NUMERO: 1250 DATA: 27/10/2014

A P P U N T I

STUDENTE: Arcangeli

MATERIA: Reti di Calcolatori + Domande,Quesiti e Temi

Prof. Bianco

Corso di

RETI DI CALCOLATORI

Prof. Andrea Bianco
Anno Accademico 2013-2014

Studente: Marco Arcangeli

Rete di telecomunicazione.

Def.(Rete di telecomunicazione): un insieme di nodi e canali che fornisce un collegamento tra due o più punti per permettere la telecomunicazione tra di essi.

TIPI DI CANALE:

1. canale punto-punto
2. canali multi-punto (nelle reti non molto usato)
3. canale broadcast

CANALE PUNTO-PUNTO: due soli nodi collegati per comunicare. Non c'è neanche bisogno dell'indirizzo. (Bidirezionale). Vi è quindi un'unica destinazione e per questo è un canale dedicato. Es. due antenne, una trasmette e l'altra riceve.

CANALI MULTI-PUNTO: più nodi collegati ad un unico canale: un nodo master e numerosi slave. Devo usare degli indirizzi per far capire con che apparato (e solo quello) parlo. Ci sono tanti trasmettitori e un ricevitore quando gli slave parlano con il master (gli slave non comunicano tra loro e fintanto uno occupa la linea un altro slave non può palare), mentre c'è un solo trasmettitore e tanti ricevitori quando il master trasmette. È pertanto un canale condiviso in trasmissione.

CANALE BROADCAST: canale condiviso, unico canale con tanti ricevitori. Si devono avere delle regole di condivisione di canali. Il segnale viene trasmesso in tutte le direzioni e arriva anche a chi non è interessato. Un esempio è il professore che parla alla classe: parla a tutti. Se poi parla ad uno studente in particolare, il suddetto studente è l'interessato della comunicazione ma viene recepita da tutta la classe anche se gli altri non sono i destinatari del messaggio.

Tipologia delle reti di telecomunicazione.

È la disposizione dei vari nodi e segmenti (grafo – G). Gli archi possono essere diretti e non diretti (canali bidirezionali).

Definiamo G = (V,A) N = |V| dove V = numero dei NODI
C = |A| dove A = numeri dei CANALI (segmenti)

Vediamo i vari criteri che si sono adottati per le tipologie di reti:

1. topologia a maglia completa
2. topologia ad albero (non molto diffusa)
3. topologia a stalla attiva
4. topologia a stella passiva
5. topologia a maglia (è la più diffusa)
6. topologia ad anello
7. topologia a bus

A MAGLIA COMPLETA (massimo numero di canali per N nodi): C = N(N-1)/2 numero canali per una maglia completa dove N sono il numero di nodi (ordine dell'N^2). Esistono molti percorsi alternativi ma un solo percorso diretto (1 solo canale); esiste una scelta ovvia di percorso di minima distanza. È usata solo quando i nodi sono pochi (come i nodi di commutazione della rete telefonica nazionale, che sono quelli più nevralgici. In questo modo se per qualche motivo si guasta una linea che andava da Roma-Torino o Torino-Milano è presente un collegamento alternativo nell'attesa che si ripari il guasto).
Vantaggio: l'instradamento è veloce ed è molto tollerante ai guasti. Infatti si hanno tante alternative, cioè tanti percorsi possibili per arrivare a destinazione passando da dei nodi di intermezzo invece della strada diretta (come nell'esempio di Roma-Torino).
Svantaggi: è ridondante per i molti cavi e quindi molto costoso.

AD ALBERO (minimo numero di canali per N nodi): C = N-1. L'instradamento è banale, infatti esiste una sola scelta di percorso tra ogni coppia di nodi. C'è un solo percorso possibile tra due nodi. Usata per ridurre i costi e semplificare la stesura dei canali.
Vantaggio: costo ridotto per i pochi canali.
Svantaggio: un guasto mi spezza la rete, quindi è molto vulnerabile ai guasti.

A STELLA ATTIVA(deve essere alimentato): C = N. Il centro stella NON è un nodo. Anche in questo caso ogni nodo ha un solo percorso possibile, ma non unisce due nodi. Tutti i fili infatti partono dalla stella atti-

Spiegazione più chiara: innanzitutto la quantità di traffico smaltibile da una rete è la quantità di bit da "portare fuori" dalla rete (cioè da trasportare). Inoltre la "scelta migliore" non è univoca ma è soggettiva, perché dipende da rete a rete. Se ho una distribuzione non uniforme di traffico sarà meglio avere la distanza minima tra i nodi che scambiano più informazioni, perché visto che comunicando porto via risorse alla rete (cioè quel canale lo posso usare solo io, o cmq una parte di capacità di quel canale la uso solo io) se lo faccio per meno tempo la gestione della rete diventa più efficiente. Mentre per nodi che si scambiano poche informazioni non è necessario avere una distanza minima. Se invece il traffico è distribuito nella rete uniformemente allora sarà rilevante la distanza media dei nodi.

2. SERVIZI NELLE RETI DI TELECOMUNICAZIONE.

Def.(SERVIZI): ciò che viene offerto da un gestore pubblico o privato ai propri clienti al fine di soddisfare una specifica esigenza di telecomunicazioni.

I servizi non sono tutti uguali e, avendo requisiti diversi, hanno un impatto diverso sulla rete.
Un esempio immediato può essere dato da un servizio di file-transfer e da un servizio di telefonia. Il primo deve garantire la correttezza dei file inviati, quindi il servizio deve concentrarsi di più sulla rilevazione degli errori. Se riusciamo a renderlo anche veloce e non troppo lento ancora meglio, ma la velocità non è cmq il requisito fondamentale che, come detto, è la rilevazione di eventuali errori.
Per la telefonia invece si deve porre l'enfasi sulla velocità della comunicazione perché io mi aspetto una risposta immediata dal mio interlocutore. Posso trascurare il controllo su eventuali errori che, nel caso telefonico, saranno dei disturbi sulla linea. Quindi l'enfasi è sulla velocità non sugli errori (che dovranno essere cmq limitati per far si che si capisca cosa si dice).
In definitiva, quindi, i servizi hanno requisiti che dipendono dal tipo di servizio che voglio offrire.

Ci sono due tipi di reti:
1. RETI DEDICATE
2. RETI INTEGRATE

Le RETI DEDICATE offrono un unico servizio. Ha dei vantaggi perché è facile da controllare ed offrendo un unico servizio è di qualità. Per contro però se voglio definire una nuova rete per un solo servizio richiede del tempo e costa molto.
Le RETI INTEGRATE sono delle reti che hanno una molteplicità di servizi (come Internet). Lo svantaggio sta nel fatto che, essendo i servizi concentrati tutti in una rete, ci possono essere ritardi. Tra le reti integrate elenchiamo la ISDN a banda stretta (N-ISDN) e la ISDN a banda larga (B-ISDN).

CLASSIFICAZIONE: dividiamo i servizi in:
1. servizi portanti: cioè il servizio che garantisce la rete. Fornisce la possibilità di trasmissione i segnali tra interfacce utente - rete.
2. teleservizi: fornisce la comunicazione tra due utenti, includendo le funzioni degli apparati di utente, secondo protocolli concordati da gestori pubblici o privati.
   Si distinguono a loro volta in:
   - servizi di base (telefonia, televisione): fornisce all'utente un servizio di telecomunicazione con le minime funzionalità richieste dal servizio stesso.
   - servizi supplementari: migliora il servizio di base con funzionalità aggiuntive. Può essere offerto solo in associazione ad un servizio di telecomunicazione di base e quindi modifica od integra uno o più servizi di base. (Esempio: l'avviso di chiamata, richiamo su occupato, trasferimento di chiamata, numero verde, segreteria telefonica per quanto riguarda la telefonia; video-on-demand per la televisione).

L'informazione nei teleservizi.

l'informazione può essere:
1. bidirezionale simmetrico
2. bidirezionale asimmetrico (es. ADSL)
3. unidirezionale

oppure:
1. punto-punto
2. punto-multipunto

3. TRASMISSIONE NELLE RETI DI TELECOMUNICAZIONE.

Ci sono due tipi di segnali:
1. ANALOGICI
2. DIGITALI (o NUMERICI)

In entrambi i casi l'informazione è trasferita per mezzo di un segnale elettrico. Nel caso dell'analogico il segnale è continuo, limitato e ha infiniti possibili valori. Per i segnali digitali (o numerici) invece il segnale è discontinuo, limitato e con un numero finito di possibili valori.

SEGNALI ANALOGICI: come detto l'informazione assume valori in un insieme continuo. Tali valori vengono rappresentati come variazione continua di un parametro elettrico.
SEGNALI DIGITALI: l'informazione assume un valore numerabile e finito di valori. Il ricevitore tramite un processo di decisione, ricostruisce, ricostruisce il segnale (informazione discreta).

Differenze tra segnale analogico e digitale.

Il segnale analogico viene ritrasmesso così com'è dall'utente iniziale a quello finale comprese tutte le deformazioni (dovute al rumore). Le informazioni possono essere quindi disturbate.
Nel segnale digitale invece il rumore si può eliminare (ovviamente non del tutto, ma in maniera accettabile). Questo processo è detto processo di decisione (come accennato poco fa). Definirò una soglia sotto la quale stabilirò quale numero corrisponda lo stato alto e basso. (Per esempio: segnale sotto la soglia --> valore 0, segnale sopra la soglia --> valore 1). Bit per bit quindi ricostruisco il segnale. Ovviamente ci possono essere degli errori causati dal rumore. Infatti se quest'ultimo è troppo elevato mi distorcerà il segnale analogico e magari una soglia bassa me la fa vedere come alta e viceversa. Per questo si mettono dei margini detti appunto margini di rumore nei progetti dei convertitori A/D per evitare questo fenomeno. Quindi entro certi limiti si può ricostruire il segnale da analogico a digitale. (Es. nella televisione prima dell'avvento del digitale il rumore era rappresentato dalle righe nel programma tv, e gli sfarfallamenti erano ritardi di ricezione, ora nel digitale, se c'è troppo rumore, non si vede più niente perché come detto si crea un errore nel processo di decisione. Le righe di disturbo è una caratteristica dell'analogico, non del digitale.

Si può "tradurre" qualunque segnale analogico in digitale (o numerico). Per esempio la voce (segnale analogico) si può "tradurre" in digitale eseguendo un campionamento del segnale analogico. Si trasforma la onda del segnale vocale in bit (processo detto processo di numerizzazione) tramite il campionamento e poi tramite un altro processo detto processo di quantizzazione lo rendo digitale.
Quindi prendo il segnale analogico, eseguo il processo di numerizzazione campionando il segnale e poi lo quantizzo per renderlo digitale.
Per campionare il segnale ci devono essere delle accortezze per far si che non lo si campioni troppo poco con il rischio di distorcere il segnale (questa volta non a causa del rumore ma per errori nella frequenza del campionamento). Il teorema di Nyquist mi assicura che per ricostruire qualsiasi segnale analogico in digitale devo campionare la sequenza analogica il doppio di volte di quanto è la banda del segnale analogico stesso.
Dopo la campionatura posso quantizzare il segnale per renderlo digitale. Se tutto è andato bene in questa fase avrò soltanto l'errore di quantizzazione che mi porterò sempre dietro. Ci sono dei metodi per rendere minimo questo errore.
Più la banda è grande, più bit mi serviranno per rappresentarlo (per diretta applicazione del teorema di Nyquist). Più frequenze ho più vado in fretta.

Per trasmettere i bit posso farlo in due modi:
1. in PARALLELO (molto usata nel mondo dei calcolatori)
2. in SERIE (dominante nelle reti)

Più importante per noi è la suddivisione del trasmettitore in serie:
1. ASINCRONA
2. SINCONA

ASINCRONA: ogni byte d'informazione viene trasmesso separatamente dagli altri. Il clock di trasmissione però non è uguale (anche se la differenza è minima) di quello di ricezione. I due clock dei due utenti quindi possono essere o più veloci o più lenti rispetto l'uno dall'altro.

La separazione dei flussi è ottenuta usando codifiche diverse. Servono quindi dei codici riconoscibili (ortogonali). I flussi non vengono separati né nel tempo né in frequenza, ma sfruttano segnali ortogonali. Così facendo si ottiene più protezione dai disturbi. L'operazione di ricezione è il prodotto scalare tra segnale ricevuto e sequenza di codice, normalizzando rispetto al numero di bit della sequenza di codice.

SPAZIO: permettono di sfruttare la diversità spaziale del sistema per far coesistere più flussi d'informazione in punti diversi. (es. una cella, cioè una parte di spazio che occupa un'antenna).
Il progetto della topologia della rete può cercare di aumentare la diversità spaziale. L'instradamento può cercare di sfruttare questa multiplazione spaziale (dovuta proprio da questa diversità spaziale) per aumentare la capacità di una rete. A differenza degli altri metodi quindi una multiplazione spaziale può aumentare la capacitò di una rete. Le celle sono un esempio di diversità spaziale.
Per la telefonia mobile: quando la linea cade, molto spesso è perché ci si sposta di cella (o addirittura ci si sposta di cella, la cui cella non fa riferimento all'antenna di cui faceva riferimento la cella prima, quindi ci si sposta anche d'antenna). Se questa cella in cui ci si sposta è però occupata da un altro utente in quel momento non potrà supportare entrambi e quindi cadrà la linea.
Questo modo di suddividere il territorio è coperta su tutto il territorio, infatti ci sono delle celle contigue e a volte addirittura sovrapposte.
Per il wii-fi: condivide nello spazio lo stesso canale per tutti. Un'antenna wii-fi infatti serve per più utenti. Questa suddivisione ha il vantaggio che non cade mai (a meno che non crolli l'antenna, ma è un'ipotesi assai remota) ma ha lo svantaggio che NON ha copertura su tutto il territorio. Infatti il wii-fi è supportato solo nelle aree dove ci sono antenne wii-fi.

Finora abbiamo parlato della multiplazione deterministica, nella quale, momento per momento, deve sapere di cosa hanno bisogno gli utenti ed agire di conseguenza.

Multiplazione statistica.
La divisione in slot dei canali nella multiplazione deterministica è fissa. Però se, per esempio, un canale trasmette al giorno solo 2h invece che 24h c'è uno spreco di 22h, quindi posso pensare di cambiare la condivisione dello spazio. Questa è detta multiplazione statistica: cioè non c'è una partizione fissa dello spazio del canale ma ci si adatta dinamicamente allo spazio del canale.
Quindi supponiamo una rete di dieci utenti con 10 byte di spazio. Non darò a ciascuno 1/10, o meglio gli darò la garanzia minima di 1/10 ma questa non sarà fissa perchè potrà aumentare o diminuire (non oltre l'1/10 cmq) a seconda di cosa fanno gli altri utenti. Se per esempio su 10 ce ne sono collegati solo 2 allora non avrò 1/10 per ognuno come nella multiplazione deterministica (dato che così avrò 8\10 sprecati) ma darò il 50% di capacità della rete ai 2 utenti che in quel momento sono collegati: quindi 5 byte ciascuno. Quando ci sono tutti e 10 però, ognuno avrà la sua "fetta" garantita di 1/10. Se però un utente si scollega allora quell'1/10 che si è rilevato disponibile viene condiviso da tutti gli altri utenti, fino a che l'utente che si era scollegato non ritornerà a collegarsi alla rete.

Condivisione di un nodo.
Prevede l'importantissimo concetto della COMMUTAZIONE:

1. Commutazione DI CIRCUITO (risorse fissate. C'è solo il ritardo di propagazione).
2. Commutazione DI PACCHETTO (condivisione di risorse. Si aggiunge ritardo ad ogni nodo. È, per certi versi, l'opposto di quella di circuito).

Def.(COMMUTAZIONE): processo di interconnessione di unità funzionali, canali di trasmissione o circuiti di telecomunicazione per il tempo necessario per il trasferimento di segnali.
Processo di allocazione delle risorse di rete necessarie per il trasferimento dell'informazione.

COMMUTAZIONE DI CIRCUITO: come detto c'è solo il tempo di propagazione su cui non si può fare niente: è infatti un limite fisico. T_propag = (distanza/velocita_luce). Il tempo di attraversamento del nodo è trascurabile anche perché le considero come entità puntiformi.
Il circuito è di uso esclusivo dei due utenti per tutta la durata della comunicazione. Se io finisco le risorse su un determinato filo non posso mettere in comunicazione più nessuno fino a che non si libereranno delle risorse (il che equivale a dire che la comunicazione tra due utenti è finita).
È vantaggioso per il traffico voce (sorgenti non intermittenti), meno per il traffico tra host (sorgenti altamente intermittenti). Per intermittenti si intende se è possibile fare una predicibilità del traffico (cosa che ovviamente non si può fare per gli host ma che si fa nella telafonia).

non solo dai diversi tempi di elaborazione ed arrivo dei pacchetti, ma anche dagli altri utenti. Infatti se molti pacchetti di utenti diversi devono prendere la stessa strada si può creare questo sovraffollamento. Non mi basta quindi costruire un nodo che abbia tante uscite quanti ingressi perché anche se ho N ingressi ed N uscite se i pacchetti che mi entrano in ingressi diversi devono essere smistati alla stessa uscita si creerà coda. In definitiva nella commutazione a pacchetto non posso garantire nulla (non so neanche che strada farà il pacchetto, figurarsi dire dei ritardi). A questa probabilità di errore si associano quindi controlli per individuarli. Se si presentano si rinvia la richiesta e si riceve nuovamente il pacchetto questa volta, si spera, senza errori (ovviamente questo rinvio del pacchetto\i richiede ulteriore tempo e allocazione di risorse in più del dovuto rispetto alla non ricezione dell'errore).

L'informazione di un utente può essere frazionata in molti pacchetti di dimensione fissa o variabile.

Tempo_trasmiss = (dimensione_pacchetto / velocità_trasmiss), varia da canale a canale.
Tempo_propagaz = (distanza_in_metri / velocità_propagaz) di un singolo canale.

In ogni modo, nel migliore dei casi, pago un ritardo di rice-trasmissione e di elaborazione. Cmq il ritardo più significativo è quello di rice-trasmissione rispetto a quello di elaborazione.
Generalmente il tempo di coda prevale, ma se non c'è prevale quello di rice-trasmissione. Anche nella commutazione di circuito c'è ritardo di coda ma è meno rilevante perché non c'è memorizzazione (lo storing) che invece è presente nella commutazione di pacchetto.

Quindi i principali ritardi sono:
SU OGNI CANALE SU CUI È TRASMESSO (quindi se trasmetto su k canali, avrò k*ritardo):

1. ritardo di RICE-TRASMISSIONE (generalmente costante): in funzione della dimensioni in bit del pacchetto e della velocità di trasmissione del canale.
2. ritardo di PROPAGAZIONE (generalmente costante): in funzione delle dimensioni in metri del canale.

SU OGNI NODO DI COMMUTAZIONE:

1. ritardo di ELABORAZIONE (generalmente costante): in funzione delle velocità con cui si eseguono le procedure di instradamento e di eventuali controlli sul pacchetto.
2. ritardo di CODA (non costante, perché come detto dipende dagli altri utenti, cioè dalle sorgenti impulsive, il cui traffico non può essere predetto a priori): in funzione del traffico generato da tutti gli utenti.

La lunghezza dei pacchetti è determinata da:

1. possibilità di parallelizzazione (pipeline).
2. ritardo di pacchetizzazione.
3. percentuale di informazione di controllo.
4. probabilità di errore.

PARALLELIZZAZIONE: invece di immettere il pacchetto intero nella rete lo divido in tre o più parti. Così facendo, quando ho ricevuto il primo pacchetto con il dovuto ritardo di propagazione, posso subito incominciare ad elaborarlo senza aspettare la fine di tutto il pacchetto diviso in tre pezzetti. Subito dopo aver ricevuto la prima parte quindi la elaboro immediatamente. In questo modo si accorcia il ritardo di ricezione dei nodi, o meglio, si abbassa la latenza di attraversamento dei nodi. È chiaro che più i pacchetti sono piccoli meglio è. Per ridurre la latenza si può cmq intervenire aumentando la velocità di trasmissione.

PACCHETIZZAZIONE: dipende dalla dimensione del pacchetto. Più è grande più ci si mette e più aumento il ritardo di campionamento. Questo è il problema predominante per il traffico voce su reti a commutazione di pacchetto. Pacchetti brevi riducono il ritardo di pacchetizzazione. Per essere efficiente, il sistema deve avere nella rete pacchetti piccoli. Però se questi pacchetti sono troppo piccoli rispetto all'header non mi conviene. Queste due esigenze devono essere bilanciate.

% INFORMAZIONE DI CONTROLLO: ritardo dovuto all'intestazione (o header) dei pacchetti.
Pacchetti lunghi riducono la percentuale d'informazione di controllo:

$$p / (s + p) \qquad \text{(frazione d'informazione di controllo)}$$

dove:
p = dimensione in bit del PCI
s = dimensione in bit dell'SDU

l'etichetta, come detto, è univoca e cambia ad ogni link, cioè ad ogni passaggio di nodo ed una connessione è identificata logicamente da un'etichetta.

Esiste quindi un accordo preliminare tra i due interlocutori e il fornitore del servizio (punto 1). Pacchetti diversi con stessa sorgente seguono sempre lo stesso percorso fissato (punto 2).

In definitiva un circuito virtuale:
1. richiede una dichiarazione da parte dell'utente,
2. assegna un percorso fisso per tutta la durata della comunicazione,
3. etichetta i pacchetti ad ogni nuovo link (cioè ad ogni nodo).

Inoltre:
1. controlla di più il traffico,
2. è vincolato da un percorso fisso.

Differenze tra circuito virtuale e commutazione di circuito.
Il servizio su circuito virtuale (e quindi a pacchetto) non è equivalente al servizio in reti a circuito in quanto, in quest'ultima, non si allocano statisticamente risorse a una comunicazione.

Differenze tra circuito virtuale e datagram.
Nel datagram non esiste la suddivisione della comunicazione nelle tre fasi descritte prima perché non c'è nessun accordo preliminare sulla fornitura del servizio. Infatti i pacchetti sono visti come indipendenti, per cui, pacchetti diversi con sorgente e destinazione uguale possono prendere strade diverse a seconda del cambiamento delle risorse della rete.
In caso di guasto nel datagram sceglierò un percorso alternativo, quindi bene o male si riesce a gestire il guasto; nel caso del circuito virtuale invece la gestione del guasto è più complessa. Ci sono tre possibili soluzioni:
1. rifare il circuito virtuale dall'inizio,
2. utilizzare un sistema di backup,
3. se cade il link si va a vedere che pacchetti avevi e li reindirizzi. Se però i pacchetti sono tanti conviene ricostruire il circuito perché il tempo sarebbe troppo elevato.

(s)Vantaggi del circuito virtuale rispetto al datagram.
Si deve mantenere la sequenza.
Minor variabilità dei ritardi.
Instradamento solo in fase di apertura di connessione.

L'ultimo (s)vantaggio richiede qualche parola in più: si potrebbe vedere anche come il modo in cui il circuito virtuale gestisce l'inidirizzamento. Ogni nodo ha nel datagram una propria tabella di instradamento perché su ogni pacchetto esegue il processo di instradamento. Ogni nodo deve contenere quindi nella sua tabella (detta tabella di instradamento) tutti gli indirizzi di tutti i possibili utenti della rete. Non solo quelli connessi, ma proprio tutti. Si intuisce che sarà una tabella molto grande e che per consultarla ci vorrà del tempo non trascurabile (il tempo di elaborazione e di instradamento). Nel modello di commutazione di pacchetto a circuito virtuale invece ogni nodo non ha bisogno di avere l'intera tabella di instradamento dato che il percorso è prefissato all'inizio della comunicazione (punto 2). Solo il primo nodo della comunicazione avrà bisogno di consultare la tabella di instradamento più grande, da qui in poi (fino alla fine della comunicazione) ogni nodo del circuito virtuale si servirà di un'altra tabella molto più piccola, detta tabella di forwording che sarà composta da solo gli utenti che sono attivi in rete e inoltre che interessano a quel particolare tipo di nodo. Cioè solo quelli che tramite quel nodo sono raggiungibili. Si intuisce che la tabella di forwarding sarà molto più piccola di quella di instradamento del datagram e quindi mi ridurrà il tempo di consultazione della stessa avendo ritardi minori. In questa tabella di forwarding è anche scritto l'etichetta del pacchetto corrispondente che cambia con le regole descritte precedentemente (punto 3).

