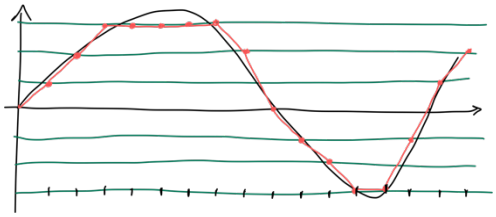
L: numero di bit
R: velocità in bit/s
tprop: tempo di propagazione lungo il mezzo trasmissivo = d/c
d: lunghezza collegamento
c: velocità propagazione sul mezzo trasmissivo
delay minimo = tprop + L/R
latenza: tempo di reazione del sistema
Compressione: ridurre le dimensioni dell'informazione attraverso tecniche lossless (zip) o lossy (jpeg)
Rapporto di compressione Rc=Bitoriginale/Bitcompresso



Digitalizzazione dei segnali analogici attraverso il teorema del campionamento: vedo il segnale in alcuni istanti e codifico l'ampiezza scegliendo dei possibili intervalli. Più è alta la frequenza dei pallini rossi e dei possibili valori verdi, più la curva digitale sarà simile a quella originale. Il **BitRate** sarà la quantità di bit prodotti in unità di tempo. BitRate = bit/sample * sample/second (sample è campione). Per **campionare** fedelmente un segnale audio serve una frequenza di camionamento fc >= 2*Ws dove Ws è la banda del segnale (ovvero il range di frequenza che possono essere emesse). Per la voce umana abbiamo Ws = 4kHz. In telefonia si usano 8000 sample/sec e 8 bit/sample ottenendo un BitRate pari a 64kbit/s che può poi essere ulteriormente compresso nella telefonia mobile. Per i segnali video abbiamo una frequenza di riproduzione in frame/secondo per cui il BitRate = M bit/pixel * WxH pixel/frame * F frame/second.

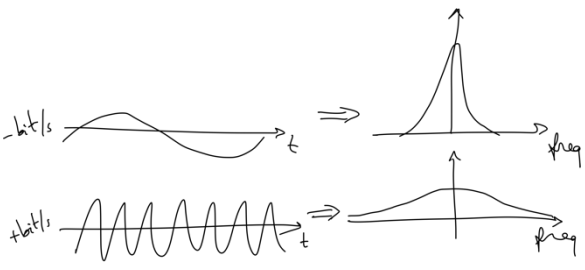
Abbiamo informazioni stream con constant BitRate (CBR) o variable BitRate (VBR). Una sorgente on-off è un tipo ci VBR che trasmette zero o max, si può calcolare il suo Rmedio = Rmax * (Ton/Ton+Toff). I canali di comunicazione possono essere cavi di rame, onde radio, fibre ottiche, … Il segnale può essere analogico (continuo) o digitale (varia fra alcuni valori predefiniti). Bisogna sempre tenere conto di 4 caratteristiche:

- **Attenuazione del segnale**: la sua ampiezza diminuisce, si usavano ripetitori per comunicazioni analogiche e rigeneratori per comunicazioni digitali
- **Distorsione del segnale**
- **Rumore additivo**: si somma al segnale esistente, si genera nel mezzo trasmissivo
- **Interferenza**: si somma al segnale esistente, proviene dall'esterno

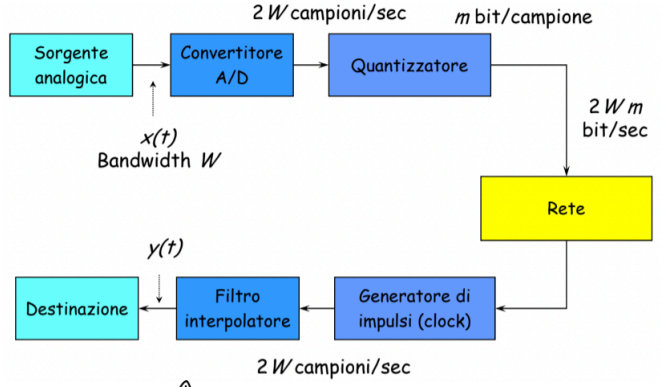
È più semplice ricostruire i segnali digitali che quindi sono preferiti.
Bisogna trovare soluzioni alle seguenti domande:

- Come aumento il BitRate?
- Come ottengo un trasferimento affidabile?
- Ci sono limiti al BitRate e all'affidabilità della trasmissione?

