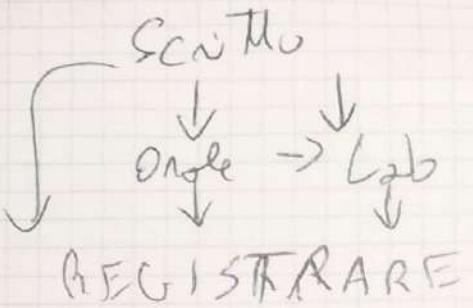


Esempio

Scritto: 15 ... 22

Onde: -5 ... 5

Lab: -5 ... 5



Sistema interconnessione - Closed Computing - IOT

Se si piccola la vita alle persone

Sistema di connessione fatto da:

- Nessun filo o trasmettore il regole — pubblica  
della costituzionalità
- Strutture logiche (software) — finché nel  
mondo

Cavo, onde elettromagnetiche ecc... — Alcuno poi

Meno — Trasmettente — Ricevitore

Il regole non è informazione, le strutture logiche  
si occupa di "trasmettere" un'informazione

Si deve individuare l'informazione al giusto modo.

Le strutture logiche si occupa di tutto

Problema: Il regole quando riguarda il suo pubblico,  
ma c'è lo stesso software che serve a  
risolvere il problema non ha questa interpretazione  
del regole (non regole)

Il regole pensamento troppo spazio e dogmatico, a  
volte il meccanismo non regolare del regole

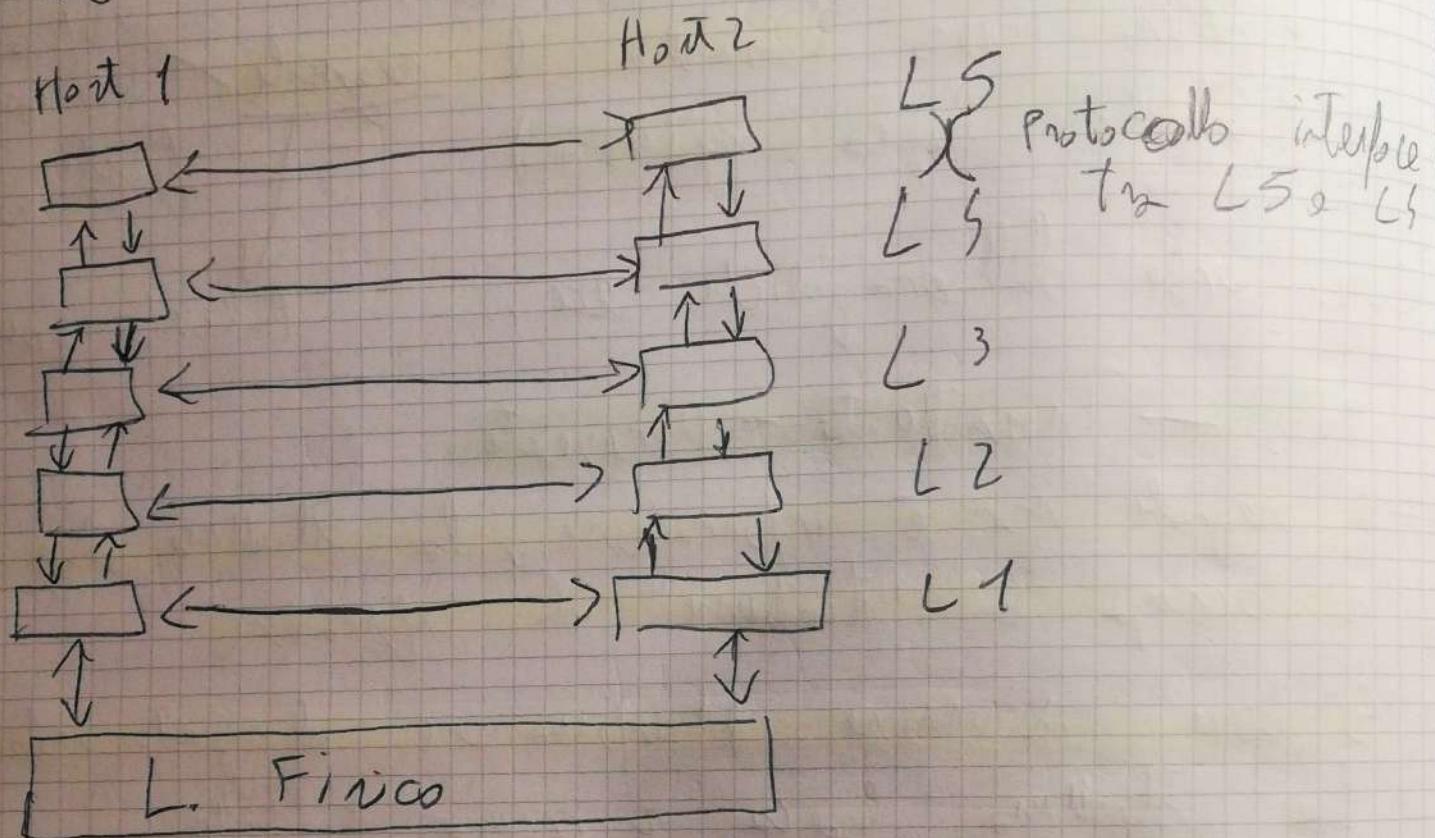
## Protocolli

definisce come e' attivato un messaggio

Fra di Host finchè ogni layer c'è un protocollo  
Fra un layer S e S+1 c'è l'interface

Host 1: Browser richiede informazione a

Host 2: WWW. E' ricevuta la richiesta con un  
protocollo di un certo tipo, ma poi subisce  
la richiesta (Host 2 invia un segnale di risposta)



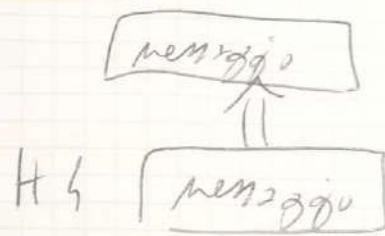
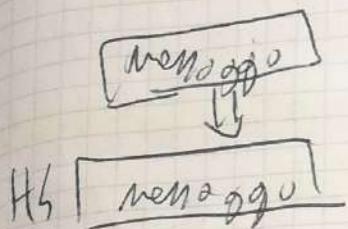
Fra vari L c'è uno dei protocolli.

Fra Host c'è un protocollo per comunicare

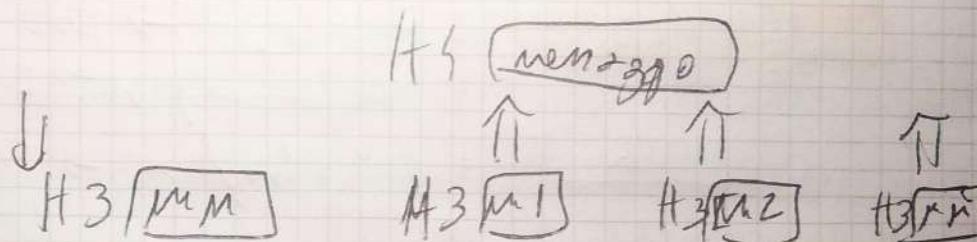
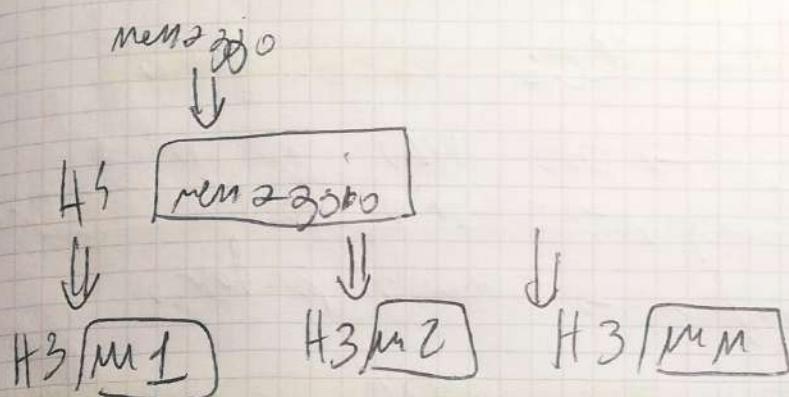
3 protocolli: verso il basso, alto, box / 1x

Protocollo def: definisce formato e ordine dei messaggi  
rispetti da entrambi in comunicazione e anche le attività in fase di trasmissione e/o ricezione  
messaggio o altro evento

## Schemi di comunicazione



E' possibile dividere il messaggio verso il basso (alto)



Interruzione: viene letto come interruzione il messaggio  
Informazione di controllo: si può cogliere cosa fare all'messaggio

La dimensione totale dei

spaziamenti è più grande del messaggio, dato che i spaziamenti oltre al messaggio altre informazioni (di controllo)

## Protocol Headers:

Viengono trasmesse varie informazioni

- source address
- dest address

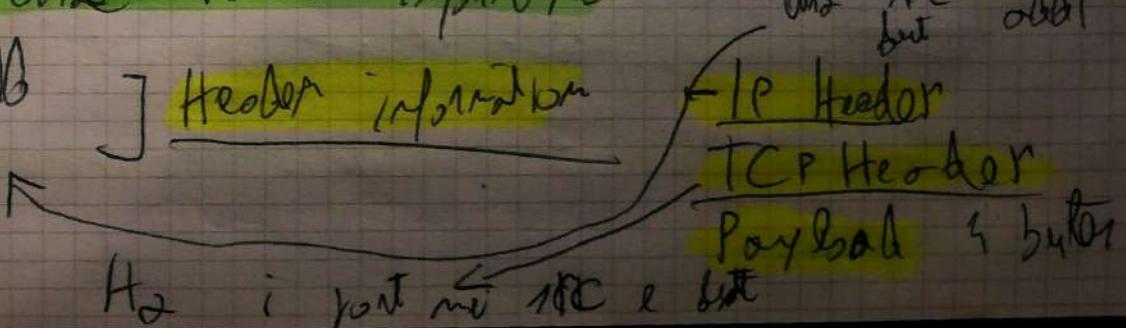
Header information

MSB 1FC but offset

IP Header

TCP Header

Payload & bytes



H2 è sotto MSB 1FC e but

A volte c'è un rapporto 50:1 tra info  
di controllo e messaggio

Ei distinguono i messaggi e i invi (messaggio)  
al livello corrispondente dell'alto livello (i invi come  
la strada di controllo Siger) (invio)

Problemi: E' necessario un canale di comunicazione  
affidabile e veloce.

Il canale è implementato così: esempio

Canale binico:  
• simplex → solo invio da parte  
• half-duplex → Entrambe le parti a turno  
• full-duplex → Entrambe le parti

• Messaggi di livello basso: non possono avere lunghezza  
(ci sono dei regoli)

• Trasmettore veloce non può trasmettere a velocità  
lenta.

• E' dove trovare il percorso migliore - fonte - destinazione  
(la ricerca è arbitraria) (ci sono molte vie)

• Ordine arrivo messaggi deve essere uguale a  
quello di ricezione

N.B.: I regoli di controllo servono per dirigere bene  
le info i regoli-messaggio

Per evitare tutti questi problemi servono dei protocolli  
ben precisi

## Comunicazione

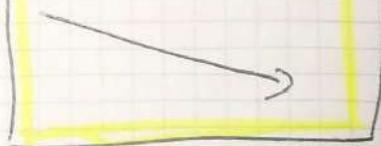
Connection bus = Affidabile

Connection socket = non affidabile

Teléfono è connection oriented

## Connection less

Morbore utile

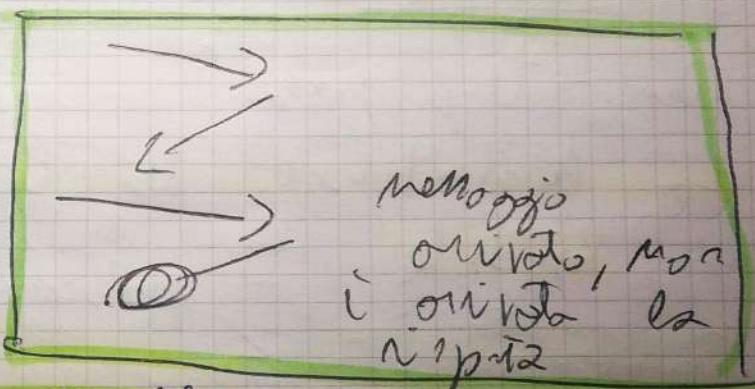
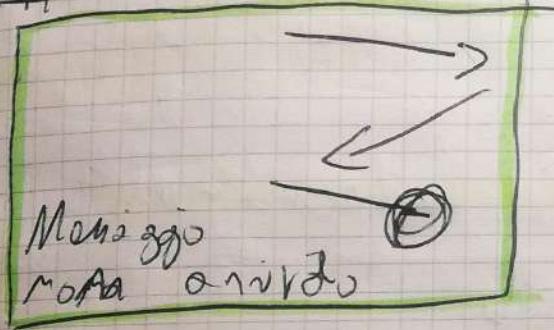


Azione — Comunicazione — Chiamata

Entro protocollo ben preciso

Non c'è un protocollo ben preciso

Affidabilità: Si paga il regalo, non è affidabile



Dunque mando il pacchetto, se ne paga poi i traccez

Soltanto Ampli: in caso di morata si ripete n. invio il pacchetto.

Quella non vale come soluzio.

Serve un protocollo per ogni possibile situazione problematica relativa ai messaggi che non arrivano/ non ricevono risposte  
I timer funzionano in base al protocollo

Problema: determinare a quale impostare il timer

Problema: il protocollo deve essere unico, non  
può esserci un protocollo per il servizio A  
Bob è un servizio per Elisa

Problema: bisogna distinguere bene i pacchetti <sup>informazione</sup> e i pacchetti incaricati.

Il messaggio e l'ACK devono essere numerati.

Nell'ACK c'è solo informazioni di controllo  
(numero come risposta per bire "OK messaggio incaricato")

"Quanto tempo passa?"

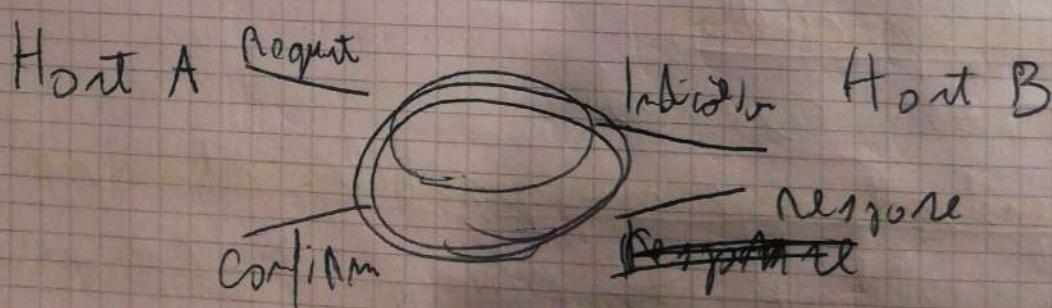
La distribuzione del tempo è estremamente variabile

Protocollo Snello

Connection-less  $\leftarrow$  c'è iraffidibilità

Nei connection-oriented c'è affidabilità

↑  
Protocollo confuso



Modello OSI, divide in 7 layer

Application

Presentation

Sessions

Transport

Network

Data Link

Physical

Ogni livello ha un protocollo per comunicare col livello connesso  
di un altro Host, e protocollo  
per far comunicare tra i due col  
successivo

**TCP/IP** ha:

- Application layer, Host-to-host
- Transport layer

Non c'è a livello Internet layer, Network Interface layer

Set di protocolli più usato

Modello OSI: è un modello teorico

TCP/IP: modello reale (molti diversi in TCP e IP)

Application: serve applicazioni & utente e definisce protocolli

Presentation: garantisce che le applicazioni che vogliono comunicare si interpretino i dati nello stesso modo

Sessions: fornisce definizione e sincronizzazione  
dei servizi dei dati

**Transport**: trasferire i messaggi dal livello di applicazione tra gradi periferici gestiti dalle applicazioni

Ex: TCP fornisce servizio orientato alla connessione (consente connessione garantita e controllata di flusso) TCP fornisce i messaggi lunghi in più pacchetti

**Rete (network)**: si occupa di trasferire pacchetti a livello di rete, detti datagrammi, da un host a un altro

Usa il protocollo IP

**Data LINK** (<sup>(da ms p<sup>a</sup> elaboratore dati)</sup>): ci sono offerto il servizio dato da questo layer per trasferire un pacchetto da un n<sup>o</sup> host al successivo nel percorso

Dunque in ogni n<sup>o</sup> host il livello di rete può utilizzare gli strumenti del Data LINK per trasportare il n<sup>o</sup> successivo

Alcuni protocolli garantiscono efficienza, altri no  
Quelli che sono ottimizzati per attraversare più nodi, lungo il percorso sono quelli che sono più efficienti

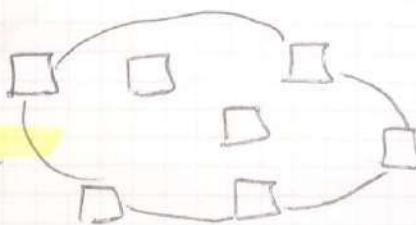
**Fisico**: trasferisce i bit dal jocattolo (a livello Data LINK) da un n<sup>o</sup> host al successivo

I protocolli qui riguardano gli strumenti fisici esistenti (cavi, onde...) Dunque i bit possono essere trasferiti secondo diverse modalità

## Tipologie di rete

- Collegamento semplice : PUNTO - PUNTO □ - □  
due macchine collegate in modo diretto

- Broad Cast Pointe di raggiungere  
tutti i altri contemporaneamente  
Per gestire rete e regole



Sempre un protocollo per decidere chi parla e quando.  
Bisogna evitare le interferenze, collisione del segnale

Condivisione / Sincronizzazione dei segnali, i segnali quando si incontrano non si interaggono e ricirca, si sovrappongono, si sommano.