Due tipi di circuiti virtuali:
1. PVC (Circuiti Virtuali Permanenti): poco usati. Creati tramite il sistema di gestione della rete. Due utenti anche quando non comunicano più avranno sempre attivo questo canale PVC.
2. SVC (Circuiti Virtuali Commutati): più usati. Creati su richiesta dell'utente tramite segnalazione alla rete, in tempo reale.

Nota: per quanto riguarda la qualità del servizio la legge stabilisce che deve essere garantita una valida copertura nel territorio. Chi riesce a garantire una copertura maggiore rispetto ad altri gestori vince l'asta indetta dal governo e viene autorizzato ad effettuare il servizio richiesto.
Per tutto questo, quindi servono modelli matematici per caratterizzare le richieste di servizio (cioè in linea di massima sapere cosa gli utenti intendano fare), descrivere l'interazione tra attività e risorse, calcolare la qualità del servizio fornito.

Caratterizzazione delle sorgenti di informazione.
Sono di due tipi:
1. Analogiche: voce, video. Caratterizzate dalle loro caratteristiche spettrali.
2. Numeriche: dati, voce (numerizzata), video (numerizzato). Caratterizzate dalla velocità di cifra e dalla loro impulsività (burstiness = vel_picco / vel_media, più è grande questo rapporto più la sorgente è impulsiva e quindi impredicibile).

SORGENTI NUMERICHE: sono divise in due tipologie:
1. a velocità costante (CBR): voce numerizzata, videoconferenza.
2. a velocità variabile (VBR): video MPEG, file transfer.

CBR (Constant Bit Rate): sono caratterizzati da tempo di arrivo uguali. Le dimensioni dei pacchetti possono essere uguali ma anche diverse. L'importante (per essere una sorgente CBR) è che mi arrivino tutti con lo stesso ritardo temporale.

$P_i = P$ per ogni i
$T_i = T$ per ogni i
$bit\_rate = P_i / T_i$ = costante.

Dove P è la dimensione del pacchetto e T il tempo che impiega per raggiungere la destinazione.

L'unica impredicibilità riguarda l'inizio e la fine della chiamata, e la durata della stessa. Non so inoltre quando passerà tra una telefonata e l'altra.

Nota: la voce umana non è CBR ma VBR, questo perché è impulsiva, non è costante nel tempo. Possiamo infatti fare pause durante il discorso, oppure stare in silenzio. È il campionamento con continuità che crea una sorgente CBR perché, come detto, durante la chiamata viene campionato anche il silenzio e quindi eventuali pause che una persona necessariamente compie. Se si mette una rete più complessa in cui quando ci sono silenzi non si abbia comunicazione, non campionando così le pause ed eseguendo una compressione di silenzi, allora questa diventerà una sorgente VBR.

VBR (Variable Bit Rate): in questo caso distinguiamo due velocità. Quella media e quella istantanea. Nel caso del CBR erano uguali. Il valore della velocità di picco è il valore più alto della velocità misurata ad ogni coppia di bit. I pacchetti possono avere uguale oppure diversa dimensione. L'impredicibilità dell'inizio e della durata della comunicazione è uguale a quella del CBR, in più, in questo caso del VBR, non so neanche caratterizzare i T_i, essendo il bit rate variabile. Quindi le sorgenti VBR sono più difficili da caratterizzare perché sono molto più impulsive rispetto alle CBR, infatti il T_i è di difficile previsione.

Nota: la velocità con cui trasmetto i bit è dipendente del mezzo del canale trasmissivo. Per vedere l'effettiva velocità istantanea devo vedere il processo della generazione di due pacchetti consecutivi.
Se due pacchetti vengono generati l'uno dopo l'altro, allora la velocità di picco è uguale a quella del canale.

Es.:

RETI TELEMATICHE.

→ Lezione 02. Architetture di protocolli.

1. ARCHITETTURE DI PROTOCOLLI.

Questo argomento per ora sarà un po' astratto ma sarà la base che ci permetterà di capire il funzionamento della rete Internet discusso più avanti. Queste nozioni saranno molto importanti per la comprensione futura.

CCITT (Consultative Committee for International Telephony and Telegraphy).
Comunicazione: trasferimento di informazioni secondo convenzioni prestabilite.

Ci deve essere un linguaggio con delle regole per permettere che la comunicazione sia chiara. Le modalità di comunicazione quindi necessitano di un modello di riferimento. Nel massimo livello di astrazione il modello di riferimento specifica una architettura di rete.
L'architettura di rete in questione definisce gli oggetti usati per descrivere:

1. il processo di comunicazione
2. la relazione tra tali oggetti
3. le funzioni necessarie per la comunicazione
4. le modalità organizzative delle funzioni

Per fare ciò si usano delle architetture stratificate. Si ha una gerarchia quindi. Le reti che stanno sopra ad altre sarebbero inutili senza le loro reti sottostanti perché non sarebbero in grado di comunicare. È quindi una dipendenza funzionale quella che lega questa gerarchia di rete.
A livello di funzionalità uno strato aggiunge complessità alla rete. Ogni strato può "parlare" solo con strati adiacenti (cioè con quello sopra di lui o sotto di lui, non può scavalcare degli strati). Per far si che questi strati adiacenti possano comunicare si sono create delle interfacce (SAP). Un esempio di strato di rete che abbiamo già visto è costituito dall'instradamento dei pacchetti in rete.

Una caratteristica fondamentale è rappresentata dalla separazione di funzioni. L'utente con il suo terminale (denominato host) rappresenta il luogo dove le informazioni arrivano (o partono). La rete sottostante (denominata subnet) ti permette di accedere alla rete (per esempio la rete del Politecnico: www.polito.it). In questa subnet ci si trasferiscono i pacchetti quindi ci devono essere strati per l'instradamento per le gestioni di controllo e così via. Se devo spedire però dei dati in una subnet che non sia la rete del Politecnico io per raggiungere un'altra subnet ho bisogno di un dispositivo (denominato router) che mi permette di collegarmi ad un'altra subnet. In questo router ci deve essere anche qua lo strato dell'instradamento per trasferire pacchetti, deve inoltre controllare errori, gestirli se ci sono, ecc..

Il modello che vedremo è l'architettura stratificata OSI (Open System Interconnection) e ci concentreremo principalmente sui meccanismi di comunicazione all'interno degli strati del modello stesso. Oggi è caduto in disuso ma il capire il suo funzionamento ci spianerà la strada per capire meglio ed approfondire il modello che oggi è il più usato al mondo: ARPAnet, il modello d'architettura stratificato di Internet (che vedremo più avanti).
Nella storia delle tlc si sono creati numerose architetture stratificate differenti, oltre all'OSI e ad ARPA i più importanti sono stati il DECNET e lo SNA (il modello d'architettura dell'IBM). Il problema è che finché si sta dentro la propria architettura i vari strati riescono a comunicare, ma se si vuole far comunicare uno strato del DECNET per esempio con quello dello SNA, questi non riescono a comunicare. C'è un problema di compatibilità.
Per far cercare di trovare un modello che sia compatibile con tutte queste architetture si è sviluppato l'OSI, ma poi, come detto, è stato soppiantato da quello d'Internet: ARPAnet.

Def.(PROTOCOLLI): descrizione formale delle procedure adottate per assicurare la comunicazione tra due o più oggetti dello stesso livello gerarchico.
In altre parole sono un insieme di regole da seguire per effettuare una corretta comunicazione.
Un protocollo è definito dalla:

1. semantica: insieme di comandi e risposte (dà origine agli algoritmi),
2. sintassi: struttura di comandi e risposte (dà origine ai formati),
3. temporizzazione: sequenza temporale di comandi e risposte ben definite.

comunicare con lui mi occorre quindi qualcosa per comunicare con quel nome, il titolo, e l'indirizzo mi conferisce questa informazione, l'indirizzo. La funzione che associa il nome con il numero è l'elenco telefonico, mentre per l'OSI è la directory). Quindi l'N-directory è la funzione che mi mette in relazione la SAP con l'indirizzo dell'entità con cui deve comunicare.

Nota: l'indirizzo è associato alla (N-1)SAP che richiede di comunicare con la N-entità dello strato superiore. Quindi ad una N-entità e associato un (N-1) indirizzo.

La funzione che mette in relazione le SAP da cui una stessa entità esegue e riceve il servizio si chiama mapping: traduce gli indirizzi di livello N ad indirizzi di livello (N-1) e viceversa. (ART: protocollo di mapping che traduce un indirizzo di livello 2 in un indirizzo di livello 3; ARP: protocollo di mapping che traduce un indirizzo di livello 3 ad un indirizzo di livello 2).
La funzione di mapping permette dato l'indirizzo (N-1), corrispondente all'(N-1)-esima SAP, di vedere quale sia l'N-entità che può ricevere quel servizio; ma non l'N-SAP. Cioè non c'è una funzione mapping diretta tra SAP diverse. Quindi la funzione mapping prima individua l'N-entità poi si mette in contatto con l'N-SAP.

Per collegare logicamente le SAP si usano delle connessioni. Questa connessione può essere punto-punto oppure multi-punto. Le SAP utilizzano queste connessioni per poter fornire all'utente più servizi contemporaneamente. (Un esempio è rappresentato da Facebook. Infatti la pagina principale è la Home e su questa pagina ci sono un sacco di collegamenti ad altri link, come, ad esempio, ad un video su YouTube. In questo caso è la SAP che ti mette in contatto con l'indirizzo del video di YouTube).
Importante è il concetto di CED (Connection and Point – una specie di punto di connessione finale). Tutte le connessioni che riceve la SAP devono essere differenziate. Il CED è un puntatore che serve a differenziare tutte le informazioni che la SAP riceve. Anche i CED hanno degli identificatori che sono "parenti" delle etichette (o header) visti prima.
Come già detto il trasferimento delle informazioni può essere:

1. Connection-oriented (CO): si stabilisce un accordo preliminare tra rete ed interlocutori, poi si trasferiscono i dati ed infine si rilascia la connessione.
2. Connectionless (CL): i dati vengono immessi in rete senza un accordo preliminare e sono trattati in modo indipendente.

In questi casi le entità che devono comunicare chiedono il servizio alle SAP. Il fornitore del servizio è per loro una specie di "scatola nera". Se l'(N+1)-entità richiede un servizio in quella scatola per lei sono compresi tutti i livelli da 1 ad N (se siamo all'entità 5, lei vedrà nella scatola tutti i livelli dall'1 al 4), più i mezzi trasmissivi.
Le connessioni possono essere multiplate: cioè tante connessioni di livello N su connessioni di livello (N-1) o viceversa. (Multiplazione: da tanti ad uno; demultiplazione: da uno a tanti).

Creazione di una PDU (Protocol Data Unit). Per svolgere le sue funzioni ogni strato deve aggiungere al PDU delle informazioni (PCI: Protocol Control Information). Quindi ogni strato "imbusta" il pacchetto che gli arriva ci aggiunge il suo pezzo e lo invia allo strato successivo.
PCI(1) | PDU
PCI(2) | PDU →→→ dove in questo caso la PDU è il PCI(1) | PDU di prima
e così via.
Si intuisce che al livello più alto (da dove partono i dati) il pacchetto sarà più piccolo, mentre man mano che si scende diventerà sempre più grande perché ogni strato aggiunge un pezzo, fino ad arrivare al livello fisico, dove la lunghezza è massima. Da notare che ogni strato aggiunge il suo pezzo "nella sua lingua", cioè usando il suo protocollo definito di quello strato. Infatti quando il pacchetto percorre il procedimento opposto, cioè quando viene spedito al ricevente, lo strato fisico ricevente (che adesso è il primo che gli arriva il pacchetto) è in grado di comprendere solo quello che gli ha scritto lo strato fisico del sistema trasmittente, ed una volta interpretato viene eliminato (tanto risulterebbe incomprensibile per il livello successivo). Si osserva che la lunghezza in questo procedimento di ricezione è uguale a quello della trasmissione, perché il pacchetto arrivo al livello fisico (strato 1) con la massima lunghezza, e via via che gli strati interpretano il loro pezzo e lo eliminano il pacchetto diventa sempre più corto, fino a raggiungere la lunghezza che si aveva al trasmittente nel livello 7.
Questi pacchetti se diventano tanto lunghi si segmentano. Avviene tra strati adiacenti (di solito tra strato 1 e strato 2). Se si evita è meglio perché servirà una numerazione dei segmenti e quindi altri bit di controllo, ma se il pacchetto è troppo lungo non c'è altra soluzione. Lo strato che segmenta deve ricompattare. Non può essere fatto da un altro strato (anche perché attuano protocolli diversi). Questo significa che se un si-

Supponiamo di avere un dispositivo che ha N entrate e ad ogni entrata mi permette di trasferire 1 Gbit allora avrò, con un ugual numero di uscite, un dispositivo che (più o meno) mi permette di gestire un flusso di N-Gbit. Il più o meno deriva dalla scelta dell'instradamento. Infatti, nel peggior caso possibile, se tutti i pacchetti dei miei N ingressi sono instradati tutti su un'unica uscita, allora avrò un problema di congestione, perché quella sola uscita non potrà supportare tutto quel traffico. In questo modo si creerà congestione perché i pacchetti saranno costretti ad infilarsi in coda e quando la memoria del dispositivo è piena, i pacchetti che arriveranno verranno scartati.
Il problema della congestione è cmq transitorio, perché se così non fosse, sarebbe un problema del progetto della rete. Quindi in definitiva la congestione è dovuta al "riempimento" dei nodi.

STRATO 4 (Strato trasporto – Transprot Layer).
Fornisce alle entità di strato sessione le connessioni di strato trasporto. È il primo strato protocollare che collega direttamente i due utenti (senza l'uso della rete). Gli strati precedenti (1, 2 e 3) per questo strato sono delle "scatole nere" nelle quali "entrano ed escono i bit", nulla di più. Quindi questo strato sarebbe l'equivalente del livello 2 per gli utenti, se equipariamo quest'ultimi alla rete.
Funzioni.

1. Controllo errore.
2. Controllo di flusso.

Queste due funzioni possono essere ridondanti.

3. Controllo di sequenza: la più importante di questo strato. È un "parente stretto" del controllo d'errore. Se mi arrivano dei pacchetti numerati 1, 2, 3, 5 saprò che la sequenza è sbagliata perché i pacchetti non sono in sequenza. La causa dell'errore però non la potrò sapere, perché potrebbe essere per una congestione o per un effettivo errore sui dati di quel pacchetto. Quindi al livello 4 (trasporto) non è garantita la sequenzialità dei pacchetti.
   Oggi come oggi questo strato è molto critico e delicato da gestire, perché i terminali utente sono molto eterogenei tra di loro. Questo causa, delle volte, dei collegamenti tra due terminali molto diversi, dal punto di vista della velocità, delle novità portate nel tempo (come nuovi sistemi operativi o applicazioni) e possono causarsi dei problemi di sincronizzazione e di compatibilità delle velocità del trasferimento dati.

Fino a qui le funzioni e gli strati sono in linea di massima identici a quelli utilizzati dalla rete Internet. I tre strati che rimangono da elencare del modello OSI nella rete Internet sono fusi assieme.

STRATO 5 (Strato Sessione – Session Layer).
Assicura alle entità di presentazione una connessione di sessione. Organizza inoltre il colloquio tra le entità di presentazione e struttura e sincronizza lo scambio di informazioni in modo da poterlo sospendere, riprendere e terminare ordinatamente mascherando le interruzioni dello strato trasporto.
Introducendo un esempio si capisce meglio: se io devo trasmettere 10 file ed ad un certo punto cade la linea (o la connessione) devo teoricamente ricominciare da capo, ritrasmettendo il primo dei 10 file, perché non ho tenuto conto dei file che avevo già trasmesso. Il livello 5 svolge proprio questa funzione: ti permette di ricordare i file già trasmessi così che se cade il collegamento tu possa riprendere a ritrasmettere dal file in cui la connessione è caduta e non da capo. Quindi questi punti di sincronizzazione (EOF – End Of File) ti permettono di ricordarti quanti e quali file sono già stati trasmessi all'interno del flusso dati.

STRATO 6 (Strato (Rap)Presentazione – (Rap)Presentation Layer).
Risolve i problemi di compatibilità per quanto riguarda la rappresentazione dei dati da trasferire. Risolve anche i problemi relativi alla trasformazione della sintassi dei dati e può fornire servizi di cifratura delle informazioni. Per esempio se prendiamo un byte che è composto da 8 bit, ho 2 modi per rappresentarlo: con la cifra più significativa a dx oppure a sx. Pertanto devo essere in grado di capire i vari formati dei pacchetti che mi arrivano.

STRATO 7 (Strato Applicazione – Application Layer).
Fornisce ai processi applicativi i mezzi per accedere all'ambiente OSI. Si interfaccia con l'utente, quindi. È lo strato che permette al browser (cioè un software: Google Chrome, Mozilla, Safari, Explorer) di navigare e connettersi in rete, interfacciandosi con l'applicativo software. Di solito c'è un protocollo applicativo per ogni servizio utente. Esempi di servizio sono la posta elettronica (X.400), il terminale virtuale (VT), il trasferimento di file (FTAM) ed altri.

Intestazione di pacchetti.
I bit di parità si introducono tra le informazioni di controllo all'interno delle PDU. Sovente i bit di parità sono calcolati con un codice ciclico e sono detti CRC (Cyclic Redundancy Check).
Quindi il bit di parità è un elemento che si "inserisce nella busta". Ogni strato, come già precedentemente detto, aggiunge il suo PCI. (In una rete affidabile nell'intestazione oltre agli indirizzi ci sono i bit di parità e i bit di numerazione dei pacchetti nell'eventualità che qualcuno vada perso).

PROTOCOLLO ARQ.

C'è un controllo coniugato su una connessione di: errori, flusso e sequenza. Si introducono dei bit di numerazione tra le informazioni di controllo all'interno delle PDU. Devo numerare i pacchetti per saper dire quale pacchetto è eventualmente errato. Questa numerazione è contenuta sempre nella nostra "busta".

Ho tre tecniche ARQ:
1. Stop and wait (Alternating Bit)
2. Go back N
3. Selective repeat

Ora descriveremo queste tre tecniche di un ambiente di comunicazione unidirezionale.

STOP AND WAIT: trasmetto, mi fermo e aspetto la conferma del ricevente. Gestisce un pacchetto alla volta. Il ricevente mi deve avviare la conferma, detta ACK (da acknowledgment), numerata con il pacchetto che deve ricevere subito dopo questo, oppure con lo stesso numero del pacchetto ricevuto. Ci si mette preventivamente d'accordo. Quindi ad esempio se trasmesso il PDU(1) mi aspetto un ACK(2) per la prima scelta ed un ACK(1) per la seconda scelta. Se la numerazione ha un solo bit disponibile (quindi 0 od 1) il protocollo è detto alternative bit protocol.
Trasmettitore: invia un pacchetto PDU facendone una copia memorizzandola sul suo buffer. Aspetta quindi una risposta del destinatario ricevente. Se un pacchetto ha un errore, quest'ultimo viene ignorato dal ricevitore, cioè è come se si "buttasse via". Se accade ciò, la ricevuta di ritorno non arriva. A questo punto quindi il trasmettitore, dopo un certo lasso di tempo (si parlerà successivamente di questo punto un po' delicato), lo ritrasmette automaticamente. Questo lasso di tempo è regolato da un timer (detto tempo di timeout), che quando scade, fa sì che si ritrasmetta il pacchetto.
Se invece l'ACK arriva, il trasmettitore controlla se è corretto e che il numero dell'ACK stesso sia quello giusto. Se questo avviene, trasmette il pacchetto successivo.
Se invece l'ACK arriva ma ha un errore (oppure è congestionato e non arriva prima della scadenza del timer) lo "butta via" ed aspetta quello giusto fino allo scadere del timer.
Se l'ACK era "affetto" di errore, avendolo scartato dopo che è passato il tempo del timer il trasmettitore ritrasmette il pacchetto e aspetta nuovamente l'ACK.
Se invece l'ACK era congestionato la faccenda è un po' più intricata: infatti il trasmettitore che nel frattempo ha ritrasmesso il pacchetto si aspetta un ACK del pacchetto nuovo (cioè quello ritrasmesso). Quando gli arriva l'ACK vecchio (quello del primo pacchetto) il trasmettitore crederà sia l'ACK del pacchetto ritrasmesso. Invece dopo un po' si vedrà arrivare un altro ACK: quello del pacchetto ritrasmesso la seconda volta (se il pacchetto non è andato perso ed e senza errori). Per questo serve una numerazione. L'ACK avrà pertanto lo stesso numero, ed essendogli già arrivato prima, il trasmettitore non considera questo ACK duplicato e manda il secondo pacchetto.
Quando si è completato un ciclo (cioè trasmetto il pacchetto e mi arriva il l'ACK, tutto esente da errori) allora il trasmettitore può inviare il pacchetto successivo e cancellare dal suo buffer di memoria il pacchetto che ha inviato precedentemente. Il processo si ripete così.
Ricevitore: è più semplice da gestire perché guarda solo se ci sono errori nella PDU e controlla la sua numerazione. Se è tutto corretto invia l'ACK, mentre se c'è un errore scarta il pacchetto e aspetta il termine del timer del trasmettitore per ricevere nuovamente il pacchetto.

che manda l'ACK(3) che viene ricevuto correttamente dal trasmettitore. Quindi quest'ultimo passa al PDU(3), questo viene ricevuto correttamente e anche l'ACK è emesso e ricevuto senza errori. Il trasmettitore invia quindi il PDU(4), che per il ricevitore è il PDU(0). Questo si "perde" nella rete e quindi il ricevitore non fa assolutamente niente perché non gli è arrivato il PDU. Il punto cruciale è questo. Se in questo preciso momento arriva al trasmettitore l'ACK(1) che si era perso prima, il trasmettitore crederà che il PDU(4) (che per il ricevitore, ricordiamo, è il PDU(0) e quindi l'ACK corrispondente è proprio l'ACK(1)) sia stato correttamente ricevuto dal ricevitore perché elabora l'ACK(1) che si aspettava dal ricevitore ma non sa che questo era il vecchio ACK(1) inviato per la conferma del primo PDU(0). Quindi la finestra del trasmettitore si sposta sulla "casella" 5. A questo punto le finestre del trasmettitore del ricevitore sono sfasate:

						√		
TX	0	1	2	3	4	5	6	7
	PDU(0)	PDU(1)	PDU(2)	PDU(3)	PDU(4)	PDU(5)	PDU(6)	PDU(7)
RX	0	1	2	3	0	1	2	3
	PDU(0)	PDU(1)	PDU(2)	PDU(3)	PDU(0)	PDU(1)	PDU(2)	PDU(3)
				√				

Il trasmettitore invia il PDU(5) aspettandosi come risposta l'ACK(2). Il ricevitore si vede arrivare il PDU(1) ma gli manderà sempre l'ACK(1), questo viene pertanto ignorato dal trasmettitore in quanto già ricevuto prima e dopo il timeout rimanda il PDU(5) (che per il ricevitore è il PDU(1)) che però non sarà confermato dall'ACK(2) del ricevitore che manda sempre l'ACK(1) che sarà scartato dal ricevitore. Il protocollo si blocca. Per prevenire questi blocchi si implementa il protocollo in modo che dopo un tot di cicli uguali si resetti la comunicazione. Inoltre per prevenire queste ricezioni di ACK(1) persi se un pacchetto nella rete è "in giro da tanto tempo" nella rete lo si provvede ad eliminare.
Si osserva che se ci fosse stata una numerazione più grande per il ricevitore, questo non sarebbe accaduto perché anche se fosse arrivato l'ACK(1) nel preciso momento descritto dall'esempio questo sarebbe stato ignorato perché sarebbe stato errato. Quindi maggiore è il numero di bit per la numerazione, minore sarà il numero di perdite del protocollo. Si potrebbe obbiettare che così però si è spostato il problema più avanti, infatti quando la numerazione finisce (anche se molto lunga) se sia ha la sfortunata (seppur remota) coincidenza di ricevere l'ACK al momento sbagliato il protocollo si impalla lo stesso, ma le probabilità sono minime e, in più, come detto, dopo un lasso di tempo i pacchetti che sono presenti nella rete vengono eliminati (riducendo ulteriormente la probabilità di blocco protocollare).