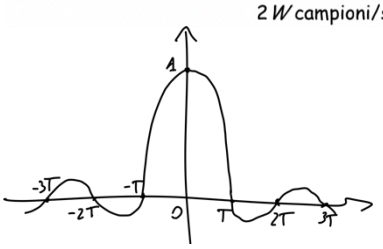
Ci sono dei canali che fanno passare solo alcune bande: canale passa alto (frequenza minima), passa basso (frequenza massima) e passa banda (banda definita Wc). Non posso aumentare troppo il bitrate perché allargando ogni impulso rischio che si mischino (interferenza intersimbolica). In un canale limitato in banda Wc posso far passare al massimo 2*Wc impulsi per il **teorema di Nyquist**. Posso aumentare ancora il BitRate sfruttando la **trasmissione multilivello**, ovvero aumentando il numero di bit che passo in ogni impulso definendo più livelli di ampiezze. Con M = 2^m livelli posso avere un BitRate = 2*m*Wc bit/s. In assenza di rumore posso aumentare i livelli del segnale riducendo però la distanza fra livelli adiacenti. Da m dipende l'errore di quantizzazione.



Larghezza di banda misura quanto velocemente varia un segnale: la banda aumenta tanto più varia il segnale. La rappresentazione di segnali in serie di Fourier ci permette di analizzare il segnale in frequenza.
SNR: **signal to noise ratio** = Potenza media del segnale / Potenza media del rumore. Più è elevato m, più è alto il rapporto segnale rumore e più sarà preciso il segnale ricostruito (perché ho tanti livelli e quindi il segnale sarà preciso).



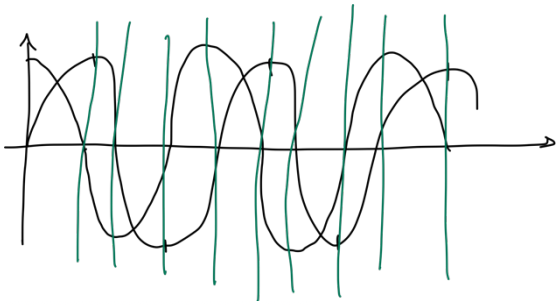
Esempio: segnale con banda W=4kHz, fc=2W=8000 campioni/secondo, cerchiamo un errore massimo dell'1% per cui SNR = 10log(1/0.01)^2=40dB. Considerando che SNR(dB) = 6m – 7.27 dB, abbiamo quindi 40dB <= 6m – 7 e quindi m >= 8bit/campione. Un'altra formula è SNR = 3(σx/v)^2 * 2^(2m), dove σx è la potenza del segnale. Un segnale distorto non permette di leggere correttamente il bit passato. **Nyquist** propone un modello con questa funzione s(t)=sen(2*pi*Wc*t)/2*pi*Wc*t. Che si annulla ogni kT con k!=0 per cui campionando agli istanti t=KT ho **interferenza intersimbolica** (ISI) nulla. s(t) è un esempio della classe degli impulsi di Nyquist con ISI nulla. Questo metodo richiede però una sincronizzazione molto accurata. I trasmettitori digitali creano l'onda e la moltiplicanzo per +1 o -1 in base al bit da trasmettere. Posso sempre aumentare il BitRate moltiplicando per +1 +1/3 -1/3 -1 per trasmettere rispettivamente 00 01 10 11 e avrò quindi raddoppiato il BitRate mandando due bit in ogni impulso.



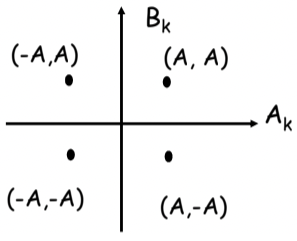
Ho sempre M livelli = 2^m bit che mi danno R BitRate = 2*Wc*m. I livelli rischiano però di confondersi se sono molti a causa del rumore. Si ha quindi la **capacità limite di Shannon** C = W*log2(1 + S/N) bit/s dove S/N è il rapporto segnale/rumore. Il BER bit error rate è la percentuale di bit riconosciuti in modo errato. Esempio: W=3kHz, M=8, m=3, SNR=20dB. [dB=10log10(x) e quindi x=10^(dB/10)]. R=2*3000*3=18kbit/s. C = 3000log2(1+10^(20/10)) = 19,96 kbit/s. Poiché la capacità limite di Shannon del mio canale è maggiore del BitRate di cui ho bisogno non avrò problemi.