La tipologia di rete è reticolare:  
Condizione: chi ascolta non capisce più nulla

E se un segnale è molto più grande dell'altro,  
questo segnale si sovrappone all'altro, la somma è insensibile

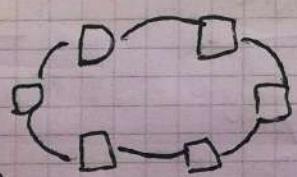
## Tipi di broad cast

Bus condiviso



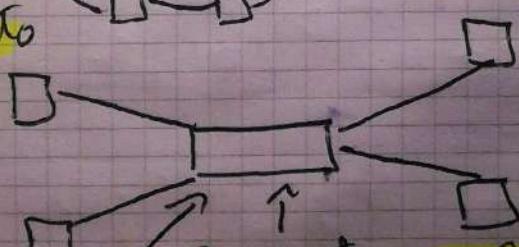
Tutti i interi tra di loro ignorano il bus

Anello



è un modo per fare broad cast con cori Punto-Punto. Non deve esistere collisioni

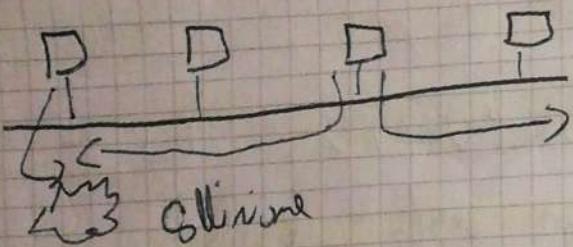
Stella



Concentrazione Collegato Punto-Punto con tutte le macchine

Disvantaggio delle macchine

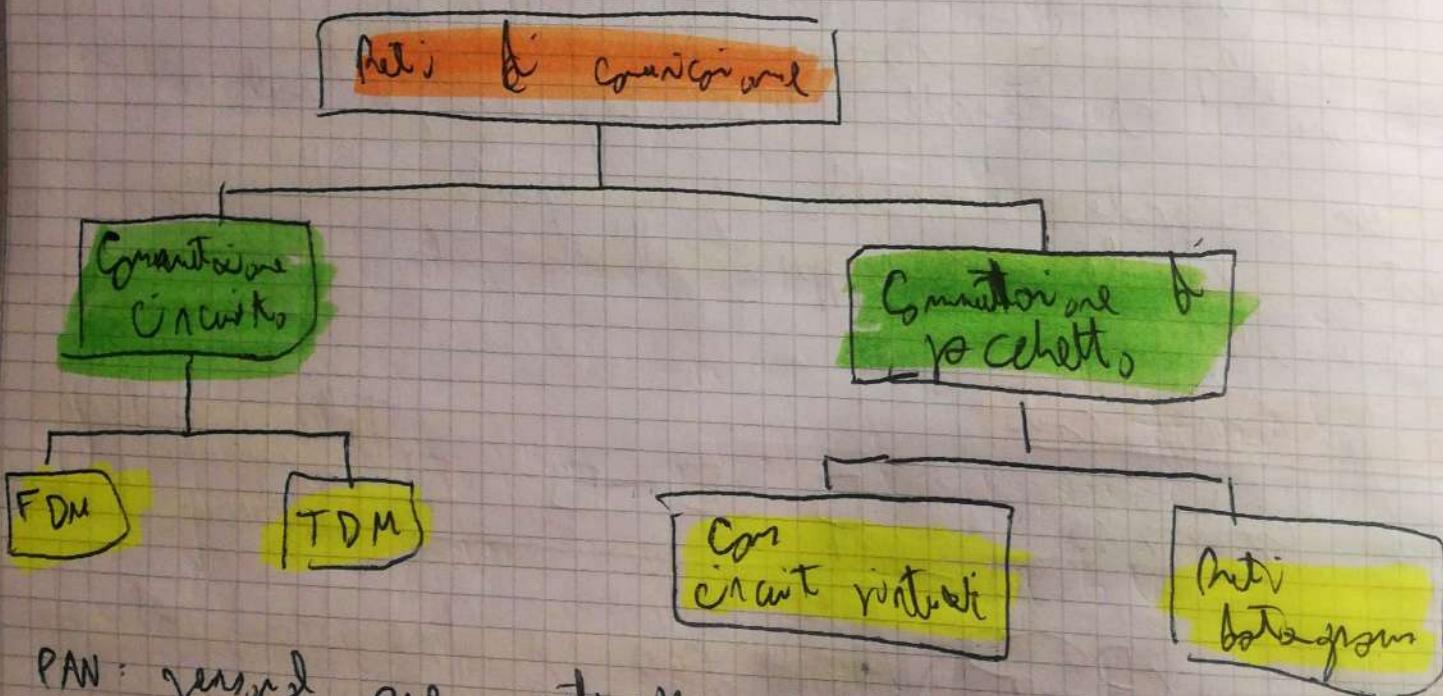
difficile; ascolta tutti, non ha il segnale a tutti



~~Ritardo dovuto a processi  
basingo a resezione regole~~

Il broad Cast abusivo permette di ignorare il regole, in questo modo si evita la sovrabbondanza eccessiva del regole. ~~ritardo dovuto a ritardo dovuto a processi~~ e rigenerazione regole

Nel broad cast ~~abusivo~~ e stellare il regole non è operato, qui siamo basati solo a messa in Hub per creare un colpo di regole da inviare i request a tutti;



PAN: personal area network locali

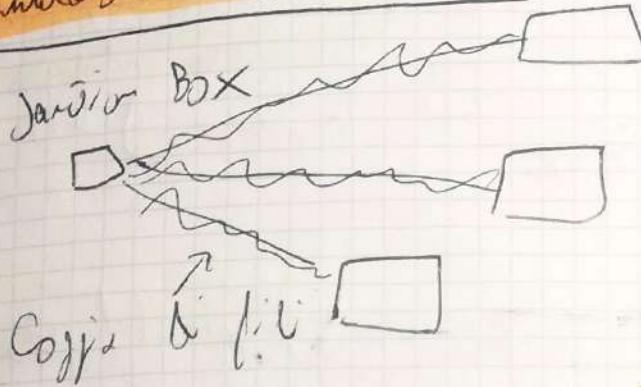
LAN 1 Km

MAN 10 Km

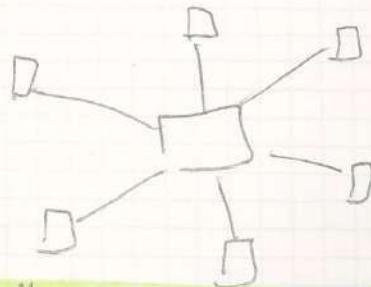
WAN 100-1000 Km

Internet 10K + Km

## Connessione a circuito



F1: Sintesi filosofica



Fili collegati fiancate

## Connessione a pacchetto

Faccia un giro

Traessi verso rigore

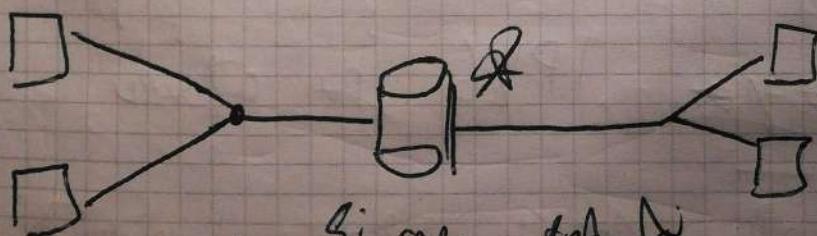
Progetto Airport: →

~~Faccia~~ Ormai non è più già connessione a circuito, ma a pacchetto.

La connessione consiste in uno switch, il solo switch il pacchetto nel senso rigore.

La linea tratta la parola mare per indicare gli informi / segnali sulle linee Ogni.

In realtà creando una colonna pacchetti che torna in attesa per essere inviati al nodo successivo



E' uno colonna di pacchetti

E si vede lo spazio del buffer. Per la colonna, non fa open buffering

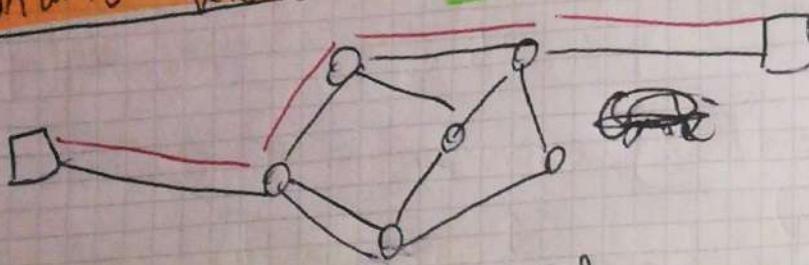
I generatori di pacchetti hanno il giro colonna

Nel caso di passaggio a sing. intelligenti per gestire traffico pacchetto

Crediti ritardo

Jitter minimo

Ogni pacchetto ~~scende~~ negli  
ha meno ritardo  
In tempi Jitter basso



Si realizza nel grado il raggiungimento minimo (finito)

Vantaggi: pacchetti arrivano in ordine, non c'è nessun  
processamento, la decisione di ritardo.  
Il routing è fatto una volta solo  
determinando delle informazioni del 1° pacchetto, tutte  
oggetto

Svantaggi: se un router rotta; si deve riavviare  
ogni pacchetto che deve uscire quel router

Datagram: Per ogni pacchetto si decide una tratta,  
il pacchetto scatta e rotta

Vantaggi: se un router rotta, ci sono  
altro

Jitter: L'utente vuole stampo la propagazione molto, nel  
datagram non è più possibile.

Inoltre  
variazione di ms o gli caratteristiche di un rete

IPV4 ha solo Datagram, IPV6 ha entrambi

→ N.B. non dice se le altre tecnologie mantengono  
i intervalli di 50 ms, vogliono dire sinistramente  
con intervallo di 50 ms.

Più è grande la diffidenza al guadagno e più è fine più è alto il settore.

## Livello Applicativo

Le applicazioni richiedono alla rete un insieme di librerie applicative per realizzare messaggi.

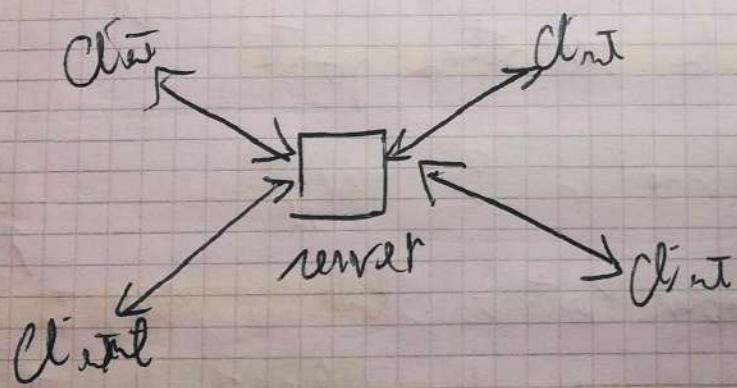
## Comunicazione tra processi

Nel sistema di distribuiti due processi remoti possono procedere in parallelo se i sincronizzano.

La sincronizzazione è garantita con lo scambio di messaggi.

Il processo invia messaggi alla rete e riceve messaggi dalla rete attraverso un'interfaccia software chiamata socket (che reflette come una "porta")

## Modello Client - server



Client: Node, legge

Server: scrive, scrive

Server: scrive storia

Client: guarda storie

Il processo server è attivato ~~per~~ da domande storie attivate da altri clienti.

Il client-server è un duopolio dove, quando vede la sua branca, gli rispondono.

## Registi applicazione

### Tolleranza ai punti

- Throughput - lunghezza di banda : Entra in cache
- Fine codice - visual temporal : "Voglio che io veda lo my"
- Sicurezza

Ognuno ha delle caratteristiche diverse dal servizio stesso, nel trasformare di un file non voglio questi, in un file tollestante i tollerabili. ecc ecc

Sono previsti due servizi UDP e TCP, due protocolli di trasporto.

Processi - interi ? tranne le porte \* Somewhere in mobi stabab  
le connessione / collegare

Il processo è legato a un socket dove c'è una porta. Questo socket è collegato per il corrispondente socket della stessa porta

Porte di rete dirette in 3 gruppi

Winter, P28

0-1023 Well Known Port: tutti i servizi già impostati

1024-49151 che mi specifico Registered ports

Dynamic and / or Private ports 49152-65535 ]

\* A oggi processi sono legati una porta per portare tutte in internet. Ogni porta è monitorata

Ogni porta è monitorata e impostata con UDP o TCP

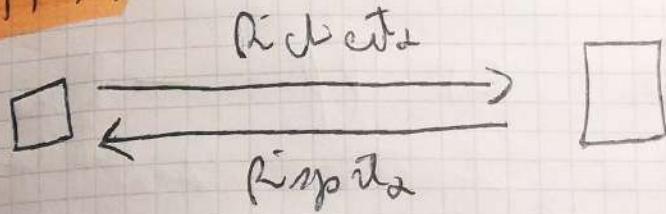
Sono già state

(TCP) Telnet : il servizio su internet, genera accesso a una shell remota, basta e password  
Funzione : utilizza socket, invia output sul socket

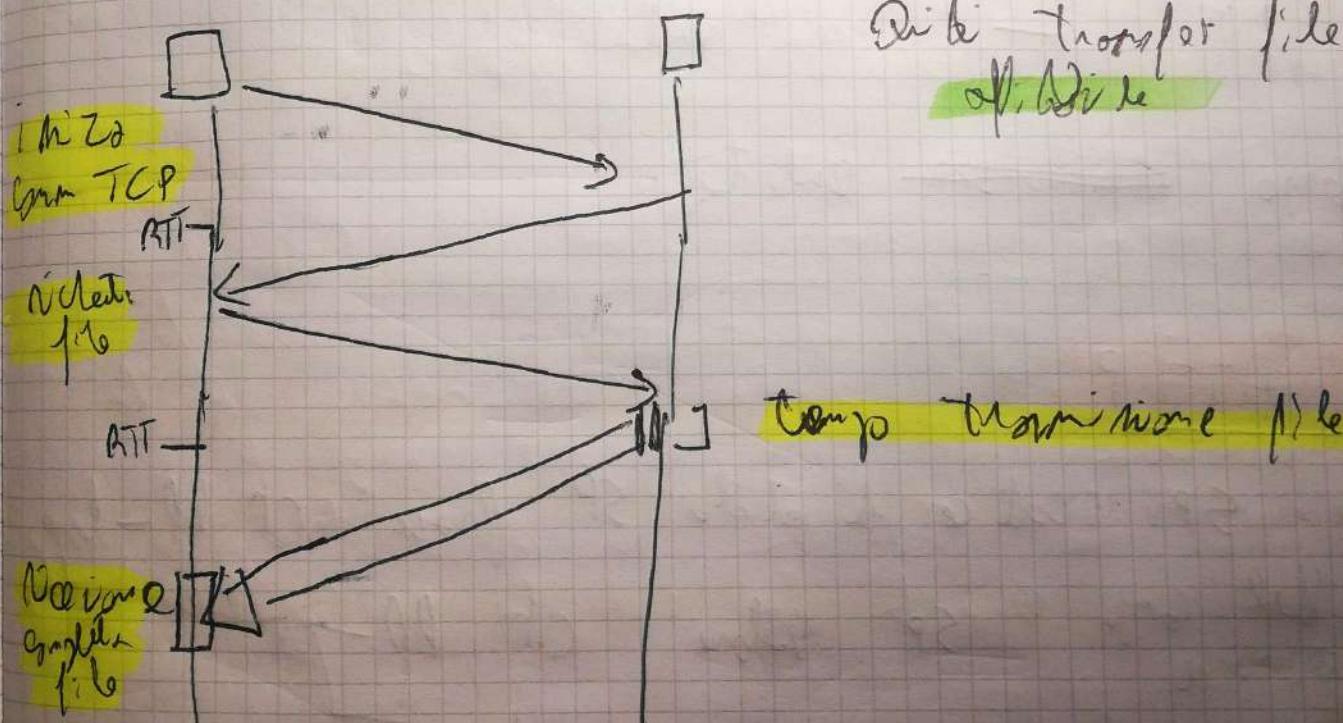
Client, server tutto → invia → Arrivo a server

Telnet ≠ Secure, la password inviata in chiaro può essere spiazzata e la password è in chiaro o trattata.  
Non serve a proteggere i dati. (client-server)

HTTP



Ecco TCP è un canale comune nelle transazioni



HTTP 1.0 chiede connessione TCP per trasferire singoli oggetti

HTTP 1.1 Max TCP permettimento

HTTP 2 usi compression intestazioni Primo Push Adi  
offro re il dato deve fare due domande

Messaggio

Domanda 1 → segnale risposta → Domanda 2 →

messaggio Domanda 1 → Domanda 2 ...  
completa download parallel

### Risposte HTTP

metodo SP URL SP versione

content controls  
CR LF

- linea  
newline

none del

segno interazione

SP valore

CR LF

- linea  
interazione

; segnali

CR LF - linea vuota

Coppia ~~intestazione~~ entità

### Risposte HTTP

versione SP codice di stato SP frase CR LF - linea  
di testo

none del segn.  
interazione

SP valore CR LF

di testo

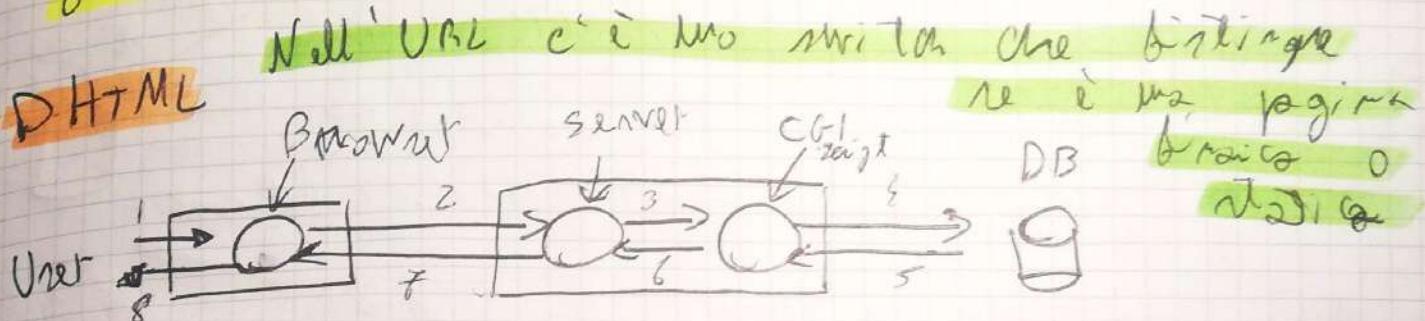
None ; segn.

— linea di interazione

CR LF — linea vuota

[Coppia entità]

Nella risposta HTTP compare un codice di stato,  
che non riguarda che i subordinati sono OK,  
o ci sono errori o altro.