GO BACK N: cambia solo il trasmettitore, complicando i suoi compiti, mentre il ricevitore svolge esattamente le stesse funzioni dello stop and wait. Quest'ultimo protocollo infatti può essere poco efficiente a causa di elevati ritardi di attesa delle conferme. Il Go Back N permette la trasmissione di più di una PDU prima di fermarsi in attesa delle conferme, migliorando così le prestazioni.
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 (cioè gli ACK). La dimensione della finestra W_T è limitata dalla quantità di memoria allocata del trasmettitore. Infatti i pacchetti inviati devono essere memorizzati nel qual caso uno avesse un errore o si perdesse e si necessitasse di una ritrasmissione. Inoltre W_T rappresenta il numero massimo di PDU contemporaneamente presenti sul canale o in rete. Posso gestire cmq un solo PDU per volta. La dimensione della finestra di ricezione è sempre unitaria (W_R=1).

$$\text{Bit\_Rate} = (\text{Dim. PDU}) / \text{RTT}$$

Se RTT è grande il bit rate mi tenderà a zero e questo ovviamente non va bene. Per aumentarlo posso agire sulle dimensioni del pacchetto ma fino a un certo punto: non posso infatti "caricarlo" troppo. Però posso mandarne più di uno.

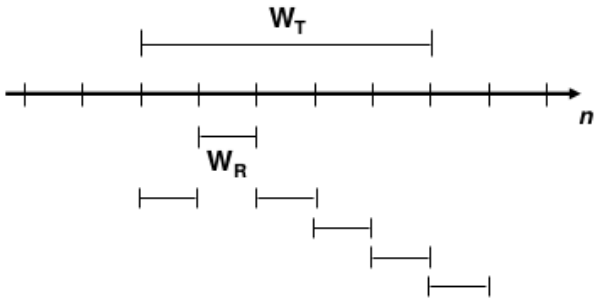
$$\text{Bit\_Rate} = [(\text{Dim. PDU}) / \text{RTT}] * W_T$$

Trasmettitore e ricevitore si devono accordare preventivamente sulla semantica degli ACK.
I primi due sono detti ACK positivi.
Il metodo selettivo parrebbe migliore ma è effettivamente così quando non si hanno errori.
Il metodo cumulativo è invece più robusto per le perdite di ACK: vedi esempio sottostante.

```
                    |X----------X----------|
TX    0    1    2    3    4    5    6
RX    0    1    2    3    0    1    2
                    ||
```

X → il rx non si muove per l'errore sul pacchetto 2. Quando gli arriva il pacchetto 3 lo scarterà perché fuori finestra e manderà l'ultimo ACK cumulativo in cui prima è andato tutto correttamente, pertanto invierà ACK(2). Al 4 non invia niente perché erroneo mentre per il 5 vale lo stesso discorso fatto per il pacchetto 3: manderà l'ACK(2) e lo stesso accadrà per il pacchetto 6 spedito dal trasmettitore.
Questo esempio mostra come sia possibile l'arrivo al trasmettitore di ACK duplicati utilizzando la tipologia di ACK cumulativo.

Esiste anche un ibrido, cioè una via di mezzo tra il tipo selettivo e quello cumulativo, che però non può essere usato nel Go Back N per via della dimensione degli ACK che devono avere e quindi non basta una finestra unitaria per la ricezione (tipica del Go Back N). La dimensione degli ACK è dovuta al dover ricordare tutta la storia dei vari bit di un pacchetto. Infatti l'ACK ibrido manda un ACK in cui c'è l'informazione di tutti i bit arrivati correttamente e quello erronei. Quindi uno svantaggio sono le dimensioni elevate degli ACK.
Piggybacking. Lo sfruttano quasi tutti i protocolli. Nel caso di flussi d'informazioni bidirezionali è sempre possibile scrivere l'informazione di riscontro (l'ACK) nell'intestazione di PDU d'informazione che viaggiano nella direzione opposta.
Posizioni relative corrette tra W_T e W_R.



La numerazione dei PDU è ciclica. C'è un vincolo che non deve essere violato. Non devono esserci due numeri uguali nei k pacchetti inviati, perché in questo modo si creerà ambiguità. Avendo due numeri uguali, infatti, non so a chi si riferisca.
Quindi nella somma complessiva delle due finestre (quella di trasmissione e quella di ricezione) non devono esserci numeri uguali.

SELECTIVE REPEAT: nel protocollo Go Back N, come abbiamo visto il ricevitore può accettare solo PDU in sequenza. Cioè se ho un errore in un bit di un pacchetto quelli che mi arrivano dopo, anche se sono corretti vengono scartati e il ricevitore rimanderà sempre l'ACK relativo all'ultimo bit corretto. Il selective repeat migliora questa condizione restrittiva accettando anche dopo un errore gli altri pacchetti senza errori, complicando non poco il ricevitore. Usa finestre di ricezione di dimensioni maggiori di 1, quasi sempre di dimensioni uguale.

RETI TELEMATICHE.

→ Lezione 04. Mezzi trasmissivi e strato fisico.

1. STRATO FISICO – RETI DI TRASPORTO E DI ACCESSO.

Sono i mezzi trasmissivi. Si distinguono in tre tipi principali:

1. elettrici: come il doppino telefonico e il cavo coassiale;
2. ottici: come la fibra ottica e i raggi laser;
3. radio: come i ponti radio, i satelliti e le reti cellulari.

ELETTRICI: la caratterizzazione dei mezzi elettrici dipendono dalla geometria, il numero di conduttori e dalla loro distanza, il tipo di isolante usato per fabbricarli e il tipo di schermatura dalle eventuali interferenze. Il mezzo elettrico ottimale è caratterizzato da resistenze, capacità parassite ed impedenze basse, da una buona resistenza alla trazione e dalla flessibilità.
I parametri di merito dei mezzi elettrici sono la sua impedenza (in funzione della frequenza), la velocità di propagazione del segnale (0.5c – 0.7c per cavi, 0.6c per le fibre ottiche), l'attenuazione (che cresce linearmente in dB con la distanza e con la radice quadrata della frequenza) e la diafonia (o cross-talk), cioè la misura del disturbo indotto da un cavo vicino (cresce con la distanza fino a stabilizzarsi).
- Doppino: a parte l'aria è il mezzo più economico. Generalmente composto da fili di rame ritorti tra loro per diminuire la diafonia. Ha il vantaggio di costare poco e di essere di facile installazione. Ci sono varie categorie di doppini. Maggiore è la categoria e più veloce è la trasmissione e la complessità del doppino (avrà più fili paralleli), ma la distanza a cui si può comunicare è minore.
- Cavo coassiale: è un sistema trasmissivo condiviso composto da un connettore centrale e da una o più calze di schermo. Ha una maggiore schermatura da interferenze esterne (con gabbie di Faraday) rispetto al doppino e ha una maggior velocità trasmissiva. Ha lo svantaggio però di essere più costoso e di difficile installazione. Esempi di cavo coassiale sono quello della TV oppure quello dell'oscilloscopio.

OTTICI: faremo qualche cenno solamente alla fibra ottica.
- Fibra ottica: fino ad oggi è il mezzo trasmissivo più complesso. Ha attenuazioni bassissime con vantaggio di poter amplificare con distanze elevate. È fatto di silice (quindi anche di vetro) ed è un materiale facile da trovare. Per la legge di Snell (una formula che descrive le modalità di rifrazione di un raggio luminoso nella transizione tra due mezzi con indice di rifrazione diverso) il raggio luminoso introdotto nella fibra entro un "angolo di accettazione" rimane confinato nel core. Come vantaggi possiamo dire che è immune da disturbi elettromagnetici, ha una altra capacità trasmissiva, una bassa attenuazione ed ha dimensioni molto piccole e costi contenuti. Gli svantaggi: sono adatte solo a collegamenti punto – punto, sono difficili da collegare tra loro con connettori, hanno un ridotto angolo di curvatura (non si possono piegare più di tanto per il discorso fatto prima sulla legge di Snell) ed infine sono fragili alle vibrazioni.

RADIO (ARIA): in questo caso la propagazione del segnale è influenzata da ostacoli naturali (o artificiali, come può essere un muro). Oggi la maggior distanza di propagazione di un segnale senza amplificazione è data dai segnali satellitari (circa 36000 km dal suolo terrestre). Questo è logico perché man mano che ci si allontana dalla superficie terrestre l'aria (cioè la fonte dell'attenuazione dei segnali radio) è più rarefatta, fina a scomparire praticamente del tutto quando si supera l'atmosfera terrestre. Quindi il segnale avrà bisogno di meno amplificazione o addirittura di neanche un'amplificazione per propagarsi.
Le interferenze principali sono due (importanti per la telefonia cellulare):

1. fanding: variazione veloce dell'ampiezza del segnale dovuta alla combinazione in fase di "copie" dello stesso segnale (in parole povere i segnali "rimbalzano" contro i muri in maniera controllata e quelle che hanno la stessa fase di sommeranno di ampiezza);
2. shadowing: variazione lente dell'ampiezza del segnale (i segnali non passano oltre i muri).

Il problema d'interferenza principale è dato dall'interferenza da altri segnali (detta interferenza co-canale). Il segnale si attenua proporzionalmente col quadrato della distanza.

Le reti di strato 1 sono, come intuibile dal titolo, di due tipi:

1. RETI DI TRASPORTO
2. RETI DI ACCESSO

Sono ammesse teoricamente accessi del tipo nB + mD con n ed m arbitrati, ma in pratica ci sono solo alcune combinazioni di m ed n. Vediamole.
Esistono due interfacce utente:

- BRI (Basic Rate Interface)
  2B + D (da 128 Kb/s)
  Offerto per utenti a "casa". Distribuisce il segnale numerico per mezzo del cosiddetto S-bus.
- PRI (Primary Rate Interface)
  30B + D (EU)
  23B + D (USA)
  Offerte per small business.

4. DSL: acronimo di Digital Subscriber Line. È una famiglia di tecnologie. La più diffusa e conosciuta è l'ADSL (Asymmetric Digital Subscriber Line). L'ADSL sfrutta le potenzialità del doppino che può andare ad una velocità molto superiore dei 64 Kb/s. L'ADSL ha una banda asimmetrica bidirezionale ed ha velocità maggiori quando scarico informazioni (downstream, in media 9 Mb/s) rispetto a quando le carico in rete (upstream, in media 640 Kb/s). Tale velocità dipende in modo significativo dalla distanza utente-centrale.
   Oltre all'accesso della rete, però mi deve garantire anche l'accesso alla rete telefonica. Per far ciò la rete è dotata di un dispositivo detto splitter, con banda dedicata che ha il compito di separare i segnali vocali dai dati. Opera cioè una divisione in frequenza, dividendo la banda in maniera asimmetrica.
   Un possibile scenario di utilizzo è descritto nei 3 seguenti tipi.
   - Un solo telefono. In questo caso non ho bisogno neanche dell'ADSL ed avrò solo la voce che si convoglierà in un dispositivo detto public switch.
   - Un telefono e l'ADSL: il più comune e quello maggiormente più utilizzato nelle case. In questo caso nella rete viaggiano sia dati numerici che voce e in questo caso avrò bisogno dello POTS splitter (un filtro) che mi separerà in frequenza voce e dati, convogliando la voce nel public switch e i dati in un dispositivo detto DSLAM.
   - Un solo dispositivo ADSL: il più usato dalle aziende. Qua ci sono solo di dati che vengono convogliati nel DSLAM.

   Distinzione degli apparati:

APPARATI UTENTE.
- Filtri splitter (separa dati dalla voce).
- Modem ((de)modula il segnale alle opportune frequenze).

APPARATI DI CENTRALE.
- Filtro/Modem POTS (funzione duale del filtro splitter, separa voce e dati).
- DSLAM (DSL Access Multiplexer: riceve flussi di dati e li convoglia su di un unico canale).

Fino ad ora per arrivare al primo nodo di rete (cioè al canale di accesso) abbiamo una banda dedicata per ogni utente. C'è pertanto un'allocazione statica di risorse. Ora vediamo un accesso non con banda dedicata ma con banda condivisa (divisa in frequenza – FDM).

5. HFC: acronimo di Hybrid Fiber Coax (dette anche reti CATV). Sono pensate per un funzionamento unidirezionale.
   Principi di funzionamento: usa lo stesso mezzo fisico della TV via cavo (fibra nella rete e cavo nell'"ultimo miglio"), la banda viene multiplata tra tutti gli utenti usando una tipologia ad albero. Visto che i segnali TV occupano porzioni diverse di banda occorre un filtro presso l'utente. Per decodificare i dati l'utente usa un cable modem (del tutto simile al modem).

6. FIBRE OTTICHE PASSIVE: stessa cosa dell'HFC solo che è fatto in ottica. In Italia non esiste, ma è usato pesantemente in Asia. Con la fibra che arriva a casa dell'utente avrò molto bit/rate.

RETI TELEMATICHE.

→ Lezione 05. Architetture stratificate. Strato collegamento (strato 2).

1. STRATO COLLEGAMENTO – ARCHITETTURE STRATIFICATE (LIVELLO 2).

Il livello (o strato) 2 è il primo che realizza un protocollo. Trasferisce bit (o simboli) al livello 1 come trasmettitore e riceve bit (o simboli) dal livello 1 da consegnare ai livelli superiori come ricevitore.

Funzioni di strato collegamento.

1. Delimitazione della trama (cioè dell'unità dati, PDU). Deve esserci sempre. Questa funzione capisce quando inizia e finisce una PDU. Dal livello 1 gli arrivano infatti un sacco di bit in sequenza che sono un'ininterrotta sequenza di PDU. Se non capisce la fine e l'inizio di una singola PDU non riuscirà a passarla ai livelli superiori. Ci possono essere tre sistemi per risolvere questo problema.
    - Delimitatori espliciti. Sequenze di bit riservate che mi dicono dove inizia e dove finisce una PDU. Questa determinata sequenza però non deve essere contenuta all'interno della PDU altrimenti la dividerò a metà. Si risolve questo problema aggiungendo dei bit detti bit stuffing (lo vedremo meglio più avanti).
    - Indicatori di lunghezza. Sequenza di bit che mi dice quanto è lunga una PDU.
    - Lunghezza fissa. In questo caso la sequenza è unica e non viene inviata per ogni PDU, ma rimanendo costante solo alla prima comunicazione.
    - Silenzi tra pacchetti. Cioè ricevo bit con continuità e quando non ricevo più (silenzio) questo significa che la PDU è finita.
2. Multiplazione. Può essere necessaria se ho più strati 3 che essendo di sistemi diversi non riescono a comunicare (stessi strati ma di sistemi diversi infatti "parlano lingue diverse"). Allora io devo dividere le varie PDU di un determinato strato 3 al giusto sistema il quale ha il livello 3 che riesce a capire la PDU. Oggi è poco usato anche perché Internet non ha questo problema perché lo strato 3 è unico.
3. Indirizzamento. Per canali condivisi. Per il punto – punto, infatti è inutile. Vi è un'identificazione di interlocutori su un preciso canale.
4. Rivelazione di errore. C'è quasi sempre. Almeno sempre c'è la rilevazione (senza correzione) dell'errore. Il controllo è focalizzato soprattutto sull'intestazione (o header) perché ci sono gli indirizzi.
5. Controllo di flusso. Come abbiamo già visto può esserci il caso che il ricevitore "chieda" al trasmettitore di rallentare il flusso di PDU per permettere lo smaltimento delle PDU stesse, senza creare congestione o ritardi troppo elevati e soprattutto perdita di pacchetti quando la memoria del ricevitore è piena. Ora sappiamo (dalla lezione 03) che questo ritardo può essere effettuato tramite una riduzione da parte del ricevitore della finestra di trasmissione W_T.
6. Controllo di sequenza.
7. Correzione errore.
8. Protocolli accesso multiplo. Solo per canali condivisi, per canali punto – punto non ci sono. Invece di usare i TDMA (tempo), FDMA (frequenza), CDMA (codici), si usano protocolli di accesso.
9. Controllo di flusso sull'interfaccia.

I protocolli che vedremo derivano da protocolli vecchi (SDLC ed HDLC). Ci occuperemo di protocolli di strato 2 con collegamenti punto – punto, cioè non condivisi.

Vediamo l'HDLC: nella famiglia di questi protocolli appartengono:

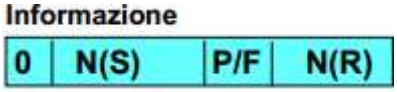
- LAPD (Link Access Procedure D-Channel): protocollo strato 2 per lo strato fisico ISDN canale D (per la segnalazione, visto nella lezione 04).
- LAPF (Link Access Procedure to Frame Mode Bearer Service).
- LLC 802.2 (Logical Link Control).
- PPP (Point-to-Point Protocol).
- LAPDm (LAP for the mobile D channel).

Descriveremo le caratteristiche principali dell'HDLC studiando in dettaglio:

- LAP-B, protocollo di strato collegamento per X.25 e ISDN canale B.
- LLC per le reti locali.

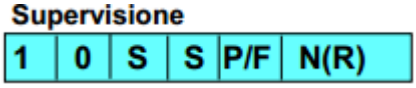
LAP-B.

Trame di informazione.



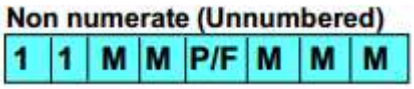
Dove N(S) è il numero di sequenza della PDU trasmessa ed N(R) è il numero di sequenza della PDU attesa. Permettono di trasferire i dati. I campi N(S) ed N(R) consentono un controllo di errore con protocollo a finestra.

Trame di supervisione.



Permettono di trasferire riscontri.
RR (Receiver Ready – C/R): riscontro positivo.
RNR (Receiver Not Ready – C/R): riscontro positivo e dichiarazione di non disponibilità del ricevitore.
Queste prime due trame sono ACK positivi e mi permettono il controllo di flusso esercitato nel modo on/off, cioè o parli a finestra massima o non parli. Questo capita quando il ricevitore non riesce più a gestire tutti i pacchetti che gli arrivano dal trasmettitore e quindi gli chiede di non trasmettere più. Dopo aver elaborato un numero sufficiente di pacchetti poi dovrà mandare un ACK di RR per far riprendere al trasmettitore la comunicazione.
REJ (Reject – C/R): richiesta di ritrasmissione di tutte le PDU a partire da N(R). Il corrispettivo del NACK. Questo è trasmesso quando si è sicuri che una determinata PDU(n) non mi è arrivata e quindi è andata persa.

Trame non numerate.



Principalmente praticano un controllo della trasmissione.

Queste trame di comando e risposta dell'LAPB determinano chi è slave e chi master. Se è una PDU di comando ho un master (indirizzo slave destinatario). Se è una PDU di risposta è uno slave (indirizzo slave sorgente).

Spieghiamo meglio ora il funzionamento alla base dei bit Poll/Final (P/F). Questo bit è presente nel campo controllo e forza la risposta del ricevitore. Quando un master setta come bit di Poll 1, lo slave risponde settando il bit di Final ad 1. Così facendo quel master è sicuro che la risposta è dello slave. Quindi setto i P=1 e mi aspetto una risposta con settato F=1. Sia master che slave non possono settare un altro P=1 prima di aver ricevuto un F=1. Questo metodo si può usare per ACK cumulativi.
Nell'LAPB gli indirizzi permettono di distinguere i comandi dalle risposte, e quindi di sapere se è stato trasmesso un bit di poll o di final.

Esempi di dialoghi nel protocollo LAP-B.

Nota: I pacchetti sono scritti nel seguente formato: l'indirizzo dello slave (Q,V), una lettera I identificativa, il numero del PDU e il numero dell'ACK.

Nota: negli esempi che seguono il codice Q,I00 e anche gli altri sono pacchetti che trasmette V e, in questo caso è lui ad essere il master, mentre lo slave è Q (infatti compare nell'indirizzo). Questo significa che V manda dei pacchetti a Q e, tramite l'operazione già descritta nelle lezioni precedenti detta pickybagging, manda oltre i dati che ha da mandare a Q, anche l'ACK (in questo caso cumulativo) dei pacchetti che riceve da Q.

Recupero errore con REJ. Trasmissione con errore sulla PDU.

Il trasmettitore Q invia il pacchetto PDU(0): V,I00.
I pacchetti che manda V (Q,I00 e altri) seguono lo stesso ragionamento della nota nel caso della trasmissione senza errori.
Il pacchetto V,I21 si perde e non arriva a destinazione. Quindi il ricevitore V una volta che si vede comparire il pacchetto V,I32 (il 2 alla fine è riferito all'ACK del pacchetto ricevuto da parte di V: Q,I21) manda un V,REJ2 cioè una richiesta di ritrasmissione delle PDU a partire dalla 2 perché gli manca proprio la PDU(2) che si era persa precedentemente. Quindi il trasmettitore rinvierà V,I23 (con l'ACK diverso, ma il pacchetto è sempre il 2) e V,I34 che vengono spediti al ricevitore senza ulteriori errori.

Recupero errore con bit P/F.

Q manda il pacchetto 2 V,I11 (PDU(1)) e si perde. Il ricevitore non avendola ricevuta ad ogni nuovo arrivo di un pacchetto rimanderà sempre l'ACK(1) riferito al conseguimento della V,I00P (PDU(0)) e dell'attesa della PDU(1). Il ricevitore continua a mandare pacchetti perché la PDU(1) potrebbe essere in ritardo per congestione della rete. Quando però il trasmettitore manda il pacchetto V,I33,P il ricevitore manda un ACK V,RR1,F. Settando il bit F ad 1 il trasmettitore capisce che il PDU(1) è andato sicuramente perso. Prima infatti Q non poteva saperlo. Notiamo anche che in questo caso il ricevitore V usa il protocollo Go Back N perché (rispondendo al rinvio del PDU(1) cioè il secondo pacchetto V,I15,P) con V,RR2,F (cioè con un ACK(2)) mi fa capire che i PDU 2,3,4 e 5 non li aveva salvati. Se avessi avuto il protocollo Selective Repeat, invece, avrebbe memorizzato gli altri pacchetti (il numero di tali pacchetti dipende dalla dimensione della sua finestra di ricezione). Mi avrebbe risposto con un ACK riferito all'ultimo pacchetto in sequenza che gli è arrivato correttamente. Quindi presupponendo che abbia memorizzato tutti i pacchetti che il trasmettitore gli ha inviato dopo la perdita di V,I11 (cioè i pacchetti V,I22 V,I33,P e V,I45) mi avrebbe risposto con un ACK(5), cioè mi aspetto il pacchetto 6 (PDU(5)).

PPP.
Il PPP (Point to Point Protocol) è utilizzato nei collegamenti su linea telefonica tra host di utenza residenziale e provider Internet, oltre che su connessioni SONET/SDH o su circuiti ISDN. Questo protocollo è forse il più semplice tra i Data Link Protocol.
Gli obiettivi del PPP sono la delimitazione delle PDU, la trasparenza del contenuto, il riconoscimento (senza correzione) di eventuali errori, la multiplazione di più protocolli di strato di rete, il controllo dell'attività sul collegamento, la negoziazione dell'indirizzo di livello rete (tipicamente IP): i nodi ai due estremi del collegamento apprendono o configurano i propri indirizzi di rete.