In una stessa banda posso avere più canali ognuno centrato in una frequenza fc. Questo schema è noto come **Multiplazione a Divisione di Frequenza**. Ho una frequenza portante fc e il canale passa banda è centrato intorno alla frequenza fc.

- **Modulatore ASK**: per trasmettere una sequenza di bit con una sinusoide trasmetto per mandare 1 e non trasmetto per mandare 0, ovvero basato su ampiezza
- **Modulatore FSK**: modulo il segnale in base alla frequenza per trasmettere 1 con frequenza alta e 0 con frequenza bassa



- **Modulatore PSK**: cambio la fase per trasmettere 1 o 0. In questo caso modulo la sequenza di bit Ak e ogni T mando yi(t)=Ak*cos(2*pi*fc*t). In ricezione demodulo con un filtro passa basso per levare alcune frequenze e far rimanere solo Ak. Con questa modulazione perdo metà impulsi/sec ma posso risolvere con la **Quadrature Amplitude Modulation** (QAM): essendo i segnali ortogonali possiamo distinguerli dopo averli sommati perché sappiamo che in alcuni specifici punti uno dei due è zero. Per farlo devono naturalmente essere sincroni, con stessa frequenza e periodo. Così con due segnali Ak e Bk posso passare il doppio dei bit. Aumentando i livelli posso sempre trasmettere più bit creando varie costellazioni ma rischio di fraintendere i bit passati se ho molto rumore. In questo caso devo ridurre le modulazioni possibili. Il BER si alza se il rapporto segnale/rumore (SNR) è basso e se aumento i livelli.

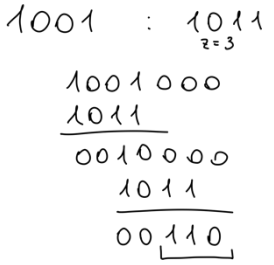


Lo **strato di collegamento** (data link) è sopra lo strato fisico. Host e router sono nodi, i canali di comunicazione che collegano nodi adiacenti sono i collegamenti (link). Con il **Framing** incapsulo i bit del livello di rete in un frame. I **protocolli MAC** fanno riferimento al livello 2 e si utilizzano nei link multiaccesso per distinguere i dispositivi. È molto importante la rivelazione e correzione degli errori causati dal transito nel mezzo trasmissivo. Avviene grazie all'inserimento di bit di controllo. In altri casi posso addirittura perdere dei frame o riceverli in ordine sbagliato. Il livello di collegamento garantisce anche un controllo di flusso. La ritrasmissione in caso di perdita o errore può essere effettuata anche dallo strato di trasporto ma verrebbe rinviato il pacchetto per l'intero percorso con spreco di pezzi di rete, in compenso però non dovrei implementare i

controlli su ogni nodo. I link possono essere half-duplex o full-duplex, nel primo caso la comunicazione avviene sullo stesso canale con versi alternati, nel secondo caso invece ho due canali che possono funzionare contemporaneamente nei due versi. Il mittente **incapsula** i bit in un frame che contiene anche altri dati fra cui bit di rivelazione errori, per il trasferimento affidabile, il controllo di flusso, … Il ricevente attraverso **l'header** può svolgere tutte le funzioni, estrarre i bit di informazione e utilizzarli. L'interfaccia fra due strati della pila OSI è il SAP (service access point). All'inizio e alla fine di ogni frame viene aggiunta una sequenza fissa di bit (il flag) che serve a distinguere i singoli frame. Un possibile esempio è l'uso di 01111110 come **flag**. Per evitare simulazioni all'interno della sequenza di bit si fa **bit stuffing** (in emissione aggiungo uno zero ogni volta dopo cinque 1) e **bit destuffing** (in ricezione controllo bit dopo cinque 1, se è un 1 allora ho il flag, se è 0 allora lo rimuovo). Nel protocollo PPP (point to point protocolo) si usa **byte stuffing** (premetto byte 01111101 a byte uguale a questo o al flag) e **byte destuffing** (se ho due byte 01111101 elimino uno dei due, se ho un byte come quello e poi un byte come il flag elimino il primo, se ho il flag senza quel byte davanti allora sono sicuro che sia il flag). Rischio comunque un errore di trasmissione nel flag che farebbe perdere l'intero frame. Potrei anche mettere dopo il flag di apertura la lunghezza in byte del frame che sto inviando al posto di usare un flag di chiusura. Per il **controllo di errore** posso usare due approcci: ARQ (**Error detection and retransmission**) o FEC (**Forward error connection**). All'informazione dell'utente viene sostituita dal codificatore una **codeword** che sarà poi analizzata dopo la trasmissione sul canale. La lunghezza della codeword è n bit, del messaggio da proteggere è k bit, per cui ho n-k bit di controllo. Ho bisogno che gli errori di trasmissione non facciano sembrare i dati ricevuti una sequenza corretta. La distanza fra le codeword deve quindi essere alta.

