RIASSUNTO RETI

NOVARESE DANIELE

Definizioni generali

Comunicazione >> trasferimento di informazioni secondo convenzioni prestabilite

Telecomunicazione >> qualsiasi trasmissione e ricezione di segnali che rappresentano segni, scrittura immagini e suono, informazioni di qualsiasi natura, *attraverso cavi, radio o altri sistemi ottici e elettromagnetici*.

- *Gli apparecchi telefonici sono terminali di utente collegati a una rete che fornisce servizi di telecomunicazione*

Servizio di telecomunicazione >> ciò che viene offerto da un gestore pubblico / privato ai propri clienti al fine di soddisfare una specifica esigenza di telecomunicazione

- *Funzioni di una rete di telecomunicazioni >>>* operazioni svolte all'interno della rete al fine di offrire servizi (es. inizio di procedura di chiamata, ***segnalazione di utente***)

Servizi portanti, teleservizi e servizi complementari:

- **Portanti >** offerti dalla linea telefonica – essenziale senza il quale non possiamo avere gli altri tipi di servizi; fornisce possibilità di trasmissione di segnali tra interfaccia utente-rete.
- **Teleservizi >** servizi abilitati dai servizi portanti
- **Supplementari >**

Funzioni di una rete di TLC

Segnalazione >> scambio di informazioni che riguardano *l'apertura, il controllo e la chiusura* di connessioni e la gestione di una rete di telecomunicazione.

Commutazione >> individuazione delle risorse necessarie per collegare i due utenti, stabilendo un circuito. È il processo di interconnessione di

- Unità funzionali (risorse)
- Canali di trasmissione
- Circuiti di telecomunicazione

Trasmissione >> trasferimento di segnali da un punto > più punti

Gestione >> gestione del funzionamento del sistema; modifica / aggiunta di nuovi utenti o canali (breve o lungo termine, spostare utenti su un altro canale per riparare guasti ecc.)

Topologia reti TLC

Mezzo trasmissivo >> mezzo fisico in grado di trasportare segnali tra due o più punti

- **Capacità** > massima velocità trasmissiva (bit/s) >> dipende dalla tecnologia con cui è realizzato il mezzo trasmissivo

Canale >> singolo mezzo trasmissivo / concentrazione di mezzi trasmissivi

- **Banda >** quantità di dati (bit) per unità di tempo (bit/s)
- **Capacità >** capacità del mezzo trasmissivo a bit rate inferiore tra quelli che lo compongono (*trasmetto alla velocità massima del più lento – collo di bottiglia*)
- **Velocità**

Traffico

- ***Offerto >*** quantità di dati per unità di tempo (bit/s) che una sorgente *cerca di inviare in rete*
- ***Smaltito (****throughput****)>*** porzione di traffico offerto (bit/s) che riesce ad essere consegnata correttamente alla destinazione

- ***Throughput <= capacità del canale && throughput <= traffic offerto***

Il traffico ricevuto non può essere maggiore della capacità del canale, né maggiore di quello offerto.

Rete di TLC

Insieme di nodi e segmenti che fornisce un collegamento tra due o più punti per permettere la telecomunicazione tra essi

- **Nodo >** punto in cui avviene la trasmissione
- **Segmento >** mezzo di trasmissione / canale

Tipologie di canale

- *Punto-punto* > due nodi collegati agli estremi del canale utilizzato in modo paritetico (usato da entrambi a stesso modo)
- *Multipunto* > più nodi *slave* collegati ad un nodo *master*

- *Broadcast* > unico canale condiviso da tutti i nodi, nodo trasmette in contemporanea a tutti gli altri qualsiasi informazione (se però un dato inviato contiene un indirizzo di destinazione, in quel caso avrò un punto-punto)

Aspetto topologico di una rete

Ricorda: lo stato critico di una rete si ha quando, da un nodo, non è possibile raggiungere gli altri.

Rete vista come grafo G=(V,A) con V vertici ed A archi. **C** = |A| **N** = |V|

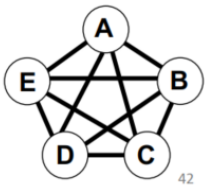
Gli archi rappresentano i canali, che possono essere

- Diretti > orientati
- Non diretti > non orientati, bidirezionali

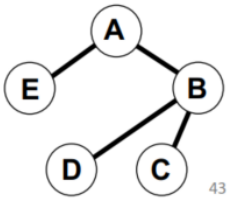
Topologia:

- **Fisica >** tiene conto del percorso dei mezzi trasmissivi (struttura fisica del collegamento)
- **Logica >** definisce l'interconnessione tra nodi mediante canali (definisce come i nodi comunicano effettivamente tra loro e i percorsi che effettueranno)

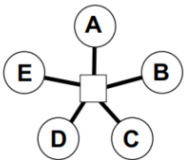
1) **Maglia completa**
   C = N(N-1)/2
   Alta tolleranza ai guasti (tutti nodi sono collegati agli altri) – ma troppi canali
   Solo usata con pochi nodi
   Cammini minimi semplici, canali diretti da nodo a nodo



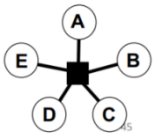
2) **Ad albero**
   C = N-1
   Molto vulnerabile ai guasti(tutti nodi collegati da un unico canale) – basso numero di canali
   Costi molto bassi, unico percorso tra i nodi
   Connettività minima



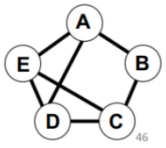
3) **Stella attiva**
   C = N (il centro non è un nodo)
   Molto vulnerabile ai guasti al centro – basso numero canali
   Usata in reti locali, satellitari, radio cellulari



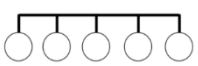
4) **Stella passiva**
   C = 1 (ci sono N fili)
   Molto vulnerabile ai guasti al centro stella – basso numero canali
   Costi ridotti, unico canale di trasmissione >> *broadcast*

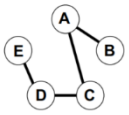


5) **A maglia**
   N-1 < C < N(N-1)/2
   Topologia non regolare, alta tolleranza ai guasti e numero di canali selezionabile a piacere
   Ci sono vari percorsi alternativi
   Più usata (internet, reti mobili…)



6) **Ad anello**
   C = N/2 se unidirezionale
   C = N se bidirezionale >> *se elimino un arco, diventa albero*
   Reti metropolitane / locali per costruire topologie magliate
   Bidirezionale resistente ai guasti, unidirezionale entra in stato critico
   Bidirezionale è la forma più semplice di topologia per l'instradamento in caso di guasto
7) **Bus**
   C = N-1 bus *attivo (caso particolare di albero)*
   C = 1 bus *passive (come stella passiva ma non c'è nodo centrale)*





Servizi nelle reti di telecomunicazioni

Servizio di telecomunicazione >> ciò che viene offerto da un gestore pubblico / privato ai propri clienti al fine di soddisfare una specifica esigenza di telecomunicazione

Possono essere

- Dedicate > unico servizio
- Integrate > offrono più servizi, es. internet

Servizi portanti > sono i servizi fondamentali -> trasmissione di segnali tra interfacce utente – rete (*collegamento ASDL, collegamento radiomobile, rete di trasporto*)

Teleservizi > telefonia, streaming, web browsing, videoconferenza

Come accedo ai servizi?

Modelli di accesso ai servizi

1) **Client-server >>** client interagisce con il server – il client è colui che richiede l'informazione, mentre il server è colui che resta in attesa di richieste del client per fornirgli informazioni. *Il client esiste solo in relazione alla comunicazione con il server (dopodiché sparisce)*
2) **Peer-to-peer >>** modello di comunicazione paritario, derivato dalla telefonia – tutti i nodi sono allo stesso livello. Sistema decentralizzato, ogni nodo mette a disposizione degli altri un'informazione condivisa. (ogni utente è sia client che server)

Trasmissione nelle reti di TLC

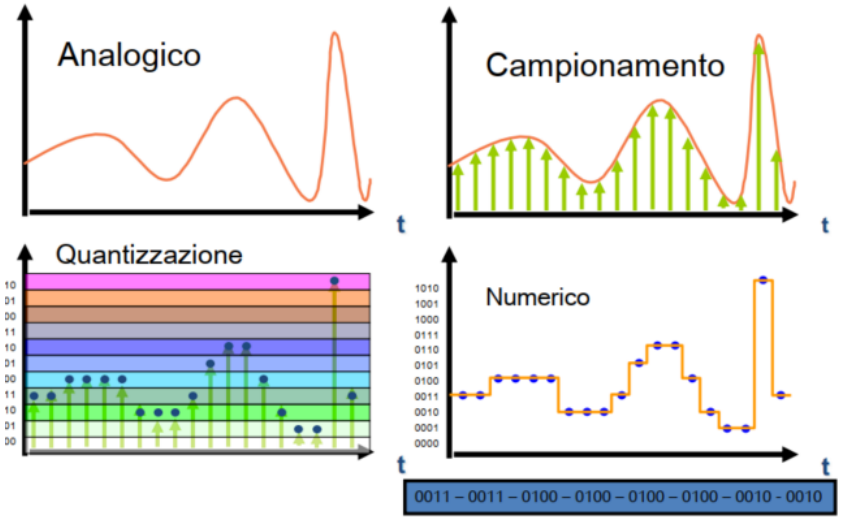
1) **Analogica** > informazione trasmessa per mezzo di un segnale elettrico *continuo, limitato, di infiniti possibili valori.*
   - Informazione assume valori di un insieme continuo – rappresentati come variazione continua di un parametro elettrico.

2) **Numerica** > informazione trasmessa per mezzo di un segnale elettrico *discontinuo, limitato, con un numero <u>finito</u> di possibili valori.*
    - L'informazione assume valori in un insieme numerabile e finito (es. bit di un file) >> ogni informazione discreta ha un suo segnale
    - Il ricevitore prende i segnali e ricostruisce l'informazione
    - ***I segnali che trasferiscono l'informazione sono continui (***questo per garantire che l'informazione venga ricevuta nell'ordine giusto)

Nelle reti telematiche, l'informazione è trasferita in forma digitale, usando però segnali sia *analogici* che *digitali*.

Questo perché una sorgente non sempre trasmetterà in forma digitale >> se trasmette però in forma analogica, si attuerà un meccanismo di **numerizzazione** **>> trasformazione da analogico a digitale**

Come? *Si campiona il segnale continuo elettrico, e poi lo si quantizza. Ad ogni fascia corrisponderà una serie di bit che rappresenteranno l'informazione.*



Tipologie di trasmissione

- *<u>Parallela ></u>* informazione trasmessa in parallelo (ad esempio su più canali alla volta) su bus di comunicazione
- *<u>Seriale ></u>* informazione viene prima *<u>serializzata</u>* e poi trasmessa, un bit alla volta.
    - In entrambi i casi si ha il problema di stabilire *quando* inizia la trasmissione.
- *<u>Asincrona >></u>* ogni byte di informazione è trasmesso *separatamente* dagli altri
    - il ricevitore deve conoscere il sincronismo del trasmettitore, bisogna perciò mantenerla ad ogni trasmissione
- ***<u>Sincrona >></u>*** informazioni da trasmettere sono strutturate in ***trame >>*** trasmettitore e ricevitore sincronizzano i loro clock prima della trasmissione e li mantengono sincronizzati durante tutta la durata della trama

Modi di trasferimento nelle reti di TLC

Condivisione di canale e condivisione di nodo

1) Condivisione di canale

Un canale non è tipicamente usato da un unico flusso di comunicazione.

Si parla di *multiplazione* se tutti i flussi sono disponibili in un unico punto. **(tipico in tipologia a stella, dove un punto comunica con tutti gli altri)**

Si parla di *commutazione* se i flussi accedono ad un canale da punti differenti. **(in questo caso non si ha la certezza di chi sta comunicando tramite quel canale)**

Entrambi i casi hanno l'obiettivo di condividere una risorsa – tramite un processo caratterizzato da una frequenza e da un tempo.

- Naturalmente la condivisione della risorsa dipenderà dalla *capacità* del canale >> se bassa capacità, lavoreremo sul cambiare la frequenza o il tempo.

Metodologie secondo cui è possibile svolgere la *condivisione di canale:*

1) *Multiplazione di frequenza (**FDM-FDMA**) >>* risorsa suddivisa usando bande di frequenza diverse – ogni banda di frequenza è "protetta" da una banda di guardia, per non *interferire* con le altre. **Trasmetto per tutto il tempo ad una velocità fissa *N***.
2) *Multiplazione di tempo (**TDM-TDMA**) >>* canale separato in slot tramite intervalli di tempo diverso. Le risorse sono quindi disponibili agli utenti per un limite di tempo ***N***, dopo il quale verrà messa a disposizione di un altro utente.
3) *Multiplazione di codice(**CDM-CDMA**)>>* nata recentemente, permette di elaborare più segnali contemporaneamente grazie alla elaborazione numerica del segnale. Permette di avere sullo stesso canale più flussi allo stesso tempo, ma con codifiche diverse >> ogni ricevitore sarà in grado di codificare solo la sua parte di quel flusso totale.
    - *Fatto attraverso l'utilizzo di codici riconoscibili*
    - *Alta resistenza ai disturbi*
    - *Trasmissione / ricezione non è altro che un prodotto tra* bit di informazione e codice
    - *non richiede necessariamente di essere accoppiata a multiplazione di tempo o di frequenza*
4) *Multiplazione di spazio >>* permette di riutilizzare frequenze che normalmente, dopo una determinata distanza, non persistono più (non sono più codificabili).
    - **Permette di far coesistere un flusso di informazione in punti diversi >> *usato per incrementare la capacità di una rete ed il suo reach***

La distribuzione delle risorse di un canale rispetto agli utenti può essere fatta in modo:

- **Predeterminato** (fisso) >> ogni utente ha una parte fissa del canale di trasmissione
- **Statistico** >> l'allocazione dei canali viene fatta in modo dinamica, *in base alle variazioni istantanee di traffico*

2) Condivisione di nodo

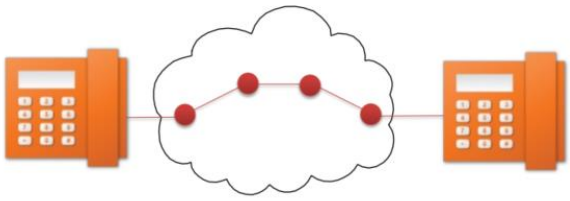
Condivisone di un nodo tra diversi flussi di informazione.

- Commutazione di *circuito* >> flussi di natura continua (telefonia)
- Commutazione di *pacchetto* >> flussi di natura intermittente (trasmissione dati)

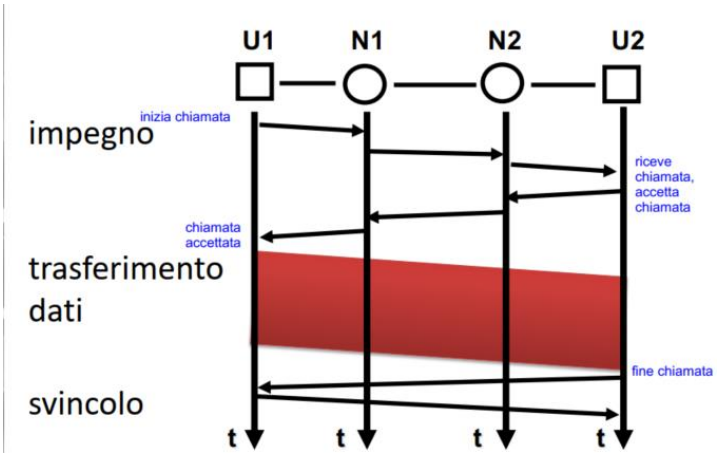
Commutazione >> Processo di interconnessione di unità funzionali, canali di trasmissioni o circuiti di telecomunicazione per il tempo necessario per il trasferimento di segnali.

- ***Processo di allocazione delle risorse di rete necessarie per il trasferimento dell'informazione***

Commutazione di circuito >> la rete usa le risorse disponibili per allocare un circuito a ogni richiesta di servizio – un circuito stabilisce un collegamento fisico tra due terminali di utente. Le risorse vengono rilasciate alla fine della comunicazione, che viene comunicata da uno degli utenti.



Si ha quindi la fase iniziale di segnalamento – una volta che la richiesta di trasmissione viene accettata dal ricevitore, può iniziare la trasmissione



Fasi di un sistema a commutazione di circuito

1) **Fase di impegno**
2) **Fase di trasferimento**
3) **Fase di svincolo**

Vantaggi => banda costante, ritardi costanti, circuito trasparente – bassi ritardi nell'attraversamento dei nodi

Svantaggi => risorse dedicate a una sola comunicazione ; c'è un tempo di apertura del circuito e nessuna conversione di formati, velocità, protocolli – tariffazione in base al tempo di esistenza del circuito – **non adatta a sorgenti intermittenti (il canale continua ad esistere anche se non trasmetto alcun dato)**

Commutazione di pacchetto

Nata per sorgenti intermittenti.

NO ACCESSO MULTIPLO

L'informazione da trasmettere è organizzata in unità dati (PDU) *Protocol Data Unit,* formata da

- **PCI > Protocol Control Information (***header, informazioni di controllo***)**
- **SDU > Service Data Unit (***actual data***)**

Un PDU non è nient'altro che una *trama / pacchetto / segmento / messaggio, formato quindi da una header (PCI) e dei dati (SDU).*

La funzione di commutazione non alloca agli interlocutori in uso esclusivo un sottoinsieme delle risorse di rete; tutte le risorse di rete sono sempre disponibili per la trasmissione dei pacchetti; i pacchetti attendono in apposite code che le risorse necessarie diventino disponibili

Store and Forward

Ogni PDU è consegnata al nodo, che lo *memorizza, lo elabora e stabilisce su che canale inoltrarlo.*

Lo mette quindi in coda per quel canale.

La memorizzazione ed elaborazione della PDU può essere assegnata sia ad un nodo, che ad una matrice di commutazione, che farà appoggio ad un Sistema di controllo per il corretto instradamento del pacchetto.

Occorre quindi prima disporre dell'intestazione, controllare se ci sono errori nella parte di controllo e dei dati (PCI e SDU), e determinare poi il *bit rate* di trasmissione richiesto dal ricevitore.

> memoria -> elaborazione + instradamento + check errori -> coda di uscita -> **Uscita**

Non è molto differente da una commutazione di circuito in sé >> si ha solo una parte aggiuntiva in cui viene memorizzato il pacchetto,

Attenzione! *Le code di entrata / uscita potrebbero essere impegnate in momenti diversi >> si creano le così dette code di uscita (*se arrivano altri pacchetti mentre coda di uscita è piena, verranno scartati)

- le code di uscita sono tra le maggiori cause di perdita di pacchetti e causa di errore.

Variabili da tenere in considerazione nella commutazione di pacchetto:

- **Ritardo/tempo di trasmissione e ricezione Tx**>> funzione della dimensione in bit del pacchetto e velocità di trasmissione del canale >> **dimensione pacchetto / velocità trasmissione canale**
    - ***c'è sempre ritardo di propagazione >>*** *proporzionale alla distanza tra stazione sorgente e nodo da raggiungere o primo nodo intermedio)*

- **Dimensione del pacchetto (in bit)**
    - **ATTENZIONE!!!!** La lunghezza di ogni pacchetto è stabilita in base alla possibilità o meno di effettuare il ***pipelining*** >> **trasmissione in contemporanea di più pacchetti**
    - Ad esempio, in una comunicazione s -> n -> d, mentre n->d trasmette un pacchetto, s->n starà già trasmettendo un altro pacchetto

Es. pacchetto di dimensione 1500byte, velocità 10 Mbit/s

➔ Tx = 1500*8 / 10^7 -> ...

- **Ritardo/tempo di elaborazione** >> funzione della velocità con cui si eseguono le procedure di instradamento e controlli sul pacchetto
- **Ritardo di accodamento** >> funzione del traffico generato dagli utenti

Nel calcolo della trasmissione di un pacchetto da una sorgente ad un nodo destinatario, ciò di cui teniamo conto maggiormente è il *tempo di trasmissione Tx – e quindi della velocità sui diversi canali e della dimensione dei pacchetti*

Più nodi ho, maggiore sarà il tempo di trasmissione >> questo perché l'operazione di store sarà dispendiosa in base alla dimensione di ogni pacchetto

Come stabilisco la lunghezza di un pacchetto?

3 fattori da tenere in conto:

Pipelining

Ritardo pacchettizzazione

Percentuale di informazione di controllo

Pipelining

Anche detto *parallelizzazione*, è il processo per cui una sorgente può trasmettere in contemporanea più pacchetti verso una destinazione.

Questo viene fatto ad esempio, dato un nodo intermedio, nel caso in cui inizio a mandare al nodo intermedio un secondo pacchetto, dopo che il primo è stato già ricevuto dal nodo intermedio e sta venendo trasmesso al nodo di destinazione da quest'ultimo.

