoff esponenziale)**.

Efficienza dei protocolli casuali

Se si rappresenta su un grafico: su asse x il traffico offerto, su asse y il traffico smaltito normalizzato alla capacità del canale.

- Lineare: tutto il traffico offerto diventa traffico smaltito
- Saturazione: per Aloha corrisponde a **0.18**. Vuol dire che nella migliore delle ipotesi quello è il valore che raggiunge. Una volta raggiunta la saturazione il traffico continua a scendere fino a 0 in cui nessuna trasmissione avviene con successo.
- Instabilità: quasi tutti i protocolli ad accesso casuale l'hanno.

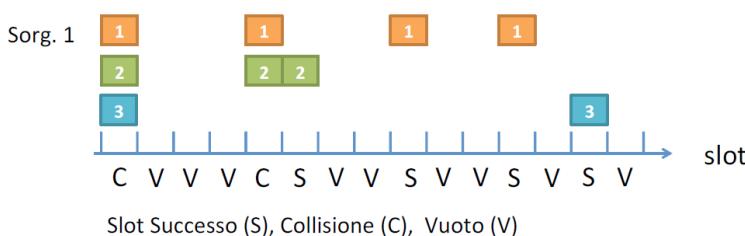


Slotted Aloha

Nello slotted Aloha si divide il tempo in istanti fissi detti **slot**. Un utente deve aspettare l'inizio di uno slot per poter iniziare a trasmettere. Viene rimossa solo apparentemente la casualità.

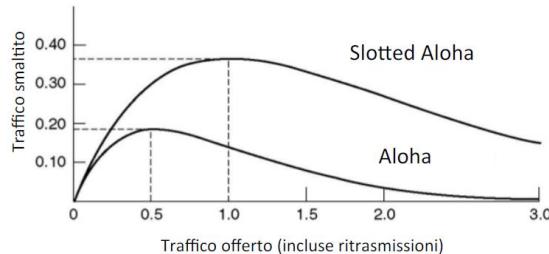
In pratica prendendo il disegno precedente si continua a collidere con slot arancione, ma non si collide più con quello blu. Lo slot deve essere lungo quanto una trama per far sì che tutto funzioni.

Il protocollo Slotted Aloha raggiunge **0.37** di efficienza.



Se c'è collisione si ritrasmette in un altro slot con probabilità **p** (equivalente a backoff) fino al successo. Es. trasmetto solo se ottengo 6 lanciando un dado. Dove c'è V nessuno ha ottenuto 6.

La caratteristica di questi protocolli è che sono **semplici**. Il throughput è però limitato a valori bassi a causa delle collisioni. Il protocollo Aloha è **instabile!** Ma è a basso carico, poiché il ritardo di accesso è nullo o contenuto. I ritardi di accesso però non sono controllabili a priori in modo deterministico. Cioè non è possibile assicurare quando si effettuerà la trasmissione perché non si può sapere gli altri cosa stanno facendo.



Il traffico è normalizzato rispetto alla durata della trama

Può essere domanda d'esame chiedere di descrivere il grafico e le tre fasi citate prima.

CSMA (Carrier Sense Multiple Access) accesso multiplo a rilevazione di portante

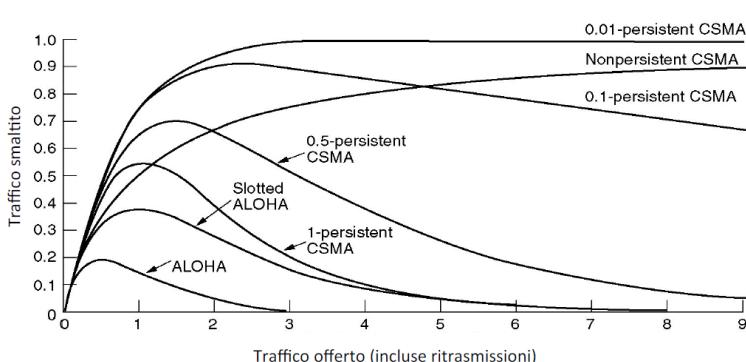
Il protocollo Aloha ha bassa efficienza perché i nodi trasmettono senza coordinarsi tra loro. Per poter aumentare il throughput e diminuire le collisioni, è possibile **ascoltare il canale (carrier sense)** prima di trasmettere. Se si sente che il canale è libero si trasmette, altrimenti la trasmissione viene ritardata. Si utilizza la portante perché è la più semplice da sentire sul canale senza dover per forza demodulare.

In caso di canale occupato, sono possibili le seguenti strategie di ritardo trasmissione:

- **CSMA 1-persistent**: aspetto che si liberi il canale, poi trasmetto immediatamente
- **CSMA non-persistent**: riprovo a sentire il canale dopo tempo casuale, se libero trasmetto
- **CSMA p-persistent**: aspetto che si liberi il canale e trasmetto con probabilità p o rimando la trasmissione con probabilità $(1-p)$.

Non persistente fa riferimento a **prima** la collisione. Se il canale è occupato si ASPETTA un tempo casuale per trasmettere.

Le collisioni nei protocolli ad accesso casuali sono **inevitabili**. La collisione è uno spreco di tempo di trasmissione della trama. La distanza gioca un ruolo fondamentale nella probabilità di connessione e con trame di grandi dimensioni (nel tempo) a parità di traffico trasmesso, riduco il numero di contese e quindi di collisioni. Maggiore è la distanza, maggiore è la finestra di vulnerabilità. Se riesco a far sentire la trasmissione di un nodo a tutti gli altri, riesco a trasmettere senza collisioni e **conviene usare trame di grandi dimensioni**.



A basso carico il CSMA non persistente, forzandosi ad aspettare la trasmissione, rimanda l'occasione di trasmettere.

Il traffico è normalizzato rispetto alla durata della trama

Varianti del CSMA

La variante del CSMA 1-persistente con rilevazione delle collisioni (**CSMA-CD, collision detection**) adatta per mezzi cablati, è detta **ethernet**.

La variante CSMA con prevenzione delle collisioni (**CSMA-CA, collision avoidance**) adatta per canali radio è detta 802.11 (Wi-Fi).

CSMA/CD (Collision Detection)

La stazione che trasmette monitora il canale durante la trasmissione: se sente solo la propria trasmissione, prosegue; se sente altre trasmissioni contemporanee (collisione), interrompe la propria.

In questo caso non occorre la conferma di ricezione se la trasmissione si è conclusa senza collisione, purché la durata minima della trama sia **superiore al doppio del tempo massimo di propagazione nella rete**.

Questo perché è necessario dare alla stazione più lontana il tempo di sentire la trasmissione (e di farla tornare indietro).

Non è possibile usarlo su topologie ad Anello, ma va bene su tutte le altre.

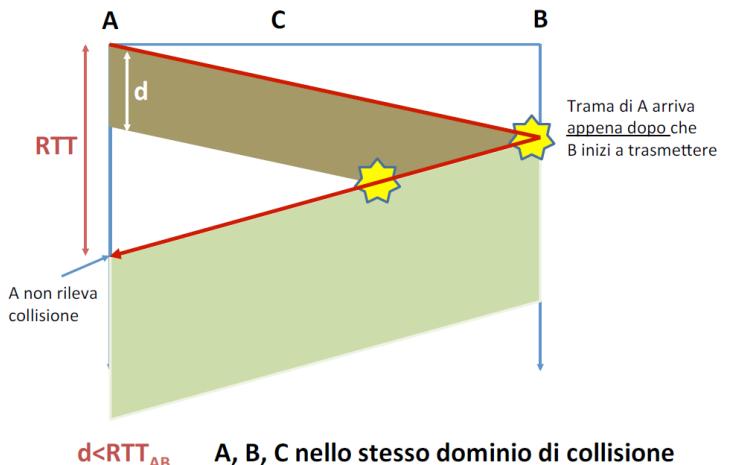
Dominio di collisione

Il dominio di collisione è la porzione di rete in cui, se due stazioni CSMA trasmettono simultaneamente o quasi, le due trame collidono. È importante la relazione tra:

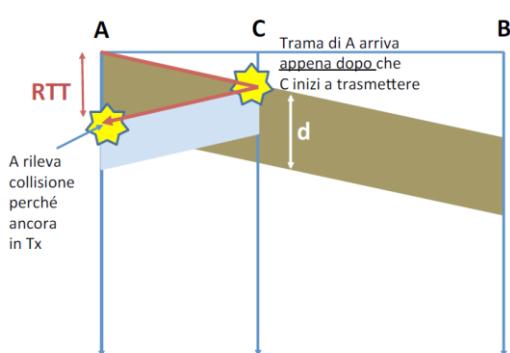
- Dimensione minima delle trame
- Distanza tra le stazioni nello stesso dominio di collisione

Se due stazioni sono troppo distanti, la trama raggiunge una delle due quando l'altra ha già terminato di trasmettere

- La collisione NON è rilevata!



Si verifica solo se la durata della trama (d) è minore del RTT che corrisponde a $2t_p$, infatti:



I vantaggi del CSMA/CD su CSMA sono i seguenti: se mi accorgo delle collisioni **sospendo** la trasmissione della trama; **riduco lo spreco dovuto ad una trasmissione inutile**.

La **collision detection** è facile nelle LAN cablate: misuro la potenza del segnale, confronto il segnale ricevuto e trasmesso.

Questo metodo non è però possibile in LAN wireless per modalità half duplex (quando trasmetto, il ricevitore è disattivato).

In generale con il CSMA/CD si hanno prestazioni migliori su **reti piccole** perché **riduco** il periodo di vulnerabilità (pari al ritardo di propagazione sul canale). Su reti piccole rispetto alla dimensione della trama (parametro 'a' piccolo, che indica $a = \frac{t_p}{t_{TX}}$): collisione rilevata prima, riduco lo spreco. Il parametro è piccolo se t_p è piccolo oppure se T_{TX} è grande (quindi se trasmetto pacchetti grandi).

Si preferisce 1 persistente perché è migliore a basso carico visto che si ha un ritardo di accesso inferiore ed il costo di collisione è piccolo su reti piccole (perché mi accorgo subito della collisione, fermo e lascio il canale libero). Si ha comunque instabilità, che diventa predominante se continuiamo a trasmettere appena il canale si libera. È inoltre difficile separare il traffico a **diversa priorità**.

CSMA/CA

Non potendo utilizzare il CSMA/CD su mezzi radio, l'obiettivo è di utilizzare un protocollo **conservativo** che previene le collisioni. Le stazioni utilizzano una variante del protocollo CSMA **non persistente**.

Il ricevitore **deve inviare ACK di ricezione** e le stazioni non possono trasmettere e ricevere contemporaneamente (collisione non rilevabile).

Una stazione che deve trasmettere ascolta il canale:

- Il canale viene ascoltato per un intervallo di tempo (che nello standard è chiamato DIFS) molto breve. Se il canale dopo questo tempo è rimasto libero, si inizia a trasmettere.
- Se il canale è occupato o diventa occupato durante questo intervallo, la stazione interrompe la trasmissione e determina un tempo di backoff (casuale).
- Una stazione decrementa il backoff **solo mentre il canale rimane libero**. Se il canale è occupato il tempo di backoff non scorre. (Es. se si trasmette e il canale è occupato, viene estratto il tempo di backoff che partirà solo dopo che la trasmissione è finita e il canale si è liberato).
- Quando il backoff arriva a 0 viene ripetuta la procedura di trasmissione.

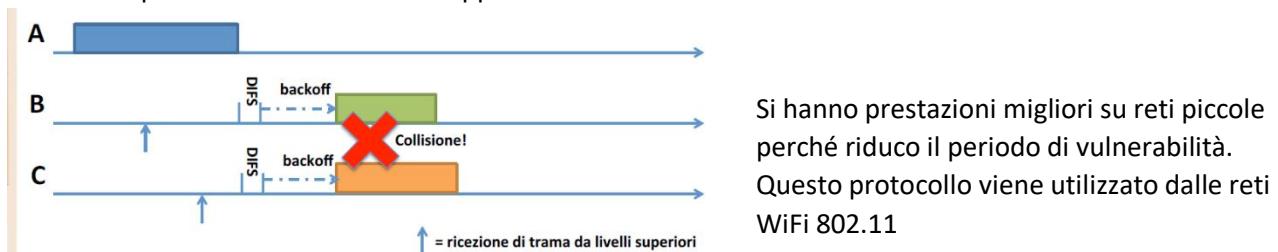
Il ricevitore verifica la correttezza della trama. Se è corretta, si risponde con una trama di ACK dopo un tempo SIFS (Short Interframe Space)<DIFS, perché la trama di ACK **ha la priorità** su ogni altra trama.

Se il trasmettitore non riceve ACK, dopo un timeout:

- Estraе tempo di backoff e inizia a decrementarlo
- Quando backoff è 0, riprova la procedura di trasmissione.

(se il trasmettitore non ricevesse ACK potrebbe aver colliso).

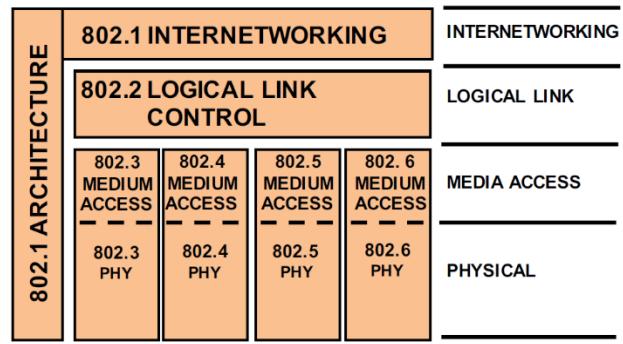
Le collisioni si possono comunque verificare in caso di contemporaneo inizio di trasmissioni. Le stazioni coinvolte ripetono trasmissione raddoppiando massimo backoff.



8. Standard per reti locali

- Ethernet (CSMA-CD 1-p)
- 802.11, WiFi (CSMA-CA non-p)

Questi standard sono standard che hanno come promotore **IEEE** che standardizza tutti i protocolli locali. Gli standard 802 sono quelli dedicati alle reti locali. I differenti standard si differenziano per la sigla che segue “802”.



- 802.1: Introduzione all’Internetworking di LAN
- 802.2: sottolivello LLC
- **802.3: CSMA/CD (Ethernet)**
- 802.4: Token Bus
- 802.5: Token Ring
- 802.6: DQDB (per reti MAN)

Etc..

Il livello 2 è diviso in due sottolivelli:

- LLC: Logical Link Control (802.2)
- MAC: Medium Access Control (802.3 Ethernet o 802.11 WiFi)

I protocolli fatti finora regolano l’accesso al canale, da qui “MAC”.

- La delimitazione della trama è fatta dallo strato MAC
- La multiplazione nel caso 802.2 viene fatta da LLC, oppure direttamente dal protocollo Ethernet (e quindi MAC).
- La rilevazione dell’errore è dello strato MAC
- L’indirizzamento nel sottostrato MAC serve per identificare la scheda, mentre nel sottostrato LLC per la multiplazione.

Indirizzi MAC

Gli indirizzi MAC che servono a identificare la scheda sono di dimensione tipica di **6 byte**. Inizialmente erano scritti in una ROM della scheda dal costruttore, mentre adesso sono configurabili via software. L’indirizzo MAC è composto da due parti:

- 3 bytes più significativi: OUI (Organization Unique Id) e sono 3 byte che può utilizzare soltanto il produttore di quella scheda.
- 3 bytes meno significativi: numerazione progressiva interna decisa dal costruttore.

Esempio: **EC-22-80-07-9A-4D** è una scheda DLink. Gli indirizzi MAC possono essere:

- **Singoli o unicast**: cioè destinati a una sola stazione
- **Multicast**: se riferiti a gruppi di stazioni
- **Broadcast**: (FF FF FF FF FF FF) se riferiti a tutte le stazioni.

Esistono due modalità di multicast:

- **Solicitation**: richiesta di servizio ad un gruppo multicast
- **Advertisement**: periodica diffusione di informazioni di appartenenza ad un gruppo multicast

In generale una scheda MAC quando riceve un pacchetto per prima cosa controlla se è corretto e successivamente:

- Se l'indirizzo MAC destinazione coincide con quello di stazione lo accetta
- Se l'indirizzo MAC è broadcast lo accetta
- Se l'indirizzo MAC ha destinazione multicast, lo accetta se il gruppo multicast è stato abilitato

Le schede possono essere configurate in maniera **promiscua**, cioè possono inviare ai livelli superiori tutte le trame, indipendentemente dall'indirizzo MAC.

ETHERNET e IEEE 802.3 (Strato MAC)

I due protocolli differiscono per alcuni dettagli riguardo la lunghezza delle trame, ma sono fondamentalmente la stessa cosa.

Il protocollo Ethernet è di tipo CSMA-CD 1-persistente, cioè si trasmette solo se il canale è libero. Se durante la trasmissione si rileva la presenza di un'altra trasmissione, questa viene fermata e si aspetta il termine dell'altra. Quando la trasmissione viene interrotta, viene inviata una **sequenza di jamming**, che è una sorta di rumore che avvisa gli altri che la trama che precede questa sequenza deve essere ignorata perché non valido.

Le stazioni che collidono attendono un tempo casuale prima di riprovare (Backoff) e non si ha conferma dell'invio.

La trama ha una parte iniziale che si chiama **preambolo**, che non è altro che una sequenza alternata di bit (1 e 0), in modo da far allineare il ricevitore col trasmettitore sui tempi di simbolo. Il preambolo termina con lo start of frame delimiter, che indica la fine del preambolo.

Successivamente sono presenti i 6 byte dell'indirizzo MAC di destinazione e successivamente quello di sorgente. Viene messo prima il MAC destinazione così se non corrisponde al ricevitore che sta sentendo può scartare il pacchetto.

Nel formato Ethernet i 2 byte successivi stanno ad individuare il protocollo di livello superiore che ha generato i dati trasportati nel campo dati.

Nello standard IEEE 802.3 tale campo non serve perché è già presente lo SNAP-PDU nel campo dati. Quel campo viene solo utilizzato per indicare quanto è lungo il campo dati.

La lunghezza è un parametro importante perché Ethernet ha una lunghezza minima per il campo dati. Tale necessità obbliga a formare al livello MAC una trama anche se il protocollo superiore ha passato pochi bit da inviare. La trama deve avere lunghezza minima di 64 byte (escludendo il preambolo e SFD). Se i byte non sono almeno 46, viene aggiunto un campo di **padding**, dati fittizi che ingrossano la trama.

I parametri di progetto della rete ethernet:

fino agli anni 90 valeva la condizione che “*se stazioni sono nello stesso Dominio di Collisione, il tempo minimo di trasmissione di una trama non può essere inferiore al massimo Round Trip Time*”. Da qui in poi è cambiato il modo in cui viene utilizzato Ethernet. Questo perché il protocollo venne creato in un mezzo

	BYTE
Preambolo = 101010.....	7
SFD = 10101011	1
Indirizzo MAC Destinazione	6
Indirizzo MAC Sorgente	6
Tipo protocollo livello superiore > 1500	2
D A T I	46 - 1500
FCS	4
Inter Packet GAP (silenzio)	Equivale a 12

	BYTE
Preambolo = 101010...	7
SFD = 10101011	1
Indirizzo MAC Destinazione	6
Indirizzo MAC Sorgente	6
Lunghezza (<1500)	2
D A T I	0 - 1500
Padding	0-46
FCS	4
Inter Packet GAP (silenzio)	Equivale a 12

broadcast (cavo coassiale) mentre ora va sul doppino che non è di tipo broadcast. Grazie a ciò il vincolo è stato attenuato, fino a far venir meno il concetto di collisione.

Su ethernet oggi non vi è più collisione.

Caratteristiche MAC Ethernet

- Protocollo con buone prestazioni se non si carica troppo la rete. (dopo gli anni 90 si è attenuato il vincolo)
- Protocollo semplice e distribuito
- Non avendo un ritardo massimo non è adatto ad applicazioni real-time (ora non c'è più il vincolo)
- Ritardi di accesso piccoli a basso carico
- Standard per LAN più diffuso quindi ampia disponibilità di componenti di basso costo
- Non esistono conferme di avvenuta ricezione
- Non gestisce priorità

ETHERNET e IEEE 802.3 (Strato fisico)

In origine prevedeva velocità di trasmissione di 10Mb/s con codifica **Manchester** con Clock che va al doppio della velocità. Inizialmente i mezzi erano il **cavo coassiale** e dal 1995 in poi sono entrati i **doppini e fibre ottiche**.

Oggi si utilizzano soltanto cavi **UTP e fibre ottiche** e si utilizzano solo **topologie a stella** in cui la contesa è eliminata se **la stella è attiva** (Switch) e la velocità raggiunge i 100 Gb/s con i nuovi standard.

Le reti ethernet oggi

Oggi, grazie al **cablaggio strutturato** è possibile collegare più topologie a stella, collegando tramite una topologia a stella i vari switch (centri della stella).

Il centro stella può essere passivo utilizzando una banda condivisa e viene detto **hub** (non si utilizza, ha ancora collisioni perché il centro stella passivo riampilifica le trasmissioni trasmettendoli a tutti i rami)

Il centro stella attivo viene detto **switch**, ricevendo la trasmissione e indirizzandola al ramo di destinazione soltanto.

Si utilizzano le reti locali interconnesse per poter aumentare l'estensione geografica della rete, aumentando il numero di utenti collegabili, senza intaccare la sicurezza. Tutto ciò fatto senza modificare protocolli.

Hub (Bridge) - Banda condivisa

Apparato multiporta per l'interconnessione di LAN Ethernet che opera al livello 1. Sfrutta il modello centro-stella passivo, rigenerando su tutte le porte di uscita la codifica di linea (o la modulazione) ricevuta su una porta di entrata.

Non riconosce le trame, poiché opera solo sui segnali (e quindi sui bit) e non separa i domini di collisione.

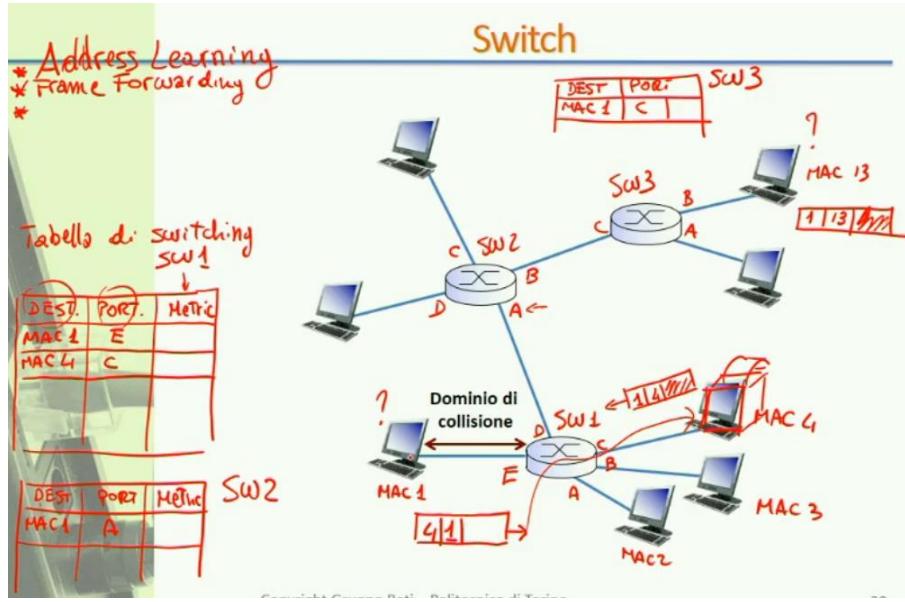
Switch – Banda dedicata

Lo switch è un apparato multiporta per l'interconnessione di LAN Ethernet che opera al livello 2. Esso è un nodo **store-and-forward** (riconosce le trame) ed utilizza il modello **centro-stella attivo** ritrasmettendo in modo selettivo su una o più porte di uscita le trame ricevute su una porta di entrata, secondo le regole del protocollo MAC. È possibile avere perdita di pacchetti per overflow delle memorie.

Lo **switch elimina le collisioni**, e con esse la necessità del CSMA/CD, l'Ethernet diventa quindi un **protocollo di framing di livello 2**.

Instrandamento mediante switch

Un insieme di segmenti di LAN interconnessi mediante bridge è detto anche LAN estesa. La tecnica di switching più diffusa prevede che le stazioni non modifichino il loro comportamento a causa della presenza degli switch (trasparenza). È necessario dunque che ogni apparato abbia un indirizzo di livello 2 unico all'interno della LAN estesa. Ogni switch ha un suo indirizzo (switch_ID) e un identificativo per ogni porta (port_ID).



Quando lo switch viene configurato, possiede la tabella di switching vuota, che riempie man mano che i pacchetti vengono inviati. Se un host con MAC 1 fa una richiesta, lo switch si segna a quale porta corrisponde il MAC 1. Quando il destinatario risponde, farà lo stesso.

Se sono connessi altri Switch in cascata, lo switch farà altrettanto, segnando chiunque faccia richieste, anche se non sono destinate agli host di quello switch.

Tale procedura è detta **frame forwarding**.

Nel caso in cui fossero presenti anelli (congiungendo, ad esempio sw3 con sw1), si possono creare ambiguità sulla strada da far fare ai pacchetti.

In questo caso interviene l'algoritmo di **spanning tree**, che viene avviato all'accensione di uno switch e viene riconosciuto solo dagli switch. Questo pacchetto serve agli switch per riconoscere la loro presenza nella rete. A quel punto parte lo spanning tree, che ha il compito di costruire un grafo, cioè topologia ad albero mettendosi d'accordo su quali porte disabilitare per evitare di creare anelli sulla topologia. Quindi in caso di anelli alcuni rami vengono disabilitati.

Le procedure di **Address Learning**, **Frame Forwarding** e **Spanning Tree** prendono il nome di **Transparent Switching**, poiché l'utente non si deve preoccupare di fare tali configurazioni.

Vantaggi dell'interconnessione di LAN

Si partiziona la rete in k reti locali “indipendenti” con un aumento potenziale della banda aggregata di rete, con perdita dei vantaggi nel caso di traffico broadcast e multicast. La probabilità di collisione con protocolli a contesa si riduce al crescere del numero di hub e si elimina con l'uso di switch.

La rete locale si trasforma in una rete classica a commutazione di pacchetto (multiplazione).

E' possibile anche migliorare la gestibilità e sicurezza con l'uso di questo protocollo.

VLAN – Lan virtuali

Sono LAN costituite da host fisicamente collegati allo stesso segmento di rete, ma logicamente partizionati in LAN separate.

In questo modo si separano i domini di broadcast e vengono raggruppati gli utenti che comunicano spesso. È però un costrutto di livello 2 che deve essere supportato dallo switch.

IEEE 802.11 – WiFi

Il protocollo 802.11 segue i principi del CSMA/CA ed ha una caratteristica: aggiunge al CSMA/CA alcuni dettagli, definendo la **distributed coordination function (DCF)**.

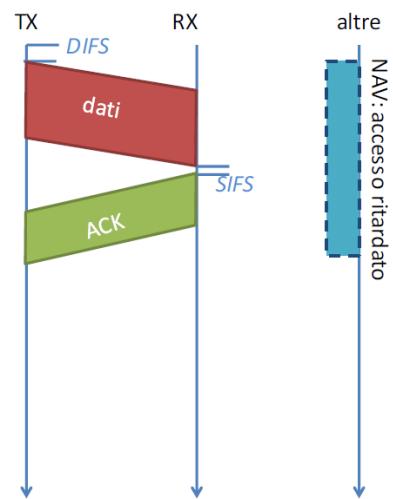
Per ogni pacchetto da inviare la stazione deve fare una **contesa** per il possesso del canale.

Le stazioni sono **half-duplex**: o trasmettono o ricevono. Il ricevitore **deve** confermare i pacchetti ricevuti con un ACK.

DCF: Principi

Il trasmettitore:

- Se il canale è libero per DIFS trasmette una trama
- Se il canale è occupato aspetta per un tempo casuale e trasmette. Se fallisce (quindi nessun ACK) si utilizza un backoff esponenziale (CA).



Il ricevitore:

- Se i dati vengono ricevuti correttamente manda un ACK dopo SIFS.

Le altre stazioni:

- Non compiono alcuna azione per un tempo pari alla durata dello scambio di trame (NAV). Il DCF prevede che le altre stazioni, quando sentono la trasmissione di un'altra scheda, leggono nell'intestazione della scheda alcuni campi che permettono al ricevitore di sapere chi è il mittente e quanto durerà la trama. Tutte le stazioni leggono tale valore, aggiungono il tempo di SIFS (standard), aggiungono la durata della trama ACK (Standard) e aspettano.

Un'altra particolarità del DCF è quella del **terminale nascosto**.

Questo fenomeno si verifica quando due o più stazioni sono a portata radio dell'Access Point, ma non sono a portata radio reciproca (cioè non si sentono).



Se A trasmette una trama all'AP,

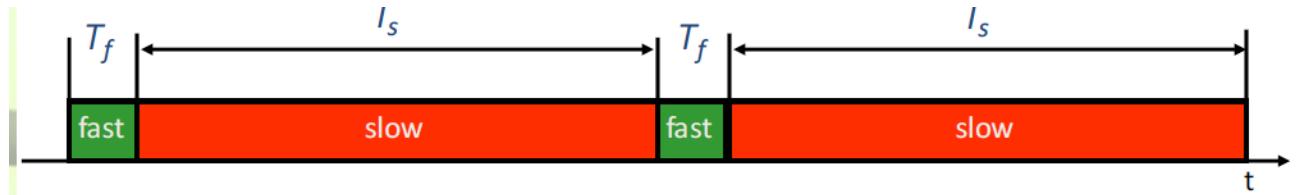
B non sente A e pensa che il canale libero perciò trasmette a sua volta andando a finire in una **collisione**.

Una possibile soluzione è il **DCF con handshaking**.

Quando A deve mandare una trama, prima di inviarla invia una microtrama detta **RTS** con tre informazioni fondamentali: a chi è destinata, da chi arriva e quanto dura l'invio. L'AP riceve il biglietto, toglie un indirizzo, ritrasmette a tutti quanti questa microtrama avvisando che A sta per inviare una trama che durerà per degli istanti. Questa microtrama detta **CTS** viene recapitato a B e lo informa che il canale sta per essere inviato. In questo modo B ricevendo CTS imposta il NAV e dà per scontato che il canale sia occupato anche se non lo sente come tale.

Anomalia nelle velocità trasmissione

Le stazioni connesse allo stesso Access Point hanno velocità diverse. Per tali motivi il throughput di tutti si uniforma alla stazione più lenta. Per esempio, se si suppone di avere stazioni a 1 Mb/s e stazioni a 11 Mb/s che inviano trame di L bit, si avrà:



Calcolo throughput approssimato:

- Slow: $(L/1)/(L/11 + L/1) * 1 \text{ Mb/s} = 0.91 \text{ Mb/s}$
- Fast: $(L/11)/(L/11 + L/1) * 11 \text{ Mb/s} = 0.91 \text{ Mb/s}$

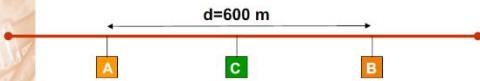
Stesso throughput... basso!

La maggior parte del tempo del canale viene occupata dalla stazione più lenta.

9. Esercizi

1. Reti locali

- Si consideri una rete composta da un mezzo fisico broadcast a 10 Mb/s su cui sono attestati tre host CSMA:



- A e B sono separate da $d=600\text{ m}$ di cavo, mentre C è equidistante da A e B
- Si assuma la propagazione nel mezzo pari a $2/3 c$ (velocità della luce nel vuoto)

$$T_{TX,A} = \frac{64 \cdot 8}{10^7} = 51,2\text{ }\mu\text{s}$$

$$t_{P,AB} = \frac{s}{v_p} = \frac{600}{2 \cdot 10^8} = 3\text{ }\mu\text{s}$$

All'istante $t=0$ A ha già verificato che il canale è libero e inizia a trasmettere.

All'istante t_1 B deve verificare se il canale è libero.