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

- **Controllo di parità singola**: aggiungo un bit di parità che mi permette di rilevare ogni configurazione che modifica un numero dispari di bit.
- **Controllo di parità bidimensionale**: struttura i bit informativi su più righe e calcola bit di parità di ogni riga e colonna. Rileva sempre configurazioni con 1, 2 e 3 errori, con 1 errore può anche individuarne la posizione e correggerlo. Ha troppi bit di controllo (overhead).
- **Internet checksum** si usa in TCP IP UDP, viene ricalcolato in ogni router. Si considerano L parole da 16 bit come interi che vengono sommati = x, il checksum è dato da x modulo(2^16 -1) e sarà quindi una parola da 16 bit. Esempio con 4 bit: in binario li sommo e avrò un risultato di k bit, sommo agli ultimi 4 bit i primi k-4 bit (saranno il riporto) e ottengo un risultato di 4 bit, li inverto tutti e trovo il checksum.
- **CRC – codici polinomiali a ridondanza ciclica**. Considera le k cifre binarie come coefficienti di un polinomio P(x) di grado k-1 e sceglie un polinomio G(x) generatore comune a emittente e ricevente di grado z. Si fa La divisione fra x^z * P(x) e G(x) e si usa il resto come CRC che avrà grado <= z. Aggiungo alla fine delle k cifre binarie da trasferire i z-1 bit del CRC e ottengo la codeword. A destinazione, divisa per G(x), dovrà dare resto zero. Errori a burst sono una serie di bit giusti e sbagliati che inizia con un errore e finisce con un errore e ha una certa lunghezza. Siamo in grado di rivelare errori singoli, doppi, isolati con molteplicità dispari e a burst di lunghezza <= z. Se l'errore E(x) (ovvero stringa di 1 dove c'è errore e 0 altrimenti) è multiplo di G(x) non potrò rilevare l'errore. Una sequenza è multiplo di G(x) se è uguale ma con degli zeri a destra.



- **Codici di Hamming**: la distanza d di un codice è il minimo numero di bit di cui differiranno due parole qualsiasi, se è dispari si possono rilevare e correggere d/2 errori, se è pari si possono rilevare d/2 errori e correggere d/2-1 errori. Codice di Hamming (7,4) usa 3 bit di parità per 4 bit di informazione e garantisce distanza d=3. Chiamo d1 d2 d3 d4 i bit da proteggere e p1 p2 p3 i bit di parità. Li ordino così p1 p2 d1 p3 d2 d3 d4 e calcolo p1 come bit di parità fra 1 3 5 7 bit, p2 come bit di parità fra 2 3 6 7 bit, p3 come bit di parità fra 4 5 6 7 bit. Controllo bit di parità p1 p2 p3 e in base alla tabella so dove sta l'errore. Esempio 1011110 ha errore in p1 e p3 e quindi l'errore sta in d2. Se ho due errori non riesco ad individuarli.

Medium Access Control: non trattiamo più di collegamenti punto-punto ma di collegamenti broadcast. I segnali possono però mischiarsi e generare collisioni per cui implementiamo protocolli per evitarlo. Se ho un canale di capacità R bit/s, gli M nodi che devono inviare dati avranno una banda di R/M bit/s. Consideriamo che non ci siano nodi master né sincronizzazione dei clock (protocollo decentralizzato). I protocolli a **suddivisione del canale** possono operare su diversi fattori: tempo, frequenza, … Ogni nodo ha l'utilizzo