Il pipelining è fondamentale per ridurre il tempo di trasmissione di un'informazione >> la trasmissione in parallelo permette infatti di trasmettere in contemporanea diversi pacchetti

- Ne segue che spezzettando il dato iniziale in tanti pacchetti, potremo ottenere un tempo totale di trasmissione ridotto rispetto al trasmettere il dato per intero.

esempio

Confrontare il tempo necessario complessivo per trasmettere un messaggio di 10.000 byte in un sistema

A->B->C->D, con una velocità di trasmissione da canale a canale di 80 Kb / S

- Per intero
- 10 pacchetti da 1000 byte ciascuno

---

1) Nel caso del messaggio per intero, Tx sarà == 10.000 * 8 (dimensione totale in bit) / 80 * 10^3 (velocità) → **1s**
   - **Il tempo totale di trasmissione sarà 3s**, contando che si ha la trasmissione da A->B, poi B->C, C->D
2) Nel caso di una trasmissione in 10 pacchetti, la trasmissione di un singolo pacchetto sarà ->
   - **T0x** == 1000 * 8 / 80*10^3 -> **0.1s**; quindi ogni **0.1s** manderò un messaggio (una volta che il primo è andato da A->B, allora il secondo partirà da A->B)
   - ne segue che il tempo totale per un pacchetto ad arrivare a D sarà **0.1s * 3 -> 0.3s**
   - ne segue che tutti i pacchetti saranno partiti dopo **1s** -> uno ogni 0.1s; più **0.1s** perchè l'ultimo vada da B->C, e **0.1s** perchè vada da C->D
   - **il tempo finale sarà quindi 1s + 0.1s + 0.1s -> 1.2s**

Ritardo pacchettizzazione

Pacchetti brevi riducono naturalmente la durata della pacchettizzazione >> questo perché pacchetti più grandi impiegheranno più tempo ad essere riempiti (nei casi dei pacchetti audio ad esempio è più conveniente fare più pacchetti di dimensioni ridotte, così da mantenere una costanza nella trasmissione dei dati) -> *si favorisce così il pipelining*

La % di errore di un pacchetto aumenta con l'aumentare della sua dimensione >>>> pacchetti corti riducono la probabilità di errore

Dati pacchetti di **n bit** e la probabilità di errore su ogni bit **p**, la probabilità che un pacchetto sia corretto è:

- **(1-p)^n**
- Maggiore è la dimensione del pacchetto, maggiore è la probabilità di errore.

Percentuale di informazione di controllo

Laddove la riduzione della dimensione di un pacchetto e quindi l'aumento del numero di questi favorisca il pipelining e quindi la riduzione del tempo complessivo di trasmissione, *più pacchetti vuol dire più informazioni di controllo ripetute.*

Pacchetti troppo piccoli comportano un uso eccessivo della memoria per quanto riguarda la PCI (header)

Quantità di informazioni di controllo *header* = ***i / i + s***

- Dove i = dimensione PCI e s = dimensione SDU

Riepilogo – commutazione di pacchetto

Vantaggi

- *Utilizzo efficiente delle risorse, anche con traffico intermittente (al contrario di commutazione di circuito che è utilizzo costante di risorse)*
- *Possibilità di controllo di correttezza*
- *Conversioni di velocità, formati, protocolli*
- *Tariffazione in funzione del traffico trasmesso*

Svantaggi

- Ritardo di trasferimento *variabile* (si basa su velocità trasmissione, capacità, distanza, dimensione pacchetto)
- I pacchetti vengono elaborati in ogni nodo

Modalità di trasferimento in una rete a commutazione di pacchetto

Datagram e circuito virtuale

Datagram metodologia di trasmissione che non utilizza né segnalazione (rete non sa che c'è trasmissione in corso) né divisione in fasi.

Circuito virtuale metodologia di trasmissione che riproduce nella commutazione di pacchetto alcune delle funzionalità di quella a circuito – *come, ad esempio, al suddivisione in fasi:*

- ***Frame relay** usa circuito virtuale e non datagram (come invece fa IP). Entrambi utilizzano pacchetti di dimensione variabile.* >> **frame relay è orientato alla connessione, IP no.**
- *Apertura connessione >>> segnalazione*
- *Trasmissione dati*
- *Chiusura connessione >>> segnalazione*

La gestione del percorso dei pacchetti è gestita dalla **commutazione di etichetta:**

- Ogni pacchetto non avrà più informazioni di instradamento, poiché nodi / router assegnano alla sorgente e alla destinazione un'etichetta particolare per quella trasmissione – così ogni pacchetto può solamente finire in destinazione che ha quella particolare etichetta.
    - *Questo viene fatto nel momento in cui un nodo T1 vuole comunicare con un nodo T2 – vengono assegnate due etichette univoche ai due nodi, e ogni nodo che parte dal nodo T1 con quella particolare etichetta potrà SOLO andare nel nodo T2 che avrà quell'altra etichetta.*
- **Pacchetti con stessa sorgente e destinazione seguono lo stesso percorso.**
- **Risorse allocate dinamicamente.**
- **Il numero di etichette utilizzate è proporzionale al numero di connessioni attive**
- **Router hanno un solo indirizzo ip**

Perché usare il circuito virtuale rispetto al datagram?

Circuito virtuale non usa instradamento nei pacchetti. L'instradamento viene effettuato solamente all'inizio della trasmissione (fase di segnalazione)

- *Le etichette hanno dimensioni minori rispetto a informazioni di controllo quali IPv4 o MAC address.*

Il circuito virtuale mantiene inoltre la sequenza originale dei pacchetti.

Molto utile per collegamenti *LAN / flussi multimediali voce e video.*

Datagram *utilizza infatti degli identificativi locali >> ogni pacchetto contiene informazioni che identificano la destinazione a livello globale.*

- *Domanda >>* La complessità delle procedure di instradamento utilizzate, all'interno di un nodo, per inoltrare informazioni ha maggior influenza sulle prestazioni di una rete nel caso di datagram (no circuiti virtuali): occorre identificare in ogni pacchetto la coppia sorgente/destinazione cosa che comporta una maggiore complessità delle prestazioni di un nodo di rete. Nel caso di circuiti virtuali è sufficiente invece identificare il circuito virtuale. La coppia sorgente/destinazione è utilizzata solamente in fase di apertura del circuito. Vedi lucidi "Informazione di indirizzamento".
- **Comporta un'assegnazione delle risorse trasmissive per la sola durata della trasmissione del pacchetto**

I circuiti virtuali *utilizzano invece degli identificativi locali >> lo stesso nodo può usare lo stesso identificativo per comunicare con altri terminali, senza creare situazioni di errore.*

Tecniche di segnalazione

Scambio di informazioni riguardo all'apertura e il controllo di connessioni…e la gestione di una rete di telecomunicazione

- ***Di utente >>*** scambio informazioni utente-nodo
- ***Internodale (di rete) >>*** scambio informazioni nodo-nodo
- ***Associata al canale >>*** quando stesso canale viene utilizzato per segnalazione e scambio di informazioni – si dice che c'è una relazione biunivoca tra canale *controllante (informazioni di segnalazione)* e canale *controllato (informazioni di utente*) -> se esistono **k** utenti, ci sono **k** canali di segnalazione.

    - ***in banda >*** canale controllante e controllato coincidono, ma usati in tempo diverso. (chiamate telefoniche, prima canale viene usato per stabilire connessione, e poi per trasmettere dati. Avviene lo stesso per chiamate GET http > http viene prima usato per trasmettere gli header)
    - ***fuori banda >*** canale controllante e controllato sono distinti > **k** informazioni, **k** canali controllanti. Es. ftp
- ***A canale comune >>*** dati **k** utenti, esiste un solo canale su cui viaggiano tutte le informazioni di controllo degli utenti. (reti tecnologie avanzate, SS 7, reti telefoniche); **opera in modalità pacchetto**
    - ***Particolare rilevanza agli indirizzi degli utenti (*** *indirizzi ed enumerazione***)**

Tecniche di gestione

Network management

Diverse funzioni di gestione >>

- *Gestione configuration*
- *Prestazioni performance*
- *Guasti fault*
- *Sicurezza internal & external security*
- *Tariffazione accounting*

Qualità del servizio delle reti di TLC

Il progetto di una rete deve definire come le informazioni sono emesse dalle sorgenti

- *Problema di **progetto** =>* stabilire le risorse necessarie per raggiungere un obiettivo e una qualità mirata
- *Problema di **analisi** =>* stabilire la qualità del servizio dato un sistema già esistente.

Analisi effettuata tramite modelli *quantitativi*, per stimare la *qualità del servizio.*

- Si inizia partendo da simulazioni su modelli matematici, per poi passare a simulazioni con più variabili, e poi si passa all'implementazione.
    - Modelli matematici per caratterizzare richieste del servizio (frequenza / volume), gestire grandi flussi ecc.
    - Descrivere interazione attività – risorse

Sorgenti di informazione

- **Analogiche >** voce, video; *sono caratterizzate dalle loro caratteristiche spettrali (occupazione di banda, correlazione)*
- **Numeriche >** dati, voce numerizzata, video numerizzato; *caratterizzate dalla velocità di cifra, burstiness*

- **A velocità costante CBR (*constant bit-rate*) >>** *sorgenti impulsive, ho fasi in cui la velocità è massima e altre in cui è media – minima. Si cerca di mirare ad una velocità costante massima-media*
    - **Molto usato in videoconferenze**
    - *Velocità = bit/s; Durata = s;*
    - *Bit rate può cambiare in base al processo di generazione delle chiamate > frequenza con cui si presentano gli utenti (aumento / diminuisco bitrate in base a quanti utenti mi aspetto in quel momento)*
- **A velocità variabile VBR (*variable bit-rate*) >>** *video streaming, file transfer*

Process Key indicators – indici di qualità

Diversi tipi di informazioni richiedono alla rete prestazioni diverse – la qualità di un servizio è descritta utilizzando i seguenti parametri

- *Ritardo (valore medio / percentile >> qual è la probabilità che un ritardo superi una fascia specifica)*
    - ***Jitter >*** *variazione intorno al valore medio del ritardo*
- *Velocità*
- *% di errore*
- *% di perdita*
- *% di blocco*

Process key indicators – cbr, vbr – *alcuni esempi*

1) Telefonia -> cbr (constant bit rate) > trasmissione su ordine dei **Kbit/s**
- Ritardo Massimo => qualche 1/10 di s
- Velocità max => 64 kb/s
- % di errore => qualche unità
- Probabilità di blocco bassa

2) Posta elettronica -> vbr (variable bit rate)
- Ritardo massimo => diversi minuti
- Velocità bassa
- % di errore / blocco trascurabile

3) Video streaming -> vbr (variable bit rate)
- Ritardo Massimo => qualche secondo
- Velocità => da 100 kb/s a qualche Mbit/s
- % di errore => non superiore a qualche unità
- % di blocco => molto bassa
    - Questo perché non è importante che il video si avvii subito, ma che venga mantenuto durante la riproduzione senza interruzioni.

Architetture e protocolli di rete

Comunicazione >> trasferimento di informazioni secondo convenzioni prestabilite

Comunicazione necessita di:

- *Cooperazione*
- *Regole che definiscono l'interazione tra due o più elementi di una rete >>* ***Protocolli di comunicazione***
- *un'architettura, definita dall'insieme dei protocolli e delle loro gerarchie >>* ***Architettura di rete***

Protocollo definito da > algoritmi, formati, temporizzazioni.

1) **Protocolli di comunicazione >>** insieme delle procedure adottate perché due elementi dello stesso livello comunichino tra loro. Un protocollo di comunicazione deve definire:
    - ***Tipologia =>*** request / response
    - ***Sintassi =>*** struttura dei messaggi
    - ***Semantica =>*** significato di campi di bit dentro ai messaggi
    - ***Temporizzazione =>*** sequenze temporali di comandi e risposte
    - Il protocollo di comunicazione non gestisce infatti *come vengono interpretate* le informazioni da un nodo all'altro – esso gestisce *solamente come viene effettuata al comunicazione tra le parti.*

2) **Architettura di rete >>** definisce invece:
    - Il processo di comunicazione
    - Le relazioni tra le diverse entità coinvolte (gerarchie)
    - Le funzioni necessarie alla comunicazione
    - Come vengono organizzate le funzioni tra loro (*ad esempio su strati diversi…)*

Le architetture di rete utilizzate sono del tipo *stratificate:*

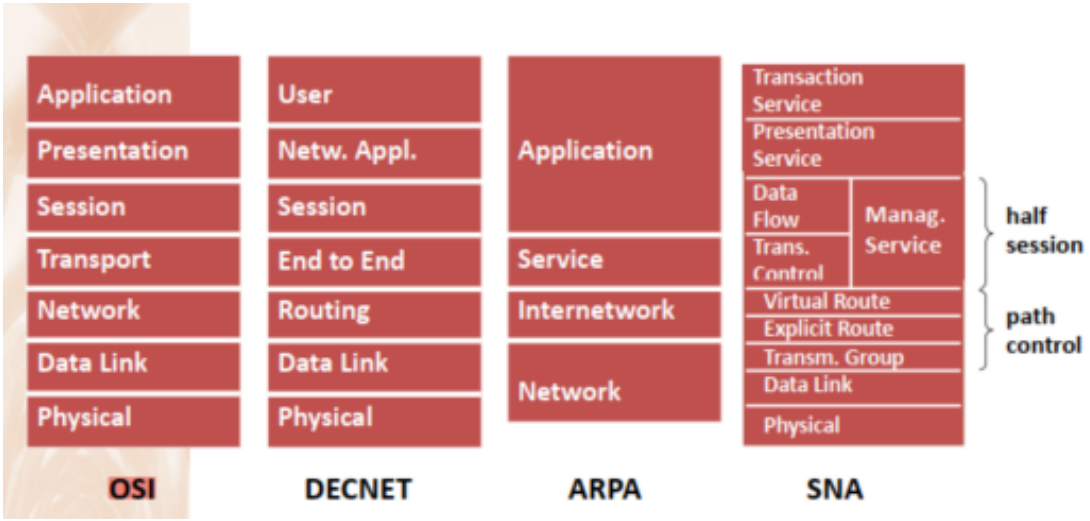
- Semplicità di progettazione => più facile la gestione e la standardizzazione
- Separazione delle diverse funzioni su strati diversi => ***in caso di problemi dovrò operare su un singolo strato, invece di cambiare l'intera architettura***

Architetture (*stratificate*)- Il modello OSI

Open System Connection

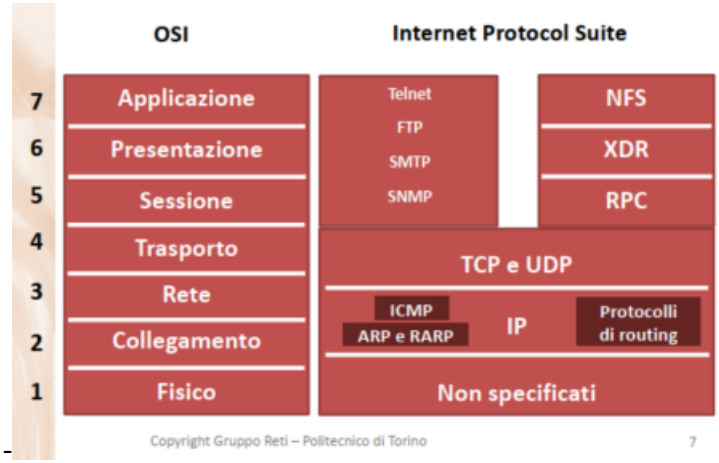
Il modello OSI è uno dei primi modelli a strato utilizzati, e viene tutt'ora usato come riferimento (ma non per tutte le architetture)

- Sono infatti presenti varie tipologie di architetture – alcune a 5, altre a 7 livelli…*cambia il modo in cui sono organizzate le diverse funzioni, ma le funzioni in sé non cambiano.*
- OSI presenta 7 livelli.



Applicazione - presentazione - sessione - trasporto - rete - data link - fisico

Un altro modello tipico delle architetture stratificate è quello di ISP > Internet Protocol Suite



Come è composta una rete?

Rete come insieme di ***sistemi*** (terminali,nodi…) collegati tra loro tramite mezzi trasmissivi.

- A comunicare non sono i terminali, ma ***processi applicativi***
  - I processi applicativi comunicano a loro volta con altri processi applicativi all'interno dello stesso terminale (in un certo senso, in modo verticale, verso il basso) fino a giungere al livello fisico > a questo punto, l'informazione verrà trasmessa ad un altro terminale.
  - La comunicazione avviene quindi, *verticalmente all'interno del sistema, e orizzontalmente tra diversi sistemi.*
  - Gli strati di un sistema sono detti ***entità*** *>> comunicano con altre entità dello stesso strato*
  - Ogni *entità* ha lo scopo di fornire funzioni allo stato superiore => *usando funzioni dello strato corrente e servizi di quello inferiore* (***in un certo senso ogni strato prende quelli dello strato inferiore e li migliora, per offrirli poi a quello superiore che farà la stessa cosa***)

Un entità non può raggiungere uno strato inferiore a quello corrente >> strato N fornisce servizi a strato N+1

- **Servizio connection-oriented (CO) >>** servizio che stabilisce un'apertura della connessione tra rete e interlocutori, un trasferimento e una fase di rilascio
    - **Es. *circuito virtuale > si occupa di instradamento***
- **servizio connectionless >>** dati vengono immessi in rete senza un accordo preliminare e sono trattati in modo indipendente

Comunicazione tra strati avviene tramite *service-access-points (SAP) => comunicazione one-to-one oppure one-to-many (strato N riceve informazioni da più strati N-1)*

- ***strati di diversi sistemi possono comunicare tra loro >> questo viene fatto tramite protocolli, e non servizi*** *(entità forniscono servizi – protocolli permettono di comunicare tra stessi strati ma di sistemi diversi)*
- *una **connessione** è infatti una comunicazione tra Service-Access-Points di sistemi diversi, per lo scambio di dati tra interfacce >> i socket sono dei service access points SAP – definiscono il punto infatti in cui sono inserite le informazioni*

RICORDA!!!! Protocolli comunicano in modo orizzontale tra stessi strati – trasmissione comunica in modo verticale tra diversi strati

N-service data unit (SDU)=> dati contenuti in uno strato N

N-service control information (PCI) => informazioni di controllo in uno strato N > vengono aggiunte tante intestazioni quanti sono gli strati superati (contrario in discesa)

Quando vengono mandati al livello superiore, SDU e PCI vengono chiusi in una PDU:

- *se SDU da contenere in PDU troppo grande, si può usare **segmentazione >>** operazione rischiosa, se un pezzo va perso il resto della PDU è inutilizzabile*
    - ***POSSO SEGMENTARE>>***
    - ***1 PDU >> X SDU***
    - ***CREARE X SDU (N-1,livello inferiore) da una PDU N***
    - Quindi da una PDU posso generare più SDU di livello inferiore, oppure tante PDU da una SDU di stesso livello
    - segmentazione applica gli stessi header a tutti i segmenti che genera
- in alternativa, si può utilizzare la **concatenazione**

7 Strati OSI

Come specificato prima, l'architettura stratificata OSI utilizza 7 livelli >> questo si applica ai terminali

I sistemi di trasporto che invece si trovano sul percorso svolto dalla comunicazione utilizzano 3 livelli >>

- *rete*
- *collegamento*

- *fisico*

1) ***Livello fisico***

Trasferimento bit – fornisce mezzi meccanici / fisici / funzionali per attivare e mantenere connessioni fisiche

Definizione di codifiche di linea / connettori / tensione

2) ***Livello collegamento / data – link layer***
   Fronteggia malfunzionamenti stato fisico – mezzi funzionali per trasferimento a livello rete
   Controllo di flusso
   Delimita unità dati (dove inizia e finisce trama)
   Rilevazione + recupero errori di trasmissione
3) ***Livello rete / network layer***
   IP
   Instradamento
   Controllo di flusso e congestione
   Tariffazione

Questi livelli sono presenti sia nei terminali, che nei sistemi di trasporto.
I livelli seguenti, e quindi superiori, non sono presenti nei sistemi di trasporto. (solo terminali)

4) ***Livello trasporto / transport layer***
   TCP/UDP
   Controllo di errore – sequenza – flusso
   Multiplazione – demultiplazione di connessione
   Segmentazione pacchetti – ricomposizione pacchetti
5) ***Livello sessione / session layer***
   Assicura una connessione di sessione al livello presentazione
   Maschera interruzioni del servizio trasporto
   Spesso integrato nelle funzioni di livello superiori
6) ***Livello presentazione / presentation layer***
   Presenta i dati in modo indipendente dal livello applicazione (traduce quello che viene detto anche se non so cosa dice…)
   Servizi di cifratura delle informazioni
7) ***Livello applicazione / application layer***
   Fornisce ai servizi applicativi i mezzi per accedere all'ambiente OSI (non è applicazione finale)
   Es. trasferimento file, posta elettronica

Protocolli a finestra

Trasferimento dati affidabile

Richiedono un handshake iniziale – e quindi sono connection oriented

Protocollo a finestra permette di avere

- Controllo e recupero *errore*
- Controllo del *flusso di trasmissione*
- Controllo della *sequenza* > anche se pacchetti arrivano in ordine sbagliato, destinatario deve essere in grado di ricostruire dato originale

Protezione dagli errori >> trasmissione mai esente da errori – si può *rimediare tramite codifica di canale >>* invece di trasferire n bit, ne trasferisco k + (n-k)

- **concetto simile a quello dei bit di parità >> maggiore ridondanza dei dati**
- **come intervengono questi bit di parità ?**
  - *rilevano e correggono errori senza ristrasmettere il pacchetto (**FEC, Forward error correction**)*
  - *rilevano l'errore e richiedono di ritrasmettere il pacchetto (**ARQ, Automatic Retransmission request***)
  - bit di parità vengono inseriti dentro la PCI (*informazioni di controllo*)

ARQ – *automatic retransmission request*

Permette di avere un controllo di errore, in aggiunta al normale controllo di flusso e di sequenza.