- 1) User fills in form
- 2) Form sent back
- 3) Handled to CGI
- 4) query
- 5) record found
- 6) build pag
- 7) page returned
- 8) page displayed

E' un sistema dinamico che permette migliore  
interazione client-server, l'output è l'output di  
uno script nello language CGI scritto - (l'output è  
comunicato in HTML)

### Cookie

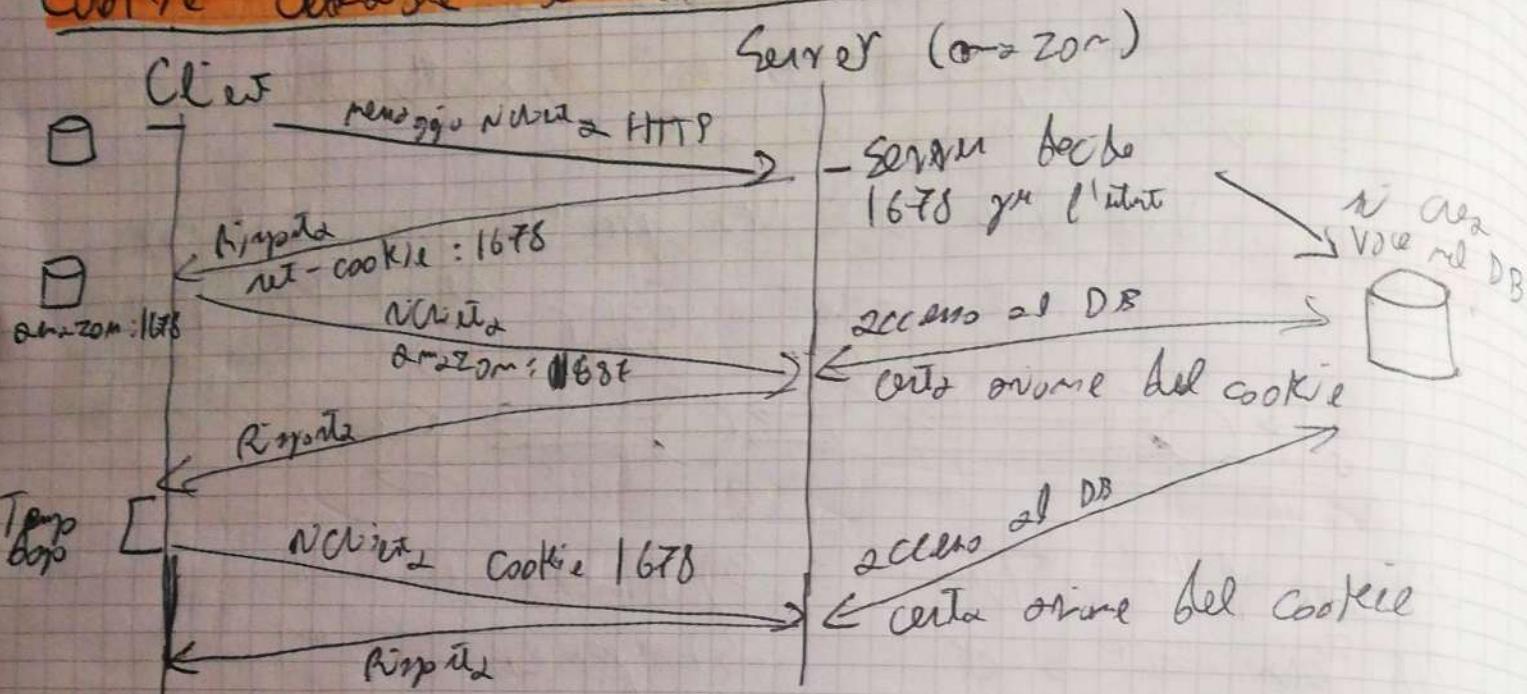
E' una stringa alfanumerica per l'identificazione  
utente.

HTTP offre i cookie che permettono ai server  
di tener traccia degli utenti.

I cookie non sono mai mai sicuri

NB: la versione TCP non si chiude in HTTP  
(quello di oggi)

## Cookie: creazione e utilizzo



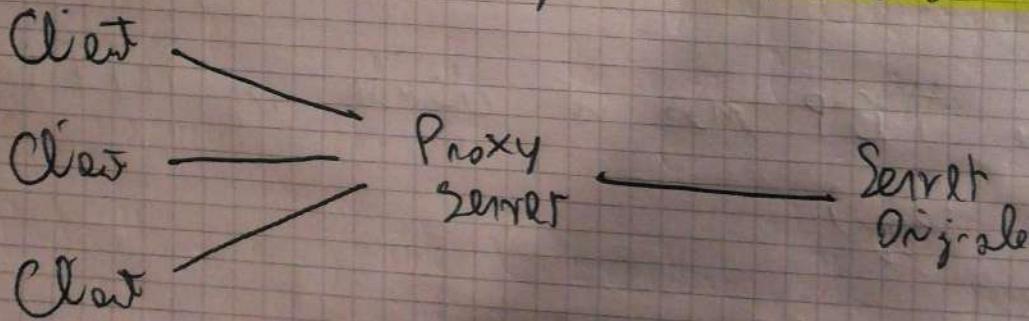
## Proxy Server

Si crea un'interazione tra Client e Server basta di mettere, sotto Web cache o proxy server.

- Entità interne che soffrono (ne guadagnano) richiesta HTTP al posto del web server effettivo.

Come funziona?

- 1) Browser invia richiesta HTTP al server proxy.
- 2) Se l'oggetto richiesto è in memoria si sottopone alla richiesta HTTP e basta, altrimenti lui fa una richiesta HTTP al server originale.
- 3) Sottopone la richiesta in memoria l'oggetto immagine e sottopone la richiesta all'client.

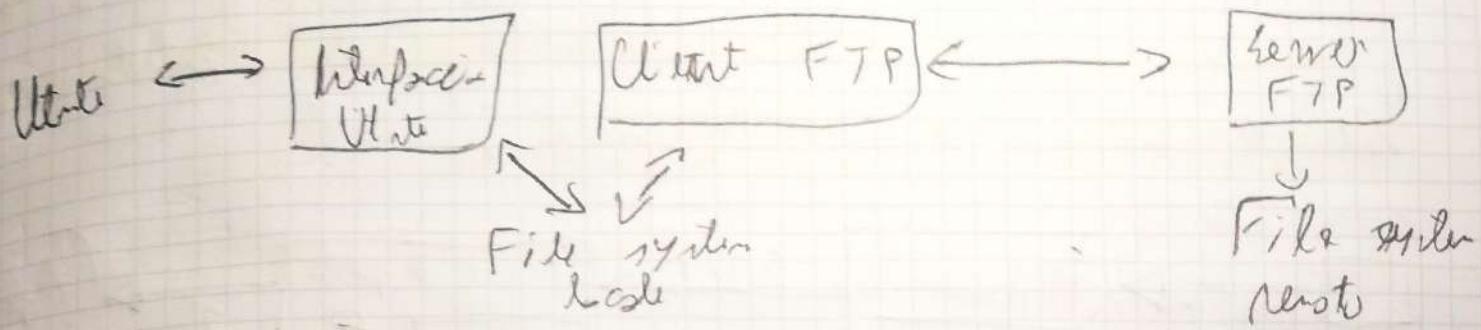


FTP: no autenticazione

SFTP, FTPS: versione sicure di FTP

## FTP

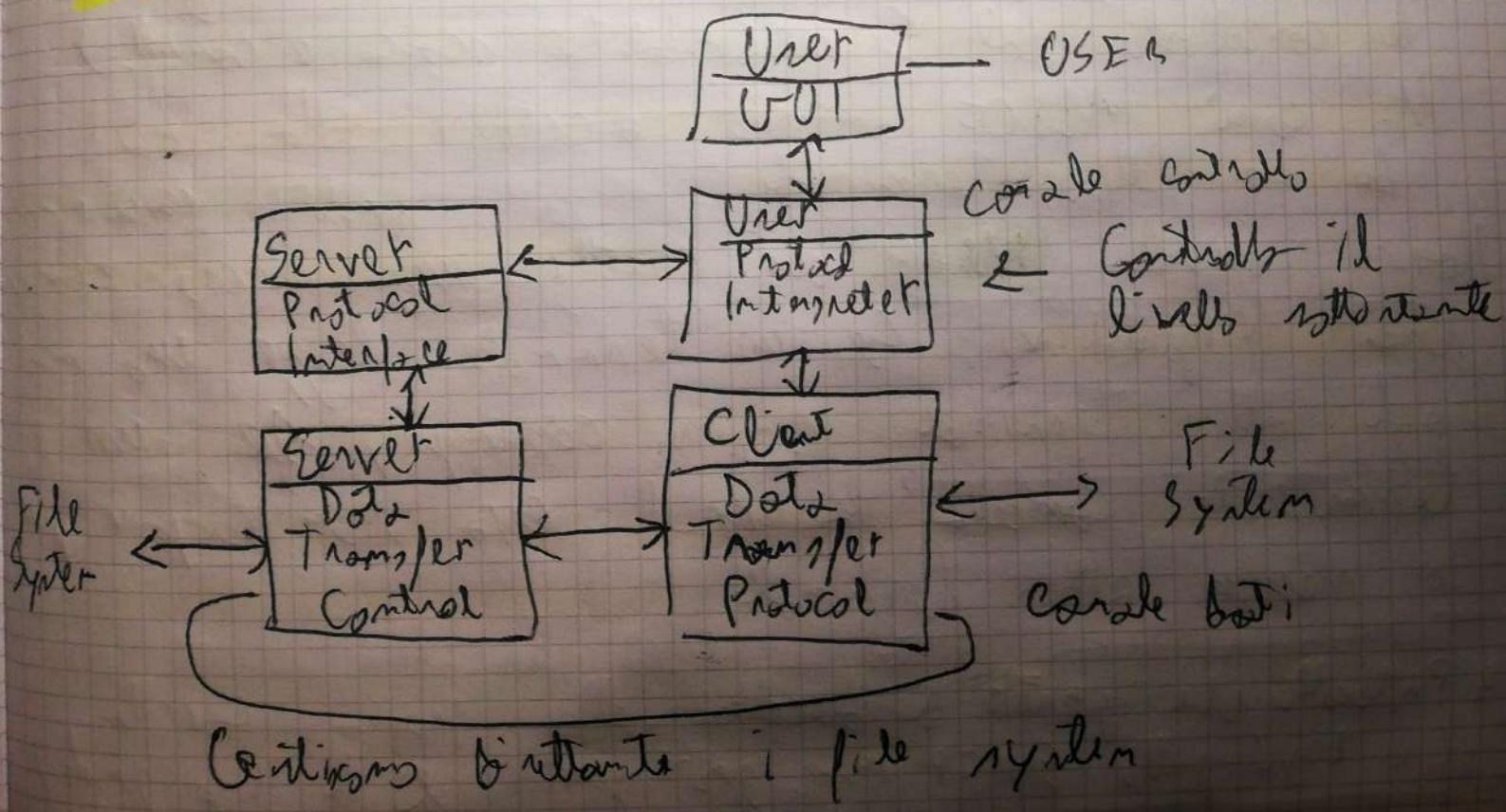
Usare due porte separate; una per i dati e una per i messaggi. Una autenticazione utente (in chiaro) + controllo.



Lo stesso è lavoro simile al controllo di HTTP

con la rete TCP, qui però non c'è proxy  
+ cookie

E' un protocollo Pull e Push [ Passo graduale file nel server remoto e viceversa ]  
Non è nulla questo protocollo per motivo di sicurezza  
Esistono varianti di FTP che proteggono da accessi  
senza autenticazione

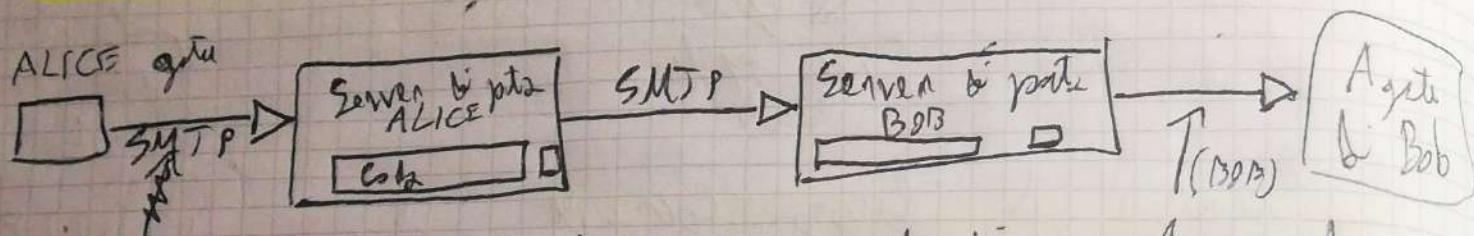


Certificarsi di entrando i file system

**SNTP**, è protocollo **Push**  
**protocollo per le email**

Nel senso che i posti sono  
generati da colo di rete  
e le corrispondenti.  
Server di posti possono fare il  
client del rete.

Utente ha account nello macchinari, riceve messaggi  
che viene messo ~~dai~~ tra spazio nello contenitore  
di un altro server di posti.



Ora ci sono i protocolli che fanno di accedere al  
server di posta di ~~ALICE~~ BOB

**SNTP** gestisce comunicazioni tra server di posti

Un protocollo tipo la segnale è **Server push**

Come ad esempio **POP**.

E' occupata di far ricevere la mail nell'agente e  
di cancellare dal server (mobilità servizio e cancella)

Ma come ALICE non può accedere alla mail  
da un altro terminal.

(mobilità servizio e cancella)

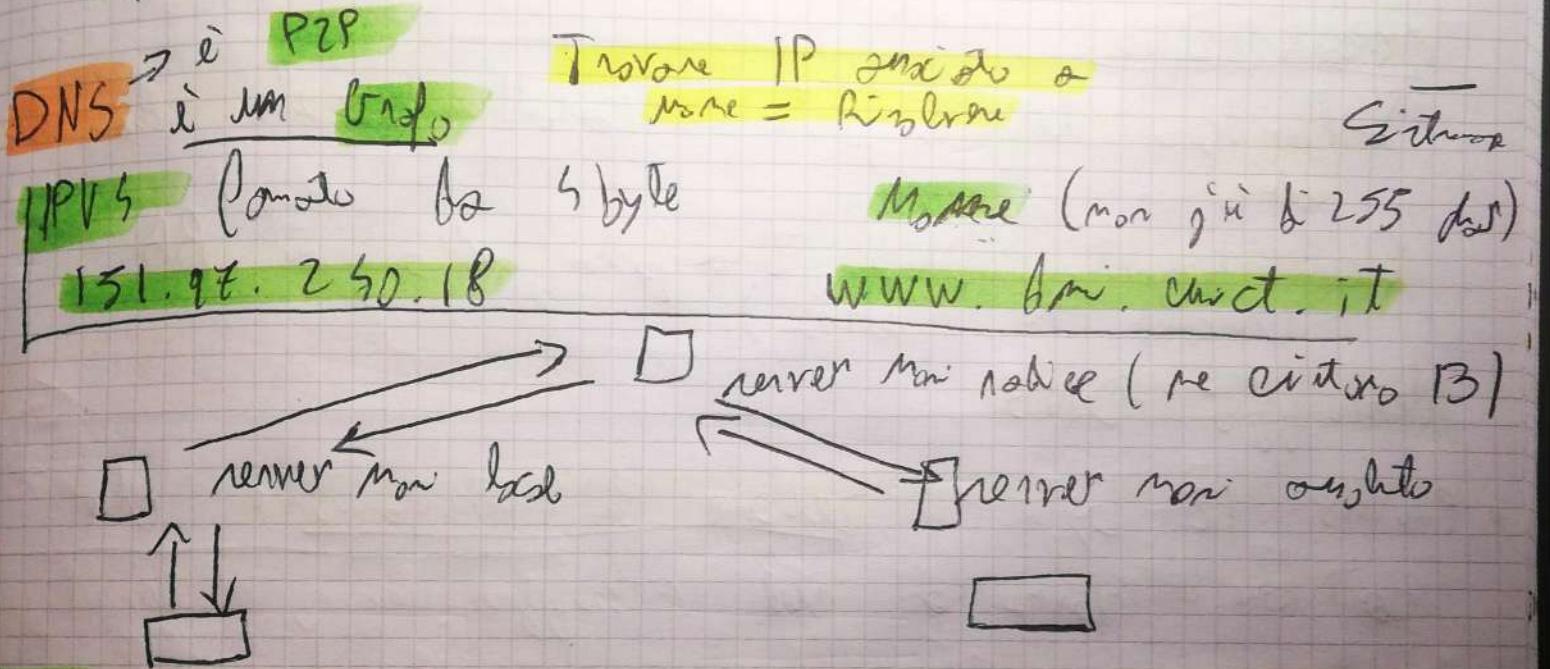
Nel caso quindi **IMAP** che garantisce una sincronizzazione  
tra i due punti. Se un terminale legge la mail  
dal altro terminal comando comunque, AR un  
terminale cancella la mail che non c'è più  
in nessun altro terminale.

\* Protocollo che garantisce di accedere allo stesso spazio  
(è tipo pull, SNTP non può gestire il tip push)

I server di posta conservano tra loro col SMTP  
informazioni messaggi (lettere) codici per segnalare l'oggetto  
di seguito nel protocollo un modo per finire i messaggi.  
Ex: end with <CRLF>. <CRLF>

Mime: standard che permette di estendere la definizione  
di formato dei messaggi di posta elettronica

SMTP supporta anche strumenti per ~~risolvere~~ rilevare le  
spame e bloccarla.



Si cercano e creare collegamenti  
allo stesso layer

Si crea in Server non intemto tra server non  
load e server di non andato per collegare  
in maniera più diretta

DNS non è client-server ma P2P cioè ogni  
macchina è sia server che client

L'idea del DNS è "ne non si niente un collegamento Web e chi sta sopra"

I dati sono composti come un albero per collegarsi ma in realtà è un grafo.  
Il tipo DNS è molto gerarchico, si suddivide in  
DNS privati e pubblici.

(I record ~~sono~~ servono a individuare il traffico)

Si utilizza sempre un iteratore di coding

### Record DNS:

Definiscono le diverse esistenze dei dati

Ogni risposta DNS trasporta logici RR  
(record di risposta)

### Messaggio DNS

- Primi 12 byte: Sezione di Introduzione che contiene informazioni sul messaggio
- Sezione delle domande: ha informazioni sulle richieste effettuate
- Sezione delle risposte: ha gli RR in risposta alle query
- Sezione autoritative: ha i record di altri server autorizzati
- Sezione aggiuntive: ha altri record d'aiuto

Il DNS può servire per monitorare il traffico

DNS non è molto sicuro, è soggetto ad attack  
e quando i dati in chiave

## SNMP: Simple Network Management Protocol

è un protocollo per la monitoraggio di un rete  
o altro.