Nota: gli indirizzi IP sono univoci, ma non sono statici. O meglio alcuni lo sono, come per esempio quello degli uffici, ma quelli casalinghi, sono assegnati dinamicamente alla connessione dell'host specifico. Cioè ogni volta che mi collego la rete mi assegna un indirizzo IP.

Quindi il PPP non ha come obiettivi la correzione degli errori, il controllo di flusso, il mantenimento della sequenza e la gestione di collegamenti multipunto.

Il formato delle PDU del PPP è formato da:

- flag: delimitatore,
- address: mantiene la compatibilità con l'HDLC,
- control: che è come per l'address,
- protocol: per la multiplazione di protocollo di livello superiore.

Per ottenere la trasparenza voluta, i dati devono poter contenere il byte 01111110 (che non deve essere interpretato come flag delimitatore). Il trasmettitore quindi inserisce un byte detto byte di escape 01111101 (che svolge similmente il ruolo del bit stuffing già visto) prima di ogni sequenza 01111110. Il ricevitore presuppone che quello che c'è dopo di byte di escape sia "buono", quindi butta via la sequenza di escape e tiene la successiva sequenza. Se per caso io devo trasmettere una sequenza che è proprio uguale al byte di escape allora lo duplico. Così il ricevitore mi scarta il primo byte di escape e visto che presuppone che tutto quello che c'è dopo sia buono (anche se la sequenza dopo è uguale a quella di escape) non la butta via e la tiene per essere elaborata.

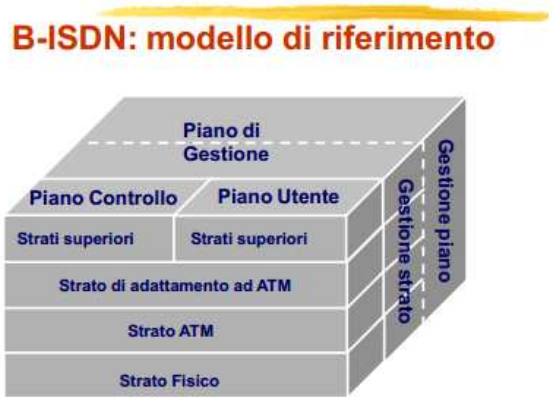
Frame Relay.
Il Frame Relay è un po' più di un semplice protocollo di livello 2. È nell'ambito dell'ISDN. Oggi è utilizzato molto per realizzare reti private in ambito aziendale, per interconnessioni LAN e dagli ISP per collegare router. La vera novità e che questo protocollo non è basato sul DataGram come i precedenti, ma crea un circuito virtuale. Questo implica che ci devono essere dei bit dedicati alle etichette (o header) del circuito virtuale. Come abbiamo già visto, queste etichette sono assegnate dinamicamente al momento della creazione del circuito virtuale, inoltre i PDU mi arrivano in sequenza. La massima lunghezza dei pacchetti è di 4.000 byte.
Facciamo un esempio. Supponiamo che io debba collegare delle città a grandi distanze: New York, Boston, Atlanta, Miami, Seattle. Devo costruire una rete logica in base alle mie esigenze. Creerò infatti un collegamento diretto tra le città che scambieranno più informazioni mentre farò passare le informazioni tramite nodi intermedi tra città che comunicano di meno. Questa scelta la faccio in base ad una determinata matrice di traffico che mi dice il traffico tra le varie città. Ad ogni canale avrò associata anche una capacità che sarà ovviamente più grande tra città più "comunicative" e meno grande in quelle meno. Se io dovessi far costruire concretamente la mia rete logica in fisica collegando tramite fili le mie città secondo la scelta fatta mi costerebbe molto per installarla, in più devo aggiungere costi di manutenzione e settare per ogni città i vari slot loro disponibili. È un'operazione tutt'altro che veloce e tutt'altro che economica. Quindi si sceglie un'altra strada, che è la seguente: si sfrutta già la rete che c'è nelle città. Per far capire meglio, tra Miami e New York voglio un collegamento diretto con una tot capacità di canale. Nel mio modello logico ho il mio bel collegamento diretto ma in pratica se sfrutto la rete che c'è già (cioè guardo a livello fisico) tra New York e Miami ci potrebbero essere centinaia di nodi intermedi, ma se il gestore del servizio mi riesce a mantenere la capacità e la velocità di trasmissione che voglio io non mi interessa a me come è fatto il sistema fisico, e posso tranquillamente guardare il mio modello ad un livello più alto (il mio sistema logico). Quindi questo è un sistema a banda garantita ed ha il vantaggio della flessibilità perché se mi si rompe un collegamento, tra Miami e New York dell'esempio; io posso trovare nella rete fisica un percorso alternativo (che mi deve sempre garantire la banda promessa) che invece in un collegamento

ATM.

ATM è l'acronimo di Asynchronous Transfer Mode. Questo è un primo esempio vero di rete integrata ai servizi. Il suo scopo era di arrivare a casa dell'utente: obiettivo sempre molto costoso. Commercialmente non ha avuto successo e ad oggi è utilizzato come supporto di altri protocolli di livello 2. Il suo obiettivo di arrivare direttamente a casa dell'utente è stato per cui abbandonato. Vediamo cmq le sue caratteristiche principali.

L'ATM è uno standard nato nell'ambito di una evoluzione di ISDN denominata B-ISDN (Broadcast ISDN, anni '90). Ideata come una rete a pacchetto con servizio circuito virtuale (per poterla controllare) su scala geografica. Avrebbe dovuto avere i seguenti obiettivi:

- velocità elevate (622Mb/s e superiori),
- bassa latenza per il trasporto di voce e video,
- uso di celle di dimensione fissa (53 byte di dati).

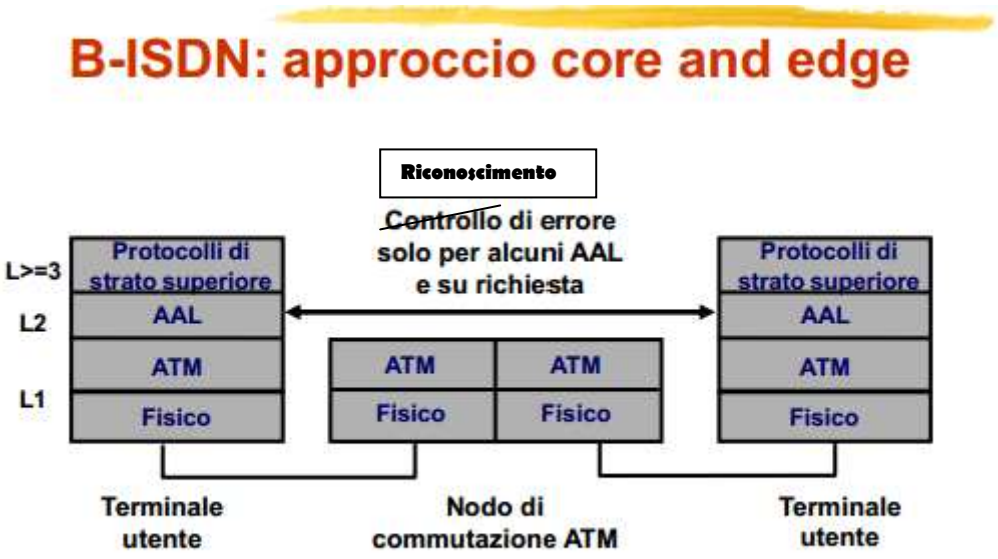


Modello di riferimento. Non è bidimensionale ma è tridimensionale (3D).

Si differenziano due piani:

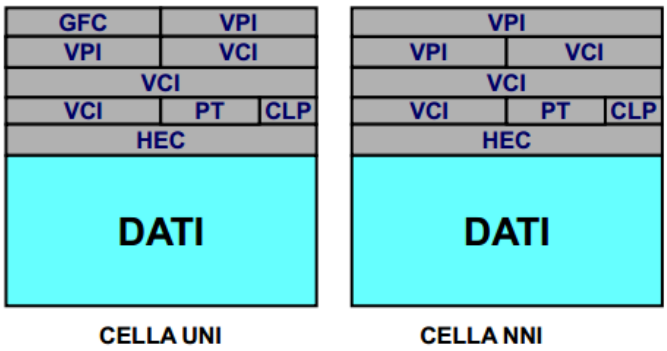
1. piano controllo: ci sono protocolli molto complessi. Questo stabilisce il percorso virtuale migliore e farà uno switching da passare al livello 2;
2. piano gestione: è il piano che mi crea la tridimensionalità. In realtà anche nei protocolli precedenti è presente, ma fa parte del piano controllo. Noi li abbiamo semplificati guardando solo la parte utente.

Anche in questo caso il controllo di errore è sempre ai bordi. Si ha sempre un approccio core and edge (che in questo caso però, a differenza del Frame Relay che controllava solo la presenza di un errore, c'è un riconoscimento di errore). Inoltre qua, il core non fa nessuna operazione CRC, mentre nel Frame Relay il CRC lo faceva sull'intestazione. Altra differenza è che il terminale utente in questo caso dell'ATM avrebbe dovuto essere rappresentato dal PC dell'utente, mentre per il Frame Relay era un indirizzo IP.



Formato cella ATM.

Le righe grigie sopra i DATI sono l'etichetta. Ogni riga rappresenta 1 byte dell'etichetta.



La cella UNI (cella fatta per l'utente) si differisce dalla cella NNI per avere a disposizione meno bit per l'etichetta del gruppo del canale virtuale. Nella cella NNI questo bit in più lo sfrutta per eseguire un controllo dei dati utente.

- Intestazione cella ATM (5 ottetti = 40bit)
  - GFC (4 bit): Generic Flow Control
  - VPI (8-12 bit): Virtual Path Identifier
  - VCI (16 bit): Virtual Circuit Identifier
  - PT (3 bit): Payload Type
  - CLP (1 bit): Cell Loss Priority
  - HEC (8 bit): Header Error Code

Vediamoli uno per uno.

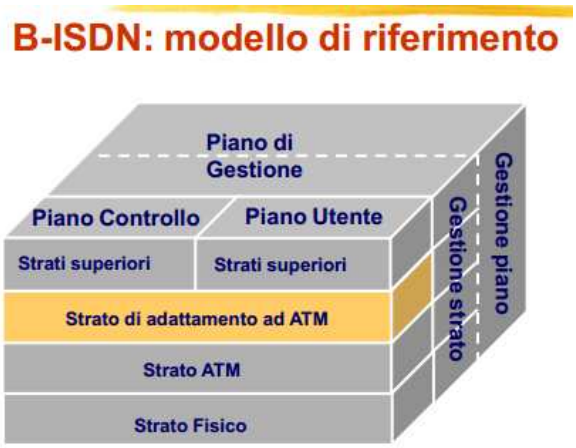
- GFC - Generic Flow Control
  - È presente solo all'interfaccia UNI.
  - Permette alla rete di trasmettere all'utente informazioni riguardanti la quantità di celle che può essere immessa in rete.
  - Due algoritmi di controllo:
    - ON-OFF
    - Crediti

- VPI - Virtual Path Identifier
  - Lunghezza variabile:
    - 8 bit alla UNI (256 VP's)
    - 12 bit alla NNI (4096 VP's)
  - Un certo numero di VPI sono destinati alla rete per scopi di gestione

- VCI: Virtual Circuit Identifier
  - Identifica il singolo circuito virtuale all'interno di un VP.
  - Sono disponibili 65536 VC's in ogni VP.
  - Un esempio: link a 2,4 Gb/s, 1 VP, VC di identica capacità ⇒ 36Kb/s per ogni VC.

- PT - Payload Type
  - Classifica il tipo di informazione presente nel payload.
  - Contiene l'identificativo denominato Payload Type Identifier (PTI).
  - Degli otto possibili codici PTI, quattro sono riservati alle funzioni di rete, gli altri alle funzioni d'utente.

Vediamo ora l'AAL, un protocollo sempre facente parte dell'ATM.
AAL (ATM Adaptation Layer).



Questo protocollo AAL è presente solo nel terminale utente e non in commutazione. Esegue la segmentazione alla trasmissione (48 byte o meno) e l'assemblaggio alla ricezione.
Integra il protocollo ATM per offrire servizi agli utenti.
Esempi di alcune funzioni AAL:

- gestione degli errori di trasmissione,
- gestione della pacchettizzazione,
- gestione della perdita di celle,
- controllo di flusso.

Definisce 4 classi di servizio per gli utenti:

1. Classe A: AAL tipo 1.
2. Classe B: AAL tipo 2.
3. Classe C e classe D: AAL tipo 3/4 ed AAL tipo 5.

Nota: l'AAL tipo 3/4 non si utilizza praticamente più. Viene utilizzato quasi sempre l'AAL di tipo 5. Due terminali che comunicano devono avere lo stesso tipo di protocollo AAL.

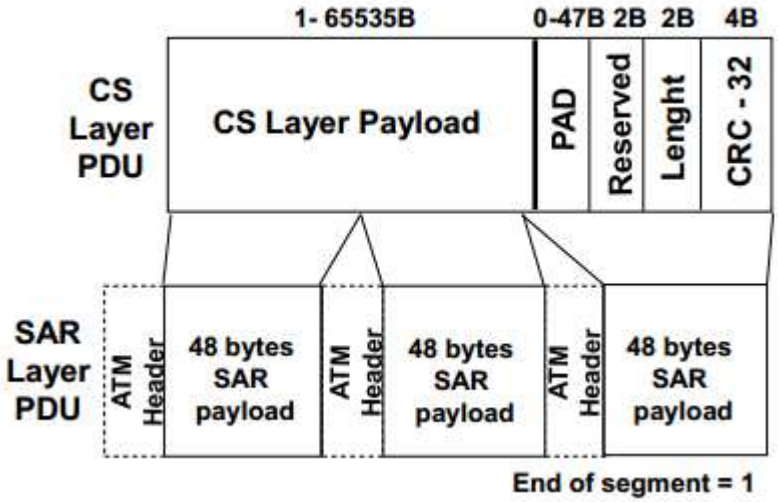
Queste classi sono definite attraverso tre parametri principali:

- la velocità di trasmissione della sorgente (cioè la tipologia della sorgente),
- la modalità di connessione,
- la relazione di temporizzazione tra punti terminali di connessione.

La classe A e B servono a supportare la voce e i video, mentre la classe C e D mi supportano i dati.
La classe A lavora su sorgenti CBR (cioè costanti), mentre le altri su sorgenti VBR (cioè variabili). Calcolare le distanze dei pacchetti mi permette di stabilire la velocità della sorgente.

Problemi per la telefonia. Nella commutazione a circuito ho solo il ritardo di propagazione e se avessi una sorgente CBR, dopo la prima comunicazione (cioè dopo il primo pacchetto che mi è arrivato) ci sommo il ritardo di propagazione citato prima e da quel momento in aventi so con esattezza i tempi di arrivo dei miei pacchetti. Tutti i pacchetti quindi mi arrivano allo stesso tempo di emissione traslati del ritardo di propagazione. Nella commutazione a pacchetto (come il circuito virtuale) ho altri ritardi oltre a quello di propagazione, come per esempio il tempo di coda. Se il primo pacchetto mi arriva per sfortuna con un ritardo minimo (cioè non deve aver incontrato "traffico" durante il suo viaggio in rete) mi trovo in una situazione difficile e delicata, perché non posso tenere quel tempo di riferimento. Mi sfasa il reale tempo che il pacchetto impiega per arrivare fino a me, per cui io mando i dati che mi sono arrivati ma poi non riesco ad inviare il resto perché non mi sono arrivati (perché questa volta il "traffico" è aumentato). L'istantaneità quindi ne risente e di molto. Quindi per evitare questo, quando mi arriva un campione, io aspetto prima di trasmetterlo cosicché prima della mia trasmissione ho un po' di campioni bufferizzati nella mia memoria pronti da essere inviati. Non è comunque accettabile un metodo del genere per l'interattività, infatti si usa questo metodo per lo streaming.
Questo è un problema intrinseco per ogni protocollo multimediale di voce e video. Oltre a quello già detto, in più, i pacchetti si possono perdere, e se prendo un gruppo di campioni che non sono in ordine sequenziale è un grosso problema. Quindi bisogna anche numerare i pacchetti: funzione che l'ATM non ha. La esegue il livello AAL. Inoltre mi serve anche il timestamp. Il timestamp(0) è l'istante logico a cui il pacchetto in questione è stato generato. I timestamp successivi (timestamp(n)) saranno tempi che inizieranno dopo un tot di tempo (che dipende dalla codifica usata nella comunicazione) dal timestamp(0) e ti indicano il tempo in cui devi "farli vedere" all'utente. Da notare che la numerazione dei pacchetti e il timestamp sono due cose differenti anche se sembrano la stessa cosa. Ma in realtà sono da tenere ben distinte, perché, come già detto, nella telefonia si campiona anche il silenzio (per far diventare la voce umana una

Vediamo ora la struttura dell'AAL 5.



Il principio base è che segmenta il PDU solo dopo aver messo prima gli appositi controlli.
Il CS Layer Payload mi arriva dal livello 3. Il campo PAD mi allinea le dimensioni del pacchetto al valore della dimensione di pudding (cioè di quanti byte ho a disposizione 0-47 Byte). È il "famoso" campo che mi aggiunge dei bit (detti bit di pudding) quando il pacchetto non è multiplo dei miei byte a disposizione per formare il pacchetto stesso. Se PAD = 0, questo significa che il pacchetto era multiplo di 48 byte. Il numero di questi bit aggiuntivi li scrive poi nel campo Lenght. Quindi l'AAL 5 alla fine aggiunge prima di segmentare il pacchetto un CRC e un PAD (i campi più importanti).

Un'ultima precisazione sull'AAL e più in generale sull'ATM riguarda la fine di un segmento (end of segment). L'AAL, una volta segmentati i pacchetti con 48 byte, "li passa" all'ATM e "gli chiede" di mettere un bit per contrassegnare la fine di quel determinato pacchetto. Usa un bit di ATM per portare una informazione sua (dell'AAL). Questa è una violazione dell'indipendenza tra strati.
L'unico svantaggio che ha AAL 5 e che se si perde questo bit di end of segment (o contiene un errore), perdo due pacchetti: perché la fine reale non viene rilevata e io continuerò a pensare che quello che mi arriva dopo faccia parte del pacchetto prima che invece è finito. Così mi perdo anche quello successivo. Se infatti ho un pacchetto da 99 bit, un bit di end, ed un altro pacchetto da 99 bit, se mi perdo il bit di end avrò un pacchetto di 198 bit formato dall'unione dei due, ma questo è irriconoscibile dal protocollo e quindi perderò entrambi i pacchetti.

Inoltre a questo c'è da aggiungere che nell'ATM non c'è la delimitazione della trama (cosa invece irrinunciabile negli altri protocolli di livello 2 visti finora). Questa informazione viene eseguita infatti dal livello 1, quello fisico, violando, un'altra volta, l'indipendenza degli strati.

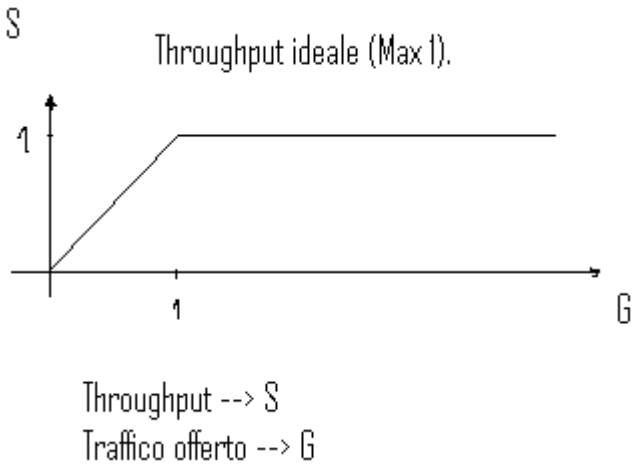
Confronto tra protocolli di strato 2

Protocollo	Delimitazione pacchetti	Multiplazione protocolli strato 3	Rilevazione errore	Correzione errore (protocollo a finestra)
LAPB	Delimitatore	Realizzato in strato superiore	SI	SI
LAPF core + LAPF control	Delimitatore	Mediante circuiti virtuali	SI, in LAPF core	Opzionale in LAP-F control
ATM (core)+ AAL (edge)	Demandato al livello fisico	Mediante circuiti virtuali	SI in AAL (edge)	NO
PPP	Delimitatore	SI	SI	NO
LLC	Demandato a MAC IEEE 802.3	SI	Opzionale	Opzionale
Ethernet MAC	Silenzi	SI	SI	NO

In grigio le informazioni non presentate nel corso o esaminate in seguito

Throughput.
Un parametro fondamentale di una rete è il suo throughput: la quantità misurata in bit/s dei bit che effettivamente il mio ricevitore riesce ad elaborare. Da non confondere con la capacità del canale. Quindi se ho conflitto, questo non mi porta un contributo al throughput, anzi me lo farà diminuire.

- Nei protocolli ad accesso casuale non si riesce ad avere il throughput ideale. Infatti, per definizione, l'accesso e libero e prima o poi avrò una collisione che mi farà diminuire il throughput.
- Nei protocolli ad accesso ordinato non si riesce, come prima, ad avere throughput ideale. Infatti, in questo caso, ho da tenere conto il tempo di propagazione del token, e quindi non potrò avere mai un throughput massimo.
- Nel protocollo a slot di prenotazione, invece il throughput massimo è garantito, ma a patto che la prima stazione a cui arriva il treno di slot parli sempre. E per quanto detto del DQDB, questo significa che parlerà sempre lui. Pertanto si garantisce il throughput massimo, ma non si garatisce l'equità della rete.
  In realtà esistevano dei modi per evitare di avere un monopolio della comunicazione da parte della stazione a monte. Uno dei quali era di permettere alle stazioni a valle di prenotare degli slot che non potevano essere riempiti dalle altre stazioni prima. La gestione però non era affatto semplice e complicava di molto il protocollo.

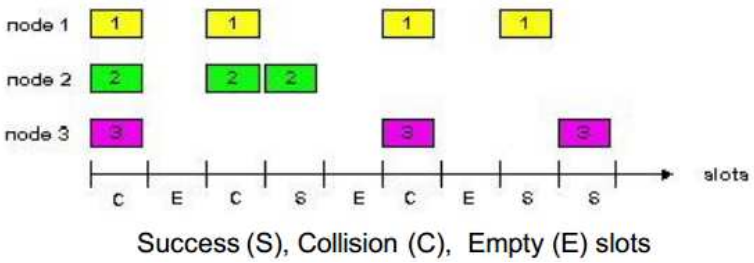


Noi parleremo solo di protocolli ad accesso casuale, oggi i più diffusi (come Ethernet).

Protocolli ad accesso casuale.
Ricordiamo che questo protocollo non fa niente se la collisione non è rilevata (di come fare per rilevarla ne parleremo tra breve), mentre si attiva se la collisione è rilevata.
I protocolli MAC ad accesso casuale specificano come riconoscere una collisione e come recuperare a fronte di una collisione (quasi sempre per ritrasmissione).
Ne vedremo tre:

1. Slotted Aloha;
2. ALOHA;
3. CSMA/CD (per aumentare il throughput).

SLOTTED ALOHA.



Il tempo è diviso in slot di uguale dimensione e i nodi trasmettono all'inizio di uno slot.
Questo protocollo prevede che tutte le stazioni siano sincrone (cosa non facile, in generale, da ottenere). C'è solo uno slot e tutte le stazioni connesse alla rete cercano di accedervi.

Posso pertanto avere le tre situazioni seguenti:

- lo slot è vuoto, per cui non ho nessuna trasmissione (non mi incide sul throughput);
- ho una sola trasmissione (mi incide sul throughput);
- ho più di una trasmissione che mi porta ad una collisione (non mi incide nel throughput).