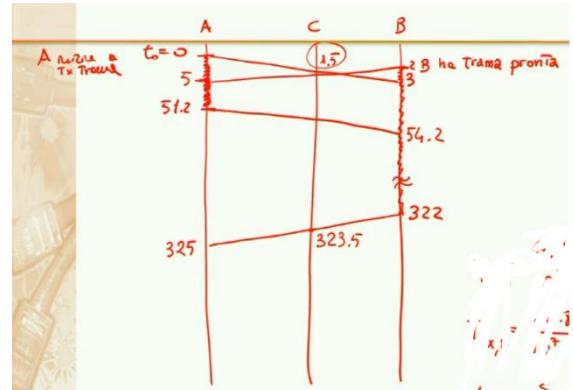
Quando B ha la trama pronta, il canale è libero (perché ancora la trasmissione non si è propagata su B) e la trasmissione raggiungerà A all'istante 5 ($2+3$). Perciò B inizierà la trasmissione e ci sarà collisione. La trasmissione di B andrà avanti fino a $322\text{ }\mu\text{s}$ e C sentirà B fino a $323,5\text{ }\mu\text{s}$, ovvero per $(323,5 - 1,5) = 322\text{ }\mu\text{s}$. Anche se B avesse iniziato la trasmissione all'istante 3, sarebbe partita.

1. Reti locali

- L'host A inizia a trasmettere una trama di dimensioni pari a 64B all'istante t_0 , mentre B ha una trama di 400B pronta per la trasmissione all'istante $t_1 = t_0 + 2\text{ }\mu\text{s}$

- Determinare:
 - B inizia subito la trasmissione della trama?
 - in caso affermativo, si verifica una collisione?
 - per quale intervallo di tempo la stazione C sente il canale occupato?

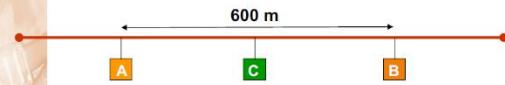
$$T_{TX,B} = \frac{400 \cdot 8}{10^7} = 320\text{ }\mu\text{s}$$



Esercizio 2

2. Reti locali

- Si consideri una rete composta da un mezzo fisico broadcast a 10 Mb/s su cui sono attestati tre host CSMA 1-persistenti:



- A e B sono separate da un cavo di lunghezza $d=600\text{ m}$, mentre C è equidistante da A e B
- Si assume la propagazione nel mezzo pari a $2/3\text{ c}$ (velocità della luce nel vuoto)
- Si assume che un host attenda che il canale resti libero per 5 μs prima di iniziare la propria trasmissione

2. Reti locali

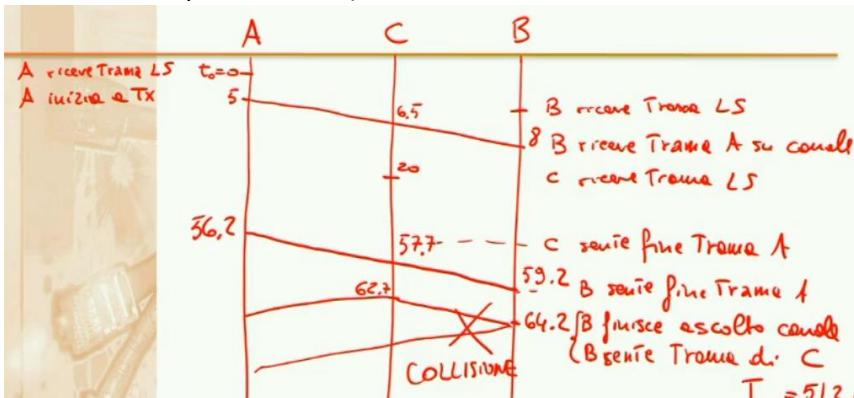
- Gli host inviano trame da 64B e hanno la seguente attività:
 - A ha una trama pronta all'istante t_0
 - B ha una trama pronta all'istante $t_1=t_0+5\text{ }\mu\text{s}$
 - C ha una trama pronta all'istante $t_2=t_0+20\text{ }\mu\text{s}$
- Determinare:
 - Ci sono collisioni?
 - In caso affermativo, quali host collidono?

Tutti gli host devono fare la procedura di sensing.

$$T_{TX} = \frac{64 \cdot 8}{10^7} = 51,2\text{ }\mu\text{s}$$

$$t_{P,AB} = \frac{s}{v_p} = \frac{600}{2 \cdot 10^8} = 3\text{ }\mu\text{s}$$

All'istante $t_0=0$ A riceve la trama dal livello superiore e deve attendere **5 μs** . A questo istante, A trasmette e si propaga fino a B all'istante $8\text{ }\mu\text{s}$ e su C a $4,5\text{ }\mu\text{s}$. B riceve una trama dal LS ma non trasmette perché attendendo i $5\text{ }\mu\text{s}$ sente A e quindi non trasmette. Idem C.



Il primo a sentire il canale libero è C, che aspetta e vedendo il canale libero trasmette.

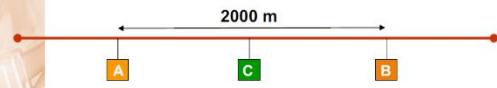
Successivamente sarà B che contemporaneamente riceverà la trama e finirà l'ascolto del canale.

In questo caso vince la trasmissione e B inizia a trasmettere, effettuando collisione. A collidere saranno C e B.

Esercizio 3

3 - Reti locali

- Si consideri una rete composta da un mezzo fisico broadcast a 10 Mb/s su cui sono attestati tre host CSMA/CD:



- A e B sono separate da 2000 m di cavo, mentre C è equidistante da A e B
- Si assume la propagazione nel mezzo pari a $2/3 c$ (velocità della luce nel vuoto)
- Si assume che un host attenda che il canale resti libero per 5 μs prima di iniziare la propria trasmissione

3 - Reti locali

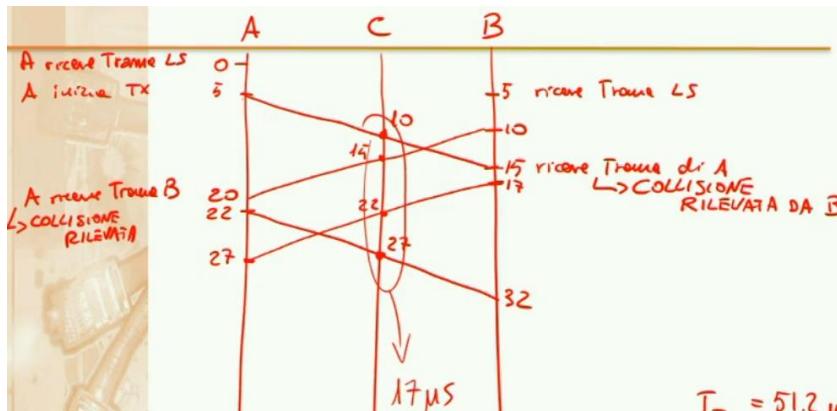
- La stazione A ha una trama di dimensioni pari a 64B pronta per la trasmissione all'istante t_0 , mentre B ha una trama di 400B pronta per la trasmissione all'istante $t_1 = t_0 + 5 \mu s$
- In caso di collisione, questa viene rilevata da una stazione in un tempo di 2 μs
- Determinare:
 - B inizia la trasmissione della trama?
 - in caso affermativo, si verifica una collisione e questa viene rilevata?
 - per quale intervallo di tempo la stazione C sente il canale occupato?

Finiscono di trasmettere se rilevano una connessione (CD).

$$T_{TX,A} = \frac{64 \cdot 8}{10^7} = 51,2 \mu s$$

$$T_{TX,B} = \frac{400 \cdot 8}{10^7} = 320 \mu s$$

$$t_{P,AB} = \frac{s}{v_p} = \frac{2000}{2 \cdot 10^8} = 10 \mu s$$



A inizia le procedure al tempo $t=0$. Come prima, fino al tempo 5 non succede nulla, perciò A inizia la trasmissione. B al tempo 5 riceve la sua trama e deve nominalmente aspettare fino al 10 per cominciare la trasmissione.

Il tempo di propagazione da A a B è di 10, perciò verrà ricevuta da B al tempo 15. Per questo motivo, al tempo 10 B inizierà la trasmissione

e si ha collisione. La collisione viene rilevata perché la trasmissione di B dovrebbe durare 320, perciò viene rilevata da B.

A invece dovrebbe finire nominalmente a 56,2 e la trama di B arriva ad A all'istante 20 perciò anche A rileva la collisione.

B essendo il primo all'istante 17 rileva la collisione e smette di trasmettere e verrà sentito da A fino all'istante 27. A a sua volta interrompe la trama all'istante 22 e verrà ricevuta da B all'istante 32.

C sente il canale occupato per $27-10=17$.

Esercizio 4

4 - Reti locali

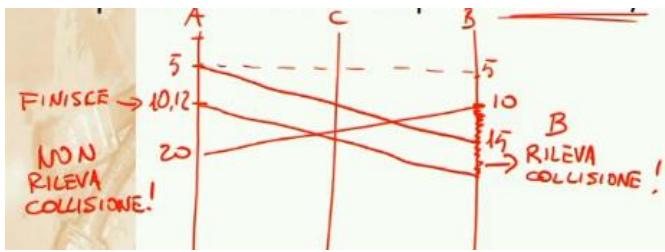
- Ripetere l' esercizio 3 assumendo che la capacità del canale sia pari a 100 Mb/s

$$T_{TX,A} = \frac{64 \cdot 8}{10^8} = 5,12 \mu s$$

$$T_{TX,B} = \frac{400 \cdot 8}{10^8} = 32 \mu s$$

$$t_{P,AB} = \frac{s}{v_p} = \frac{2000}{2 \cdot 10^8} = 10 \mu s$$

Ora la trama dura di meno perché la trama è 10 volte superiore. Il tempo di propagazione rimane lo stesso.



In questo caso c'è il pericolo che la collisione non venga rilevata perché la trama di A impiega molto meno del tempo di andata/ritorno.
La trama di A arriva su B all'istante 15 e quando all'istante 5 B inizia ad aspettare e trasmette a 10, B rileva la collisione ma A ha già terminato e non la rileva.

SECONDA PARTE

1. Il livello rete

Il livello 2 non basta per progettare una rete, questo perché sorgono diversi problemi:

- È possibile che si verifichino problemi di gestione inefficiente della ridondanza (protocollo Spanning Tree). È necessaria perché in caso si verifichino problemi su un link sarebbe possibile raggiungere gli host tramite un altro link, ma tali rete possono creare problemi di loop (che vengono risolti dallo Spanning Tree). Generare ridondanze su reti molto grandi, non è efficace/economico.
- Saturazione delle tabelle di inoltro degli switch nelle reti molto grandi. È necessario quindi riassumere le informazioni nelle tabelle di inoltro, raggruppando ad esempio i dispositivi in base al posto in cui si trovano. Per risolvere tale problema verrà utilizzato il **routing IP**, effettuando route aggregation (che non è possibile fare sugli switch perché i MAC sono costruiti in maniera diversa e non secondo le zone in cui si trovano).
- Propagazione del traffico broadcast: un pacchetto broadcast inviato a uno switch viene inviato a tutte le stazioni, ma nel caso di rete globale basata su switch, il pacchetto broadcast andrebbe a finire ovunque e ciò non è possibile perché la rete sarebbe piena di pacchetti broadcast.
Il traffico broadcast è importante, ma va confinato.

Cosa fa il livello Rete

Trasporta un segmento di livello trasporto da un nodo di partenza a un nodo destinatario

Il livello rete è disponibile sugli host e deve essere disponibile **anche** sui dispositivi intermedi che devono inoltrare il traffico (che sono i **router**).

I router esaminano gli header (PCI) dei pacchetti di livello 3 (**datagram**) per cercare di capire come inoltrare questi pacchetti.

Nel network layer vanno distinti due termini che riguardano l'operato di un router:

- Se un router legge i campi di un header di livello 3 per decidere che strada far prendere al pacchetto, sta effettuando un'operazione di **forwarding**. Per fare ciò viene letto il pacchetto e vengono cercate le informazioni nelle **tabelle di inoltro** (il router non fa altro che leggere queste informazioni)
- Il router effettua **routing** quando vengono riempite le tabelle di inoltro, perché all'avvio di un router non si conoscono le strade che possono essere intraprese. Per effettuare il routing devono essere utilizzati degli **algoritmi di routing** (come Djikstra, cammini minimi di un grafo).

Servizi connessi/non connessi

Una prima cosa da decidere per realizzare una rete globale è se realizzare una rete **connessa o non connessa**. Ovvero scegliere se collegare il servizio come nei **circuiti virtuali (VC)** oppure utilizzare un servizio **datagram** che crea un servizio non connesso.

Datagram o rete VC?

B-ISDN era una proposta alternativa al protocollo IP, che utilizzava il protocollo **ATM (VC)** e quindi venivano emulati le reti a circuito per garantire qualità ottimale. Il problema era la difficoltà nel creare tale rete perché uno switch ATM è molto più complesso rispetto allo switch Ethernet o al router IP.

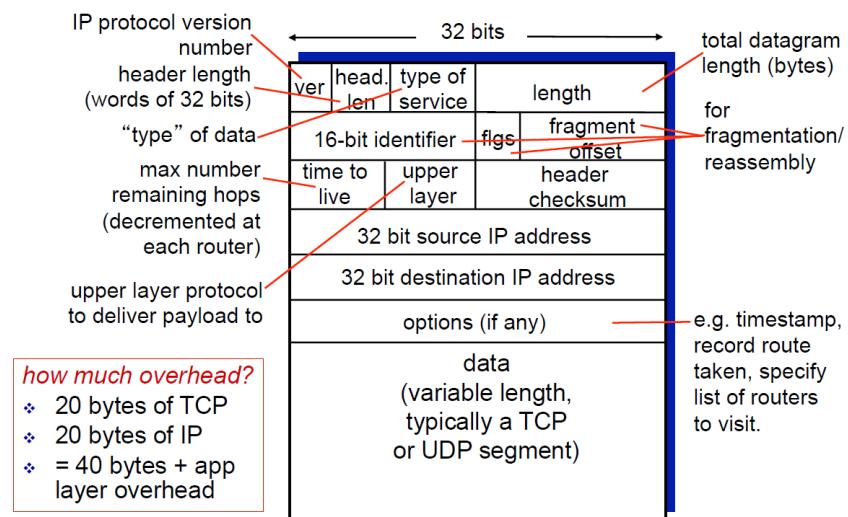
Per contro c'era l'idea di internet basata su **Internet** e quindi sul protocollo **TCP/IP**. Quando è necessario inviare un pacchetto, tale pacchetto viene inviato senza creare circuiti e ciò **non** garantisce qualità perché non c'è l'approccio basato su circuiti virtuali. Nonostante ciò vince quest'ultimo perché è più semplice ed è possibile anche sopperire a questa mancanza di affidabilità. Al livello trasporto verrà utilizzato il protocollo TCP che sarà in grado di recuperare i pacchetti persi, re-inviandoli.

Il livello rete di Internet

Il protocollo che regna questo livello è il protocollo **IP**. Tale livello è composto da: **protocolli di routing**, **routing table**, **protocollo IP** e **protocollo ICMP**.

- Le routing table servono per instradare i pacchetti
- I protocolli di routing riempiono in maniera ottimale le routing tables (RIP, OSPF, BGP)
- Protocollo IP che si occupa di definire il formato dei pacchetti e le convenzioni di addressing
- ICMP Protocol serve come supporto al protocollo IP per gestirlo meglio ed identificare eventuali errori o segnalazioni.

Formato datagram IP



I bit che identificano sorgente e destinazione nella versione **IPv4** sono 32, perciò si hanno 2^{32} combinazioni. Tali combinazioni sono in esaurimento a causa di errori all'avvento del protocollo e all'aumentare dei dispositivi IoT. 20 byte è la lunghezza base dell'IP (da ver a IP address).

- Ver: identifica la versione di IP (IPv4, IPv6)
- Header length: necessario per il campo "opzioni", che rende la lunghezza dell'header variabile
- Type of service (o DS): serve per differenziare il traffico su un dispositivo (magari se si vuole dare priorità al traffico voce rispetto a semplici file)
- Length: identifica la lunghezza **totale** del pacchetto. Serve insieme al campo 16-bit identifier, flags e fragment offset per utilizzare la **frammentazione** che oggi **non** viene più utilizzata.
Inviando un pacchetto IP di 4000 byte, nella rete ethernet il limite è inferiore (1500 byte) il pacchetto deve essere frammentato. Se un frammento di questi pacchetti viene perso, è necessario accorgersene e servono strategie per accorgersi di questi problemi. Ciò è possibile grazie a questi campi.

Oggi praticamente tutte le reti locali sono fatte con **ethernet** dove la trama massima è di **1500 byte** (**della MTU maximum transfer unit**). Nei collegamenti router sarà presente di nuovo Ethernet oppure il protocollo PPP (che non ha limiti sulla dimensione dei pacchetti). Per tali motivi, anche il protocollo IP viene ridotto a 1500 byte come dimensione massima per evitare frammentazione.

- Time to leave: numero massimo di "passaggi" del pacchetto tra i router prima della sua distruzione. Numero massimo di router che un pacchetto può attraversare. Il valore da scrivere è tipicamente 255 (numero massimo di questo campo). In generale numeri molto elevati rispetto al numero massimo di router che un pacchetto può attraversare.

- Upper layer: serve per capire qual è il protocollo di livello 4 contenuto nel payload. Anche ICMP (che non è di livello 4) viene “imbustato” in IP quindi non necessariamente dentro il datagram c’è un protocollo di livello 4.
- Header checksum: un controllo di errore sui bit. Se per caso nella trasmissione di un pacchetto c’è qualche bit che cambia valore, tale pacchetto viene scartato. Il checksum viene fatto **solo** sulla parte di header. Siccome ogni router controlla il checksum e decrementa il TTL (Time to Leave), bisogna **ricalcolare** il checksum.

Indirizzamento IP

Gli indirizzi IP sono composti da una stringa di 32 bit che viene assegnata a tutti i dispositivi connessi alla rete (sia host che router). Gli indirizzi vengono assegnati alle **interfacce**. Per esempio, se su un pc si attivano sia la scheda WiFi che Ethernet, si hanno due indirizzi IP differenti. Gli indirizzi IP vengono comunemente scritti utilizzando la notazione decimale puntata e ogni gruppo viene successivamente convertito in binario.

IP definisce delle regole per la comunicazione globale. IP è una rete logica costruita per interconnettere le varie reti fisiche (di livello 2, create con mano). L’IP non si interessa di quelle che sono le tecnologie di livello 2 utilizzate. Le varie reti di livello 2 devono essere quindi interconnesse tramite router alle cui interfacce vengono assegnati indirizzi IP.

Logical IP Subnet (LIS)

Le subnet IP sono un insieme di nodi IP che hanno la stessa **subnet part**. L’indirizzo IP è diviso in due parti: **subnet part** (bit di ordine superiore) e **host part** (ordine inferiore). Una subnet è una serie di interfacce di dispositivi con la stessa subnet part. Queste interfacce devono **fisicamente** raggiungersi le une con le altre, cioè essere fisicamente connesse con la stessa rete di livello 2.

Rete fisica: rete di livello 2. Insieme di dispositivi che possono raggiungersi fisicamente tramite un insieme di link layer.

Nello standard IP si assume una corrispondenza biunivoca tra le reti fisiche e le subnet. Su una rete fisica si può avere una e una sola subnet (e viceversa). Questo lo assume lo standard IP per poi definire le modalità di inoltro dei pacchetti. IP NON mette bit sul livello fisico, quello è compito del livello 2.

Quando un dispositivo IP deve inviare un pacchetto a un altro dispositivo IP che appartiene alla stessa subnet, quindi con la stessa prima parte di indirizzo, effettuerà una **comunicazione diretta** cioè utilizza dei meccanismi di livello 2 (link layer) per comunicare col dispositivo.

Quando invece un dispositivo IP deve inviare un pacchetto a un dispositivo che non appartiene alla stessa subnet, dovrà effettuare una **comunicazione indiretta** e quindi utilizzare un router. Chiederà quindi al router, inviandogli il traffico e chiedendogli aiuto per raggiungere la destinazione. Per fare ciò utilizzerà una comunicazione diretta col router (che appartiene alla sua subnet). La comunicazione indiretta è quindi basata su comunicazione dirette “a intervalli”.

Nuovamente, IP crea una rete logica ma **non** gestisce l’accesso al mezzo. IP si preoccupa solo di creare la rete logica sopra quella fisica per avere metodologie più scalabili per effettuare il routing. Dentro il router non dovrà scrivere la lista di tutti i dispositivi connessi alla rete globale.

Successivamente: si può avere una subnet su più di una rete fisica (ma oggi NON esistono reti del genere, si basavano sul protocollo aggiuntivo **proxy ARP** e senza questo installato la rete non funziona).

Si possono avere più subnet su una sola rete fisica. Questo funziona senza bisogno di aggiungere altri protocolli, basta aggiungere un indirizzo IP sulla stessa interfaccia del router (che avrà 2 indirizzi IP invece di 1). Essendoci due subnet nella stessa rete fisica, l’host non vede l’altro host sulla stessa subnet e quindi

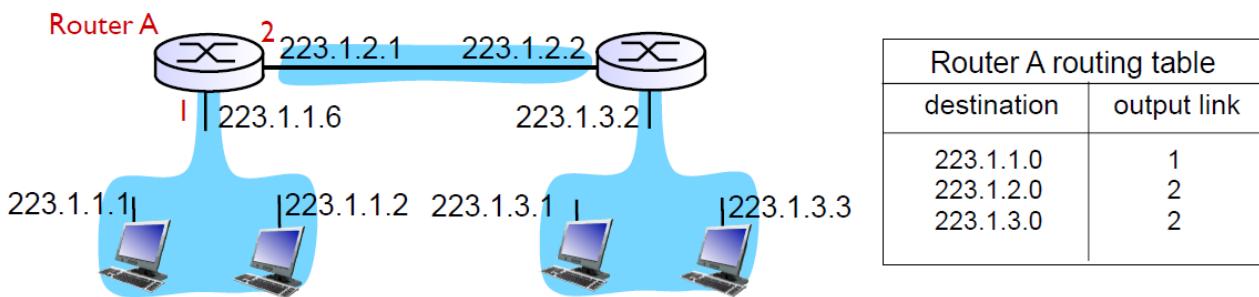
invia il pacchetto al router. Il router guarderà le tabelle di routing, capirà che l'host è raggiungibile e allora invierà il pacchetto. Questo si chiama **one-arm router**.

Indirizzi speciali

Subnet	All 0s	the (sub)network (network ID)
All	All 1s	limited broadcast (local net)
Subnet	All 1s	directed broadcast for net
127	Anything (often 1)	loopback

- Un indirizzo IP in cui si ha una certa subnet part e tutti 0 sulla host part è un indirizzo speciale che si chiama (sub)net ID (o network ID). Identifica la subnet della rete IP. Rappresento quindi tutti i nodi che appartengono alla subnet.
- Un indirizzo IP con tutti 1 rappresenta il broadcast. Se si definisce un indirizzo IP broadcast, per definizione tutta la rete IP percepisce il pacchetto, ma nell'IP questo broadcast è **confinato** alla rete locale.
- Se invece un indirizzo IP è composto da una certa subnet part e poi tutti i bit a 1 (tipo 223.1.1.255 dove 223.1.1 è la subnet part) questo è un **broadcast diretto a tutti gli host di una certa subnet**.
- Se un indirizzo IP è composto da una subnet part pari a 127 e seguito da altro (ma spesso 1) è il **loopback** (esempio 127.0.0.1) è un indirizzo che identifica il **localhost**. Se si manda il traffico a questo indirizzo non va nella rete ma resta nel dispositivo.

Gli indirizzi IP dipendono dalla rete fisica in cui un host è connesso. Se si spostano degli host dovranno avere diverso indirizzo IP (perché cambia la subnet). L'indirizzo IP ha quindi una struttura completamente diversa dall'indirizzo MAC (che è univoco e resta tale sempre perché deciso dal costruttore). L'indirizzo IP **non è portabile** (a differenza dell'indirizzo MAC che è portabile) proprio per questo motivo. In questo modo si favorisce la **scalabilità** del routing.



Se bisogna realizzare la tabella di routing di A non bisogna più scrivere tutta la lista dei dispositivi, ma si possono scrivere soltanto le reti. La rete a sinistra è rappresentata dal suo subnet ID ovvero 223.1.1.0 perciò posso scrivere che tale subnet è raggiungibile dalla porta 1.

Analogamente la rete 223.1.3.0 al router basta sapere che si raggiunge tramite il link 2. Il router A non ha quindi alcun motivo di distinguere i singoli host, ma gli basta sapere che quella rete si raggiunge tramite quel link. Ci penserà qualcun altro a consegnare all'host.

In questo modo non devo scrivere la lista di tutti i dispositivi.

Ciò però non basta perché anche scrivendo solo le subnet della rete globale, la lista è molto lunga. Si utilizzano quindi gli indirizzamenti gerarchici.

Protocollo ARP (Address Resolution Protocol)

Serve per avere corrispondenza diretta tra Indirizzo IP e indirizzo MAC (essenziale a Ethernet per spedire il pacchetto). Le **ARP Table** che contengono questi dati devono essere riempite in modo automatico (per rendere il tutto **plug & play**). Il metodo più semplice è chiedere a chi ha un determinato indirizzo IP di rispondere con l'indirizzo MAC.

L'host invia una trama broadcast detta **ARP Request**. L'host che ha quell'indirizzo IP risponderà con una **ARP Reply** in maniera unicast con il proprio indirizzo MAC.

L'ARP Table viene spesso chiamata ARP Cache. Questo perché entro un certo lasso di tempo, se la Entry non viene aggiornata, essa viene cancellata.

L'ARP si pone a metà tra il livello IP e il livello 2. Non è un protocollo di livello 3 perché le funzionalità sono ben distanti da tale livello ma allo stesso tempo non si occupa del livello 2.

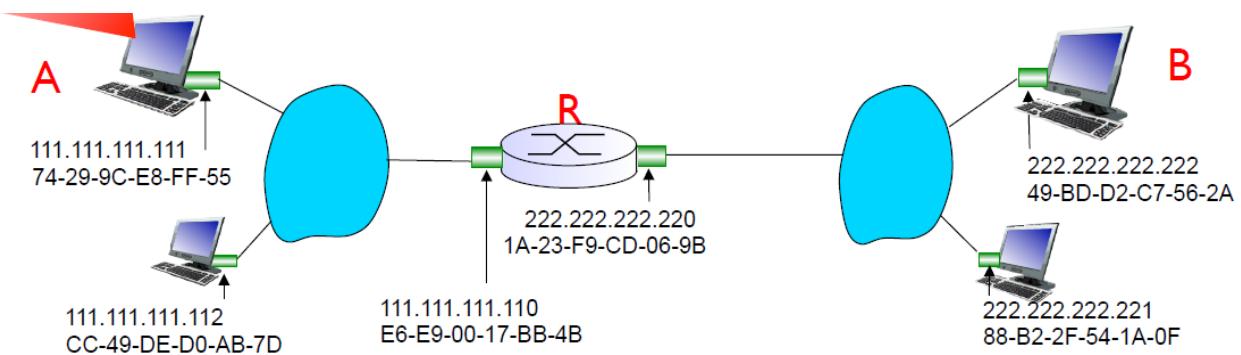
L'ARP NON viene imbustato nel datagram, ma sono il payload di una trama.

MAC broadcast	MAC A	ARP Req	MAC A	IP A*	??	IP E*
ARP Request						
MAC A	MAC E	ARP Reply	MAC E	IP E*	MAC A	IP A*
ARP Reply						

(Esempio in cui A deve scoprire l'indirizzo MAC di E).

Addressing: routing verso un'altra LAN

Supponiamo che si voglia inviare un datagram da A a B attraverso R, come in figura.



A sinistra la subnet 111.111.111.0 e a destra 222.222.222.0. In questo esempio si assume che:

- A conosca l'indirizzo IP di B. Questa assunzione non è così forte come sembra, grazie al protocollo **DNS**. Il DNS è un protocollo ma anche un sistema (per via di server collegati tra loro interrogati tramite un protocollo) che serve per convertire un nome in un Indirizzo IP. Tutte le volte che si utilizza internet per qualsiasi scopo, es (www.facebook.com) si sta scrivendo l'indirizzo IP di tale sito web. E' inoltre possibile risalire all'indirizzo IP anche senza conoscere il nome, grazie ai **motori di ricerca**.
- Supponiamo anche che A conosca il **default gateway** di R, cioè l'indirizzo IP dell'interfaccia di un router che permette a un host di uscire dalla propria rete. Questa assunzione è più forte della precedente, perché A deve essere in grado di riconoscere un indirizzo IP default gateway. Anche in questo caso è possibile dire che queste assunzioni non sono così forti grazie al protocollo **DHCP**. Un

server DHCP darà ad A un indirizzo IP e darà anche il default gateway della rete. Il server DHCP starà dentro l'AP (router, etc..).

- Assumiamo che A conosca l'indirizzo MAC del suo default gateway (R). Anche questa assunzione non è forte perché esiste il protocollo ARP.

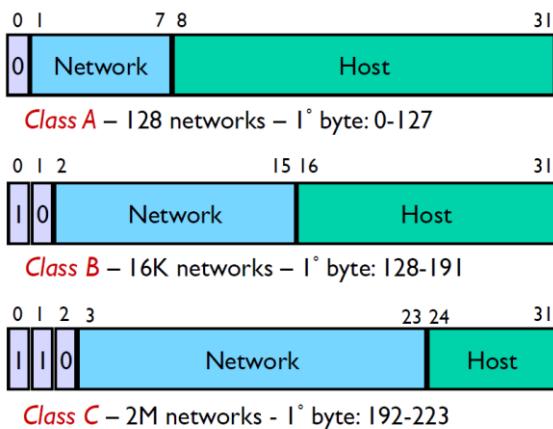
Per far avvenire la comunicazione si parte dal livello Applicazione, ad esempio un browser, che deve inviare una richiesta a un server che sta sulla rete (B). Il browser, a fronte dell'invio dell'URL della pagina a cui si vuole accedere, crea un messaggio che viene passato al livello 4 (nel caso del browser il TCP). Il messaggio viene incapsulato in un segmento e passato a IP. Questo livello crea il datagram, il cui segmento farà parte del payload. Verranno aggiunti l'IP sorgente e destinazione all'header e IP si chiederà se l'IP è nella propria rete. In questo caso non lo è, perciò dovrà effettuare una consegna **indiretta** e invierà il datagram al **default gateway**. Nell'IP destinazione **NON** viene scritto il default gateway, ma l'IP di B. Fatto ciò passa il tutto a Ethernet, che effettuerà la comunicazione tra A e R. Qui interviene **ARP** ma si suppone che l'indirizzo MAC di R sia già presente nelle tabelle ARP perciò viene aggiunta la trama ethernet al pacchetto (con MAC sorgente e destinazione, che contiene **il MAC del router**) ed è giusto che sia così perché il datagram è per B, ma la trama creata è per il router, che è l'unico modo per spedire il datagram a B.

Il router riceve la trama, nota che l'indirizzo MAC è il proprio e allora manda la trama a ethernet, che toglierà la propria trama e verrà passato al livello IP del Router. Si leggerà l'indirizzo IP di destinazione, lo cercherà nella routing table e scoprirà che l'indirizzo IP di destinazione è raggiungibile direttamente attraverso l'interfaccia 222.222.222.220 perché è in quella subnet. Viene decrementato il TTL e quindi ricalcolato l'header checksum. Successivamente R ridà il datagram a ethernet e di nuovo per conoscere l'indirizzo MAC di B, interverrà di nuovo ARP che se non avrà la corrispondenza IP-MAC effettuerà un'ARP Request. Una volta trovato l'indirizzo MAC di B, verrà costruita la trama Ethernet che conterrà come MAC sorgente quello del router e come MAC destinazione quello di B. La trama arriverà a B il quale riceverà la trama, che verrà processata passando ai piani superiori i vari dati.

Indirizzamento IP

- **Classful addressing:** divisione statica tra la parte di rete e la parte di host definita da alcuni bit iniziali. Con questa suddivisione le reti IP erano possibili solo con 3 dimensioni possibili: 2^8 , 2^{16} o 2^{24} . All'epoca non esisteva neanche il concetto di subnet.
- **Subnetting:** partendo da un indirizzamento classful si definiscono delle reti IP più piccole chiamate "subnets". Viene quindi introdotto il concetto di subnet e subnet mask. Le reti di grosse dimensioni vengono quindi ridotte in reti più piccole. Le subnet mask servono per capire dove finisce la subnet ID e inizia l'host ID (VLSM, Variable Length Subnet Masking).
- **Classless addressing:** il concetto di classe viene completamente rimosso e quindi viene rimosso anche il concetto di subnet. Non ho quindi reti più grosse che contengono reti più piccole, ma vado a costruire tante sottoreti (si utilizza il concetto di **netmask** e non più di **subnet**, anche se viene comunemente utilizzata).