Si basa sull'agent e manager che controllano  
ogni altra unità nel DB MIB.

Il manager può interrogare (get) ma anche  
modificare (set)

Gli agent interagiscono col SO e possono inviare  
anche trap

Questi sono per gestione / monitoraggio in remoto

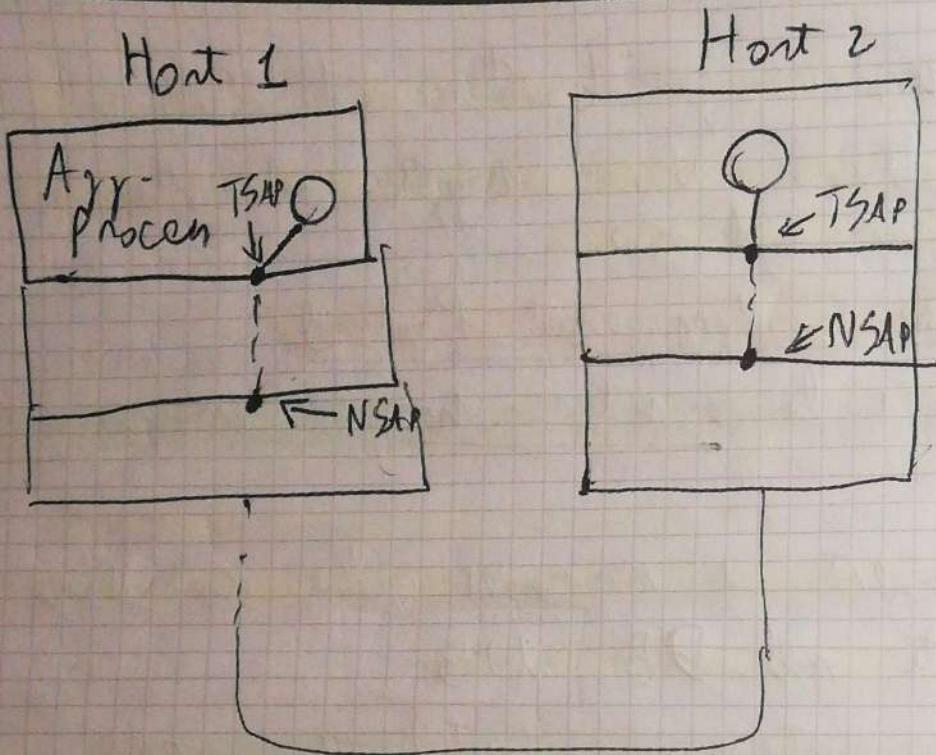
## Livello di Transporto

Unità / Segnale parte dalla LSI a parte  
bus e LSI

Il livello di trasporto facilita la comunicazione logica  
tra gli host, soltanto le problematiche della  
comunicazione di rete da coinvolgere; si riporta  
nel mezzo nel collegamento

TSAP: collega App a Transport, qui inizia la  
trasport connection

NSAP: collega Trasport a Network, qui inizia la  
network connection



Sostanzialmente il TSAP sostituisce la Porta, collega App. a Transport

Dunque c'è una comunicazione, nel registro rete  
il livello di trasporto è uno dei punti di determinazione che lo si vuole

Il Socket che ageva la comunicazione è individuato  
da 5 parametri (Connector-socket)

- Porta src .IP src
- Porta dest .IP dest
- Type socket

*Multiplexing e Demultiplexing*  
e pag 180-185

Con questi 5 parametri il web server può fare il multiplexing per la comunicazione in parallelo con più client

## 3.2 Multiplexing e demultiplexing

In questo paragrafo analizzeremo il multiplexing e il demultiplexing, cioè come il servizio di trasporto da host a host fornito dal livello di rete possa diventare un servizio di trasporto da processo a processo per le applicazioni in esecuzione sugli host. Per praticità tratteremo questo servizio di base del livello di trasporto nel contesto di Internet, anche se il servizio di multiplexing e demultiplexing è presente in tutte le reti di calcolatori.

Nell'host destinatario il livello di trasporto riceve segmenti dal livello di rete immediatamente sottostante. Il livello di trasporto ha il compito di consegnare i dati di questi segmenti al processo applicativo appropriato in esecuzione nell'host. Consideriamo un esempio. Supponiamo che vi troviate di fronte al vostro computer e che stiate scaricando pagine web mentre sono in esecuzione una sessione FTP e due sessioni Telnet. Avrete pertanto quattro processi applicativi in esecuzione: due Telnet, uno FTP e uno HTTP. Il livello di trasporto nel vostro calcolatore, quando riceve dati

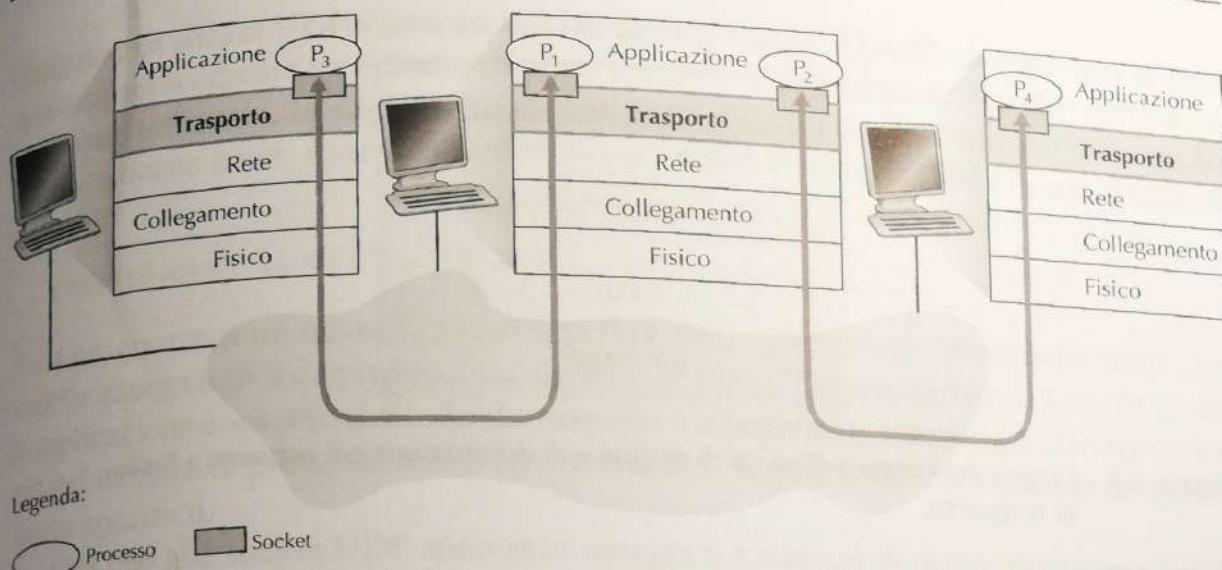
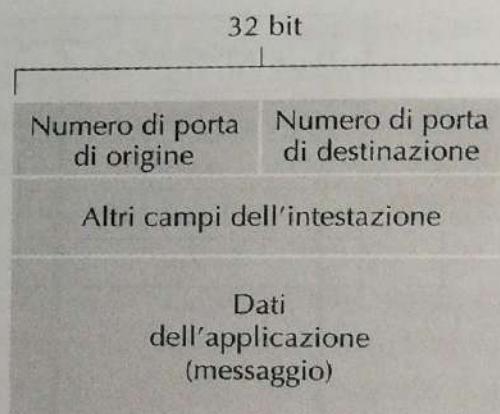


Figura 3.2 Multiplexing e demultiplexing a livello di trasporto.

dal livello di rete sottostante, li deve indirizzare a uno di questi quattro processi. Esaminiamo come ciò venga realizzato.

Innanzitutto ricordiamo (Paragrafo 2.7) che un processo (come parte di un'applicazione di rete) può gestire una o più **socket**, attraverso le quali i dati fluiscano dalla rete al processo e viceversa. Di conseguenza (Figura 3.2) il livello di trasporto nell'host di ricezione in realtà non trasferisce i dati direttamente a un processo, ma piuttosto a una socket che fa da intermediario. Siccome, a ogni dato istante, può esserci più di una socket nell'host di ricezione, ciascuna avrà un identificatore univoco il cui formato dipende dal fatto che si tratti di socket UDP o TCP, come vedremo tra breve.

Consideriamo ora come l'host in ricezione indirizzi verso la socket appropriata il segmento a livello di trasporto in arrivo. Ciascun segmento a livello di trasporto ha vari campi deputati allo scopo. **Lato ricevente**, il livello di trasporto esamina questi campi per **identificare la socket** di ricezione e quindi vi dirige il segmento. Il compito di trasportare i dati dei segmenti a livello di trasporto verso la giusta socket viene detto **demultiplexing**. Il compito di radunare frammenti di dati da diverse socket sull'host di origine e incapsularne ognuno con intestazioni a livello di trasporto (che verranno poi utilizzate per il demultiplexing) per creare dei segmenti e passarli al livello di rete, viene detto **multiplexing**. Si noti che il livello di trasporto nell'host centrale della Figura 3.2 deve effettuare il demultiplexing dal livello di rete di segmenti che possono arrivare sia per il processo  $P_1$  sia per  $P_2$ ; ciò avviene indirizzando i dati del segmento in ingresso alla giusta socket. Il livello di trasporto nell'host centrale deve, inoltre, raccogliere i dati in uscita dalle socket dei due processi, creare i segmenti a livello di trasporto e passarli al livello di rete. Sebbene abbiamo introdotto il multiplexing e il demultiplexing nel contesto dei protocolli di trasporto Internet, è importante rendersi conto che queste funzioni hanno uno specifico interesse ogni volta che un protocollo a un qualsiasi livello (di trasporto o qualsiasi altro) è utilizzato da più entità al livello immediatamente superiore.



**Figura 3.3** I campi del numero di porta di origine e di destinazione nei segmenti a livello di trasporto.

Per mostrare il compito del multiplexing e del demultiplexing richiamiamo l'analogia del paragrafo precedente. Anna effettua un'operazione di multiplexing quando raccolge le lettere dai mittenti e le imbuca. Nel momento in cui Andrea riceve le lettere dal postino, effettua un'operazione di demultiplexing leggendo il nome riportato sopra la busta e consegnando ciascuna missiva al rispettivo destinatario.

Ora che abbiamo compreso i ruoli del multiplexing e del demultiplexing a livello di trasporto esaminiamo come vengono realizzati negli host. Da quanto visto in precedenza, sappiamo che il multiplexing a livello di trasporto richiede (1) che le socket abbiano identificatori unici e (2) che ciascun segmento presenti campi che indichino la socket cui va consegnato il segmento. Questi (Figura 3.3) sono il campo del numero di porta di origine e il campo del numero di porta di destinazione. I segmenti UDP e TCP presentano anche altri campi, come vedremo più avanti. I numeri di porta sono di 16 bit e vanno da 0 a 65535, quelli che vanno da 0 a 1023 sono chiamati numeri di porta noti (*well-known port number*) e sono riservati per essere usati da protocolli applicativi ben noti quali HTTP (porta 80) e FTP (porta 21). L'elenco dei numeri di porta noti è fornito nell'RFC 1700: la sua versione aggiornata è consultabile tramite <http://www.iana.org> [RFC 3232]. Quando si sviluppa una nuova applicazione, è necessario assegnarle un numero di porta.

Ora dovrebbe essere chiaro come il livello di trasporto possa implementare il servizio di demultiplexing: ogni socket nell'host deve avere un numero di porta, e quando un segmento arriva all'host il livello di trasporto esamina il numero della porta di destinazione e dirige il segmento verso la socket corrispondente. I dati del segmento passano, quindi, dalla socket al processo assegnato. Come vedremo, questo è fondamentalmente il modo in cui agisce UDP, mentre il multiplexing/demultiplexing TCP è ancora più raffinato.

### Multiplexing e demultiplexing non orientati alla connessione

Ricordiamo, dal Paragrafo 2.7.1, che i programmi Python in esecuzione possono creare una socket UDP con l'istruzione

```
clientSocket = socket(AF_INET, SOCK_DGRAM)
```

Quando una socket UDP viene definita in questo modo, il livello di trasporto le assegna automaticamente un numero di porta compreso tra 1024 e 65535 che non sia ancora stato utilizzato. In alternativa, un programma Python potrebbe creare una socket UDP associata a uno specifico numero di porta (per esempio, 19157) con il metodo `bind()`:

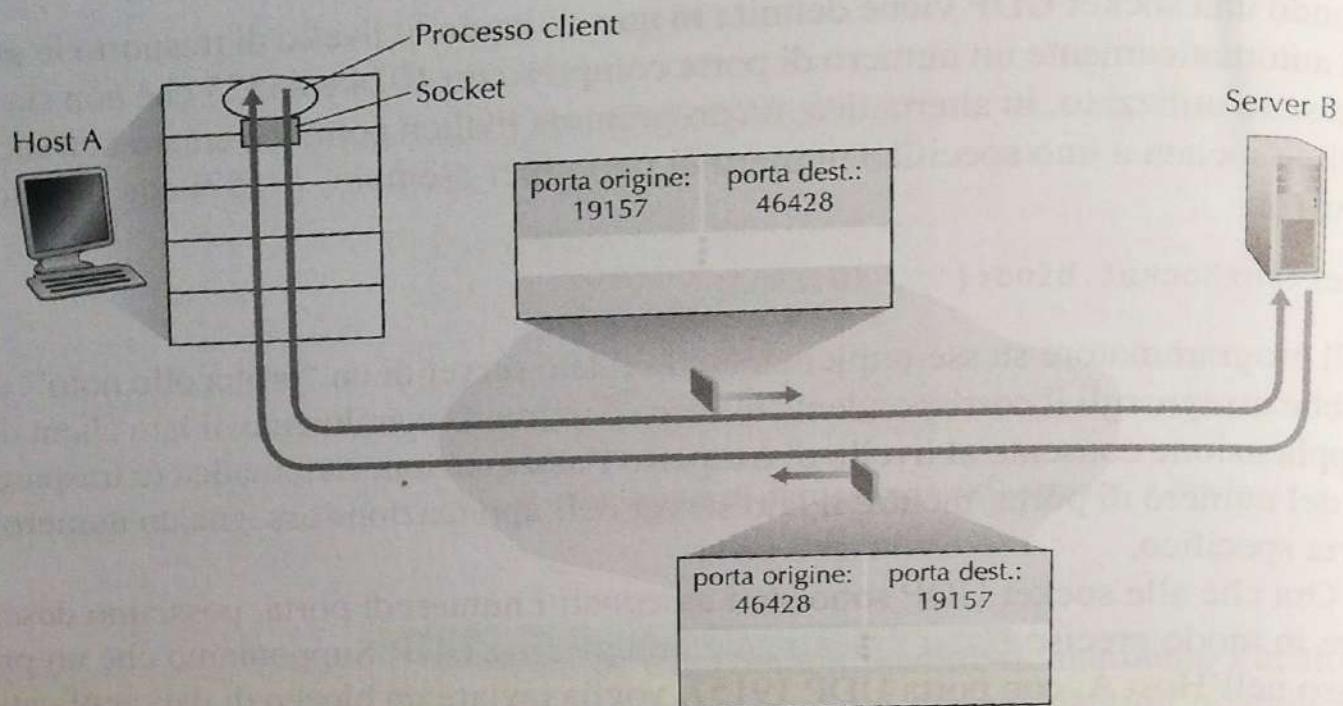
```
clientSocket.bind(('', 19157))
```

Se il programmatore stesse implementando il lato server di un “protocollo noto”, dovrebbe assegnargli il corrispondente numero di porta. Generalmente, il lato client dell’applicazione consente al livello di trasporto l’assegnazione automatica (e trasparente) del numero di porta, mentre il lato server dell’applicazione assegna un numero di porta specifico.

Ora che alle socket UDP sono stati assegnati i numeri di porta, possiamo descrivere in modo preciso il multiplexing/demultiplexing UDP. Supponiamo che un processo nell’Host A, con porta UDP 19157, voglia inviare un blocco di dati applicativi a un processo con porta UDP 46428 nell’Host B. Il livello di trasporto di A crea un segmento che include i dati applicativi, i numeri di porta di origine (19157) e di destinazione (46428) e due altri valori (che non sono importanti per l’attuale discussione e verranno trattati più avanti). Il livello di trasporto passa, quindi, il segmento risultante al livello di rete, che lo incapsula in un datagramma IP, ed effettua un tentativo best-effort di consegna del segmento all’host in ricezione. Se il segmento arriva all’Host B, il suo livello di trasporto esamina il numero di porta di destinazione del segmento (46428) e lo consegna alla propria socket identificata da 46428. Osserviamo che l’Host B potrebbe avere in esecuzione più processi, ciascuno con la propria socket UDP e relativo numero di porta. Quando i segmenti UDP giungono dalla rete, l’Host B dirige ciascun segmento (ossia ne esegue il demultiplexing) alla socket appropriata esaminando il numero di porta di destinazione del segmento.

È importante notare che una socket UDP viene identificata completamente da una coppia che consiste di un indirizzo IP e di un numero di porta di destinazione. Di conseguenza, se due segmenti UDP presentano diversi indirizzi IP e/o diversi numeri di porta di origine, ma hanno lo stesso indirizzo IP e lo stesso numero di porta di destinazione, saranno diretti allo stesso processo di destinazione tramite la medesima socket.

Per quanto riguarda il numero di porta di origine, osserviamo la Figura 3.4 in cui, nel segmento che va da A verso B, il numero di porta di origine serve come parte di un “indirizzo di ritorno”: quando B vuole restituire il segmento ad A, la porta di destinazione del segmento da B verso A assumerà il valore della porta di origine del segmento da A verso B. L’indirizzo di ritorno completo è costituito dall’indirizzo IP di A più il numero di porta di origine. Per esempio, vediamo il programma server UDP studiato nel Paragrafo 2.7. In `UDPServer.py`, il server utilizza il metodo `recvfrom()` per estrarre il numero di porta di origine dal segmento ricevuto dal client; poi invia un nuovo segmento al client, in cui il numero di porta di origine estratto viene usato come numero di porta di destinazione del nuovo segmento.