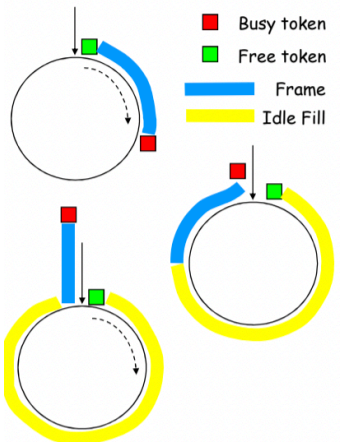
esclusivo della sua parte di canale ma sprechiamo risorse nel caso in cui un canale sia assegnato e non utilizzato. Esistono per questo i protocolli ad accesso dinamico che si dividono in "**accesso casuale**" (canali non divisi, possono esserci collisioni e in questi casi i nodi ritrasmettono il frame) e "**accesso controllato**" (ogni nodo ha un suo turno che può essere più lungo se ha molto da trasmettere). Si chiama **prodotto banda ritardo** PBR la "lunghezza del canale in bit", ovvero il numero massimo di bit presenti contemporaneamente sul canale, si calcola R banda * tpropagazione e si misura in bit.
Rete a bus: ogni nodo trasmette quando vuole ma servono strategie da attuare in caso di collisione. Si può suddividere il canale in turni temporali (TDMA) o in frequenze (FDMA) o entrambi. Altrimenti **accesso casuale**:

- **MAC con due nodi**: nodo A inizia a trasmettere e dopo tprop B se ne accorgerà, se B inizia a trasmettere prima di tprop si accorgerà a un certo punto della collisione e anche A dopo tprop dall'inizio della comunicazione di B si accorgerà della collisione. Quindi B non potrà trasmettere da tprop prima dell'inizio di A a tprop dopo e quindi l'intervallo di vulnerabilità è 2*tprop. Consideriamo L la lunghezza del frame, l'efficienza ρmax = 1/(2+a) e il throughput massimo Reff = ρ*R, conoscendo il prodotto banda ritardo normalizzato a = tprop/(L/R) = PBR/L. Il tempo di trasmissione di una frame è L/R.
- **Aloha**: il nodo trasmette quando vuole, se non riceve l'ACK entro 2*tprop dalla fine dell'invio a causa di una collisione allora calcola il tempo di ritrasmissione (backoff time B) e al suo scadere ritrasmette. B aumenta tanti più sono i tentativi falliti di invio. L'efficienza massima è 18,4%.
- **Slotted Aloha**: divido in intervalli di tempo sincroni fra gli host, le unità informative devono avere lunghezze costanti in base agli slot temporali per cui dovrò spezzare i frame e aggiungere bit di overhead. L'intervallo di vulnerabilità diventa solo il periodo di slot. L'efficienza massima è 36%. È spesso utilizzato come meccanismo di contesa in vari protocolli MAC per prenotare un posto nel mezzo trasmissivo. Così abbiamo efficienza bassa ma solo per piccoli slot di tempo mentre per il resto trasmetto senza problemi.
- **CSMA Carrier Sensitive Multiple Access**: prima di trasmettere ascolto il canale e trasmetto se nessuno lo usa, aspetto se qualcuno lo sta usando. L'intervallo di vulnerabilità è sempre 2*tprop. Se a è maggiore di 1 non ho guadagno rispetto ad Aloha e Slotted Aloha, ovvero se tprop > L/R tempo di cui ho bisogno per trasmettere il mio frame. Ci sono diversi algoritmi di persistenza:
    - 1-persistent: trasmetto appena il canale si libera, basso ritardo e efficienza, ok con pochi utenti
    - Non-persistent: applico backoff e ricontrollo, alto ritardo e alta efficienza, ok con molti utenti
    - P-persistent: quando si libera trasmette con probabilità p e attende un piccolo tempo prima di ricontrollare con probabilità p-1, ritardo ed efficienza possono essere modulati

  Se parlo per più di tprop sto tranquillo che nessuno mi parlerà sopra perché ho acquisito il canale. L'efficienza di CSMA peggiora al crescere della velocità del mezzo.
- **CSMA/CD with collision detection**: ascolta anche mentre sta parlando, in caso di collisione interrompe la trasmissione e rischedula la trasmissione dopo un tempo di backoff. Riduce le durate delle collisioni e quindi aumenta l'efficienza. Se B si accorge di collisione continuerà a mandare bit a caso per far capire ad a che c'è stata una collisione.
- Il protocollo **ethernet** è basato sul CSMA/CD, 1-persistent CSMA, R=10Mbit/s, tprop=51.2μs ovvero 512 bit ovvero 64 byte slot, distanza massima 2.5Km + 4 repeaters. Utilizza il Truncated Binary Exponential Backoff: dopo l'n-esima collisione il tempo di backoff è scelto fra {0, 1, …, 2^(k-1)} dove k=min(n,10).