IP addressing: le Classi

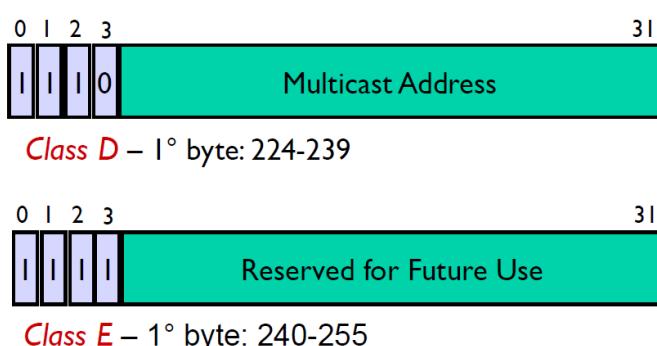


Per identificare in questo primo caso le dimensioni della rete, lo capisco guardando i primi bit. In particolare, la rete di classe A da 2^{24} indirizzi, la si identifica grazie allo 0 nel primo bit. Ad esempio, 11.0.0.0 è una rete di classe A perché 11 sta tra 0 e 127. Per essere di classe A devo avere 24 bit a 0. Altro esempio: 11.2.0.0 non è un net ID perché "11" mi dice che devo avere 24 bit per la rete, quindi è uno degli indirizzi che si possono assegnare ai dispositivi.

La classe B la si identifica con un 10 sui primi bit, perciò tutti i byte che vanno da 128 a 191. Quindi

130.182.0.0 è di classe B e la classe B ha una host part lunga 16 bit ed è un network ID, perché ho almeno 16 bit finali a 0. Altro esempio 130.192.1.0 non è un network ID.

La classe C ha nei primi 3 bit "110" perciò tutti i numeri da 192 a 223. Per esempio, 192.168.0.0 è un network ID e la parte di rete è 192.168.0 (ATTENZIONE A QUESTO ESEMPIO). Ci si ferma a 223 in questo caso perché ci sono altre due classi:



La classe D è riservata agli indirizzi multicast (da 224 a 239).

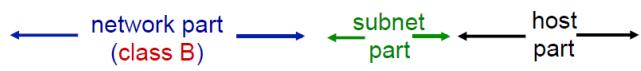
La classe E è riservata per utilizzi futuri.

IP addressing: subnetting

Possibilità di definire reti più piccole all'interno di reti più grandi.

Devo trovare un modo per definire dove finisce la subnet part e la parte di host.

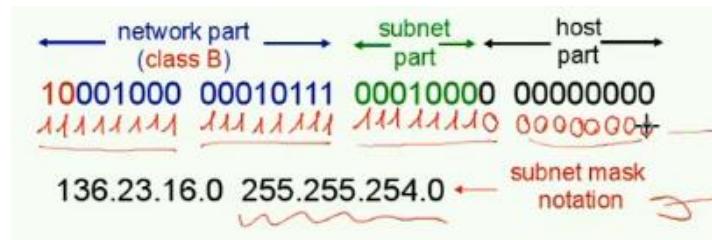
La **subnet mask** è una stringa di 32 bit in cui si hanno tutti 1 nella subnet part e tutti 0 nella host part.



10001000 00010111 00010000 00000000

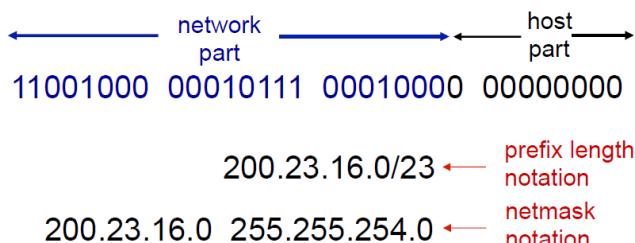
136.23.16.0 255.255.254.0 ← **subnet mask notation**

Parto da una rete di classe B e applico la subnet mask come segue:



136.23.0.0 era la classe B di partenza, ma applicando il concetto di sub network ho creato al suo interno una rete più piccola da soltanto 512 indirizzi. In particolare, ho scelto la 16.0 che in un concetto di classful sarebbe un indirizzo per uno dei 2^{16} host da usare all'interno della classe di 136.23.0.0 adesso in questo concetto di subnetting diventa un network ID grazie alla subnet mask.

Altro esempio:



Anche in questo caso si hanno 2^9 indirizzi, ma si nota che è possibile scrivere la subnet mask come **prefix length**. Le due notazioni sono equivalenti.

L'unica differenza è il "200" presente invece di "136".

La coppia dei due indirizzi avrebbe senso in un contesto di subnetting? No, perché essendo una rete di Classe C può contenere al massimo 256 indirizzi, mentre la subnet ne può contenere 512.

Questo metodo si chiama **CIDR (Classless InterDomain Routing)**.

Le netmask valide sul 4° byte possono essere:

0	0000 0000	(256)
128	1000 0000	(128)
192	1100 0000	(64)
224	1110 0000	(32)
240	1111 0000	(16)
248	1111 1000	(8)
252	1111 1100	(4)
254	1111 1110	(2)*
255	1111 1111	(1)

Es. 255.255.255.128 mi dà una rete con 128 host possibili.

Un caso speciale è 255.255.255.254 che non è possibile scrivere, perché a causa degli indirizzi speciali, una rete IP l'indirizzo che ha tutti 0 o 1 è speciale e non si possono usare sugli host, ed una rete di questo tipo conterrebbe solo questi due indirizzi.

*not valid in the 4° byte

Viceversa, la subnet mask 255.255.255.255 esiste, ma questo non è utilizzato nella definizione delle reti, ma è un modo che si può usare su una tabella di routing per identificare un singolo host.

ATTENZIONE: Si può usare 254 sul 3° byte, solo sul 4° è vietato.

Altri esempi

- 130.192.0.0/16 – 130.192.0.0 255.255.0.0
- 130.192.0.0/24 – 130.192.0.0 255.255.255.0
- 130.192.0.0/25 – 130.192.0.0 255.255.255.128
- 130.192.2.0/23 – 130.192.2.0 255.255.254.0
- 130.192.1.4/30 – 130.192.1.4 255.255.255.252
- ~~130.192.1.0/31 – 130.192.1.0 255.255.255.254~~

Each IP network **must** contain at least the network ID and the broadcast address!

Come si può notare, il primo equivale a un indirizzo di classe B poiché si hanno tutti i possibili indirizzi di tale classe, ovvero 2^{16} . Il primo corrisponde alla rete intera del politecnico.

Il secondo esempio può corrispondere alla rete assegnata a un solo laboratorio (es. LAIB1) del politecnico, che si suppone avere massimo 256 host. Quindi assegno una piccola fetta di tutti gli indirizzi che il politecnico possiede.

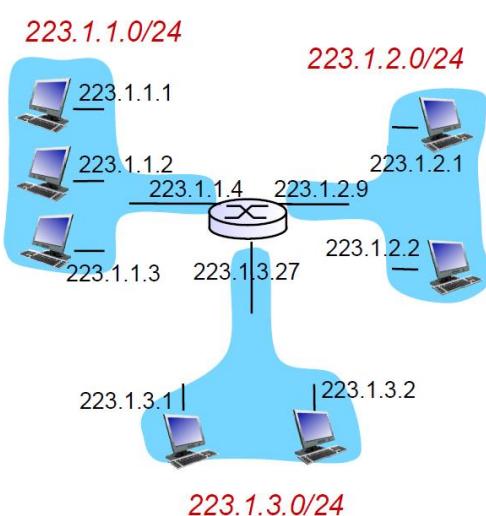
Nel terzo esempio, supponendo di dare 128 indirizzi al LAIB2 non si rispetta la corrispondenza biunivoca se teniamo conto dell'esempio precedente, poiché essendo la stessa porzione di indirizzi, potrebbero esistere due host con lo stesso indirizzo IP.

Il quarto esempio non si sovrappone con il secondo, perché l'indirizzo "130.192.2.0/23" è fuori dagli indirizzi del laboratorio (che finiscono con 0.255 e dunque da 1.0 sono liberi). Inoltre, è un net ID perché l'indirizzo 130.192.2.0/23 possiede 9 bit a 0.

Non avrei potuto utilizzare "130.192.1.0/23" perché non ha 9 bit a 0 (essendo un numero dispari l'ultimo sarà sicuramente un bit a 1).

Anche il 5° esempio è un net ID perché ha due bit a 0 (anche se non compaiono zeri nell'indirizzo IP).

Esempio



netmask: 255.255.255.0

Tornando all'esempio iniziale, si percepisce perché il professore diceva che "223.1.1" era la prima parte dell'indirizzo IP che identificava la rete.

Nell'esempio iniziale si utilizzava un approccio classful per identificare la rete, adesso invece viene specificata la rete.

Adesso si definisce una netmask e si esplicitano gli indirizzi con la notazione classless.

In questa notazione ogni dispositivo deve avere:

- Un indirizzo IP
- Una netmask (per capire la sua network ID e comunicare)
- Un **default gateway**, ovvero un first-hop router.

Indirizzamento Gerarchico

Se si volessero avere tanti indirizzi IP da dare ad una ipotetica azienda, bisognerebbe comprarli. Una volta acquistati si assegnano alle varie stazioni.

Una volta li si comprava direttamente da chi vendeva gli IP che assegnava degli slot completamente a caso. Tramite l'indirizzamento gerarchico ciò non si verifica più.

I provider comprano delle grosse reti IP (in porzione /8). Di questa rete IP i provider utilizzeranno una porzione per i propri router, ma la maggior parte resta inutilizzata per essere utilizzata in futuro. Quando ci si rivolge al provider per avere la connettività a internet, egli fornisce anche l'indirizzo IP utilizzando uno dei propri che sono avanzati. Così via tutti gli indirizzi vengono assegnati dal provider man mano che vengono richiesti. Visto che si prende sempre gli indirizzi dallo stesso posto, tutti i clienti di quel provider avranno dei bit in comune.

ESEMPIO

ISP's block	<u>11001000</u> <u>00010111</u> <u>00010000</u> <u>00000000</u>	200.23.16.0/20
Organization 0	<u>11001000</u> <u>00010111</u> <u>00010000</u> <u>00000000</u>	200.23.16.0/23
Organization 1	<u>11001000</u> <u>00010111</u> <u>00010010</u> <u>00000000</u>	200.23.18.0/23
Organization 2	<u>11001000</u> <u>00010111</u> <u>00010100</u> <u>00000000</u>	200.23.20.0/23
...
Organization 7	<u>11001000</u> <u>00010111</u> <u>00011110</u> <u>00000000</u>	200.23.30.0/23

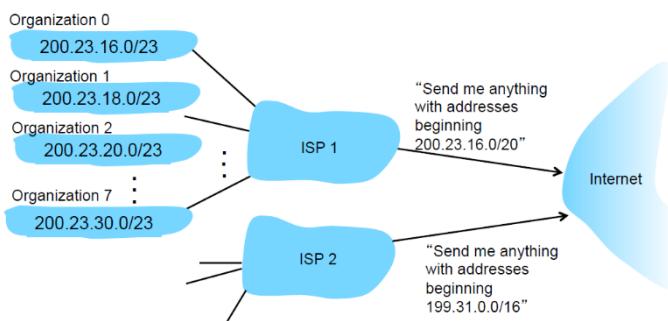
Il provider acquista gli indirizzi 200.23.16.0/20. Questo sarà il suo **address range** (range di indirizzi).

Appena un cliente chiede degli indirizzi, il primo range di indirizzi assegnabili sarà 200.23.16.0/23 (primo blocco da 512 indirizzi contenuto dentro 200.23.16.0/20).

La stessa cosa accadrà con l'organizzazione 1 che assegnerà lo slot 200.23.18.0/23. Questo è un net ID valido perché almeno gli ultimi 9 bit (23) devono essere a 0 e lo è perché 18 è numero pari e finisce per 0 perciò ne ha almeno 9.

Con l'ottavo cliente termina l'**address range** perché tutti i clienti hanno chiesto 512 indirizzi e il provider ne aveva $2^{12} = 4096$.

Si nota che le persone connesse con lo stesso provider sono rappresentabili con un unico range, quello del provider.



Tutti i clienti connessi a ISP 1 il router presente in Internet non deve sapere che ogni singola organizzazione è raggiungibile tramite diversi range di indirizzi, ma gli basta sapere che quelle organizzazioni sono raggiungibili con 200.23.16.0/20. Toccherà ai router del provider capire dove stanno le singole organizzazioni.

Uno dei motivi per cui si sta migrando verso IPv6 perché l'attuale IPv4 non è molto scalabile (si hanno comunque tante entry).

Funzionamento routing IP

Un host per capire qual è la propria rete IP fa un AND bit a bit (**bit-wise AND**) tra il proprio indirizzo IP e la maschera.

interface IP address: 192.168.10.69

1	1	0	0	0	0	0	0	0	1	0	1	0	1	0	0	0	1	0	1				
1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	
<hr/>																							
1	1	0	0	0	0	0	0	0	1	0	1	0	1	0	0	0	1	0	1	0	0	0	0

bit-wise AND result: 192.168.10.64

netmask: 255.255.255.192

La terza stringa binaria è di nuovo un indirizzo IP che però ha i bit della parte di host tutti a 0. Questa è la definizione di network ID.

Quando un host deve capire se fare consegna diretta/indiretta andrà a considerare l'indirizzo IP del destinatario e farà un AND bit a bit tra l'indirizzo IP destinatario e la propria netmask

destination IP address: 192.168.10.101

1	1	0	0	0	0	0	0	0	1	0	1	0	1	0	0	1	0	1					
1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0
<hr/>																							
1	1	0	0	0	0	0	0	0	1	0	1	0	1	0	0	0	1	0	1	0	0	0	0

bit-wise AND result: 192.168.10.64

netmask: 255.255.255.192

Applicando la propria maschera si riesce a capire dove sta il destinatario rispetto a sé stessi.

Se il risultato di questa seconda operazione coincide con quello della prima, vuol dire che il destinatario sarà sulla stessa rete, altrimenti no.

Non serve sapere a quale rete appartiene, ma basta sapere che non appartiene alla propria per iniziare una consegna indiretta.

In questo esempio il destinatario appartiene alla propria rete. Nel successivo invece, no

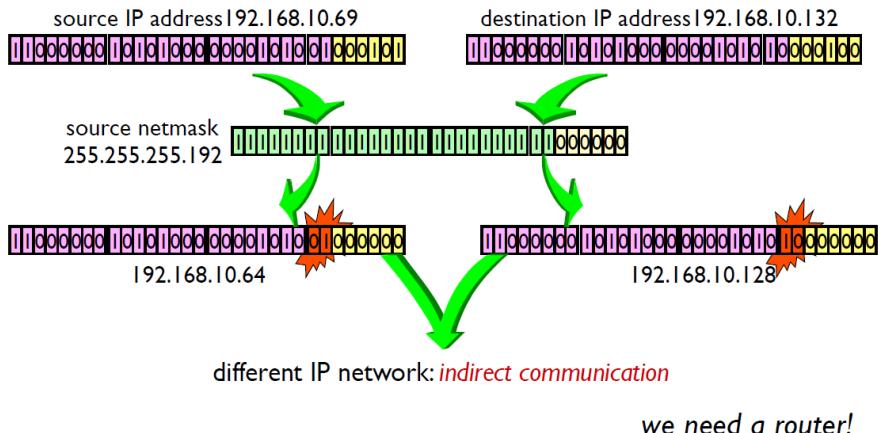
destination IP address: 192.168.10.132

1	1	0	0	0	0	0	1	0	1	0	1	0	0	0	0	1	0	0	1	0	0	0	0	
1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0
<hr/>																								
1	1	0	0	0	0	0	0	0	1	0	1	0	1	0	0	0	1	0	1	0	0	0	0	

bit-wise AND result: 192.168.10.128

netmask: 255.255.255.192

Perché il risultato del bit-wise AND è diverso dalla prima operazione.



In quest'ultimo esempio io 192.168.10.69 provo a contattare 192.168.10.132 e faccio le due operazioni per sapere che tipo di comunicazione utilizzare.

ESEMPIO SENZA AND BIT A BIT

Supponiamo di avere un indirizzo IP 192.168.10.69/26.

E' un ID di rete? No, perché è un indirizzo IP ma perché sono richiesti 6 bit a 0 ed essendo 69 un numero dispari ho 0 bit a 0.

Quale sarà l'indirizzo di rete che contiene questo indirizzo IP? Scendo fino ad avere un primo numero che ha 6 bit a 0. L'id di rete sarà 192.168.10.64/26.

Questo id di rete avrà $2^6 = 64$ possibili indirizzi che arriveranno fino a 10.127.

Se devo contattare 192.168.10.101 sarò dentro la mia rete e farò consegna diretta.

Se devo contattare 192.168.10.132 sarò fuori dalla rete e farò consegna indiretta.

Funzionamento Routing IP

L'algoritmo utilizzando su un router per trovare la corretta porta di uscita è sempre quello del **bit-wise AND**.

routing table	
destination	output link
200.23.16.0/20	1
199.31.0.0/16	2

Legge l'indirizzo IP destinazione del pacchetto e utilizza quello per accedere alla tabella di routing. Farà quindi l'AND bit a bit tra l'indirizzo IP di destinazione e la maschera di tutte le entry nella routing table (esempio dalla foto /20 e /16). Questo finché l'operazione di AND bit a bit restituisce l'indirizzo IP di quella maschera. Se viene fuori un risultato diverso non si ha un match. Se tra tutte le entry non si ha un match, il pacchetto viene scartato e si restituisce **destination unreachable**.

Esempio

destination IP address: 200.23.16.1

11001000 00010111 00010000 00000000
11111111 11111111 11110000 00000000
11001000 00010111 00010000 00000000

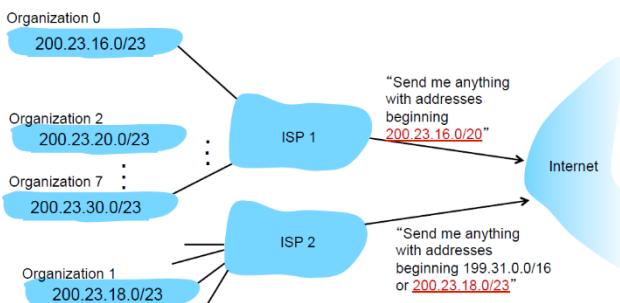
bit-wise AND result: 200.23.16.0 → match!

netmask: 255.255.240.0

200.23.16.1 come indirizzo IP di destinazione con maschera /20. Facendo l'AND bit a bit ottengo la prima entry della routing table, quindi ho un match e posso raggiungere tale indirizzo IP con la porta 1.

Se devo contattare 199.31.1.1 non avrò match con la prima entry facendo AND bit a bit ma proverò con la seconda entry e avrò match.

ISP 2 has a more specific route to Organization 1



Nel caso in cui l'organizzazione 1 decide di cambiare provider ma di mantenere i propri indirizzi IP, verrà fatto il trasloco di questi indirizzi a ISP 2. Dunque, sorge il problema perché bisogna gestire tale eccezione opportunamente con una entry apposita nelle tabelle di routing.

Come lo si gestisce? Longest prefix matching.

Longest prefix matching

Quando si hanno più match nella tabella di routing, si sceglie sempre quello ottenuto con una netmask più lunga.

Destination Address Range	Link interface
11001000 00010111 0001***** *****	0
11001000 00010111 0001001* *****	1
11000111 00011111 *****	1
otherwise	2

examples:

DA: 11001000 00010111 00010110 10100001 which interface?
DA: 11001000 00010111 00010010 10101010 which interface?

Gli asterischi definiscono la parte di host.

La prima entry è relativa al

200.23.16.0/20

La seconda è 200.23.18.0/23 che serve per gestire l'eccezione.

La terza è 199.31.0.0/16.

Sono le tre entry della figura precedente.

Se arriva un pacchetto IP destinato a 200.23.16.1 con chi farà match? Farà match con la prima entry e con la quarta (ma si tralascia questo caso). Escludendo la quarta riga si avrà match soltanto con la prima e si utilizza il solito algoritmo.

Se arriva un pacchetto IP con 200.23.18.1 farà match sia con la prima entry che con la seconda. Questo algoritmo ci dice che dovrà entrare nella porta che ha il prefisso più lungo, perciò prenderà la strada del link 1. Si usa questo algoritmo perché più il prefisso è lungo, più la rete è piccola.

Routing IP: una vera routing table

Una prima considerazione da fare è la seguente: lavorando con il protocollo IP è probabile che in una routing table debba comparire un indirizzo IP e non una porta. Non basta indicare la porta da cui il pacchetto "deve uscire", ma bisogna definire come funziona davvero una routing table.

In una routing table è possibile trovare 3 tipi di routes:

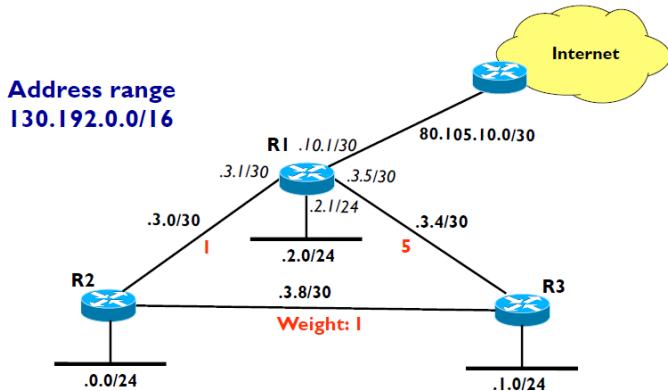
- **Direct routes:** route dirette che fanno riferimento a reti connesse direttamente al router. Non tutte le reti sono raggiungibili direttamente.

Le reti remote, connesse indirettamente possono essere:

- **Static routes:** route statiche che si scrivono manualmente
- **Dynamic routes:** route dinamiche scritte direttamente da un protocollo di routing

Esempio

Supponiamo di avere la seguente rete (le rette orizzontali sono reti ethernet):



Si parte cercando di capire quali sono le route statiche (perché scritte a mano) che devono comparire in R1. Una route statica è una route che serve per raggiungere una rete non direttamente connessa. Ci si aspetta di trovarne 4 (le due reti di R2 ed R3 e la rete che collega R2 ad R3, oltre che ad internet).

La prima colonna conterrà tutte le reti, quindi utilizzerò l'address range o più in generale id di rete e netmask. Come segue:

R1 routing table

Type	Destination	Next-hop	Cost
S	130.192.0.0/24	130.192.3.2	2
S	130.192.1.0/24	130.192.3.2	2
S	130.192.3.8/30	130.192.3.2	2
S	0.0.0.0/0	80.105.10.2	2
D	130.192.2.0/24	130.192.2.1	1
D	130.192.3.0/30	130.192.3.1	1
D	130.192.3.4/30	130.192.3.5	1
D	80.105.10.0/30	80.105.0.1	1

remote interface!!
local interface!!

Le prime 3 entry corrispondono alle 3 reti di cui prima. L'altra colonna deve contenere la strada per raggiungere tale destinazione. Visto che ci sono dei pesi sui link da considerare, se devo andare da R1 alla rete .0.0/24 vado da sinistra, idem per .1.0/24 che senza pesi sarebbe raggiungibile da R3, ma essendo a peso 5 conviene passare da R2.

Nella seconda colonna c'è quindi un unico indicatore di strada per tutte e 3 entry. L'indirizzo 130.192.3.2 è l'indirizzo dell'interfaccia di R2. Per capirlo, noto che l'id di rete del collegamento punto punto è .3.0/30 mentre l'interfaccia di R1 è .3.1/30. Essendo quindi entrambe occupate, la prima disponibile sarà .3.2/30. Si utilizza 3.2/30 e non 3.1/30 perché devo scrivere l'IP del prossimo router che può aiutare nel consegnare il pacchetto e quindi è quello il router a cui inviare il traffico.

La quarta entry è la default route. Qualunque IP farà match con l'indirizzo IP 0.0.0.0/0. Se ci sarà qualche match con prefissi più lunghi utilizzerà quella strada lì, altrimenti andrà in 0.0.0.0/0. Questa stringa rappresenta l'intera internet. In quel caso allora indirizzerà il pacchetto verso il next-hop che è il router 80.105.10.2.

Essendo le prime 3 entry tutte appartenenti alla rete 130.192.0.0/16 si potrebbero sostituire scrivendo soltanto un'unica entry 130.192.0.0/16 -> 130.192.3.2.

Essendo le prime 3 entry tutte con next-hop identico, si potrebbero sostituire tutte e 3 le entry con 0.0.0.0/0 e next-hop 130.192.3.2 (supponendo di non avere internet ed eliminando quella 0.0.0.0/0 già presente).

Successivamente sono presenti le reti connesse direttamente a R1 che sono altre 4 e sono illustrate nel grafico e nella tabella. Come next-hop andrò a scriverci l'interfaccia locale del dispositivo, che dipende dai vari link.

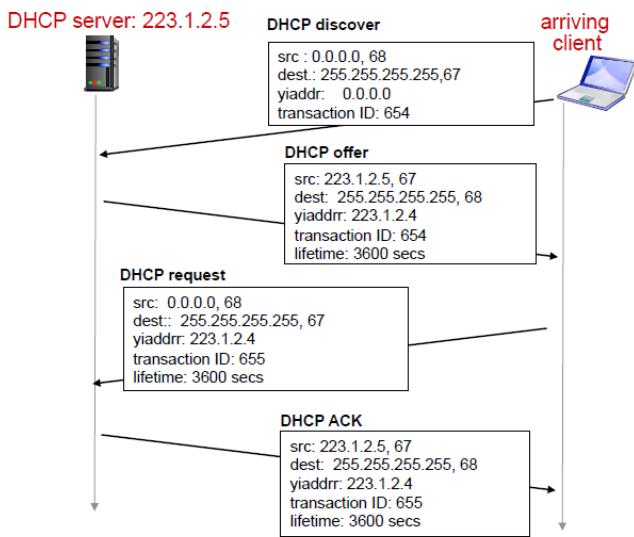
Minuto 1:50:30 lezione 29 per approfondimento con esempi.

Per ottenere un indirizzo IP si possono configurare a mano su tutti i dispositivi, che non è una buona scelta, oppure è possibile farsene assegnare uno da un server detto **DHCP (Dynamic Host Configuration Protocol)**.

DHCP

Questo è un approccio completamente plug-and-play perché il dispositivo viene configurato automaticamente. Il DHCP è composto da quattro messaggi:

- DHCP discover
- DHCP offer
- DHCP request
- DHCP ack



Alla base del protocollo DHCP c'è un DHCP server che qualcuno dovrà aver configurato.

Il client si connette alla rete e non avendo indirizzo IP deve comunicare col server DHCP. Non sapendo dove si trova il servizio, si utilizza una richiesta in **broadcast**. Il DHCP è un protocollo di livello **applicazione**, sfrutterà quindi anche le **porte** che sono gli indirizzi del livello 4.

Nella DHCP discover troviamo la destinazione che contiene l'indirizzo IP broadcast e la porta 67 perché si conosce che il software che implementa il DHCP server userà la porta 67 (che è fisso).

Questo messaggio raggiungerà quindi tutti gli host, che la spaccheranno ma trovando

all'interno l'indirizzo IP broadcast processeranno il datagram e lo passeranno al livello 4. Soltanto uno avrà un servizio attivo sulla porta 67 perciò tutti gli host lo scarteranno e solo il DHCP riceverà il DHCP discover.

La porta sorgente è la 68 ma l'indirizzo IP sorgente è a tutti 0, perché non è presente alcun indirizzo IP.

Solo il DHCP server sarà in grado di creare la **DHCP offer**, che non è altro che la risposta del server. Dentro tale risposta è presente l'indirizzo IP sorgente che è l'indirizzo IP del server. La porta sorgente è la porta destinazione che c'era prima nella discover. La destinazione della offer è di nuovo broadcast perché il client non ha ancora un indirizzo IP. Successivamente si ha il campo relativo a **yiaddr** che contiene l'indirizzo IP che il server sta proponendo al client che vuole collegarsi.

Successivamente devo fare una **DHCP request** dove accetto l'offerta e propongo di collegarmi con quell'indirizzo IP. Questo serve perché sulla rete potrebbero esserci più di un indirizzo IP perciò più DHCP potrebbero rispondermi con delle offer. Tra tutte le offer verrà accettata una e inviata una request.

Viene scelta la prima ricevuta. Ancora una volta la request avrà sempre sorgente a tutti 0 e destinazione 1. È ancora una volta in broadcast in modo tale da avvisare tutti i server DHCP quale offerta si ha scelto.

Infine, il server risponderà con un **DHCP ACK** ancora una volta in broadcast per avvisare tutti i DHCP che il server ha accettato la richiesta e il processo finisce.

Il lifetime dei pacchetti serve perché molti indirizzi IP vengono riutilizzati. Siccome quando ci si disconnette non vengono dati avvisi, l'unico modo per sapere se un indirizzo IP si è liberato è tramite un timer in cui se non refresho il mio indirizzo IP, tale indirizzo viene riutilizzato per un altro client. Se lo voglio rinnovare farò un'altra request.

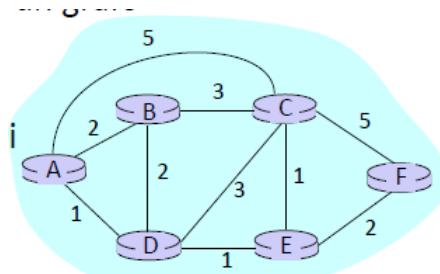
2. Strato 3: Instradamento

Una delle funzioni dello strato rete è l'**instradamento** (o **routing**) effettuato consultando delle **tabelle di instradamento**, in modo da decidere la strada da far prendere alle PDU della rete datagram. Per fare ciò vengono utilizzati degli **algoritmi di instradamento**.

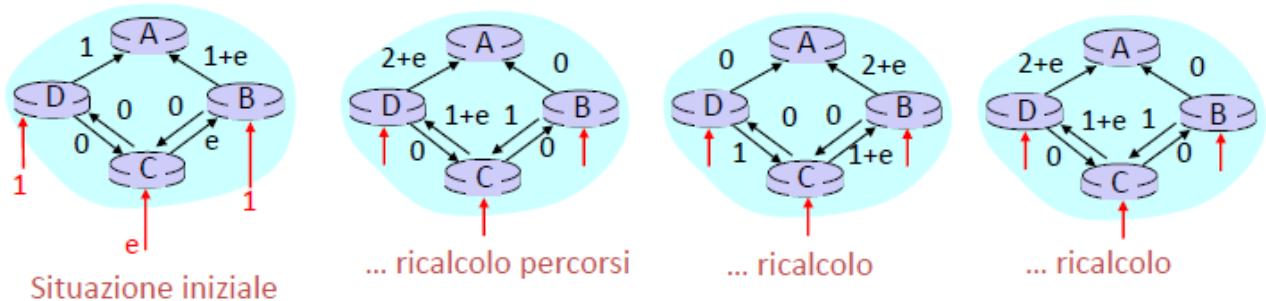
Un **protocollo di instradamento** e un **algoritmo di instradamento** lavorano insieme col compito di compilare le tabelle di routing. È possibile configurare le tabelle di instradamento manualmente, ma ciò è di estrema difficoltà nel caso di reti globali. Il protocollo definisce le modalità di scambio di informazioni sullo stato della rete, mentre l'algoritmo rappresenta le operazioni necessarie per **scegliere** il percorso verso la destinazione, date le informazioni sullo stato della rete. N.B. (è l'algoritmo che sceglie, non il protocollo).

Algoritmi di instradamento

L'obiettivo degli algoritmi di instradamento è quello di determinare un **buon** percorso (sequenza di link o nodi) nella rete da nodo sorgente a nodo destinazione. Per fare ciò si trasforma la topologia in un **grafo** e si assegnano dei **costi** agli archi e il **buon** percorso non è altro che il **percorso a costo minimo**. I costi sono una quantità adimensionale.



Il costo può essere un mix di distanza, ritardo, euro, livello di congestione. Questo costo può essere **statico** o **dipendente dallo stato della rete**. Una possibile oscillazione può essere quella dovuta al costo dei canali legato al carico trasportato (in foto).



Nella prima figura D manda il carico verso A perciò l'arco avrà peso 1. Idem B ad A. C ha due alternative: senso orario o antiorario. Nella prima foto va in senso antiorario e manda il carico a B (e dunque l'arco verso A diventa $1+e$). B a quel punto si vedrebbe un costo minore andando in verso orario, e la stessa cosa farà C arrivando alla figura 2. A quel punto verrà ricalcolato nuovamente il percorso andando a creare un effetto **ping-pong** dell'instradamento.