Quando si ha collisione, la stazione a due possibili scelte:

- aspetta il trascorrere di quello slot e ritrasmetto su quello dopo. Si richiede una probabilità p della ricetrasmissione delle stazioni in modo da garantirne l'equità. La stazione che avrà più probabilità delle altre con cui ha avuto la collisione trasmetterà per prima.
- aspetta un tot di slot prima di provare a trasmettere.

In quest'ultimo caso posso avere tre differenti scelte:

1. CSMA persistente (1-persistente): riprovo immediatamente appena il canale è libero. Si collide di più;
2. CSMA non-persistente (0-persistente): riprovo dopo un tempo casuale ricontrollando nuovamente la rete (se è di nuovo occupata aspetto per un altro po' di tempo casuale e così via). Collido di meno rispetto al caso precedente.
3. CSMA p-persistente: con una probabilità p sono o 1-persistente oppure 0-persistente.

Da notare che questi sistemi non hanno nulla a che fare con la ritrasmissione.

Sembrerebbe che così facendo abbiamo evitato per sempre le collisioni. Ma non è così. Vediamo perché. Prendiamo una rete che ha come stazioni dalla A alla Z in ordine alfabetico in una tipologia a bus. La stazione C, poniamo, trasmette perché ascoltando la rete non ha trovato nessun altro segnale. Questa informazione mandata da C nella rete avrà un suo determinato tempo di propagazione. Per cui se mi si sveglia una stazione vicino, mettiamo la E, il segnale avrà fatto in tempo a raggiungere tale stazione e quando ascolterà la rete saprà che è occupata. Ma se mi si sveglia la Z (molto più distante rispetto alla E) è possibile che il segnale inviato da C non si sia ancora propagato fino a lì. Pertanto quando Z ascolterà la rete la riterrà libera, mentre invece il segnale gli deve ancora arrivare. Quando gli arriverà, visto che anche lei sta trasmettendo si creerà collisione. Le collisioni sono determinate dai ritardi di propagazione. Questo periodo di tempo nel quale il pacchetto può essere a rischio di collisione è chiamato periodo di vulnerabilità.

Prestazioni del CSMA.

Questo periodo di vulnerabilità a cosa è legato? O meglio, quando è che una trasmissione è sicura, cioè sono sicuro che non potrò più avere collisioni? La risposta non è difficile: sono sicuro di ciò quando il primo bit del mio pacchetto si è propagato su tutto il canale. Infatti in questa condizione, anche una stazione lontanissima dalla stazione che ha generato il segnale (se questo ha percorso tutto il canale) sentirà che la rete è occupata, perciò non trasmetterà a sua volta e non si avranno di conseguenza delle collisioni.
Il tempo di vulnerabilità, pertanto è quello della propagazione nel canale:

$$\frac{\text{Tempo propagazione (nel canale)}}{\text{Tempo trasmissione pacchetto}}$$

Il risultato di questo rapporto mi dà un parametro per stabilire l'efficienza di un protocollo. Più piccolo e meglio è. Se il tempo di occupazione del canale è comparabile con il tempo di propagazione del pacchetto, allora il protocollo non è efficiente. Il protocollo risulta più efficiente se la dimensione del pacchetto è grande e la dimensione della rete è piccola. Avrò infatti PDU grandi e bit rate piccolo.

Ci sono pertanto 3 variabili in gioco:

1. dimensioni delle reti;
2. velocità di trasmissione;
3. dimensione dei pacchetti.

La prestazione è ottimale se ho reti piccole (rispetto alla dimensione del pacchetto) con velocità di trasmissione basse e pacchetti grandi.
Si ha un vincolo riguardo alla capacità di riconoscere le collisioni ed è incentrato sulla dimensione minima che i pacchetti devono avere. Questo è legato al funzionamento del protocollo e se si genera un pacchetto più piccolo della dimensione minima ammissibile per riconoscere le collisioni allora il protocollo non funziona. Questa minima dimensione dei pacchetti è pari al doppio del tempo di propagazione (nella rete Ethernet). Cioè, in altre parole, il pacchetto deve essere come minimo il doppio della distanza tra due stazioni della rete per far si che le collisioni si possano rilevare.

Si preferisce il tipo 1-persistente perché è migliore a basso carico (anche se abbiamo visto che è più probabile incappare in collisioni), infatti il ritardo di accesso è inferiore rispetto agli altri.

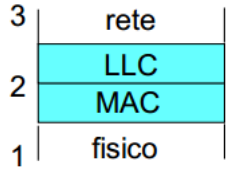
Gli svantaggi del CSMA sono principalmente due:

- la difficoltà nel separare il traffico di diversa priorità, non sono previsti infatti meccanismi per far si che si preferisca privilegiare un determinato tipo do traffico rispetto ad un altro;
- la sua instabilità per il controllo del parametro G. Se questo cresce ci sarà un aumento, di conseguenza, delle collisioni. Ma la rete come capisce che G sta aumentando? Per fare ciò conta il nu-

RETI TELEMATICHE.

→ Lezione 07. Standard LAN.

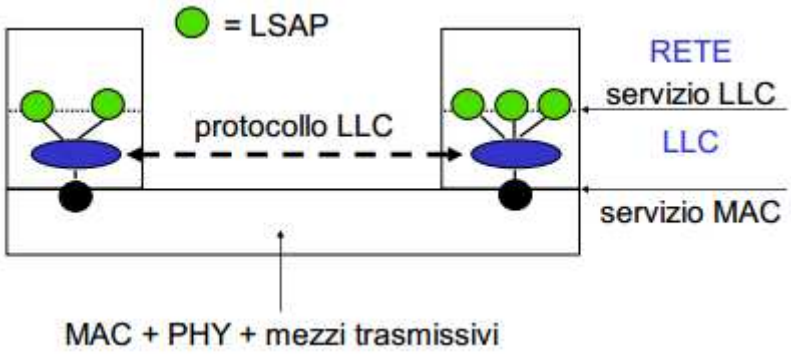
Standard per reti locali.



Il livello 2 è diviso in due sottolivelli:
- LLC: Logical Link Control,
- MAC: Medium Access Control.

Funzioni strato 2 nelle reti locali.
- Delimitazione delle trame delegato al MAC eseguito con silenzi tra i pacchetti (SFD).
- Multiplazione: IEEE 802.2 LLC, MAC Ethernet.
- Rilevazione di errore eseguita dal sottostrato MAC.
- Correzione degli errori con protocolli a finestra eseguita dal sottostrato LLC (opzionale).
- Indirizzamento eseguito dal sottostrato MAC per identificare le schede e dal sottostrato LLC per la multiplazione.

Indirizzi LLC.
Permettono la multiplazione di più protocolli di strato superiore (livello 3).



Indirizzi MAC.
Sono grandi (di solito 6 byte). Solitamente scritti in una ROM della scheda dal costruttore (ora anche configurabili, ma nella stessa rete locale non se ne possono avere due uguali).
Sono composti da due parti.
- 3 bytes più significativi: lotto di indirizzi assegnati al costruttore (detti Organitation Unique Id.);
- 3 bytes meno significativi: numerazione progressiva interna decisa dal costruttore.

Gli indirizzi MAC possono essere:
- single od unicast: se riferiti ad una sola stazione, cioè ad una sola interfaccia della rete;
- multicast: se riferiti a gruppi di stazioni;
- broadcast (FF FF FF FF FF FF – tutti 1): se riferiti a tutte le stazioni.

In multicast ho due modalità possibili:
1. solicitation: richiesta di servizio ad un gruppo multicast (tutti lo ricevono ma solo chi è abilitato elabora l'informazione);
2. advertisement: periodica diffusione di informazioni di appartenenza ad un gruppo multicast.

Una scheda MAC quando riceve un pacchetto corretto (se non è corretto, come ogni altro protocollo lo scarta) lo elabora guardando gli indirizzi. Se è uguale al suo allora lo accetta altrimenti lo scarta. Quindi:
- se l'indirizzo MAC destinazione coincide con quello di stazione lo accetta;
- se l'indirizzo MAC destinazione è multicast lo accetta se il gruppo multicast è stato abilitato;
- se indirizzo MAC destinazione è broadcast lo accetta.

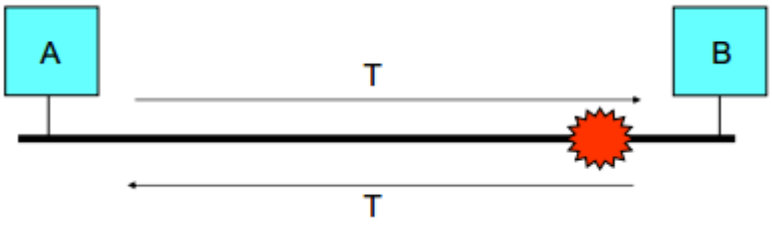
Le schede possono essere anche configurate in modo promiscuo (promiscous mode – si veda IP – Lezione 05 – WINDUMP/TCPDUMP).

Per sapere quali sono (e quanti sono) i bit di padding mi servo del campo lunghezza, che mi dice quanto è effettivamente lungo un pacchetto.

Si noti che questi due formati, l'Ethernet e l'IEEE 802.3 coesistono sulla rete.

Round Trip Delay (RTD) e Collision Domain.
L' RTD è il tempo necessario, nel caso peggiore, al segnale inviato da una stazione per arrivare all'altro estremo del cavo e a tornare indietro.

RTD = 2 T



È un indice di misura per scoprire che lunghezza deve avere il mio pacchetto per far si che il mio protocollo possa rilevare le collisioni (vedi RT – Lezione 06).

Il tempo di trasmissione di una trama non può essere inferiore al RTD, per cui la velocità del mezzo trasmissivo e le dimensioni della rete determinano quindi la lunghezza minima della mia trama.

Definiamo ora il concetto di Collision Domain: è una porzione di rete Ethernet all'interno della quale se due stazioni trasmettono simultaneamente, le due trame collidono. Devo stabilire pertanto le 3 variabili viste nella lezione precedente: le dimensioni delle reti, la velocità di trasmissione e la dimensione dei pacchetti.
Con il termine diametro di un collision domain si indica la distanza massima tra ogni possibile coppia di stazioni. Il diametro massimo di un collision domain a 10Mbit/s è di 2800m (quasi 3 Km) e dipende da due vincoli:

- la lunghezza massima dei cavi (attenuazione del segnale che induce uso di repeater);
- ritardo di propagazione (RTD) del pacchetto più piccolo ammissibile.

Se la voglio fare di 6 Km devo collegarle con uno switch (a questo si rimanda alla Lezione 08 - Interconnessione di reti locali).

Ethernet: livello fisico.

- Velocità trasmissione: 10 Mb/s (bit time = 0.1μs)
- Codifica Manchester (20Mbit/s di clock per facilitare mantenimento sincronismo in rete asincrona)
- Stazioni: max 1024
- Mezzi trasmissivi:
  - 10 BASE 5: cavo coassiale spesso RG213
  - 10 BASE 2: cavo coassiale sottile RG58
  - 10 BASE T: doppino telefonico UTP da 100 Ohm
  - 10 BASE FL, 10 BASE FB, 10 BASE FP: fibra ottica multimodale

Perché 20Mbit/s nella codifica Manchester?
La codifica Manchester si utilizza per mantenere il sincronismo nella rete Ethernet che, come detto, è asincrona. Infatti dopo che il ricevitore setta i suoi parametri di sincronizzazione con quelli contenuti nel preambolo della trama Ethernet man mano che il tempo passa, la sincronizzazione non è più garantita. Il clock mi si disallinea con il bit rate. Per cui accetto un bit rate doppio (20 Mbit/s) per stare dietro a quello del trasmettitore (di 10Mbit/s).

na successiva dopo un certo lasso di tempo e così via. Il polling quindi è un'interrogazione continua a delle macchine, alle quali gli si da il permesso di comunicare. Nel 100VG questo permesso lo esegue lo switch (che fa la parte del token). Lo switch quindi interroga periodicamente uno per volta gli host che sono connessi a lui. Con questo sistema posso caratterizzare il mio traffico in rete (cioè se video, voce, immagini od altro). Questa informazione viene chiesta dallo switch al momento dell'interrogazione all'host. In questo modo posso permettere anche una via privilegiata ad un determinato tipo di traffico. Infatti posso programmare lo switch in modo che permetta prima la trasmissione di traffico video, ad esempio, e poi gli altri, fornendo una priorità a quel determinato formato di dati.
Tutto questo discorso però lo posso fare solo se il dispositivo è uno switch. Se ho un hub ovviamente non lo posso fare, perché come si vedrà nella lezione 08, la sua funzione è praticamente quella di un amplificatore: mi prolunga la dimensione della rete senza attenuazione.
Questo tipo di soluzione però non ha sfondato sul mercato, perché per renderla effettiva si sarebbe dovuto cambiare tutte le schede degli apparati, e questo richiede tempo e denaro. Come si è sottolineato più volte, la compatibilità con l'installato è di fondamentale importanza per far si che una tecnologia si diffonda.

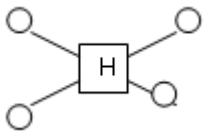
A questo punto si può fare un salto in più e provare a sviluppare un protocollo Ethernet da 1Gbit/s. In questo caso, però, la distanza della rete diventerebbe troppo piccola: se scaliamo di un altro fattore 10 rispetto a quello da 100Mbit/s abbiamo una lunghezza di 30 metri andata e ritorno, quindi il primo dispositivo di accesso alla rete deve essere a 15 metri. Questa tecnologia Ethernet, per questo motivo, si esegue su reti completamente switchate (modalità full-duplex, come si vedrà nella lezione 08). In una rete completamente switchata, le collisioni non ci potranno più essere, perché tutti gli host avranno un collegamento diretto con lo switch (collegamento punto-punto). In questo caso non devo più tener conto della distanza (perché era il vincolo che non mi avrebbe permesso di rilevare le collisioni che, con il full-duplex, non ho più). Ethernet diventa come un qualsiasi protocollo di livello 2. Qua la dimensione minima dei pacchetti è di 640 bytes.

Nota: oggi le schede degli host e dei dispositivi hanno implementato il funzionamento sia da 10Mbit/s, sia da 100Mbit/s e, in alcuni casi più recenti, anche da 1Gbit/s. Questo perché se io un giorno volessi mettere nella rete un dispositivo che va a 100Mbit/s, invece che 10Mbi/s, l'host può sfruttare questo aumento di capacità avendo la scheda che supporta i 100Mbit/s e passare automaticamente da 10 a 100Mbit/s senza difficoltà. Se invece altri host non ce l'hanno, questo non è un problema, perché, lo switch da 100Mbit/s introdotto potrà anche andare a 10Mbit/s. Man mano che il tempo passa anche i pc si evolvono, per cui quando sostituirò il mio pc da 10Mbit/s con uno da 100Mbit/s portò essere in grado di sfruttare al massimo lo switch introdotto prima.

Fino ad ora abbiamo parlato del protocollo dal punto di vista funzionale. Vediamo cosa possiamo dire dal punto di vista prestazionale. Prima di tutto chiariamo bene un punto. Le prestazioni di questi protocolli Ethernet hanno un senso se c'è contesa, cioè se si possono creare delle collisioni. Nell'Ethernet ad 1 Gbit/s, quindi non ha senso parlarne. Le prestazioni dipendono da quanto è grande il pacchetto. Più è grande, meglio è.
Nell'Ethernet ad 1 Gbit/s si utilizza:

- un Jumbo frame (pacchetti di 8000/9000 byte) per migliorarne le prestazioni,
- codifica 8B10B per facilitare il recupero di sincronismo con ridotto overhead.
  Questa codifica mi permette di mappare 2^8=256 bit su 2^{10}=1024 bit. Avrò pertanto delle sequenze di bit aggiuntive a quelle che mi servono: 1024-256=768 bit in aggiunta. Questi over di sequenze mi permettono di scegliere quali sequenze mettere nei pacchetti e gli altri non li utilizzerò. Questi sono sequenze di bit che nel traffico dati non si usano praticamente mai, per cui vengono usati per funzioni di altro tipo (come per creare una sequenza di jamming, o dei flag, per il sincronismo ecc. ecc.). Se devo usare una sequenza di sincronismo sceglierò, ad esempio una sequenza di 1 e 0 alternati, e non una tutti 1 o 0, perché così ho ben definiti i fronti di salita.

Hub e repeater, quindi sono lo stesso oggetto. Dal punto di vista trasmissivo, infatti, hanno lo stesso funzionamento.
L'unica differenza è che il repeater ha una porta per ogni spezzone di rete che unisce, mentre l'hub ha una porta per ogni pc. Si capisce quindi che l'hub ha molte più porte del repeater.



Quindi, l'hub ha un'interfaccia a doppino che è cablato come la tecnologia elettrica e fa collassare la rete con la topologia a bus in un punto trasformandola in una stella passiva. È come se avessi un cavo solo. Si noti che l'hub è alimentato è quindi è un dispositivo attivo che simula un comportamento passivo. Se si rompe il filo dell'alimentazione all'hub, ovviamente le comunicazioni si interrompono.

Quindi avendo un canale solo la rete è equivalente a quella che avevo prima ed ogni utente può aspettarsi al meglio C/N, dove C è la capacità del canale ed N il numero degli utente. Questo risultato lo avrei se ci mettiamo nel caso di avere un protocollo ideale, cioè senza collisioni e se tutti gli N utenti vogliono comunicare. La capacità di tutti questi canali collegati da hub e da repeater è sempre C.

Bridge e switch.
Questa è una vera e propria interconnessione, infatti mi cambia radicalmente la rete locale.

La differenza tra questo disegno e quello del repeater è che qua ho 3 collision domain (a differenza del repeater che ne aveva uno solo), infatti le reti sono indipendenti.
Il bridge non è più un amplificatore, infatti questo dispositivo "guarda" i pacchetti, senza però alterarne il contenuto. Si limita a controllare l'indirizzo di livello 2 e decide dove spedirlo. Fatto importante è che è un dispositivo con modalità store and forward.
Da notare che la presenza del bridge non mi modifica il comportamento dell'host, infatti questo non sa a che dispositivo è collegato: il bridge è trasparente nella rete. Infatti se al posto del repeater metto un bridge il funzionamento della rete non cambia.

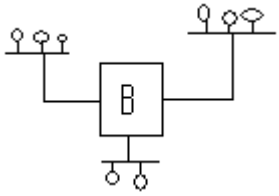
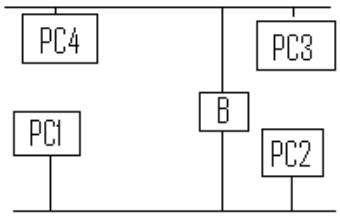
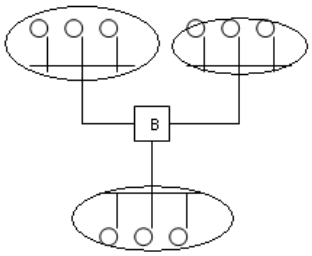


Fig.: il bridge deve capire che se il pacchetto spedito da PC3 deve arrivare a PC4, non deve instradarlo, perché non ce n'è bisogno, in quanto i due PC sono connessi alla stessa LAN. Se invece devo consegnare un pacchetto dal PC4 al PC2, allora il bridge deve intervenire. Prima di tutto legge l'indirizzo del pacchetto (a livello 2) e lo inoltrerà alla porta della LAN sulla quale è collegato PC2.



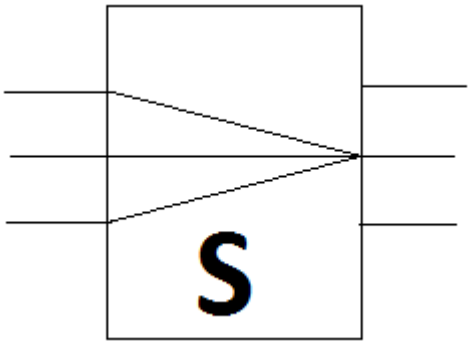
Nel disegno sopra se ogni connessione ha come capacità 10Mbit/s lo sfrutterò per 40Mbit/s.
Quando riseco a sfruttare questo guadagno?

1. Tutti gli host comunicano stando sempre dentro la propria rete. Ho quindi solo un traffico locale. Il K sono le porte del bridge/switch.



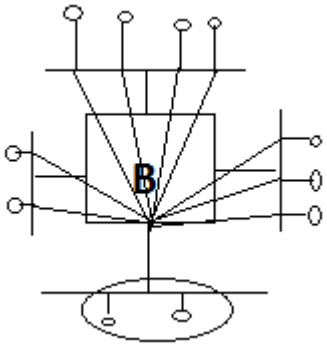
$C/(N/K) = K\,(C/N)$
ogni utente guadagna un fattore K.

5. Caso peggiore: si presenta quando tutti gli host vogliono comunicare con la stessa LAN. Non si riesce ad avere il guadagno del fattore K, e quindi non si riesce a sfruttare il vantaggio che una partizione di rete switchata potenzialmente ti può dare. Come si vede dal disegno a dx si creerà un problema di congestione, lo stesso che affliggeva la commutazione di pacchetto. Si tira fuori C per tutti gli utenti. Se la rete rispetta il principio di equità avrò C/N, cioè lo stesso del caso dell'hub, che, come visto, non mi fa guadagnare nulla. Infatti è come avessi un filo solo.

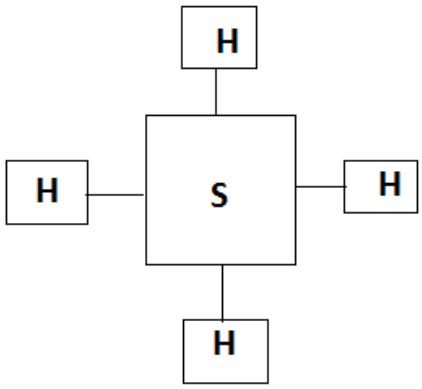


Quindi, nelle reti switchate, a seconda di come è la distribuzione del traffico, posso avere o meno guadagno di un fattore K (dove K sono le porte dello switch). Come abbiamo visto nei casi precedenti ho guadagno nel caso di traffico locale, uniforme e nel caso particolare che ogni host trasmette ad un host differente. Con il caso peggiore, cioè tutti che vogliono accedere ad una stessa LAN allora mi riconduco al caso dell'hub (C/N). In questo caso potrei anche perdere i pacchetti (per le collisioni), cosa che negli altri casi non avevo (perché non potevo avere collisioni). Questo è uno dei grossi limiti di questo modo di partizionare la rete.

Vediamo un altro caso particolare:

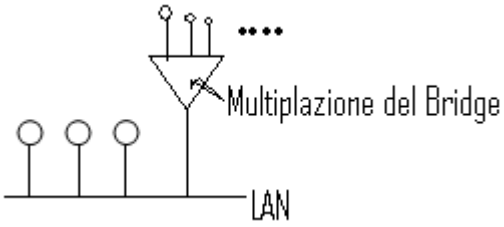


Equivalente alla rete disegnata a lato --->



Tutti vogliono trasmettere su una stessa LAN. Tutto questo traffico
per chi riceve, è al massimo C. Non si sfruttano pertanto i vantaggi potenziali legati alla distribuzione del traffico.

Esaminiamo solo la parte della LAN in cui arriva tutto il traffico (cioè la parte cerchiata nel disegno sopra a sinistra):



in questa LAN avrò [(N/K)+1] interfacce (il +1 è dovuto alla presenza del bridge). Quindi similmente agli altri casi abbiamo una capacità del canale divisa tra {C/[(N/K)+1]}, se la rete opera in modo equo.
Tutti gli utenti che non fanno parte di quella LAN (che, quindi, dovranno accedere a quella LAN tramite il bridge) sono (k-1)(N/K) e riceveranno tutti sempre la capacità pari ad un utente che è connesso direttamente a quella LAN cioè {C/[(N/K)+1]}. Questo perché per arrivare a quella LAN si deve per forza passare da quella determinata interfaccia del bridge, che ha quella capacità. Questo creerà delle iniquità sull'accesso nella rete, perché si capisce immediatamente che chi è connesso direttamente a quella LAN è favorito rispetto ad un utente che deve accedervi tramite bridge. Infatti ogni utente connesso direttamente alla LAN avrà per sé {C/[(N/K)+1]}, mentre nel secondo caso è l'accesso del bridge ad avere {C/[(N/K)+1]} e non solo un utente, per cui l'host collegato alla LAN con il bridge dovrà dividere {C/[(N/K)+1]} diviso tutti gli utenti che sono connessi al bridge e che vogliono connettersi a quella LAN.