- Il controllo di sequenza viene fatto aggiungendo *bit di numerazione* all'interno della PCI *(PCI ARQ contengono quindi bit di parità, indirizzi e bit di numerazione)*
  - Una volta che il pacchetto viene ricevuto dal destinatario, quest'ultimo rimanda al trasmittente un pacchetto di ***acknowledgement,*** contenente nella PCI gli stessi elementi ma al posto del bit di numerazione, *il numero di sequenza atteso.*
- ARQ necessità di un rapporto esplicito tra trasmettitore e ricevitore -> *deve essere conclusa la fase di segnalazione*

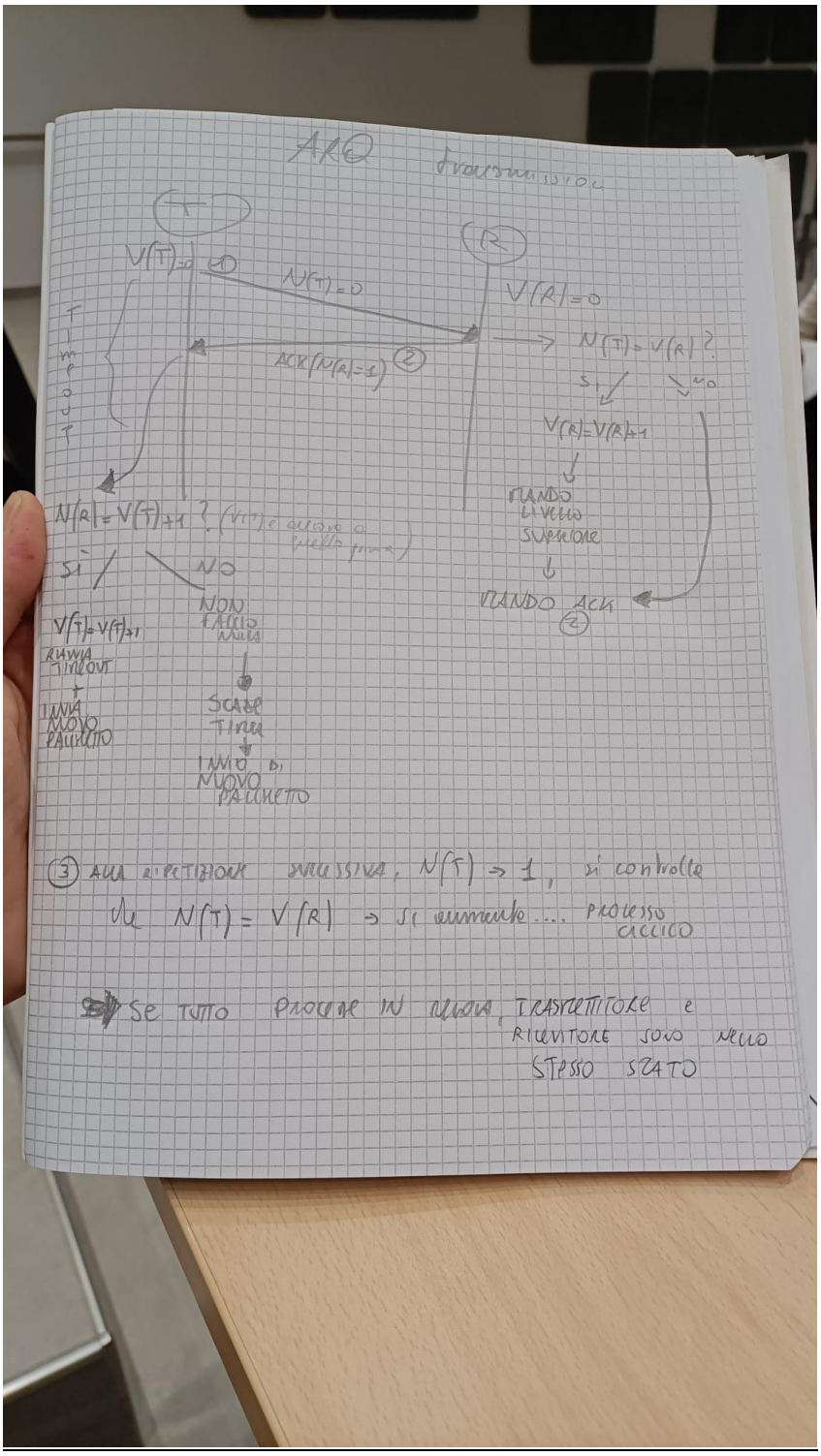
Tecniche ARQ

1) **Stop and Wait**
2) **Go back N**
3) **Selective Repeat**

1) Stop and wait – *alternating bit protocol*

Il trasmettitore > fa una copia della PDU e la invia al ricevente, facendo partire un timer di *timeout*, e rimane in attesa del pacchetto di ***acknowledgement*** – se il *timeout* scade prima dell'arrivo di quest'ultimo, il pacchetto viene ristrasmesso.

- Se l'ACK ricevuto corrisponde all'ultimo PDU inviato, allora procede con quella dopo. Sennò l'ACK viene scartato e viene ritrasmesso il pacchetto precedente.
- Si parla quindi di **variabili di stato >** il trasmettitore ha una variabile che tiene conto del suo *numero di sequenza attuale* > se il numero di sequenza dell'ACK ricevuto non corrisponde a quello attuale del trasmettitore, allora il pacchetto verrà ritrasmesso.



Spiegazione del funzionamento del processo di trasmissione e ricezione di un pacchetto

Inizialmente, trasmettitore e ricevitore fissano loro variabili di stato ad un valore uguale tra loro.

- Trasmettitore genera quindi una PDU con variabile di stato uguale a quella iniziale, e la trasmette, avviando il *timeout*.
- Il ricevitore che riceve la PDU aggiorna la sua variabile di stato (+1), e manda l'ACK.
- Il trasmettitore, quando riceve l'ACK, controlla se il suo *numero di sequenza* corrisponde alla variabile di stato *del trasmettitore +1* -> se vero, allora la trasmissione è avvenuta con successo, e trasmette il prossimo pacchetto, aggiornando la sua variabile di stato a +1, e resettando il *timeout*.
    - Sennò, resetta timeout e invia di nuovo pacchetto precedente, senza aggiornare la sua variabile di stato.

RTT >> *round trip time,* durata del processo di trasmissione e ricezione di un singolo pacchetto.

Alternating bit protocol >> utilizzo di un solo bit per rappresentare lo stato del trasmettitore e ricevitore (0,1) >> *entrambi a fine trasmissione saranno 0 oppure 1*

Si possono ridurre le possibilità di malfunzionamento usando un maggior numero di bit per la numerazione oppure un tempo di vita massimo per le PDU e gli ACK.

Stop and wait >> casi di errore

- Perdita pacchetto in ricezione (ricevitore) >> *scatta timeout,* trasmettitore rimanda pacchetto
- Perdita pacchetto ACK in ricezione (trasmettitore) >> *scatta timeout,* trasmettitore rimanda pacchetto > *ricevitore ha già ACK per quel pacchetto (variabile di stato è già == a quella del pacchetto ricevuto) >> rimanda ACK al trasmettitore*
- Timeout troppo breve >> ricevitore e trasmettitore sprecano risorse – ogni pacchetto rischia di essere mandato due volte se non di più – ad esempio, se mentre sta arrivando l'ACK del primo pacchetto, scatta il timeout, il trasmettitore rimanderà il primo pacchetto, e intanto riceve subito dopo l'ACK di quel pacchetto– ricevitore lo scarta perché ce l'ha già, e *rimanda* l'ack di quel pacchetto (che però trasmettitore ha ricevuto poco fa)
- **Spazio di enumerazione insufficiente >** può infatti capitare che un pacchetto arrivi molto tempo dopo quello previsto > *magari arriva in un momento in cui il suo valore di enumerazione ACK è uguale a quello dello stato del trasmettitore (molto probabile visto che i possibili valori sono solo 0 e 1) >>>>* il pacchetto viene così accettato anche se fuori sequenza, blocco dell'intero protocollo
    - **La soluzione è aumentare i bit di enumerazione >> per evitare che pacchetti fuori sequenza vengano comunque accettati**

Stop and wait >>>>> poco efficiente, deve tenere conto anche del tempo di propagazione

- ***Efficienza*** >>> Tp / (Tp+2D)
- Tempo trasmissione / (Tempo trasmissione + ritardo trasmissione + ritardo ricezione)

2) Go back N – *sliding window protocol*

SEMPRE Più EFFICIENTE DI STOP & WAIT

Stop and wait può essere poco efficiente >> devo aspettare la conferma di ogni pacchetto – *potrebbe impiegare troppo tempo per la trasmissione di informazioni divise in molti pacchetti*

Si definisce infatti la **finestra di trasmissione *Wt* >>** *la quantità massima di PDU in sequenza che il trasmettitore è autorizzato ad inviare in rete senza averne ricevuto riscontro (ACK) >> posso inviare più pacchetti alla volta*

- ***Finestra di ricezione Wr >>*** *quantità massima di PDU che il ricevitore è autorizzato ad accettare e memorizzare. (limitata dal buffer del ricevitore)*

- **Una finestra fa riferimento ad un *singolo processo di comunicazione*** >> possono coesistere più finestre, e quindi il buffer del ricevitore sarà diviso tante volte quante finestre e comunicazioni si hanno.
- **Il protocollo è detto "*sliding window*" perché la finestra procede in avanti man mano che riceve gli ACK di cui era in attesa >>>** in caso di *timeout*, rimanda tutti i pacchetti ancora in attesa di conferma.

Spiegazione del funzionamento del processo di trasmissione e ricezione di un pacchetto

- Trasmettitore invia N = Wt PDU (mantiene di ognuno una copia in caso dovesse ritrasmetterli)
- *Attiva un solo timeout, che viene resettato alla trasmissione di ogni PDU*
- Ricevitore si aspetta di ricevere pacchetti in sequenza > se corretto lo guarda, sennò lo scarta (e scatterà timeout per quel pacchetto particolare)
    - *Se timeout scatta, vengono rimandati tutti pacchetti **non ancora confermati** a partire dal pacchetto che ha scaturito timeout*

Ne segue che se Wt = N, allora Wr = 1, poiché il destinatario invierà sempre una sola conferma alla volta.

- *Finestra di ricezione Wr non può mai trovarsi a sx della finestra di trasmissione Wt*

- *Qualsiasi posizione assume Wr all'interno di Wt, tutti i pacchetti alla sua sx sono pacchetti da confermare.*
- *Tutti i pacchetti a sx di **Wt** sono pacchetti confermati.*

Tipologia ACK

In un'ambiente in cui si hanno diversi pacchetti inviati allo stesso momento, è importante definire la natura degli ACK.

- *Cumulativo* > *notifica la corretta ricezione di tutti i pacchetti con numero di sequenza inferiore a quello specificato (dà la conferma che ho ricevuto tutti i pacchetti prima quindi)*
- *Selettivo* > *notifica la corretta ricezione di un singolo pacchetto*
- *Negativo* > *notifica la richiesta di trasmissione di un singolo pacchetto (caso in cui quindi serve ritrasmetterlo poiché potrebbe non essere stato ricevuto, oppure ricevo pacchetti 5,6,8 -> mando quindi ACK negativo 7, mi manca pacchetto 7 e quindi devi ritrasmettermelo)*

PiggyBacking

Tecnica usata nei flussi di comunicazione *bidirezionali* >> in trasmissione bidirezionale da A <-> B, invece che usare pacchetti appositi per ACK, integro gli ACK dentro altri pacchetti che trasmetto ad una parte all'altra

- Pacchetto A -> B con informazione quindi potrà contenere un ACK di un pacchetto precedentemente inviato da B -> A

Numerazione PDU

Numerazione delle PDU avviene secondo una numerazione ciclica, usando come base *k bit di numerazione*. (se ho 3 bit, da 0 a 8)

Quindi in go back N >>

- Trasmettitore deve tenere conto di numerazione, maggiore utilizzo di memoria e gestione del *timeout*
- Numerazione delle PDU può avvenire tramite algoritmi
- Il ricevitore invece rimane invariato rispetto allo Stop & wait – Wr = 1
- **Finestra di trasmissione non può essere superiore ai bit di numerazione >>>> Wt < 2^k**
- **Quindi un protocollo *go back n* funzionerà solamente se userò come bit di numerazione un k tale che**
  - **K >= log2(Wt + 1)**
  - Wt <= 2^k-1 ; Wt+1 <= 2^k ; k>=log2(Wt+1)

3) Selective Repeat

Permette di avere finestra di ricezione maggiore di 1, garantendo maggiore efficienza

Wr > 1

Ne segue che i pacchetti possono essere ricevuti *anche fuori sequenza*

- Posso usare un *timeout* per ogni pacchetto
- Posso usare *ACK cumulativi / selettivi*

Il vantaggio sta infatti nel fatto che non ritrasmetterò, in caso di errore, pacchetti di cui ho già ricevuto la conferma – o che ho comunque memorizzato nel ricevitore ma non ancora confermato.

SIA NEL CASO DI SELECTIVE REPEAT, CHE GO-BACK-N, SE NON HO PACCHETTI IN SEQUENZA, POSSO AUMENTARE IL NUMERO DI BIT USATI PER LA NUMERAZIONE DELLE UNITA' DATI PER GARANTIRE MAGGIORE ROBUSTEZZA AL CANALE.

Spiegazione del funzionamento del processo di trasmissione e ricezione di un pacchetto

- Trasmettitore invia N PDU, e per ognuna avvia un *timeout* >> se scade *timeout per un pacchetto in particolare, ritrasmette quel pacchetto*
- Ricevitore riceve PDU >
  - Se in sequenza e corretta > accetta e manda ACK
  - Se corretta ma non in sequenza > la *memorizza*
- *In caso di errore >> trasmettitore ritrasmette tutti pacchetti che non sono memorizzati dal ricevitore*
- *Per ogni ACK ricevuto, la finestra avanza.*

Utilizzando la memorizzazione da parte del ricevitore, si riduce l'occupazione del canale, poiché il trasmettitore ritrasmette solo i pacchetti persi.

Numerazione PDU == 2^k

Wt+ Wr <= 2^k

- Non rispettare tale enumerazione potrebbe causare pacchetti erroneamente accettati due volte – pacchetti scartati erroneamente.

Efficienza e throughput

a = Tp / Ttx

- Efficienza di un protocollo Stop&Wait ideale senza errori

$$\eta = \frac{T_{TX}}{T_{TX} + 2t_p} = \frac{1}{1+2a} < 1$$

- Efficienza di un protocollo Go-Back-N ideale senza errori

in questo caso ho una finestra che trasmette di continuo (tempo trasmissione > ritardo)

Wt = tempo impiegato per la trasmissione

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

- Throughput su canale a capacità C

$$\theta = \eta C$$

A parità di condizioni – ovvero a parità di *dimensione finestra* e *volume di dati da trasmettere*

- Se RTT grande, efficienza e throughput minore

Come miglioro l'efficienza e quindi il throughput?

- Miglioro il RTT > minore distanza
- Utilizzo connessioni breve >> maggiore *throughput*
- ***Maggiore throughput >> maggiore quantità di dati e velocità con cui escono, quindi più velocemente la finestra scorre >>***
- **Il controllo della finestra corrisponde quindi alla capacità di controllare il throughput (***e quindi regolare la velocità di trasmissione delle sorgenti – tenendo sempre a mente che spesso non posso cambiare il RTT o la distanza* **)**
  - *Se non posso infatti cambiare l'RTT / distanza, posso cambiare la finestra di trasmissione* ***>> TCP fa questo tramite il controllo di congestione, cambiando la finestra di trasmissione in modo dinamico***

Strato fisico

Mezzi trasmissivi – reti di accesso e trasporto

Mezzi trasmissivi:

1) Elettrici > *doppino non schermato / cavo coassiale*
2) Ottici > *fibra ottica*
3) Radio

Mezzo di trasmissione deve essere **ottimale >**

- Resistenza, capacità parassite e impedenza basse
- Resistenza a trazione
- Flessibilità
- Facilità di collegamento dei ricetrasmettitori

1) Mezzi elettrici

Sono mezzi di trasmissione ottimali – presentano tutte le caratteristiche annunciate.
Le loro caratteristiche dipendono strettamente dalla loro *geometria, numero di conduttori, distanza reciproca, tipo di isolante, tipo di schermatura.*

Parametri >>>

- **Impedenza >** facilità con cui corrente scorre sul mezzo
- **Velocità di propagazione segnale >** 0.5/0.7 c 0.6c per fibre ottiche
- **Attenuazione >** cresce in base a distanza
- **Diafonia / cross talk >** disturbo dovuto a vicinanza trasmettitore – ricevitore (diminuisce con aumentare distanza)

Il doppino (*twisted pair – coppia*)

Mezzo trasmissione tipico della telefonia – ridurre interferenze elettromagnetiche mantenendo differenza di potenziale costante.

Costi ridotti e facile installazione – due fili di tensioni opposte (*non serve riferimento comune, terra – si ha perciò un disturbo costante e non variabile*)

- Unshielded twisted pair >> senza schermatura, reti telefoniche e dati fino a 100 Mb/s e 1Gb/s
  - In un cavo UTP (Unshielded Twisted Pair) la trasmissione del segnale avviene sfruttando la differenza tra le tensioni sui componenti di una coppia di conduttori
- Folied / shielded twisted pair >> con schermatura, da reti 10 Gb/s fino a 40 Gb/s

Il cavo coassiale

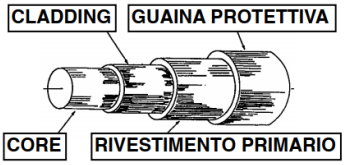
È un cavo schermato, riduce quindi i disturbi esterni

Connettore centrale avvolto da calze di schero – costi elevati, più difficile da installare – maggiore velocità di trasmissione (centinaia di mb/s)

2) La fibra ottica

Filo di vetro costruito da due parti > *core* e *cladding* – *hanno indice di rifrazione opposto:* la fibra ottica si basa infatti sulla propagazione di un raggio luminoso, riflettendo il raggio luminoso e propagandolo lungo il materiale all'interno del *core*.

- **Molto utilizzata per collegamenti intercontinentali / cavi sottomarini**



>>Immune da disturbi elettromagnetici, alta capacità trasmissiva, bassa attenuazione – dimensioni ridotte e costi contenuti

<< Non adatti a collegamenti *broadcast* – difficili da collegare tra loro, ridotta curvatura, soffre vibrazioni

3) Canali Radio – *"etere"*

Tramite l'uso di antenne, supporta la diffusione da parte di un singolo trasmettitore verso più ricevitori.

- Si parla di Canale *Radiomobile* se il trasmettitore è in movimento.
- La qualità della trasmissione dipende dalla potenza ricevuta e quella trasmessa.

Potenza ricevuta (ipotetica)

Aumenta all'aumentare della frequenza – diminuisce all'aumentare della distanza

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

Lunghezza d'onda $\propto$ 1/freq.

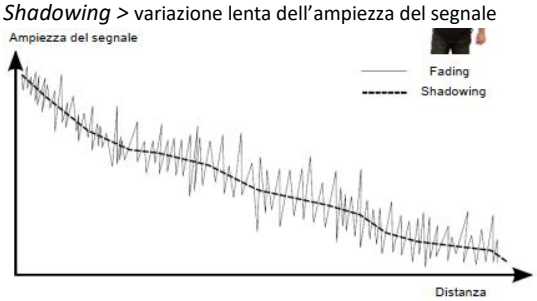
Distanza TX/RX

Eq. di Friis: propagazione nello spazio libero

Potenza ricevuta (*<u>reale</u>*)

In realtà, potenza dipende anche da altri fenomeni >> *atmosferici, interferenza di altre trasmissioni su stesso canale, ostacoli fissi / in movimento* (causano riflessioni, copie del segnale…)

- *Fading* > variazione *<u>veloce</u>* del segnale dovuta a ricezione di diverse copie, dovuta da ostacoli in movimento
- *Shadowing* > variazione lenta dell'ampiezza del segnale
-



La trasmissione sul mezzo fisico

La trasmissione di segnali e informazioni è sempre analogica > la trasmissione avviene infatti tramite assegnazione di bit / simboli di informazione a segnali diversi > *sta poi al ricevitore decidere quale significato assegnare ai diversi simboli,* tramite un processo di **hard decision** *(spesso causa di errori)*

In base al tipo di mezzo fisico utilizzato, ci sono due tecniche diverse di interpretazione dei segnali:

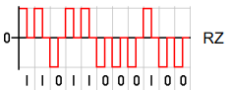
- ***Codifiche di linea >*** *mezzi elettrici ed ottici a bassa frequenza*
- ***Modulazioni digitali >*** *mezzi radio, ottici ed elettrici con modulazioni numeriche e ad alta frequenza*

1) *Codifiche di linea*

Le codifiche di linea sono usate per rappresentare informazioni numeriche (digitali) su mezzi elettrici ed ottici a bassa frequenza.
Seguono le tipologie di codifiche di linea, con i relativi problemi

- ***Unipolare >*** uso un bit di informazione per rappresentare un'informazione > tensione nulla 0, tensione positiva 1
  - *Necessario sincronismo tra ricevitore e trasmettitore, che sono però tra loro lontani*
  - *Spreco di energia >* valore medio è V/2 invece che essere nullo
  - *Disturbo per il ricevitore / sovraccarico (fasi di 1 consecutivi sono troppo pesanti per mezzi ottici)*
- ***Polari >*** alternano fasi di tensione positiva 1 e fasi di tensione negativa -1
  - ***NRZ >*** non return to zero > il livello del segnale viene mantenuto per tutta la trasmissione – risolve problema del valore medio che è ora nullo

  - **RZ >** return to zero > mantengo valore medio nullo + non mantengo la tensione per ogni simbolo – opero però su una frequenza doppia del necessario > richiede bit rate elevato



  - **Bifase – codifica Manchester >** risolve problema sincronismo – 0 e 1 sono rappresentati come cambiamenti di un unico livello – ho però problema che utilizzo sempre il doppio della frequenza necessaria > **richiede bit rate elevato**
- ***Bipolari (alternate mark inversion) >*** unione Unipolare e Polari > ho tensione nulla 0 e tensione negativa -1 e positiva 1
- ***nBmB >*** rappresento n bit usando m simboli – molto usato
  - richiedono meno banda di codifiche polari
  - posso scegliere le parole *codice*, limitando così sovraccarichi dovuti da troppi 0 / 1 consecutivi (limita la componente continua)
  - fornisce caratteri speciali per delimitazione pacchetti, "*padding pacchetti*"

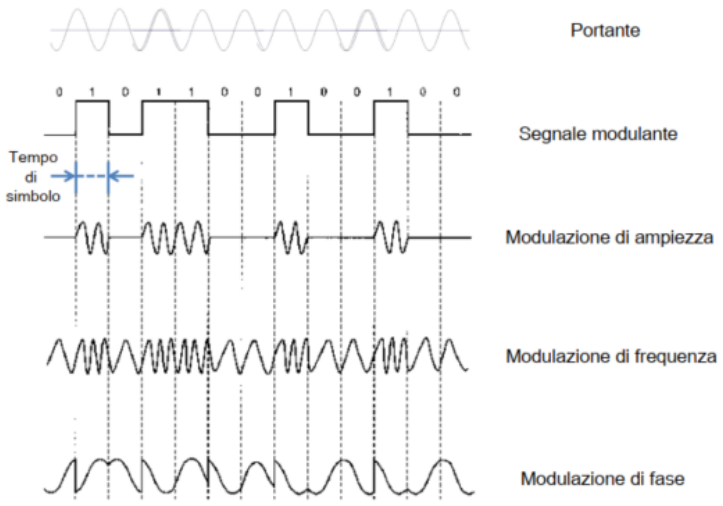
La codifica nBmB viene usata anche insieme ad altri tipi di codifiche, come quella *Polare NRZ* per evitare di raddoppiare la frequenza e mantenere il sincronismo

2) Modulazioni digitali

Sono tecniche per rappresentare informazioni numeriche tramite segnali *analogici* su mezzi *radio, ottici ed elettrici*.