Esempi di algoritmi di instradamento

Semplici algoritmi senza necessità di coordinamento da parte dei nodi:

- **Random**: scelgo a caso una porta di uscita
- **Flooding**: instradato verso tutte le porte disponibili
- **Deflessione o hot potato**: su tipologie regolari, instradato verso la porta corretta, se libera. Se occupato, instradato verso un'altra porta libera.

Questi protocolli semplici sono utili quando i router non conoscono ancora la topologia della rete e non sanno quanti e quali router sono connessi (per far conoscere la topologia).

Gli algoritmi complessi per il calcolo del percorso "ottimo" hanno una diversa classificazione.

Il calcolo del percorso può essere:

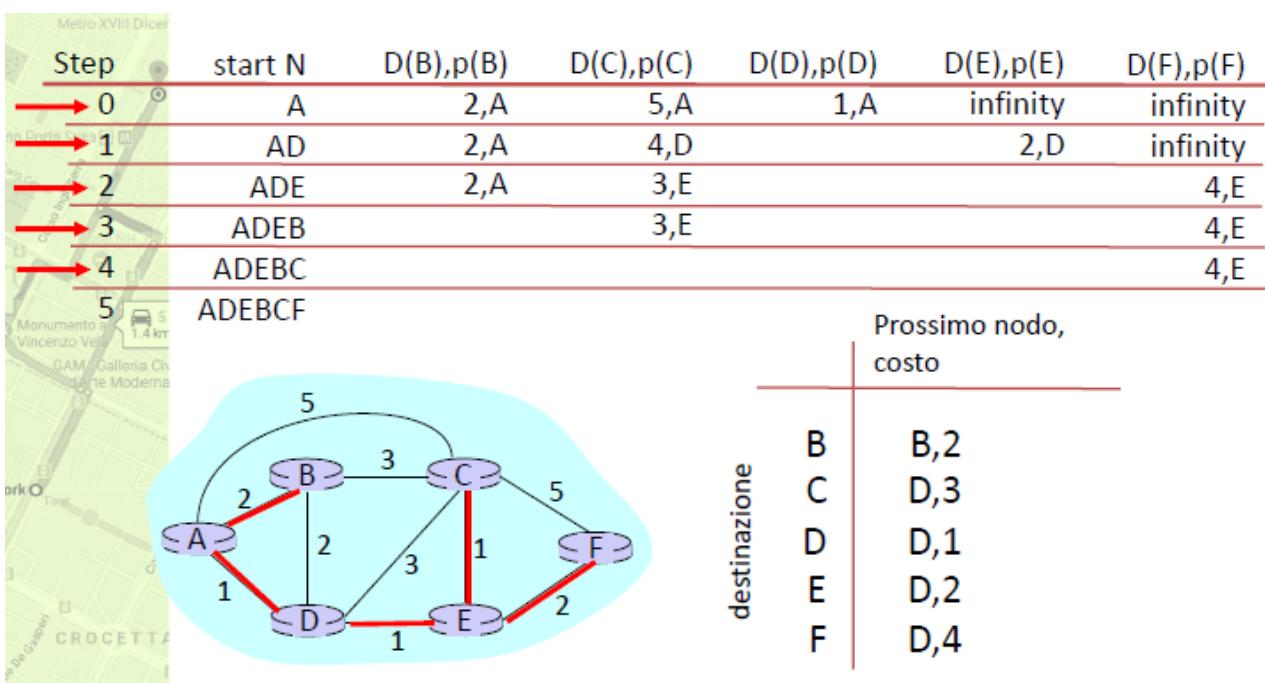
- **Centralizzato:** un nodo si occupa di raccogliere l'informazione da tutti gli altri nodi, calcola i percorsi e ridistribuisce il risultato del calcolo a tutti gli altri nodi
I vantaggi risiedono nella possibilità di utilizzare algoritmi e metriche complessi e tutti i nodi usano un piano di instradamento coerente.
Lo svantaggio risiede nei possibili guasti del nodo centrale (o del link tra il nodo e gli altri router) e inoltre il nodo può essere congestionato poiché riceverà dati da tutti quanti.
- **Distribuito:** tutti i nodi si scambiano informazione tra loro (utilizzando protocolli di instradamento) e calcolano i percorsi (indipendentemente o in base a quanto fatto dai nodi adiacenti).
I vantaggi risiedono nella robustezza ai guasti e lo scambio di informazioni che è uniforme in tutta la rete.
Gli svantaggi riguardano le incongruenze nell'instradamento (loop) e la richiesta di "intelligenza" ai nodi.

Gli algoritmi distribuiti - informazione

- **Globale:** tutti i nodi conoscono la topologia completa (nome dei nodi, connettività dei nodi) compresi i costi dei canali. Vi è quindi uno scambio di informazioni tra tutti i nodi e questi algoritmi vengono detti **link state** (ad esempio il protocollo **OSPF** usa questi algoritmi).
- **Parziale:** i nodi hanno una sorta di "lista limitata" e conoscono i nodi cui sono fisicamente collegati e conoscono le destinazioni raggiungibili attraverso questi router. Si scambiano informazioni **solo** con i nodi adiacenti. Questi algoritmi vengono detti **distance vector** (ad esempio il protocollo **RIP**, **BGP** usa questi algoritmi)

Algoritmi link state

Ogni nodo invia informazioni di costo dei suoi canali a tutti gli altri nodi della rete (in broadcast o multicast). Tutti i nodi costruiscono una topologia della rete e conoscono i costi di tutti gli archi. Questi algoritmi vengono utilizzati nelle reti degli operatori (che hanno un numero ridotto di router all'interno della sua rete). Data la topologia ogni nodo calcola i percorsi a minimo costo verso tutti gli altri nodi tramite **algoritmo di Dijkstra**.



Tutti i nodi estrarranno lo stesso risultato (cioè l'albero a costo minimo). Nella figura si suppone che l'algoritmo venga eseguito da A.

- Step 0: A vede qual è il costo minimo che conosce per raggiungere gli altri nodi della rete collegati a lui. (B,C,D). A per andare verso B ha costo 2 (e così via). Tra tutte le alternative si fissa quella a costo minimo (A->D).
- Step 1: D può raggiungere B,C,E. B si può raggiungere con costo 2 (tramite A), C si può raggiungere con costo complessivo 4 (tramite D) ed E con costo 2. Completo lo step selezionando il percorso a peso minimo (in questo caso ho il percorso verso B ed E e prendo E).
- Step 2: B continua ad avere costo 2, C ha costo 3 (passando attraverso E) e vedo anche F passando attraverso E con costo 4.
- Step 3: Prendo il nodo a peso minore (B) e confermo che tale percorso (A->B) è il migliore poiché non migliora alcuna stima.
- Step 4: prelevo il nodo C e noto che non migliora alcuna stima.
- Step 5: prelevando F concludo l'algoritmo e ho costruito l'albero a costo minimo.

	Prossimo nodo, costo
destinazione	
B	B,2
C	D,3
D	D,1
E	D,2
F	D,4

Complessità con M nodi $O(M \log(M))$

Algoritmi Distance Vector

Sono algoritmi iterativi che continuano fino a quando i nodi non scambiano più informazioni e termina in modo **autonomo** (cioè non vi è alcun segnale di fine algoritmo) ed è di tipo distribuito. Ogni nodo comunica soltanto con i nodi adiacenti.

Ogni nodo scambia periodicamente con i vicini diretti un **vettore** contenente: le **destinazioni** che può raggiungere e la **distanza** dalle destinazioni misurata in **costo** (ad esempio: numero nodi da attraversare compreso se stesso).

Il nodo che riceve il vettore lo confronta con la propria RT (Routing Table, tabella di instradamento) ed effettua eventuali modifiche: se scopre nuove destinazioni, se cambia instradamenti poiché i nuovi sono più brevi, modifica i costi se usa un nodo adiacente come miglior scelta.

I distance vector sono **facili da implementare** ma hanno dei problemi.

Questi algoritmi sono **lenti a convergere**, cioè impiegano molto tempo a raggiungere la configurazione finale e possono sorgere problemi durante lo scambio di informazioni. Eventuali errori di routing vengono **propagati** (es. costo 1 verso un nodo lontano non veritiero). Infine, **non è molto scalabile** (le dimensioni dei messaggi scambiati dai nodi crescono al crescere della rete).

Per implementare l'algoritmo è necessaria una **tavella delle distanze**. Ogni nodo ne possiederà una e conterrà **una riga** per ogni possibile destinazione e **una colonna** per ogni nodo adiacente.

- Esempio: nel nodo X, per la destinazione Y attraverso nodo adiacente Z:

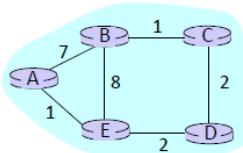
$$\text{Sorgente } X \xrightarrow{\quad} D(Y,Z) = c(X,Z) + \min_w \{D^Z(Y,w)\}$$

Destinazione Next hop

La formula dice che per andare da X a Y passando da Z, bisogna sommare il costo per andare da X a Z + il minimo percorso che lega Z a Y (che viene fornito da Z).

Quindi la prima parte dipende dal proprio nodo, mentre la seconda parte della formula è soggettiva e dipende dai calcoli che tale nodo effettua.

Tabella distanze: esempio



$$D^E(C,D) = c(E,D) + \min_w \{D^D(C,w)\} = 2+2 = 4$$

$$D^E(A,D) = c(E,D) + \min_w \{D^D(A,w)\} = 2+3 = 5 \text{ anello!}$$

$$D^E(A,B) = c(E,B) + \min_w \{D^B(A,w)\} = 8+6 = 14 \text{ anello!}$$

		Costo verso destinazione attraverso nodo		
		A	B	D
destinazione		D ^E (A)		
	A	1	14	5
	B	7	8	5
	C	6	9	4
	D	4	11	2

In questo caso si parte dal nodo E e si calcola la tabella delle distanze del nodo E. Si ha una riga per ogni destinazione (da E) e si ha una colonna per ogni nodo **adiacente** ad E.

Per calcolare la prima riga si effettua la seguente domanda:

“Qual è il costo per andare da E in A?”

“Qual è il costo per andare da E in A attraverso B?”

In questa seconda domanda, il costo per andare da E ad A attraverso B potrebbe sembrare 15. Però bisogna pensare che il costo per andare da E ad A dipende da quale costo minore B ha riferito ad E. B sa che il costo minore per andare in A è 6 (B->C->D->E->A) perciò E calcolerà il costo per andare a B + il costo fornito da B.

“Qual è il costo per andare da E in A attraverso D?” In questo caso D dirà ad E che il costo per andare ad A è 3. D però conosce ciò perché gli è stato fornito da E. Il 5 nella prima riga nasconde un **loop**, che se l'algoritmo viene eseguito correttamente, non verrà mai preso perché si troverà un percorso che avrà costo inferiore.

Una volta ottenuta la tabella delle distanze, si passa a quella di instradamento prendendo su ogni riga qual è la colonna che ha costo minore e si prende quella come **next-hop** (prossimo nodo).

E darà ai suoi vicini tale **vettore** (cioè l'estratto della tabella delle distanze, che sarà anche la sua tabella di routing).

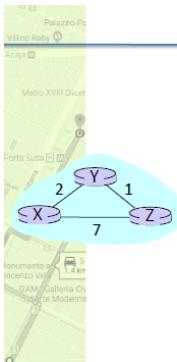
	next-hop
A	A,1
B	D,5
C	D,4
D	D,2

Instradamento Distance Vector

In generale questo algoritmo procede in modo **iterativo** ed in maniera **asincrona**, cioè ogni nodo invia periodicamente ai nodi vicini degli aggiornamenti. Il protocollo prevede che si mandino degli aggiornamenti ai vicini, ma si arriva al punto in cui non viene più eseguito alcun aggiornamento alla tabella di routing poiché si conoscono le informazioni di tutti. Questo a meno di modifiche, ad esempio nel caso in cui varia un costo per raggiungere una destinazione).

Ogni nodo aspetta in un **loop infinito** che gli arrivino degli aggiornamenti e quando arriva ricalcola, se necessario, la tabella delle distanze.

Esempio



Distance Vector: esempio

	X	Y	Z
D ^X	2	∞	7
D ^Y	2	∞	1
D ^Z	∞	1	3

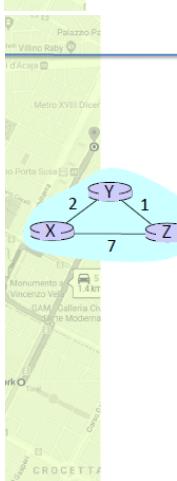
	X	Y	Z
D ^X	2	8	7
D ^Y	2	8	1
D ^Z	3	7	1

$$D^X(Y,Z) = c(X,Z) + \min_w \{D^Z(Y,w)\}$$

$$= 7+1 = 8$$

$$D^X(Z,Y) = c(X,Y) + \min_w \{D^Y(Z,w)\}$$

$$= 2+1 = 3$$



Distance Vector: esempio

	X	Y	Z
D ^X	2	∞	7
D ^Y	2	∞	1
D ^Z	∞	1	3

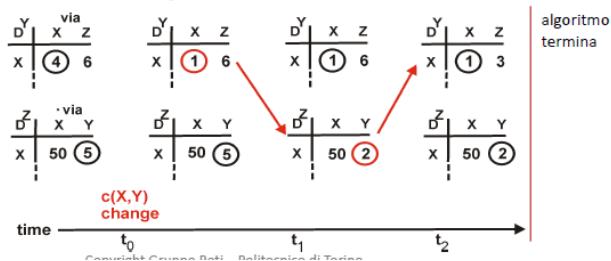
	X	Y	Z
D ^X	2	8	7
D ^Y	2	8	1
D ^Z	3	7	1

	X	Y	Z
D ^X	7	∞	1
D ^Y	7	∞	1
D ^Z	9	1	1

Se il costo di un canale viene modificato, il nodo ne riconosce la modifica, aggiorna le proprie tabelle delle distanze e se la modifica implica la modifica del cammino migliore vengono avvisati i nodi adiacenti.

Le modifiche al canale possono essere: **good news travels fast**, **bad news travels slow** (problema del count to infinity). Si nota che nel secondo caso l'algoritmo prosegue, perché Z ha ancora salvato il vecchio valore.

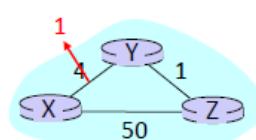
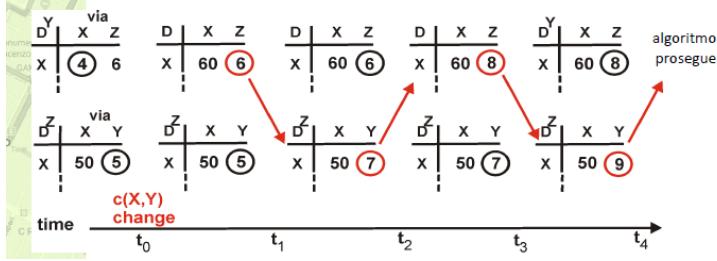
"*good news travels fast*"



All'inizio le tabelle delle distanze avranno un aspetto come indicato per ciascuno dei nodi. X vedrà che il costo per andare a Y passando da Z è 2, mentre il costo per andare a Z è sconosciuto (e così via). Andrà quindi a chiedere come raggiungere tali nodi e Z dirà che per andare a Y il costo minimo è 1 (che sarà +7 col nodo X->Z) quindi 8.

Così via fino a riempire tutte le tabelle di tutti i nodi.

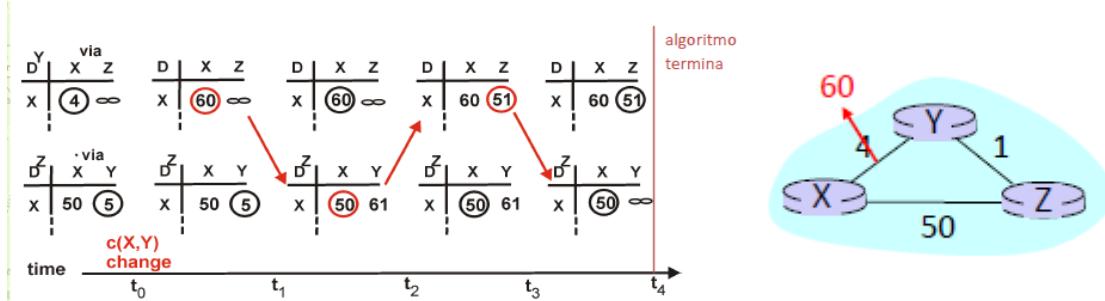
- **bad news travels slow** - problema del "count to infinity"!



Nel secondo caso, all'istante t_0 cambia il costo del canale. Z non se ne accorge, ma se ne accorge per primo Y e, siccome il costo per andare a X passando da X è aumentato, si sceglie il percorso passando da Z (che è 6). Questo valore è però un loop, perché quel valore è la visione della rete che aveva Z **prima** del guasto. Y prende per buona una informazione non vera, perciò inizia a propagare l'informazione e informa Z che adesso per andare a X costa 6. Per Z adesso il costo per raggiungere X passando da Y è modificato, perciò incrementerà il costo (6+1). A questo punto il costo è cambiato e lo comunica a Y e così via, entrando in un loop.

Per cercare di risolvere il problema, si utilizza l'algoritmo **Poisoned Reverse**.

Se Z instrada via Y per raggiungere X: Z comunica ad Y che la sua distanza verso X è infinito (e quindi Y non instraderà verso X passando da Z) ma ciò non risolve completamente il problema.



Guardando Z il peso minore per raggiungere X è tramite Y (cerchio nero sul 5), Z comunicherà a Y che la sua distanza verso X è infinito.

In questo modo, al cambio di costo, Y aggiorna il nuovo costo, che è anche il nuovo costo minore (60) e lo comunica a Z che provvederà ad aggiornare le proprie tabelle. Adesso per Z raggiungere X tramite Y costa $60+1$, perciò il costo minore è 50 (passando per X) e lo comunica a Y. Y salverà tale valore nella tabella ed essendo inferiore a 60, il costo minimo viene aggiornato e comunicato a Z. Visto che però Y comunica con X attraverso Z, dirà che il costo è infinito. Z aggiornerà la propria tabella e il costo minimo continua ad essere 50. L'algoritmo termina.

Confronto tra algoritmi LS e DV

Riguardo la complessità dei messaggi: con M nodi, E canali per nodo

- Algoritmi link-state: ogni nodo invia $O(M)$ messaggi a tutti gli altri $M-1$ nodi, ciascuno lungo $O(E)$. Ogni messaggio sarà quindi breve.
- Distance Vector: ogni messaggio contiene tutte le destinazioni $O(M)$ (messaggi lunghi), inviato però a $O(E)$ vicini.

Quindi complessità **$O(E M)$** . Il distance vector è però migliorabile, cercando di compattare le informazioni nei messaggi. Velocità di convergenza:

- Algoritmi link-state: ogni volta che un link state è propagato, si ha una nuova topologia: convergenza immediata.
- Distance Vector: le scelte del nodo dipendono dalle scelte dei nodi vicini; si richiedono dunque più scambi di messaggi e il tempo di convergenza è variabile.

Se un nodo non funziona correttamente, nel caso del link state i nodi possono annunciare costi dei canali scorretti e dunque ogni nodo, calcolando la propria tabella, sbaglia. All'annuncio successivo viene però corretto il tutto (non vengono generati anelli). Ne caso del distance vector, i nodi possono annunciare i costi dei cammini scorretti ed ogni annuncio viene usato da tutti i nodi (indirettamente) perciò gli errori si propagano nella rete e possono creare anelli.

3. Domain Name System – DNS

Protocollo accessorio (rispetto al protocollo IP) implementato al livello Applicazione.

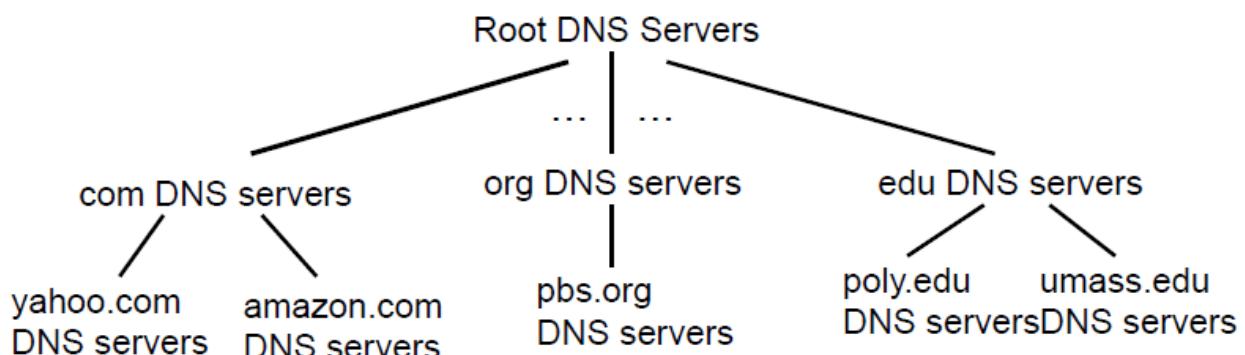
Non è corretto dire protocollo DNS perché è più un sistema che un protocollo (perché c'è una struttura che implementa il DNS).

Il DNS permette di utilizzare dei nomi (stringhe di caratteri) invece di indirizzi IP. Si utilizza ciò perché è più facile ricordare i nomi piuttosto che gli indirizzi IP.

Esiste un'infrastruttura di server che tutti insieme offrono un servizio di **database distribuito**. Questi server si chiamano **name servers** (o **DNS Server**). I vari dispositivi IP possono utilizzare questo protocollo per inviare delle richieste al sistema DNS per avere indietro le risposte (dato un nome si riceve il relativo indirizzo IP).

Si usano database distribuiti e non centralizzati perché avere un unico punto per tutte le informazioni, vuol dire che diventa fondamentale e se c'è un problema in quel server tutto il sistema non funziona più, perché si avrebbero problemi nella gestione del traffico (perché arriverebbero tutte le richieste in quel punto), perché la velocità diminuisce con l'aumentare della distanza del server dall'utente (più che altro 40 anni fa, oggi non è un problema) e c'è anche un discorso legato alla manutenzione del server perché il gestore di quel server avrebbe in mano le sorti dell'intera rete. Il tutto si riassume dicendo che un database centralizzato **non scalerebbe**.

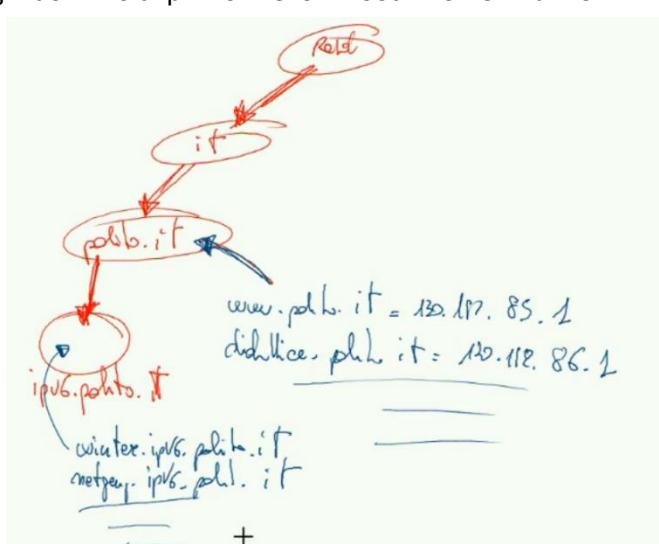
I DNS sono suddivisi in una scala gerarchica



Sul Root DNS Server è presente solo la lista di altri server che sono responsabili dei **domini di primo livello (TLD Top Level Domain)**. Questi domini di I livello sono i .com, .org, .edu, etc..

I root server sanno chi sono i server responsabili per ogni dominio di primo livello. I Root DNS **non** hanno il mapping tra indirizzo IP e stringa ma solo dei "puntatori" ad altri server.

I **domini di II livello** sono quelli identificati dalla parte a "sinistra" del .com/.it/etc. come "yahoo.com", "amazon.com" e in questi server è possibile vedere dei mapping tra nomi e indirizzi IP. Ad esempio "www.amazon.com" non è più semplicemente un dominio che contiene altri domini ma è un nome associato ad un certo indirizzo IP. Amazon.com invece è la famiglia di tutti i nomi che sono all'interno di questo dominio (www.qualcosa.amazon.com per esempio). Questi domini di III livello (es. didattica.polito.it) hanno all'interno i mapping relativi al proprio



dominio.

I server relativo ad amazon.com non avrà all'interno il mapping tra www.polito.it e il suo indirizzo IP (perché non sta dentro quel dominio). All'interno sarà contenuto ad esempio www.amazon.com.

Non è detto che ci si detta fermare in questo livello di gerarchia, ma all'interno del dominio polito.it esiste il dominio ipv6.polito.it che è un **dominio di III livello** che sta dentro polito.it al cui interno ci sono poi dei nomi associati alle varie macchine, ad esempio netgroup.ipv6.polito.it.

I server che hanno dei mapping per un certo numero di nomi sono detti **autoritativi** (per il dominio a cui sono stati associati). Ovvero polito.it è autoritativo per polito.it e NON per ipv6.polito.it. Ognuno è autoritativo per la serie di nomi di cui è responsabile.

Una **richiesta DNS (risoluzione DNS)** funziona entrando nella gerarchia accedendo al root server, dopodiché il Root Server comunicherà qual è il server responsabile del dominio di I livello che si sta cercando. A sua volta il dominio di I livello comunicherà qual è il server che gestisce il dominio di II livello e così via fino ad ottenere il mapping.

Root name servers

Il Root Server non è unico, ma ne esistono molteplici (circa 15) e quindi si hanno più punti di accesso e in genere si utilizza il più vicino.

Questi Root Server dislocati nel mondo devono essere allineati tra loro e la difficoltà si presenta quando queste informazioni sono molto dinamiche ma le informazioni sul Root Server non lo sono, perché contiene soltanto i "Puntatori" ai server di I livello. Le informazioni vengono modificate solo quando un server di I livello cambia indirizzo IP oppure quando si aggiunge un nuovo dominio di I livello ma questa cosa succede molto lentamente.

I DNS Server autoritativi per un certo dominio possono essere mantenuti dall'organizzazione che ha richiesto quel dominio, ma può essere anche fornito ad altri (ad es. il provider che fornisce il servizio). Quando ad esempio si acquista un dominio su Aruba, il DNS viene gestito da Aruba stesso che avrà un unico server che è autoritativo per tanti domini. Cioè comunicherà a chi gestisce il server di I livello che il nuovo dominio dovrà essere linkato correttamente al proprio server.

Local DNS server e caching

E' un server DNS locale che aiuta il client nell'accesso all'infrastruttura. Ogni utente possiede il client (come browser o client di posta) che invia le richieste al server locale (vicino a noi) che si fa carico della richiesta e andrà dal root server ad effettuare la richiesta. Alla fine dell'interazione il DNS locale ci invierà la risposta.

La cosa più importante è però la funzione di **caching** delle informazioni recuperate dalla gerarchia di server. Il server DNS una volta ottenuto l'indirizzo IP di un server di I livello, non passerà più tramite il root server poiché a successive richieste per quel dominio di I livello si avrà già il mapping.

In questo modo la gerarchia si nota non essere molto utilizzata.

Questo sistema è molto efficiente poiché la lista di nomi utilizzati è comunque ristretta, quindi è molto probabile trovare in cache un sito che si sta cercando.

Il local DNS server in casa sta nel router e viene configurato automaticamente dal DHCP quando si accede alla rete di casa. In genere il local DNS server avrà lo stesso indirizzo IP del default gateway.

Possono essere presenti più livelli di caching, cioè dopo il local DNS della propria abitazione possono essere presenti nella rete del provider i local DNS del provider, che garantisce maggiore scalabilità. In sostanza i local DNS posseggono dei puntatori ai Local DNS del provider che a sua volta punteranno al Root Server.

Il puntatore dal Local DNS al Local DNS ISP verrà fornito dal server DHCP alla connessione.

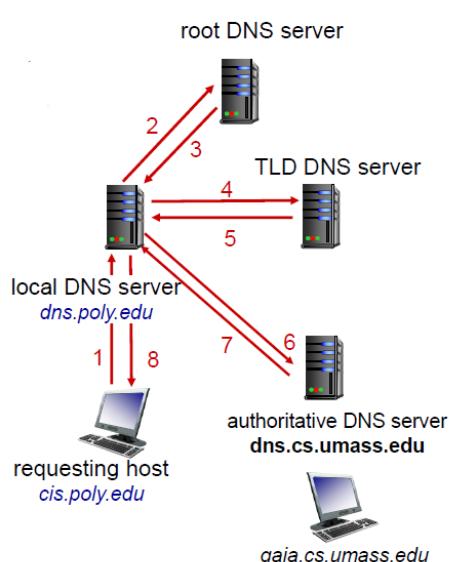
E' possibile manualmente un server DNS per farlo puntare a qualcosa' altro (magari direttamente a un Root Server) direttamente dal proprio dispositivo. Le richieste verranno quindi inoltrate al server stabilito.

Anche nel caso del caching è presente il problema dell'allineamento. Queste cache devono essere allineate tra loro e aggiornate (cioè le informazioni nel Local DNS devono essere identiche a quelle del DNS autoritativo). Esiste un meccanismo **RFC 2136** dove i DNS si scambiano tra loro delle informazioni per mantenere aggiornate le entry. Questo meccanismo però non viene mai utilizzato perché il problema non si pone, perché le informazioni legate al DNS non sono dinamiche.

Tra le varie informazioni contenute nel server c'è anche un parametro che indica il tempo di validità massimo di tale informazione.

Esempio: se google.com dovesse cambiare indirizzo IP si può andare nel server autoritativo e modificare il parametro di validità (rendendolo più breve) qualche giorno prima, così che all'effettivo cambiamento dell'indirizzo IP tale informazione viene immediatamente percepita dagli utenti (che avranno aggiornamenti della entry frequenti) e il sito può essere raggiungibile anche dopo il cambiamento di indirizzo IP.

Esempi di risoluzione



Esistono due modi per interagire con l'infrastruttura gerarchica. In entrambi i casi si utilizza un local DNS intermedio. Qualsiasi sistema operativo chiederà a qualcuno di risolvere il dominio (in questo esempio il DNS locale). Nell'esempio in foto "requesting host" vuole risolvere il dominio `gaia.cs.umass.edu` perciò il server locale, non avendo in cache la risposta va a chiedere al Root Server, che restituirà l'indirizzo IP del server responsabile del ".edu". Il local server andrà a fare la domanda al TLD DNS server che non avrà il mapping, ma restituirà l'indirizzo IP del server autoritativo per il dominio necessario e finalmente si andrà al server autoritativo per il dominio che avrà l'indirizzo IP per tale stringa.

Questo è un esempio di **query iterativa**.

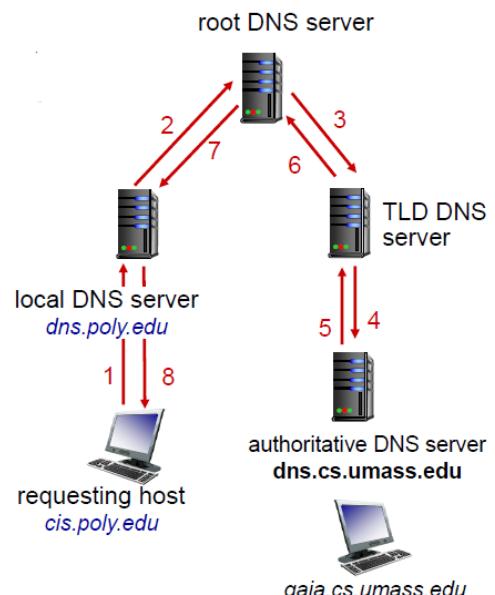
Esiste la versione **query ricorsiva**, dove si chiede al DNS successivo di diventare client.

In questo esempio, il local DNS chiede al root DNS ma quest'ultimo diventerà client ed effettuerà la richiesta al TLD DNS Server, che a sua volta chiederà al Server autoritativo, mandando indietro il mapping fino al local DNS (passando per tutti i server come in foto).

In questo modo c'è più carico gerarchico ai livelli più alto, ma essendo che il local DNS nel 90% risolve tramite le cache i domini, il carico non è poi così elevato.

Un vantaggio è che in questo modo tutti i server possono fare caching (nel caso precedente poteva farlo solo il local DNS).

In rete è possibile trovare anche un mixto delle due cose. Ogni server gestisce diversamente la richiesta.