**Figura 3.4** Inversione dei numeri di porta di origine e di destinazione.

### Multiplexing e demultiplexing orientati alla connessione

Per comprendere il **demultiplexing TCP** dobbiamo analizzare da vicino le socket TCP e il modo in cui si stabiliscono le connessioni TCP. Una sottile differenza tra una socket TCP e una socket UDP risiede nel fatto che la prima è identificata da quattro parametri: indirizzo IP di origine, numero di porta di origine, indirizzo IP di destinazione e numero di porta di destinazione. Pertanto, quando un segmento TCP giunge dalla rete in un host, quest'ultimo utilizza i quattro valori per dirigere (fare demultiplexing) il segmento verso la socket appropriata. In particolare, e al contrario di UDP, due segmenti TCP in arrivo, aventi indirizzi IP di origine o numeri di porta di origine diversi, vengono diretti a due socket differenti, anche a fronte di indirizzo IP e porta di destinazione uguali, con l'eccezione dei segmenti TCP che trasportano la richiesta per stabilire la connessione. Per chiarire meglio questo aspetto riconsideriamo l'esempio del Paragrafo 2.7.2.

- L'applicazione server TCP presenta una “socket di benvenuto” che si pone in attesa di richieste di connessione da parte dei client TCP (Figura 2.29) sulla porta numero 12000.
- Il client TCP crea una socket e genera un segmento per stabilire la connessione tramite le seguenti linee di codice:

```
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverName, 12000))
```

- Una richiesta di connessione non è nient'altro che un segmento TCP con numero di porta di destinazione 12000 e uno speciale bit di richiesta di connessione posto a 1 nell'intestazione (Paragrafo 3.5). Il segmento include anche un numero di porta di origine, scelto dal client.

- Il sistema operativo dell'host che esegue il processo server, quando riceve il segmento con la richiesta di connessione con porta di destinazione 12000, localizza il processo server in attesa di accettare connessioni sulla porta 12000. Il processo server crea quindi una nuova connessione:  
`connectionSocket, addr = serverSocket.accept()`
- Inoltre il livello di trasporto sul server prende nota dei seguenti valori nel segmento con la richiesta di connessione: (1) numero di porta di origine nel segmento, (2) indirizzo IP dell'host di origine, (3) numero di porta di destinazione nel segmento e (4) il proprio indirizzo IP. La socket di connessione appena creata viene identificata da questi quattro valori. Tutti i segmenti successivi la cui porta di origine, indirizzo IP di origine, porta di destinazione e indirizzo IP di destinazione coincidono con tali valori verranno diretti verso questa socket. Ora che la connessione TCP è attiva, client e server possono scambiarsi dati.

L'host server può ospitare più socket TCP contemporanee collegate a processi diversi, ognuna identificata da una specifica quaterna di valori. Quando il segmento TCP arriva all'host, i quattro campi citati prima vengono utilizzati per dirigere (fare demultiplexing) il segmento verso la socket appropriata.

La situazione è schematizzata nella Figura 3.5 dove l'Host C dà inizio a due sessioni HTTP verso il server B, mentre l'Host A apre una sessione verso B. Gli Host A

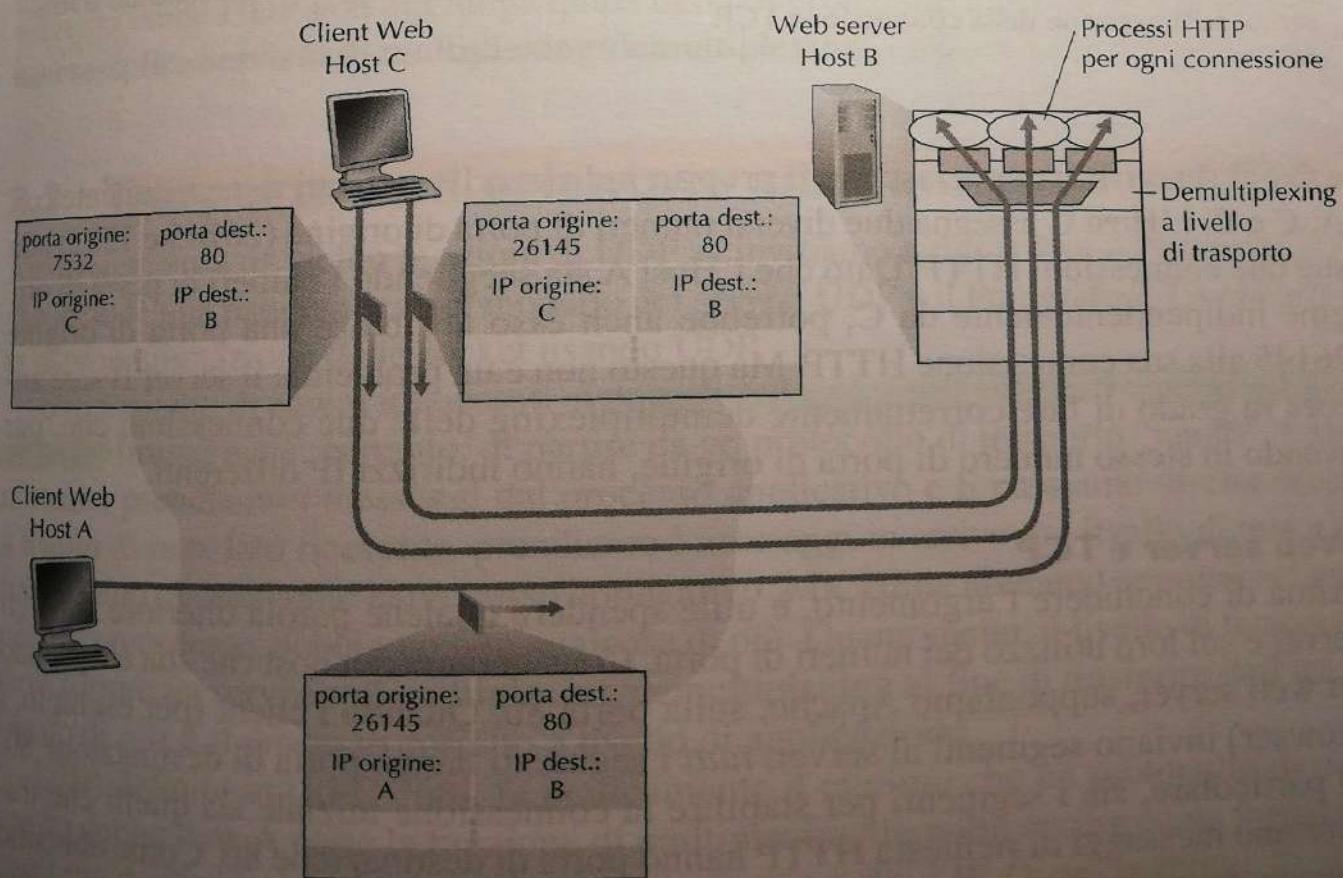


Figura 3.5 Due client che usano lo stesso numero di porta di destinazione (80) per comunicare con la stessa applicazione sul web server.

UDP è un protocollo di Transporto

- Il protocollo plusso basso è lo strato dell'applicativo
- Non gestisce la connessione, non occupa slot di memoria ricevuti. Si può trasferire a livello applicativo
- Non c'è stato
- Non c'è overhead di gestione

Il suo ruolo è insieme Source port e Dest port  
UDP è un servizio bottegaio alle applicazioni  
cioè i segmenti sono passati immediatamente al livello  
di rete e UDP non ha protocollo End-to-end

Dunque UDP non è orientato alla connessione

### Struttura Segmento UDP

Num src-port	Num dest-port
Lung	Checksum
Messaggio	

Intestazione  
è composta da 2 byte

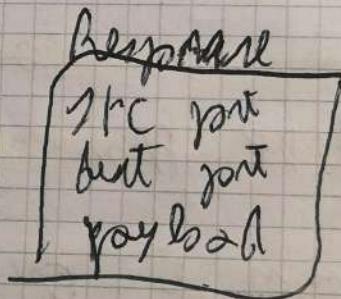
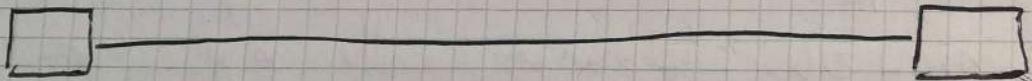
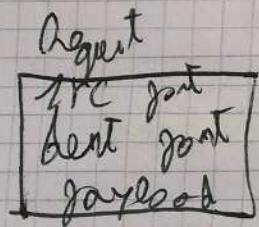
- I numeri di porta permettono di fare nel modo corretto il Demultiplexing
- Lunghezza: definisce in byte la lung. del segmento
- Checksum: usato per verificare se ci sono errori nel segmento. (il checksum è calcolato)

Il checksum controlla l'osservazione XOR per il controllo (superficie) degli errori.

Il checksum serve per il rilevamento errori, è usato per vedere se i bit delle regole sono stati alterati durante il trasferimento.

Collegamento

Client



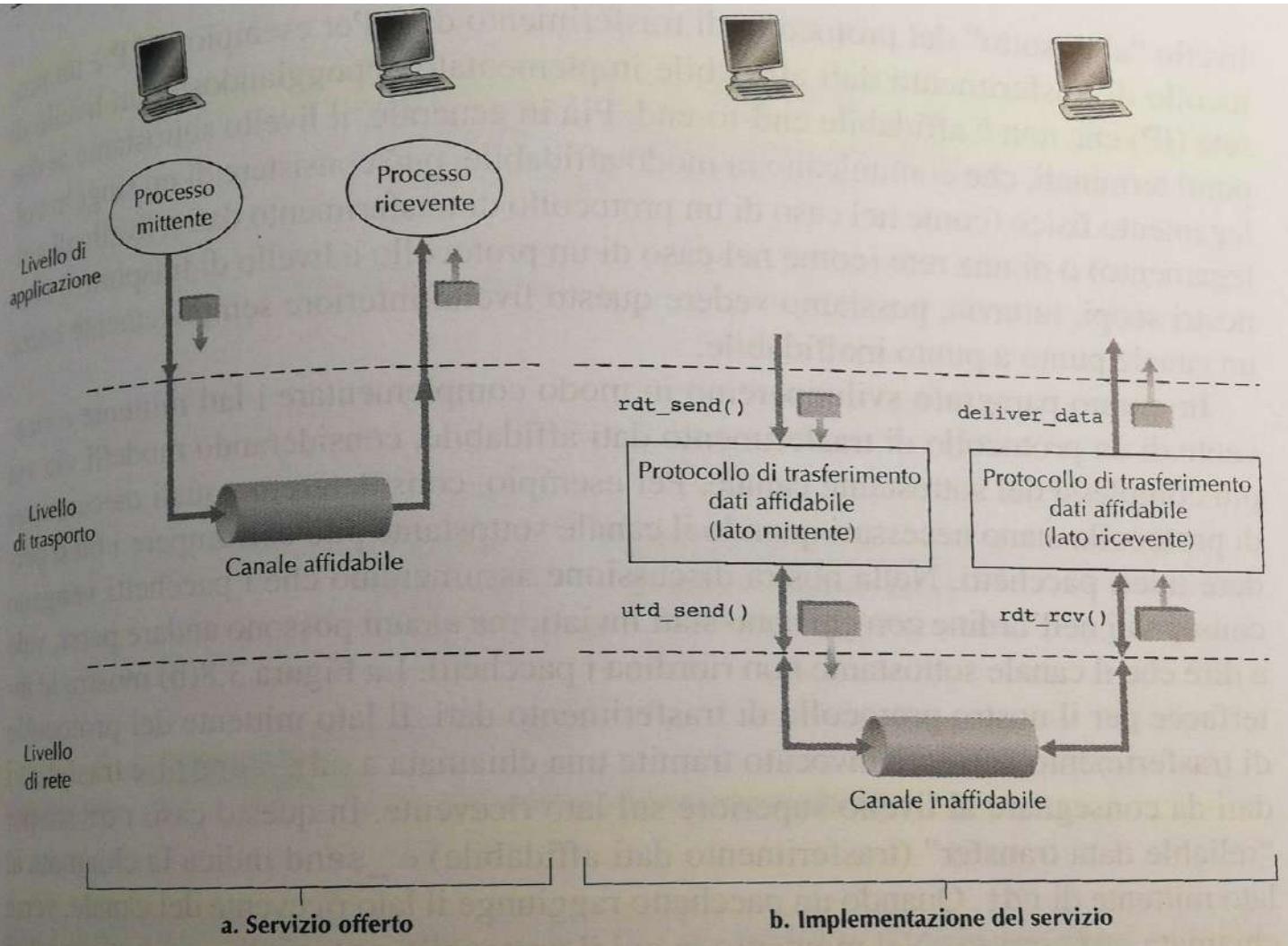
Cerchiamo di implementare un sistema affidabile di comunicazione. (verso TCP) SCHEMA PAG. 183-185

RDT (1.0)

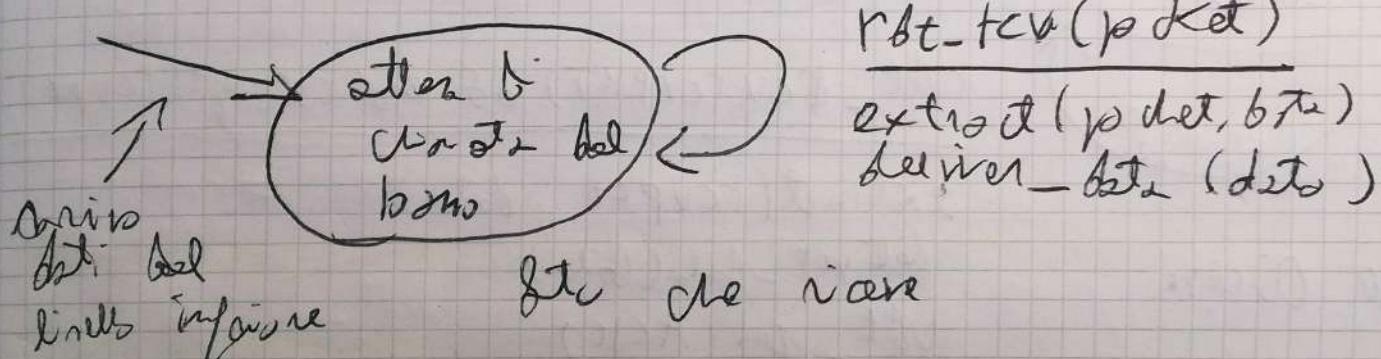
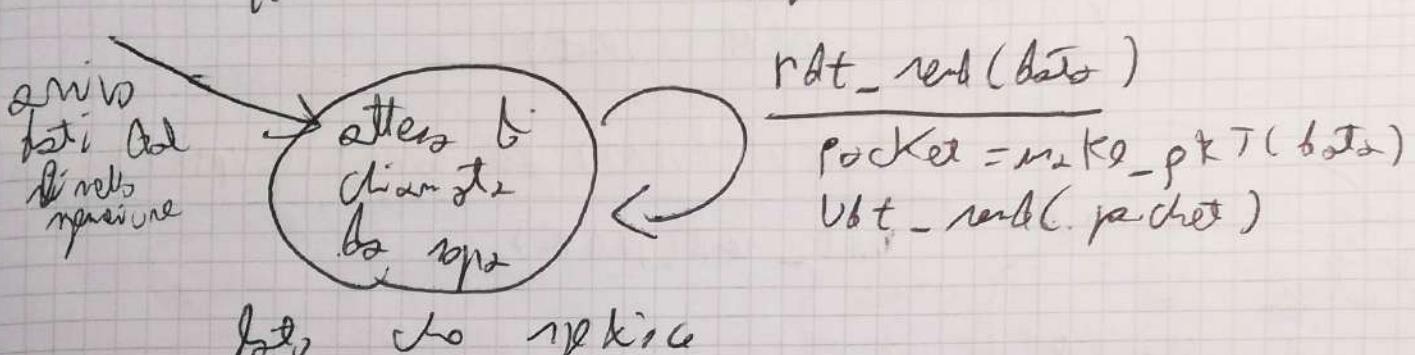
(Reliable data transfer)

Implementare come un affidabile protocollo di comunicazione.

