RETI LOCALI (LAN)

Servizi di connessione contingentati verso un'area locale, cioè geograficamente non estesa

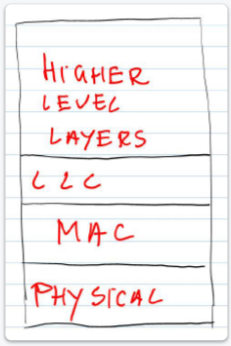
- Pensato decine di anni fa
- Vedi reti ring, bus, star, etc... (introduzione corso)
- Specifica livelli bassi (1 e 2)
- Si suppone che i dispositivi interni alla rete abbiano a disposizione *mezzi trasmissivi di qualità*
  - Infatti l controllo dell'integrità (che nella struttura OSI si effettua a livello collegamento su base link-to-link), qui viene inserita agli estremi (edge - base end-to-end) ed è gestita dall'LLC (vedi dopo)
- Standard riferimento delle reti locali: IEEE 802, in particolare diremo

$$\text{IEE } 802.xxx \quad , \quad xxx \rightarrow \text{servizio da specificare}$$

  - Ad esempio: $\text{IEE } 802.2 \quad \rightarrow \quad$ standardizzazione livello LLC (comune a tutte le reti locali)

Architettura Protocollare 802

Fatta a strati (livelli)



Higher Layer → non sono specificati (indica che lì ci saranno i livelli più alti della pila TCP)
LLC → Logical Link Control
MAC → Medium Access Control
Fisico → livello più basso

Un altro modo di vederlo è il seguente (fonte: Wikipedia):

Modello OSI	
7.	Livello di applicazione
6.	Livello di presentazione
5.	Livello di sessione
4.	Livello di trasporto
3.	Livello di rete
2.	Livello di collegamento dati • Sottolivello LLC • Sottolivello **MAC**
1.	Livello fisico

LLC (LOGICAL LINK CONTROL)

Strato comune a tutti gli standard IEEE introdotti (a differenza del MAC che è caratteristico delle particolari reti LAN che esamineremo)
Ha il compito di verificare su base *end-to-end* *l'integrità dei dati* ricevuti

- Non crea problemi come concetto se stiamo utilizzando mezzi "di fiducia", di qualità (come supposto all'inizio)

Definisce uno standard *unico* per tutte le reti LAN della famiglia IEEE 802

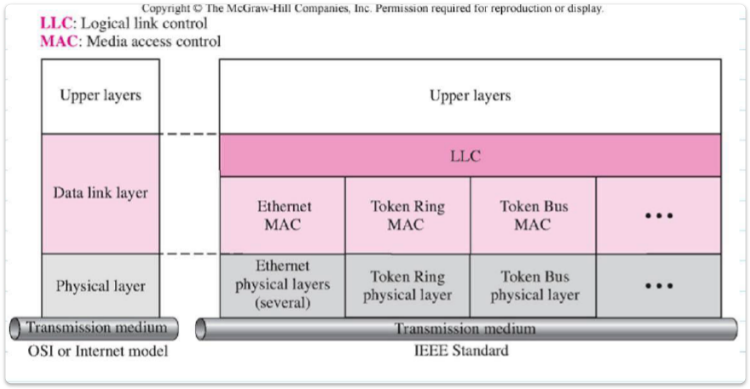
- cioè va bene per tutte le tipologie di reti (LAN)

Gestisce/implementa le seguenti modalità:

- Connection less non affidabile (senza riscontro)
  - Modalità detta logical data link (collegamento logico)
- Connection less affidabile (con riscontro)
  - Modalità detta logical data link alternativo (collegamento logico alternativo)
  - In caso di errori si richiede il re-invio
- Connection Oriented
  - Modalità detta data link connection (collegamento effettivo e affidabile)

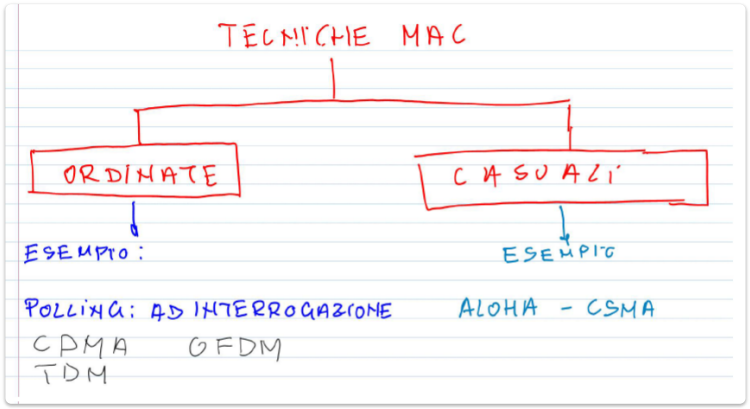
CONFRONTO OSI vs IEEE

- Si noti come LLC sia lo stesso
- MAC invece specifica a seconda della topologia della rete (ethernet, ring, bus...) e del tipo di collegamento (Ethernet, Token Ring, Token Bus...)