In un dominio, non sempre l'aggiunta del “.” corrisponde al salire di un livello. Guardando l'esempio precedente si nota il passaggio dal server “.edu” al server “.cs.umass.edu” senza passare per quelli intermedi. Semplicemente, il server può essere autoritativo per “.cs.umass.edu” oppure in realtà il nome del dominio è “cs.umass” cioè non esiste un sottodominio dentro “umass” che si chiama “cs”. Il “.” è obbligatorio per dividere i domini, ma non sempre il “.” divide un dominio. In altre parole, è condizione sufficiente ma non necessaria.

Record DNS

Le informazioni contenute nei server DNS sono organizzate in **record** e possono avere vari tipi di record. Il più comune è quello di **tipo A** cioè un record composto da **nome e valore** di indirizzo IP. Cioè il formato generico è del tipo: **(name (key), value, type, ttl)**

dove il ttl è il valore citato prima della scadenza del campo. Nel record di tipo A la chiave (cioè il nome) è direttamente l'hostname e il valore è l'indirizzo IP.

Nel **tipo CNAME** il campo nome è un **alias** di un altro nome detto **canonico**, che è il vero nome associato a un indirizzo IP. Si utilizza questo tipo per indicare che il nome associato non è quello reale ma è un alias.

Esempio

In configurazione di tipo A, per definire più servizi per un dominio bisogna fare come segue:

A	www.polito.it	130.181.85.1
A	didattica.polito.it	130.181.83.2
A	mel.polito.it	130.181.85.1

Lo svantaggio risiede nel caso in cui bisogna cambiare indirizzo IP: è necessario modificare tutte le entry presenti. Si può quindi optare per una configurazione di tipo CNAME. Si avrà un'unica entry di tipo A col nome canonico della entry (dando un nome a piacere e non più www.polito.it) mentre tutte le altre entry avranno l'alias:

A	workHP.polito.it	130.181.85.1
CNAME	www.polito.it	workHP.polito.it
CNAME	didattica.polito.it	workHP.polito.it

In questo modo, in caso di cambio di indirizzo IP, basterà cambiare una sola entry.

Nel **tipo NS** il campo **nome** è il dominio, mentre il valore (value) è l'hostname del server autoritativo (cioè non è nient'altro che il “puntatore” citato prima). Quindi questo tipo viene utilizzato per poter scendere nella gerarchia. Si assuma di essere all'interno del TLD di .it e si supponga di voler definire “polito.it” dentro “.it”. Si dovrà dire a chi gestisce il dominio “.it” qual è il server in grado di gestire il mio dominio, perciò all'interno del TLD qualcuno dovrà scrivere all'interno il record:

NS	polito.it	leonardo.polito.it
A	leonardo.polito.it	130.181.3.24

che vuol dire che tutto il dominio “polito.it” è gestito dal server DNS “leonardo.polito.it” il cui indirizzo IP è 130.192.3.24

L'unico tipo che può avere come campo valore un indirizzo IP è il tipo A. Tutti gli altri non possono avere l'indirizzo IP.

L'ultimo tipo è il **tipo MX** che serve per trovare il **mailserver** associato a un certo dominio.

Tipo PTR: serve per le risoluzioni inverse, cioè nel caso in cui si voglia risalire dall'indirizzo IP al nome del dominio girando al contrario l'indirizzo IP e aggiungendo **.in_addr.arpa**. Esempio: se si vuole capire qual è il nome associato all'indirizzo 130.192.85.1 si farà una query del tipo **PTR per 1.85.192.130.in_addr.arpa** che è un dominio che gestisce tutte le risoluzioni inverse.

Il tipo AAAA associa un nome a un indirizzo IPv6.

Protocollo DNS e messaggi

Si tratta di una serie di **query** e **reply** con formato **identico**. All'interno è presente:

- **Identification:** per conoscere a cosa si sta facendo riferimento (a quale richiesta)
- **Flags:** identificano tra domanda o risposta; richiedono la ricorsione; esiste anche un flag che si setta a 1 se si ha una risposta autoritativa, cioè se la domanda non ha trovato informazioni in nessuna cache ed è arrivato fino al server autoritativo.

4. Livello trasporto

Il livello trasporto è necessario per poter capire, una volta ricevuti i pacchetti al destinatario, a quale applicazione tale pacchetto deve essere consegnato. Per differenziare le applicazioni, si introduce il concetto di **porta**. Questa operazione è detta **demultiplexing**, poiché il flusso di traffico omogeneo al livello 3 arriva per un unico host (unico IP destinatario) e deve essere demultiplexato alle applicazioni in base al numero di porta (qualcosa che IP non vede). Dal lato opposto ho il **multiplexing**, cioè tante applicazioni che inviano del traffico ed IP deve multiplexerlo, cioè uscirà un unico flusso di traffico con un unico indirizzo IP sorgente.

Questa funzionalità è obbligatoria perché altrimenti le reti sarebbero inutilizzabili. Il multiplexing è una funzionalità presente in tutti i protocolli di livello trasporto (UDP e TCP).

È possibile aggiungere funzionalità al livello trasporto: la rete IP è di tipo datagram e non orientata alla connessione, infatti è di tipo **best effort** (non dà garanzie). Per tali motivi è possibile garantire che il traffico arrivi a destinazione, implementando gli **ACK** che confermano la ricezione. In questo modo viene offerta **affidabilità** che il traffico arriva a destinazione (**reliable data transfer**). È possibile fare anche **controllo di flusso e controllo di congestione**.

Il controllo di flusso fa sì che il ricevitore **non** venga sovraccaricato (in caso di scambio dati tra due dispositivi con performance diverse, è quindi inutile inviare a 100Mb/s se il ricevitore riceve massimo 10Mb/s).

Il controllo di congestione fa sì che la rete **non** venga sovraccaricata (in caso di scambio dati tra due dispositivi con stesse performance, ma nella rete vi sono dei link con scarse prestazioni, è quindi inutile inviare a 100Mb/s anche se il ricevitore riceve a 100Mb/s se in rete vi sono link che hanno massimo 10Mb/s di scambio dati).

Il modo con cui si evitano questi problemi è **rallentando**. Per rallentare viene **ridotta la dimensione della finestra**.

UDP **non** offre queste funzionalità. Le offre soltanto TCP.

A questo livello, a differenza del livello 3 dove il controllo di errore avveniva solo nell'header per evitare sovraccarico dei router, viene effettuato un checksum sull'intero segmento (sia UDP che TCP).

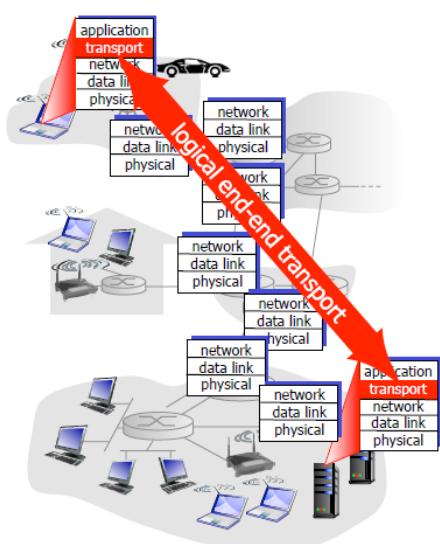
Per offrire questi servizi:

- UDP offre un servizio **non orientato alla connessione**, quindi invierà i dati senza preoccuparsi di aprire o chiudere connessioni
- TCP invece offre un servizio **orientato alla connessione**, aprendo e chiudendo le connessioni con gli host. Si informa quindi il ricevitore che si sta per inviare qualcosa. **Non** esistono circuiti virtuali creati sulla rete andando a riservare risorse sui router.

Il livello trasporto su un host prende i messaggi e li imbusta in **segmenti** (spezzando i messaggi in dimensioni opportune, ad esempio ethernet 1500 byte) e passandoli al livello rete. Nella foto a destra si nota che i router **non** entrano in gioco poiché non si occupano del livello 4.

Al ricevitore arrivano i segmenti, che vengono ricombinati nel flusso dati continuo che vengono poi passati al livello applicazioni.

I servizi che **non forniscono** TCP e UDP sono **garanzie sui ritardi e garanzie sulla banda**.



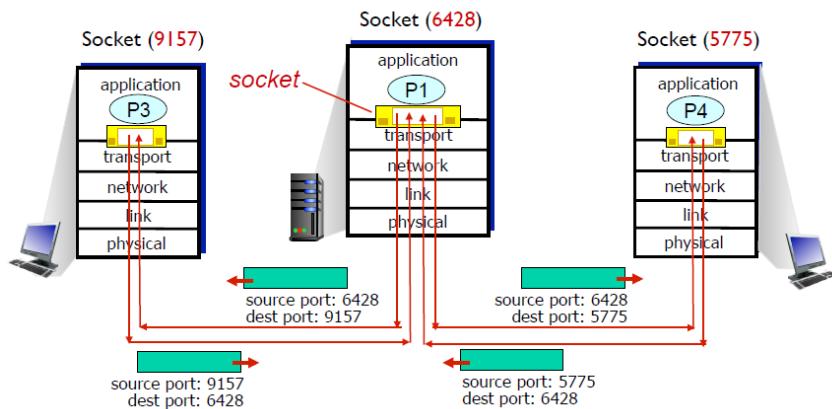
Multiplexing e Demultiplexing

Il multiplexing e il demultiplexing si basano sul numero di porta. In particolare, nel multiplexing ci si concentra sulla **porta sorgente** e nel demultiplexing sulla **porta destinazione**.

Riguardo al demultiplexing, la porta destinazione ci dice a quale applicazione bisogna inviare il pacchetto ricevuto.

Nel caso di UDP ciò è vero perché fa demultiplexing basandosi unicamente sulla porta destinazione del segmento (guarda la porta e capisce che il segmento è diretto verso una particolare applicazione) e lo manda su quella porta dove c'è un **processo in ascolto** su una certa porta. Ciò viene fatto tramite la **socket API** che sono il modo più comune per accedere alla rete. Un socket è l'end-point di una comunicazione a due vie di programmi che sono in ascolto nella rete.

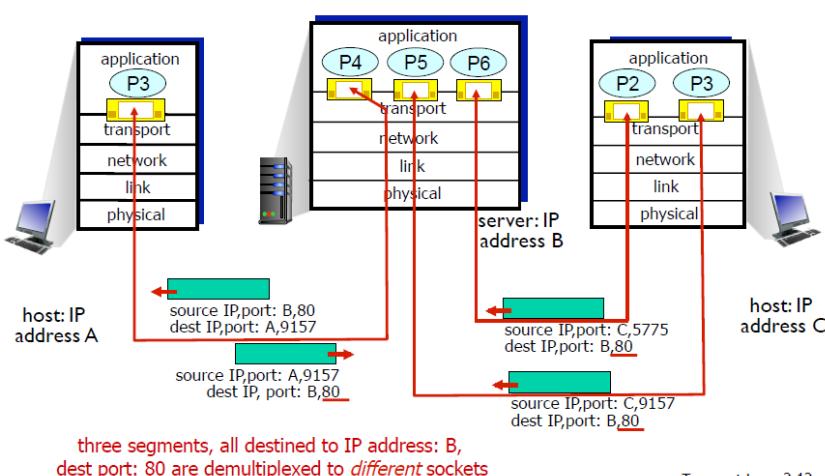
In UDP, essendo gestito solo tramite porta destinazione, viene configurato come segue:



Il demultiplexing presenta un **socket associato per ogni applicazione** (per ogni app, una porta) perciò se il processo P3, associato a una porta, vuole inviare del traffico a P1, lo farà puntando alla porta corrispondente associata al socket che è associato al processo in esecuzione.

Se si considera TCP, ciò non basta poiché quest'ultimo sfrutta:

- **Indirizzo IP sorgente**
- **Porta sorgente**
- **Indirizzo IP destinazione**
- **Porta destinazione**



Transport Layer 3-13

Sostanzialmente sono gli elementi che caratterizzano un singolo flusso di traffico. Ogni flusso di traffico avrà **almeno 1** elemento tra questi diverso. Per implementare l'affidabilità, essa viene offerta tramite **acknowledge**, cioè sul destinatario bisogna tenere traccia degli ACK già inviati e della finestra di ricezione per **quella singola applicazione** (tramite delle variabili) e perciò è necessario utilizzare tutti e 4 gli elementi.

Osservando la figura precedente, anche se in TCP si ha **una singola applicazione**, sarà necessario avere più istanze della stessa applicazione con socket diversi. Tutti i processi (P2 e P3) puntano alla stessa porta 80, perciò bisogna distinguerli opportunamente e i socket sono evidentemente più complessi.

Porte note

Per poter scambiare correttamente i dati tra le applicazioni, è stato definito lo standard delle **well-known ports** dove sono stati stabiliti dei valori specifici per determinate applicazioni. Ad esempio, la porta TCP 80 è quella dell'HTTP. Questo fa capire che i server in grado di parlare il linguaggio HTTP devono essere messi in ascolto sulla porta 80.

I server mail saranno in ascolto col protocollo SMTP sulla porta 25.

I server DNS sulla porta UDP 53.

Non esiste alcun controllo sulle porte utilizzate. Si può quindi modificare una porta, ma non rispettando tale standard il resto del mondo non riuscirà a raggiungere tale applicazione che non ha la porta standard, ma un'altra. Per poterlo raggiungere bisognerà specificarlo.

Con la porta sorgente **non** serve uno standard, poiché bisogna soltanto inviare e non ricevere. Questo perché chi riceve leggerà la porta utilizzata e può usarla per rispondere. Per questo motivo le porte sorgente sono decise in maniera **randomica**.

Il numero di porte massimo è 2^{16} .

UDP

UDP è molto semplice perché in sostanza non migliora il servizio best-effort di IP. Possiede un **checksum** sull'intero segmento e non fa nient'altro.

UDP non ha ACK, non offre garanzie e non fa controllo di flusso/congestione.

UDP è meglio da utilizzare durante le applicazioni di streaming multimediale, poiché non si hanno i ritardi dovuti alle ritrasmissioni e il controllo di flusso/congestione che abbassa il rate di trasmissione.

Se non si ha un ACK al flusso di dati però, alcuni dati possono andare persi.

Ad esempio durante una telefonata ciò è tollerabile, perché anche se alcune parole vengono perse si può comunque capire il contenuto del messaggio oppure si può chiedere di ripetere all'interlocutore.

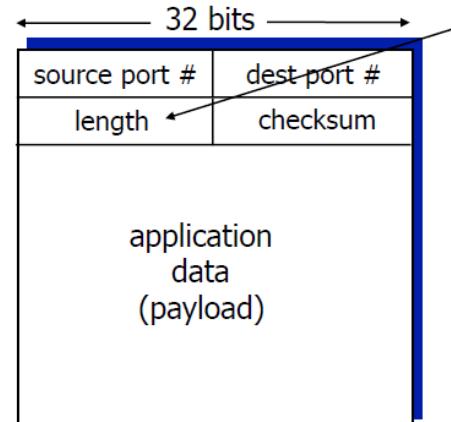
Nel caso dello streaming, si sacrifica la qualità dei dati (perdendo quindi qualcosa) per evitare il controllo di flusso e congestione (per evitare i ritardi).

UDP viene utilizzato anche nel DNS e si preferisce a TCP perché non è necessario aprire la connessione definendo tutti i parametri, visto che verrà inviato un unico pacchetto (DNS Query). Non ha senso utilizzare un protocollo a finestra per inviare un unico pacchetto.

Se si invia una DNS Query si ha soltanto la DNS response, perciò questo pacchetto di risposta viene utilizzato come ACK.

L'affidabilità viene quindi gestita direttamente nell'applicazione. È il server DNS che fa partire un timer all'invio della risposta e se non si riceve risposta entro lo scadere del timer, la DNS query viene inviata nuovamente.

L'header è molto piccolo (4 byte) poiché non serve nient'altro.



UDP segment format

TCP

La principale differenza tra TCP e UDP è l'**affidabilità**. TCP implementa determinate soluzioni basate su protocolli a finestra per cercare di offrire su un canale non affidabile, servizi affidabili alle applicazioni che stanno al livello superiore. Ricapitolo protocolli pipeline:

- **Go-back-N:** nella pipeline (finestra di trasmissione) si hanno fino a N pacchetti non ancora riscontrati. Il ricevitore manda solo **ACK cumulativi** cioè (non vuol dire che se ricevo un pacchetto non mando nulla, un secondo pacchetto e non mando nulla e al terzo pacchetto mando un ACK, questo è un ACK ritardato) **dopo ogni pacchetto** invierò un ACK, che vale anche per i pacchetti ricevuti in precedenza. Quando si hanno N pacchetti in attesa di ACK si ha un unico timer (relativo al pacchetto più vecchio) che quando scade invia nuovamente tutta la finestra di pacchetti.
- **Selective repeat:** il mittente può avere fino ad N pacchetti in attesa di ACK nella pipeline e il ricevitore invia **ACK individuali per ogni pacchetto**. Dentro l'ACK cioè si riferisce al singolo pacchetto e non vuol dire che si è ricevuto "fino a N". Il ricevitore mantiene un timer per **ogni pacchetto riscontrato**.

TCP è un protocollo punto-punto, cioè vuol dire che si ha un'unica sorgente ed un'unica destinazione (comunicazione unicast a livello 3). Le comunicazioni multicast non sono permesse con TCP poiché bisogna mantenere le variabili di stato relative alle singole comunicazioni tra due punti. Una comunicazione multicast IP dovrà avere UDP come livello trasporto.

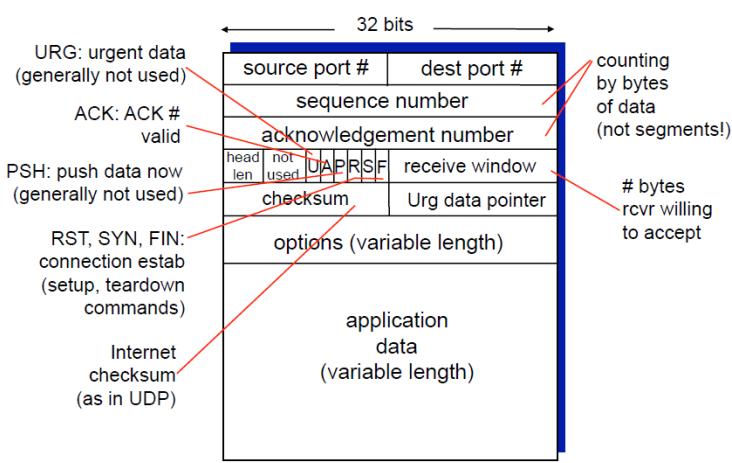
TCP offre un servizio affidabile basato su uno stream di byte consegnato **in ordine** (poiché usa un protocollo a finestra). Ciò vuol dire che TCP riceve dal piano superiore i byte senza dei confini di messaggio ed è TCP che crea i segmenti e si occupa di consegnarli in ordine.

E' **pipelined** e il controllo di congestione e di flusso definiscono la dimensione della finestra.

TCP crea uno scambio dati **full-duplex**: si crea una connessione full-duplex in cui su un canale logico si invieranno le richieste e su un altro le risposte. Questo canale logico non ha nessun impatto sulla rete, poiché i router **non** vedono questo canale logico (perché ci si mette solo d'accordo sui parametri).

Se la MTU è 1500 byte, la MSS (maximum segment size) sarà di 1500 byte. Questo è il parametro della connessione e non si avranno segmenti di dimensione superiore.

TCP è **orientato alla connessione** poiché si effettua l'**handshaking** tra ricevitore e trasmettitore (cioè ci si mette d'accordo sui parametri).



L'header TCP è formato da 20 byte (5 righe + campo opzioni).

Il numero di sequenza e di ACK sono i campi che servono per contare i segmenti e per riscontrare i segmenti ricevuti (equivalenti ai campi dei protocolli a finestra).

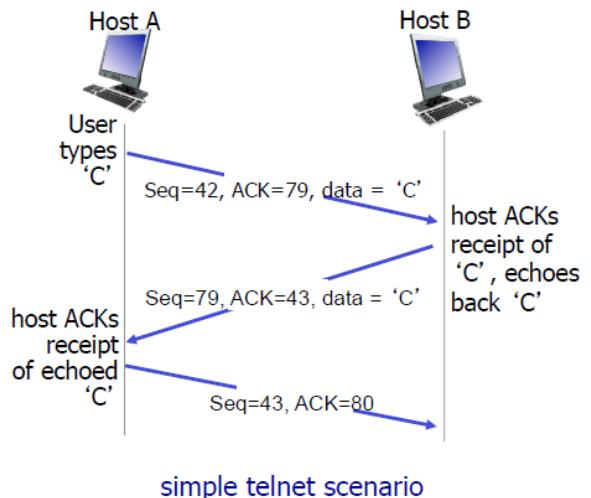
Successivamente vi sono dei flag: RST, FIN, SYN servono per aprire e chiudere la connessione.

La receive window serve per il controllo di flusso.

Il checksum è equivalente a quello di UDP (su header e payload).

Le opzioni sono poche e non se ne vedono quasi mai. Soltanto quando si apre una connessione, dentro le opzioni di TCP c'è il MSS. Questo è inoltre l'unità minima di dimensione della finestra (può crescere o decrescere di N MSS).

- **Sequence number:** rispetto ai protocolli a finestra dove negli esercizi si inseriscono dei numeri che indicano i pacchetti, in TCP non vengono contati i segmenti ma i **byte**. Il numero di sequenza rappresenta il primo byte contenuto dentro il segmento all'interno dello stream di byte totale. In caso di stream di 4000 byte e MSS di 500 byte, supponendo che il S.N. parta da 0, si avrà: 1° segmento: 0 (cioè invio da 0 a 499); 2° segmento: 500 (da 500 a 999); 3° segmento: 1000; e così via. In realtà il S.N. è scelto randomicamente perciò non parte mai da 0.
- **L'acknowledge number:** è **cumulativo** ma non dice qual è l'ultimo byte ricevuto, ma dice qual è **il prossimo byte che si aspetta di ricevere**. TCP utilizza una versione modificata del Go-Back-N. Quest'ultimo non gestisce i pacchetti fuori sequenza, poiché li scarta visto che la finestra viene nuovamente inviata. In TCP nello standard non c'è scritto che la finestra deve essere di dimensione 1, ma è lasciato all'implementatore decidere come gestirli. Tutti i sistemi operativi progettano TCP per gestire i pacchetti fuori sequenza, perciò la finestra è maggiore di 1. Un utente sull'Host A schiaccia "C" e deve trasferire il byte all'host B. Il numero di sequenza è randomicamente partito da 42 (manda il byte 42) e l'ACK è 79 per lo stesso motivo di Seq. Il payload conterrà 1 byte, cioè il carattere C. B riceve il segmento e deve farne l'Acknowledge (e l'eco, visto che è attivato da telnet) perciò l'ACK sarà 43 (42+1 byte) cioè B si aspetta di ricevere il byte 43. Come seq.num. c'è il 79 per via dell'ACK precedente. Il payload è ancora una volta un byte. A invia l'ACK (che è l'80 perché ha ricevuto il 79°) e come seq mette 43 perché B ha chiesto il 43. Quest'ultimo ACK non ha payload. Visto che il payload è vuoto, il messaggio successivo di A verso B sarà con Seq=43. Inviare un ACK insieme a del payload vuol dire fare **piggybacking**.



TCP e la trasmissione affidabile

TCP realizza un servizio affidabile sopra una rete non affidabile tramite:

- Pipeline di segmenti
- Acknowledge cumulativi
- Singolo timer di ritrasmissione

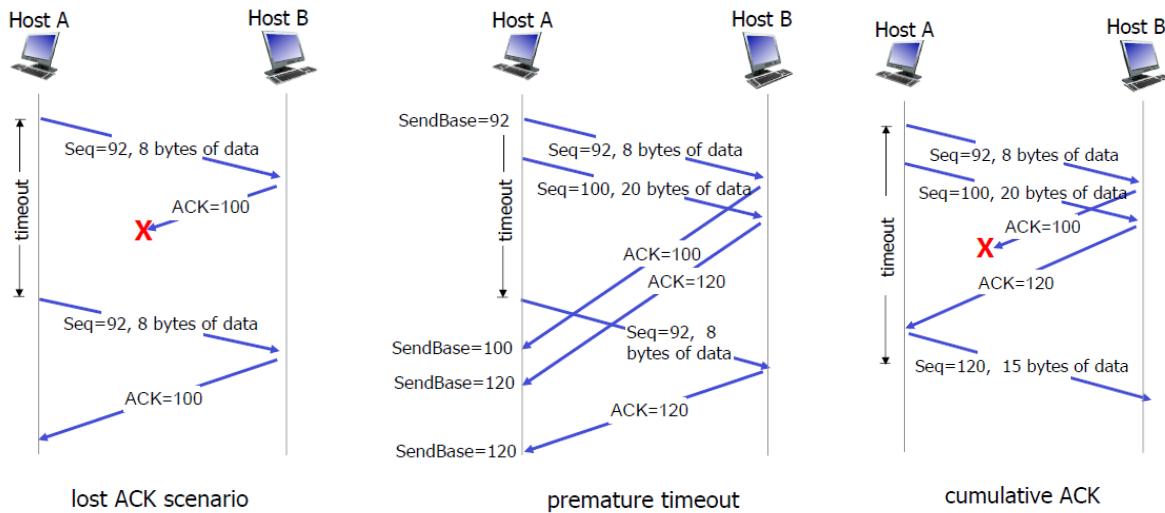
Le trasmissioni possono essere triggerate da due eventi: il primo è il timeout, ovvero ad ogni timeout bisogna ritrasmettere. Il secondo evento sono gli ACK duplicati. Per iniziare verrà fissata una finestra (che non verrà variata dai controllo di flusso/congestione e ignoriamo gli ACK duplicati).

Con tali premesse, un trasmettitore TCP:

- Riceve dal livello applicazione lo stream di byte e costruisce i segmenti utilizzando dei numeri di sequenza.
- Se non c'è già un timer attivo, ne fa partire uno. Il timer può fare riferimento al più vecchio pacchetto non ancora riscontrato.
- Quando scatta un timeout **viene ritrasmesso solo il segmento che ha causato il timeout** (che sembra selective repeat). Non inviare nuovamente tutta la finestra è problematico se **non** si sono gestiti i pacchetti fuori sequenza. Nella realtà non si hanno troppe perdite, poiché quando la rete

inizierà a essere congestionata interverrà il controllo di congestione perciò non ha senso re inviare ogni volta tutta la finestra. Una volta ritrasmesso il pacchetto che ha fatto scattare il timeout, il timer deve ripartire e sarà di nuovo relativo al pacchetto più vecchio (identico al timer precedente poiché quello è il pacchetto non ancora riscontrato).

- Alla ricezione degli ACK il timer relativo viene annullato e bisogna far partire un nuovo timer se ci sono ancora ACK da riscontrare. Tutto funziona perché le perdite sono molto poche.



Ricevitore TCP

Il ricevitore TCP è in grado di gestire gli **ACK ritardati**. Quando al ricevitore si verifica l'arrivo in sequenza di un segmento con il corretto numero di sequenza e tutto funziona alla perfezione, il ricevitore può decidere di non inviare subito l'ACK. Si aspetta di ricevere un altro pacchetto e si invierà l'ACK alla ricezione di tale pacchetto, ed essendo cumulativo verrà confermato anche il pacchetto precedente.

È quindi un ACK **cumulativo ma anche ritardato**.

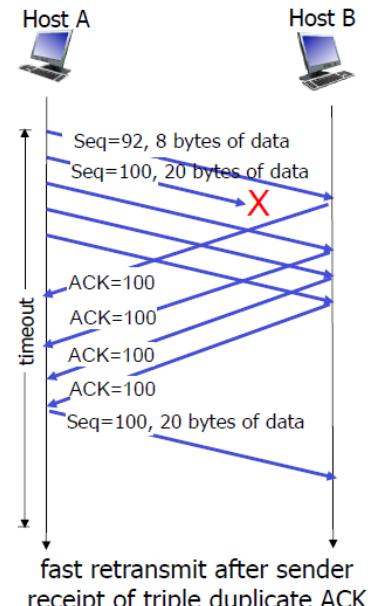
Inoltre, se tutto sta funzionando e si hanno arrivi in ordine tranne l'ultimo segmento (cioè si ha già un pacchetto in attesa di riscontro per via della regola precedente) si manda immediatamente l'ACK.

In generale se ne salta 1 ogni 2 ACK.

Se invece arriva un pacchetto fuori ordine (quindi con numero di sequenza maggiore di quello atteso) il ricevitore invia un **ACK duplicato**, indicando il numero di sequenza del byte successivo atteso. Infine, se il trasmettitore invia un segmento che parzialmente o completamente riempie il "buco" di pacchetti fuori ordine, il ricevitore invia l'ACK relativo al pacchetto ricevuto in sequenza e invierà il numero di sequenza relativo al byte successivo che ci si aspetta di ricevere. Questi ACK duplicati possono essere utilizzati come indicazione del fatto che c'è stata una perdita (a differenza dei timeout). Il fatto di ricevere tanti ACK duplicati può essere indice del fatto che c'è stata una perdita.

Se un ACK viene duplicato più volte e si riceve 3 ACK duplicati si intuisce che c'è stato un problema e viene inviato nuovamente il pacchetto relativo. Questo si chiama **TCP fast retransmit**.

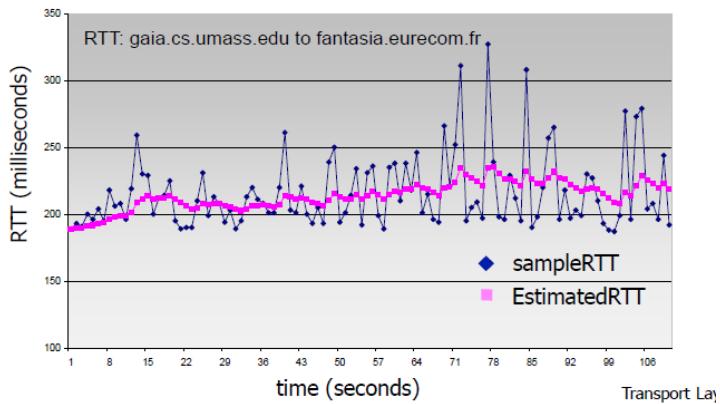
RTT: il valore ottimale in un protocollo a finestra è 2 volte il tempo di propagazione. In generale il timeout del timer TCP è più lungo del RTT ma tale valore varia. Per poterlo stimare si può utilizzare il metodo



SampleRTT: misura il tempo di trasmissione tra l'invio del segmento e la ricezione dell'ACK.

$$\text{EstimatedRTT} = (1 - \alpha) * \text{EstimatedRTT} + \alpha * \text{SampleRTT}$$

- ❖ exponential weighted moving average
- ❖ influence of past sample decreases exponentially fast
- ❖ typical value: $\alpha = 0.125$



Nel tempo il RTT varia in base alla congestione della rete. È difficile dunque stimare il RTT con cui inizializzare il timeout.

La stima utilizzata è quella dell'EstimatedRTT in foto. Si prende il valore passato della stima al tempo $t-1$ e lo si media col valore misurato al tempo T aggiungendo i pesi (α e $1 - \alpha$). Se $\alpha=0$ si guarda solo il passato, se $= 1$ solo il presente. Tipicamente il valore è 0.125.

Si può anche utilizzare la deviazione standard per stimare il RTT.

Controllo di flusso

Il controllo di flusso è la capacità del trasmettitore di non sovraccaricare il ricevitore. Per realizzarlo, bisogna aggiungere qualche dettaglio al concetto di **socket**.

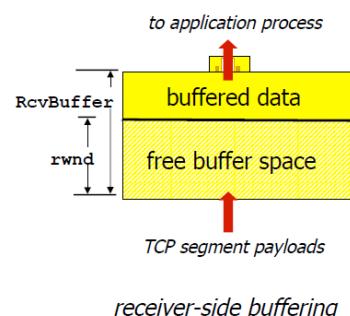
La ricezione del traffico in un SO funziona come segue: i frame arrivano, si spacchettano fino a TCP ma quest'ultimo non li invia direttamente all'applicazione, ma stanno in un buffer e successivamente è l'applicazione che preleva i dati dal buffer e li va ad utilizzare.

Arrivando traffico dalla rete in questo buffer, se l'applicazione fatica a leggerli, il buffer cresce fino ad arrivare in overflow (termina la memoria) e i byte ricevuti non hanno più spazio per essere depositati.

Il controllo di flusso evita di far riempire il buffer.

Il buffer del ricevitore è composto da una parte con i dati ricevuti in attesa di essere letti ed una parte vuota che attende i nuovi segmenti. La parte vuota corrisponde alla **finestra di ricezione**. Tale dimensione potrebbe decrescere se l'applicazione non riesce a star dietro alla velocità con cui i pacchetti arrivano nel buffer.

rwnd: receiver window.