Come un affidabile servizio gestito (implementazione)



**Figura 3.8** Trasferimento dati affidabile: modello di servizio e implementazione.



Ri scrive già insieme emoji nei pacchetti  
= li battono gettino

Quindi il receiver manda

ACK se il pacchetto è OK  
NACK // // // e' male

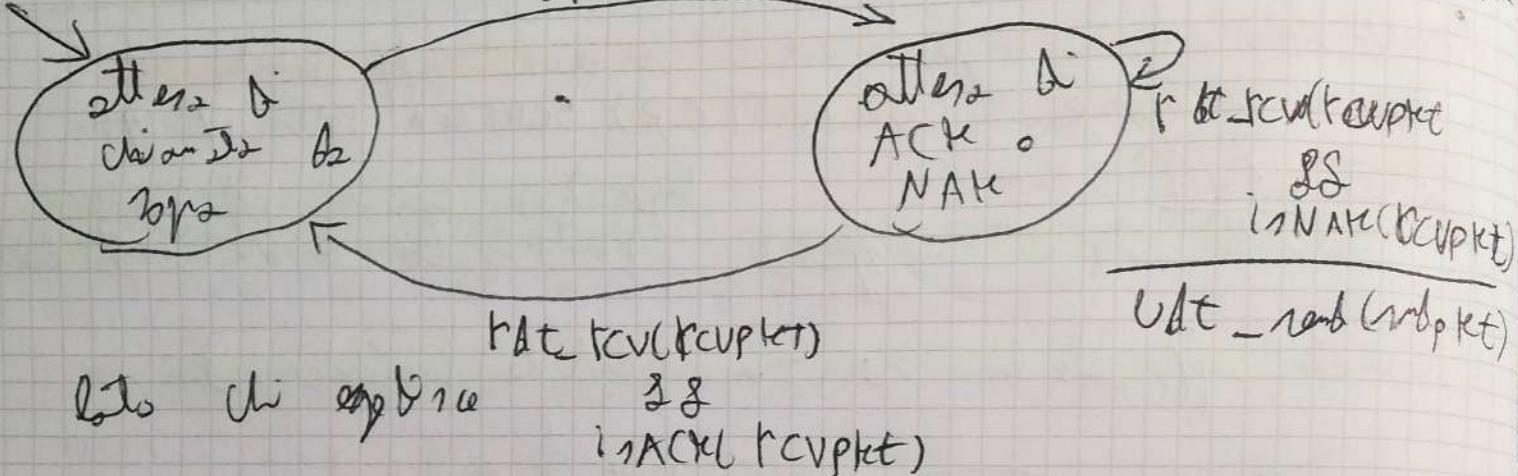
BDT 2.0 = Canale con errori ma senza perdite

I pacchetti arrivano regolare, ma possono essere errati  
ma non mancano

Serve funzione per escludere o controllare  
checksum

rat-read(data)

rat-read(subpkt) = rat-read(data, checksum) Udt-read(subpkt)



rat-tcv(rcvpkt) & convert(rcvpkt)

Udt-read(NAK)

rat-tcv(rcvpkt) & convert(rcvpkt)

extract(rcvpkt, data)

deliver-data(data)

Udt-read(ACK)

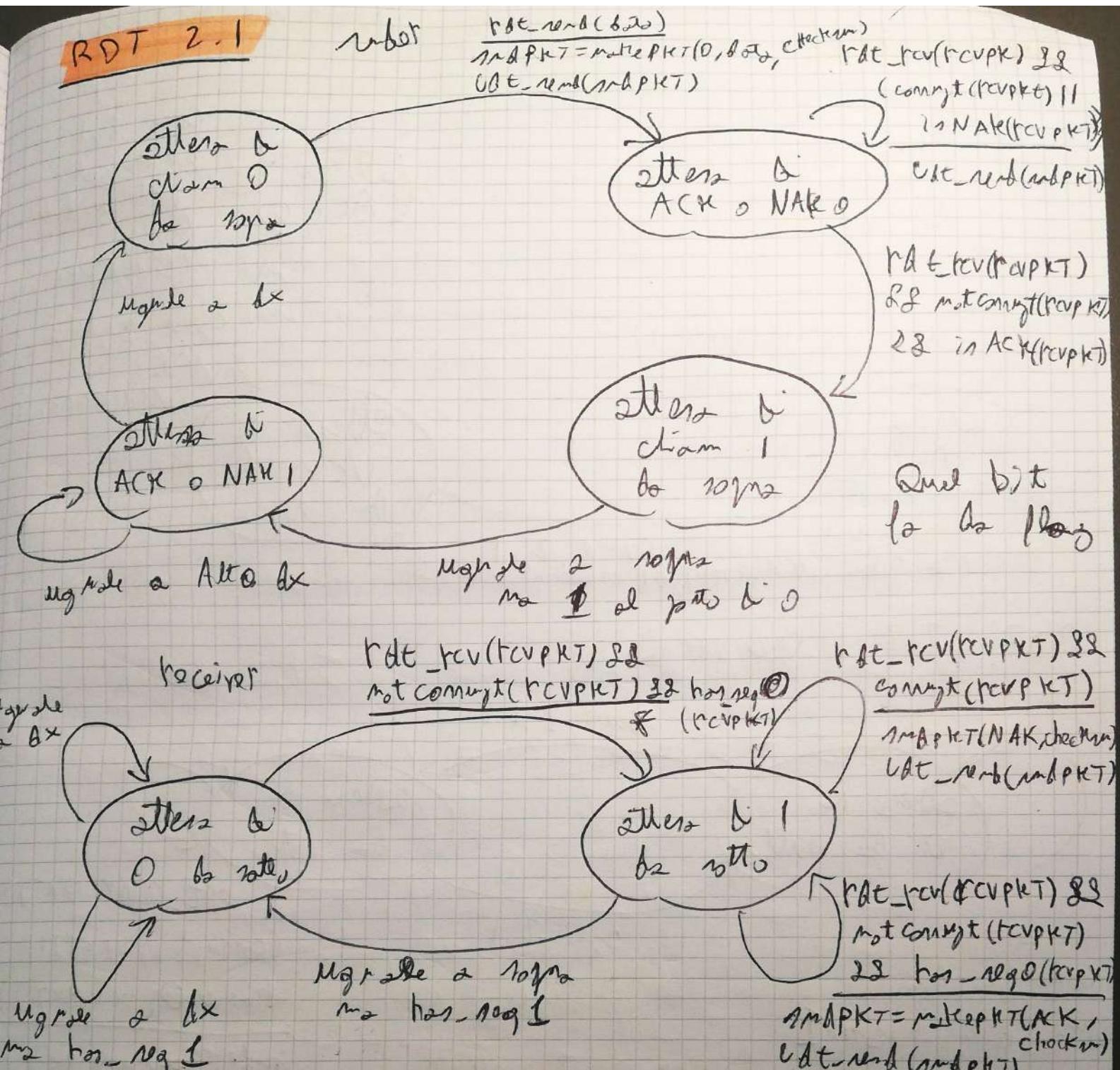
lato che riceve

Cosa succede se ACK e NAK sono inviati con  
errore?

Il writer si blocca

Miglioriamo il protocollo ottimizzando ACK e NAK

## RDT 2.1



Antico rischio alla duplicazione dei pacchetti: ACK e NAK  
 I NAK possono essere eritati rispettando l'ACK  
 Relativo all'ultimo pacchetto ricevuto. Sono comunque

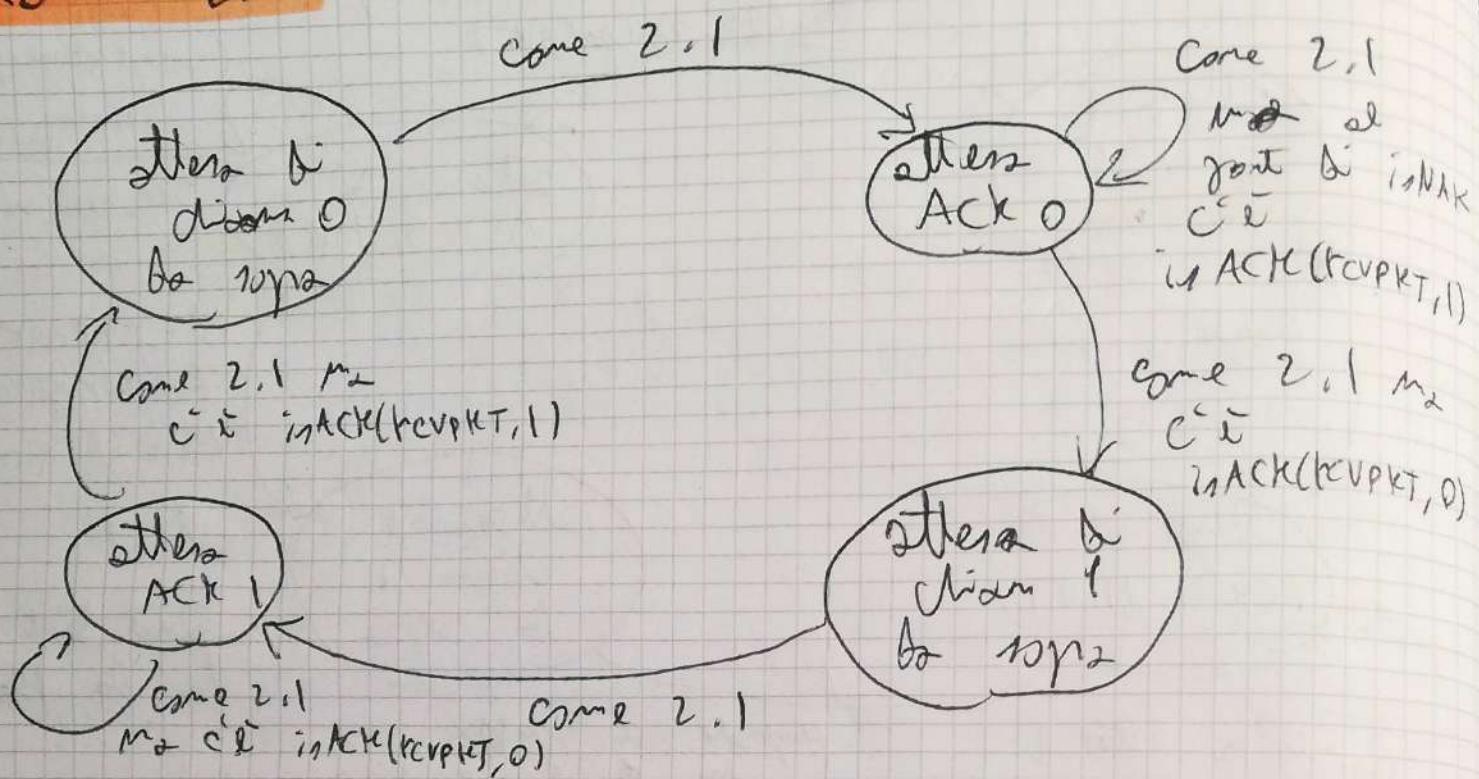
extraJ(rcvpkt, b<sub>22</sub>)  
 deliverB22(b<sub>22</sub>)  
 mdpkt=make\_PKT(ACK, checksum)  
 vdt\_send(mdpkt)

0 e 1 sono per  
 distinguere tra pacchetti  
 ricevuti e nuovi

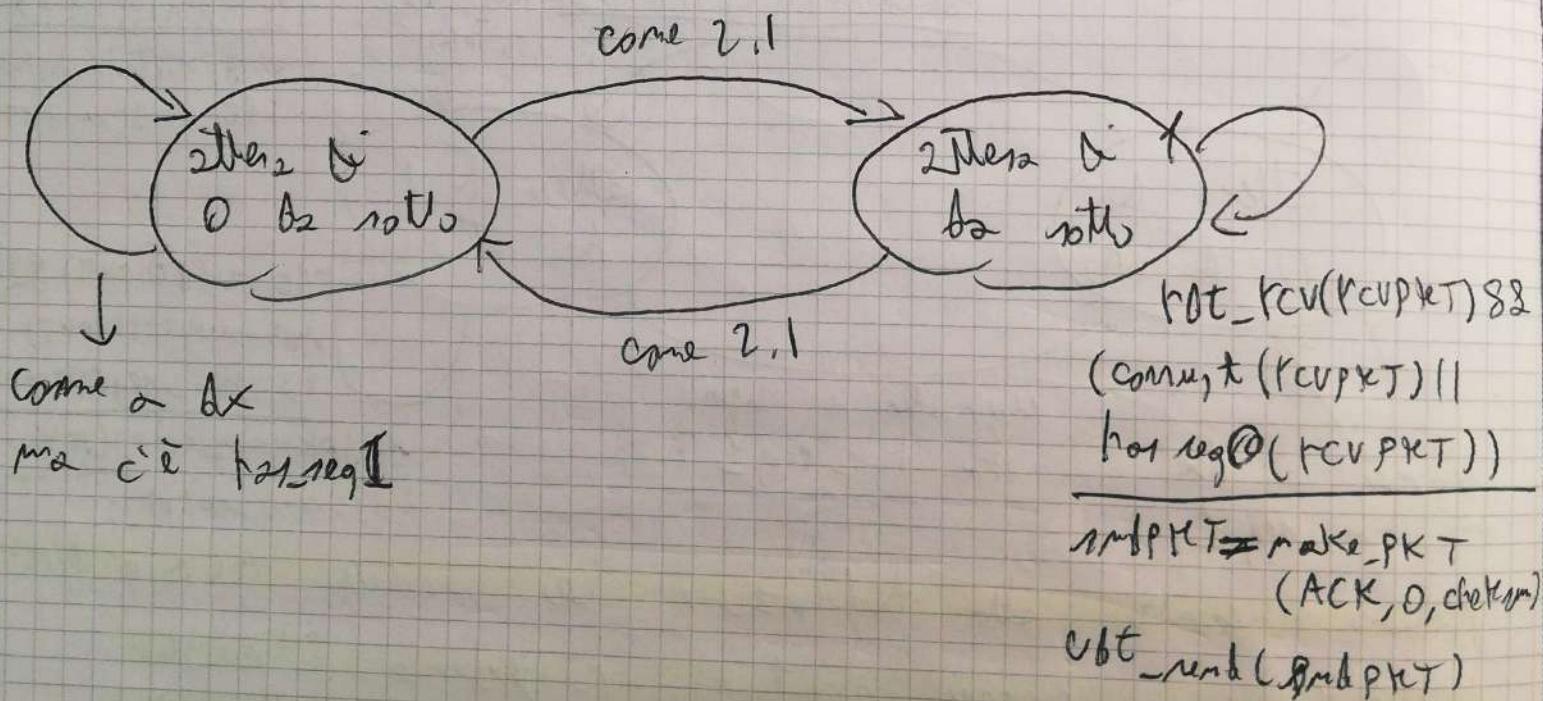
RDT 2.2

remember

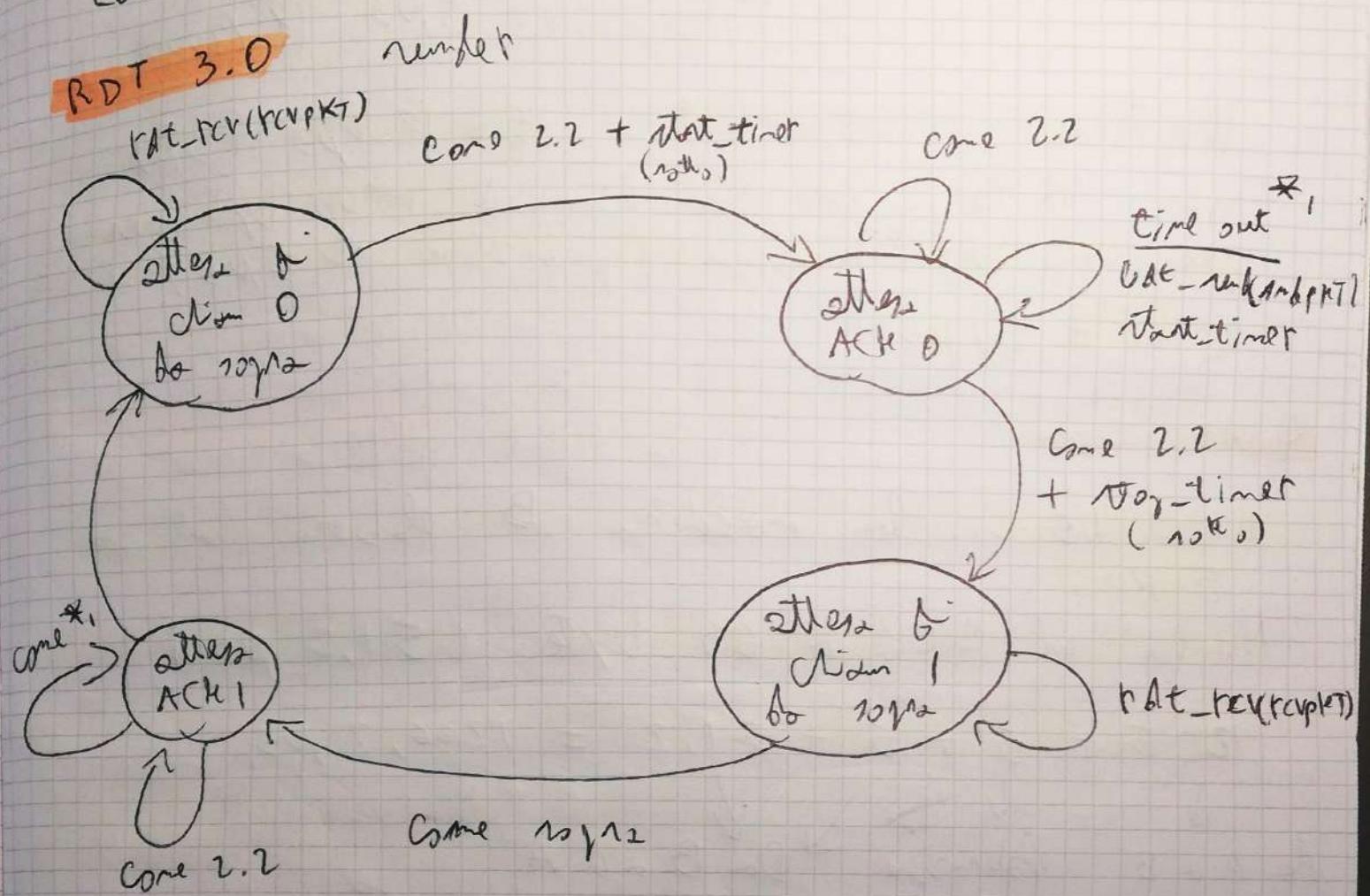
The function is NATE now or does it



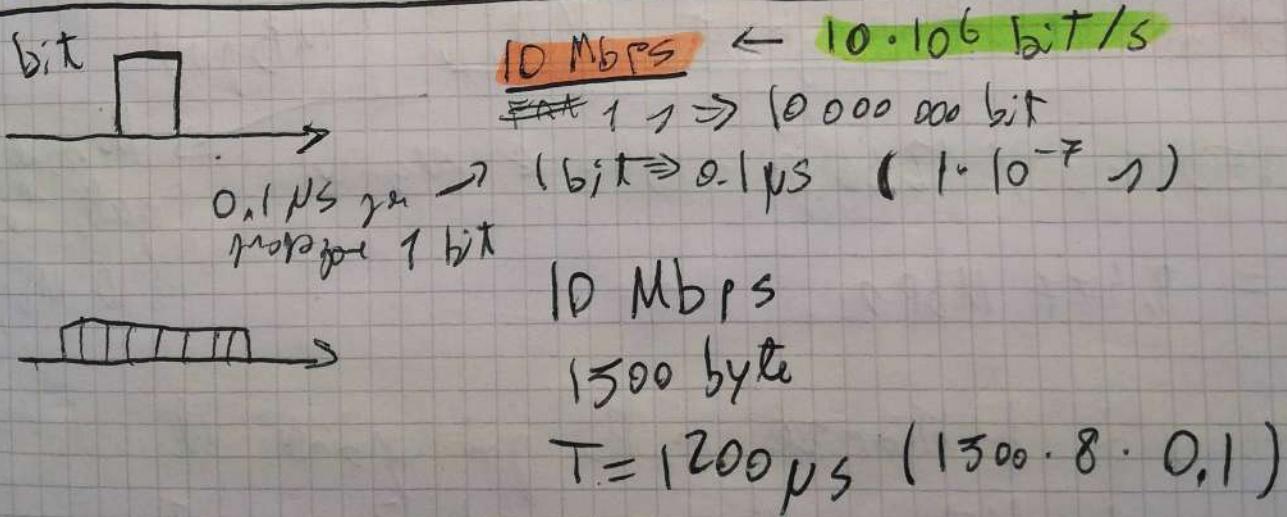
RDT 2.2 receiver



ADM, considerando la possibilità di perdita  
pacchetti da gestire nel timer  
LOSSY CHANNEL WITH BIT ERRORS



If a de forms were measured, for cosine to repeat a QMATE with its neighbours



(A)

(B)

$$RTT = 10 \text{ ns}$$

1 Km

$$v = 200,000 \text{ km/s} \quad \checkmark$$

$$T_{\text{propagazione}} = 5 \text{ ns} \quad (1 / 200,000)$$

$$\text{Il frame A: } 1500 \text{ byte} \quad t = 1200 \mu\text{s} + 5 \text{ ns}$$

Ora si calcola il tempo per lo frame B: 1500 byte per  
giungere in Km (con propagazione di 5 ns) e  
(1200 + 5) ns

### Summarizzazione

1205 ns ... lo modifica B elaborazione in  
5 ns.