MAC

Livello con la caratteristica di avere la tecnica di *condividere l'accesso a uno stesso mezzo fisico* a più utenti (in istanti diversi) utilizzando una tecnica specifica.
Esistono due grandi famiglie di queste tecniche:



- **Casuali**: il primo utente che accede al dispositivo ne diventa il possessore, senza preoccuparsi degli altri utenti che potenzialmente potrebbero accederci. Possibili collisioni (pensiamo se due utenti fanno richiesta contemporaneamente a una risorsa, o se un utente chiede l'accesso a una risorsa già occupata) [i nodi non collaborano tra loro, sono indipendenti]
  - Tecniche di accesso casuale (cfr. approfondimenti dopo):
    - ALOHA
    - CSMA (usato in reti Ethernet)
- **Ordinate**: si stabilisce una regola che vale per gli utenti autorizzati ad accedere al servizio/dispositivo (esempio: schedulare una tabella per gestire i tempi di accesso circa l'utilizzi di un PC). In questo modo si evitano conflitti (collisioni)
  - Tecniche di accesso ordinato:
    - **FDMA***, **TDMA** (accesso multiplo a divisione di frequenza [FDMA] o tempo [TDMA]: simile ad esempio all'FDM/TDM, che divideva la banda/tempo_utilizzo di un canale in canali e ciascuno viene assegnato a un utilizzatore)
    - *CDMA* (Code Division Multple Access) → utilizzata oggigiorno soprattutto (più complessa): sfruttano i segnali ortogonali e la loro correlazione $\int_0^t x(t)y(t)\,dt = 0$. APPROFONDIMENTO:Basterà allora assegnare a ogni utente un segnale che rispetta tale proprietà con tutti gli altri. In ricezione soltanto quando si considera l'integrale con la copia esatta del segnale che vogliamo estrarre avremo valore diverso da zero (gli altri saranno nulli). In questo modo abbiamo estratto ciò che volevamo (nota: è un caso ideale)
    - **POLLING**: scheduling ordinato secondo un criterio ben definito (vedi dopo)

TECNICHE ORDINATE

POLLING (interrogazione)

Prevede invio di un *messaggio* con struttura nota che *abilita chi lo riceve* ad accedere al canale

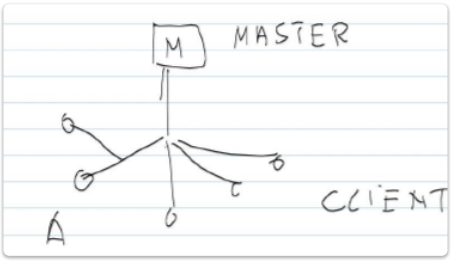
- Per evitare conflitti sull'accesso è necessario prevedere una mobilità di invio del messaggio adatta, che abiliti cioè un solo (specifico) utente del gruppo

Ci sono varie modalità per effettuare ciò, ad esempio:

ROLL-CALL

Tecnica di tipo *centralizzato* - esiste una **entità** che si occupa di coordinare l'accesso di ne fa richiesta in modo ordinato. Letteralmente si traduce con $fare\ l'appello$ e infatti:

- Come quando l'insegnante recita l'appello: ogni utente viene interrogato seguendo un ordine prefissato (alfabetico ad esempio)
- Così fa l'entità centrale: invia il messaggio di autorizzazione al primo user che ha nella lista (del database): l'utente riconosce il messaggio, lo trattiene e accede al canale. Terminata la fase di accesso il terminale dell'utente rilascia l'autorizzazione e lo comunica alla entità centrale rilasciando il messaggio che aveva trattenuto. Si procede quindi con le successive interrogazioni (se un utente non ha informazioni da riferire, rilascia subito il messaggio)

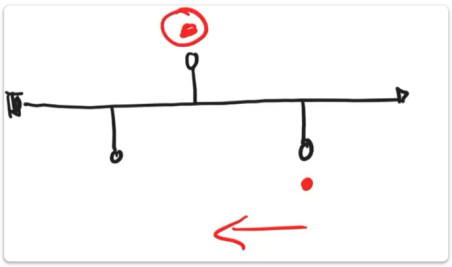


😡: tempo di non utilizzo della rete (tipo quando un utente poi non richiede il servizio quindi rilascia subito il messaggio), delay dei tempi di rilascio e tempo interrogazioni varie
> Tecnica alternativa: esiste e si chiama Hub-Polling (vedi dopo)

HUB-POLLING

Tecnica di tipo *distributivo* e cooperativo. Implementato su una struttura a bus