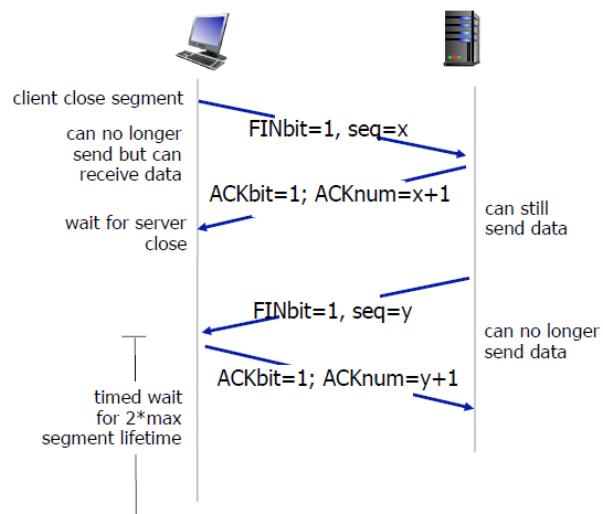
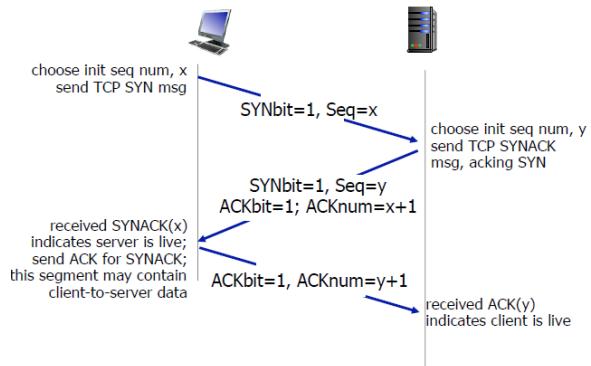
Per implementare il controllo di flusso, bisogna comunicare al trasmettitore qual è il valore della rwnd. Se il trasmettitore fissa il valore della sua finestra di trasmissione pari alla rwnd, il buffer non andrà mai in overflow. Il valore della rwnd viene inserito come campo nell'header TCP (tipicamente negli ACK che il ricevitore invia al trasmettitore). Il trasmettitore setterà la finestra di trasmissione ad un valore pari al valore di rwnd.

Management connection: per aprire o chiudere una connessione il trasmettitore e il ricevitore si mettono d'accordo tramite l'**handshaking**. Ci si mette d'accordo sui parametri della connessione e sul fatto che si voglia stabilire una connessione. Per aprire la connessione si utilizza il flag **SYN**. Il trasmettitore invia un segmento vuoto al ricevitore in cui si setta a 1 il SYN bit e si inizializza il numero di sequenza ad un certo valore. Il ricevitore invia l'ACK relativo al segmento inviato (con SYN bit a 1) incrementando di 1 il numero di ACK. La procedura prevede che in piggybacking il ricevitore dica al trasmettitore che vuole aprire a sua volta una connessione, per questi motivi il SYN bit è impostato a 1 sull'ACK. Comunicherà il suo numero di sequenza, scelto dal ricevitore (in foto y). Il trasmettitore risponderà con un ACK scrivendo anche in questo caso y+1 nell'ACK number.

Questo metodo viene detto **3-way handshake**. Viene scambiato anche il valore del **MSS** poiché è l'unità di variazione della finestra di trasmissione.

Per chiudere la connessione si utilizza il flag **FIN**. In questo caso x non è scelto randomicamente, ma è l'ultimo numero di sequenza (di N segmenti inviati). Il ricevitore lo riscontra e di nuovo per convenzione riscontra con ACK number di x+1. Anche il ricevitore a sua volta per convenzione vorrà chiudere la convenzione (perché magari anche se A ha chiuso la connessione, B deve ancora inviare dei pacchetti), ed invierà un segmento con FIN bit a 1 a sua volta.

Il trasmettitore risponderà con un ACK con ACK number y+1. Il trasmettitore fa partire in questo istante un timer, durante il quale il trasmettitore può ricevere ancora segmenti dal ricevitore (magari se qualche pacchetto è in ritardo sulla rete).



Controllo di congestione

Il controllo di congestione si occupa di evitare di far riempire il buffer dei vari router della rete.

La congestione è presente quando troppe sorgenti inviano troppo traffico molto velocemente per la rete.

La congestione si manifesta con la perdita di pacchetti o con lunghi ritardi di trasmissione. Il router inizia ad accodare i pacchetti all'interno del proprio buffer provocando ritardi, fino a perderli se il buffer è pieno.

Per effettuare controllo di congestione esistono due modi:

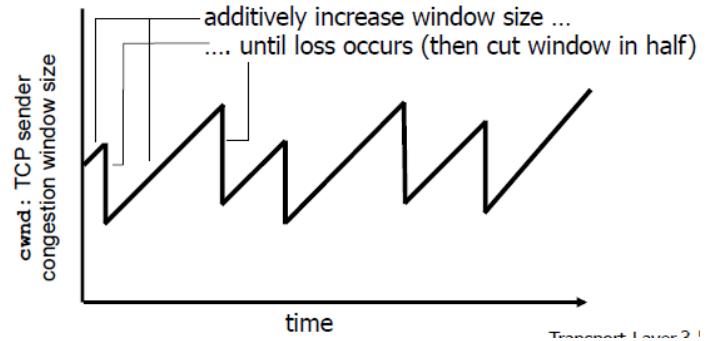
- **Controllo di congestione end-end:** si confina la complessità agli host e si reagisce senza avere alcun feedback dalla rete. I timeout sono un buon indicatore della qualità della rete, ma anche gli ACK duplicati. TCP reagisce sulla base degli eventi di timeout e ACK duplicati senza chiedere informazioni alla rete.
- **Controllo di congestione network-assisted:** i router offrono un feedback ai sistemi connessi in rete, ma ciò ne aumenta la complessità e infatti non viene mai implementato poiché va contro i principi di internet.

TCP utilizza il primo metodo per il controllo di congestione.

Un primo algoritmo che TCP utilizza è **AIMD** (**Additive increase, multiplicative decrease**) e lo si usa nella fase di **Congestion Avoidance**.

Il trasmettitore incrementa la finestra di trasmissione **linearmente** (da qui additive increase) incrementando di 1 MSS ad ogni RTT la finestra di congestione (cwnd) e per fare ciò si incrementa la finestra di $\frac{1}{cwnd}$ per ogni ACK ricevuto e ciò viene fatto fino a quando non si registra una perdita.

Una volta riscontrata una perdita, il **multiplicative decrease** taglia di $\frac{1}{2}$ la cwnd.



La rwnd è un buffer del socket e la $twnd \leq rwnd$. Nei router è presente però un buffer per ognuno e, non potendo conoscere lo stato dei singoli buffer si modella la rete con un ricevitore fittizio accanto al trasmettitore, che serve per “modellare” tutti i router che stanno in rete ma **non esiste fisicamente**.

Per tali motivi la **cwnd** è la finestra di ricezione di questo ricevitore fittizio. La cwnd è quindi un valore che serve per evitare di entrare in congestione ed è una stima della congestione dei router.

Avendo introdotto questa variabile, la funzione diventa:

$$t wnd \leq \min \{c wnd, r wnd\}$$

Se la rete è molto performante ma si invia traffico a un piccolo sensore, la cwnd tende a infinito ma la rwnd tende a 0 perciò vincerà la rwnd. Se invece si hanno due host potenti ma una rete molto vecchia, vince cwnd perché tenderà a 0 mentre rwnd tenderà a infinito.

TCP Slow Start

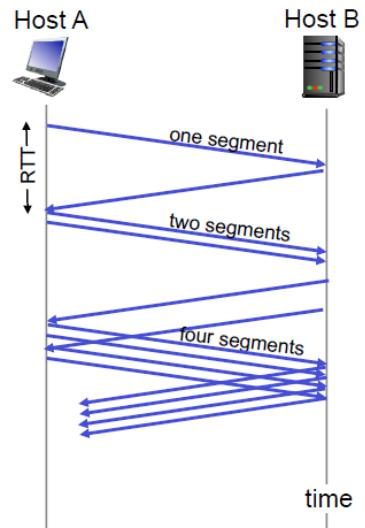
La congestion avoidance va bene se si è vicino a una situazione di potenziale congestione, ma il problema è all'inizio poiché non si sa nulla dello stato della rete. È necessario inizializzare la rete a un valore tale da evitare di trovarsi subito in congestione. Per fare ciò, siccome non si può sapere lo stato della rete, si utilizza l'approccio conservatorio di inizializzare la $c wnd = 1 MSS$. Si crescerà però in maniera esponenziale, andando a raddoppiare la $c wnd$ ad ogni RTT. Per fare ciò si va ad aggiungere 1 MSS ad ogni ACK ricevuto. Questo funziona bene all'inizio, ma dopo un certo momento bisogna tornare al congestion avoidance per evitare di crescere esponenzialmente fino a finire all'interno di una situazione di congestione.

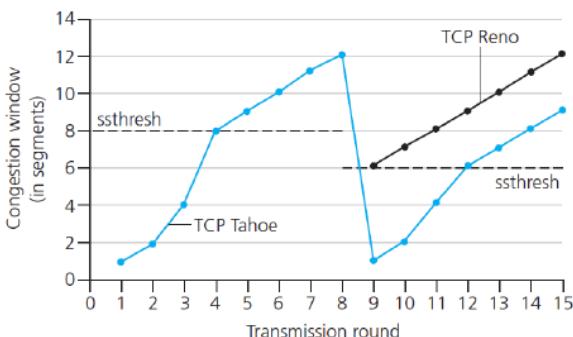
Gli eventi che fanno decrescere il valore della finestra possono essere il timeout e la ricezione di ACK duplicati. Nell'uno o nell'altro caso è possibile reagire diversamente. Se vi è stato un timeout è perché non si è ricevuto nessun ACK. Se si ricevono ACK duplicati ci si accorge della perdita prima dello scadere del timeout.

Se si ricevono gli ACK duplicati è perché la rete sta funzionando piuttosto bene. Se invece si è verificato un timeout è perché probabilmente si sono persi anche i pacchetti successivi che non hanno creato ACK duplicati e la situazione è più grave.

TCP Tahoe non faceva distinzione tra i due eventi, rimandando la cwnd a 1 in entrambi i casi.

Le versioni successive invece ottimizzano la cosa distinguendo i due casi: nel caso di ACK duplicati allora si resta nella fase di congestion avoidance tagliando a metà il valore della cwnd, mentre nel caso di timeout si ricomincia con cwnd pari a 1.



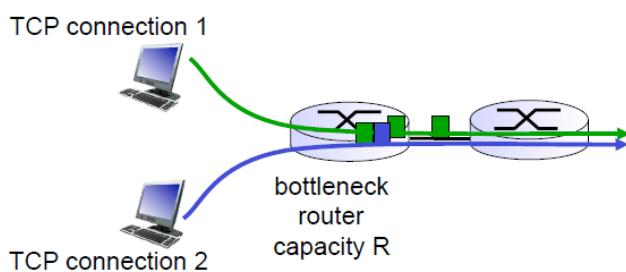


Dal grafico si nota quindi che si cresce in low start esponenzialmente fino a un valore di soglia e poi si inizia a crescere linearmente. La prima spezzata rappresenta TCP Tahoe che in entrambi i casi decrementava fino a cwnd pari a 1. TCP Reno utilizza il multiplicative decrease.

La soglia detta **ssthresh** è settata a metà della cwnd calcolata appena prima dell'evento di perdita. Se a 12 si ha una perdita, la soglia viene messa a 6. Quindi si torna a 1, si sale esponenzialmente fino a 6 e poi si riprende linearmente.

Il valore di soglia all'inizio non è possibile conoscerlo, perciò si cresce sempre esponenzialmente e "si spera". Nel momento in cui, crescendo esponenzialmente, si arriva a una perdita si spera di tornare in una situazione "sicura" dimezzando la finestra (e non tornando a 1). Da qui in poi si imposta la variabile ssthresh.

TCP Fairness: conseguenza degli algoritmi. Si supponga di avere lo scenario in figura

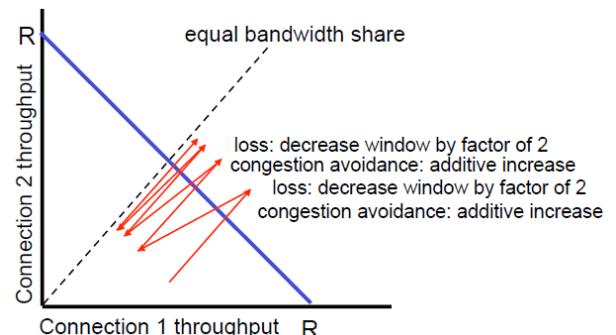


Si supponga di avere due connessioni TCP che vogliono inviare traffico sul link che si suppone avere capacità di 1 Mb/s. Entrambe le connessioni vogliono inviare 4 GB di dati, anch'essi con link a 1 Mb/s. Il ricevitore TCP sui dispositivi sarà in grado di inviare fino a 1 Mb/s. Essendo il terzo link a 1 Mb/s si avrà il problema del **queueing** ed entrambi i trasmettitori andranno ad abbassare la propria finestra di trasmissione andando a condividere entrambi lo stesso link. La banda viene quindi suddivisa, ma in che modo?

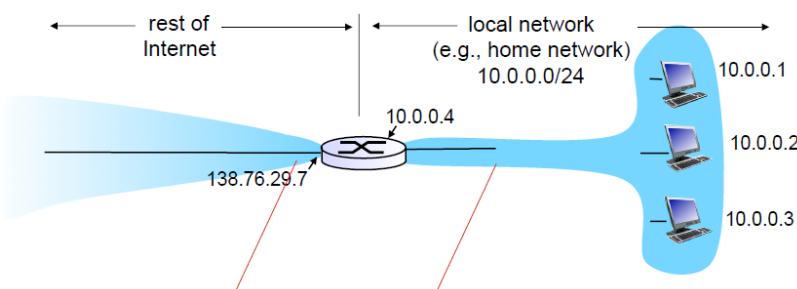
TCP è **fair**, cioè dopo un periodo transitorio la banda verrà suddivisa **equamente** e si avranno due flussi da 0.5 Mb/s ciascuno.

Guardando il grafico si nota che inizialmente R1 ha un throughput maggiore di R2 perché magari era già in connessione da prima con una finestra maggiore. Si ha quindi una crescita lineare su entrambe le connessioni finché non si ha congestione e si dimezza il valore di entrambe le cwnd. Si nota quindi che via via ci si avvicina alla linea tratteggiata, che rappresenta una situazione di divisione **equa** dove R1 ed R2 sono uguali.

UDP invece **non è fair** e prevarrà su TCP.



5. Indirizzamento privato



Gli indirizzi privati risalgono fin dallo standard originale. Gli indirizzi privati sono divisi in classi, come segue:

all datagrams **leaving** local network have **same** single source NAT IP address: 138.76.29.7, different source port numbers

datagrams with source or destination in this network have 10.0.0.0/24 address for source, destination (as usual)

10.0.0.0 - 10.255.255.255	1 class A network
172.16.0.0 - 172.31.255.255	16 class B networks
192.168.0.0 - 192.168.255.255	256 class C networks

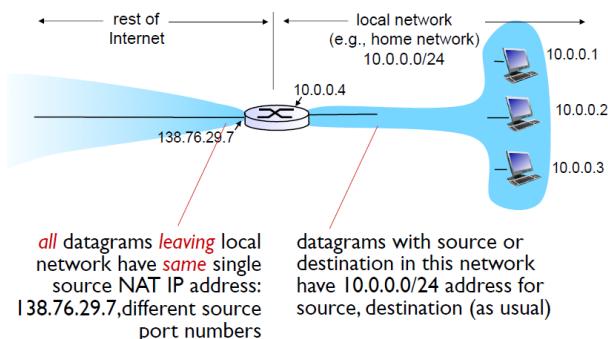
Le reti di tipo privato nascono dall'idea di voler creare una rete chiusa, cioè permettere a una cerchia di dispositivi di scambiare dati (senza comunicare con l'esterno). Quindi non ha senso in questo caso utilizzare degli indirizzi IP pubblici poiché bisognerebbe comprarli. Nel caso di rete privata serve soltanto avere un indirizzo diverso da tutti gli altri, e non un indirizzo IP pubblico per ogni utente.

La rete 10.0.0.0 era stata definita come una rete di classe A privata. Oggi, non essendoci più le classi viene definita come 10.0.0.0/8. La stessa cosa vale per 172.16.0.0/16.

Il tutto viene gestito evitando di instradare pacchetti su internet a questi indirizzi (poiché esisterebbero più punti in cui si accede allo stesso indirizzo IP). Perciò non saranno mai presenti sulle routing table dei router **pubblici**.

Ad esempio, nella rete interna del politecnico vengono utilizzate spesso, poiché i router interni sono in grado di inoltrare correttamente tali indirizzi IP poiché sono all'interno del politecnico, ma non fuori.

Il grosso passo è stato quello del **NAT (Network Address Translator)** che permette di connettere una **rete privata a una rete pubblica**.

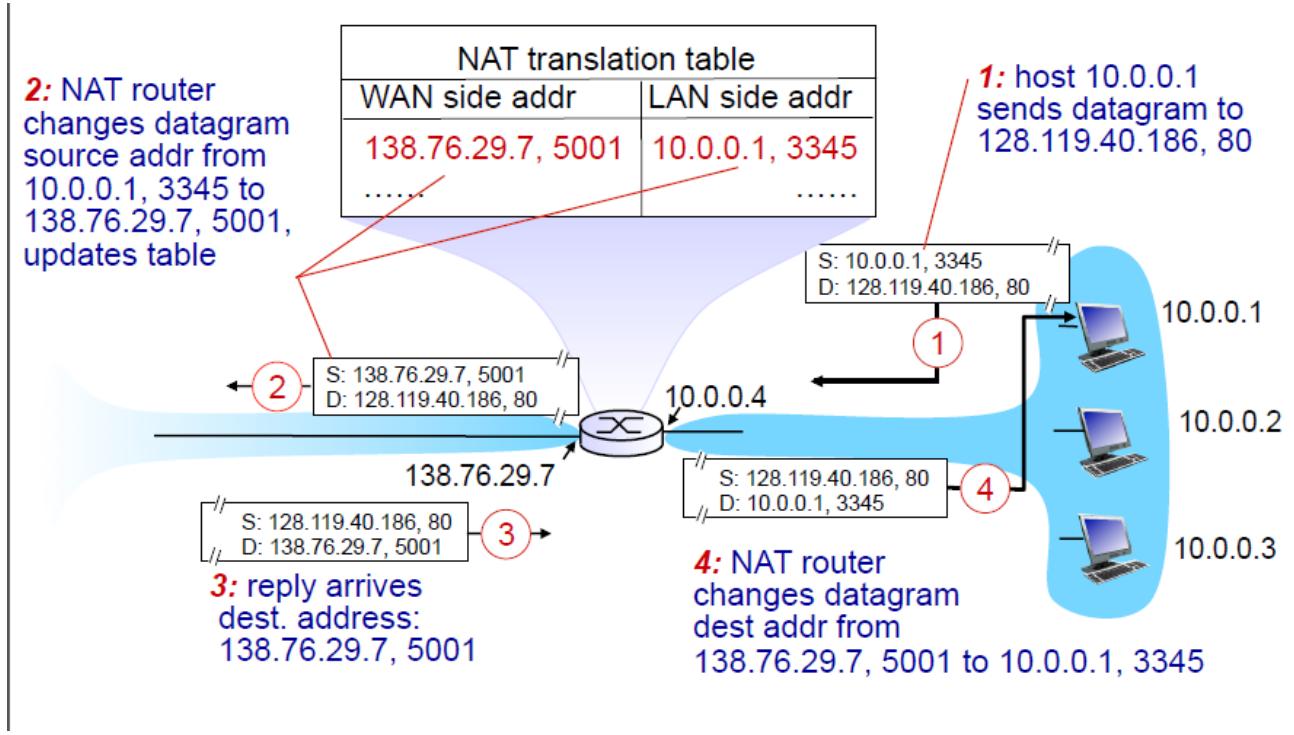


Il concetto è semplice. Si supponga che l'host 10.0.0.1 (privato) voglia raggiungere il sito google.com. Non vi sono problemi, poiché il router guarda soltanto l'indirizzo IP destinazione perciò il pacchetto arriverà tranquillamente al server. I problemi si hanno quando il server deve rispondere, perché dovrà leggere l'indirizzo IP sorgente. La particolarità del NAT è quella di sostituire l'indirizzo privato dell'host con l'indirizzo IP pubblico di quella rete (configurato sull'interfaccia esterna del router).

Per raggiungere la vera sorgente bisogna aggiungere sul router una tabella che ricorda da chi viene usato attualmente l'indirizzo IP pubblico, così appena il traffico torna indietro si ha corrispondenza tra l'indirizzo IP pubblico e privato.

Questa è solo l'idea di base, i nuovi NAT sono progettati per funzionare meglio e in contemporanea.

Una soluzione è basata sull'utilizzo delle **porte**.



Si suppone sempre che l'host 10.0.0.1 voglia inviare un pacchetto all'indirizzo 128.119.40.186 raggiungibile tramite la porta 80. L'host sorgente utilizzerà una porta scelta randomicamente (3345).

Il router intermedio, quando il pacchetto arriva, va a sostituire l'indirizzo IP sorgente con l'indirizzo IP pubblico, ma sostituisce anche la porta sorgente. Invece della porta 3345 viene utilizzata la porta 5001 su cui il router ha il controllo.

Quando il traffico tornerà indietro, sarà diretto proprio all'indirizzo IP pubblico e alla porta scelta dal router, ma quest'ultimo saprà che tale combinazione era destinata all'host con porta 3345, perciò sostituisce nuovamente i dati nel pacchetto e viene consegnato.

Se si suppone che anche 10.0.0.2 vuole comunicare allo stesso indirizzo IP con la stessa porta 80, vi saranno due diverse entry nel router. Il secondo host avrà una porta diversa dal primo (invece di 5001 avrà 5002).

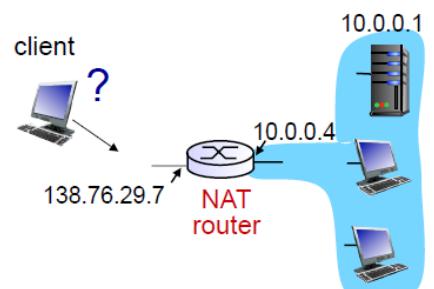
Tutto ciò funziona finché qualcuno dall'interno vuole aprire una sessione verso un target esterno.

Problema del NAT traversal (parente di firewall traversal)

Una prima soluzione banale è basata sulla configurazione statica della tabella sul NAT. Scrivo sulla entry la riga che manca per permettere a un host esterno di entrare nella rete. Ciò viene fatto manualmente e viene detto **port forwarding**.

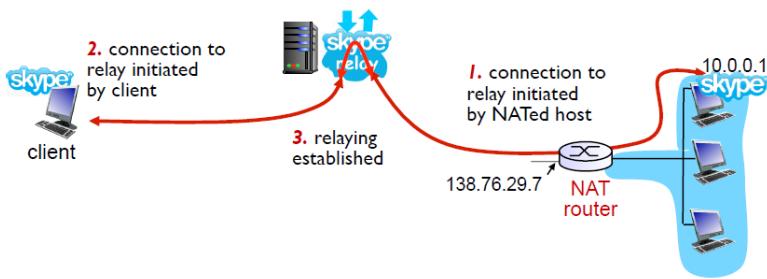
Se si volesse installare un server web nella rete di casa, bisognerebbe assegnare a tale host un indirizzo IP privato. Per accedere dall'esterno al momento non è possibile. Per renderlo possibile, si va a scrivere sulla tabella del NAT che "in presenza di traffico all'indirizzo IP pubblico del modem alla porta 80 deve essere inviato all'host 10.0.0.1 alla porta 80"

Ovvero 138.76.29.7:80 -> 10.0.0.1:80



Questo metodo non è però plug & play.

Un esempio di servizio che tutti usano e che potrebbe avere problema è la telefonia via IP (Skype). Se qualcuno dall'esterno vuole far squillare il telefono è difficile gestire questa cosa tramite port forwarding.



La seconda soluzione è quella del **relaying** che viene utilizzato nel caso di Skype. Quando si attiva Skype sul PC c'è uno scambio di messaggio con i Server skype. Durante questo scambio di messaggi ci si registra nella rete skype, cioè si va a dire in quel momento qual è il proprio indirizzo IP in modo tale da

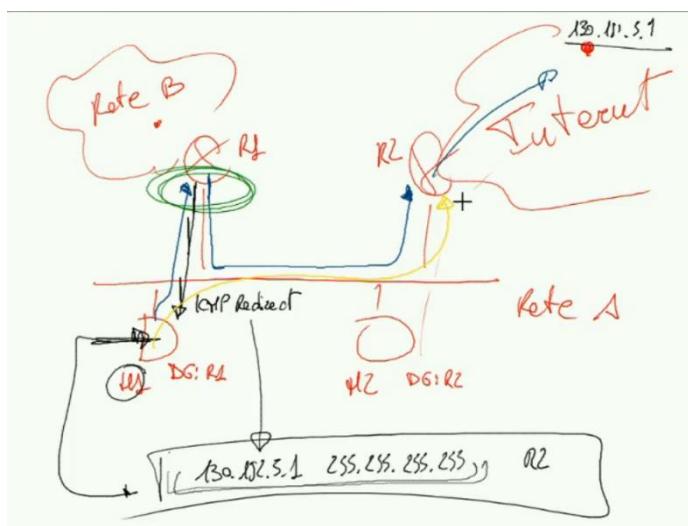
rendere questa informazione disponibile agli altri utenti Skype nel caso in cui volessero contattare l'host. Si va anche ad inviare un messaggio a un nodo particolare di Skype che farà da relay. Tale nodo avrà un indirizzo IP pubblico noto perciò sarà raggiungibile dagli host che si connettono a Skype.

Si va a comunicare a tale host se può fare da Relay, cioè si traduce nel fatto che nel sistema di Skype non ci sarà scritto che l'utente si trova all'indirizzo IP 10.0.0.1 (ovvio) ma non sarà neanche scritto che l'utente è raggiungibile all'indirizzo IP 138.76.29.7 (pubblico) ma sarà scritto che l'utente è raggiungibile tramite l'indirizzo IP del relay.

Quando qualcuno vuole contattarci, cercherà l'username nella rete Skype il quale come risposta dirà che l'host è collegato all'indirizzo IP del relay, che è raggiungibile perché pubblico. Il traffico verrà inviato al relay che processerà i dati fino al livello applicazione che capirà che il traffico dovrà essere inviato all'utente per cui lui sta facendo da relay. Quindi prende il traffico che arriva dall'utente e lo gira verso il proprio indirizzo IP pubblico.

Il vantaggio è che a inoltrare indietro il traffico sarà qualcuno che dall'interno abbiamo contattato perciò nel NAT sarà presente la tabella e perciò è tutto automatico.

Pacchetto ICMP redirect (breve citazione)



H1 mette come default Gateway R1 e H2 come DG mette R2.

Se H1 vuole comunicare con internet contatterà R1, ma quest'ultimo non può far altro che inoltrarlo verso R2. Siccome si sta inoltrando due volte pacchetti sulla stessa interfaccia, R1 comunica a H1 con una ICMP redirect che l'host può raggiungere direttamente Internet contattando R2.

Si punta ad un solo indirizzo IP (contrario del DG). Questo lo fa soltanto se non ci sono due reti IP configurate sulla stessa rete fisica.

6. Livello applicazione – Parte 1

È il livello in cui vi è maggiore innovazione. Esistono due tipi di architettura per le applicazioni:

- Client-server
- Peer-to-peer (P2P)

Nell'architettura client-server vi sono dei nodi che sono sempre online (i server) che non cambiano mai indirizzo IP. Ciò ci fa pensare che sono dei nodi che esistono per offrire un certo indirizzo ed è normale assumere che l'indirizzo IP non cambi mai.

Nel caso in cui il numero di utenti cresce, i server devono essere aumentati in modo da soddisfare tutti i client. La cosa migliore da fare è raggruppare tutti insieme i server e far diventare quel luogo un data center. In questo modo è più semplice gestirli. I client sono i dispositivi che devono comunicare con i server per ottenere un certo servizio e possono essere connessi in maniera intermittente ed avere anche indirizzi IP dinamici. I client non comunicano tra loro ma soltanto con i server.

Nell'architettura P2P non esistono server ma tutti i client sono dei **peer** cioè il servizio è offerto da tutti i peer messi insieme, che allo stesso tempo sono i fruitori del servizio.

Mentre sfruttano il servizio di cui hanno bisogno, forniscono il servizio stesso.

La rete P2P si dice sia **self scalability** (auto scalante) poiché se si aggiunge un peer si aggiunge un potenziale utente che nel giro di qualche secondo potrà partecipare alla fornitura del servizio.

Il P2P è costituito da peer che sono connessi in maniera intermittente e modificano l'indirizzo IP perciò la loro gestione è complessa.

Application Layer

Fino ad ora si è parlato di applicazioni che devono dialogare con altri client.

Si parla di protocollo di livello applicazione perché le applicazioni da sole non sarebbero in grado di capire quando è il momento di inviare una certa informazione.

Non ha senso che il server si svegli a caso per inviare il file html a qualcuno ma è necessario che gli si dica che ciò va fatto solo quando un client fa richiesta al web server che deve essere in grado di dare una risposta. Bisogna quindi definire i tipi di messaggi scambiati (richiesta, risposta), la sintassi (perché la richiesta deve essere comprensibile al server), una semantica (cioè definire cosa vuol dire richiesta http per esempio). Senza tutte queste cose l'applicazione da sola non funzionerebbe.

Riassumendo, bisogna definire le regole con cui le applicazioni si scambiano le informazioni. I protocolli possono essere aperti (e quindi definiti in RFC) come HTTP e SMTP, mentre altri possono essere proprietari come quello di Skype. La differenza è che di quelli proprietari non si conosce il funzionamento.

Che tipo di servizio di trasporto richiede un'app?

Un'applicazione potrebbe voler chiedere l'**integrità dei dati**, cioè richiedere la consegna dei dati. Alcune app possono richiedere questo servizio (trasferimento file, transazioni web) mentre altre applicazioni (come ad esempio l'audio telefonico) possono tollerare alcune perdite.

Un altro servizio è quello del **throughput garantito**. Alcune applicazioni non lo richiedono, come ad esempio il trasferimento dei file (cioè se si aspetta un po' di più non importa perché comunque il file viene ricevuto) e vengono definite **applicazioni elastiche**.

Tutto ciò non è vero per le **applicazioni multimediali** perché, pensando a come è fatto un video ovvero una sequenza di immagini, nel momento in cui si stabilisce la codifica della singola immagine per poterlo vedere alla destinazione le immagini vanno ricevute entro un certo intervallo di tempo e ciò vuol dire che se la rete offre un bit rate più basso rispetto alla codifica del video non si sarà in grado di vedere quel video. Ogni codifica di video fornisce un rate minimo di rete necessario per visualizzarlo. Queste sono le **applicazioni anelastiche**.

Un altro servizio è il **timing** che può essere richiesto da alcune applicazioni **interattive**, come le telefonate via internet e i giochi. Queste app richiedono che il ritardo sia minimo.

Esempi di applicazioni e livello trasporto

application	application layer protocol	underlying transport protocol
e-mail	SMTP [RFC 2821]	TCP
remote terminal access	Telnet [RFC 854]	TCP
Web	HTTP [RFC 2616]	TCP
file transfer	FTP [RFC 959]	TCP
streaming multimedia	HTTP (e.g., YouTube), RTP [RFC 1889]	TCP or UDP
Internet telephony	SIP, RTP, proprietary (e.g., Skype)	TCP or UDP

Si potrebbe pensare che per applicazioni di telefonia sarebbe meglio utilizzare UDP poiché, anche se non garantisce la ricezione di tutti i pacchetti, permette di “sovrrastare” TCP e quindi occupare più banda possibile (pur NON garantendo un throughput alto o ritardi bassi), ma in realtà i servizi come YouTube utilizzano HTTP, che si trova al livello sottostante

solo TCP.

Usare TCP oppure UDP in una rete poco congestionata (dove non scattano timeout) non fa differenza utilizzare TCP o UDP.

Per YouTube i ritardi non sono fondamentali poiché non è un'applicazione interattiva, perciò è meglio avere qualche ritrasmissione in modo tale da avere qualità migliore.