Protocolli ad **accesso controllato**:

- **Polling**: un nodo master gestisce gli utenti ma c'è una fase di richiesta che aggiunge ritardo. Elimina collisioni e slot vuoti.
- **Token-passing**: generalmente con configurazione a ring. Un messaggio di controllo circola fra i nodi seguendo un ordine prefissato e dà il diritto a trasmettere. È decentrato ed altamente efficiente ma il guasto ad un nodo può mettere tutto fuori uso. Un flag inserito nei frame può essere il token. Un nodo che deve trasmettere cattura il token free mettendo busy e inizia lui a trasmettere, al termine manda un frame con il flag free. Posso avere più comunicazioni contemporaneamente. La ring latency è il numero di bit che possono essere trasmessi simultaneamente sul ring. τ = tempo impiegato da un bit per circolare nel ring, T = tempo di trasmissione di un frame, a = τ/T ring latency normalizzata. Esistono tre tipi:



○ **Multi-token operation**: il free token è tramesso dopo l'ultimo bit di un frame. ρmax=1/(1+a/M)
○ **Single-token operation**: il free token è inserito dopo che l'ultimo bit del busy token è tornato al nodo origine, il tempo di trasmissione è almeno uguale alla ring latency. ρmax=1/(a/M + max(1,a))
○ **Single-frame operation**: il free token è emesso dopo che il nodo emittente ha ricevuto l'ultimo bit del suo frame. Aggiunge un trailer uguale alla ring latency al frame. ρmax=1/(1 + a(1 + 1/M))
con a molto minore di 1 è accettabile ogni strategia di reinserimento del token, con a circa 1 è accettabile la single token operation, con a maggiore di 1 è necessaria la modalità multi token operation.

Gestione d'errore: **ARQ** (automatic repeat request), non provo a recuperare l'errore ma chiedo un reinvio. È utilizzato anche nel protocollo TCP ma end-to-end e non point-to-point. Gli elementi chiave sono codici di rivelazione dell'errore, ACK e NACK, Timeout. Il piggybacking è l'inserimento di informazioni aggiuntive nell'unità informativa inviata, utili ai fini dei diversi protocolli.

- **Stop and wait**: l'unità informativa è integrata da codici di controllo, il ricevitore manda Ack o Nack e il sender la aspetta prima di inviare l'unità informativa successiva (con timeout per eventuali perdite). Si può usare un solo bit per l'ack che assume i valori 0 e 1 alternati. Se non viene mantenuta questa alternanza è implicato un nack. Chiamiamo Slast il numero di sequenza del frame e Rnext il numero di sequenza dell'ack (indica il prossimo Slast che voglio). Il timeout deve essere corretto altrimenti rimando troppo spesso i frame o attendo troppo in caso di perdita. Deve essere sicuramente maggiore di t0=2tprop+2tproc+tf+tack dove tproc è il tempo di processamento in ogni nodo, tf e tack sono i tempi di trasmissione dei due frame di dati e di ack. L'efficienza è inversamente proporzionale al PBR.
- Il **Go-back-N** è un protocollo a finestra, trasmetto frame consecutivi e mi fermo solo se non ho ricevuto gli ultimi n ack. Elimina le attese dei riscontri. I frame fuori sequenza sono scartati e il sender dopo un timeout rimanda tutte i frame partendo da quella che non ha ricevuto l'ack. Con una finestra di trasmissione abbastanza grande posso arrivare ad ottenere un throughput pari ad 1. La numerazione di frame ed ack può essere ciclica, dovrò avere i possibili valori di numerazioni = dimensione finestra di trasmissione +1. La finestra di trasmissione dovrà essere abbastanza grande da poter mantenere il canale occupato per tutto il periodo di timeout (Timeout/Tframe arrotondato per eccesso).
- Il **Selective Repeat** ritrasmette solo i frame persi e non tutti, c'è una finestra in ricezione che indica i numeri di sequenza che possono essere accettati (anche se non in ordine). Se non ricevo il frame 2 manderò nack 2 e poi continuerò a mandare ack 2 per ogni frame finché non ricevo il frame 2. A quel punto avrò già ricevuto ad esempio 3, 4, 5 e quindi manderò ack 6.