- L'entità centrale (Master) chiama la stazione più lontana
- La stazione riceve l'autorizzazione e accede, rilasciando il messaggio
- Il messaggio viene rilasciato dalla stazione immediatamente più vicina alla precedente
- Continua il "passaparola" fino a tornare al Master



In questo modo si *riducono al massimo i tempi di latenza*: il messaggio non deve tutte le volte ritornare al Master

MODALITA' (tempi) DI ACCESSO al canale (valide per entrambe le tecniche)

- Come i dispositivi gestiscono i tempi della fase di accesso una volta che è stato autorizzato
  Due modalità:
- **Gated** (limitato): Intervallo di tempo di accesso e quindi numero di pacchetti trasmessi non costante e fisso.
  - Un nodo è autorizzato a trasmettere soltanto quei pacchetti che risiedono nel proprio buffer "di accumulo", che si riempie nell'intervallo che intercorre tra due autorizzazioni successive. Quei pacchetti che giungono sul buffer di accumulo durante l'effettivo accesso (che dura un non predeterminato tempo, a seconda di quanto ci mettono a essere trasmessi) dovranno attendere il "turno" (autorizzazione) successivo per poter essere trasmessi
  - Viene prefissato solo un tempo massimo di accesso, oltre il quale si passa al nodo successivo (indipendentemente dallo svuotamento o meno del buffer di accumulo)
  - Tempo comunque limitato
  😃: accesso più "democratico"
  😡: non conveniente sempre, spesso trasmissione troppo "spezzettata"
- **Esaustivo**: un nodo termina la sua fase di accesso solo quando non ha più pacchetti da trasmettere (svuota cioè tutto il suo buffer di accumulo obbligatoriamente, che si è riempito nel tempo compreso tra due autorizzazioni successive).
  - Vengono trasmessi anche quelli che arrivano durante il tempo di accesso (a differenza del caso precedente)

- Una volta inviato tutto, se poi pian piano si riaccumula il buffer, tali pacchetti saranno (tutti) inviati alla "iterazione" successiva
- Non si introducono limitazioni di tempo/pacchetti

😃: accesso più veloce
😡: rischio di tempo di traffico molto alto - monopolizzazione (altri utenti non contenti)

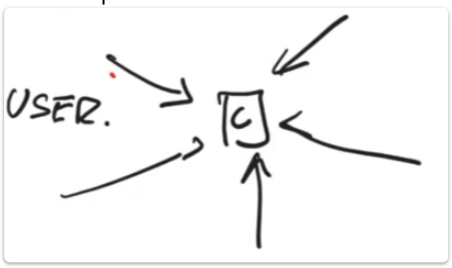
TOKEN PASSING

- Tutti i nodi della rete si passano un messaggio detto $token$ (gettone)
- Implementata su rete di tipo $ring$ (ogni nodo ha un predecessore e un successore)
- Il $token$ autorizza un nodo all'accesso. Terminata la fase, il gettone viene passato all'utente successivo

TECNICHE CASUALI

ALOHA

- Tecnica di accesso concorrente (cioè casuale appunto) su uno stesso punto ($access\ point$)
- Implementata su reti a stella
  - Il centro stella è il suddetto punto di acceso "voluto" da più user

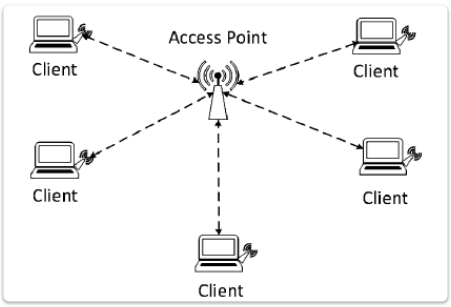


- Il canale utilizzato è quello *wireless* (radio)

ALOHA PURO

Nella modalità $base$ (puro), non prevede alcun coordinamento preventivo tra gli utenti (cioè ognuno accede al canale quando ne ha necessità, senza preoccuparsi degli altri)

- 😡Causa inevitabilmente (possibili) **collisioni**: almeno due segnali generati da utenti diversi sono contemporaneamente presenti nel canale condiviso
  - Il segnale risultante quindi è la somma dei due, pertanto diventa non interpretabile (perché disturbato)
  - Spreco di tempo ed energia



Per evitare il problema delle collisioni, o almeno ridurlo al minimo, dobbiamo chiederci:

- Q: Come si riconoscono le collisioni?
  - A: Si utilizza il **riscontro** dei tentativi. Infatti gli user provano ad accedere utilizzando una certa banda di frequenza. L'access point notifica in modalità broadcast l'esito del tentativo con una banda di frequenza diversa (per non creare sovrapposizioni con l'altra) e notifica tutti i client che hanno richiesto l'accesso ($Nota$: ogni utente conosce a priori quanto tempo deve passare tra la richiesta e il riscontro, perché è ben definito per tutti un certo tempo di $timeout$).
    - Se tutto va bene, allora vuol dire che i client hanno ricevuto il riscontro entro la finestra temporale prevista
    - Se c'è qualche problema, vuol dire che nell'intervallo previsto i client non hanno ricevuto alcun riscontro $\rightarrow$ accesso non avvenuto
- Q: Come si risolvono le collisioni?
  - A: Vedi $(**)$