Anche Skype utilizza TCP per le comunicazioni telefoniche, poiché è meglio per comunicare con i Relay/NAT e per evitare i ritardi forzano (truccando, modificando) TCP a NON ritrasmettere nel caso di pacchetti non consegnati.

Sicurezza TCP

Finora non si è mai citata la parola sicurezza su TCP. Le due opportunità sono quindi:

- Internet non è sicuro: spetta alle applicazioni implementare le funzioni
- Visto che tutte le applicazioni avrebbero il problema della sicurezza è meglio realizzare soluzioni direttamente nel sistema operativo (reso in maniera generica al livello applicazione).

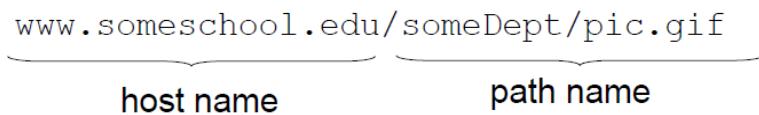
Viene implementata la seconda opzione, andando ad aggiungere un livello tra l'applicazione ed il trasporto chiamato **SSL** (Secure Socket Layer) e il suo compito è rendere sicuro il socket (la nuova versione è detta **TLS**).

Ad esempio, con HTTPS l'applicazione non varia, ma neanche il trasporto. Viene solo aggiunta la sicurezza SSL.

Web e HTTP

Una **pagina web** è un insieme di **oggetti** che possono essere: file HTML, immagini, applicazioni Java, file audio, etc..

Tutte le pagine web consistono di un file **HTML di base** che include diversi oggetti referenziati. Ogni oggetto è indirizzabile da un **URL** formato come segue:



Una prima parte è l'**host name** che è nient'altro che il nome associato al luogo in cui si trovano gli oggetti (l'indirizzo IP in cui si trovano gli oggetti)

Il **path name** è la posizione all'interno del file system di quell'oggetto in modo da trovarlo.

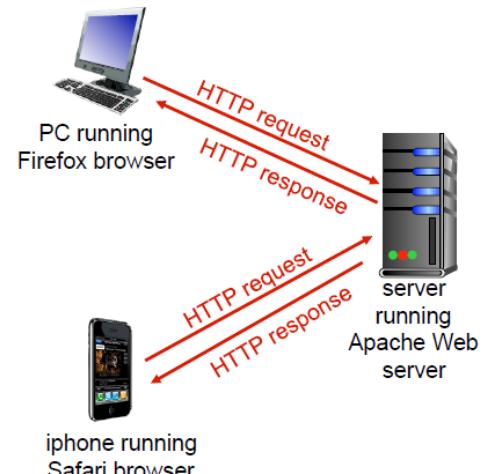
Per trasferire questi oggetti si utilizza il protocollo **HTTP** (Hypertext transfer protocol) che è di tipo client-server. I **client** sono i browser che richiedono e ricevono gli oggetti web utilizzando il protocollo HTTP. Il **server** è invece colui che invia, tramite protocollo http, gli oggetti in risposta alle richieste.

I server sono dei software (come Apache) che permettono di offrire pagine web.

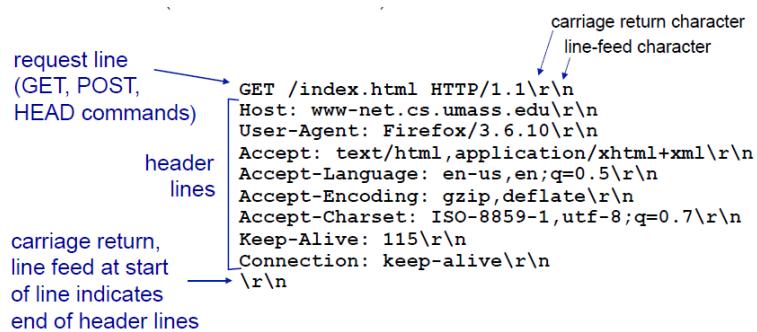
HTTP utilizza **TCP** e la sua porta è **80**. Il client aprirà una connessione TCP e scambierà dei messaggi HTTP che servono per ottenere la pagina web. HTTP è **stateless**, cioè il server non mantiene alcuna informazione sulle richieste passate del cliente (ovvero senza ulteriori aggiunte non sarebbe possibile creare un carrello nei siti di ecommerce) perciò ogni richiesta è completamente indipendente dalla precedente.

HTTP può essere **persistente** oppure **non persistente**, cioè si può decidere di mantenere una connessione TCP aperta per scaricare più oggetti oppure no.

Tra le due scelte la migliore è quella **persistente** poiché avendo la connessione già aperta non è necessario attendere i RTT necessari per aprire/chiudere la connessione.



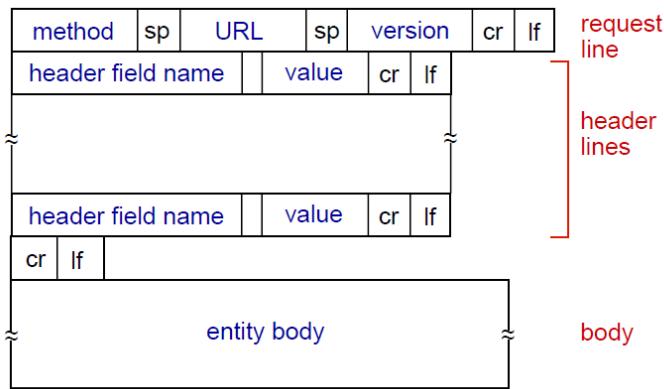
Richiesta HTTP



Le richieste HTTP sono codificate in **ASCII** (cioè sono human-readable). Ciò comporta che si stanno usando più bit. I tipi di richiesta possono essere di tipo GET, POST, HEAD. In binario basterebbero due bit per codificare le tre richieste, mentre in questo caso si utilizzano 8 bit per ogni lettera. Essendo le applicazioni molto vicine allo sviluppatore devono

essere facilmente leggibili e facili da ricordare.

Il formato generale è il seguente:



Nell'esempio precedente il body non c'era perché era una richiesta. Il body è separato dall'header da una linea bianca.

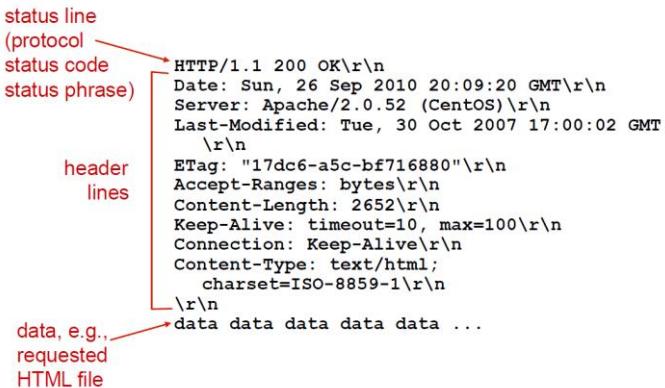
Il campo corpo del messaggio è necessario nel metodo **POST** che serve per gestire i form (ad esempio quando si ha uno spazio di N caratteri per scrivere del messaggio).

Si può anche usare l'**URL method** che utilizza il metodo GET per aggiungere le informazioni direttamente nel path name.

In HTTP 1.0 si ha anche l'**HEAD** che è esattamente come la GET dove però chiedo al server di non inviare l'oggetto. Ciò serviva soltanto per effettuare debugging.

In http 1.1 si sono aggiunte le richieste **PUT** e **DELETE** che servono per aggiungere o rimuovere dei file al server (tipico dei servizi cloud).

HTTP response message



Nella risposta si ha una **status line** in cui si va a scrivere nuovamente il protocollo utilizzato, un codice numerico ed una “frase” (in questo caso OK) che meglio descrive il codice numerico. Il codice 200 indica che tutto è andato a buon fine. In coda (data) si avrà il messaggio richiesto. Dopo la status line anche in questo caso vi è un header con vari parametri tra cui: data, ora, last modified (ultima modifica).

Alcuni codici di risposta:

200 OK

- request succeeded, requested object later in this msg

301 Moved Permanently

- requested object moved, new location specified later in this msg (Location:)

400 Bad Request

- request msg not understood by server

404 Not Found

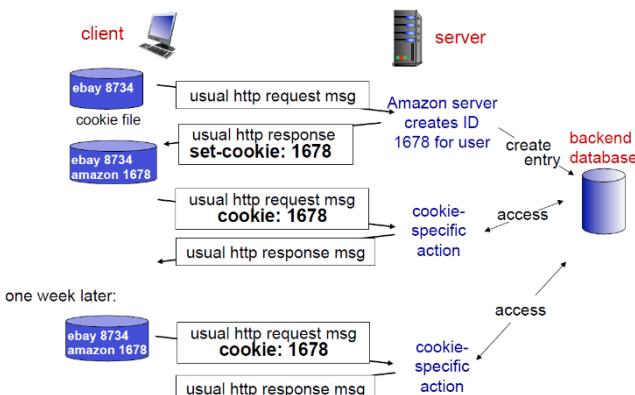
- requested document not found on this server

505 HTTP Version Not Supported

404: restituisce questo codice quando si chiede un oggetto che non c'è (o magari c'era ma non c'è più).

301: si può configurare in modo da avvisare che un oggetto è stato spostato. E' possibile aggiungere la “location” per informare dove è possibile reperire l'oggetto.

Cookies



"**set-cookie**" e va a dire al client che da quel momento in avanti sarà associato a quel numero.

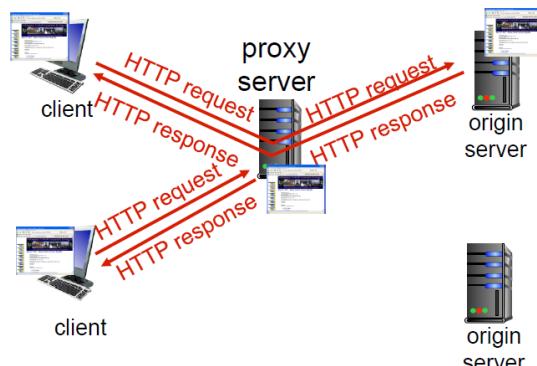
Da quel momento in poi è possibile tenere traccia delle operazioni che il client svolge su quel sito web. Ciò è possibile perché anche il client ha memorizzato in un database locale il cookie ed effettuando le richieste aggiungerà all'header il cookie. In questo modo il server potrà associare le operazioni a tale cookie, andando ad aggiornare le entry sul database (del server).

Il problema dei cookie è che i server possono ledere la privacy degli utenti utilizzando i cookie, cioè andando a leggere ciò che si è fatto sul sito web (eventuali ads cliccati, prodotti selezionati, etc..)

Web caches

Il web caching è un concetto simile al DNS caching cioè evitare che l'utente vada ad utilizzare delle risorse in rete. Nel caso del DNS si evitava che l'utente non andasse a interrogare server esterni per risolvere un indirizzo IP. In questo caso le cache in giro per la rete serviranno per evitare di andare al server finale per ottenere gli oggetti. Come nel caso del DNS è presente una cache locale nel client ed una eventuale cache che si può posizionare in rete detto **proxy server http**. La cache sul client è interna al client (dunque al browser). Ciò vuol dire che effettuando una richiesta http prima di far uscire la get si effettuerà un controllo in cache.

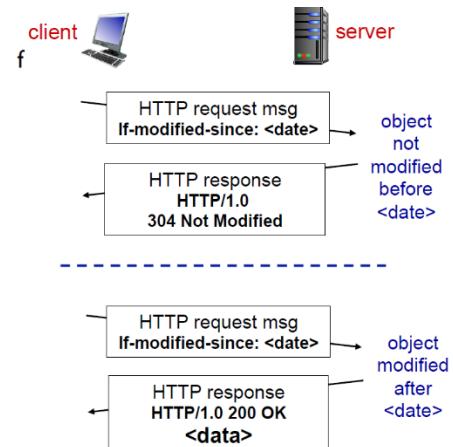
La cache di rete messa a disposizione da un proxy http evita che i client che utilizzano quella rete propaghino la richiesta fino al server di origine.



Conditional GET

Quando il contenuto della cache non è aggiornato ciò può diventare un problema. I siti web più famosi non sono per nulla statici e il loro contenuto viene modificato continuamente. Tali server possono richiedere alla cache di non effettuare il caching, in modo tale da avere sempre la pagina più aggiornata.

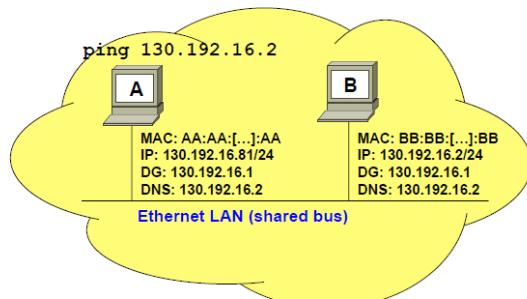
Esiste però una possibile soluzione in cui si effettua una GET che arriva sino al server che però contiene una condizionale e si fa restituire dal server l'oggetto soltanto se tale dato è stato modificato dall'ultima volta che è stata effettuata una copia in cache locale. Se la copia che si possiede è valida, il server risponde col codice 304, in questo modo si è sicuri che la versione in cache è aggiornata.



7. Esercizi di Traffic Analysis

Esercizio 1

- Assuming that all caches are empty, indicate the number and the type of the frames captured by a sniffer located on the network cable of Host A.
- Solution: 10 frames (if MS Windows host)



sorgente/destinazione.

Svolgimento:

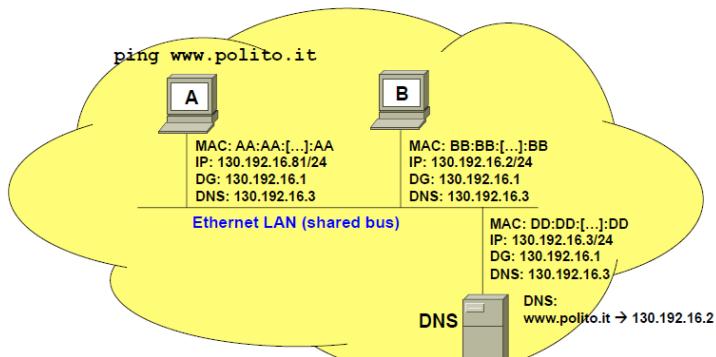
N	L2		L3		PAYLOAD
	MAC SRC	MAC DST	IP SRC	IP DST	
1	AA:...:AA	FF:...:FF	//	//	ARP Request di A per MAC B
2	BB:...:BB	AA:...:AA	//	//	ARP Reply B->A
3	AA:...:AA	BB:...:BB	130.192.16.81	130.192.16.2	ICMP Echo Request
4	BB:...:BB	AA:...:AA	130.192.16.2	130.192.16.81	ICMP Echo Reply

Bisogna controllare che A e B si raggiungano (controllo IP e subnet). Nelle ARP non interviene IP dunque non c'è IP.

La soluzione dice 10 perché si suppone che Windows invii 4 volte la ICMP Echo Request (dunque $2 \times 4 = 8$).

Esercizio 2

- Assuming that all caches are empty, indicate the number and the type of the frames captured by a sniffer located on the network cable of Host A.



Assumendo che tutte le cache siano vuote, indicare il numero e il tipo di trame catturate da uno sniffer (network analyzer) situato sul cavo di rete dell'host A (In altre parole wireshark gira su A).

Quando viene lanciato il ping sull'host A quando A è 16.81/24 e B è 16.2/24. Il primo pacchetto è l'**ARP Request** seguito dall'**ARP Reply**. Entrambe verranno viste da A. In ogni caso la rete è di tipo shared bus quindi per definizione la rete è un mezzo broadcast ed A vedrà tutto il broadcast chiunque sia

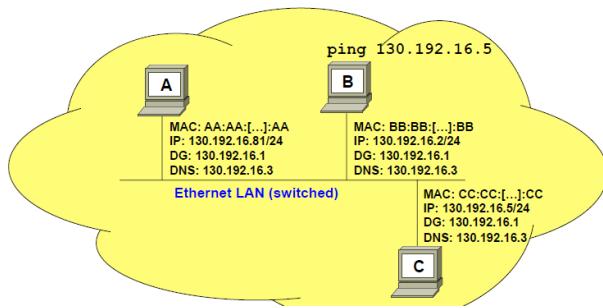
Si vuole fare il ping a www.polito.it avendo tra le proprie configurazioni di rete anche un server DNS che conosce il mapping del sito desiderato. Si procede allo stesso modo di prima ma bisogna ricordarsi che prima di poter fare il ping a B bisogna effettuare la risoluzione del nome. Verrà effettuata una ARP Request (poiché tutte le cache sono vuote) ma verso il server DNS che è raggiungibile direttamente poiché è nella stessa rete.

	L2		L3		
N	MAC SRC	MAC DST	IP SRC	IP DST	PAYLOAD
1	AA:...:AA	FF:...:FF	//	//	ARP Request di A per MAC DNS
2	DD:...:DD	AA:...:AA	//	//	ARP Reply DNS->A
3	AA:...:AA	DD:...:DD	130.196.16.81	130.196.16.3	DNS Query
4	DD:...:DD	AA:...:AA	130.196.16.3	130.196.16.81	DNS Response
5	AA:...:AA	FF:...:FF	//	//	ARP Request di A per MAC B
6	BB:...:BB	AA:...:AA	//	//	ARP Reply B->A
7	AA:...:AA	BB:...:BB	130.192.16.81	130.192.16.2	ICMP Echo Request
8	BB:...:BB	AA:...:AA	130.192.16.2	130.192.16.81	ICMP Echo Reply

In questo caso prima di fare l'ARP request a B bisogna effettuare una query al DNS chiedendo prima il mac address tramite ARP e successivamente si ritorna all'esercizio precedente.

Esercizio 3

- Assuming that all caches are empty, indicate the number and the type of the frames captured by a sniffer located on the network cable of Host A.

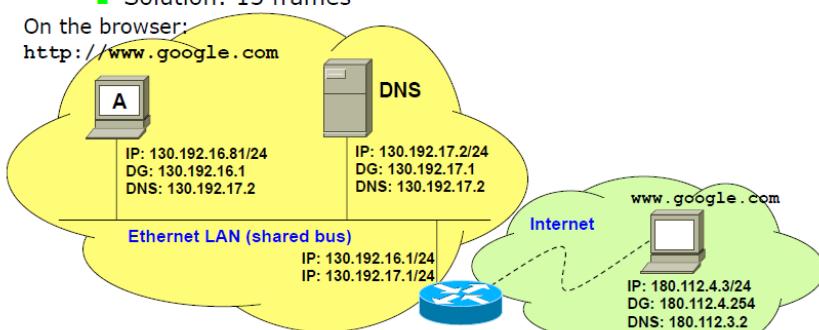


di B e C verso A.

Risultato: **1 solo pacchetto, ARP Request.**

Esercizio 4

- Assuming that all caches are empty (except that of the DNS server), indicate the number and the type of the frames captured by a sniffer located on the network cable of Host A. Consider that the webpage can be included in a single packet
- Solution: 13 frames



In questo esercizio l'host B vuole effettuare il ping all'IP dell'host C. In questo caso il tipo di rete è importante (switched) perché lo sniffer rimane comunque sull'host A.

Se la rete è switched, lo switch sarà sempre in grado di sapere dove sono le varie destinazioni. Quando B deve effettuare il ping di C, questo farà un'ARP request a cui susseguirà una reply e poi le ICMP. A parte l'ARP Request, le altre son tutte trame unicast tra B e C. Per tali motivi, A vedrà soltanto l'ARP Request di B. Lo switch dunque non invierà il traffico

In questo caso non vi è più un ping ma sul browser di A viene lanciato il comando http. Verrà dunque generata una GET. Sotto http c'è TCP dunque va aperta la connessione. In più sono presenti due reti logiche sulla stessa rete fisica. A ed il DNS non si parlano direttamente ma dovranno comunicare attraverso il router.

Soluzione:

	L2		L3		
N	MAC SRC	MAC DST	IP SRC	IP DST	PAYLOAD
1	MAC A	Broadcast	//	//	ARP Request di A per D.G.
2	MAC DG	MAC A	//	//	ARP Reply D.G.->A
3	MAC A	MAC DG	130.196.16.81	130.196.17.2	DNS Query
4	MAC DG	Broadcast	//	//	ARP Request di D.G. per DNS
5	MAC DNS	MAC DG	//	//	ARP Reply DNS -> D.G.
6	MAC DG	MAC DNS	130.196.16.81	130.196.17.2	DNS Query
7	MAC DNS	MAC DG	130.196.17.2	130.196.16.81	DNS Response
8	MAC DNS	MAC A	130.196.17.2	130.196.16.81	DNS Response
9	MAC A	MAC DG	130.196.16.81	180.112.4.3	TCP SYN
10	MAC DG	MAC A	180.112.4.3	130.196.16.81	TCP SYN ACK
11	MAC A	MAC DG	130.196.16.81	180.112.4.3	TCP ACK

Controllo le configurazioni: 16.1 come default gateway per A va bene e 17.2 come default gateway per il DNS va bene.

Primo pacchetto da inviare: A si chiederà come risolvere il nome google.com e dunque cercherà di raggiungere il DNS. L'host A farà l'AND bit a bit tra la sua maschera e il suo indirizzo IP e tra l'indirizzo IP destinazione e la sua maschera. Il 16.81 con la maschera fa 16.0 mentre 17.2 con la sua maschera fa 17.0 dunque sono due reti diverse e l'ARP Request viene fatta per l'indirizzo MAC del default gateway.

Il DG risponde e allora A effettuerà la DNS Query inserendo come IP Destinazione quello del **DNS**.

Il DG farà un'ARP Request al DNS il quale risponderà e a quel punto potrà partire la DNS Query pendente (gli indirizzi IP restano gli stessi).

Il DNS per rispondere deve ripetere il processo poiché l'host A non è nella rete del DNS ma possiede in cache l'indirizzo MAC del DG.

Il DG possiede gli indirizzi MAC di entrambi e la richiesta viene inoltrata senza alcuna ARP Reply.

Il pacchetto 9 sarà una TCP SYN che conterrà come IP destinazione l'indirizzo IP di google.com che però viene inviata all'indirizzo MAC del DG.

Il sito risponderà con una TCP SYN ACK verso l'host A (anche in questo caso il MAC address destinazione sarà il DG).

8. Livello applicazione – Parte 2

Posta elettronica

L'**user agent** è un client di posta. Quest'ultimo permette di avere delle copie delle e-mail per poterci lavorare anche in modalità offline. Lo user agent permette di scrivere le e-mail e poi trasferirle (cliccando invia) sui server mail.

In questo caso la mail non viene inviata direttamente al destinatario perché sorgerebbero numerosi problemi: è difficile sapere dove il destinatario si trova, il destinatario potrebbe essere offline, etc..

Per risolvere questi problemi esistono dei protocolli che permettono di custodire le email nelle **email box** che vengono create sui server di posta. L'obiettivo dell'user agent è trovare il server che è responsabile della mailbox della persona da contattare.

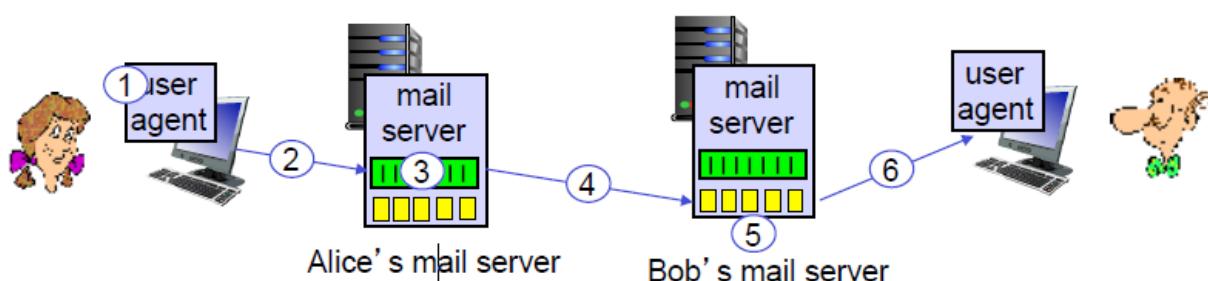
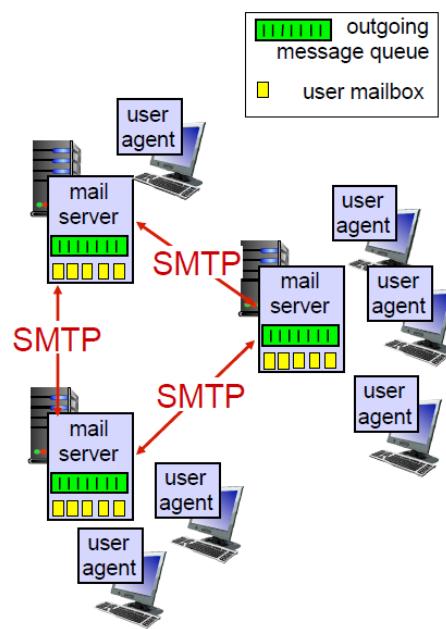
Anche in questo caso però non si contatta direttamente il server di posta ma ci si appoggia su un server locale detto **outgoing server**.

Ciò avviene perché una volta non si era "always on" e dunque il sovraccarico era molto maggiore. L'invio di una email prevede queste fasi:

- Un utente invia un messaggio tramite l'user agent a un indirizzo email (a.e. bob@someschool.edu)
- L'user agent invia il messaggio al mail server del client e dunque i messaggi vengono inseriti nella **message queue**.
- Il client SMTP apre una connessione TCP col mail server del destinatario
- Il client SMTP invia il messaggio dell'utente tramite connessione TCP
- Il mail server del destinatario posiziona la mail nella mailbox
- Il destinatario apre l'user agent per visualizzare il messaggio.

Ricapitolando nell'invio della posta intervengono:

- **Mailbox** che contengono i messaggi in arrivo per l'utente
- **Message queue** dei messaggi in uscita da inviare
- **SMTP protocol** tra i server di posta per inviare i messaggi.



Protocollo SMTP

Il protocollo SMTP utilizza **TCP** che resta in ascolto sulla porta **25**. Il trasferimento è diviso in tre fasi: **handshaking** (diverso da TCP), **trasferimento di messaggi**, **chiusura sessione** (seguita dalla chiusura della connessione).

I messaggi SMTP sono anch'essi codificati in ASCII (e non più in binario) e si nota che anche i messaggi devono essere 7-bit ASCII (l'8bit è arrivato dopo).

In SMTP è il server che parla per primo (diverso da http). Il server invia un codice numerico ed una descrizione (che tendenzialmente è il nome del dominio ma non per forza deve essere così). Quando il client riceve il messaggio, quest'ultimo risponde col

suo messaggio di benvenuto seguito dal nome del suo dominio (poteva essere qualsiasi altra cosa, non per forza nome del dominio). Il server risponde col codice 250 e termina la fase di saluto. Il prossimo messaggio inviato dal client al server informa il server chi è il mittente della mail che sta per trasferire. La risposta è nuovamente 250 che indica che al server va bene gestire una mail che arriva da quell'utente.

Successivamente il client scrive il comando **RCPT TO** che serve ad indicare il destinatario della mail. Il server risponde nuovamente 250 per dire che il destinatario va bene. Termina qui la fase di handshaking e può avvenire il trasferimento di dati.

SMTP sta comunicando tra il server di alice e il server di bob (vedi foto precedente). SMTP comunica anche tra client (Alice) e server (Alice server mail).

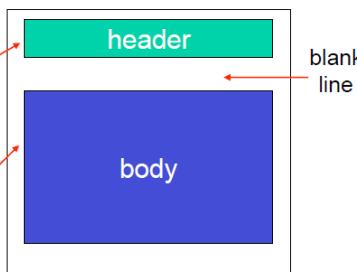
Il “.” Indica la fine del messaggio (per inviarlo come carattere e non come fine messaggio si scrive CRLF.CRLF).

Una volta ricevuto il punto il server comunica che ha ricevuto il messaggio ed è pronto per essere inviato e dunque termina la connessione (a cui seguirà la chiusura di TCP).

Si nota che da nessuna parte avviene l'autenticazione e/o la codifica del messaggio mail (oggi non è più ragionevole). In realtà oggi sulla porta 25 non sempre vi è il servizio di posta attivo, poiché oggi si utilizza la versione sicura di SMTP con SSL.

Formato messaggio mail

- SMTP: protocol for exchanging email msgs
- RFC 822: standard for text message format:
- ❖ header lines, e.g.,
 - To:
 - From:
 - Subject:
- different from SMTP MAIL FROM, RCPT TO: commands!*
- ❖ Body: the “message”
 - ASCII characters only

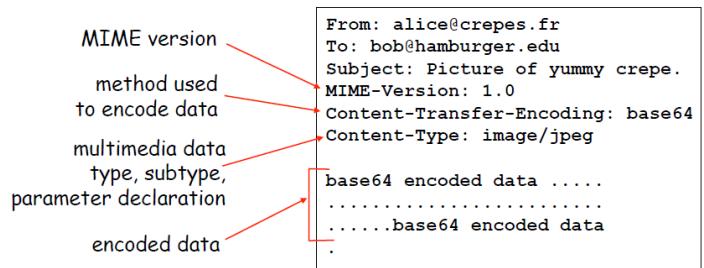


I campi **MAIL FROM** e **RCPT TO** hanno validità temporale minima: il server capisce chi sono e se ne dimentica. Solo la mail a finire nella mailbox e dunque tutti i campi from/subject/to sono da inserire nel **messaggio** all'interno del campo header.

Formato del messaggio: estensioni per i media

Nella stessa mail è possibile avere la composizione di più elementi: testo, foto, video, etc.. E per tali motivi nascono dei problemi: avere elementi diversi e far capire al destinatario come decodificare il messaggio; evitare i problemi di codifica dovuti a SMTP che utilizza il 7bit ASCII.

Il **MIME (Multimedia Mail Extension)** prevede l'aggiunta di ulteriori header che specifica (tramite ulteriori header) cosa si sta inviando.



Per evitare il problema del "CRLF.CRLF" sono stati standardizzati delle apposite codifiche come il **base64** che **non** hanno tale codifica (a capo punto a capo). I client di posta contengono già tale codifica.

Riguardo alle mail che comprendono testo+foto si utilizza il **mime multipart** si possono dichiarare le varie sezioni della mail con il relativo contenuto. I vari tipi di MIME sono:

Text

- ❖ example subtypes: plain, html

Image

- ❖ example subtypes: jpeg, gif

Audio

- ❖ example subtypes: basic (8-bit mu-law encoded), 32kadpcm
(32 kbps coding)

Video

- ❖ example subtypes: mpeg, quicktime

Application

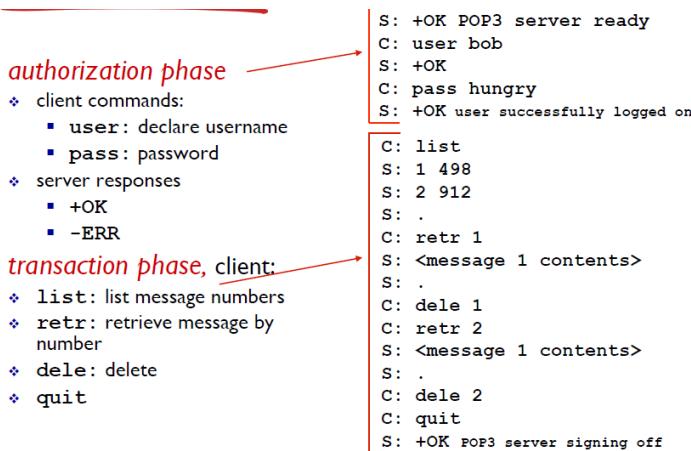
- ❖ other data that must be processed by reader before "viewable"
- ❖ example subtypes: msword, octet-stream

Protocolli di accesso mail

Per poter permettere al destinatario di accedere al server e leggere la posta servono ulteriori protocolli di accesso. Questo perché l'operazione in questo caso è **asincrona** (cioè dipende dal momento in cui il destinatario va online). Esistono due possibili protocolli utilizzabili dagli user agent: **POP** e **IMAP** ma esiste anche la possibilità di utilizzare **HTTP** (come gmail che però è un sito web che avrà un backend che comunicherà col server di posta).

POP3

Il POP3 è un protocollo testuale dove i messaggi vengono scambiati allo stesso modo del protocollo SMTP. La fase di **autorizzazione** serve per identificare la casella di posta corretta.