Selective-Repeat

$$\eta_{SR} = (1-P_f)(1-\frac{n_o}{n_f}) \approx (1-P_f)$$

Go-Back-N

$$\eta_{GBN} = \frac{1-P_f}{1+(W_S-1)P_f} = \frac{1-P_f}{1+LP_f}$$

Stop-and-Wait

$$\eta_{SW} = \frac{(1-P_f)}{1+\frac{n_a}{n_f}+\frac{2(t_{prop}+t_{proc})R}{n_f}} \approx \frac{1-P_f}{1+L}$$

Indirizzi MAC identificano un host in un'area locale e sono usati per l'indirizzamento locale (IP per globale). Vengono utilizzati a livello di collegamento. Sono indirizzi unici da 6 gruppi di coppie di esadecimali. Una parte dell'indirizzo è il marchio del produttore e l'altra parte è gestita dal produttore. Il protocollo ARP (address resolution protocol) serve ad associare indirizzi MAC ed indirizzi IP locali. Si avvale di una tabella ARP che contiene la corrispondenza fra indirizzi IP e MAC. Se ARP non ha l'indirizzo MAC a cui deve inviare un frame farà una richiesta broadcast a cui il destinatario risponderà mandando il suo MAC.
Ethernet ha varie configurazioni: a **bus** (un cavo con host interconnessi) o a **stella** (switched ethernet, al centro della stella c'è uno switch che lavora comunque in broadcast). È basato su CSMA/CD con lo slot time (51.2 μs) come parametro principale. La scheda di rete incapsula i pacchetti IP aggiungendo nell'header un preambolo di 7 byte (10101010 per attivare la scheda di rete e sincronizzare i clock con il trasmettitore), uno start delimiter di 1 byte (10101011), destination e source address di 6 byte ciascuno, length, data, PAD per riempire se l'unità informativa risulta minore di 64 byte, CRC di 4 byte. Alla ricezione conservo il frame ethernet se il destination address coincide con il mio MAC address. È un protocollo senza connessione e non

affidabile. Il segnale di disturbo JAM è di 48 bit e si invia appena viene rilevata una collisione per avvertire gli altri. Crescendo la velocità dovuta all'evoluzione del mezzo fisico, la lunghezza del collegamento deve decrescere per permettere la rilevazione di collisioni prima che scada il tprop. Per ovviare a questo problema si è passati al **Fast Ethernet** che ha velocità 100Mbit/s e poi al **Gigabit Ethernet** con velocità 1Gbit/s. Nella versione iniziale gli host erano interconnessi su un bus, successivamente si è passati alla tipologia a stella con **hub** che semplicemente propaga su tutti quello che riceve da uno. Sono poi stati inseriti gli **Switch** (o Bridge) che non operano a livello fisico (come gli hub) ma a livello di link filtrando e inoltrando le frame ethernet all'interfaccia corretta in base all'indirizzo MAC. Ogni link fra switch e host è un dominio di collisione separato dagli altri, si può evitare con collegamento full-duplex. Gli switch si autoconfigurano senza un protocollo. La **tabella di switch** ha tre campi: indirizzo MAC di destinazione, interfaccia per arrivarci e time stamp (per riconoscere le modifiche più recenti alla tabella). Viene riempita automaticamente salvando l'indirizzo MAC appena riceve qualcosa da un host, se non ha l'indirizzo MAC in memoria rilancia il frame su tutte le interfacce. Gli switch possono essere interconnessi e sono più rapidi dei router perché attraversano uno strato in meno della pila ISO/OSI. Per evitare cicli infiniti nelle interconnessioni fra LAN tramite switch si trasforma il grafo che rappresenta la struttura della rete in un albero escludendo alcuni collegamenti. Se ne occupa il **protocollo Spanning Tree** che in ogni grafo sceglie un Root Bridge come radice (ad esempio quello con MAC minore), successivamente ogni switch trova le interfaccie che lo collegano al root bridge e se sono più di una identifica come root port quella che ci arriva con meno passaggi. La designated port è quella che permette di raggiungere una LAN, se è connessa da più switch questi si accorderanno per tenere aperta solo un'interfaccia. Alla fine si escludono le porte diverse dalle root e dalle designated. Se ci fossero problemi si potrebbe immediatamente cambiare le porte attive per aggirarli. Questo protocollo funziona attraverso lo scambio di BPDU bridge protocolo data unit che contengono Root ID, Switch ID e Root Path Cost.