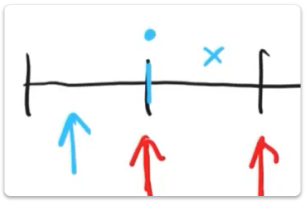
Quindi: tempo di pericolo uguale al doppio del tempo di pacchetto (collisioni che possono avvenire in tutto il tempo di pacchetto del client che ha già l'accesso e che durano per tutta la durata di accesso del nuovo client)
😡: basso utilizzo della rete efficiente (si parla del 18%)

ALOHA SLOTTED

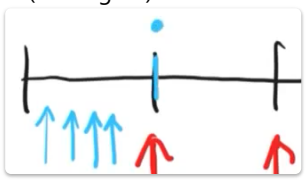
- Introduce un minimo coordinamento: gli host possono accedere solo in precisi istanti temporali

Cioè ad esempio nei punti indicati dalle frecce

Se un utente richiede l'accesso durante uno slot (freccia celeste) già occupato, non accede immediatamente, ma attende per il prossimo istante temporale di riferimento utile per l'accesso (prima freccia rossa, che indica anche la fine per il segnale che attualmente occupa il canale). In questo modo l'accesso sarà esclusivo



- Nota: ci saranno ancora collisioni se le richieste son tante (vedi figura)



Quindi:
tempo di pericolo pari a un singolo tempo di pacchetto (richieste concorrenti solo nell'intervallo di tempo occupato dal pacchetto che ha già l'accesso sul canale)

😃: rendimento il **doppio** migliore rispetto ad Aloha Puro (si parla del 36% di utilizzo in media)

> **Risolvere le collisioni: analisi statistica**
>
> $(**)$

- Prendiamo come riferimento Aloha Slotted

> *Dato che la collisione coinvolge almeno due client, una volta notificato a essi dell'avvenuto problema è necessario implementare un sistema di coordinamento per evitare che richiedano immediatamente l'accesso, causano nuove collisioni e così via.*

- L'idea è quella di *schedulare* le fasi di accesso
  - Modalità di tipo statistico (alternativamente detto casuale) personalizzata a ogni utente

```
- 😒Ci immaginiamo in primis una distribuzione Gaussiana: in questo caso avremmo un grosso addensamento di scelte centrato sul valor medio della distribuzione (in questo caso intorno allo $0$) - ovvero probabilità alta che due utenti vadano nuovamente a scegliere *uno stesso istante di accesso* $\to$ non realistica, perché *non equamente probabile su ogni istante di tempo*
```

  - 😉La **distribuzione uniforme** è quindi quella più adatta: tutti gli istanti hanno la stessa probabilità di essere scelti come istante di accesso. Scegliamo quindi questa ✓

CSMA

Carrier Semsing Multiple Access: tecniche ad accesso multiplo con rivelazione di portante

- Topologia di rete a bus associata a queste tecniche

Prevede una **fase iniziale** preventiva all'accesso stesso che ha l'obiettivo di individuare la presenza di *attività del canale*

- Viene realizzata andando a ricercare all'interno del canale la presenza del **segnale portante**, ovvero quella frequenza che trasporta l'informazione *dati* - è un tono ben preciso (e riconosciuto), quindi quando lo si ricerca si individua bene e significa proprio che qualcuno sta utilizzando quel canale.
    - Se ci si accorge di ciò, *non* si accede al canale (evitando collisioni)
        - Se è presente attività sul canale ma non si accede effettivamente allo stesso, si parla di *collisione virtuale*. Dopodiché si riprogramma un accesso con le stesse modalità di Aloha Slotted (cioè su un intervallo statisticamente uniforme detto di $take\text{-}off$)

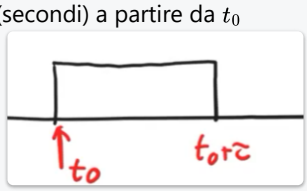
😃: evito collisioni
😡: tempi di propagazione di un segnale sul supporto. Sono piccoli, ma mai nulli: la fase iniziale potrebbe introdurre ritardi (vedi dopo)

Tempi di Propagazione del Segnale: possibile svantaggio concreto?