Tramite l'autenticazione è possibile identificare la corretta casella di posta. Successivamente vi è la **fase di transazione**. Il tutto parte col comando **list** che permette di ricevere l'elenco delle mail che possiedono un numero identificativo e il numero di byte della mail. Viene scelta la dimensione della mail perché sono il client di posta ed il server a comunicare e dunque sarà l'utente successivamente a decidere quali mail tenere e quali scartare. Dunque, l'informazione utile al client di posta è la dimensione delle mail. Il comando **retr** permette di scaricare il messaggio e successivamente è possibile eventualmente inviare il messaggio **dele** per scegliere se lasciare o meno una **copia sul server**. Infine, vi sarà il **quit** per uscire e terminare.

Anche in questo caso il server oggi sta su una diversa porta ed è criptato con protocollo SSL.

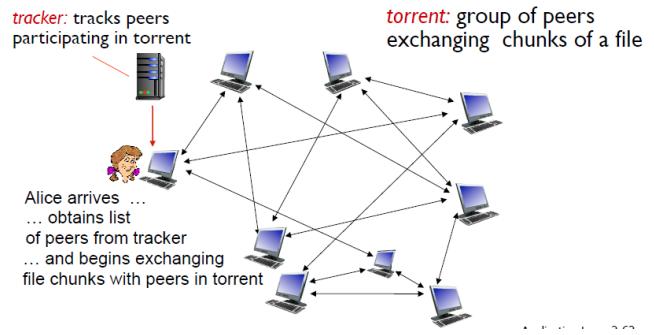
In POP3 non esiste alcun tipo di stato e dunque se nel client si effettua una suddivisione in cartelle delle email, sul server rimane semplicemente la lista delle e-mail e le modifiche fatte sul client non vengono salvate sul server. Per risolvere il problema esiste il protocollo **IMAP**

IMAP

Imap è una versione di POP **stateful** poiché viene mantenuto lo stato della posta. Vengono mantenuti tutti i messaggi sul server e permette l'utente di organizzare le mail in cartelle. Dunque, una cartella creata sul client di posta viene replicata sul server e in caso si configuri la stessa mail su un altro client, verrà anch'esso sincronizzato con le varie cartelle.

BitTorrent

I file vengono divisi in vari pezzi detti **chunks** ed il **torrent** è un gruppo di peer che stanno scambiando i vari chunks di un file. Il primo problema da affrontare nel P2P è **individuare** gli altri peer. Per poter fare ciò in BitTorrent si utilizza il **tracker** che è un server che traccia tutti i peer che partecipano al torrent. Ogni peer che vuole connettersi deve andare dal tracker ed “annunciare” di voler diventare un peer e ottenere la lista degli altri peer. Questo è un esempio di **indexing server**. Il protocollo BitTorrent però non utilizza tutti i peer a disposizione, ma un **sottogruppo**. Il tracker non dà gli stessi dati a tutti i peer, e ciò permette di creare una topologia magliata che permette ai contenuti di diffondersi.



- **Richiedere un chunck:** periodicamente un client chiede a tutti i peer la lista di tutti i chuncks che possono offrire ed effettua una classifica di diffusione dei chuncks. Tra tutti quelli che le servono il client sceglierà il **chunck più raro**. Questo è l’algoritmo del **local rarest first**. Questo perché è molto più utile scaricare un chunck raro in modo da permettere di aumentare il numero di peer che possiedono tale chunck per condividerlo più facilmente.
- **Tit-for-tat:** se vi sono più persone che chiedono a un utente di scaricare un chunck, il client sceglie prima coloro che hanno scambiato più chunck in passato con tale utente. In questo modo si favorisce chi **contribuisce** a condividere le informazioni. Ogni 30 secondi però viene anche scelto un peer in modalità **random** per “rimescolare” il tutto e permettere a tutti di ristabilire una situazione equa (a.e. chi magari aveva una connessione lenta e poi è migliorata).

DHT – Distributed Hash Table

Introducendo BitTorrent si è detto che l’**indexing**, cioè la funzione necessaria in un sistema distribuito per la localizzazione delle informazioni, è un indice (a.e. conoscere i peer che forniscono un file). In BitTorrent è tutto centralizzato tramite il **tracker**. L’applicazione non è dunque completamente P2P.
La DHT permette di decentralizzare l’indexing utilizzando un database completamente distribuito. Non esiste più un tracker ma per conoscere i peer di un file bisogna contattare un peer della rete. Questo rende più difficile “oscurare” un file poiché più peer possono condividere l’informazione che prima erano contenute nel tracker.

Il database DHT possiede delle coppie (**chiave, valore**) dove in questo contesto la chiave può essere il nome di un file da scaricare mentre il valore può essere la lista degli indirizzi IP che offrono tale file. L’idea è dunque quella di distribuire tra tutti i peer le coppie di chiave e valore in modo tale da rendere i peer responsabili di una o più risorse (cioè avrà la lista dei peer che offrono quel file e non che si possiede tale file). Se un peer si scollega dalla rete tali dati verranno indirizzati verso un altro peer.

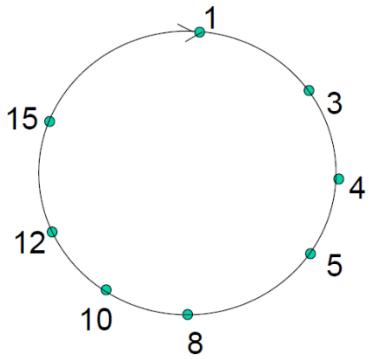
L’idea di base è quella di convertire la risorsa in un identificativo numerico, dopodiché si converte anche il peer. Entrambi saranno negli stessi intervalli (a.e. entrambi su 32 bit). Successivamente si assegna la risorsa all’identificativo più vicino (se non uguale).

Questo meccanismo funziona finché ogni risorsa possiede un identificativo univoco. Si utilizza un numero di peer sufficientemente alto (+- 128 bit) e si utilizzano delle funzioni di **hash** che hanno la caratteristica di rendere poco probabile la possibilità che due risorse abbiano lo stesso identificativo. Dunque, una volta generato l’identificativo verrà abbinato al peer con identificativo “più vicino”. Generalmente per “più vicino” si intende l’**immediato successore** della chiave. Ad esempio, se ho una lista di peer: 1,3,4,5,6,10,12,14 e ho una chiave=13 il successore sarà 14. Se invece la chiave è 15 il successore è 1.

Circular DHT

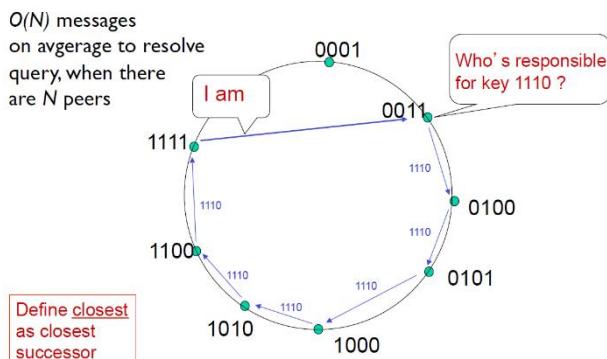
Un utente che accede alla rete bisogna individuare l'identificativo del peer che possiede la risorsa. Accedendo alla rete si conosce l'algoritmo da utilizzare per la ricerca e dunque viene applicata la funzione di hash alla parola cercata e si ottiene l'identificativo del peer che possiede tale risorsa. Una volta ottenuto il peer è necessario però ricavare l'indirizzo IP di tale peer e per poter comunicare con tale utente si è creata una struttura tra i peer su cui viene definito un algoritmo di routing.

Nel momento in cui un peer vuole connettersi deve posizionarsi in uno specifico punto della topologia logica della rete. Ogni peer deve conoscere alcuni altri peer e queste conoscenze non possono essere casuali, ma devono essere create secondo una struttura specifica: quella **circolare**.



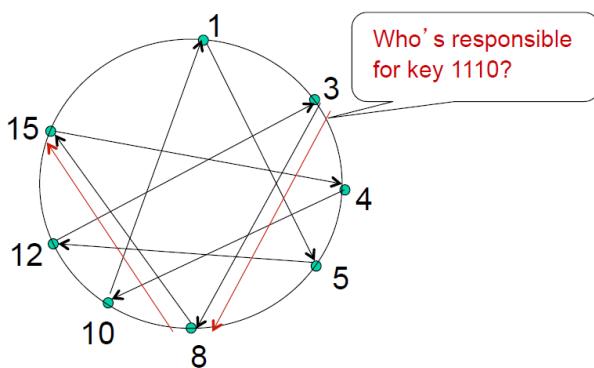
Ad esempio, il peer 1 deve essere a conoscenza del successore (peer 3) e del predecessore (peer 15). Se arrivasse il peer 2 (che in foto non c'è) deve posizionarsi nel corretto punto della rete e dunque conoscere il peer 1 e 3. A questo punto il peer 1 dovrà puntare a 2 e il peer 3 dovrà puntare a 2. Per poter creare questa lista linkata devono essere utilizzati gli indirizzi IP. Alla creazione della rete vengono inseriti dei peer di startup di cui si conoscono gli indirizzi IP e, non appena un peer si collega alla rete, tramite uno scambio di messaggi, riuscirà ad inserirsi correttamente nella rete.

Queste reti vengono dette **reti P2P strutturate**.



Il peer 0011 cerca la chiave 1110 e inizia a "girare" su tutta la rete (lo può fare perché ogni peer conosce il successore) finché non arriva al peer che è responsabile di tale chiave e potrà rispondere direttamente a chi ha fatto la domanda inviando la lista di peer che posseggono tale torrent.

In realtà tale ricerca è **lineare**. Esistono delle versioni più efficienti di complessità logaritmica.



In versioni più evolute è possibile sfruttare il collegamento a ritroso (tramite due ricerche in parallelo) oppure utilizzare degli **shortcuts**, ovvero delle scorciatoie che realizzano connessioni con successori e con peer distanti a distanza 2^N . In questo modo ci si avvicina molto più velocemente al peer responsabile della risorsa.

9. Multimedia networking

- **Audio:** il segnale audio analogico viene convertito in un segnale digitale tramite campionamento e quantizzazione. La frequenza di campionamento determina gli standard per gli audio digitali: per il telefono si utilizzano 8000 campioni/s per un rate di 64kbit/s; per il CD si utilizzano 44100 campioni/sec. Le varie codifiche utilizzano rate diversi: CD 1.411 Mbps; MP3 95, 128, 160 kbps; telefonia internet 5.3 kbps e superiori.
- **Video:** sequenza di immagini visualizzate con rate costante (a.e. 24 immagini/sec). L'immagine digitale è un vettore di pixel ciascuno rappresentato tramite bit. Le codifiche video utilizzano delle tecniche di **ridondanza spaziale o temporale** che permettono di ridurre il “peso” delle immagini codificate. La ridondanza spaziale permette di evitare di inviare più volte pixel che rimangono dello stesso colore tra un frame ed un altro. La ridondanza temporale invece contiene solo le differenze con l'immagine precedente. Queste codifiche generano un flusso **VBR (variable bit rate)** mentre l'assenza di queste tecniche viene detta **CBR (costant bit rate)**. Con la codifica MPEG4 si riescono a inviare video di buona qualità con un rate basso (< 1 Mbps).

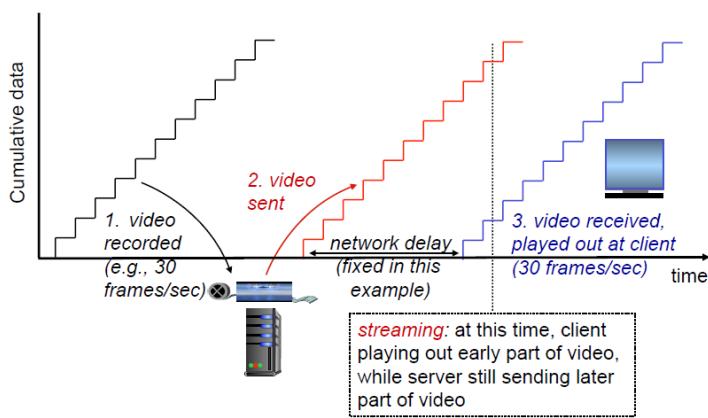
Streaming: il playout inizia prima del download dell'intero file. Se si aspetta di avere tutto il file prima della riproduzione si tratta di file sharing.

stored (at server): il contenuto dello streaming è già stato prodotto in precedenza e salvato da qualche parte.

Streaming live: in questo caso se vi fossero dei ritardi con il server durante la visualizzazione dell'evento live si potrebbero avere dei problemi.

VoIP: conversazioni audio/video interattive tra due utenti sulla rete.

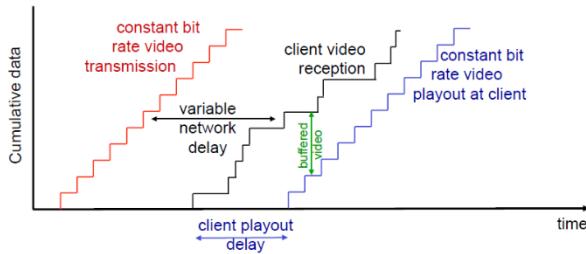
Streaming stored video



All'istante 0 viene inviato il primo frame e finché non si invia il secondo frame non succede nulla (dunque resta piatto, guardare il gradino in foto). I gradini sono di pari altezza poiché si suppone una tecnica **CBR**.

Nella seconda parte avviene l'invio di ciò che è stato codificato. Quando si ha un ritardo nella rete, succede che alla riproduzione sul client la riproduzione partirà con un certo ritardo rispetto a quando il video è stato inviato. Se questo ritardo è costante non è un grosso problema, poiché per ogni singolo bit inviato vi sarà sempre una distanza temporale uguale con cui verrà riprodotto il video sul client.

Il problema risiede nel fatto che i ritardi non sono costanti (dovuti a tutti i problemi della rete come la congestione).



Quello che accade realmente è rappresentato in questo grafico, con ritardi differenti per ogni frame. Se si riproducesse il video come mostrato nella seconda "scala" non si riuscirebbe a visualizzare nulla.

Per questi motivi al client non viene visualizzato il video appena lo si riceve ma si aggiunge un ritardo aggiuntivo, memorizzando i vari frame in un **buffer**

(playout delay). Aggiungendo dunque del ritardo aggiuntivo si risolve il problema dei ritardi della rete (non è ottimale per le chiamate telefoniche). Tutto ciò funziona bene finché il secondo grafico non incrocia il terzo. Nel momento in cui succede il buffer è vuoto e dunque si blocca il video al client.

Streaming multimedia: UDP

Per questo tipo di streaming sarebbe da preferire l'utilizzo di UDP poiché non è soggetto al controllo di congestione e si riesce ad utilizzare un playout delay relativamente basso. Non essendoci il ripristino degli errori in caso di mancata consegna, deve essere il client ad accorgersene e a richiedere i frame. Inoltre, UDP potrebbe non riuscire a oltrepassare i firewall.

Streaming multimedia: HTTP

I file multimedia vengono ricevuti tramite HTTP get e dunque TCP è una buona scelta in questo tipo di applicazioni, utilizzando un playout buffer più ampio e riuscendo a recuperare i pacchetti persi. HTTP/TCP riesce a passare meglio attraverso il firewall.

Streaming multimedia: DASH (Dynamic Adaptive Streaming over HTTP)

Il DASH è la vera soluzione al problema poiché con questa tecnica di trasmissione multimediale è stata introdotta la possibilità di adattare il video alle condizioni della rete. Questo perché il problema maggiore è dovuto alla congestione, ma in assenza di questo problema l'utilizzo di UDP/TCP è indifferente.

Il DASH monitora lo stato della rete e permette di adattare la qualità del video allo stato della rete (se la qualità della rete è buona, il flusso dati video aumenta e la qualità è migliore, viceversa la qualità scende ma la riproduzione è comunque possibile).

I file vengono dunque salvati sui server con varie risoluzioni e vengono divisi in "chunck" e, se la qualità della rete peggiora il prossimo chunck viene inviato a risoluzione inferiore.

Live Streaming

La richiesta del live streaming è quella di fornire la stessa esperienza che un utente avrebbe con la TV (ma ciò non è esattamente possibile). Questa richiesta si traduce in: bassi ritardi, throughput appropriato e basso packet loss ratio per garantire qualità video.

- **TCP:** garantisce un servizio con basso packet loss ratio
- **UDP:** migliore per ritardi e throughput ma peggiore per packet loss ratio

In realtà TCP è paragonabile all'utilizzo di UDP tramite l'utilizzo del **DASH** anche in questo contesto. Essendo un evento live bisogna prevedere una codifica in più bit-rate molto in fretta. Superato questo problema è possibile utilizzare il DASH.

Reti di distribuzione di contenuti (live o salvati)

I server che offrono contenuti vengono realizzati tramite le **Content Distribution Networks (CDN)**. Per poter scaricare video live/salvati è necessario connettersi ad un server ma per gestire gli utenti che possono essere interessati a tale evento è possibile utilizzare un unico server centralizzato (**non conveniente**) oppure utilizzare più server dislocati (**CDN**).

La soluzione adottata da tutti i content provider è quella dei CDN, cioè quella di dislocare i server in tutto il mondo per offrire il servizio agli utenti più vicini a tale server. Queste infrastrutture sono gestite da **terze parti** che si offrono di garantire l'efficienza della struttura stessa (gestire il download in maniera ottimale, posizionare i server in punti ottimali, etc..). Alcune aziende offrono questi servizi come l'Akamai (anche Google si affida alle CDN).

Per realizzare una CDN esistono due approcci (la differenza sta nel grado di diffusione della capillarità con cui vengono distribuiti i server):

- **Enter deep:** si utilizzano più server meno potenti da distribuire in maniera capillare
- **Bring home:** si utilizzano meno server più potenti (meno distribuiti)

VoIP (Voice-over-IP)

Esistono una serie di motivazioni che hanno portato alla creazione del VoIP, uno tra questi è la creazione di un **servizio aggiunto** oppure l'offerta di chiamate gratuite. Un altro motivo (lato provider) è quello di ridurre i costi con il servizio ToIP, cioè trasformare una telefonata che in origine non era VoIP in una chiamata VoIP (evitando di mantenere due reti, una per la telefonia e una per internet).

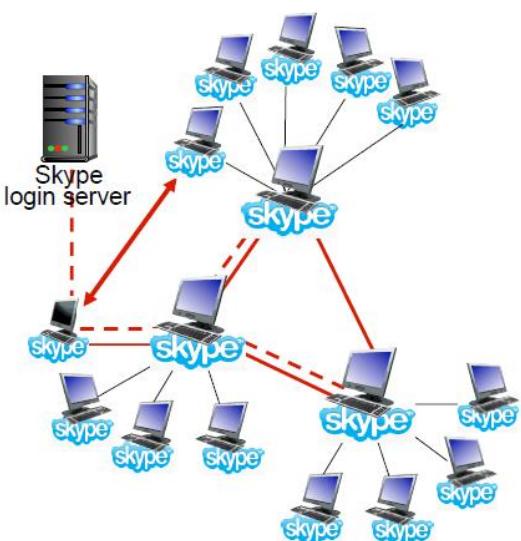
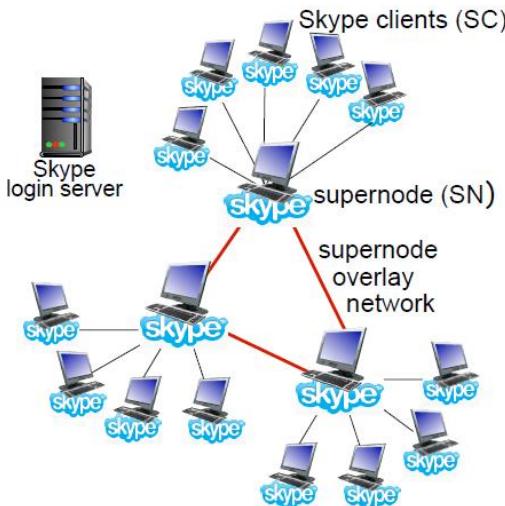
Requisiti: basso ritardo, mantenimento dell'aspetto di “conversazione” (oltre un certo ritardo non si riesce a dialogare, devo dunque mantenere basso questo ritardo)

- < 150 msec: buona conversazione
- > 400 msec: pessima conversazione

Per ottimare l'efficienza di trasmissione si aspettano più campioni per evitare di avere “header” di pacchetti enormi e pochissimi bit di contenuto. Questo implica l'introduzione di un ritardo.

La telefonia VoIP richiede anche la creazione di una **segnalazione** che deve mettere in piedi la telefonata (tono di libero, tono di occupato, squillo del telefono, etc..). Questa segnalazione non sarà identica a quella a circuito, ma sarà di tipo end-to-end.

Skype



Un primo esempio applicazione VoIP è Skype. La rete Skype è composta da peer detti **supernodi** che **mantengono** l'infrastruttura P2P di skype. Non è dunque una rete strutturata (manca la DHT) ma è non strutturata a cui vengono agganciati altri nodi.

Dall'acquisizione di Microsoft della struttura, tale struttura è sempre meno P2P e molte funzionalità dei supernodi vengono inglobati nei server Skype.

Un requisito per essere un supernodo era quello di possedere un indirizzo IP pubblico.

La segnalazione di skype è anch'essa proprietaria (non si conoscono le specifiche funzionalità) ma in linea di massima si tratta di una serie di operazioni di seguito elencate.

Un utente, per contattare un altro utente su Skype, dovrà (oltre a loggarsi) tramite i supernodi ricavare l'indirizzo IP che il peer sta utilizzando e successivamente potrà effettuare la chiamata direttamente a quel peer. Susseguiranno quindi dei messaggi per far squillare e/o per segnalare problemi (occupato, offline, etc..).

Real-Time Protocol (RTP)

Protocollo utilizzato per l'invio dei campioni vocali e dunque non è utilizzato solo per il VoIP. Non si prende il campione vocale e lo si imbusta in UDP ma il protocollo applicazione RTP migliora il servizio cercando di realizzare il recupero dell'errore oppure, in caso di arrivi fuori sequenza (cioè un pacchetto arriva dopo rispetto al successivo) si aggiunge un numero di sequenza ai pacchetti che trasportano campioni vocali per poter rimettere in ordine i pacchetti alla destinazione. Questo perché UDP non offre questi servizi.

<i>payload type</i>	<i>sequence number type</i>	<i>time stamp</i>	<i>Synchronization Source ID</i>	<i>Miscellaneous fields</i>
---------------------	-----------------------------	-------------------	----------------------------------	-----------------------------

L'header contiene:

- **payload type** che serve per indicare il tipo di codifica utilizzato per quei campioni (a.e. PCM).
- **Sequence number type** per svolgere la funzione detta prima (cioè di numerare i campioni per riordinarli)
- **Time stamp** simile al campo precedente come scopo
- **Synchronization Source ID, Miscellaneous fields** realizzano il **multicast** su una rete **unicast** (cioè comunicazioni di gruppo, che IP non offre).

SIP (Session Initiation Protocol)

Questo protocollo serve per la **segnalazione end-to-end**. Nel VoIP non è necessario utilizzare segnalazioni che coinvolgano la rete, perché qui non si ha una rete a circuito ma una rete a pacchetto.

In questo contesto di VoIP serve una segnalazione end-to-end come il “far squillare” il telefono, il segnale di libero, etc..

Questo protocollo standardizzato in RFC 3621 è detto SIP.

Il protocollo serve per:

- segnalare al destinatario che si vuole **iniziare** una conversazione
- segnalare se si vuole fare una videochiamata, chiamata normale o di gruppo
- terminare la chiamata
- **determinare l'indirizzo IP** corrente del destinatario (cioè l'identificativo deve essere sempre abbinato all'indirizzo IP che si ha in quel momento)
- Aggiungere servizi aggiuntivi: aggiunta/rimozione di partecipanti, passare da chiamata a videochiamata e viceversa, messa in attesa.

Esempio di messaggio SIP

```
INVITE sip:bob@domain.com SIP/2.0
Via: SIP/2.0/UDP 167.180.112.24
From: sip:alice@hereway.com
To: sip:bob@domain.com
Call-ID: a2e3a@pigeon.hereway.com
Content-Type: application/sdp
Content-Length: 885

c=IN IP4 167.180.112.24
m=audio 38060 RTP/AVP 0
```

La struttura del messaggio è molto simile a quella di HTTP. Questo messaggio rappresenta un **INVITE**, cioè un invito ad avviare una chiamata. Nella prima riga è contenuto l'identificativo e la versione di SIP utilizzata. Nel corpo del messaggio si utilizza l'header content-type con possibilità simili a quelle del MIME dove informo cosa vi sarà nel messaggio: SDP. L'SDP (Section Description Protocol) è un protocollo generico che serve per descrivere una sessione, cioè specificare le varie caratteristiche di una sessione multimediale (riguardo ai codec, etc.). In particolare, in

questo caso l'SDP viene utilizzato per descrivere il tipo di flusso audio (m=audio).

Questo basta per far squillare un telefono.

SIP registrar

La funzionalità di indexing è gestita dal **registrar** (server di indexing del SIP).

```
REGISTER sip:domain.com SIP/2.0
Via: SIP/2.0/UDP 193.64.210.89
From: sip:bob@domain.com
To: sip:bob@domain.com
Expires: 3600
```

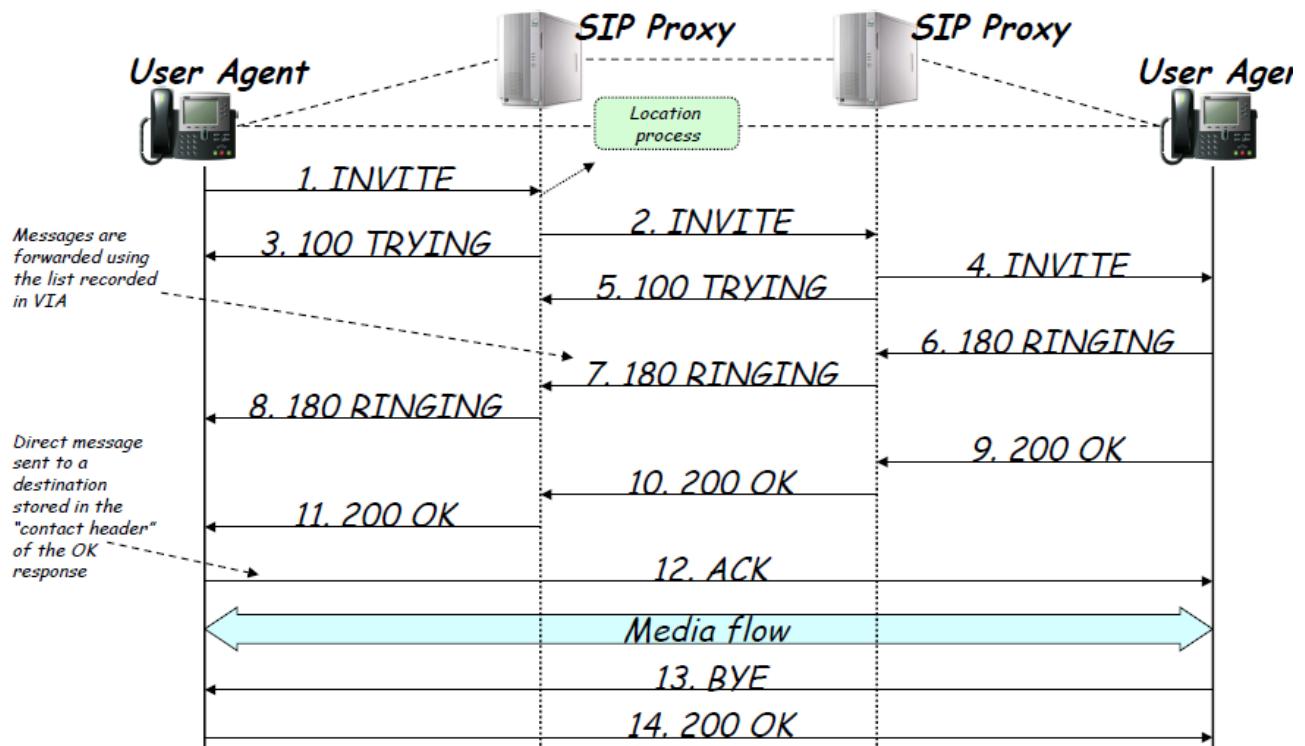
Un utente deve registrarsi nel momento in cui accede alla rete (quindi è necessario un account SIP per ogni utente), e dunque inviare al registrar di quel dominio specifico un messaggio di registrazione, dove comunicherà il proprio

indirizzo IP ed il server salverà tale dato e funzionerà da indexing per la rete.

SIP Proxy

I SIP server forniscono anche la funzione di **proxy** che in questo contesto si comporta come un dispositivo intermedio che viene in aiuto per gestire le telefonate. Sarà dunque il proxy a recuperare l'indirizzo IP della persona da chiamare.

Trapezoide SIP



1. L'utente manda l'invite al SIP Registrar
2. Un location process basato su DNS (solo accennato) si occupa di cercare l'indirizzo IP della persona da chiamare (n.b. il SIP Registrar non cerca direttamente l'IP del destinatario). Il SIP Registrar dovrà dunque capire l'indirizzo IP del **SIP Registrar** del destinatario. Una volta trovato, inoltrerà l'invite a tale Registrar. (funziona come i DNS)
3. Parte il messaggio 100 TRYING dal proxy verso il mittente, per informarlo che il registrar ha inoltrato l'invite verso il registrar del destinatario e si è in attesa.
4. Il SIP Registrar del destinatario inoltrerà a quest'ultimo l'INVITE poiché avrà scritto al suo interno l'indirizzo IP, e a quel punto il telefono squillerà.
5. Il SIP Registrar del destinatario invia un messaggio di successo verso il SIP Registrar del mittente.
6. Quando l'INVITE arriva all'User Agent B e il telefono squillerà, partirà il messaggio 180 RINGING che informa il SIP Registrar e sua volta fino al mittente per informare che l'INVITE ha avuto successo. Si nota che il 180 ringing **non è lo stesso messaggio** inoltrato a tutti, ma solo lo **stesso tipo** di messaggio. Questo perché ogni Register processa il messaggio fino al livello applicativo e dunque lo deve ricreare.
7. Quando si risponde al telefono, si invia un messaggio 200 OK nuovamente fino al mittente tramite i SIP Registrar e la chiamata può partire. È presente un ACK (diverso dall'ACK TCP) perché SIP può funzionare sia su UDP che TCP solo che UDP non ha gli ACK. Infine, BYE e OK terminano la chiamata.

Nella foto c'è scritto SIP Proxy perché, nonostante sia SIP Proxy che SIP Registrar stanno all'interno del server SIP, si vuol far notare che in molti documenti SIP Proxy si usa come sinonimo di SIP Server.

Si nota che alcuni messaggi passano dal SIP Server ed altri no. Ad esempio, nel caso in cui si voglia tariffare gli utenti, il messaggio 200 OK verso il SIP Proxy potrebbe far partire la tariffazione ma nell'esempio precedente il messaggio di BYE passa direttamente tra i due interlocutori, perciò nel caso in cui si voglia applicare una tariffazione bisognerebbe far passare anche il messaggio di BYE dai SIP Server.

H.323

H.323 è stato standardizzato in ambito ITU (telefonia) prima di SIP ed è un altro protocollo di segnalazione. Oggi viene poco utilizzato perché, essendo un antenato di SIP, ed essendo derivato da ITU, risulta molto più complicato.

Conclusioni sul supporto della rete per i multimedia

Approach	Granularity	Guarantee	Mechanisms	Complex	Deployed?
Making best of best effort service	All traffic treated equally	None or soft	No network support (all at application)	low	everywhere
Differentiated service	Traffic “class”	None or soft	Packet market, scheduling, policing.	med	some
Per-connection QoS	Per-connection flow	Soft or hard after flow admitted	Packet market, scheduling, policing, call admission	high	little to none

- **Servizio Best effort:** Il DASH rientra nella prima riga ed è anche il più utilizzato.
- **Differentiated service:** si differenziano i servizi decidendo a quale traffico dare priorità (a.e. audio/video). Questa funzionalità, benché sia interessante, è difficile mettere d'accordo i provider e dunque non viene utilizzata. Il Differentiated service si usa solo all'interno dei singoli provider. (ad esempio Sky e Fastweb hanno partnership e dunque i pacchetti TV di Sky hanno priorità nella rete Fastweb per garantire qualità superiori).
- **Per-connection QoS:** cerca di offrire garanzie su singoli servizi. In questo caso la chiamata può essere rifiutata (segnale di occupato se non vi sono risorse disponibili) questo grazie a una funzionalità aggiuntiva che, in caso il router non possa creare un flusso che rispetti la richiesta, rifiuta la richiesta. Questa funzionalità non è mai stata implementata.