Numero in ACK di 64 byte (51,2 ns)

$$1205 + 5 + 5 + 51,2 = 1266,2 \text{ ns}$$

$\downarrow$   $\downarrow$   $\downarrow$   $\downarrow$   
Da A a B elaborazione  $\rightarrow$  Da B a A

Dunque se avessi voluto fare un timer basedoc in  
queste caratteristiche sarebbe dovuto essere più  
di 1266,2 ns

Calcolare il throughput effettivo (MB/s)

$$\begin{aligned} \rightarrow & 1500 \cdot 8 / (1266,2 \cdot 10^6) \leftarrow \text{risultato} \rightarrow \text{Numero di quadri} \\ & = 9,577 \text{ Mbps} \quad \text{(10 Mbps)} \end{aligned}$$

Metterebbero a distanza molto più grande il  
throughput effettivo diminuendo drasticamente

Questo protocollo è in cui trasmetti, invia e aspetti  
l'ACK e fatto Stop and Wait

Il mittente non invia messaggi dati finché non è  
certo che il destinatario abbia ricevuto correttamente  
il pacchetto corrente

RTT = tempo ritardo di propagazione + attesa e  
ritorno

R: tempo tra invio del codice

L: dimensione pacchetto

Tempo richiesto per trasmettere ~~un bit~~ il  
pacchetto sul collegamento

$$d_t = \frac{L}{R} \quad (\text{a questo istante è messo l'ultimo bit})$$

1° bit giunge al destinatario in  $t = RTT/2 + 0$

Ultimo bit giunge al destinatario in  $t = RTT/2 + L/R$

L'ACK dell'ultimo bit giunge al mittente in

$$t = RTT/2 + L/R + RTT/2$$

Utilizzo del mittente:

$$\text{Utilizzo} = \frac{L/R}{RTT + L/R} = \frac{\text{Tempo in cui invia}}{\text{tempo totale}}$$

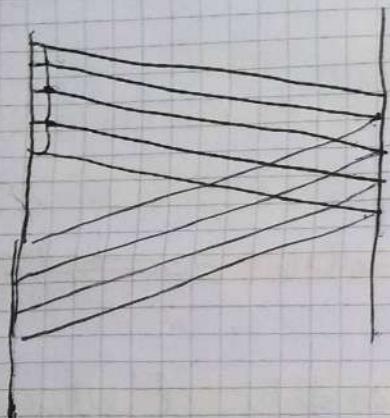
Notiamo che c'è una convenzione fatta che  
nella rete che il mittente è stato già inviato nulla  
tempo Totale.

Allora facciamo inviare al mittente più pacchetti  
insieme belli per farla aspettare l'ACK

Giungiamo a un meccanismo di tipo Pipeline.

Non possiamo utilizzare il metodo di ACK 0 e i come in RDT 3.0 poiché ci sono già pacchetti e di conseguenza già ACK

Sender Receiver Ci sono l'ACK 0, 1 ...



Definiamo la WMA di un certo numero di frame.

Mettendoendo un numero di frame tip 10, ovvero il throughput effettivo (ma oltre il teorico)

### Conseguenze del pipeline

- Intervalli numeri di seq. non finiti di 2
- Lato di invio e ricezione passo dovrà maximizzarsi in un buffer già N in pacchetti
- Gestione errori / pacchetti persi  $\xrightarrow{\text{Go back } N}$   $\xrightarrow{\text{Rigettazione relativa}}$



$\underbrace{\quad\quad\quad\quad\quad}_{N\text{-finestra}}$

N-finestra

- pacchetti con ACK
- // inviati senza ACK
- // disponibili da inviare
- // non disponibili

Visione solo n-th frame.

Anche lato ricevente visto bel numero N  
(bere ragione quanto è grande una finestra)

## Protocollo Go back N

Per ogni pacchetto invio un timer, se non timer esce (non è arrivato o danneggiato) torna indietro  
battendo i pacchetti ricevuti fino anche se corretti  
Go back N = in caso di errore torna indietro di N pacchetti

Quindi se il mittente vede scadrere il timer del pacchetto 50, torna a 50 e ripete 50...51...52...

Il ricevente è a conoscenza di questa politica e invia l'ACK di conferma cumulativo, cioè che è arrivato l'ACK del 90 vuole dire che tutti i pacchetti fino al 90 sono confermati

Autori pag 209.

In modo approssimativo G-BN <sup>risponde a 3 errori</sup> risponde a 3 errori

- Invocazione dell'atto: Controlla finetino e invia molti pacchetti.
- Ricezione ACK  $\rightarrow$  basta un ACK di conferma cumulativo
- timeout  $\rightarrow$  basta un ACK di conferma cumulativo

## Azioni GBN receiver

- Arriva pacchetto con numero reg N  $\rightarrow$  cerca nel buffer se quel pacchetto è già arrivato, se no lo mette nel buffer
- Nel frattempo se riceve ACK per quel pacchetto e per gli ultimi pacchetti corretti  $\rightarrow$  invia ACK per l'ultimo pacchetto corretto

Svantaggio: si eliminano molte volte pacchetti ricevuti correttamente perché tornano indietro di N pacchetti

## Rigettazione selettiva

Per ogni pacchetto che ricevo mando un ACK relativo solo a quel pacchetto.

Il mittente se sta inviando il PRT 100 e il timer del PRT 90 scade, finisce di inviare il 100 e invia (nuovamente) il 90.

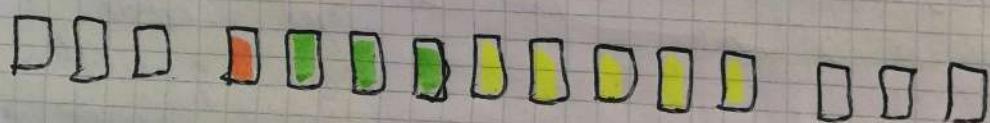
Approssimativamente 6s dopo back N che ritorna brevemente indietro a N anni e dopo tutto.

Qui si cerca di imbarcare le parole e rimuovere solo quella.

Non è un protocollo migliore di 6s back N, la rigettazione selettiva è utile se ci sono diversi pacchi e i vari, se invece su quel che permane si riuscisse a leggere sequenze di bit contigui 6s back N è più utile.



Sequenza vista dal reader



Sequenza vista dal receiver

- atteso
- arrivato, sincronizzato (prima ordine)
- accettabile
- non disponibile

finisce la ricezione e invia ~~getto~~ non comprende  
perché magari il sender invia un pacchetto con  
SWR ma l'ACK è verde.

Ciò accade perché il sender lo invia, invia di nuovo  
il receiver ricalcamente lo invia.

Il receiver regola poi in altri modi che il  
pacchetto è arrivato e scende questa ricezione.

Il receiver ne vede un pacchetto arrivare che  
ovviamente non ha elaborato ma non ha  
l'ACK, né l'errore (ogni m<sup>a</sup> non si riferisce)

## TCP

Si preoccupa di fare la bandwidth, multiplexing  
e demultiplexing.

Fa inizializzazione, ripartizione e terminazione  
della comunicazione.

Fa trasferimento affidabile (con manipolazione e  
ingaggiamenti).

Controlla qualità e congestione, e flusso dati.

NON basta alla sicurezza, il TCP NON  
è sicuro per nulla dall'applicativo

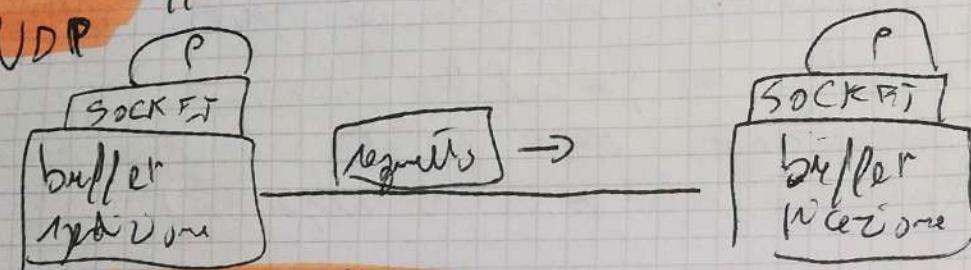
NON c'è garanzia sulla comunicazione

## Caratteristiche

- Orientato allo connessione
- bidirezionale
- E' multiplex - symmetric (entrambe le parti)
- Applicabile
- Usa i messaggi ACK

- Stream - Orientato: il flusso trasmesso di dati è visto come unico, non a blocchi
- Dati NON strutturati
- Gestisce il flusso dati

TCP bufferizza dati prima di inviarli, non come UDP



Segmento:  
Intestazione + dati  
e opzioni

### STRUTTURA

Il pacchetto ha un certo formato 20 byte + opzioni + dati

- 16 bit N. porta TCP
- 16 bit N. porta Dest
- 32 N. seq      • 32 N. ACK
- 6 bit lung. intestazione      6 bit Non usato
- 6 flag      • 16 bit Finestra ricezione
- 16 che ck sum (per correggere errori)
- 16 puntatore a dati seguenti

Dopo questi 20 byte vengono le opzioni e i dati

FLAG:	ACK: indica se l'ACK-field è valido URG: bit dati urgenti PSH: bit di push RST: usato per resettare una connessione SYN: per creare una connessione (gradi 2 pacchetti) FIN: per chiudere una connessione
-------	--

Nel corso di ogni sessione si possono trasferire varie informazioni. Es: lunghezza relativa  
finestra: indica il numero di byte in linea il max  
Dimensione  
intervale di ricezione

Header IP = 20 byte

Header TCP = 20 byte

(min) Dati TCP = 536 byte

TCP richiede che la trascrizione pacetti  
di almeno 576 byte

MSS: dimensione pacchetto

Connessione connessione inviato tramite un solo pacchetto

Problemi se cercano le tre mosse contemporaneamente  
di aprire la connessione

Se per esempio arriva un pacchetto "reciso" e il receiver invia l'ACK e il mittente chiude la connessione.

Fibonacci: se c'è un errore blocca tutto (non è mai buon)

Creatore: 1° pacchetto da H1, il 2° ACK deve essere purissima del 1°, quindi 3° deve essere purissima del 2°.

Three-way Handshake per aprire la connessione  
La creazione che avrà entro pochi secondi o negozi, non fa  
Chiusura della Connessione comunque dovrà  
alla connessione

Più delicate della creazione

Non ci sta un protocollo norma per la chiusura  
della connessione (ACK degli ACK degli ACK ...)

N.B.: Se è inviato il pacchetto con  $\text{Seq} = 42$ ,  $\text{ACK} = 42$   
la risposta sarà  $\text{Seq} = 43$ ,  $\text{ACK} = 43$   
ACK ha il val del nuovo byte stesso  
Pendativo

## Sfriore (e ne cosa sia)

Nello flag FIN, oggetto un ACK se non arriva più lo si sente.

TCP prevede elenco di byte, A chiede verso B e B chiede verso A.  
A volte può capitare che in coda ci siano e l'altra no.

## Telnet

Cos'è un bigotto da client inviato al server e reinviato al client

Usa TCP, ha un rapporto 50:1 in interazione e file: rei

Il telnet produce pacchetti piccoli con spazio di banda

Il receiver può regolare il mittente di non inviare momentaneamente dati per non ne uscire il buffer con la risposta finita di ricezione.

Quando si apre una connessione si può allocare memoria per fare il buffer.

Se questa base (richiesta di Macce memoria all'inizio dell'apertura della connessione) non sono di Macce Datas, rizzibacking per non sprecare pacchetti reali ecc.

Algoritmo di Nagle: fa bufferizzare se le

distanze sono lunghe per non far saltare le reti

oppure riduce l'overhead bufferizzato dati.

N.B.

L'ACK identifica il byte successivo ottenuto, non

RTT basso; tanti RTT → bigotto da come segnale

RTT alto: bufferizzazione

\* Grazie alla finestra di ricezione con celle  
me misura nello stesso abbinato la  
stanzione del buffer del ricevente  
(esempio TCP full-duplex, ci sono due buffer)  
sono fornite nel pacchetto informazioni necessarie  
per far entrare da il mittente intesi il ricevente  
In questo modo si dice che TCP ha un controllo  
del flusso, e della congestione.

### Finestra di ricezione ritratta & protocollo congestione

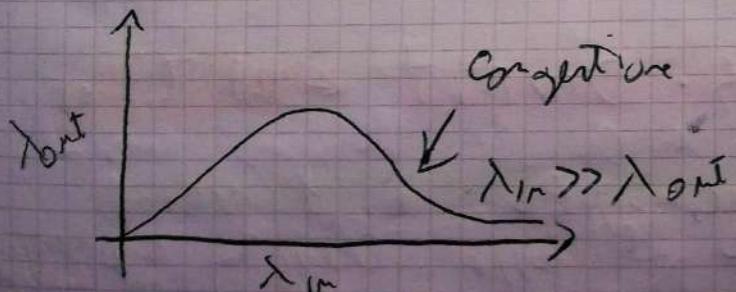
In cosa si risolve variazioni nel buffer (si libera  
ma allo) si avvia il mittente che fa ripetere  
consegnare i tanti pacchetti giusti che restano la  
rete.

Soluzione di Clark: il ricevente non aggiorna  
controllivamente la variabile, ma app che si  
sono liberate già celle.

Con questo si invia sulla rete tantissimi  
piccoli pacchetti

Congestione di rete | La rete ha quindi il traffico offerto  
dalla rete e vicino o meno la capacità  
di rete. Si creano code  $\rightarrow$  ritardi  
buffer giornaliero di gettare le mitate dei pacchetti,  
ma ritardano il ritratta.

Il ritardo causa timeout e negozio molto sojole  
(e cos'è di spie)



I pacchetti in rete  
vengono di molto i  
pacchetti per i  
invati a livello  
applicativo.

La congestione NON è soluzone della banda, ma  
può negare completamente i buffer

L'obiettivo da seguire è raggiungere  
completamente il canale e tenere gli utenti  
periferici i buffer.

La congestione è un problema della rete legato a  
i buffer, che però c'è molto al di fuori.

Lo scatto del timer è fondamentale per abbattere  
la congestione.

Mettere un ~~scatto~~ timer alto diminuirebbe la  
congestione ma creerebbe altri problemi.

Un buon valore per il timer permette di  
migliorare la congestione e la gestione.

TCP fa una stima di Round Trip per  
finire in rete, basandosi sulla stima precedente.

$$\bullet \text{Estimated RTT}_n = (1-\alpha) \cdot \text{Estimated RTT}_{n-1} + \alpha \cdot \text{Single RTT}_n$$

EWMA : Exponential weighted moving average

$$\alpha = 0,125$$

A volte la differenza tra gli stimati e i reali è  
troppo alta

Variabilità di Long RTT

$$\bullet \text{DevRTT}_n = (1-\beta) \cdot \text{DevRTT}_{n-1} + \beta \cdot |\text{Single RTT}_n - \text{Estimated RTT}_n|$$
$$\beta = 0,25$$

$$\bullet \text{BTO} = \text{Estimated RTT} + \zeta \cdot \text{DevRTT} \leftarrow \text{Vel Timer}$$

Di solito il  $\zeta$  valore scelto è 1 secondo.  
Se ci tiene conto, timer raddoppia al doppio. Cresce la esperienza  
del timer iniziale.

Il timer è eseguito

TCP usa un timer per volta, non uno per pacchetto,  
la ricezione di un ACK segnala l'arrivo del timer  
per il segmento successivo (che non è ancora stato  
confermato)

Per bloccare il rallentare la comunicazione si

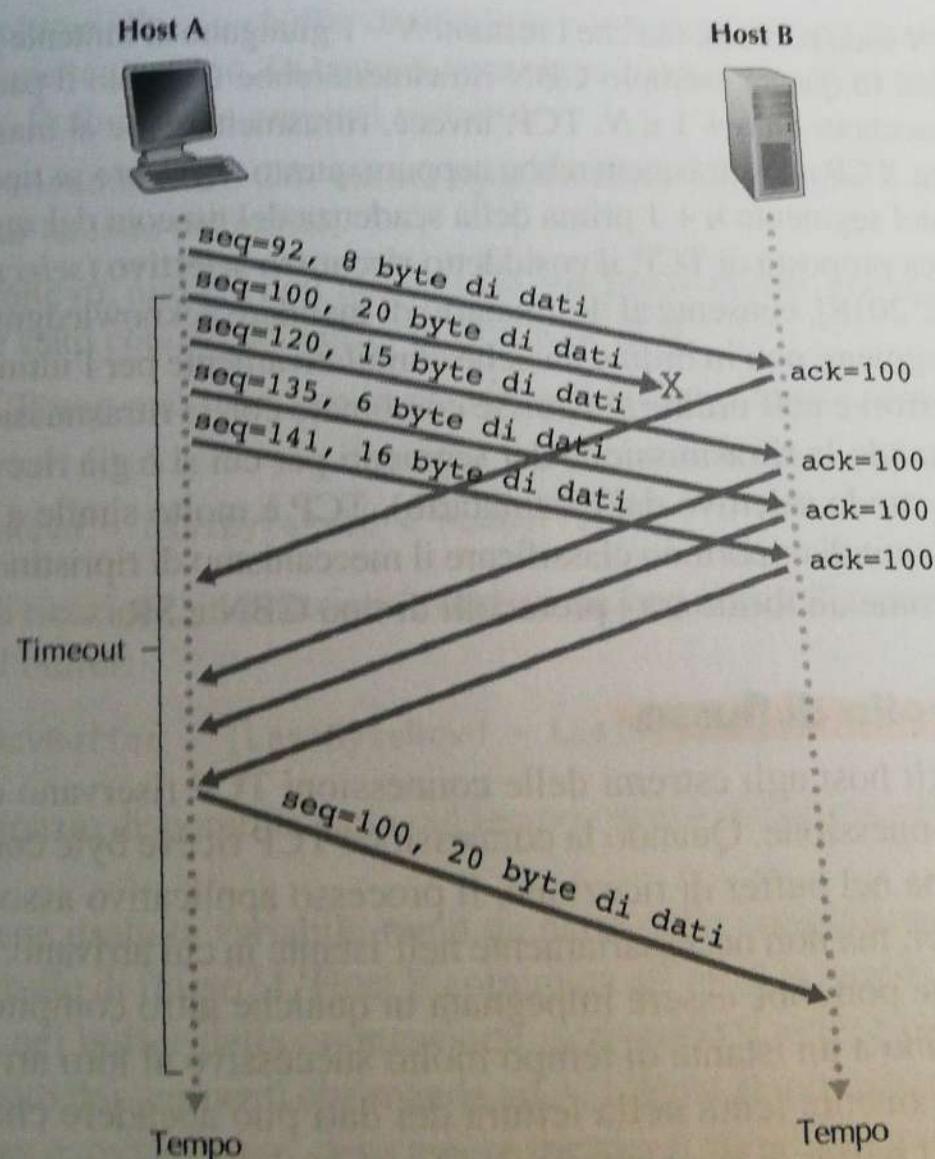
Fast Retransmit (mette una pausa al timer eseguito)

Se arriva un segmento from above mondo un  
ACK duplicato relativo all'ultimo pacchetto  
arrivato in ordine.