L'informazione è rappresentata su un **segnale sinusoidale portante** *variandone la frequenza, ampiezza, fase o combinazioni di queste.*

- **ASK >** *amplitude shift keying*
- **FSK >** *frequence shift keying*
- **PSK >** *phase shift keying*
- **QAM >** *quadrature amplitude modulation* > *combinazione delle precedenti >>>* ***preferita per trasmissioni radio***



Ogni sigla è preceduta dal numero di simboli che va a rappresentare quel segnale modulante

Es. 8-ASK modulazione in ampiezza che codifica 8 segnali tramite l'utilizzo di 3 bit (2^3)

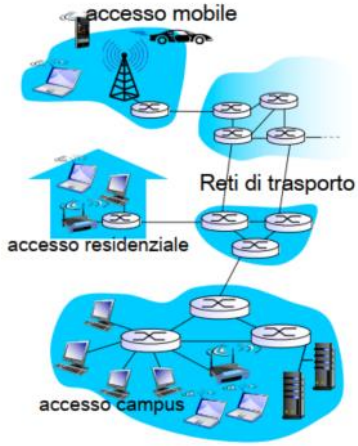
Un maggior numero di segnali rappresentabili e quindi maggior numero di bit è utilizzabile se non sono presenti eccessivi disturbi sul canale. (se ho un maggior bit rate e i miei dati vanno persi, il segnale risulterà ancora di più disturbato e incomprensibile)

Nella trasmissione dei dati su un canale radio, è preferibile l'utilizzo una modulazione numerica come il 64-QAM per poter trasmettere nella banda di frequenza opportuna

Reti di accesso e di Trasporto

Utente -> Rete

Utenti si collega a rete sfruttando *rete di accesso e rete di trasporto.*



le reti di casa, telefono e università sono reti di accesso - la comunicazione tra reti di accesso è una rete di trasporto.

1) Rete di **accesso** => *sta tra utente e prima rete di telecomunicazione* – collegano l'utente con l'access point (*fornito da un fornitore di TLC); sono*
   - Residenziali / mobili / istituzionali
2) Rete di **trasporto** => *apparati e mezzi trasmissivi appartenenti a gestori di un servizio TLC* – che hanno come scopo la comunicazione tra due nodi di accesso.
   - Metropolitano / nazionale / sovranazionale

Reti di accesso

Le reti di accesso permettono di arrivare all'ultimo miglio – *l'utenza residenziale, l'utente ultimo.* Sono dette *local loop => reti d'accesso.*

- ***xDSL >*** *Digital Subscriber Loop*
  - alta velocità sulla rete di accesso – downstream > upstream – applicazioni client<>server
  -

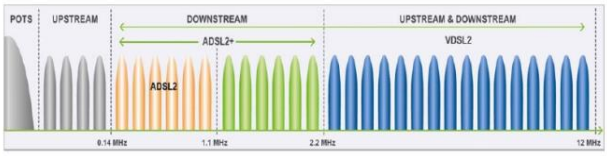
	ADSL	ADSL2	ADSL 2+
Downstream	6 Mb/s	8 Mb/s	24 Mb/s
Upstream	1.5 Mb/s	3.5 Mb/s	3.5 Mb/s

  - L'accesso viene effettuato tramite *Modem >>* fa da *modulatore* e *demodulatore* >> traduce il segnale idoneo alla comunicazione su rete pubblica in segnale analogico, comunicabile sulla rete telefonica. (***trasformano il segnale da analogico a digitale, e viceversa)***
  - Nel caso i dati inviati da un modem all'esterno, su linea telefonica, siano ad alta e bassa frequenza – quindi dati mischiati con *flussi vocali*, questi passeranno da uno *splitter >*lo splitter dividerà i dati ad alta frequenza da quelli a bassa -> quelli a bassa

frequenza, ovvero i dati in sé, verranno mandati ad un DSL Access Multiplexer, che ha lo stesso compito di un normale *Modem*, che invierà a sua volta i dati all'ISP. (per effettuare la funzione di *trasporto* tra nodi diversi)

- **Lo splitter**, quindi, ha lo scopo di dividere i dati dai flussi vocali – quindi dati a bassa frequenza da quelli ad alta.
- **Il DSLAM** ha lo scopo invece, come un modem, di unificare diversi flussi dati in un unico flusso, per comunicarlo all'esterno – o viceversa.
- **La divisione dei dati da flussi vocali avviene tramite multiplazione di frequenza**

- **VDSL >** *altra tipologia di rete d'accesso, sempre sotto la famiglia DSL >>* permette di avere un bit rate maggiore rsipetto ad un normale DSL – con la possibilità di scegliere quanta banda allocare ad ogni singolo utente, sia in downstream che upstream.
  - *Vdsl lavora infatti su una banda di frequenza maggiore, con migliori standard di trasmissione e modulazione.*



Ogni pila è in realtà un sotto-canale / micro-canale, per ognuno dei quali viene fatta una valutazione e stabilito quanti bit possono essere trasmessi – quando trasmetto quindi 100bit/s in realtà li trasmetto in modo parallelo e suddiviso su diversi micro-canali ->> ***sulle frequenze in cui ho meno disturbo, manderò più bit, mantenendo il bitrate più veloce possibile e sfruttando i canali migliori alla trasmissione in quel momento***

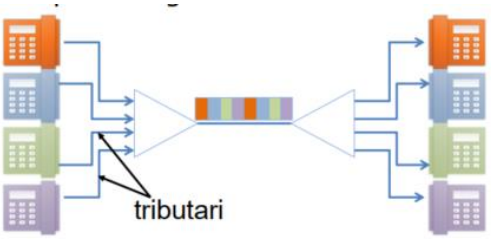
- Le reti di accesso VDSL **hanno un funzionamento ottimale al di sotto dei 200m. A crescere di distanza, aumentano attenuazione e rumore.**

- ***PON >*** *Reti di accesso Ottiche*
  - *Passive optical networks* > tecnologie di ultima generazione – non richiedono elettricità, funzionano *tramite il partizionamento del segnale luminoso.*
  - Come accennato sui mezzi di trasmissione, sono mezzi a *basso costo – arrivano "direttamente in casa"*
  - *Accesso multiplo a divisione di tempo >* condivise tra più utenti, che possono utilizzare allo stesso momento sia upstream che downstream
  - Funzionano come una specie di albero che ad ogni ramo dimezza la sua potenza e trasmette in broadcast a tutte le sue foglie.
  - Ogni canale è usato in modo unidirezionale (sia quindi in upstream che downstream)
  - Sono composti da diversi componenti >>
    - **OLT – Optical Line Terminator** -> parte iniziale della rete, in centrale
    - **ONU – Optical Networks Unit** -> cabine in strada
    - **ONT – Optical Networks Terminals ->** case utenti
  - Due tipologie >> **GPON – EPON**
    - **GPON >** 2.5 Gb/s downstream – 1 Gb/s upstream – supportano *diverse tipologie di trame*
    - **EPON >** 1 Gb/s simmetrico > trame ethernet >> **velocità molto alta, sarebbe da abbassare estremamente la distanza dei cavi >> non si può, si aumenta quindi dimensione media dei pacchetti**
- ***Reti mobili >*** *Accesso Broadband Mobile*
  - ***Reti cellulari a banda larga >>>>****Offrono connettività voce – dati ad utenti residenziali / in mobilità > Reti cellulari, Reti satellitari, FWA (Fixed Wired Access)*
  - Il funzionamento si basa su una copertura del territorio, garantita da aree diverse tra loro sovrapposte:
    - Ogni area è rappresentata da un'antenna a portata limitata – minore è la sua portata, maggiore è il suo bit rate. (se decido di coinvolgere più utenti, avrò una distanza maggiore, ed un bit rate minore).
    - Supportano la mobilità degli utenti >> in base all'antenna, posso rintracciare l'utente (***Rintracciabilità****l*)
    - Poiché aree sovrapposte, fornisce continuità della connessione (***Handover***)
  - Rete divisa tra **reti di accesso (*antenne / celle*) e rete di *core (rete fissa operatore)* ->** tutte antenne/celle sono connesse ad una rete di *core*, appartenente all'operatore
  - Oggi operatori telefonici usano **GSM** per la telefonia >> non consuma banda delle reti 3g e LTE
    - Per telefonate, si utilizza Voice-over-LTE (telefonate ad alta qualità audio)
  - **Reti satellitari >> GEO** (35.000km copertura, 3 satelliti per copertura globale – trasmissioni broadcast e servizi dati, alto RTT) **MEO** (23.000km copertura – 10+ satelliti, GPS) **LEO**(<1000km, 50+ satelliti – telefonia satellitare e servizi dati a bassa latenza)
- ***Tecnologie obsolete > reti radio / ISDN / HFC / reti satellitari***

Reti di trasporto

Permette connessione tra diverse reti di accesso – possiede *commutatori telefonici* e *router* – topologia a maglia / gerarchica

- Le reti di trasporto sono formate da più operatori telefonici / dati (ISP)
    - *Alcuni sono proprietari delle infrastrutture, altri le affittano e offrono servizi di trasporto*
- **Trasmissione interamente digitale**
- Basata su ***multiplazione gerarchica a divisione di tempo***



Reti di trasporto si basano su *gerarchie sincrone* >> SONET, SDH, STS

- *Sonet* >> synchronous optical network – segnali ottici multipli della velocità base di segnale di 51.84 Mbit/s
- *SDH >>* synchronous digital hierarchy- Sonet ma a livello europeo
- *STS >>* synchronous transport signal (*standard corrispondente ma con segnali elettrici)*

Le reti di trasporto utilizzano spesso una topologia **ad anelli bidirezionale >>** ***alta affidabilità e resistenza ai guasti***

- Sono infatti sistemi altamente sincroni, in cui si ha un flusso continuo di trame /bit in cui vengono incapsulati i dati.
- Gestione trame sempre con divisione PCI / SDU >> *nelle PCI si hanno informazioni di sincronizzazione, canali vocali di servizio, gestione guasti ed errori.*
- **SDH fa sì che diversi flussi possano essere aggregati per crearne uno superiore**

OC level	STS level	SDH level	Mbit/s
OC-1	STS-1		51.84
OC-3	STS-3	STM-1	155.52

Strato collegamento

Funzioni fondamentali strato *collegamento*

- **Delimitazione trama** >> *stabilire dove comincia e finisce il pacchetto* – delimitatori espliciti, indicatori di lunghezza, silenzi tra pacchetti
- **Multiplazione** >> *capacità di trasportare più flussi distinti sullo stesso collegamento (*posso distinguere quale entità dovrà consegnare a quale livello la SDU)

Altre funzionalità

- Indirizzamento locale
- Rilevazione errore
- Controllo di *flusso / sequenza / errore* >>> tramite protocolli a finestra precedentemente accennati
- Protocolli ad accesso multiplo (*solo per canali condivisi*)

Protocolli di strato collegamento

Derivano tutti da ***SDLC > Syncronous Data Link Control*** (vecchio IBM SNA) > SDLC è diventato ADCCP, Advanced Data Communication Control Procedure

- **ISO è diventato HDLC** > High level data Link control, da cui deriva LAP Link Access Procedure e LAPB Link Access Procedure Balanced
- **Da questa famiglia derivano in particolare:**
    - LLC 802.2 – *Logical link control – LAN*
    - PPP - *Point-to-point Protocol – ASDL*
    - LAPDm – *LAP for the mobile D channel*

LLC (LAN) e PPP sono ancora utilizzati sia in reti pubbliche che private.

Nelle reti locali è introdotto il sottostrato MAC, per risolvere il problema dell'accesso multiplo

Dove vengono utilizzati i protocolli di livello 2:

- *Reti pubbliche > collegamenti punto-punto casa <> rete di un gestore. (ex SDLC)*
  - In particolare, operano tra il livello DTE (*data terminal equipment, apparati utente)* e DCE (*data circuit-terminating equipment, apparati del gestore)*
- *Reti private - locali > per connettere apparati in ambienti circoscritti – lab, data center, casa ecc.*
- *Reti pubbliche di trasporto >>* ***atm, frame relay***

FORMATO DELLE PDU IN RETI PUBBLICHE / PRIVATE

- Formato *generico* delle PDU nelle reti pubbliche e private:

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

- **Prima e ultima serie di 01111110 >>> flag di delimitazione >>** delimita la pdu, garantisce trasparenza dei dati >>
  - Evitare che strati superiori facciano utilizzo di questo flag / lo traducano in dati >>
  - ***Bit stuffing >*** ogni sequenza di 5 "1" consecutivi si inserisce uno 0 – ricevitore eliminerà il primo 0 dopo una sequenza di 5 "1"
  - ***Byte stuffing > byte di escape,lo scopo è proteggere il byte successivo – verrà poi scartato dal ricevitore*** (è quello in figura all'inizio e allla fine)
- Indirizzo >> *serve a configurazione comunicazioni multipunto (master -> slave)*

1) Reti pubbliche

PPP – point to point protocol >>> *collegamenti linea telefonica / ASDL tra provider e utenza residenziale.*

Facile da gestire – 3 protocolli:

- **Incapsulamento**
- **Link connection protocol (LCP)**
- **Network Control protocol (NCP)**

PPP presenta le caratteristiche principali del livello collegamento precedentemente accennate – delimita le PDU tramite byte stuffing, *riconosce gli errori*, multiplazione di più protocolli di strato rete, ***negoziazione dell'indirizzo di livello rete (IP) >>> i nodi ai due estremi del collegamento apprendono / configurano i propri indirizzi di rete***

NON UTILIZZA CAMPI DI INDIRIZZO, DEVE OPERARE SU COLLEGAMENTI PUNTO PUNTO

PPP NON SI OCCUPA DI

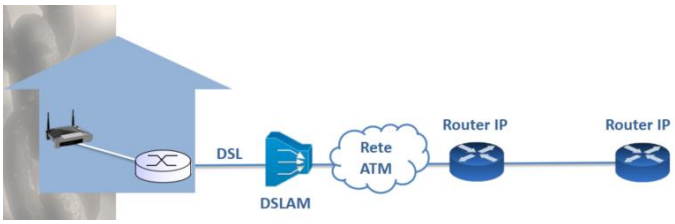
- *Correggere errori(ma li rileva)*
- *Controllo di flusso e mantenimento sequenza*
- *Gestione collegamenti multipunto*

Funzionamento circolare > parte da uno stato di *dead*, passa ad autenticazione e configurazione rete (NCP) negoziando gli indirizzi di rete ai due estremi, apre connessione, rilascia e ritorna a stato di *dead*.

- **NCP** negozia indirizzi IP e definisce modalità trasferimento dei dati

ATM – altro protocollo

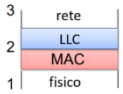
- Rete a pacchetto con servizio a circuito virtuale
- Obiettivo iniziale >> *velocità elevata, bassa latenza per trasporto voce e video*
- *oggi >* ***ASDL / DSLAM && centrale operatore***



2) Reti private / locali

!!!IMPORTANTE!!! Livello 2 diviso in:

- **LLC >** *Logical Link Control (ex. HDLC)*
- **MAC >** *Medium Access Control*



IEEE 802.2 LOGICAL LINK CONTROL >>> LLC

LLC NON RILEVA ERRORI

Protocollo orientato al byte – *non ci sono delimitatori (*delimitazione spetta a MAC) – *nessun controllo degli errori*

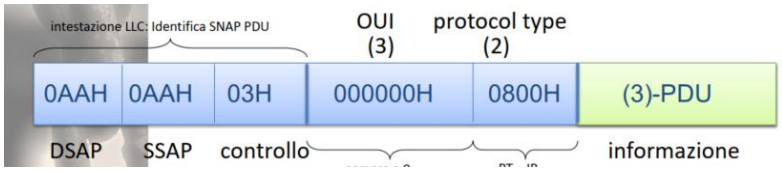
PDU di dimensione variabile, contiene indirizzo sorgente & destinazione

Normalmente protocolli di livello superiore, ovvero 3, necessitano che il tipo di protocollo utilizzato venga specificato dal livello 2 durante la comunicazione >>

- In ethernet c'è un campo apposito, *Protocol Type*
- *In 802.3 Non c'è >> si utilizza SNAP PDU*

802.3 e SNAP PDU

È una PDU aggiuntiva introdotta in 802.3 – poiché i byte che dovrebbero essere usati per indicare il *Protocol type* sono usati per indicare invece la lunghezza della trama, servono byte aggiuntivi -> *Snap PDU* >> ***indica il protocollo utilizzato al livello superiore (per il livello 3 quindi)***



DSAP - SSAP destination e source service access point - i 5 byte (OUI e PROTOCOL TYPE) servono ad identificare possibili protocolli non conosciuti dal livello superiore (SNAP)

Protocolli di accesso per reti locali (LAN)

Local Area Network – estensione geografica ridotta

- Mezzo trasmissivo solitamente *condiviso* (trasmissione *broadcast*)
  - *Un solo nodo può trasmettere alla volta >> + velocità*
  - ***Topologie >>** bus / bus monodirezionale, anello, stella*
- Solito problema di condivisione del canale >> *non si usa **multiplazione con allocazione statica** ma bensì **multiplazione statistica >>>** ad ogni terminale si attribuisce una quantità dinamica del canale*
- **Multiplazione statistica (*dinamica*) si basa su variazione di:**

- *Time / Frequency / Code division*

Parametri reti LAN

- ***Throughput >*** Capacità e traffico smaltito
- ***Equità***
- ***Ritardo***
- ***Numero di nodi >*** lunghezza della rete, topologia, robustezza

Tipologie reti LAN

- **Contesa / accesso casuale (Ethernet, WiFi)**
- Accesso ordinato (Token Ring, Token Bus, FDDI)
- Slot con prenotazione(DQDB)

Comunicazione di un nodo può andare in collisione con un altro nodo che sta cercando di comunicare >> ***collisione,*** **Protocolli MAC ad accesso casuale riducono questa possibilità**

- *MAC riduce la probabilità di collisione, inoltre permette di riconoscerla e recuperarla*
- *Es. Protocollo Aloha >> nel caso due trasmettitori andassero in collisione, veniva applicato ad entrambi un timer casuale prima del quale potessero ritrasmettere (**backoff**)>>>* sistema però instabile poiché in caso di collisioni si avrebbero ritrasmissioni continue che abbatterebbero il *throughput* del sistema – **aloha garantisce minor ritardo di accesso (nno devo attendere slot)**
- *Slotted Aloha >>* simile ad Aloha ma funzionamento a slot – throughput rimane comunque troppo basso – **protocollo aloha non è stabile, i canali trasmettono senza coordinarsi tra di loro – trasmissioni coincidono con inizio delle trame**

CSMA

Carrier Sense Multiple Access

Canali ascoltano prima di trasmettere – se libero, trasmetto, se occupato, ritardo la trasmissione.

- **CSMA 1-persistenza >>** trasmette appena il canale è libero
  - **CSMA non-persistente >>** dopo un tempo casuale controlla se canale è libero, e ritrasmette *(può comunque causare collisioni nel momento in cui due canali sono in attesa e trasmettono insieme appena canale è libero)*
- **CSMA p-persistente >>** trasmetto appena canale è libero con probabilità p, o rimando con probabilità (1-p)

Collisioni rimangono possibili >> trasmettitori capiscono se canale è libero tramite misurazioni *locali* dello stato della rete – quindi eventuali ritardi di trasmissione possono influenzare questa capacità – canale può sembrare libero ma in realtà occupato >>> *la distanza aumenta la probabilità di collisione*

CSMA-CD, CSMA-CA

Collision detection && collision avoidance

CSMA-CD >> *Ethernet* >> rilevazione delle collisioni – monitoro il canale durante la mia trasmissione – se sento solo la mia procedo, se sento altro interrompo.