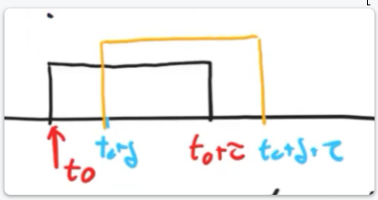
Supponiamo di trasmettere pacchetti di durata τ ciascuno
Si considerino ora due stazioni A, B distanti tra loro un tempo δ

> *Ipotizziamo quindi che sia A che B decidano di accedere al canale all'istante t_0*
> *Dato che il canale si suppone libero, entrambi trovano il canale disponibile (non rilevano segnali portanti)*

A accede e trasmette quindi un pacchetto di durata τ (secondi) a partire da t_0



B ha fatto lo stesso, cominciando a trasmettere da $t_0 + \delta$, ma i due tra loro non se ne sono accorti. In particolare, guardando dal punto di vista di A, essa si accorgerebbe della presenza di B solo δ secondi dopo.
Dato che anche il secondo pacchetto ha durata τ, allora B ha trasmesso nell'intervallo $[t_0 + \delta \div t_0 + \delta + \tau]$. Riassumendo graficamente:

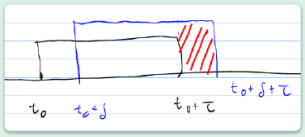


La distanza δ è quindi **l'intervallo di pericolo** (vulnerabilità), perché appunto le due stazioni non si accorgono l'uno dell'altro per una finestra temporale pari a δ

Pertanto, si auspica per avere i vantaggi delle reti CSMA che δ sia *più piccolo possibile*, ovvero vorremmo

$$\boxed{\delta << \tau} \quad ,$$

✓ **quindi vogliamo che la durata di trasmissione di un pacchetto sia molto superiore al massimo tempo di propagazione della rete (vogliamo area rossa piccola)**



TECNICHE CSMA

Esistono varie di tecniche CSMA, e sono le seguenti:

- $Persistent$: una volta che un nodo trova la rete occupata, continua con persistenza ad ascoltare il canale. Appena l'attività termina, si procede quindi all'accesso (logica ascolta prima di parlare)
  😃: efficiente quando l'utilizzo del canale è sporadico
  😡: se l'utilizzo della rete è intenso, potrebbero esserci numerosi client che vogliono entrare in collegamento, e quindi entrambi stanno in ascolto. Quando l'attività di riferimento termina, $entrambi$ entrano, causando *collisioni*

- $Non\ Persistent$: una volta ascoltato il canale (che si rivela occupato), lo si interpreta come una *collisione virtuale*. Si procede quindi a schedulare un nuovo accesso in futuro (senza rimanere a riprovare in continuazione)

- $P\text{-}Persistent$: (compromesso) - se il canale è occupato, si schedula l'accesso in modalità casuale. Se il canale libero con probabilità P si accede, mentre con probabilità complementare $(1 - P)$ si schedula l'accesso per un istante futuro

    - Essendo P un parametro possiamo scegliere un valore ottimale da assegnare. In generale:
        - Per utilizzi bassi della rete, $P \to 1$
        - Per utilizzi alti della rete, P diminuisce

Le tecniche CSMA hanno avuto fortuna nei suoi utilizzi con la creazione della seguente alternativa:

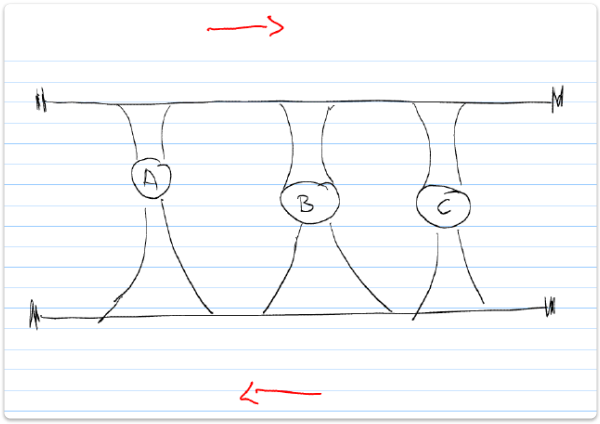
- $CSMA\text{-}CD$: (CD $\to$ collision detection) - viene mantenuto l'ascolto del canale durante l'accesso allo scopo di rilevare una collisione. Quindi si accede al canale ma si continua ad ascoltare: appena si sovrappone un nuovo client si segnala la collisione (logica ascolta prima di parlare *e mentre parli*) - La rete è così più efficiente
    - Utilizzata nelle reti Ethernet
    - Sigla $\text{IEEE } 802.3$

PROTOCOLLO DQDB

$Distributed\ Queue\ Dual\ Bus$

- Rete con topologia doppio bus con politica di coda distribuita

    - Doppio Bus: relativo alla topologia della rete
    - Coda Distribuita: significa che la rete ordina le richieste di accesso utilizzando una coda virtuale con politica FIFO. Virtuale perché non è localizzata in un unico punto della rete ma è distribuita