2. Frame forwarding: ritrasmissione di trame ricevute con filtraggio degli indirizzi.
3. Algoritmo spanning tree: in questa funzione i bridge si coordinano per passare ad una tipologia ad anello in una ad albero (più avanti sarà spiegato il perché).

Vediamole una per una più approfonditamente.

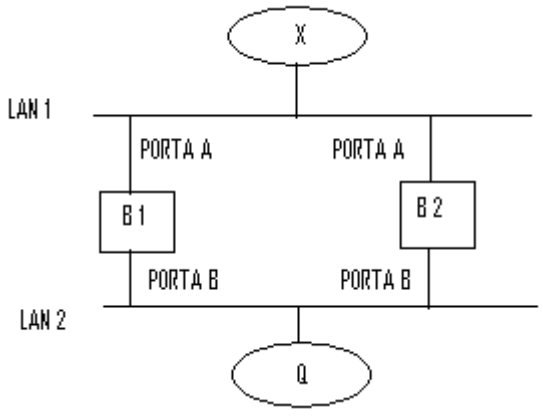
ADDRESS LEARNING. Come detto le tabelle si popolano automaticamente, quindi gli indirizzi non sono memorizzati in maniera statica, ma vengono inseriti nella tabella in maniera dinamica. Il problema ce l'ho all'inizio: la mia tabella infatti sarà vuota. Vediamo come riempirla: il primo pacchetto che arriva avrà, oltre all'indirizzo destinazione, l'indirizzo sorgente, cioè l'indirizzo della stazione che l'ha generato. Il bridge lo instrada guardando l'indirizzo destinazione, ma si memorizza da quale porta arriva. A parole la riga che il bridge scriverà nella tabella di instradamento sarà: "la stazione X la posso raggiungere tramite la porta Y". Per esempio se mi arriva un PDU dalla porta A con l'indirizzo sorgente S e quello destinatario D, il bridge memorizzerà nella sua tabellina: "S lo posso raggiungere tramite A" (così facendo mi creo una riga della tabella e da ora in avanti tutti i pacchetti diretti ad S saranno inoltrati nella porta A); dopo lo inoltrerà guardando l'indirizzo destinatario D del PDU.
Quindi quando mi arriva un PDU il bridge guarda l'indirizzo sorgente. Così facendo mi creo una riga della Tabella (ricavata proprio dall'informazione dell'indirizzo sorgente contenuta nel pacchetto). Queste informazioni, di solito, sono caratterizzate da un timer. Quando scade si perde l'informazione e il procedimento riparte.
Questo è denominato algoritmo di backward learning (cioè un apprendimento al contrario).

FRAME FORWARDING. Come già più volte detto il bridge e lo switch devono essere trasparenti nella rete, per cui si deve garantire la corretta trasmissione del pacchetto dalla sorgente al destinatario.
Quando mi arriva un pacchetto da una porta X il bridge (o lo switch) guarda la destinazione D. Ho 3 possibilità:

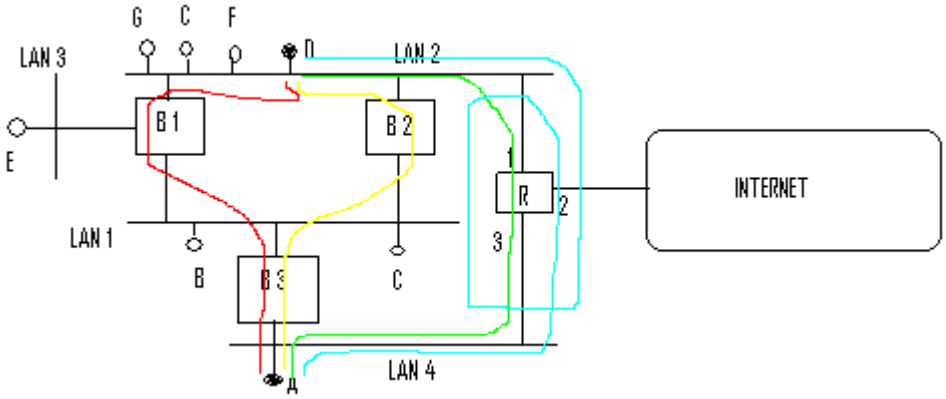
- se D è associata alla stessa porta X da dove è arrivato non faccio nulla (significa che i due host hanno già comunicato);
- se D è associata ad un'altra porta Y faccio effettivamente il forwarding;
- può capitare che D non sia associata a nessuna delle porte della tabella. Che faccio in questo caso? L'unico modo che ho per garantire che il destinatario riceva sicuramente il pacchetto è quello in cui il bridge lo spedisca a tutte le porte (tranne ovviamente in quella da cui è arrivato). Una specie di broadcast o multicast.

Nota: avere un bridge od uno switch (od altri dispositivi) quando sono in modalità broadcast non cambia nulla dal punto di vista funzionale, i metodi sono sempre quelli visti fino ad ora, ma mi cambia molto dal punto di vista prestazionale. Infatti se è un broadcast del pacchetto che ho, ne faccio K copie (dove K è il numero delle porte del bridge (o switch che sia)). Facendo questo perderò il guadagno di capacità del canale che mi garantisce lo switch (come visto precedentemente).

ALGORITMO SPANNING TREE. Gli algoritmi funzionano se non ho anelli, perché se gli avessi mi creerebbero delle copie di pacchetti ed altre problematiche ben più gravi che riguardano l'indirizzamento. Vediamole in dettaglio con un esempio.



ESEMPIO DI TOPOLOGIA DI RETE. **A → D. Partire sempre dagli indirizzi.**



Il primo dispositivo che fa qualcosa è naturalmente l'host A. Infatti deve decidere quali indirizzi mettere nel PDU. Sicuramente metterà 2 indirizzi MAC (del sorgente e del destinatario) e 2 indirizzi IP (del sorgente e del destinatario).
Quali indirizzi metterà A nel PDU?

MAC sorgente	MAC destinatario	IP sorgente	IP destinatario
A		A	D

Tre dei quattro indirizzi devono essere sicuramente quelli sopra elencati. L'indirizzo MAC sorgente infatti è A stesso. Gli indirizzi IP sono sempre gli stessi, non cambiano mai durante il percorso e sono indipendenti da ogni tipologia di rete.

L'unico dubbio lo può creare il campo MAC destinatario. Questo dipende dai due indirizzi IP degli host che vogliono comunicare tra loro. Se hanno netID uguale allora metterò D (consegna diretta), se invece hanno netID diversa devo passare per il MAC del router e quindi metterò R (consegna indiretta).

Oss.: A e D possono avere stesso netID? La condizione per avere uguale netID è che possono comunicare tra loro a livello 2. In questo caso disegnato sopra possono, pertanto possono avere stesso netID.

Ora prendiamo in esame il router R. Questo ha 3 interfacce e pertanto avrà 3 indirizzi MAC. Il MAC a cui non arriverà mai nessun pacchetto da A è sicuramente l'interfaccia 2. Si ricorda che per scoprire l'indirizzo MAC del router si deve mandare un pacchetto ARP (che è broadcast). Pertanto questo pacchetto ARP arriverà sull'interfaccia 1 e 3 di R. Nel pacchetto ARP ci sarà l'indirizzo IP del router, che di solito coincide con il mio indirizzo netID. Se dal pacchetto ARP ottengo prima l'indirizzo MAC dell'interfaccia 1 farò il percorso azzurro, mentre se ottengo quello 3 faccio il percorso verde.

Se invece mettiamo come MAC destinatario D, il router lo ignorerà, perché non ha la corrispondenza tra i suoi indirizzi MAC. Il pacchetto, invece, viene accettato dal bridge B 3 e tramite l'algoritmo di spanning tree mi disabilita un'interfaccia per poter effettuare tra bridge la tipologia ad albero con assenza di anelli.

Nota: un host per funzionare e generare pacchetti deve avere per forza avere:

- *indirizzo MAC;*
- *indirizzo IP;*
- *indirizzo IP del Router;*
- *Netmask (uguale a quello del default gateway, perché sono sulla stessa subnet);*
- *indirizzo IP DNS.*

Queste informazioni sono date per configurazione in maniera automatica dal DHCP, tranne l'indirizzo MAC della scheda dei vari dispositivi che sono settati dalle case produttrici.

RETI TELEMATICHE.

→ Lezione 09. Protocolli strato 3. Instradamento.

L'instradamento è la funzione più importante dello strato 3. È effettuato consultando le tabelle di instradamento:

- per ogni pacchetto, in rete datagram;
- per ogni connessione, in rete a circuito virtuale.

In queste tabelle si vede la "freccia" che indica la direzione che voglio prendere, non si vede tutto il percorso. Infatti contengono informazioni del tipo: per ogni destinazione next-hop (prossimo router).

L'indirizzamento si svolge tramite indirizzi univoci e tramite il mapping (cioè una risoluzione di indirizzi, per esempio, tramite ARP). Si cerca di effettuare un controllo di congestione per non perdere pacchetti. Per esempio in ATM e Frame Relay c'è un invio di informazioni per segnalare la congestione.

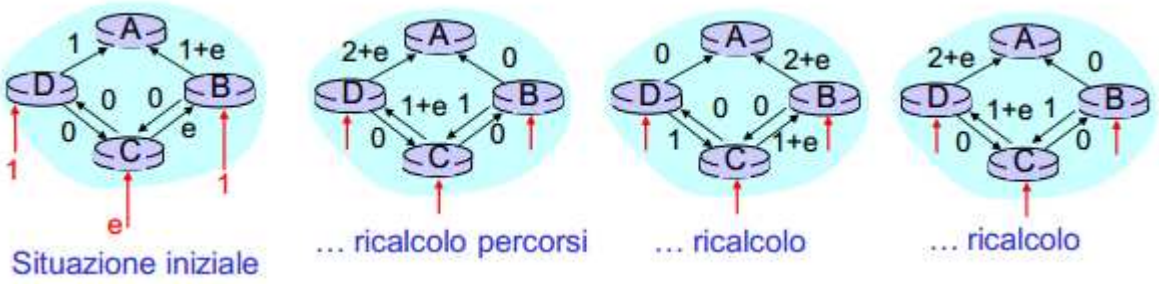
Per effettuare l'instradamento ho bisogno di tre elementi.

1. Protocolli di instradamento (routing protocols). Cioè le regole per la commutazione tra router. O meglio le regole per un corretto scambio di informazioni al fine di costruire le tabelle d'instradamento.
2. Algoritmi d'instradamento (routing algorithms). Calcola i percorsi creando le tabelle d'instradamento.
3. Procedura di forwarding (cioè inoltro di pacchetti). Inoltro dei pacchetti verso la porta d'uscita. Per farlo uso la tabella di instradamento.

Noi ci concentreremo sugli algoritmi d'instradamento. Obiettivo di quest'ultimi è di scegliere un buon percorso per l'inoltro dei pacchetti (chiamato anche percorso ottimo o migliore). Si assegna un costo ad ogni link che mi collega due nodi (il nodo sorgente e quello destinatario). In questo modo si cerca di ottimizzare le risorse di rete. Il "costo" di ogni link è valutato in base a diversi parametri: la distanza, il ritardo, la congestione, la tariffazione (euro).
Il percorso ottimo che si è scelto lo è fino a che il costo di quel link non varia. Se la rete è statica (cioè ha costante il numero di nodi, di link, e il costo) il percorso migliore una volta calcolato la prima volta, non lo si deve ricalcolare più, perché rimane sempre lo stesso. Se invece la rete è dinamica i costi possono cambiare e dovrò, pertanto, ricalcolare i percorsi. Questo mi può succedere se mi cambia il livello di congestione.
Quindi:

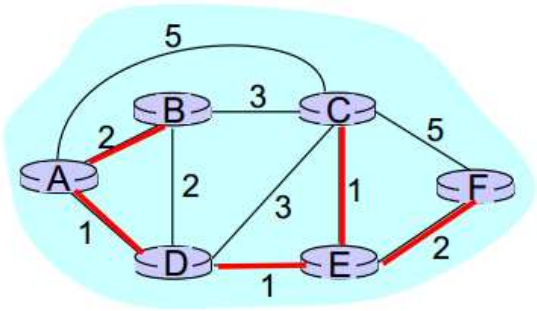
- costo statico: non posso assecondare i vari stati della rete, ma ho il vantaggio che non mi devo ricalcolare i percorsi;
- costo dinamico: devo ricalcolare i percorsi. Questo provoca però dei problemi d'oscillazione (trattato nel seguito).

Problema di oscillazione per reti con costi dinamici.



Se instrado per avere minimo carico su link si creerà un fenomeno oscillante, in cui ho il traffico tutto smaltito prima nella parte di destra della mia rete con tipologia ad anello, e successivamente ce l'ho nella parte di sinistra. Questa alternanza si protrae sempre così. Vediamo perché: io instrado una prima volta da C ad A. Posso percorrere 2 percorsi nel nostro esempio relativo alla figura sopra: o passo da B o passo da D. Nell'esempio si passa da B (il risultato ottenuto sarebbe stato uguale anche se fossimo passati da D). Quindi ho un costo sul link B→A di 1+e (1 dovuto al costo del link B→A, ed e dovuto al costo del link C→B).

ALGORIMTI LINK-STATE. Ogni nodo invia informazioni di costo dei soli suoi canali ed a quale nodo è collegato, in broadcast a tutti gli altri nodi della rete. Ogni nodo, quindi, conosce la topologia della rete ed ogni nodo conosce i costi di tutti i link. In questo modo ogni nodo si calcola il suo percorso minimo. Si ottengono quindi per ogni routing delle tabelle d'instradamento. L'algoritmo Link-State più utilizzato prende il nome del suo ideatore: è il Dijkstra. Usato per determinare i cammini minimi. Questo algoritmo funziona solo con costi positivi. Ad ogni step ho una nuova informazione che verrà poi inviata in broadcast a tutti gli altri nodi.



Supponiamo di applicare questo algoritmo al router A. Questo per crearsi la sua tabella di instradamento si crea i percorsi ottimi passando per i canali che costano di meno.

A→B: passa direttamente dal link diretto che costa 2 perché passando da D costerebbe 3.

A→C: non passa dal link diretto, perché costerebbe 5, mentre passa da D che costa 4. Quando si accorgerà che da D ad E e da E a C costa 2, non passerà più da D a C direttamente (con costo 4) ma passera da E (con costo 3).

A→D: passa direttamente dal link diretto di costo 1.

A→E: passa da D perché così il costo è di 2, mentre passando per B e C o direttamente da C il costo è 6. Qua scopre che passare da A a C tramite E costa 3, meno che da D che costava 4 (aggiorna la tabella di instradamento che riguardava il percorso A→C).

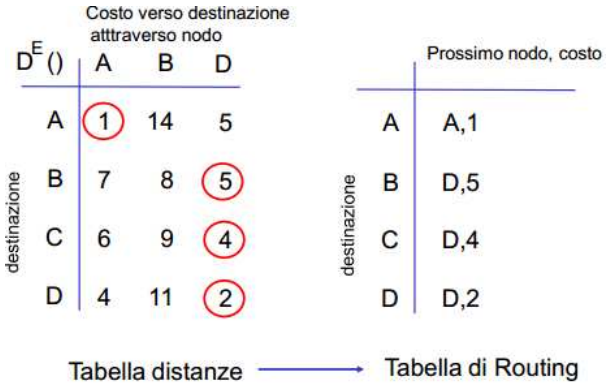
A→F: passa da D ed E, non da C. Infatti anche se c'è un canale in meno il costo sarebbe 10, mentre con l'altro percorso il costo è 4.

Quindi alla fine la tabella di instradamento sarà:

destinazione	Prossimo nodo, costo
B	B,2
C	D,3
D	D,1
E	D,2
F	D,4

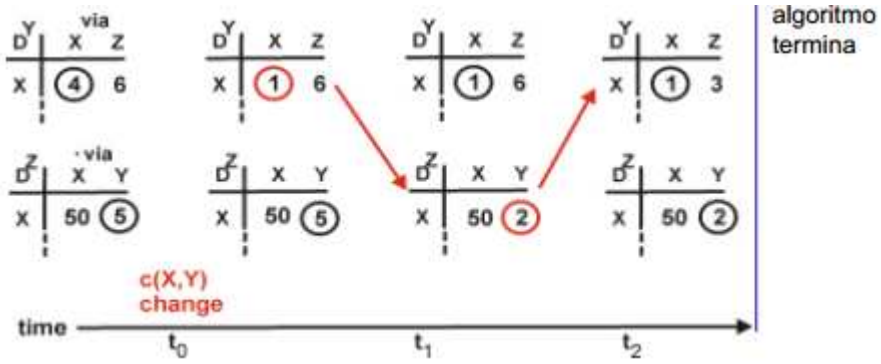
ALGORITMI DISTANCE VECTOR. È un algoritmo iterativo. È in esecuzione fino a che i nodi non si scambiano più informazioni. Termina in modo autonomo senza il bisogno di esplicitare un segnale di fine algoritmo. Come detto è un algoritmo distribuito con informazione parziale, per cui la sua caratteristica è che i nodi parlano solo con i nodi adiacenti.
All'inizio i percorsi migliori che conoscono i nodi sono solamente quelli ai quali i nodi sono direttamente collegati. Il modo di procedere è: io nodo ho un costo, tutti quelli collegati direttamente a me mi diranno il loro costo. Quando dovrò percorrere quel canale aggiungerò al mio costo, quello dell'altro nodo. Per scegliere il percorso migliore selezionerò solo i canali i cui costi sommati al mio costo sia minore degli altri. Supponiamo che io nodo A con costo 2 sia collegato direttamente ai nodi B e C. Mi arriva l'informazione che il nodo B ha un costo 3. Per ora il mio percorso migliore per inoltrare un pacchetto sarà a B (costo totale 5). Successivamente mi arriva l'informazione che il nodo C ha costo 2. Il nodo A pertanto dovrà aggiornare la tabella d'instradamento e come percorso migliore sceglierà quello passante da C (costo totale 4). Quindi se mi arriva un'informazione di un costo di un altro nodo adiacente (cioè collegato direttamente a me) che sommato al mio costo dà come risultato un costo totale minore di quello di prima, allora questo sarà il mio nuovo percorso migliore.
Un fattore negativo riguarda la criticità degli errori. Infatti se un nodo mi sbaglia a dire il suo costo, questo errore si propaga su tutti i nodi della mia rete.

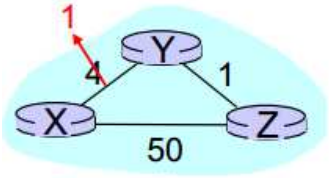
Quando un nodo apprende un percorso ottimo lo comunica ai nodi adiacenti. Se le informazioni che mi arrivano non migliorano il percorso migliore che il nodo ha già salvato nella sua tabella d'instradamento allora l'algoritmo si ferma. Nella pratica però dopo un certo periodo di tempo avviene un refresh. Si ha quindi un ricalcolo periodico dei percorsi e se nella rete non cambia nulla, allora la tabella d'instradamento sarà uguale identica a quella di prima.



<u>Si riscontra un problema quando i costi cambiano</u>.

Supponiamo che il <u>costo di un canale scenda</u> (come mostrato in figura). Y si accorge che per raggiungere X il costo da 4 è sceso ad 1. Quindi Y invia l'informazione a Z. Quest'ultimo allora scriverà il percorso migliore nella sua tabella d'instradamento.





– Prima del cambio di costo:

1° tabella D^Y.

Y per raggiungere X passando per X ha costo 4.

Y per raggiungere X passando per Z ha costo 6. Infatti: Y→Z (costo 1), Z→Y (perché se va direttamente a X il costo è 50, mentre se passa per Y è 1) e infine Y→X (costo 4): totale 1 + 1 + 4 = 6. Notiamo che questo è un anello. Pertanto il percorso migliore risulta essere il primo, cioè Y→X di costo 4.

1° tabella D^Z.

Z per raggiungere X passando per X ha costo 50.

Z per raggiungere X passando per Y ha costo 5. Infatti: Z→Y (costo 1), Y→X (costo 4): totale 1 + 4 = 5. Pertanto il percorso migliore risulta essere Z→Y→X di costo 5.

– Dopo il cambio del costo:

2° tabella D^Y.

Y per raggiungere X passando per X ha costo 1. Il costo si è ridotto (da un 4 si è passati ad un 1) e passa quest'informazione a Z.

3° tabella D^Z.

Z per raggiungere X passando per X ha sempre costo 50.

Z per raggiungere X passando per Y ora ha costo 2. Infatti: Z→Y (costo 1) ed Y→X (costo 1). Il costo si è ridotto (si è passati da 5 a 2) e quindi passa l'informazione ad Y.

4° tabella D^Y.

Y per l'informazione provenutagli da Z ricalcola il percorso di Y per raggiungere X passando da Z: costo 3. Infatti: Y→Z (costo 1), Z→Y (costo 1) e infine Y→X (costo 1): totale costo 3. Qua finisce l'algoritmo.

In questo modo, con costi che si riducono, riesco a propagare bene le informazioni (dette buone notizie, visto che il costo di un link che scende è un fattore positivo per quanto riguarda la rete).

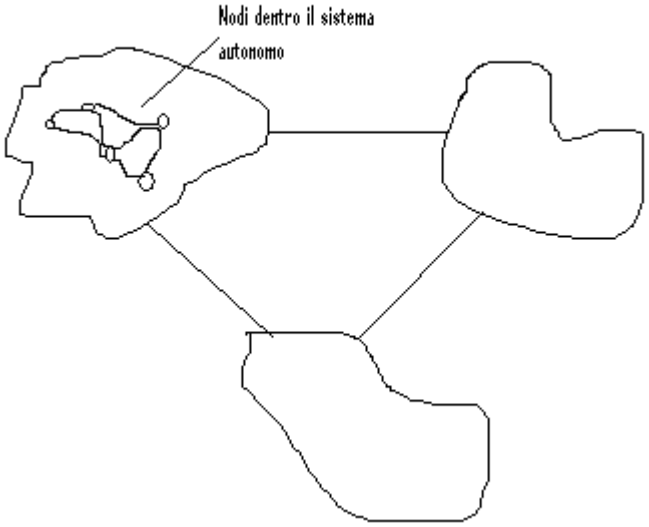
Questo procedimento non termina fino a che non si arriva al valore del costo del canale che è aumentato: nel nostro caso 60. Questo problema è definito count to infinity. Più è alto il valore del costo che è aumentato più tempo ci vorrà perché i nodi capiscano come stanno realmente le cose.
Possibile risoluzione del problema del count to infinity: se ho dei percorsi ottimi che so che passeranno per un nodo mio adiacente allora non dichiaro il nuovo percorso al nodo adiacente stesso, ma lo comunico agli altri miei nodi adiacenti, i quali non avevano percorsi ottimi che gli passavano attraverso. In questo modo non si inganna il nodo con un percorso fittizio (cioè il percorso ad anello citato sopra). Cmq questa soluzione non mi risolve ancora totalmente il problema.
Questi sono problemi di convergenza intrinsechi agli algoritmi soprattutto quando i costi aumentano.

Confronto tra algoritmi LS e DV.

⇒ Velocità di convergenza.
  - LS: ogni volta che un link state (cioè la nuova informazione riguardo lo stato di un canale) è propagato ho una nuova tipologia. La convergenza è immediata.
  - DV: le scelte del nodo dipendono dalle scelte dei nodi adiacenti. Pertanto si richiedono più scambi di messaggi. Il tempo di convergenza sarà variabile.