- **Trasmissione è completata nel momento in cui non ho rilevato collisioni**

- **Dominio di collisione >>** *parte di rete in cui se due stazioni trasmettono, entrano in collisione >>* influenzata da distanza tra le stazioni e dimensione minima delle trame
    - se due stazioni molto distanti, può essere che una trama raggiunga la destinazione dopo che l'altra abbia già concluso la sua trasmissione, e non c'è collisione

I vantaggi di CSMA-CD sono che è facile sospendere la trasmissione in caso di collisione, risparmiando risorse – *molto facile in reti LAN, prestazioni migliori su reti più piccole.*

La CSMA 1-persistente ha infatti un ritardo di accesso inferiore vista la dimensione ridotta della rete e delle distanze.

Può risultare instabile in seguito ad un *backoff* esponenziale.

CSMA-CA >> ***Canali Radio, Reti WiFi 802.11 >>*** prevenzione delle collisioni (*su reti radio bisogna prevenire le collisioni*)

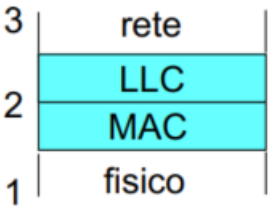
- CSMA non-persistent *based*
- Ricevitore manda ACK di ricezione
    - **L'ACK di ricezione è mandato dopo un tempo SIFS, di durata inferiore al DIFS >>> questo perché la conferma di messaggi ha priorità più alta della trasmissione in sé di altri.** Se ACK non viene ricevuto, backoff + ritrasmissione
- **DIFS >>** quantità di tempo per cui se il canale rimane libero, si procede con la trasmissione
    - Se canale occupato o lo diventa durante il DIFS, sospendo la trasmissione (*backoff, tempo casuale*) – appena termina backoff, ritrasmetto

Collisioni ancora possibili >> due stazioni lanciano backoff in stesso momento, proveranno a ritrasmettere in stesso momento -> collisione

Standard Reti Locali

Standard IEEE 802.2

- 802.1 >> *Internetworking*
- 802.2 >> *LLC, Logical Link Control* **LLC** > specificano le modalità di comunicazione, non l'oggetto > *multiplazione*
- 802.3-4-5-6 >> *Media Access && Physical Access* **MAC >** delimitazione trama (silenzi tra pacchetti), rilevazione errore, indirizzamento, multiplazione
    - **802.3 > CSMA/CD Ethenret**
    - **802.11 > Wireless Networks**
    - **802.15 > Bluetooth e reti di sensori**



Quindi allo strato 2, LLC 802.2 viene usato per la multiplazione, e MAC per delimitazione trama, instradamento e rilevazione errori.

- LLC permette di multiplare più protocolli di stato superiore

MAC > instradamento

I protocolli di sottolivello MAC (Medium Access Control) hanno scopo di permettere al condivisione di un canale broadcast mediante un algoritmo distribuito

MAC permette di identificare in maniera univoca i dispositivi, trasmettitori o ricevitori, tra i nodi della LAN.

- Identificatori univoci globalmente, 6/8 byte > 3 significativi, *organization unique id*
    - Primi 3 byte distinguono in modo *univoco* chi sta producendo il traffico

- **UNICAST / MULTICAST >** se si riferiscono a una sola / gruppi di stazioni
- **BROADCAST >** *si riferiscono a tutte le stazioni, FF FF FF FF FF FF*

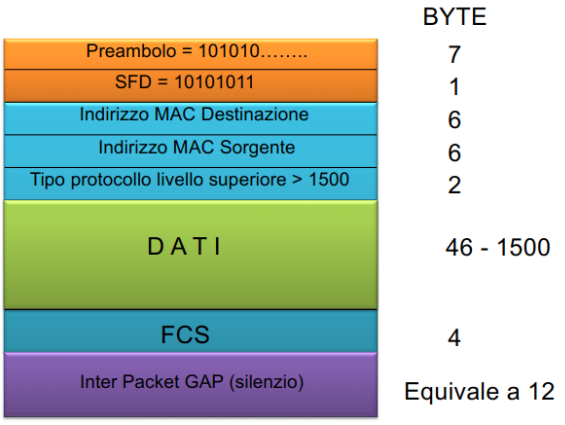
Ne segue che, un nodo, accetterà un pacchetto se >>

- Indirizzo MAC destinazione == indirizzo MAC del nodo
- Indirizzo MAC del nodo appartiene a gruppo MULTICAST
- Se indirizzo MAC destinazione *broadcast*, lo accetterà in ogni caso (*spesso usato in ARP quando non so chi ha un indirizzo MAC specifico, mando in broadcast e mi risponderà solo quello con l'indirizzo che sto cercando)*

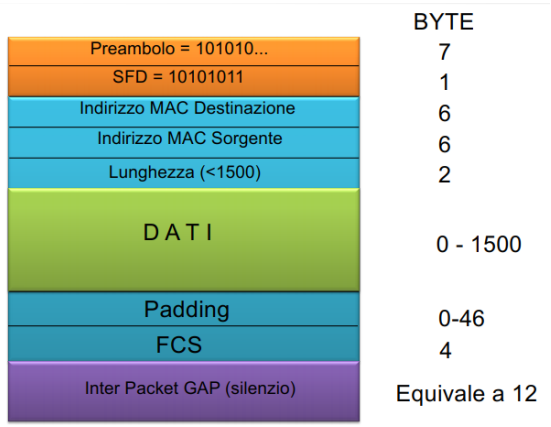
ETHERNET

Reti LAN cablate e metropolitane (MAN) – utilizzano CSMA-CD

- Commutazione di pacchetto a livello collegamento
- ETHERNET e IEEE 802.3 differiscono per caratteristiche di livello MAC e livello fisico



Ethernet, contiene tipo di protocollo



IEEE 802.3 - contiene lunghezza trama

Ethernet > fornisce anche info su protocollo per livello superiore

IEEE 802.3 > mette solo lunghezza trama – LLC si occupa poi del protocollo + payload trama

(dalle due immagini si può dedurre che se il campo è > 1500, allora è il protocollo stesso – sennò si ha la lunghezza della trama, e quindi livello rete gestirà poi il protocollo)

Il preambolo è necessario per consentire al nodo ricevitore di sintonizzare il proprio orologio con l'orologio del nodo trasmettitore

LA LUNGHEZZA DELLE UNITA' DATI VARIANO QUINDI TRA 64 E 1518 OTTETTI

ETHERNET && IEEE 802.3

STRATO MAC

Entrambi usano CSMA-CD 1-persistente – *collisioni possibili vista distanza fisica tra stazioni*

- ***Sequenza di jamming >>*** rende invalido qualsiasi pacchetto interrotto in caso di collisione
- ***Se stazioni in stesso dominio di collisione, Dimensione trama > RTT***
    - La dimensione minima della trama varia quindi in base alla velocità di trasmissione e la dimensione della rete / canale.

- Come accennato prima, ethernet classico funziona al meglio in reti ridotte e non sovraccaricate – ethernet è ad accesso casuale, quindi NO PRIORITA' - nessuna conferma di ricezione.

STRATO FISICO

Ethernet nata come trasmissione su cavo coassiale >>> un mezzo di trasmissione ethernet si rappresenta come

- *Numero (velocità di trasmissione) – modalità di trasmissione – mezzo fisico di trasmissione*
  - *Es. 10 Base T >> cavo ethernet con velocità max 10Mb/s classico (base) con doppino (twisted pair T)*

Ora ethernet si basa su un ***livello fisico commutato >>*** tutte reti Ethernet usano cavi UTP / fibre ottiche

- Topologia è stella attiva (*switch*) o passiva (*hub*)
- Sono aumentate velocità di trasmissione >>> *maggiore velocità, o riduco distanza oppure aumento la dimensione minima del pacchetto*

Nello standard commerciale Ethernet (protocolli di accesso CSMA/CD), si è definita una dimensione minima di pacchetto per garantire una corretta identificazione delle collisioni >> **su ethernet la collisione è funzione della lunghezza del pacchetto**

IEEE 802.3u Fast Ethernet > 100 Mb/s, distanza max 100m

Gigabit Ethernet 802.3ab > standard su qualsiasi pc – controllo di flusso (definisce rapporto *master – slave*) e garantisce *backwards compatibility* con altri mezzi di comunicazione.

- Può usare **Jumbo Frame >>** frame con dimensione maggiore per far fronte a maggiore velocità di trasmissione (*usati solo quando sia trasmettitore che ricevitore hanno stessa velocità di trasmissione*)

10 Gigabit Ethernet > non usa più CSMA-CD, fibra ottica, distanze medio-brevi

Commutatori di livello collegamento – *switch*

Come già visto, le reti locali sono più affidabili, veloci e meno costose.

Si passa quindi ad una topologia di stella gerarchica, in cui il centro è un *hub* (***ripetitore***) oppure uno *switch*

- Maggiore copertura geografica – maggior numero di utenti e stazioni supportate
- *Switch* sono "invisibili" >>> non operano sui protocolli né richiedono modifiche dai terminali

Switch è un centro stella attivo:

- Laddove gli *hub* si limitava a ricondividere i semplici segnali e non rilevare collisioni, **lo switch è un nodo di commutazione Store & Forward >>** riceve completamente la trama e la ritrasmette, con eventuali protocolli di accesso (se ricevuti).
  - **Switch ritrasmette quindi in modo selettivo su una o più porte di uscita le trame ricevute da una porta in entrata.**

- La trasmissione selettiva fa sì che si evitino collisioni > ad ogni porta corrisponde una connessione punto-punto, e quindi un solo trasmettitore e un solo ricevitore. *(ad ogni porta è connesso un cavo UTP – trasmetto e ricevo da cavi separati)*
- **Non serve più quindi CSMA-CD >> si usa direttamente Ethernet come protocollo di *framing* di livello 2.**

Switch non ha indirizzo MAC, ma uno *switch_id* ed un *port_id* per ogni porta.

Switch >> Transparent Switching >>

Processo per cui l'instaurazione di uno switch ed il suo funzionamento sono resi inpercepibili da parte dell'utente

- **Address Learning =>** switch acquisisce indirizzi e crea una tabella di indirizzi in modo autonomo, tramite utilizzo di un algoritmo *Spanning Tree* ed un processo di *backward learning*.
- **Frame forwarding =>** switch ritrasmette trame ricevute, inoltrandole ai corretti destinatari, controllando la sua tabella di instradamento.

Address learning > tabelle di instradamento sono inizialmente vuote; per ogni trama che riceve, si segna il MAC address del trasmettitore e lo associa ad una porta – in futuro, saprà a quale porta corrisponde quel mac address > ***backward learning.***

- Questo viene fatto utilizzando uno **Spanning tree algorithm** >> l'algoritmo permette infatti di eliminare eventuali cicli (anelli), che manderebbero in un loop i pacchetti – e di creare in modo completamente dinamico per ogni switch la sua tabella di instradamento.

Frame forwarding > quando riceverò poi in futuro una trama corretta con una determinata destinazione, cercherò in mia tabella di instradamento a quale porta corrisponde il destinatario di quel pacchetto.

Switch e VLAN – *Virtual Lan*

Le VLAN sono lan virtuali, ovvero host fisicamente collegati ma logicamente *partizionati in LAN separate.*

- ***Definisco quindi sottoinsiemi di una stessa LAN***
- Ogni indirizzo MAC avrà quindi un ulteriore tag, per indicare la sua VLAN di appartenenza.
- Usato in reti locali da amministratori per bloccare utenti e terminali indesiderati dall'usufruire della rete.

WiFi – IEEE 802.11

IEEE 802.11 è la famiglia di standard che regola le reti Wireless – si occupa di

- *Strato fisico (comunicazione radio)*
- *Strato MAC (basato su CSMA-CA)*
- *Interconnessione tra dispositivi e sicurezza*

È infatti una *certificazione di interoperabilità e aderenza allo standard*.

- Terminali comunicano tramite un ***access point*, che può fornire accesso verso Internet**
  - Il suo funzionamento è identico a quello di uno *switch* a livello MAC
- Può anche non esserci un access point -> comunicazione diretta tra terminali

Strato fisico

Opera su bande di frequenze *unlicensed* – 2.4GHz e 5GHz

- *Coesiste con Bluetooth, telefoni cordless, baby monitor, forni a microonde…*
- Access point stabilisce *infrastruttura, banda e relativo canale.*
- ***Velocità massima dipende dalla versione.***

Versione	Velocità max
802.11b	11 Mb/s
802.11g – 802.11a	54 Mb/s
802.11n (WiFi 4)	600 Mb/s
802.11ac (WiFi 5)	~7 Gb/s
802.11ax (WiFi 6)	~11 Gb/s

Strato MAC

Utilizza *DCF,* obbligatorio per IEEE 802.11 (WiFi)

- **DCF** *Distributed Coordination Function* >> si basa su CSMA-CA, per ogni pacchetto la stazione deve vincere contesa per possesso del canale
  - **Stazioni half-duplex >** o trasmettono, o ricevono.
  - **La modalità full-duplex può essere utilizzata tra una stazione d'utente e una porta di uno switch, oppure tra le porte di due switch**

Ripasso > in una possibile trasmissione, quindi, trasmettitore controlla se canale libero per DIFS (intervallo di tempo predefinito)(se occupato, aspetta tempo casuale, se fallisce – backoff esponenziale) e trasmette la trama – ricevitore se riceve dati manda ACK dopo SIFS - le altre stazioni nel mentre non compiono alcuna azione per una durata *NAV* – *nell'intestazione della trama viene inserita anche la durata del RTT (dentro PCI)*

Problema terminale nascosto > caso in cui A comunica con access point, ed esiste un terminale B non raggiungibile da A. A essendo troppo lontano non può rilevare se B sta comunicando, stessa cosa per B -> comunicano insieme e vanno in collisione. **Soluzione >** DFS con handshaking > AP invia una microtrama per verificare se avvengono collisioni, e poi procede con trasmissione **(l'invio dei pacchetti RTS e del CTS riduce la probabilità delle collisioni sui pacchetti dati, ma non le evita del tutto)**

Problema velocità trasmissive > diversi terminali connessi allo stesso Access Point rallentano la velocità di trasmissione del canale >>> questa si omologa infatti alla velocità di trasmissione più bassa tra i terminali presenti.

Bluetooth

Prima 802.15.1, ora Bluetooth special interest group (SIG)

Si basa su una *trasmissione radio a corto raggio* => collegamento periferiche, audio, reti dati domestiche (*home networking*) – trasmissione avviene infatti su *linea seriale*

Due esempi

1) **Piconet >** 1 master, tanti slave – slave interagiscono solo con il master e non tra di loro e solo sotto richiesta
2) **Scatternet >** collegamento tra più piconet. Collegamento tra diversi nodi effettuato tramite *bridge*

Es. un Telefono può essere un master, cuffie bluetooth – altre periferiche sono slave. Anche computer master, mouse / tastiera ecc. slave. Non c'è motivo per cui cuffie, mouse, tastiera dovrebbero comunicare tra loro.

Caratteristiche:

- Trasmissione radio
- Link Control baseband > *le trame e l'accesso sono definite in modo simile a MAC*
- Link Manager gestisce canali logici, cifratura
- Logical Link Control Adaptation > multiplexing protocolli, gestione messaggi (*simile LLC 802.2)*
- ***Bluetooth si basa su frequency-hopping spread spectrum >*** I dispositivi "saltano" da una frequenza all'altra in modo casuale, secondo ordine master.
    - *Master trasmette in trame dispari, slave in pari. SI mantiene così il rapporto comunicazione master <>slave e non slave<>slave, slave comunica solo se riceve comunicazione dispari.*
- **Trama bluetooth** trasmessa su due tipi di canali:
    - **SCO >** syncronous connection oriented, protetti, non vengono mai ritrasmessi, usati per voce.
    - **ACL >** asyncronous connectionless > usati per I dati, possono essere ritrasmessi
- Dispositivi bluetooth di ultima generazione introducono *modalità LE > Low Energy >* maggiore reattività, minor consumo > limita bitrate e dimensione trama.

La tipologia di servizio fornita da un dispositivo Bluetooth è definita in modo chiaro ed esplicito, da una lista di possibili servizi:

Questi servizi sono infatti definiti dalla Bluetooth SIG – un dispositivo non deve per forza implementarli tutti, ma solo quelli che fanno a caso del servizio che forniscono.

Strato Rete

Routing

Il routing (instradamento) è effettuato consultando le tabelle di instradamento – questo avviene quindi per ogni PDU trasferita, e quindi ogni connessione al circuito virtuale. *(ogni pacchetto contiene informazioni nella sua PCI per quanto riguarda l'instradamento, e quindi ne fa utilizzo)*

Il routing si basa su

- ***Protocolli di instradamento (****forniscono info su quale strada possono prendere i pacchetti; definizione delle modalità di scambio di informazioni****)***
- ***Algoritmi di instradamento (****decisione del percorso ottimale di un pacchetto****)***
- ***Procedure di inoltro pacchetti (****forwarding, operazioni necessarie per instradare i singoli pacchetti verso la corretta porta di uscita****)***

I nodi della rete non conoscono infatti tutti i nodi presenti nella rete – posseggono solamente un indirizzo (che sia Mac o IP) a cui consegnare un'informazione – spetta al *routing* effettuare con successo la consegna.

Routing è funzione principale di stato rete.

Altre funzioni sono:

- ***Tariffazione***
- ***Controllo congestione >***
    - *Pre-allocazione memorie*
    - *Scarto pacchetti*
    - *Invio segnali di congestione*
    - *Il controllo di congestione si occupa infatti di far si che il livello trasposto abbia le risorse necessarie alla comunicazione.*

Algoritmi di instradamento

Obiettivo > determinare *buon* percorso (*costo minimo*), come sequenza di *link / nodi*.

- Trasformazione della topologia in un grafo >>> nodi > vertici, link > archi
- Il *costo* di ogni attraversamento dipende dalla *distanza, ritardo, livello di congestione.*
    - Valori possono variare nel tempo; motivo per cui gli algoritmi spesso aggiornano costantemente i valori per rimanere aggiornati con lo stato della rete.
    - **In un calcolo dinamico del percorso migliore, fare attenzione alla scelta sempre del prossimo nodo >** se i costi cambiano di continuo, magari il prossimo nodo ottimale potrebbe allontanarmi. Ricordarsi sempre quindi una sorta di "direzione" del flusso.

Tipi di algoritmi di instradamento

- **Semplici >**
    - **Random >** scelgo a caso una porta in uscita
    - **Flooding >** inoltro a tutte le possibili porte (Simile concetto broadcast)
    - **Deflessione / Hot potato >** instrado verso porta corretta; sennò verso altra porta libera
- **Complessi >**
    - Centralizzato > un nodo si occupa di raccogliere info da tutti gli altri nodi – calcola poi i percorsi – e ridistribuisce a tutti i nodi i risultati.
    - Distribuito > tutti i nodi si scambiano informazioni tra di loro (usano quindi protocolli di instradamento per farlo) – calcolano così i percorsi ottimali gli uni con gli altri.

Gli algoritmi *centralizzati* permettono di usare metriche più complesse, e ne risulta un instradamento più *coerente*.

- Abbiamo però un punto critico sul nodo centrale che effettua il calcolo dell'algoritmo.
- Congestione nel momento di scambio informazione da nodo centrale >> tutti altri nodi

Gli algoritmi *distribuiti* sono più robusti ai guasti e hanno una distribuzione delle informazioni *uniforme e maggiore*.

- Richiede "intelligenza" dei nodi – errori da parte di un singolo nodo creano errori a cascata

Algoritmi distribuiti

- Globale > *tutti nodi conoscono topologia completa, compresi costi dei canali. Tutti i nodi si scambiano informazioni* > ***Algoritmo Link state***
- Parziale > *i nodi conoscono solo i nodi a cui sono collegati e i loro rispettivi costi. Si scambiano informazioni solo con i nodi adiacenti* > ***Algoritmo Distance Vector***

1) Link State

Ogni nodo invia in broadcast il costo dei propri canali – **ogni nodo calcola così la propria distanza da tutti gli altri, ovvero ricostruisce la topologia della rete.**> O(N^2)

- L'algoritmo di Dijkstra è usato come algoritmo di Link state >>> poiché l'algoritmo di Dijkstra, dato un nodo, calcola la distanza minima di tutti gli altri nodi da sé stesso, *applicare Dijkstra a tutti nodi fa sì che si abbia la distanza minima di ogni nodo da tutti gli altri.*
- **Funzionamento** > dato il nodo iniziale, setto la distanza di sé stesso dall'origine a 0. Setto quindi la distanza di tutti gli altri dall'origine a +infinito. Ad ogni iterazione, esamino il vertice con distanza minore disponibile (adiacente) al vertice attuale -> se l'arco che li collega + la distanza dall'origine del vertice attuale < distanza da origine vertice adiacente, aggiorna la distanza dall'origine del vertice adiacente al nuovo valore.(ovvero distanza origine vertice attuale + peso arco vertice attuale – vertice adiacente). Ripeti per tutti i vertici.
- **O(NlogN)**

2) Distance Vector

Raccolgo le informazioni dai vertici adiacenti – sono algoritmi iterativi, *continuano finché i nodi non hanno nuove informazioni da scambiare.*

Algoritmo distribuito, ogni nodo comunica con l'adiacente il suo costo – e le destinazioni che può raggiungere.