- Ciascun BUS ha una direzione di trasmissione (uno trasmette l'altro prenota)
- Ogni nodo è connesso a entrambi i bus (con due connessioni)
- Si suppone di trasmettere singoli pacchetti
    - Se vogliamo fare $\boxed{\text{A} \to \text{B}}$, A deve prenotare l'accesso al bus inferiore
    - Il tempo in ciascun bus è diviso in slot, seguendo quindi la tecnica TDM
        - Ogni slot ha la seguente struttura:

$$\underbrace{\boxed{\text{Stato Slot}}\boxed{\text{Prenotazioni}}}_{header}\boxed{Payload}$$

            - Stato Slot (1 bit): vale 1 se $Payload$ non è disponibile, 0 altrimenti
            - Prenotazioni (1 bit): vale 1 se è già prenotato 0 altrimenti
            - $Payload$ (n bit): campo dati

Facciamo un esempio, con le seguenti ipotesi:
Bus Superiore: Trasmissione
Bus Inferiore: Prenotazione
Vogliamo fare $\text{A} \to \text{B}$

A livello generale (logica semplice del doppio bus)
Il nodo A vuole trasmettere nell'altro bus, quindi prenota ponendo a 1 il relativo campo di un pacchetto disponibile che circola nel bus di prenotazione

- Se il campo $Payload$ è libero, ci può mettere anche i propri dati direttamente

Nel dettaglio (con coda di priorità)
Ogni nodo ha un contatore a $incremento/decremento$
> Nel bus inferiore legge i campi di prenotazione occupati
> Nel bus superiore legge i campi di $Payload$ liberi

La procedura è la seguente:

Il contatore $incremento/descremento$:

- Quando vede passare $Prenotazione = 1$ sul bus di prenotazioni si incrementa di 1
- Quando vede passare $Payload = 0$ sul bus di trasmissione si decrementa di 1
  Istante per istante il valore che ha rappresenta *quanti client sono in attesa di accedere e hanno già notificato al relativo nodo il bisogno di farlo* (ordinamento temporale)

Quando il nodo ha bisogno di accedere "si mette a caccia" del primo $Prenotazione = 0$ sul bus di prenotazione e lo cattura.
Questa operazione innesca il trasferimento del valore del contatore $incremento/descremento$ sul contatore $decremento$: Il suo valore logico rappresenta il numero di stazioni (client) che devono (ancora) accedere al canale, e lo faranno prima (cioè si deve rispettare l'ordine delle richieste già acquisite)
Quando il contatore $decremento = 0$, vuol dire che tutti i nodi (prima in fila) hanno potuto accedere $\rightarrow$ il primo pacchetto con $Payload = 0$ viene catturato e ci si scrive i nuovi dati

> *Concetto di coda virtuale (condivisa)*

× Difetti del Protocollo

😡: non consente la distribuzione equa del diritto di accesso a tutti i terminali - casi di monopolizzazione del canale (ad esempio se A accede per primo al canale prenota tutti i pacchetti prenotazione e non ne lascia liberi per un ipotetico B che vuole accedere). Se non capita monopolio comunque non è ben equalizzata molto spesso

✓ Rimedio (parziale)

Si stabilitsce un massimo numero di prenotazioni che un terminale può effettuare consecutivamente

😡: non consente di garantire quanto ci vuole alla consegna di un pacchetto: dipende dalla situazione della rete (difficoltà gestione traffico isocrono)

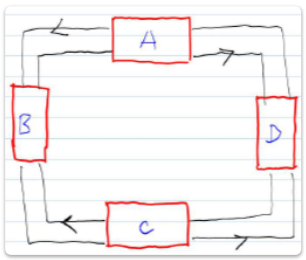
✓ Rimedio

Si permette la prenotazione di slot con una certa frequenza di tempo (esempio: concedo un slot ogni tre secondi)

PROTOCOLLO FDDI

$Fiber\ Distributed\ Data\ Interface$

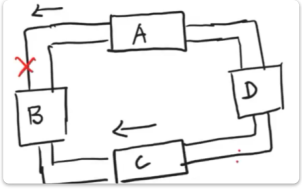
- Mezzo trasmissivo in fibra ottica (o perlomeno di elevata qualità)
- Topologia a doppio anello (doppio ring), come in figura



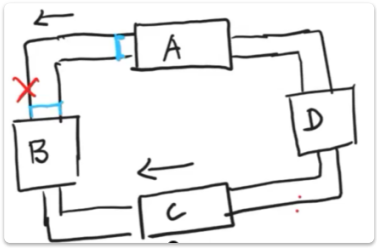
- I due anelli hanno direzione propagazione opposta:
- Uno in senso anti orario e l'altro in senso orario

✓ Il *doppio* anello porta diversi vantaggi:

- Maggiore *resilienza* della rete $\rightarrow$ più **affidabile** per quanto riguarda i guasti e qualora ce ne fossero, più rapidità per ripristinare la situazione
  Ad esempio, supponiamo si interrompa nel punto X in figura:



Prima che venga riparato si congiungono due collegamenti (parte celeste) in due punti uno precedente e uno successivo al guasto ($ponticellare$):



Vale lo stesso se si blocca un intero nodo A, B, C, D

- Aumento *velocità* di trasmissione: si può trasmettere in parallelo sui due anelli raddoppiando la portata ($rate/sec$)
  - Mediamente ogni anello ha velocità $100\ Mbit/s$

Utilizzi FDDI

- Reti Locali (esempio università)
- Ospedali: consentono di disporre sistemi di trasferimento di informazione anche di dimensioni elevate e archiviazione in modo veloce

---

- Questo protocollo viende detto *protocollo ordinato con negozazione*

Esistono due tipologie di pacchetto che circola su tale rete:

- **Token Frame**: struttura definita condivisa e nota da tutti i nodi della rete. Consente di applicare il protocoll di accesso ordinato
- **Data Frame**: struttura che dipende dal tipo di informazione che si vuole trasmettere (e quindi non nota a tutti)

Le caratteristiche di questa rete (dato che si utilizza un canale di buona qualità - fibra ottica) sono le seguenti:

- il **controllo di errore si esegue su base end-to-end** quando richiesto
- Solo chi ha *immesso* nella rete *una certa informazione* (mittente) ha la *possibilità di rimuovere* tale informazione
  - In particolare accade che: il mittente inserisce l'informazione nel pacchetto e spedisce sulla rete a un certo destinatario che ne conserva una copia. L'informazione continua a circolare sull'anello finché non torna al mittente che analizza eventuali errori e se non sente il bisogno di ritrasmettere l'informazione la rimuove dalla rete

TIPOLOGIE DI TRAFFICO

Con questa rete si possono trasmettere più tipologie di traffico, divise in classi (priorità):

- Traffico primario (*sincrono*): non è isocrono, cioè continuo nel tempo (costante)
- Traffico secondario (*asincrono*): traffico a priorità più bassa che viene fatto circolare in rete solo se questa si trova nelle condizioni giuste per farlo

TTRT E CONTATORI

Ad ogni nodo è sempre riconosciuta una fase di accesso che consenta la trasmissione del traffico sincrono (a priorità più alta)
Si definisce preventivamente un parametro detto *token target rotation time (TTRT)*: indica il tempo (nominale) che il token (cioè il messaggio di controllo) impiega a fare un giro completo (è un valore di riferimento)
Ogni nodo quindi dichiara un certo TTRT per trasmettere traffico sincrono
> Bisogna tenere conto del tempo di trasferimento del token!

Infatti, se:

- i nodi sono in tutto N;

- α_i è il tempo di accesso per traffico sincrono (primario) del nodo i
- w_i è il tempo di passaggio ($walking\ time$) dell'autorizzazione all'accesso da un nodo a quello successivo (ovvero $i \to i+1$)
  Allora TTRT si definisce con questa regola:

$$\boxed{\text{TTRT} \geq \sum_{i=1}^{N} \alpha_i + \sum_{i=1}^{N} w_i}$$

  Quindi la capacità assegnata a un nodo i è: $\frac{\alpha_i}{\text{TTRT}} \cdot \overbrace{100\ Mbit/s}^{\text{velocità nominale}}$

  In generale è un parametro, quindi il suo valore lo decidiamo noi a priori (non è che lo misuriamo, anche se va scelto con logica come visto)

Ogni nodo è dotato di due contatori:

- TRT: misura il tempo *effettivo* (non nominale) di rotazione del token ($token\ rotation\ time$) - dal punto di vista di un nodo: quanto ci mette il token in un giro a tornare da lui (tempo effettivamente impiegato a fare un giro - non è un parametro: è un dato)
- THT: ($token\ holding\ time$) - è un contatore a descremento. Ha valore dipendente dalla seguente relazione tra il TTRT e il TRT:

$$\text{THT} = \text{TTRT} - \text{TRT}$$

> - Se $\text{THT} > 0$: il nodo inizia la fase di accesso sincrono. Terminata, si controlla il valore del contatore che viene descrementato di α_i (tempo accesso).
>   - Se il contatore raggiunge un valore finale negativo o nullo, si passa l'autorizzazione al nodo successivo
>   - Se rimane un residuo (valore positivo), si utilizza il tempo residuo per attivare una fase di accesso asincrona (secondaria - perché la mole di dati è ridotta)
>   Riassumendo:

$$\text{THT} = \begin{cases} \alpha_i & 1^\circ \text{ volta oppure } \text{TTRT} - \text{TRT} \leq 0 \\ \text{TTRT} - \text{TRT} & altrimenti \end{cases}$$