⇒ Affidabilità. Cosa succede se un nodo non funziona correttamente?
  - LS: i nodi possono annunciare costi dei canali scorretti. Ogni nodo crea la propria tabella d'instradamento ma tutti sbagliano. Però non si creano anelli. Al prossimo annuncio tutto si corregge.
  - DV: anche qua i nodi possono annunciare costi dei cammini scorretti. Ogni annuncio è però usato da tutti i nodi (indirettamente). Questi errori si propagano inesorabilmente nella rete e si possono creare anelli.

Nota: dentro i sistemi autonomi ho un insieme di nodi di rete e questo sistema vede nello specifico solo i propri nodi all'interno di esso. Per lui gli altri sistemi autonomi sono come grandi nodi (router) in cui non sa che cosa ci sia dentro (e neanche gli interessa dal punto di vista del funzionamento della rete). I sistemi autonomi interagiscono per scambiarsi informazioni, ma un sistema autonomo sorgente sa che deve spedirlo ad un altro e basta. Ci penserà poi il sistema autonomo destinatario (con la sua specifica topologia di rete, con i suoi protocolli e con i suoi algoritmi) a far arrivare l'informazione a destinazione).



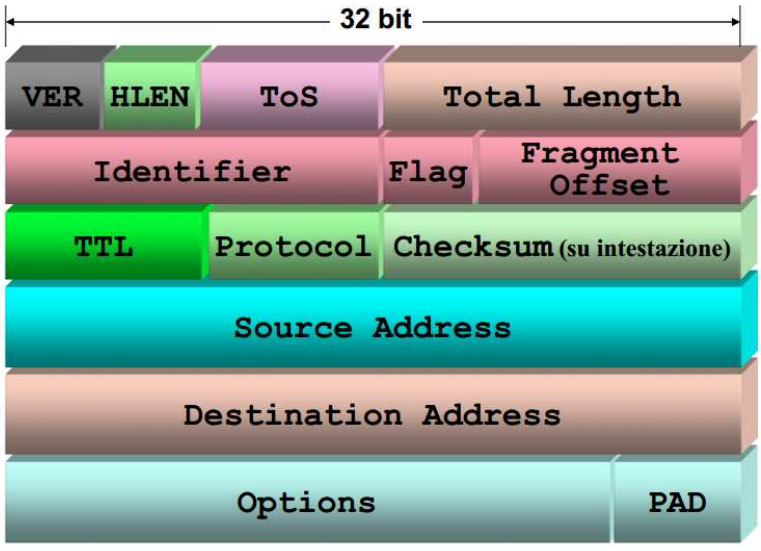
L'IP offre un servizio non connesso basato su un protocolli datagram (cioè a pacchetto). Quindi non ho circuiti virtuali di nessun genere. Io mi creo il pacchetto lo invio nella rete e, in base alle informazioni che ci sono nel pacchetto stesso, i nodi prendono delle decisioni rispetto alle loro routing table. Sono valide tutte le caratteristiche già viste per protocolli datagram, quindi non mi si garantisce nulla: non ho la certezza, quindi, che il mio pacchetto spedito in rete sia arrivato correttamente a destinazione. È il cosiddetto servizio best effort: cioè IP ti assicura che farà del suo meglio per trasportare al meglio i tuoi pacchetti ma non ti garantisce nulla.

Sebbene sia un protocollo datato, non è obsoleto, infatti lo stiamo usando ancora oggi. Tuttavia ai giorni nostri si è passati a standardizzare un altro protocollo simile che è l'IP-v6 con molti più indirizzi IP disponibili (2^{128}) rispetto all'IP-v4 (2^{32}). Questo passaggio è dovuto al fatto che stanno finendo gli indirizzi IP.

Datagram vs Connection Oriented. Nel datagram i pacchetti viaggiano su percorsi indipendenti, infatti vengono trattati uno indipendentemente dall'altro. Questo fa si che i pacchetti mi possano arrivare non in sequenza. Infatti le tabelle di routing e le altre sono dinamiche, cioè cambiano di continuo. Se per esempio un link si rompe, i protocolli di routing devono definire un percorso nuovo bypassando il link rotto. I pacchetti prenderanno strade diverse e mi potranno arrivare non in sequenza.
Inoltre IP non mi riserva una banda per un determinato traffico. Quindi posso avere problemi per servizi quali videochiamate, streaming, telefonate, ecc..
Tutti questi svantaggi hanno però il vantaggio di far in modo che il protocollo sia poco complesso. Non si richiede, infatti, nessun tipo di negoziazione, né al lato utente, né all'interno della rete. Inoltre è robusto: cioè si adatta a variazioni di traffico, alla topologia della rete, ad eventuali guasti, ecc..
È molto adatto ovviamente al traffico dati (bursty). Per esempio una mail può anche arrivare in ritardo perché non mi aspetto una risposta immediata. Altro esempio il caricamento di una pagina Internet, il cui tempo ammissibile è di pochi secondi, che per la rete è un tempo enorme.

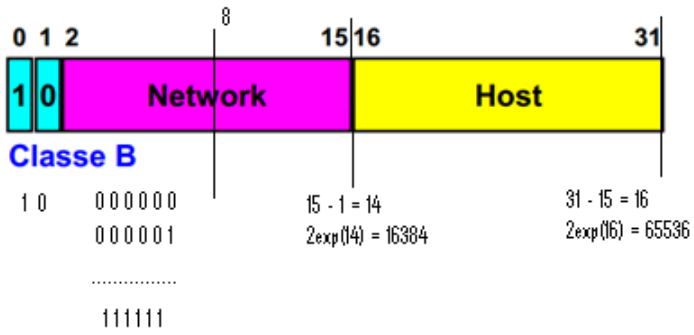
Funzionalità dell'IP. In passato si usava per la frammentazione dei pacchetti, ma oggi non lo si usa più. Gestisce indirizzi a 32 bit a livello di rete e host. Esegue il routing e configura diverse classi di servizio. Queste classi di servizio hanno senso, cmq, se non sono troppe. Facciamo un esempio: se ho una classe premium i cui pacchetti hanno precedenza se ne ho troppe in una determinata rete, anche se questi pacchetti hanno la precedenza, posso "incontrare" pacchetti simili di un'altra classe premium che hanno anche loro precedenza, e per cui è come se fossero pacchetti normali. Se mi ritrovo troppi pacchetti di quella caratteristica, mi inficia, quindi, il mio privilegio di base ed è inutile. Sarebbe come non averlo.

Formato dell'intestazione IPv4.
Questa è l'intestazione che aggiungo al segmento che mi arriva dal livello 4 per creare il mio pacchetto.

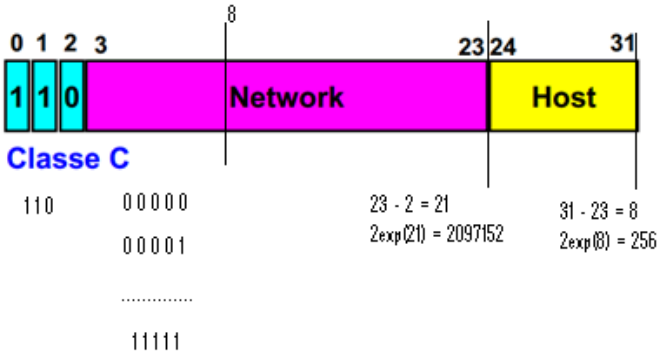


VER. Indica la versione del protocollo. Nel nostro caso la 4.
HLEN. Header Lenght (lunghezza dell'intestazione) in blocchi di 4 byte (max 64 byte).
ToS. Type of Service. Specifica la classe di servizio. A secondo del servizio, i router trattano i pacchetti in modo diverso.
Lunghezza totale. Lunghezza globale del pacchetto corrente (non quello prima della frammentazione), max 2^{18} byte.
Identificatore. ID univoco del pacchetto (o datagram). Questo è costante nel caso di frammentazione. È necessario per la deframmentazione, cioè per unire il pacchetto alla ricezione.
Flag. È posto a 0. Si può settarlo a DF, cioè Don't Fragment per non segmentare il mio pacchetto. Si noti che se il pacchetto ha come flag DF ed è più grande della massima lunghezza, il pacchetto viene scartato dal router (proprio perché è troppo lungo e non può frammentarlo per il campo DF). Se invece è settato

- Classe B. Il campo network ha 14 bit. Dopo il 1 0 ho ancora 6 bit che formano il mio byte. In questi 6 bit ho valori compresi tra 128 (= 10 000000) e 191 (= 10 111111). Ho più reti (circa 16000) ma meno spazio per indirizzare gli host (circa 64000).



- Classe C. Il campo rete ha 21 bit. Dopo l'1 1 0 ho ancora 5 bit che mi formano il mio byte. In quei 5 bit ho valori compresi tra 192 (= 110 00000) e 223 (= 110 11111). Avrò 2^{21} (= 2 milioni circa) reti diverse ma solo 2^{8} (= 256) host disponibili.



Indirizzi particolari.

Net	All 0s	The network
All 1s		Limited broadcast (local net) [1]
Net	All 1s	Directed broadcast for net [1]
127	Anything (often 1)	Loopback[2]

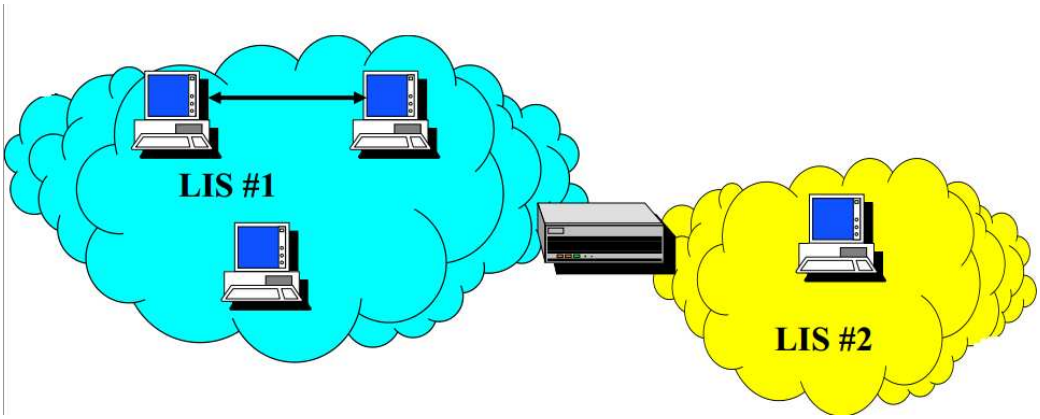
[1] Può essere usato solo come indirizzo destinazione
[2] Non deve essere propagato dai router sulla rete

- The network: è l'indirizzo che mi identifica solo la rete, non l'host. Nell'esempio precedente della Classe A avrò il 2 e poi tutti zeri: 2 . 0 . 0 . 0.
  Questo significa che all'interno di ogni rete il primo indirizzo di net allo 0s e l'ultimo net all'1s non posso utilizzarli nel scegliere l'indirizzo IP degli host nella rete. Nella classe A saranno quindi 2^{24}-2 gli host possibili da definire. Stesso ragionamento per le altre classi.
- Limited broadcast (local net): raggiunge tutte le stazioni di Internet. Questo, però sembra andare contro il principio dell'IP spiegato prima e cioè di limitare il broadcast (fatto dai router). Però anche in questo caso c'è una delimitazione infatti non vado ad inoltrare in modalità broadcast a tutti gli host di Internet, ma solo agli host della mia rete IP. Cioè vado solo negli host definiti a livello 2 a cui è connessa la rete IP.
- Direcred broadcast for net: invece di identificare la mia di rete, il broadcast identifica una rete (net) in particolare.
- Loopback: identifica la macchina stessa. È un traffico diretto a me stesso, infatti me lo restituisce a livello locale. Un esempio può essere rappresentato da un debug di un sito web che l'utente ha creato con html.

Come avvengono le comunicazioni.

Routing IP.

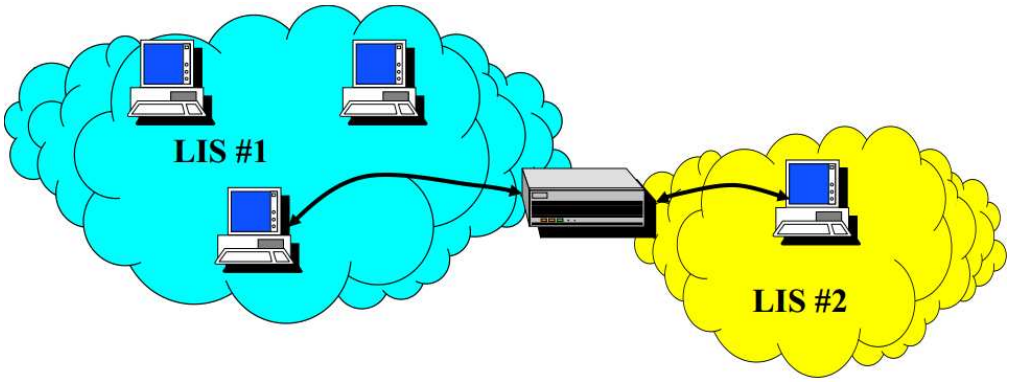
Se ho due host che vogliono comunicare e sono tutti e due sulla stessa rete logica, posso utilizzare i metodi di livello 2 visti fino ad ora.



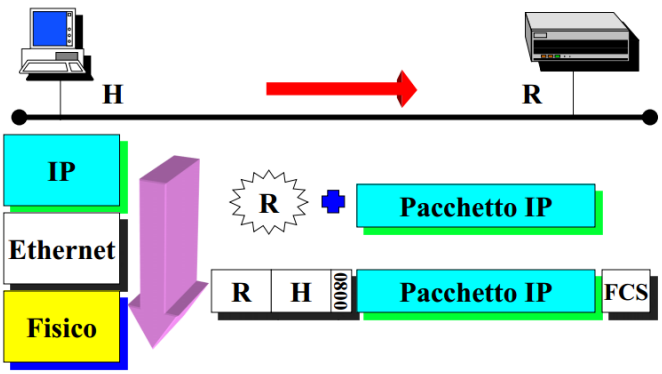
Al contrario se devo raggiungere un host che non è nella mia stessa rete logica, questo non sarà neanche nella mia stessa rete fisica, e quindi ho bisogno di un router, cioè un dispositivo di livello 3, che mi delimita le reti fisiche. Quindi: utilizzo un'intelligenza di livello 2 per raggiungere il router. Questo con intelligenza di livello 3 capirà dove mandare il pacchetto. La rete che lo riceverà, con intelligenza di livello 2 lo farà arrivare a destinazione.

In estrema sintesi:

- la consegna tra LIS differenti è affidata ai router ed avviene a livello di rete;
- l'host conosce almeno un default gateway fornito in fase di configurazione degli host. (Nel protocollo DHCP, questo viene configurato in automatico).

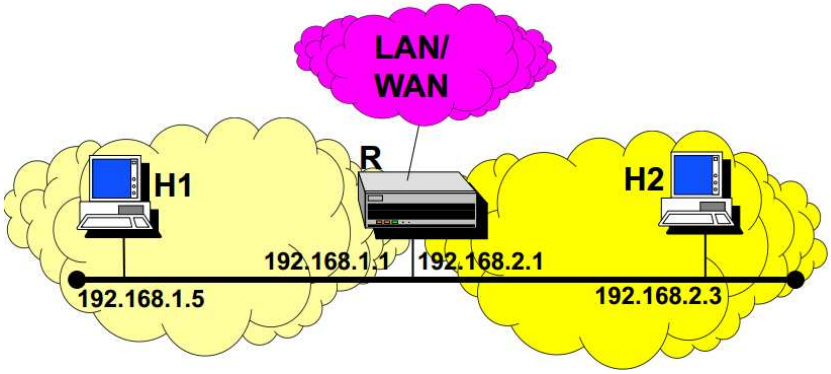


Consegna diretta. In questa situazione siamo nel livello 2.



L'host da raggiungere è nella stessa LIS, quindi non ci si sposta nel livello 3.
La procedura quindi sarà di creare la trama che verrà inviata dal livello 2.

router R. Qua si tornerà al livello 2, con la creazione della trama che questa volta includerà l'indirizzo sorgente (R) e quello destinatario (H2). Ovviamente le due trame sono diverse. Si differenziano infatti dagli indirizzi.
Si noti che gli indirizzi IP non sono toccati. Restano uguali in tutto il percorso. Quando H1 spedisce il pacchetto ad R, il livello 3 IP metterà nella sua trama l'indirizzo IP sorgente di H1 e l'indirizzo IP destinatario di H2.



Ho solo una porta fisica, anche se le interfacce (logiche) sono due.

Gli host che appartengono ad una sottorete logica possono inviare i pacchetti destinati ad altre sottoreti al router, se nelle loro routing table non c'è la destinazione; direttamente a destinazione, se nelle loro routing table c'è la destinazione.

Dopo la prima comunicazione tra gli host H1 ed H2, l'H1 si può accorgere che entrambi sono sulla stessa rete fisica (anche se su reti logiche (LIS) diverse) se riceve il messaggio ICMP xRedirect. Questo messaggio infatti permette di aggiornare in modo debito le routing table.

Come detto, questa soluzione (di avere host sulla stessa rete fisica ma su LIS diverse) non è usata praticamente mai oggigiorno, ma forse potrebbe prendere il sopravvento in futuro. Tuttora è in preparazione ed espansione.

Subnetting.
Riprendiamo il concetto di classfull concentrandoci sui suoi possibili problemi.
Scegliendo la classe B io sono costretto a comprarmi 2^{16} indirizzi. Bisogna tenere presente che ho sempre l'obiettivo di semplificare i nodi (i router), cioè di fare in modo che le loro routing table non abbiano tante righe.
Posso raggiungere questo obiettivo comprando più indirizzi di quelli che mi servono al momento stesso dell'acquisto. Se avessi un modo di mantenere la corrispondenza biunivoca tra reti LIS e fisiche e potessi definire più LIS, potrei assegnare ad una LIS più reti fisiche. Posso così semplificare le routing table con una entry.
Con l'indirizzo classfull non riesco a rispettare la corrispondenza biunivoca, cioè non riesco ad assegnare ad una LIS più reti fisiche. Questo perché nel classfull ho una divisione statica tra indirizzo rete e di indirizzo di host.

Esempio.

130 . 192 . 0 . 0	130 . 192 . 1 . 0
.	.
. LAIB 1	. LAIB 2
.	.
130 . 192 . 0 . 255	130 . 192 . 1 . 255

Ora non so quale sia la parte di rete. Ci devo mettere la Subnet Mask.
Questa Subnet Mask non viene inviata. La utilizzo solo nei dispositivi (che possono essere sia host che router). Devo configurarla all'indirizzo IP ed al default gateway.

I valori decimali leciti nei 4 byte che costituiscono la netmask sono quindi:

128	1000 0000	(128)	
192	1100 0000	(64)	→ ESEMPIO.
224	1110 0000	(32)	
240	1111 0000	(16)	→ esempio foglietto.
248	1111 1000	(8)	→ 2^3 (3 spazi per gli host)
252	1111 1100	(4)	→ 2^2 (2 spazi per gli host)
254	1111 1110	(2)	→ 2^1 (1 spazio per l'host)
255	1111 1111	(1)	

La 255 non ha senso, perché essendo tutti 1 non ho nessuno 0 e quindi nessuno spazio per gli host. La 254 in teoria è valida ma ho solo uno spazio di host disponibile, che è 1 o 0. Se è uno avrò il broadcast, mentre se metto lo 0 è la rete.
La 252 è la netmask che mi configura i 2 tipi di rete più semplice possibile (con link PPP):
1111 1100 → 2^2 (= 4 spazi disponibili meno 2 per la rete e il broadcast)
Infatti ho 4 possibili configurazioni:
0 0 → rete
0 1
1 0
1 1 → broadcast
Avrò quindi 2 soli host che avranno un link PPP.

Così anche per le altre.
Vediamone ancora una per capire meglio:
la 248 è la netmask 1111 1000 → 2^3 (= 8 spazi per gli host - 2).
Ho 16 possibili configurazioni:
0 0 0 → rete
0 0 1
0 1 0
1 0 0
0 1 1
1 0 1
1 1 0
1 1 1 → broadcast

Quindi in definitiva ne ho $2^{\text{numero di host possibili}}$ – 2 (per la configurazione della rete e il broadcast).

Nota: esistono subnet con 1 e 0 non contigui ma non le vedremo.

... combinazioni host

192 . 168 . 10 . 31 → (11000000 10101000 00001010 0001|1111) dove $00011111_2 = 31_{10}$

- Rete 3:
192 . 168 . 10 . 32 → (11000000 10101000 00001010 0010|0000) dove $00100000_2 = 32_{10}$

... combinazioni host

192 . 168 . 10 . 47 → (11000000 10101000 00001010 0010|1111) dove $00101111_2 = 47_{10}$

- Rete 4:
192 . 168 . 10 . 48 → (11000000 10101000 00001010 0011|0000) dove $00110000_2 = 48_{10}$

... combinazione host

192 . 168 . 10 . 63 → (11000000 10101000 00001010 0011|1111) dove $00111111_2 = 63_{10}$

- Rete 5:
192 . 168 . 10 . 64 → (11000000 10101000 00001010 0100|0000) dove $01000000_2 = 64_{10}$

... combinazione host

192 . 168 . 10 . 79 → (11000000 10101000 00001010 0100|1111) dove $01001111_2 = 79_{10}$

- Rete 6:
192 . 168 . 10 . 80 → (11000000 10101000 00001010 0101|0000) dove $01010000_2 = 80_{10}$

... combinazione host

192 . 168 . 10 . 95 → (11000000 10101000 00001010 0101|1111) dove $01011111_2 = 95_{10}$

- Rete 7:
192 . 168 . 10 . 96 → (11000000 10101000 00001010 0110|0000) dove $01100000_2 = 96_{10}$

... combinazione host

192 . 168 . 10 . 111 → (11000000 10101000 00001010 0110|1111) dove $01101111_2 = 111_{10}$

- Rete 8:
192 . 168 . 10 . 112 → (11000000 10101000 00001010 0111|0000) dove $01110000_2 = 112_{10}$

... combinazione host

192 . 168 . 10 . 127 → (11000000 10101000 00001010 0111|1111) dove $01111111_2 = 127_{10}$

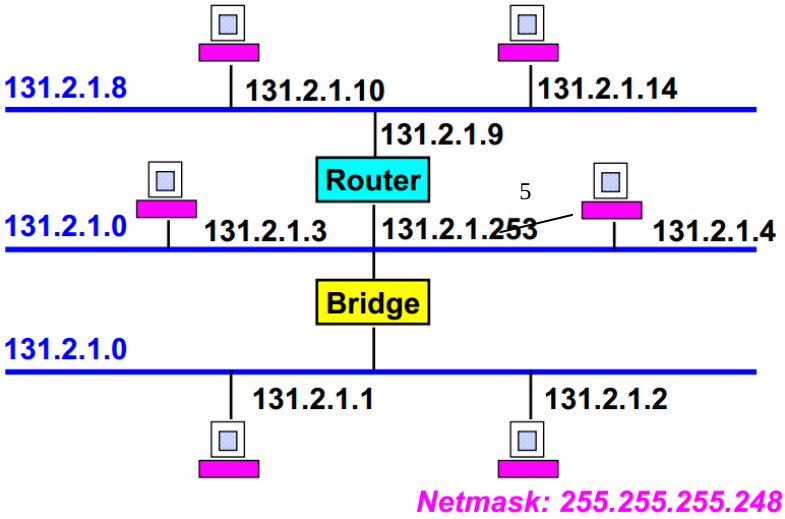
- Rete 9:
192 . 168 . 10 . 128 → (11000000 10101000 00001010 1000|0000) dove $10000000_2 = 128_{10}$

... combinazione host

192 . 168 . 10 . 143 → (11000000 10101000 00001010 1000|1111) dove $10001111_2 = 143_{10}$

- Rete 10:
192 . 168 . 10 . 144 → (11000000 10101000 00001010 1001|0000) dove $10010000_2 = 144_{10}$

... combinazione host



L'address range da cui posso attingere in questo esempio è 131 . 2 . 1 . 0 con netmask 255 . 255 . 255 . 0

La netmask dell'esempio è 255 . 255 . 255 . 248 quindi con l'ultimo byte uguale a 11111000 (=248_{10}). Questo significa che la maschera mi assegna un massimo di 2^3 – 2 host (perché devo sottrarre il network e il broadcast). Quindi le reti ammissibili sono tutte quelle multiple di otto. La prima ammissibile, pertanto oltre la 0 (che è multiplo di qualsiasi numero) è la 8. Per cui nell'esempio si è messo 131 . 2 . 1 . 8 (ma si poteva mettere qualsiasi multiplo di 8, come 131 . 2 . 1 . 16, oppure . 64). Nell'esempio in considerazione posso ancora attingere dall'address range . 16 fino al . 255.