- Il vertice che riceve le informazioni, *aggiorna se necessario la propria **routing table** con le nuove informazioni ricevute.*
- **Facile da implementare, lento a convergere && propaga errori.**
- **Funzionamento >** prendo ogni nodo in considerazione – per ogni vicino, guarda quanto costa raggiungere la specifica destinazione > li metto a confronto e scelgo quello con costo minore.Ogni nodo sceglie quindi il percorso minimo verso un nodo basandosi sulla distanza minima verso i suoi vicini. (se devo fare A->C, sceglierò i percorsi MINIMI che mi permettono di fare A->B e poi B->C, con B nodo intermedio A->B->C). In un certo senso, ogni nodo fa un ciclo infinito, in attesa di un aggiornamento dagli altri nodi. Questo però vale anche nel caso di errore >> se ho un errore, questo si propagherà, e causerà ricalcoli negli altri vertici, ripropagandolo di continuo.**Questo si risolve in parte con il metodo *split-horizon* > fa si che la distanza dal vertice precedente sia infinito, così che non si possano avere cicli (**in parte perché tanto si creeranno solo cicli più lunghi ma poi si tornerà sempre indietro in caso di errore)

Algoritmi Link State e Distance Vector a confronto:

- *Link State* è più adatto a reti molto lunghe ed estese – con molti messaggi brevi che vengono mandati a tutti.
  - **Facile convergenza**
  - **In caso di errore >** ogni nodo calcola la propria tabella, tutti sbagliano – o magari tagliano fuori un determinato nodo – l'errore di un nodo non è causa dell'errore in altri nodi.
- *Distance Vector* è più adatto a reti piccole – pochi messaggi ma lunghi.
  - **Convergenza richiede prima aggiornamento di altri nodi in modo continuo >** *tempo di convergenza variabile*
  - **In caso di errore >** se nodo ad esempio annuncia costo minimo scorretto, questo viene propagato in tutti gli altri nodi >> tutte tabelle sbagliate, si creano anelli ecc.

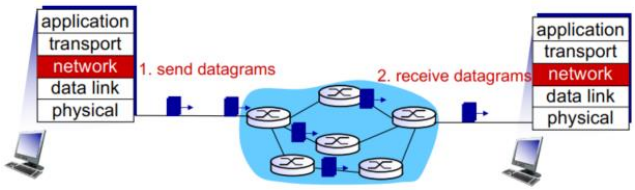
Livello rete

L'obiettivo è quello di trasportare un segmento da un trasmettitore ad un ricevitore – questo viene fatto mettendo tali segmenti dentro dei datagram.

I router analizzano i campi *header* dentro ogni datagram e decidono dove mandare i pacchetti - possono eseguire la frammentazione dei datagram, ma non il riassemblaggio

Servizi **connectionless >>** *datagram network*

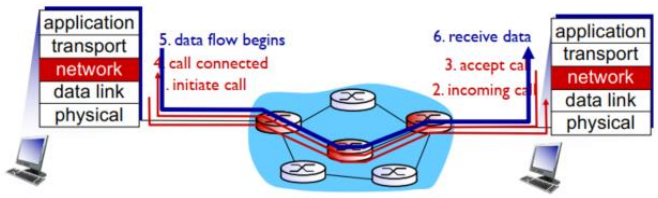
- Pacchetti inoltrati semplicemente basandosi sul campo *destinazione* dell'header



-
- Datagram network opera con le tabelle di routing accennate precedentemente. Utilizzano quindi dei protocolli di instradamento, e degli algoritmi di routing – come *Link State* e *Distance Vector.*
- **È più elastico, può adattarsi e ha recupero di errori.**
- **Garantisce un rilevamento dell'errore solo sull'intestazione del datagram**

Servizi **connection-oriented >>** Circuiti virtuali, formati da

- *Percorso sorgente -> destinazione*
- *Numero di nodi da attraversare*
- *Porte di entrata / uscita nei router*
- un circuito virtuale si occupa quindi prima di instaurare una connessione, simile alla fase di segnalazione. Procede poi dopo alla trasmissione dei pacchetti in sé



-
- **Si basa di più sulla comunicazione umana – sono sistemi evoluti da quelli della telefonia** (*motivo per cui la fase di creazione del circuito ricorda quella di segnalazione di una commutazione di circuito)*

Il livello rete sta tra il livello *trasporto (TCP/UDP) ed il livello link.*

Include

- ***Protocolli di Routing >>*** *RIP, OSPF, BGP*
- ***Protocollo IP >*** *convenzioni per indirizzi, formato dei datagram, gestione dei pacchetti*
- ***Protocollo ICMP >*** *segnalazione errori, segnalazione ai router*

IL PROTOCOLLO IP è UN PROTOCOLL CONNECTIONLESS E SENZA GARANZIA DI CONSEGNA – NON RICHIEDE ACK DI CONFERMA

Struttura datagram – riassunto

Una parte è assegnato ai *data* – l'altra, ovvero tutta la *header*, è costituita da:

- IP protocol version number
- Head length && total length
- Type of service
- **Flag che indica se c'è stata frammentazione >>** molto importante; Internet è una rete di reti, e ogni router ha la sua MTU *maximum transfer unit* (max payload trama possibile) – se passo da rete con MTU maggiore a rete con MTU minore, dovrò frammentare
- Hops rimanenti
- TTL (time to live) – deve essere > Massimo tempo trasmissione pacchetti
- Tipo di trasmissione al livello superiore (TCP/UDP)
- *Ip destinazione*
- *Ip sorgente*

Campo poll / final

Il campo Poll e Final è un solo bit e ha due utilizzi. È chiamato Poll se viene usato dalla stazione primaria per chiedere una risposta alle stazioni secondarie, invece Final quando viene usato da una stazione secondaria per indicare una risposta al termine della trasmissione. Ha significato solo se impostato a 1.

IP ADDRESSING

Indirizzo IP è un identificatore univoco a 32 bit per un host, assegnato da un router.

Un'interfaccia è una connessione tra host e router ed un collegamento fisico.

Gli indirizzi IP utilizzano la notazione decimale puntata >> 223.1.1.1

-ogni elemento è rappresentato da 8 bit

Un indirizzo IP è formato dalla parte di *rete (*anche detta subnet*), e dalla parte di *host.*

- Elementi appartenenti alla stessa rete avranno parti host differenti.
- Elementi appartententi a subnet della stessa rete avranno parte rete diverse e host diverse.
- ***una sottorete è infatti un set di interfacce in cui la parte di subnet dell'indirizzo IP è esattamente la stessa – tutte le interfacce che vi appartengono saranno quindi interconnesse.***

Ci deve essere una corrispondenza biunivoca tra le reti fisiche e le sottoreti >> se questa viene verificata, ogni host appartenente a quella rete / sottorete potrà comunicare con altri host in 2 modi:

- **comunicazione a livello collegamento >>** *comunicazione diretta >>* terminali, host non ricorrono al routing / protocollo IP
- **comunicazione a livello rete >>** *comunicazione indiretta >>* terminali, host fanno ricorso al protocollo IP per comunicare tra di loro
- la scelta tra queste due viene fatta comparando la network part di due indirizzi ip > se è la stessa, appartengono alla stessa rete, e quindi userò comunicazione diretta.
    - Se creo tante sottoreti dato un unico router, e voglio far comunicare due host con sottoreti diverse >> non posso usare livello collegamento anche se sono nella stessa rete fisica, ma dovrò usare protocollo IP (livello rete). In questo caso si dice che il router è **one-arm**
    - È anche possibile avere una sottorete identica per due host, ciascuno sotto un router diverso >> in questo caso di dice **proxy ARP**

- **Una porta su una routing table può fare riferimento ad un'intera subnet.**

Casi particolari di **IP ADDRESSING**

1) **Network id** >> 

2) **Indirizzo di broadcast** >> *indirizzo 1.1.1.1* >> i pacchetti inviati in broadcast a livello IP non vengono inoltrati dai router

3) **Broadcast a sottorete >>** 

4) **Loopback >>** 127.x.x.x x qualsiasi, spesso 1 >> pacchetti vengono rimandati a sorgente

INDIRIZZI LAN e IP

Gli indirizzi IP assegnati all'interno di ogni LAN sono univoci – ciò vuol dire che due host non possono avere lo stesso e identico indirizzo IP.

Possono però esistere indirizzi IP uguali all'interno di *diverse* LAN.

CIDR

Il CIDR (Classless Inter Domain Routing), basato sull'uso delle netmasks (maschere) negli indirizzi IP, è una tecnica utilizzata al fine di utilizzare in modo efficiente lo spazio di indirizzamento IP

ARP – Address Resolution Protocol

ARP protocollo che permette di trovare indirizzo MAC di un terminale dato il suo indirizzo IP. Come?

I protocolli di sottolivello MAC (Medium Access Control) hanno scopo di permettere la condivisione di un canale broadcast mediante un algoritmo distribuito

- Ogni nodo possiede la sua *ARP table*
- A deve mandare pacchetto ad un nodo con un determinato MAC address >> manda in *broadcast* richiesta ARP con MAC sorgente + MAC destinazione da cercare – tutti nodi ricevono richiesta – il nodo il cui MAC address combacerà con quello ricercato, risponderà a quel nodo. Entrambi aggiorneranno quindi la loro ARP table.
- ARP è detto **plug and play >** non serve intervento di amministratore
- I pacchetti ARP **non hanno IP destinazione / sorgente. ARP permette infatti di trovare risorse solo in modo locale.**

Mandare un datagram da Host A -> Router RR -> Host B

A conosce l'indirizzo ip di B

A conosce anche indirizzo IP di R (lo ottiene tramite DHCP, default gateway + DNS)

A conosce anche l'indriizzo Mac di R

A manda a R trama con indirizzo mac di R -> trama contiene datagram con indirizzi ip A e B (A->B)

R rimuove la parte di datagram, ed inoltra a B sul livello collegamento la trama mandata da A

Dato un indirizzo ip, capire quale parte è di rete e quale di host

Tre possibili classificazioni

- **Classful >** la divisione è statica – non esiste concetto di sottorete, solo tre possibili dimensioni per reti IP. Poco flessibile, causa maggiore di esaurimento degli indirizzi ip.
- **Subnetting >** prende reti di tipo classful e le fraziona >> crea tante sottoreti partendo da una statica di grosse dimensioni
- **Classless >** indirizzamento senza classi – subnet masks

1) Indirizzamento a classi

3 diverse possibili classi:

- Classe A > 128 reti > 7 bit per rete(primo è 0), 24 per host >> **0-127…**
- Classe B >16k reti, 14 bit per rete (primi 2 riservati), 16 host > **128-191…**
- Classe C > 2m+ reti, 8 per host > **192-223…**

2) Subnetting >>

L'obiettivo principale del subnetting è diminuire la congestione della rete, aumentando la performance.

Inserisco informazioni aggiuntive (subnet mask) per frazionare ancora di più

Si definisce una subnet mask mettendo tutti 1 nella parte di sottorete, e tutti 0 nella parte di host.

Subnet	1	2	4	8	16	32	64	128	256
Host	256	128	64	32	16	8	4	2	1
Subnet Mask	/24	/25	/26	/27	/28	/29	/30	/31	/32

Per una unica sottorete, ad esempio, ci saranno 256 possibili host. Se divido rete in 2 subnet, allora ne avrò 128 per ciascuna sottorete,e così via.

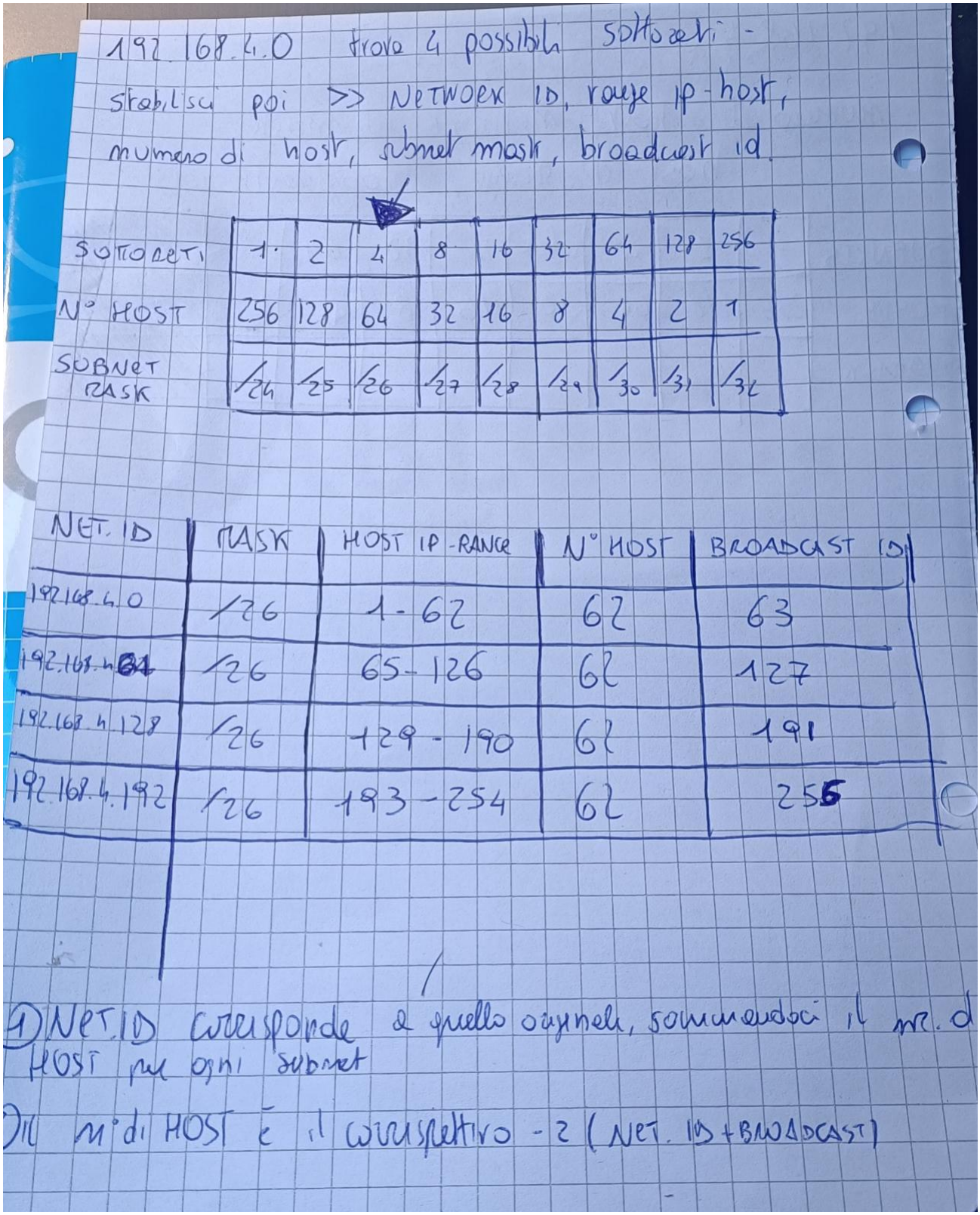
tabella utile per riconoscere divisioni in sottoreti

Prendo 4 subnet, una andrà sprecata >> ne userò 3 come richiesto

Per 4 subnet, ho la divisione della sottorete in 64 possibili host per ogni rete – e una subnet mask corrispettiva di /26

Il numero totale di host è il corrispettivo, sottraendo 2 (Network Id + Broadcast Id)

Il range di ip host validi va dal network Id della sottorete+1 al broadcast id -1 – **ovvero tutti gli ip inclusi tra il network id e il broadcast id di quella sottorete**



esempio esercizio su come ricavare sottorete dato un network id

Ogni dispositivo deve essere dotato di:

- Indirizzo IP
- Netmask >
    - Serve a stabilire il suo network id
- **Un default gateway >>** *first hop router*

Utilizzando le subnet mask, naturalmente, avrò meno host possibili rispetto a non partizionare la rete originale >> questo perché dovrò tenere conto per ogni sottorete del *network ID + broadcast ID.*

- *Partizionare una rete può però fornire grandi vantaggi per quanto riguarda il controllo della congestione e la performance di ogni sotto-rete a sé.*

Indirizzamento gerarchico >> l'utilizzo di sotto-reti rende inoltre possibile indirizzare più facilmente il traffico da internet verso punti precisi >> avrò una serie di ISP che avrà a sua volta al suo interno tantissime sottoreti – io mando a quel particolare ISP il mio datagram, e poi si occuperà lui di mandarlo nel posto esatto.

Ricavare il mio indirizzo IP rete principale >>

Ragionamento al contrario >> se sono in una sottorete, come faccio a capire l'indirizzo IP della rete (*che sarà sicuramente quindi una sottorete di una più grande, del mio ISP)* a cui appartengo?

>>operazione di AND tra indirizzo Ip della mia macchina e la sua subnet mask

Ricavare indirizzo IP RETE PRINCIPALE dato un ip di destinazione >>

Se invece devo mandare un datagram ad un indirizzo IP specifico, dovrò quindi prima contattare la sua rete principale – si occuperà poi lei di inoltrare alla sotto-rete esatta il datagram – ma come faccio?

>> operazione di AND tra indirizzo IP di destinazione (*sarà un indirizzo di sottorete)* e MIA SUBNET MASK

Come sceglie un router la porta in cui mandare dato in uscita?

>>operazione di AND tra indirizzo IP destinazione e SUBNET MASK di ogni entità della routing table >> quelle che ritorneranno un valore positivo, saranno le porte esatte a cui inoltrare il pacchetto.

- Se non viene trovato nulla, viene restituito un messaggio ICMP *destination unreachable*
- tra i risultati ottenuti, si userà **il prefisso più lungo che coincide con l'operazione di AND >> si prenderà la destinazione con la host part più piccola. (***prefisso di rete più lungo***)**

Tabella di Routing – tipologie di instradamento

- Diretto > rete connessa direttamente al router
- Statico > percorsi a reti configurati manualmente, verso host remoti
- Dinamico > percorsi configurati automaticamente verso host remoti >> *routing protocols + ICMP redirect*

Assegnamento indirizzi IP – DCHP

DCHP – *Dynamic Host Configuration Protocol* – assegna automaticamente indirizzi IP agli host, svolto da un server remoto.

Una rete può avere un solo server DHCP.

- Gli indirizzi IP non sono "assegnati", ma *dati in prestito* in un certo senso.
- Ogni host riceve un indirizzo IP nel momento in cui si unisce alla rete. Supporto quindi anche per utenti mobile ecc. – indirizzi IP vengono riutilizzati (una volta che mi sconnetto, indirizzo IP viene ceduto)
- **DHCP è un protocollo di livello applicativo >>** viene inserito all'interno di pacchetti UDP, estende Ip.

Come? Diverse funzioni offerte da DHCP

- Host usa DHCP discover, a indirizzolimited broadcast IP address 255.255.255.255
- **DHCP offre indirizzo Ip con DHCP offer / Host richiede indirizzo ip con DHCP request**
    - DHCP accetta e manda Indirizzo IP richiesto a HOST con **DHCP ack**

Scenario >> non conosco né mio indirizzo IP né quello della rete – come faccio a connettermi e ottenerne uno?

Client si connette a rete, invia DHCP discover >> DHCP server manda in broadcast un pacchetto contenente un indirizzo IP disponibile, preso da un *pool* di indirizzi Ip > *questo pacchetto contiene indirizzo Ip del DHCP server + indirizzo IP disponibile + lifetime dell'indirizzo IP (*dopo il quale DHCP si riprenderà l'IP)

- Pacchetto può contenere anche indirizzo del *first-hop-router*, indirizzo del server *dns*, *network mask.*

Client riceve il pacchetto, e invia una DHCP request con l'indirizzo IP ricevuto -> DHCP server lo riceve e accetta, mandando un DHCP ACK -> Client è ora connesso alla rete e ha il suo indirizzo IP.

Come fa un ISP ad ottenere un blocco di indirizzi? >> ICANN

- Alloca indirizzi
- Gestisce DNS
- Assegna domini

NAT – *unificazione di tutti gli indirizzi privati in un unico indirizzo IP per comunicare con l'esterno*

Gli indirizzi privati non sono univoci – ciò vuol dire che un altro host o ente non potrà contattarmi direttamente.

- Per questo motivo gli indirizzi non sono gestiti dai router, ma sono tutti assegnati ad un unico NAT >> **fa si che una serie di host con un indirizzo privato possano comunicare con dei nodi che si trovano in reti pubbliche.**
- **Dall'esterno, risulterà che la rete locale utilizzi un singolo indirizzo IP per comunicare con Internet.** Quindi un indirizzo IP per tutti i dispositivi appartenenti ad una rete.
- **I pacchetti che quindi escono dalla rete locale avranno tutti lo stesso indirizzo di destinazione (e porte diverse in base al contenuto)**

Un router NAT quindi>>

- Sostituisce nei pacchetti *in uscita* l'indirizzo IP di sorgente con l'indirizzo IP NAT, e la vecchia porta di uscita con una nuova
    - Ogni "traduzione" verrà tenuta conto nella **tabella di nat (nat table) -> ogni porta nuova è associata in modo univoco a quella vecchia usata**
- Sostituisce nei pacchetti *in entrata* l'indirizzo IP di destinazione (che è attualmente quello di NAT, poiché risposta sarà stata nei confronti dell'IP precedente) e PORT di destinazione (che sarà quella nuova inserita precedentemente) con i vecchi valori >>> ***consultando la NAT table***

In un certo senso, il NAT risolve in parte il problema della mancanza di IPv4 >> che dovrebbe però essere risolto dall'introduzione di IPv6.

NAT – problema in *entrata*

Se sono client esterno, non ho modo di comunicare con host all'interno se è presente nat >> **si può usare una PORT configurata staticamente per abilitare comunicazioni da esterno.**

Il DNS (Domain Name Service) È basato su un sistema di server gerarchici, ognuno responsabile del naming di una particolare zona

ICMP – Internet control message protocol

Usato dagli host e router per comunicare informazioni di livello rete.