EFFICIENZA

L'efficienza per una rete FDDI si definisce col seguente parametro:

$$\eta = \frac{\text{TTRT} - \sum_{i=1}^{N} w_i}{\text{TTRT}}$$

- Al numeratore abbiamo l'intervallo di tempo durante il quale la rete è impiegata effettivamente per la trasmissione del traffico. Il tutto in relazione (denominatore) al tempo nominale

Il rendimento tende al valore migliore possibile (ovvero 1) tanto più $\sum_{i=1}^{N} w_i$ si avvicina a 0

ESERCIZIO FDDI

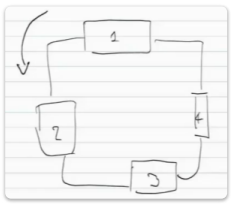
Si supponga:

- Tempo di passaggio del token da un client a quello successivo sia nullo (istantaneo)
- Rete con 4 nodi, con la stessa necessità di trasmettere i dati
- Stesso tempo di accesso per tutti i nodi, ovvero:

$$\alpha_i = 2 \quad \to \quad i = 1, 2, 3, 4$$

- Ogni nodo quando ne ha la possibilità invia traffico asincrono (cioè quando basta un $\alpha_i < 2$ in questo caso per accedere)

0. Disegniamo lo schema della rete (basta un anello per ora):



1. Definiamo TTRT. Seguendo la formula:

$$\text{TTRT} \geq 2 \cdot 4 = 8$$

   Scegliamo $\boxed{\text{TTRT} = 12}$ (cfr. dopo perché questa scelta, a cose "normali" bastava 8)

2. Costruisco le colonne di TRT e THT

1		2		3		4	
TRT	THT	TRT	THT	TRT	THT	TRT	THT
0	2+10	12	2	14	2	16	2
18	2	8	2+2	10	2	10	2
10	2	10	2	8	2+2	10	2

> *Lettura tabella:*
>
> - *Il token al primo nodo (1) ha tempo di rotazione 0 (di default). Si confronta quindi il valore misurato cioè 0 con 12. Essendo $12 - 0 \geq 0$, si carica il contatore THT con il valore 12. Di questi, due unità sono impiegate per l'accesso a priorità alta α_i, e rimane quindi un residuo di 10, destinato a un traffico asincrono.*
> - *Il token arriva quindi alla stazione 2: TRT vale 12 (perché arriva dopo 12 unità di tempo). Essendo $12 - 12 = 0$, si trasmette solo le due unità di traffico sincrono α_i.*
> - *La stazione seguente vede arrivare il token dopo un tempo dato dalla somma della sosta sul primo nodo (12) con la sosta nel secondo nodo (2), quindi $\mathrm{TRT} = 14$. Si ha quindi $\mathrm{THT} = 2$ essendo $14 - 12 = 2 \geq 0$ e perciò si trasmettono (solo) le due unità α_i.*
> - *Idem per il successivo (solo con $\mathrm{TRT} = 16$)*
>   *Il token ritorna finalmente al primo nodo:*
> - *la somma ultime quattro soste precedenti dà il valore TRT, quindi $\mathrm{TRT} = 12 + 2 + 2 + 2 = 18$. Si trasmette quindi il traffico sincrono relativo (le due unita α_i)*
> - *il nodo due ha un valore TRT dato dalle quattro soste precedenti, che sono $2 + 2 + 2 + 2 = 8$, quindi $\mathrm{TRT} = 8 \leq 12$. Rimane 4 come residuo: si trasmettono due unità in modo sincrono α_i e le restanti due unità in modo asincrono*
>   - *terzo nodo ha $\mathrm{TRT} = 10 \leq 12$ quindi solo traffico sincrono $[\dots]$*
>   - *$[\dots]$ e così via $[\dots]$*

Quindi, concludendo:
Con il $\mathrm{TTRT}{=}12$ che abbiamo scelto:

$$\frac{\alpha_i}{\mathrm{TTRT}} \cdot \overbrace{100\ Mbit/s}^{\text{velocità nominale}} = \frac{2}{12} \cdot 100\ Mbit/s = \frac{1}{6} \cdot 100\ Mbit/s$$

Con il $\mathrm{TTRT}{=}8$ invece:

$$\frac{\alpha_i}{\mathrm{TTRT}} \cdot \overbrace{100\ Mbit/s}^{\text{velocità nominale}} = \frac{2}{8} \cdot 100\ Mbit/s = \frac{1}{4} \cdot 100\ Mbit/s$$

In quest'ultimo caso avremmo avuto più banda per il traffico sincrono ma non avremmo sfruttato la proprietà di accesso per proprietà più bassa (non sfruttavo al massimo le potenzialità della rete e non era nemmeno un esempio troppo ben illustrativo)