Nota: quindi per capire quanti host ci sono al più in una rete guardo sempre il numero di zeri della netmask partendo da dx.
Per determinare le reti ammissibili:
- se il numero di zeri non mi sfora l'ultimo byte:
255 . 255 . 255 . 252 → 11111111 11111111 11111111 11111100
allora conto il numero di zeri e l'elevamento a potenza con base 2 (2^2=4) mi dà il multiplo ammissibile per le reti;
- se il numero di zeri mi sfora l'ultimo byte:
255 . 255 . 255 . 254 → 11111111 11111111 11111110 00000000
allora conto il numero di zeri del 3°byte, e l'elevamento a potenza con base 2 (2^1=2) mi dà il multiplo ammissibile per le reti.

Questi concetti sono molto importanti e devono essere molto chiari.
Vediamo altri esempi.

Es. 1

1010 1100.0001 0000.0001 0000.0000 0000 *1111 1111.1111 1111.1111 1110.0000 0000*	172.16.16.0 *255.255.254.0*	Rete locale con al più 510 host

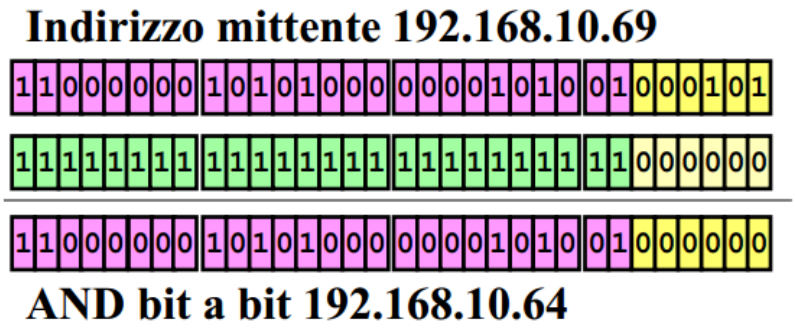
Abbiamo una netmask con 9 zeri, da cui 2^9 – 2 = 512 – 2 = 510 al più host.
Gli zeri della netmask mi sforano nel 3°byte, quindi per sapere le reti ammissibili devo vedere quanti zeri ci sono nel 3° byte in questione. Nel nostro caso ce n'è uno (1111 1110). Pertanto 2^1 = 2. I multipli di 2 saranno le reti ammissibili.
Infatti l'indirizzo di rete sopra esposto è 172 . 16 . 16 . 0 cioè una prima rete su cui ci saranno le combinazioni degli host, la seconda rete ammissibile sarà 172 . 16 . 16 . 2, sempre con le sue combinazioni di host, una terza rete potrebbe essere (non andando in ordine con i multipli di 2) 172 . 16 . 16 . 32, e così via.

Es. 2

1010 1100.0001 0000.0100 0000.0000 0000 1010 1100.0001 0000.0100 0001.0000 0000 1010 1100.0001 0000.0100 0010.0000 0000 *1111 1111.1111 1111.1111 1111.0000 0000*	172.16.64.0 172.16.65.0 172.16.66.0 *255.255.255.0*	Reti locali con al più 254 host

Abbiamo una netmask con 8 zeri. Per cui: 2^8 – 2 = 256 – 2 = 254 al più host.
In questo caso non mi si sfora con gli zeri nel 3° byte, per cui guarderò gli zeri dell'ultimo byte (che sono uguali a quelli per cui abbiamo calcolato gli host). Quindi 2^8 = 256, per cui le reti ammissibili sono tutti i

Routing: AND bit a bit. È un algoritmo che un router deve eseguire per decidere se effettuare una consegna diretta oppure indiretta. In altre parole stabilisce se l'indirizzo a cui deve mandare il pacchetto è oppure no nella sua stessa LIS.

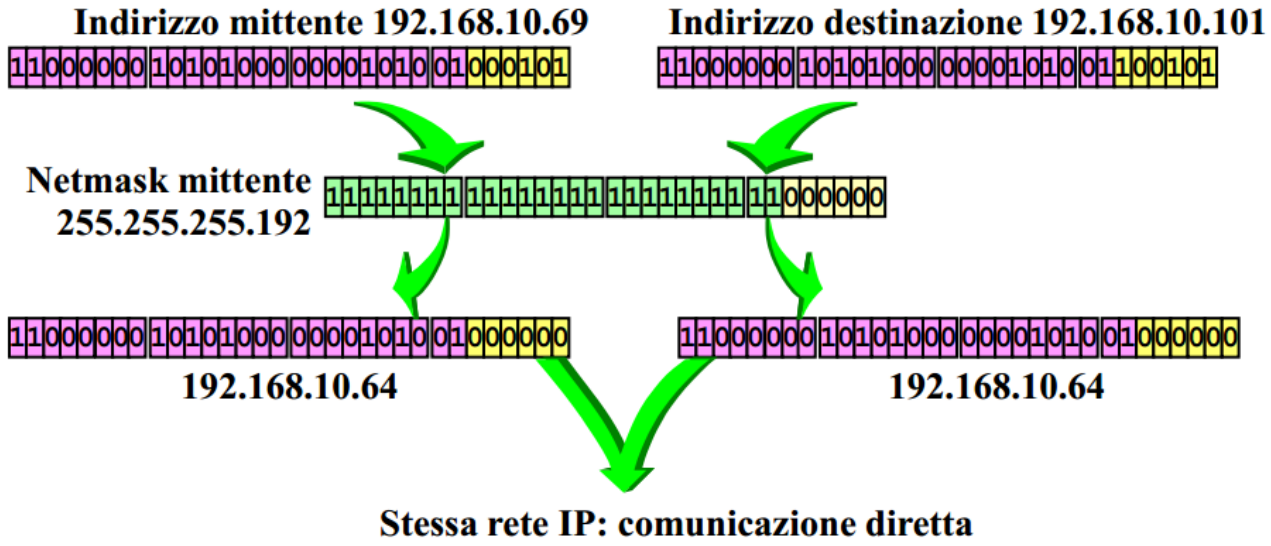


In questo caso l'indirizzo di rete dell'host è il primo delle tre righe di stringhe. Per sapere a che rete appartiene ragioniamo esattamente come spiegato prima: ho 6 zeri nella netmask, per cui ho 2^6 = 64. L'indirizzo dell'host è 69 (= 01000101_2), per cui dobbiamo chiederci: "qual è il più grande multiplo di 64 minore di 69?". Ovviamente la risposta è 64. Quindi l'identificativo di rete sarà 192 . 168 . 10 . 64, come scritto a fianco.

Netmask mittente 255.255.255.192

Ora spieghiamo l'algoritmo.
L'algoritmo AND bit a bit mi esegue l'operatore AND tra l'indirizzo di rete dell'host e la netmask. Se questo risultato dà come risultato per l'indirizzo destinatario lo stesso identificativo di rete dell'host mittente (nel nostro esempio 192 . 138 . 10 . 64) allora significa che gli host saranno nella stessa LIS, pertanto potrà essere raggiunto con protocolli di livello 2, effettuando una consegna diretta.

Vediamo due esempi:



L'address range (o identificativo di rete) è lo stesso del caso precedente. Abbiamo visto, infatti che l'indirizzo dell'host mittente 192 . 168 . 10 . 69 appartenere all'identificativo di rete 192 . 168 . 10 . 64. Per controllare che la consegna sia diretta l'indirizzo di rete del destinatario deve avere lo stesso identificativo di rete. Controlliamo. L'indirizzo dell'esempio è 192 . 168 . 10 . 101. Dobbiamo calcolare il suo identificativo di rete: ragioniamo come prima, per cui guardiamo la netmask del mittente: ha 6 zeri, pertanto 2^6=64. Qual è il maggior multiplo di 64 minore di 101? È 64, pertanto l'identificativo di rete è lo stesso, sia per il mittente che per il destinatario. La consegna può avvenire direttamente, con protocolli di livello 2.

Come si scrive la routing default?

0 . 0 . 0 . 0 0 . 0 . 0 . 0 Next Hop
Rete Netmask

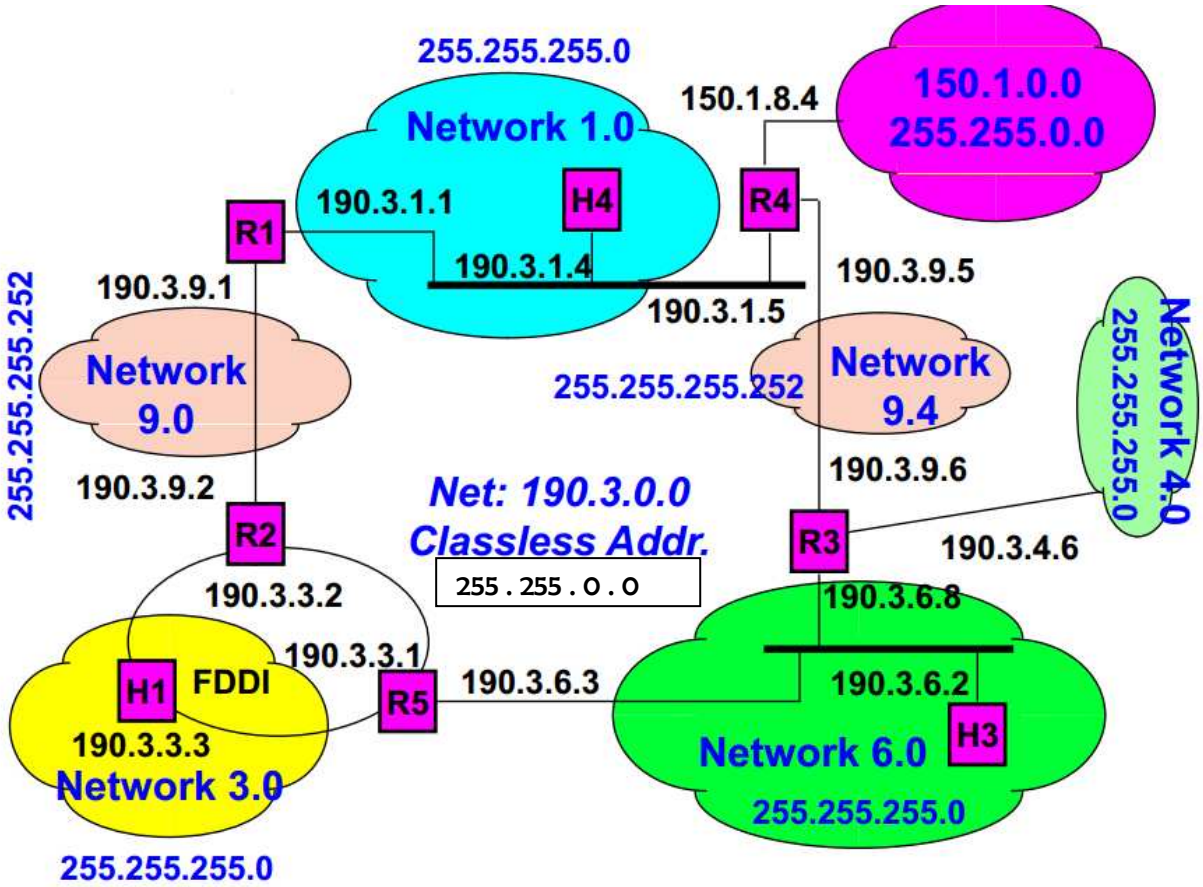
Questo significa "qualsiasi indirizzo" ed è indicato nel nostro host come default gateway e lo troviamo nella routing table come default route.
Infatti sono sicuro che avrò sempre il matching di bit utilizzando l'algoritmo di routing introdotto poco fa: l'AND bit a bit. Questo però avrà una priorità ridicola, perché come definito dall'algoritmo Longest Prefix Matching, qualunque altra stringa avrà un LPM più lungo della stringa con tutti zeri.

Entry sulle tabelle di routing. Sono di tre tipi.

1. Dirette. Address range corrispondenti alle interfacce del router (cioè tutte le reti raggiungibili direttamente).
2. Statiche. Route configurate staticamente dal gestore (è quindi l'operatore che le definisce).
3. Dinamiche. Address range appresi attraverso un protocollo di routing (quindi in modo automatico). Oppure apprese tramite un protocollo detto ICMP redirect (accenneremo più avanti, ma non lo vedremo nel dettaglio).

Nel caso la route per uno stesso address range sia appresa da diverse fonti deve essere specificato quale deve essere preferita. Si fa aggiungendo una metrica, cioè un'altra colonna che mi definisce il "peso" di quella determinata entry.

Esempio.



Nota: nella casella R5 non ho 3 interfacce ma solo 2, quella verso 190 . 3 . 6 . 3 e quella verso R2. Analogamente R2.

PROTOCOLLI INTERNET.

→ Lezione 02. ARP e RARP.

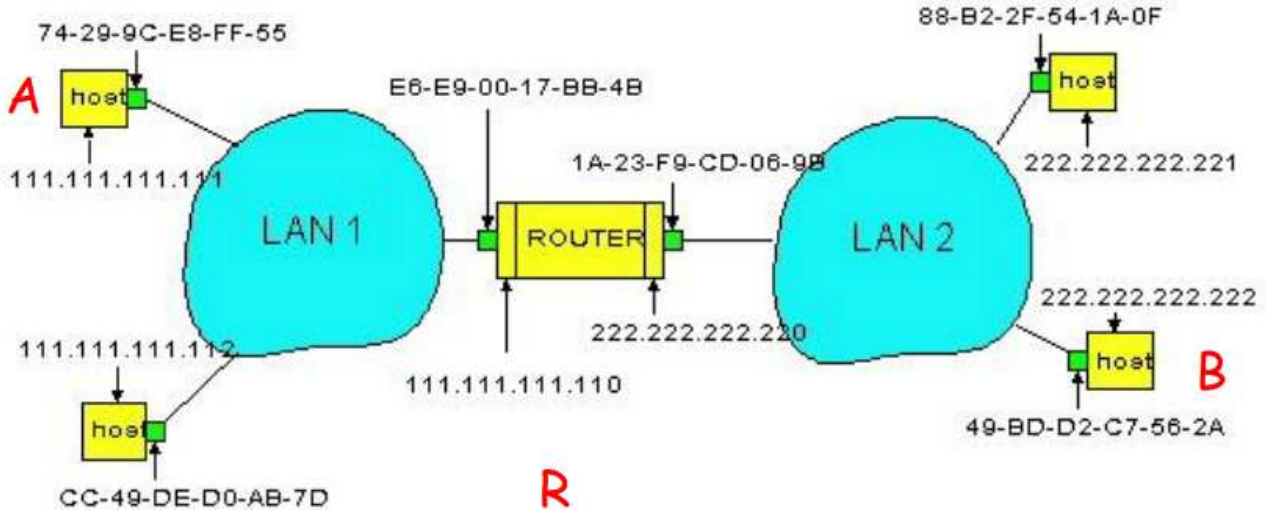
1. ARP.

L'ARP è un protocollo "dentro" IP, non perché abbia funzionalità di livello 2, ma perché è "appoggiato" sul livello 2. (Cioè, il pacchetto ARP è messo in una trama di livello 2).
Il problema che l'ARP vuole risolvere è come conoscere il MAC address a partire dall'IP address. Ogni macchina IP ha una tabella ARP nella quale ci sono scritte le associazioni tra indirizzi di livello 2 (gli indirizzi MAC) e gli indirizzi IP.

ARP in una stessa LAN.
Supponiamo che l'host A debba spedire un pacchetto (o datagram) all'host B. Inoltre supponiamo che A sappia l'indirizzo IP di B.
A consulterà la sua tabella ARP, ma non troverà corrispondenza relativo all'indirizzo IP di B e il suo indirizzo MAC (tabella vuota all'inizio).
L'host A invierà una richiesta broadcast creando una trama ARP Request dove chiede: "chi ha il seguente indirizzo IP?" (ovviamente l'indirizzo IP suddetto deve essere presente nella trama del mio pacchetto ARP Request).
Questo messaggio viene spedito in maniera broadcast, e quindi viene recepito da tutti gli host che sono connessi a quella determinata LAN. Il dispositivo che troverà la corrispondenza del suo indirizzo IP creerà a sua volta un pacchetto ARP Reply e spedirà, questa volta in unicast (infatti non c'è ragione di far sapere a tutti gli host di questa comunicazione) il pacchetto all'host richiedente (nel nostro caso l'host A). Ovviamente l'host che crea questo pacchetto ARP Reply manderà come informazione il suo indirizzo di livello 2 (cioè il suo indirizzo MAC).
Quando gli arriva il pacchetto ARP Reply, A salverà nella sua ARP table l'indirizzo MAC associato al livello IP di cui aveva mandato la richiesta di risposta. Questa informazione sparisce quando scade il timeout.

Quindi in tutti questi passaggi viene creata una trama ARP, ma non viene usato il livello 3 (cioè IP). Su questo punto osserviamo quanto segue: nella trama ARP Request (e anche nella Reply) compare l'indirizzo IP per permettere ai dispositivi di poter confrontare tali indirizzi e capire se la richiesta ARP è per loro oppure no. Però l'indirizzo IP lo inserisco nel paylod ARP cioè nella trama ARP, non nella sua intestazione. Infatti non c'è da nessuna parte l'header IP. Questo fatto sarà rimarcato anche in seguito perché è una caratteristica importante del protocollo ARP.

ARP verso un'altra LAN.



Assumiamo sempre l'ipotesi che A conosca l'indirizzo IP di B. Seguiamo passo passo i vari step della richiesta e risposta ARP.

1. A farà un controllo per sapere se B è nella stessa sua LIS oppure no. Per eseguire ciò elabora il programma di routing AND bit a bit visto nella precedente lezione e se il risultato corrisponde con il suo identificativo di rete significa che B appartiene alla sua stessa LIS, altrimenti no. In questo

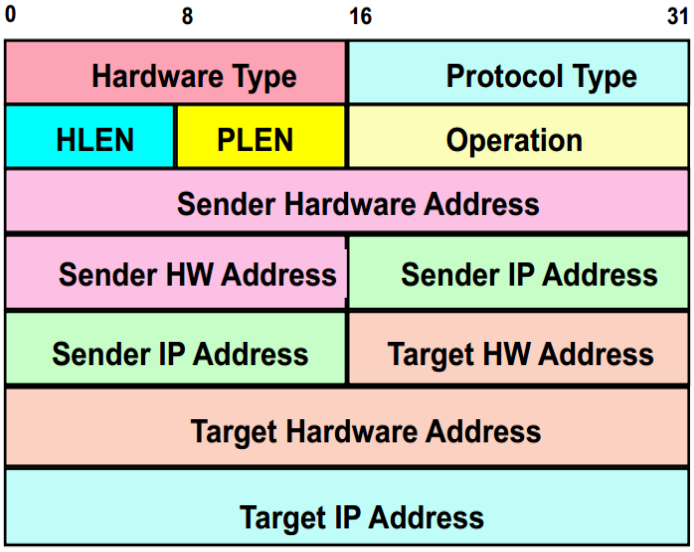
Il campo IP E è l'indirizzo a cui voglio associare l'indirizzo MAC. Come detto nell'ARP Request e nell'ARP Reply non c'è l'header IP. Conterranno invece dei campi che incrementeranno il protocollo ARP (di livello 2) non dell'IP.

La parte verde, quando viene ricevuta dall'host che gli deve dire il suo indirizzo MAC (nell'esempio sopra l'host E) viene "distrutta" e inserirà nella trama ARP nel campo al posto degli ?? (in realtà questo campo è vuoto) il proprio indirizzo MAC. Per questo motivo il campo MAC è ripetuto nella trama MAC (verde) e nella trama ARP, o il paylod, o SDU ARP (rosa).

Nota: il campo IP A ed IP E della trama serve per far capire all'host corrispettivo che voglio sapere il suo indirizzo MAC di livello 2. Ma questo campi non sono nell'header, sono nel paylod ARP.

Formato del pacchetto ARP.

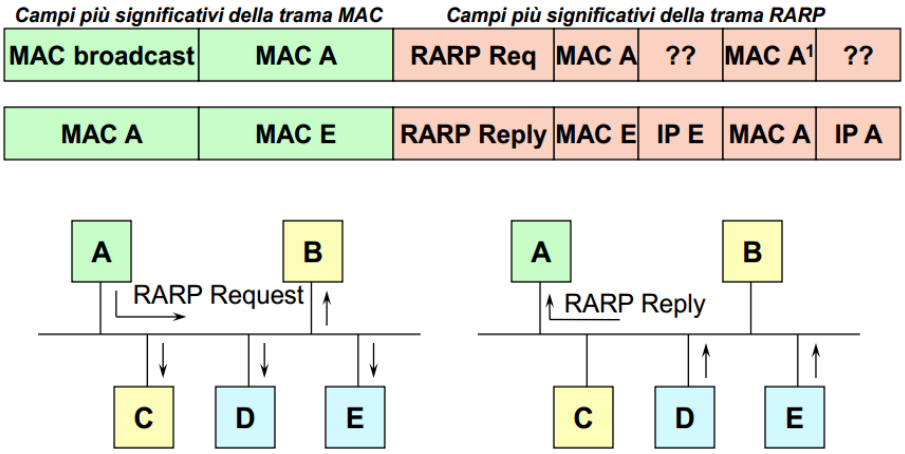
Vediamo i campi principali.



- Target hardware address. Indirizzo fisico di livello 2 a cui sono interessato. Nell'ARP Request non ha ragione di esistere, quindi questo campo sarà vuoto.
- Sender hardware address. È ripetuto due volte per rispettare l'architettura a layer. Infatti la parte della trama MAC (verde nel disegno sopra riportato) viene distrutta dal livello due dell'host che la riceve per metterci la sua e quindi il modulo ARP non lo leggerà, però deve avere cmq la possibilità di avere l'informazione che gli serve in quel pacchetto (parte rosa).
- HLEN, PLEN. Dimensione degli indirizzi di livello 2 e di livello 3 su cui avremo a che fare nei livelli successivi.

2. RARP.

Questo protocollo è molto meno importante di ARP ed è stato sostituito al giorno d'oggi dal DHCP. Il formato del pacchetto è lo stesso.
A partire da un indirizzo MAC cerca di ottenere un indirizzo di network per quella stazione.
Il grande svantaggio di RARP, che ha portato al suo disuso, è stato l'impossibilità di slegarsi dal classfull, infatti nei pacchetti RARP non c'è posto per poter definire la netmask. Il DHCP, invece, che ha sostituito RARP ed oggi è utilizzato moltissimo, mi permette non solo la definizione della netmask, ma anche molto altro, come l'indirizzo IP, il default gateway, il DSN ed altri campi.



Il richiedente emette un pacchetto di livello 2 in broadcast richiedendo l'indirizzo network relativo al proprio indirizzo hardware. I RARP servers rispondono con un pacchetto unicast di livello 2 e l'host sceglie una delle risposte arrivate (tipicamente la prima). Come detto, però, oggi non si usa praticamente più.

[1] Lo standard precisa che l'indirizzo MAC del destinatario, non essendo conosciuto, va posto pari a quello del mittente