- Report di errori – host non raggiungibile, rete non raggiungibile, port, protocollo ecc.
- I messaggi ICMP sono portati dentro *datagram IP*
- Esempio >> *Traceroute* >> tiene traccia di tutto il percorso che fa un pacchetto nel raggiungere una determinata destinazione (IP)

IPv6

Perché? Universo degli indirizzi IP a 32 bit quasi in esaurimento. In aggiunta, cambiamenti al formato *header* per migliorare la velocità di processamento e di forwarding.

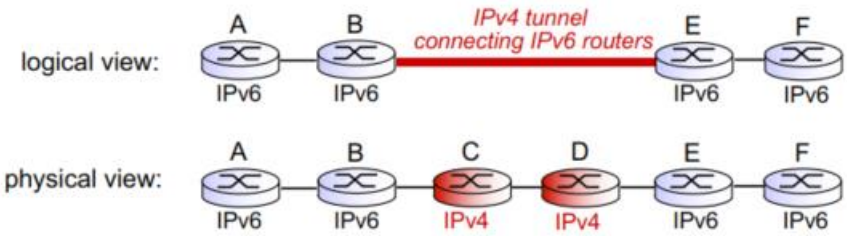
- *Header* di 40 byte
- *Nessuna frammentazione*

Differenza da Ipv4 nella struttura datagram

- *priority >* identifica la priorità del datagram all'interno del flusso
- *flow label >* identificare i datagram appartenenti allo stesso flusso
- *next header >* identificare il protocollo di livello superiore per i dati contenuti
- **è rimosso il checksum**

IPv6 introduce anche ICMPv6, che introduce a sua volta nuovi messaggi come ad esempio "Pacchetto troppo grosso" (sensato poiché non c'è più frammentazione)

Come potranno operare i router nella transizione da IPv4 a IPv6? >> IPv6 datagram verrà trasportato come *payload* all'interno di datagram tra router IPv4



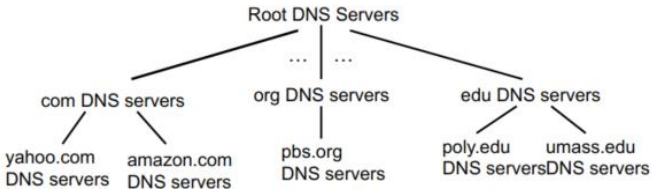
router IPv6 comunicano tra loro usando datagram IPv6- quando devono comunicare con router IPv4, inseriranno i loro datagram come payload di un datagram IPv4.

DNS

Domain name system

Formato da indirizzi ip a 32 bit usati per indirizzare i datagram – permette di *mappare* un indirizzo IP facilmente tramite un nome, e viceversa.

Il Domain Name System DNS è infatti un database distribuito in modo gerarchico, formato da una gerarchia di *name servers*.



- Host e name servers comunicano per effettuare la traduzione indirizzo IP / nome
- **DNS è un protocollo di livello applicazione**
- **DNS** permette la distribuzione del carico su uno stesso e unico nome / Dominio >> *diversi indirizzi IP possono infatti ricollegarsi ad un unico DNS*
- Il server che effettua la traduzione da indirizzo IP > Domain Name è detto **Server autoritativo**
    - il server autoritativo entra in gioco nel momento in cui un server locale non riesca a "risolvere" un host name – in quel caso, il server autoritativo recupererà il mapping esatto per quel host name e lo restituirà al server locale.
    - **Server autoritativi possono essere:**
        - *Top-level* > .com, .org, .edu, .it, .uk (TLD, top level domain)
        - *Authoritative DNS servers* > server appartenenti ad organizzazioni, forniscono mapping per IP appartenenti ad un'organizzazione specifica
        - ***Top level si trova direttamente sotto il root DNS***

Server locale DNS

Il server locale DNS non appartiene necessariamente alla gerarchia dei DNS – anche detto *default name server*.

- Mantiene una cache di tutte le ricerche di host – se risultato è già presente in cache, rimanda subito risultato > **questo perché il server locale DNS mantiene una memoria di tutte le traduzioni ip-domain name recenti**
- Se risultato non è presente in cache, contatta la gerarchia DNS inoltrandogli la query
    - *Le cache hanno un TTL, dopo il quale vengono automaticamente eliminate.*
    - *Alcuni dati in cache potrebbero non essere sincronizzati – in questo caso non si potrà effettuare la traduzione IP – domain name finché non terminerà il TTL o verrà pulita la cache.*

Tipologie DNS

DNS è database composto da diverse *resource records* ***(RR)***

- Ogni risorsa ha il seguente formato >> (nome,valore,tipo,TTL)
- In base alla tipologia, il valore contenuto nei diversi campi può variare.

1) **Tipo = A >>** *semplice traduzione nome ⇔ ip*
    - Nome => hostname
    - Valore => indirizzo IP
2) **Tipo = NS >>** nome = dominio, valore = hostname del server autoritativo per quel dominio
3) **Tipo = CNAME >>** si basa sull'utilizzo di un *alias* per quello che è in realtà un altro nome effettivo del server >> anche se il nome effettivo cambia, l'alias e quindi il nome del dominio che usa l'utente non cambia >>
    - Nome => *alias* (nome fittizio)
    - Value => *nome effettivo* del dominio
4) **Tipo = MX >>** usato per mail server, value => nome del mailserver

DNS e utente comunicano con lo stesso formato di messaggi >> sia *query* che *reply* quindi hanno stesso formato

- Header => 16 bit > contiene identificazione, flag *query / reply*, campo risposta / query, flag se risposta è autoritativa

Tipologie attacchi DNS

- **DDOS >** inondare server root con molto traffico > oggi non più molto effettivo, il traffico viene filtrato e quello indesiderato non raggiunge il server >>> molto più danno DDOS su server di *top-level domain*
- **Man in the middle >>** intercettazione di query da parte / verso DNS
- **DNS poisoning >** inserire in cache DNS query dannose

Livello trasporto

TCP & UDP

RICORDATI: i segmenti di livello trasporto sono contenuti dentro datagram IP. Ne segue che un segmento / pacchetto TCP, avrà solo source – destination PORT.

Il livello trasporto si propone di offrire un trasferimento dati *affidabile,* con *controllo di flusso e congestione.*

Protocolli del livello trasporto:

- **UDP >>** *connectionless,* aggiunge funzionalità di multiplexing a quelle stabilite da IP – aggiunge PORT. Un messaggio UDP dovrà per forzare avere le PORT di sorgente e destinazione.
    - UDP *non è affidabile >>* usato per trasferimenti dati streaming / video / audio – **UDP non ha meccanismo di ritrasmissione** *(non fornisce recupero degli errori)*
    - Non consegna i dati in ordine
    - *Un sistema interamente basato su UDP necessita di un meccanismo implementato da noi sul recupero degli errori.*
    - ***Demultiplazione connectionless inaffidabile***
- **TCP >>** *connection-oriented, controllo della congestione =>* ***TCP trasferimento affidabile, recupero degli errori.***
    - **TCP** è in grado di segmentare il *payload* in pacchetti
    - Controllo congestione, flusso – consegna dati in ordine
    - Stabilisce la connessione - *handshaking*

Il livello trasporto opera direttamente sugli ENDPOINT della trasmissione => sugli host.

- Trasmettitore segmenta il messaggio in pacchetti e lo manda sulla rete.
- Ricevitore riceve segmenti e ricostruisce il messaggio dai pacchetti.
- Livello trasporto mette in comunicazione *processi* (*livello rete mette in comunicazione host; naturalmente livello trasporto si basa su livello rete sennò non potrei mai raggiungere il destinatario*)
- Entrambi UDP e TCP non forniscono garanzie sul ritardo di trasmissione e la larghezza di banda.

Multiplexing => unione di più flussi di informazioni / messaggi in un *unico stream.*

Demultiplexing => divisione a partire da un *unico stream* di informazioni in diversi flussi => controllando *header* di ogni pacchetto e consegnandolo al giusto processo.

- Host riceve datagram IP => ogni datagram IP ha un segmento di livello *trasporto =>* all'interno di ogni segmento trasporto c'è un *port number + ip address* che determina a quale processo va consegnato.
- **Come si rappresenta in modo non ambiguo comunicazione tra due processi su internet >>> (**source IP + dest IP**) + (**source PORT + dest PORT**)**
- ***Connectionless Demultiplexing*** *>>> UDP >>>* obbligatorio specificare PORT number + IP address *>>* quando host riceve segmento, controlla che l'IP coincida con il suo, e se la PORT specificata è *in ascolto* (socket)

***Socket* =>** è una risorsa a livello locale, offerta dal sistema operativo. Si mette in comunicazione con un processo presente su un'altra macchina > permette di comunicare con un insieme di dati (fa da *tramite* sul pipe della comunicazione).

- Ogni socket ha un *port number* > il suo obiettivo è *de-frammentare* la comunicazione in arrivo su un *pipe* di comunicazione
- **UDP >>>** *connectionless demultiplexing >>* socket ha solo *destination PORT*
- **TCP >>** *connection-oriented demultiplexing >>* socket ha (source + dest IP) + (source + dest PORT)
    - L'utente usa tutti e 4 i valori per indirizzare il segmento al socket appropriato.
    - Ne segue che **TCP rende possibili comunicazioni diverse dalla stessa macchina verso un'altra.**
    - Es. A e B comunicano con C >>> comunicazione in questo caso non è ambigua perché tutti i flussi di dati di A e B sono caratterizzati da source + dest IP e source + dest PORT >> ogni flusso di comunicazione e quindi ogni segmento viene correttamente indirizzato al suo socket e alla sua applicazione.
    - In aggiunta, ogni PORT NUMBER caratterizza un servizio specifico (posso anche usare quelli che voglio io) >>> standard utile per facilitare la richiesta di servizi su un webserver.

1) UDP – User Datagram Protocol

Connessione molto semplice, pacchetti possono andare persi. Non c'è ritrasmissione e non c'è ordine di consegna.

Connectionless => no *handshaking*

No controllo di flusso / recupero errori / controllo di congestione => UDP viaggia "il più veloce possibile" sul canale

- **pacchetto UDP contiene solo dest PORT + source PORT (*header*) + payload (*dati*)**

Usato per *streaming multimedia, DNS =>* è infatti meglio non ritrasmettere pacchetti in ritardo in streaming di dati multimediali come voce e video

UDP checksum >> rilevare errori in un segmento che è stato trasmesso.

- Trasmettitore inserisce bit di checksum e lo segnala in header
- Ricevitore legge header e controlla se bit di checksum è presente >> se non lo è, c'è errore

PRINCIPI DI UN TRASFERIMENTO DATI AFFIDABILE

Necessità di un canale affidabile >> bisogna trovare una via di mezzo tra i diversi protocolli finora visti…

- Go Back N >> ACK cumulativi – ritrasmetto tutti pacchetti ancora senza ACK.
- Selective Repeat >> ACK selettivi, ricevitore manda singolo ACK per ogni pacchetto >> *richiede parecchie risorse…*
- *Stop & wait non viene neanche considerato >> non posso in una comunicazione a lunga distanza aspettare ogni singolo pacchetto…*

TCP offre una via di mezzo tra Go Back N e Selective Repeat, adattandosi alle necessità della vita reale.

2) TCP

TCP si basa su uno stream di *byte* unico, indistinto e in sequenza – *ricevitore riceve dati in sequenza, senza accorgersi di come sono stati generati o divisi => anche se TCP divide in 1000 segmenti di dimensione 1, io ne riceverò solo uno di dimensione mille.*

Caratteristiche fondamentali:

- Punto-punto > 1 trasmettitore, 1 ricevitore
- Comunicazione in ordine
- Pipelined > il controllo di congestione e di flusso di TCP determinano la dimensione della finestra >> trasmettitore non manderà in overflow il ricevitore > trasmettitore può mettere in pausa il trasmettitore se necessario
- Connection-oriented > *handshaking => prima di inizio comunicazione, trasmettitore e ricevitore si scambiano messaggi di avvenuta connessione l'uno con l'altro.*
- **TCP protocollo a finestra >> ACK cumulativi >>** *pacchetti hanno al loro interno il numero di sequenza + flag di presenza ACK o meno. (*i dati sono inviati in ordine)
  - **ACK cumulativi**
  - **Singolo timer di ritrasmissione >>** ***scatta in caso di timeout / ACK duplicati >> timer fa riferimento al pacchetto più vecchio NON ancora confermato***

TCP crea infatti una connessione bidirezionale >> si può usare il PIGGYBANKING, aumentando efficienza del protocollo.

Struttura segmento TCP >>>

- **INTESTAZIONE DEVE OBBLIGATORIAMENTE CONTENERE FLAG PER APERTURA E CHIUSURA DELLA CONNESSIONE**
- Source - dest PORT + sequence Number
    - **Sequence number indica la posizione del primo byte del messaggio nella sequenza di byte trasmessi sulla connessione TCP**
- ACK number + numero di bytes che la finestra può ancora ricevere (***receive window***) >> ogni volta che un nodo manda un segmento, informa l'altro di quanti byte può ricevere ancora nella direzione opposta (*controllo di flusso*)
    - **ACK number indica il prossimo byte che l'host trasmettitore si aspetta di ricevere nei segmenti inviati dall'host B**
- Campo checksum
- RST,SYN,FIN > comandi di setup connessione
- Payload

Fasi di trasmissione TCP

Ricordiamo che TCP è protocollo a finestra, che opera con ACK cumulativi e con un flusso di dati in sequenza. C'è un singolo timer di ritrasmissione, che scatta in caso di *timeout (pacchetti persi, errore) – ACK duplicati.*

TCP simile a Go Back N.

Fasi trasmettitore =>

- Creo segmento con numero di sequenza #x > mandalo in stream e avvia il *timer* se non è già avviato
- In caso di *timeout* => ritrasmetti segmento che ha causato *timeout, e riavvia il timeout.*
- In caso di ACK => aggiorna lista di dati confermati e avvia il timer se ci sono ancora pacchetti da confermare.

Generazione degli ACK =>

- Se segmento arriva in ordine giusto e tutti precedenti sono già stati confermati => generazione ACK viene rimandata per un intervallo, in attesa del prossimo segmento. Se non arriva niente, manda ACK.
- Se segmento arriva in ordine giusto ma ci sono ancora dati non confermati => genera ACK cumulativo e conferma tutti, compreso quello appena arrivato.
- Se arriva segmento fuori sequenza, con sequenza più alta di ACK aspettato => manda ACK duplicato con numero di sequenza del segmento atteso (quindi ricevitore indica quale segmento gli manca rispetto a quello superiore ricevuto)
- Se arriva segmento che riempie *buco* nella sequenza precedentemente rilevato => manda subito ACK di quel segmento

Ritrasmissione TCP

Il *timeout* spesso è molto lungo, e non è conveniente aspettare che scada per rimandare pacchetti non ricevuti.

- I segmenti persi vengono *rilevati tramite ACK duplicati.* Se infatti il ricevitore manderà segmenti duplicati, vorrà dire che alcuni segmenti sono stati persi -> ***TCP FAST RETRANSMIT >>*** *trasmettitore rimanda segmento con il numero di sequenza più basso non ancora confermato.*

Timeout TCP > deve essere più lungo del RTT (*round trip time*, che dipende da dimensione pacchetti + velocità trasmissione) >> RTT si basa su tempo di trasmissione + ricezione di un pacchetto ed il suo relativo ACK

- Se troppo corto, ritrasmissioni non necessarie
- Se troppo lungo, reazioni troppo lente a segmenti persi
- **Timeout =** stima RTT + 4 * deviazione RTT (margine di deviazione standard del RTT calcolato)
- **in caso di timeout, la finestra viene ridotta >> non vengono ritrasmessi tutti i pacchetti, ma il più vecchio che ha fatto scattare il *timeout* e non ancora confermato.**

TCP – Controllo di Flusso

Il controllo del flusso di TCP si basa sul concetto che il *ricevitore controlla il trasmettitore* >> così che quest'ultimo non mandi troppe informazioni al ricevitore. Come?

- Ricevitore annuncia lo spazio libero nel *buffer* al trasmettitore nell'header TCP
- Trasmettitore limita il numero di segmenti ancora da confermare alla quantità specificata massima del ricevitore > *garantisce così che non ci sarà overflow per buffer ricevitore*

TCP – Gestione della connessione

Handshaking => Prima dello scambio di dati, il ricevitore ed il trasmettitore si mettono in comunicazione (*simile segnalazione commutazione di circuito*)

- Si comunicano inoltre a vicenda eventuali parametri sulla connessione, come il *buffer size*
- *3 way handshake =>* trasmettitore manda a ricevitore TCP SYN -> ricevitore ritrasmette SYNACK -> trasmettitore manda segmento a ricevitore ed indica che è *live*
- ***Come per commutazione di circuito =>*** quando è necessario chiudere la comunicazione, mandano un segmento *TCP* con il flag **FIN = 1** -> altra parte manderà FIN ACK, connessione viene conclusa.

TCP – Controllo della congestione

Sia controllo congestione che controllo di flusso operano sulla dimensione della finestra – congestione NON sulla dimensione dei pacchetti

RICORDA! Il meccanismo che utilizza TCP per regolare il traffico in uscita si basa sulla grandezza della finestra di trasmissione > maggiore sarà grande la finestra, maggiore traffico si potrà spedire.

Si ha congestione quando troppe sorgenti trasmettono troppi dati, *troppo velocemente* per quanto il ricevitore possa gestire => => oppure sorgente trasmette troppi dati rispetto alla dimensione del buffer in ricezione

Controllo di congestione è diverso dal controllo di flusso => la congestione si manifesta infatti con *pacchetti persi (troppi pacchetti nei buffer) e lunghe attese (code nei buffer)*

Il controllo di congestione allo stesso tempo però è una delle cause maggiori di perdita dei dati.

Controllo della congestione NON opera sulla dimensione dei pacchetti

Due approcci

1) **End-end >>** TCP cerca di ricavare lo stato di congestione basandosi su effetti direttamente da lui visibili – fa misure indirette per capire se code sono piene o no
   - *Non c'è un feedback dal ricevitore*, ci si basa su perdite / ritardi rilevabili dal trasmettitore
2) **Network-assisted >>** tutti i router intermedi informano l'endpoint dello stato di congestione della rete – solitamente tramite un bit flag che indica lo stato della congestione (SNA,DECbit,TCP/IP ECN, ATM)

TCP – Incremento additivo e decremento moltiplicativo

Il controllo della congestione, inoltre, si può basare su un processo di auto-regolazione della dimensione della finestra.

- Aumento, ad esempio, la finestra finché non ho perdita di pacchetti, o raggiungo una certa soglia >> a quel punto mi fermo o diminuisco se noto perdite

L'incremento additivo e moltiplicativo si basa infatti su un aumento **della finestra di trasmissione (*transmission rate) finché non ho perdite >>***

- ***Aumento Additivo >>*** aumento di una quantità fissa ogni RTT, finché non rilevo perdita
- ***Decremento Moltiplicativo >>*** appena rilevo perdita, dimezza la finestra di trasmissione attuale. >> **TCP RENO**

TCP – SLOW START

Lo slow start TCP ha lo scopo di raggiungere velocemente la finestra di trasmissione ideale, partendo da una finestra molto piccola.

- *Finestra parte a dimensione minima, e cresce esponenzialmente per ogni ACK ricevuto.*
- ***Per ogni riscontro ACK ricevuto, cresce di un segmento.***

In caso di errore >> genericamente indicato da un *timeout*, finestra viene impostata di nuovo al minimo iniziale – crescerà di nuovo velocemente fino ad una determinata soglia, e crescerà poi in modo lineare.

- La soglia dopo il quale la crescita della finestra diventa lineare e non più esponenziale equivale a **½** della sua dimensione precedente prima del *timeout >>>* ***TCP RENO >> (in realtà viene portata a metà + 1 segmento)***

TCP FAIRNESS >> se TCP gestisce *k* flussi di informazioni diverse, ogni flusso avrà lo stesso rate di trasmissione.

- Motivo per cui per le trasmissioni audio / video si usa UDP e non TCP >> non devo diminuire il rate di trasmissione di video /Audio, ma deve essere costante, e tollerare eventuali perdite di pacchetto.
- Per la trasmissione dati invece, **TCP permette di instaurare diverse connessioni in parallelo senza perdite di pacchetti.**

LIVELLO APPLICAZIONE

Il livello applicazione si basa sul concetto di applicazioni >> possono essere programmi presenti su diversi sistemi, e che comunicano sulla rete. Le applicazioni sono programmi presenti solamente sugli *end-systems*, ovvero sui terminali degli utenti

- i *web server* non hanno bisogno eseguirle o possederle. (*motivo per cui router ecc. hanno solo 3 livelli e non tutti e 7)*

DUE TIPICHE STRUTTURE

- **client ⇔ server >>>>**
  - server sempre attivo, indirizzo IP statico, *attende di essere contattato*
  - client comunica con il server tramite richieste >> indirizzi IP spesso dinamici, non comunicano l'uno con l'altro, *client avvia la comunicazione*
- **Peer-to-peer ➔ P2P**
  - Non esiste un vero e proprio unico server >> gli utenti comunicano tra loro e forniscono servizi gli uni agli altri
  - *Sistema auto-scalabile >>* a nuovi peer corrispondono nuovi servizi e un aumento della capacità dei servizi già disponibili

Comunicazione dei processi

Processi diversi sullo stesso host possono comunicare tra di loro >> *comunicazione intra-processi*

Client hanno processi che avviano la comunicazione, *Server* hanno processi che attendono la comunicazione.

- I sistemi P2P integrano su uno stesso host sia processi *Client* che *Server*.

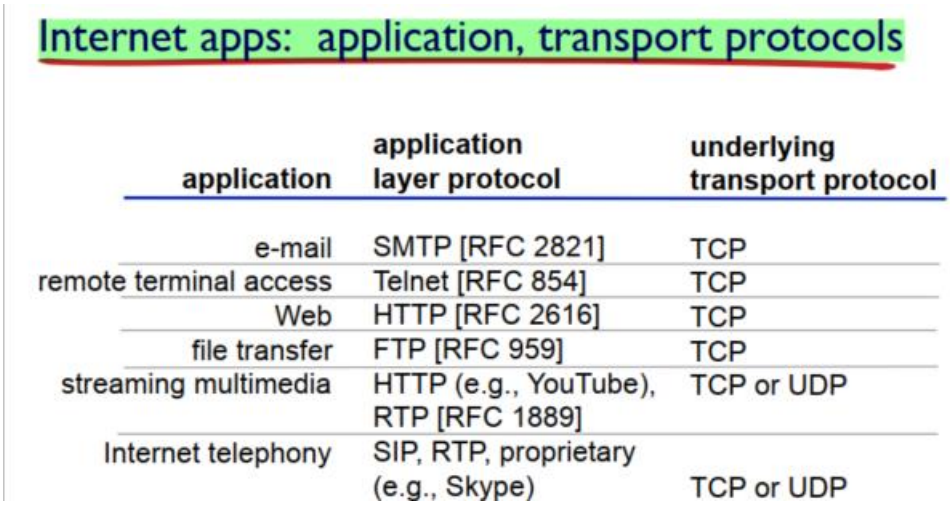
Livello Applicazione – caratteristiche fondamentali

Cosa definisce il livello applicazione?

- Tipo di messaggi scambiati => *request / response*
- Sintassi dei messaggi => *quali campi sono contenuti nei messaggi e come sono delineati*
- Semantica dei messaggi => *significato dei campi contenuti nei messaggi*
- Regolazioni su richieste / risposte dei processi
- Protocolli aperti e protocolli proprietari (protocolli aperti *disponibili a tutti*, protocolli proprietari sono protocolli sviluppati da privati e il cui funzionamento è occulto)

Applicazioni DEVONO essere accompagnati da servizi di trasporto che garantiscano

- **Integrità dei dati >** alcune app richiedono dati 100% corretti, altri permettono perdite (audio video ecc.)
- **Tempismo (*timing*) >** alcune app richiedono ritardi strettamente bassi
- **Throughput >** alcune app funzionano con throughput molto bassi, altre no – throughput richiesto varia dall'applicazione
- **Sicurezza >** dati devono poter essere sicuri – utilizzo di crittografia ecc.



Internet apps: application, transport protocols

application	application layer protocol	underlying transport protocol
e-mail	SMTP [RFC 2821]	TCP
remote terminal access	Telnet [RFC 854]	TCP
Web	HTTP [RFC 2616]	TCP
file transfer	FTP [RFC 959]	TCP
streaming multimedia	HTTP (e.g., YouTube), RTP [RFC 1889]	TCP or UDP
Internet telephony	SIP, RTP, proprietary (e.g., Skype)	TCP or UDP

SSL > TCP e Sicurezza

TCP e UDP non forniscono alcuna forma di crittografia dei dati, e quindi nessuna forma di sicurezza >> si usa **SSL** come protezione aggiuntiva ai protocolli di livello trasporto

- **SSL >** utilizza le stesse librerie di TCP, testato quindi a livello globale – si trova al livello applicazione, e comunica direttamente con TCP – *tutte le informazioni passano infatti attraverso un socket API che le encripta*
- **SSL si piazza tra il livello applicazione ed il livello trasporto >>** SSL va a creare per ogni protocollo applicativo una copia che andrà a lavorare in modo crittografato (motivo per cui possono coesistere su un sito web sia http che https)

Web e http

Pagina web costruita da oggetti >> pagina formata da diversi file HTML che contengono oggetti.

http > Hypertext transfer protocol > http si basa su un modello request / response, il cui obiettivo è scaricare e visualizzare oggetti.

- Client ha un browser che richiede e riceve dati utilizzando il protocollo http
- Il server utilizza il protocollo http inviando oggetti come risposte.
- **http si basa su TCP >** ogni comunicazione si apre infatti con lo stabilimento della connessione da parte di TCP (*handshaking*), lo scambio di messaggi http, seguito dalla chiusura della connessione.
    - **http è stateless >>** il server non mantiene alcuna informazione riguardo allo scambio di dati effettuato – ogni richiesta non è legata alle precedenti. Applicazione può mantenere memoria, ma non server (*più efficiente*)
    - *server che mantengono uno stato (stateful) sono complessi e richiedono alta sincronizzazione, e metodi di recupero del loro stato in caso di crash.*
    - ***http richiede informazioni => protcollo di tipo pull***
- ***Persistent-http >*** permette scambio di più oggetti in una sola connessione
    - Instauro una sola volta la connessione, non serve *header* ogni volta >>
- ***Non-persistent http >*** comunicazione permette scambio di una sola informazione, dopodiché si conclude.
    - *Tempo di trasmissione == 2*RTT + trasmissione file*
    - *2*RTT per ogni file quindi*

Formato HTTP

1) **Request** > formato ASCII (human-readable), tipo di richiesta (GET/POST/HEAD/PUT/DELETE) + *header*

Tipo di richiesta si basa sul metodo scelto per richiedere informazioni al server >> GET,POST

- **POST >** input viene inviato al server all'interno del body
- **GET >** utilizza l'URL, input è inviato al server tramite una richiesta nell'URL stesso
- **PUT >** contrario di GET, richiede di mandare un oggetto direttamente al server (*ad esempio un FILE*)

2) Response > messaggio di stato (es. 200, ready) + header + data

Messaggio di stato >

- **200 – messaggi con 2 >> *success***
- **301 – messaggi con 3 >> *moved***
- **400 > bad request >>** messaggio non compreso dal server (errore utente)
- **500 > *server error***

COOKIES >> http e STATE

http è stateless >> Cookie meccanismo lato utente per mantenere uno *stato*

Cookie vengono mantenuti sull'*hard disk* dell'utente, e in un *database* del webserver

- **Permettono di associare fra loro più richieste fatte da uno stesso browser o utente**
- Ad ogni connessione, web server controlla in suo database se c'è un cookie per quell'utente >> se non c'è, ne crea uno e lo manda ad utente (insieme a risposta http manda messaggio di *set-cookie*)
- Utente salva su suo *hard disk* cookie >> ad ogni sua azione, cookie la memorizzerà su lato utente e anche su lato server
- Usati per *autorizzazioni, carrelli, raccomandazioni, mantenimento sessioni utente.*
- ***Cookie sono soggetti a molti problemi sulla privacy***

Web Cache

Permette di soddisfare richieste di un client senza contattare il server originario per ottenere le informazioni >> ***web browser mantiene in cache determinati oggetti ->***

- Ad ogni richiesta, si controlla se oggetto è già presente in memoria >> se sì, non viene fatta alcuna richiesta al server.
- In un certo senso, **cache è client rispetto al server >>** *una volta che richiede dati al server, li mantiene in sua memoria, e diventa* ***server nei confronti del client originale.***
- **GET CONDIZIONALE >>** non ha senso mandare richiesta per prima controllare se ho oggetto in cache ogni volta, e poi eventuale richiesta di GET se non ce l'ho -> uso il doppio delle richieste
  - GET condizionale permette di ottenere oggetto dalla cache se presente, sennò fa richiesta al server direttamente.
  - GET CONDIZIONALE inoltre **controlla se il dato contenuto nella cache è aggiornato >>** se non lo è, richiedilo al server orginario.

FTP – *file transfer protocol*

Trasferimento file da <-> verso host remoto.

- FTP CLIENT >> utente comunica tramite interfaccia con un client FTP
- FTP SERVER >> client FTP inoltra file al server FTP

FTP utilizza TCP per stabilire la connessione su port 21, ed inoltra poi i dati su port 20.

- FTP SERVER è dotato di stato >> mantiene la connessione aperta.

Mail Elettronica – SMTP

User Agent, Mail Server e SMTP

- User Agent >> permette la creazione, modifica e lettura di mail.Non è l'interfaccia in sé che usa l'utente per leggere le email, ma è il client che permette di ricevere le mail, che saranno poi interpretate e presentate su un'interfaccia da altre applicazioni.
    - User Agent accede alle mail del Mail Server usando *mail access protocols* (pop3, IMAP, http)
- Mail server >> gestisce mail: composta da due componenti
    - **Mailbox >>** messaggi in arrivo
    - **Message queue >>** messaggi in uscita
    - **Mail server comunicano tra loro utilizzando il protocollo SMTP**
    - Lo scopo di avere un mail server è quello di permettere, anche in assenza dell'host ricevitore, la possibilità di mandare messaggi in ogni momento ad un altro utente. Fornisce inoltre sicurezza (*sistema di autenticazione*)

SMTP – port 25

Trasferimento / ricezione di email sul Mail server.

SMTP usa ASCII =>=> *non deve essere efficiente, DNS invece sì (DNS utilizza codifica in bit)*

Protocollo basato su TCP – usato per comunicazione

- Client => outgoing server (*queue*)
- Relay server => casella postale destinatario
- **SMTP trasferisce informazioni >> protocollo di tipo *push***
- **Oltre ad SMTP c'è anche MIME >> possibilità di inoltrare messaggi con contenuti multimediali** *(testo, immagini, video, applicazioni, audio)*

La comunicazione tramite SMTP avviene con 3 processi => => *handshaking, trasferimento, chiusura.*

Interazione basata sul modello FTP e http => *request / response*, *human readable (ASCII) + status codes*

Funzionamento >>> utente A deve comunicare con utente B >> A compone email, e inserisce destinatario -> messaggio viene mandato al web server di A, che lo manderà a quello di B.

SMTP non prevede meccanismo di autenticazione => => si può integrare con SSL => *richiesta di autenticazione presso il Mail Server.*

SMTP – formato messaggio

- *Header >> To, From, Subject >>> diversi da contenuti header dell'email in sé*
- *Body >> contenuto del messaggio*

Mail Access Protocols – Protocolli di accesso alle *email*

Recupero di email dal Mail Server.

Pop3, IMAP, http

- **POP3 >>** *Post Office protocol >> autorizzazione e download Mail*
- **IMAP >>** *Internet Mail Access protocol >> accesso a mail e manipolazione di messaggi su Mail server*
- **http >>** *gmail, Hotmail, Yahoo…*

POP3 >> lettura dei messaggi sul mail server tramite SMTP >> *richiede prima autenticazione, e poi permette tramite comandi da parte del client di elencare / recuperare / eliminare messaggi dal Mail server*

- ***stateless >>*** *stesso utente che accede da un altro client non avrà la stessa sessione – potrebbe non vedere le stesse email.*

IMAP >> permette di creare una struttura gerarchica all'interno di un mail server, e di mantenerlo in tutte le diverse sessioni e client di uno stesso utente

- ***stateful >>*** mantiene sessioni

Architetture P2P

P2P già accennato – *non c'è server sempre attivo*.

- comunicazione avviene tra utenti finali
- *peers* cambiano costantemente indirizzo IP e sono interconnessi
- *all'aumentare di utenti connessi, il tempo di trasmissione di un FILE aumenta in modo molto minore, poiché si hanno più utenti che potranno a loro volta trasferire altri file.*
    - Al contrario, in un'architettura client-server, il tempo di trasmissione del Server all'aumentare del numero di utenti aumenta in modo direttamente proporzionale al numero di utenti. (molto in fretta e quindi tempi spropositati di attesa)

Es. BitTorrent >> ogni *torrent* è un gruppo di utenti (*peers*) che si scambiano i pezzi del file richiesto. Ogni utente possiede un *tracker* che tiene traccia di quali *peers* possiedono i pezzi del FILE di cui ha bisogno quell'utente.

- Un utente appena entrato in un torrent inizialmente non ha alcun pezzo di file >> li accumula nel tempo, ricevendoli da altri *peers*.
- Gli utenti con il ruolo di *peers* cambiano costantemente.
- Su un arco di tempo predefinito, ogni *peer* scambia con gli altri pezzi di file, possibilmente richiedendo quelli che sono *meno diffusi tra tutti i peer (i più "rari") >>>* quando un peer riceve pezzi di file, manda a sua volta all'altro peer altri pezzi di file.

P2P >> Distributed Hash Table (DHT)

Database p2p distribuito => formato da coppia *chiave – valore*

- Es. Titolo Film => Indirizzo IP
- Coppie chiave-valore assegnate a milioni di *peers >>* quando devo ricercare un file, client manda una *query al DHT cercando la chiave desiderata >>> avrò come valore restituito una lista di indirizzi Ip, che saranno i peers che contengono i pezzi del mio file.*
- Ogni chiave è trasformata in un intero, inserito poi nella tabella di hash distribuita. Distribuita perché quando la chiave viene inserita, viene inserita in modo uniforme all'interno della tabella >> in un *range* di indirizzi, e non uno specifico. La funzione di *hash* fa sì che gli indirizzi non siano assegnati non siano mai gli stessi ma che siano distribuiti uniformemente.

Multimedia

1) Audio

Serve digitalizzare la trasmissione >> campionamento

Digitalizzazione di una frequenza >> *viene prima divisa la frequenza per intervalli - campionamento (n. di intervalli = n. di campioni), e si procede poi con la quantizzazione >> approssimando quindi ogni campione alla grandezza più vicina*

2) Video

Video è una sequenza di immagini mostrate ad un rate costante – vengono codificate come ridondanza di alcune parti di immagini già presenti + eventuali differenze >> *risparmio bit*

Compressioni >

- ***Spaziale >>*** diversi pixel simili tra loro vengono raggruppati
- ***Temporale >>*** *diversi frame rappresentati come la differenza tra l'uno e l'altro*

Tipologie di applicazioni multimedia

- **Streaming audio / video >>>>** *streaming può partire prima che sia conclusa la trasmissione del file per intero; streaming riproduce parte di file mentre viene caricata parte successiva / non ancora caricata*
  - Deve tenere conto del *jitter (variazione dei ritardi di trasmissione)*
  - *Introdotto concetto di **buffering >>** prima di riprodurre file, devono essere stati caricati un tot. Di campioni >> potrò visualizzare i campioni dopo un determinato arco di tempo detto **client playout delay***
- **Streaming live >>** ritardi bassi, throughput appropriato (qualità alta) – tasso di perdita pacchetti va mantenuto basso; si può basare sia su UDP che TCP
  - **UDP >>** miglior throughput e minori ritardi, ma non c'è recupero errori
  - **TCP >>** possibilità recupero errori, ma davvero è peggio di UDP? Possibilità di usare *DASH >> codifica in base capacità client*
- **Conversazione a voce / video tramite IP >>>>** conversazione human-to-human, ritardo tollerabile molto basso

Per lo streaming viene utilizzato UDP >> non serve recuperare errori ma necessità di mantenere ritardi molto bassi e avere una riproduzione del file costante

- Per evitare riproduzione non costante, si introduce un ritardo costante (2-5 secondi) per far fronte anche al *jitter*.

Streaming multimedia >> http

Streaming multimediale può avvenire anche tramite http >>> TCP + HTTP

- Recupero il file tramite richiesta GET, mandando alla velocità massima permessa dal protocollo TCP (controllo congestione && flusso)
- Passa più facilmente tramite *firewalls* al contrario di UDP

Streaming mutimedia >> DASH

Dynamic Adaptive Streaming over HTTP

L'obiettivo è modificare la banda richiesta, *evitando congestioni e timeout*

- **Valutazione da parte del client della qualità della connessione >>** *diminuisco traffico se rete sta per entrare in congestione (ad esempio diminuendo la qualità), diminuendo quindi bitrate >>> questo viene fatto dividendo il file in tanti segmenti, di diversa qualità e codifica.*
  - **Server >>>** segmenta video / file in tanti segmenti di diversa codifica e qualità
  - **Client >>>** misura costantemente livello congestione e flusso del client >>> sceglierà i segmenti opportuni per la finestra e livello attuale di congestione del client

Come fare streaming di contenuti a milioni di utenti?

- **No singolo server** >> si ha un unico punto critico, non offre un servizio scalabile. Troppa congestione, non tiene conto di possibili distanze eccessive tra utenti e punto di accesso al servizio.
- **Server CDN >>** più server distribuiti, più vicini ad utenti e con congestione scaricata su più punti.

VoIP – Voice over IP

Riduzione dei costi (Skype), chiamate gratuite per utenti >> *necessità di bassi ritardi, mantenere l'aspetto conversazionale*

- **Ritardo accettabile >>** sotto 150ms >> ritardo di pacchettizzazione + distanza trasmissione + playout buffer (*buffer con pacchetti da riprodurre*)
- Tenere conto di eventuale perdita di pacchetti dovuta da ***network loss / delay loss***
  - ***Network loss >>*** perdita causata dalla rete, causa congestione (*router buffer overflow*)
  - ***Delay loss >>*** perdita del pacchetto poiché arrivato troppo in ritardo >> non ha senso riprodurre un pacchetto audio se ormai è passato troppo tempo, causa solo ulteriore disturbo
- Perdita pacchetti >> 1% < x < 10%

Es. Skype >> struttura p2p con aggiunta di *supernodi* centrali con aggiunta di autenticazione >> supernodi fanno da server di appoggio per gli utenti (*peers*) >> protocollo proprietario, non si sa la struttura e il funzionamento vero e proprio ma è dedotto tramite reverse-engineering

RTP – Real Time Protocol

Real time transport / Transmission protocol >> protocollo a livello applicativo, *utilizza UDP* >> usato per trasporto stream multimediale (video / audio)

- Specifica infatti la struttura di ogni pacchetto video / audio, fornendo *tipologia payload – numerazione in sequenza dei pacchetti – tempo di invio dei pacchetti*
- *Opera direttamente sugli host finali, non ha bisogno di un server intermedio e non è rilevata dai router intermedi >>* lama a doppio taglio, router non possono assicurarsi che informazioni arrivino correttamente a destinazione
- **RTP utilizza codifica binaria >> maggiore efficienza, non human-readable**
- **NON FORNISCE GARANZIE SUI TEMPI DI TRASMISSIONE IN INTERNET**

Struttura messaggi RTP >>>

- Header >
  - Payload type >> tipo di file contenuto nel messaggio
  - Sequence # >> numero di sequenza del pacchetto
  - Time stamp >> contiene valore del primo byte contenuto nel pacchetto
  - SSRC >> identifica la sorgente del pacchetto RTC

SIP – Session Initiation protocol

Protocollo che si pone come obiettivo il trasformare il sistema analogico (SS7), rimpiazzandolo con Internet >> *oggi è usato per tutti gli operatori telefonici*

- *Mi permette di chiamare o raggiungere un utente ovunque si trovi, indipendentemente dal dispositivo IP che sto usando >>* ***Roaming Utenti,*** *raggiungibili ovunque*
- Non più numero di telefono ma email + nome >> necessario servizio di indexing
- ***KISS*** *principle => Keep It Simple Stupid*

Funzionamento >> *chi chiama rende noto dall'altra parte che si vuole avviare una chiamata -> ricevente e chiamante decidono il tipo di media su cui si baserà la comunicazione – (posso anche aggiungere membri durante la chiamata, cambiare il tipo di trasmissione e mettere chiamate in pausa)*

- *SIP mantiene funzionalità di chiamate normali analogiche, aggiungendone però nuove*
- ***Come fa un utente però a trovare l'indirizzo IP di un altro utente, sapendo solo il suo nome / email? >>*** *Protocollo DHCP, applicato ad un server intermedio* ***SIP SERVER***
- *Trasmettitore contatta infatti il SIP server e viene registrato ad esso tramite il comando* ***SIP REGISTER >>*** da quel momento è reperibile da tutti gli altri utenti

SIP PROXY

Il SIP PROXY ha lo stesso funzionamento dell'*outgoing Mail Server* >> nessuno vieta agli utenti di comunicare direttamente l'uno con l'altro; SIP SERVER comunica con SIP SERVER dell'altro utente, facilitando la connessione tra i due (*o addirittura rendendola possibile laddove i due utenti non ne sarebbero stati in grado da soli)*

- **SIP Server del trasmettitore ricava il SIP server del ricevente\ utilizzando il DNS**
- **I SIP Server hanno l'unico scopo di mettere in comunicazione i due utenti:** quando questi sono messi in contatto, i SIP Server / Proxy non intervengono più nella comunicazione e rimangono solo i due utenti. (*motivo per cui è detta comunicazione trapezioidale – prima utente -> SIP server -> Sip server -> utente, e poi solo più utente <-> utente)*
- **SIP ha nascita nell'ambiente Web**

Altri protocolli di segnalazione per reti digitali > H.323

Sempre usato per la segnalazione di comunicazioni multimediali in tempo reale; permette la segnalazione, registrazione, trasporto e codifica di messaggi multimediali.

- **H.323 ha origine dalla telefonia – operatori TELECOM**

RIASSUNTO SUL SUPPORTO MULTIMEDIALE DELLA RETE

Tre diversi possibili approcci

1) ***Non fare nulla di diverso rispetto normale comunicazione >>*** datagram trattati tutti ugualmente, secondo l'ordine di arrivo. Traffico non ha priorità – nessuna garanzia (solo DASH per prevenzione errori) >> molto semplice e impiegabile ovunque.
2) ***Servizi differenziati >>*** diverse tipologie di traffico permesse, per ognuna delle quali viene offerta una diversa qualità di servizio > nel caso di molto traffico non sono comunque impiegabili ovunque
3) ***Reservation Protocol >>*** ogni volta che voglio aprire una connessione tra A e B, mando una *reservation* ai router intermedi, che terranno traccia di quanta connessione useranno i due host. Questo rende quindi possibile una comunicazione ottimale dra i due host, ma taglia fuori altri eventuali host che avrebbero bisogno dello stesso servizio (*potrei non avere più risorse disponibili sul canale*)