Dopo 3 pacchetti fuisciti si rivede il  
10° segmento non confermato. Dicendo RTT.

Se ho  $RTO = 5NS$  e il 30° ~~pacchetto~~ ACK duplicato  
arriva a 3NS, il timer è resettato a  
quest'istante. (è stata forzata un timeout prima)  
di ricevere gli dati deve problemare i pacchetti.

~~Other Condition: A B C D, A arrives so~~



**Figura 3.37** Ritrasmissione veloce: il segmento perduto viene ritrasmesso prima che il suo timer scada.

Nel grafico Nettivo 2

RTO
SAMPLE RTT
E-RTT
dev-RTT

RTO è la media sample RTT (in modo obiettivo)

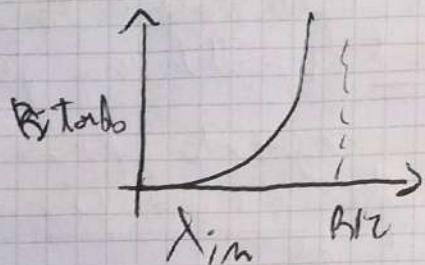
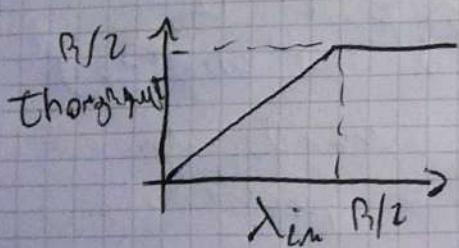
Sample e E-RTT possono differire

dev-RTT di solito ha valori bassi

## Come e così Congestion

1) Due utenti, router di mezzo con buffer illimitato

A e B comunicano, i pacchetti passano per un  
solo router con capacità di mezzo infinito (buffer)



L'host può trarre il massimo  $R/2$   
(in realtà può usare oltre ma il throughput  
può essere oltre)

Poche  $R/2$ ? "Divisione" idealmente la connessione  
in 2; utente è ricevente

Potrebbe sembrare utile avere  $\lambda_{in}$  vicino a  
 $R/2$  visto che throughput supera mai  $R/2$   
presentandosi 2 colamenti da corsa ritardati  
In questo scenario abbiamo funzioni ritardate  
di scambi

2) Due utenti, router di mezzo con buffer limitato  
I pacchetti che arrivano in un buffer sono  
non ricevuti.

$\lambda_{in}$ : tassi d'arrivo verso socket (causando overflow)  
 $\lambda_{in}$ : tassi del livello di trasporto

Caso A) mithente per un utente siamo inviati  
dati solo se esiste da il buffer è libero  
 $\lambda'_{in} = \lambda_{in}$

Ora è il caso ideale

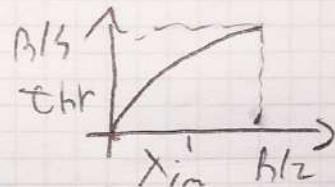
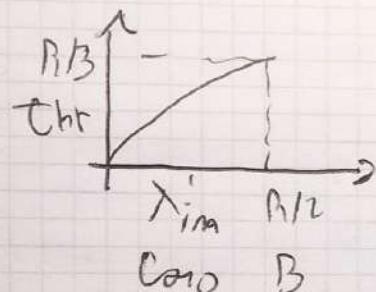
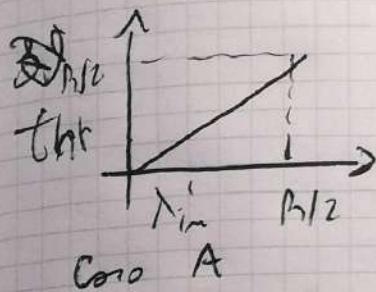
Caso B) Mittente i dati sulle emissioni  
che il pochetto è inviato.

Le emissioni costano un po' di tempo

Caso C) Mittente potrebbe ritrasmettere un pochetto  
anche se non c'è pacchetto da inviare.

Per i lavori si vorrà che i lavori  
ritrasmetta un pochetto non per

Ovvero Caso congetturali: Emissione preventiva  
e non necessaria



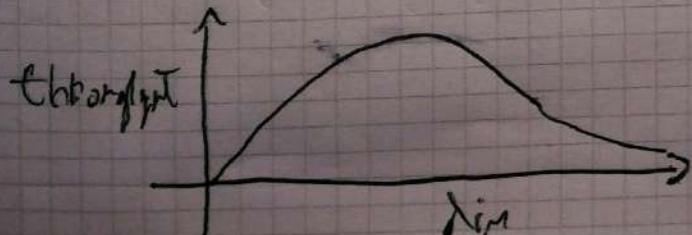
Caso C (più realistico)

3) Più intendi fronte di mezzo con buffer  
~~o mezzo~~ finiti

Arrivare una drastica diminuzione del throughput  
dovuta a una grande quantità di  
lavori inviati, nella rete (nei buffer)

L'eliminazione dei pochetti dal buffer o ~~di dimensioni~~  
dimensioni non necessarie fanno rientrare  
il throughput.

Infatti se un pochetto attraversa 2 router e  
nel terzo è bloccato, i primi 2 hanno  
per loro inutile



## Controllone by congestione

### AIMD

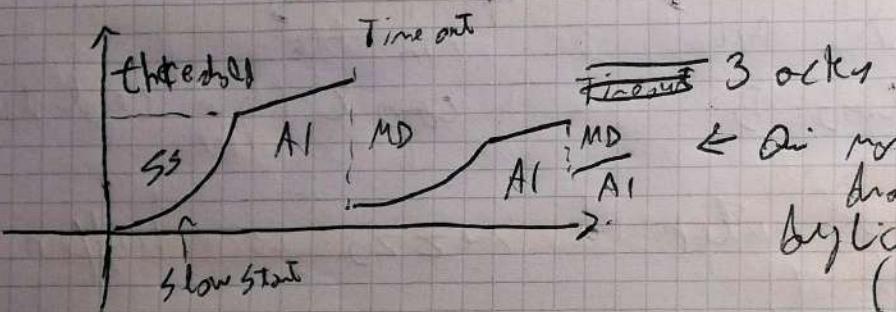
Incremento additivo - decremento moltiplicativo

Meno 8K e va bene, meno 8K ...

Se a un certo punto c'è timeout faccio decremento di moltiplicatore in modo impostando

In realtà moltiplicatore per la fine cerchiamo di andare più velocemente (a velocità quadratiche) per più procedere a velocità lineare.

In caso di timeout (grado) ritorno a velocità quadratiche per poi passare a lineare dopo tre ACK (In realtà la velocità è esponentiale, non quadratiche)



### TCP - Tahoe

- Slow Start
- Congestion avoidance
- Fast Retransmit algorithm

In caso di timeout  
aggiunto da 1 (cwnd)  
 $\text{threshold} = \text{cwnd}/2$

In realtà la cretta non è lineare, è segmentata curva

Tahoe ripete il segnale da 1 (via timeout dei 3 ACK)  
Poco riguarda da 1 in timeout, fa fast recovery (in 3 ACK)

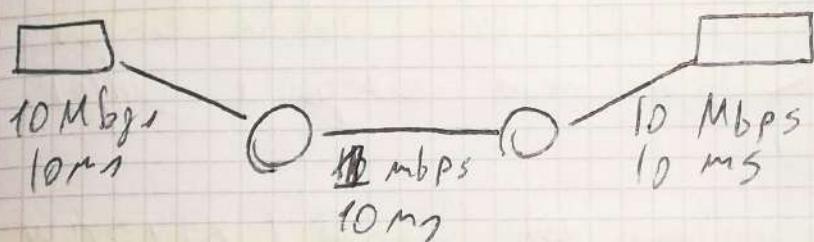
### TCP - Reno

- Slow-start
- Congestion avoidance
- fast retransmit limit
- fast recovery

In caso di timeout. Reporto delle voci già ricevute  
~~(calcolo threshold)~~ Calcolo threshold ( $\text{threshold} = \text{cwnd}/2$ )

$\text{cwnd} = \text{retx}$   
ne 3 ACK

## Controllo flusso e congestione



① Canale senza perdite  
ritardo costante  
Grafici in stile  
Prestazioni circa simili

② Canale con perdite ~~ma non variabile~~

Grafici  
Prestazioni di Retr. leggermente migliore

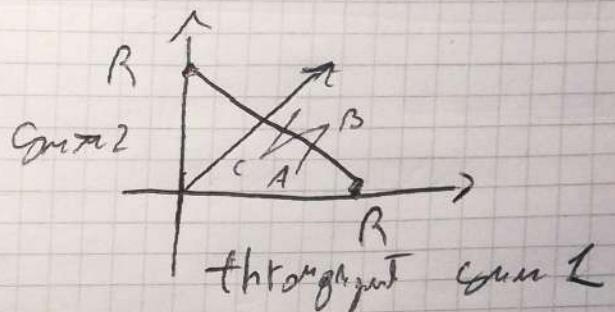
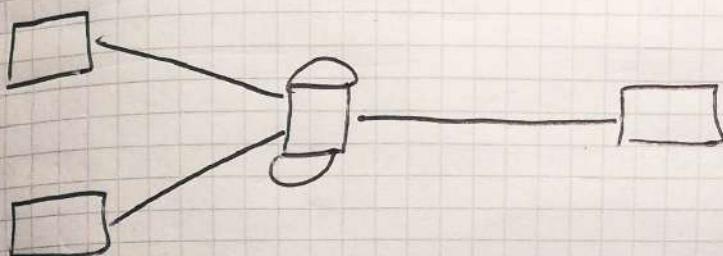
③ Canale con perdite, larghezza di banda variabile e ritardo variabile

Grafici ~~risultazione~~  
Prestazioni di Retr. leggermente migliori

④ 3 flussi concorrenti

Canale con perdite, bw variabile, ritardo variabile  
Retr. grande il raggruppamento

Le connessioni TCP hanno la proprietà Fairness  
ovvero condizione della rete / banda



La rete  $\nearrow$  ha  $x=y$  che è  
il motivo obiettivo

Idealmente vorremmo stare in  $(R/2, R/2)$ , ma è statisticamente impossibile.

Cerchiamo di stare sulla bisettrice

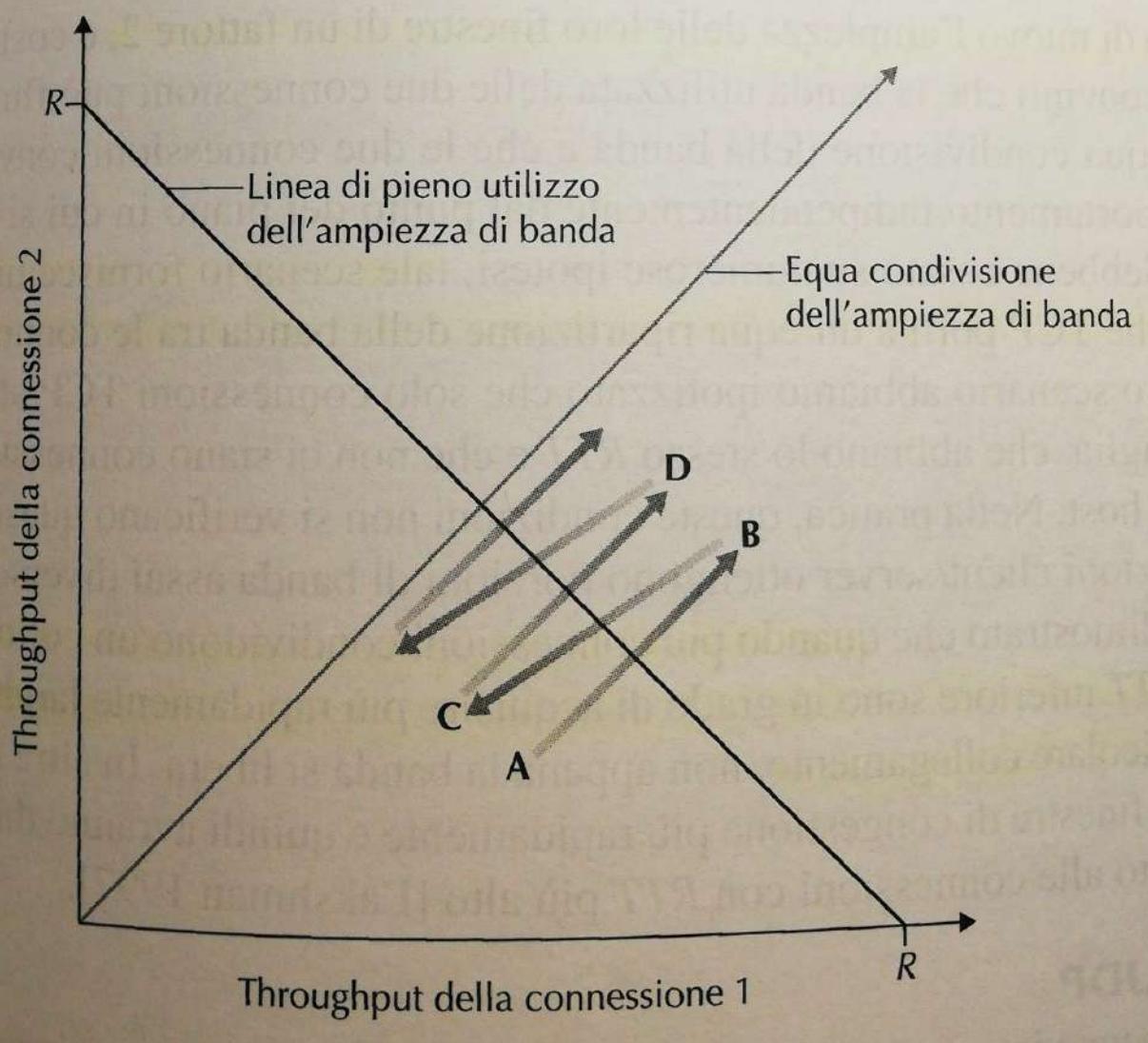


Figura 3.55 Throughput delle connessioni TCP 1 e 2.

C'è un'altra svolta binettice con un momento a zigzag  
scoperto.

Da A invia a B la coda una volta che  
(caso che  $X+Y < BW$  non viene in buffer ed  
entra in time out)

Arriva in B e avverte un timeout ( $X+Y > BW$ )

~~ma~~ da  $X, Y \rightarrow \frac{X}{2}, \frac{Y}{2}$  e arriva in C.

Ricomincia il ciclo e continua regale fino  
alla binettice.

E' difficile prevedere la fairness nel caso ci sono  
due camme ma TCP è "greedy" (Reno) e connessione  
"generosa", Reno si prenderebbe tutto

## Livello di Rete

Ultimo livello nella gerarchia protocollare  
base si parla quindi senza riferire a  
qualsiasi protocollo

Il protocollo è determinato con strategie di  
Circuito Virtuale o Datagramm

Circuit  
Circuit  
Virtual

Ognuno ha i suoi vantaggi e svantaggi

	Datagram	Circuito virtuale
Creazione circuito	No	Si
Info di stato	Indirizzi nei singoli pacchetti	Indirizzo di circuito
Imballo/decantamento	Non info	Ogni circuito va identificato
Effetti di quote	Nessuno	Reset di tutti i circuiti che usano il porto usato
Complessità computazionale	Complessa	Semplice

Abbiamo nel livello di rete diversi protocolli gen:

- **Indirizzamento** che fornisce relazione del percorso
- **Protocollo IP** | la convenzione di indirizzamento  
| definisce formato datagram  
| la convenzione per gestione pacetti
- **Protocollo ICMP** | gestisce errori  
| gestisce regolazioni dei pacetti

## Indirizzo IPv4

L'indirizzo IPv4 è formato da 32 bit.

Gli indirizzi sono suddivisi in classi

classe *	Da	A	1 byte per net, 3 per host
A	1.0.0.0	127.255.255.255	←
B	128.0.0.0	191.255.255.255	← 2,2
C	192.0.0.0	223.255.255.255	← 3,1
D (multicast)	224.0.0.0	239.255.255.255	
E (riservati)	230.0.0.0		

Ci sono degli altri "gadgetti"

0.0.0.0 this host non valido, & volte rappresentato come host  
00000...xxx Indirizzo valido ma già mappato  
nella rete

255.255.255.255 broadcast

127.0.0.0 - 127.255.255.255 loopback

10.0.0.0 - 10.255.255.255 IP privati

169.254.0.0 - 169.254.255.255 auto IP in automatico

Poi ne abbiamo altri privati

172.16.0.0 - 172.31.255

192.168...

La suddivisione in dom è via IP, possono le maschere

N sottorete  
(Nt, sottorete, host)

$\rightarrow \text{a.b.c.d/x} \leq \text{maschera}$  (numero dei  
bit n<sup>o</sup> - sottorete - host  
 $0 \leq x \leq 32$ )  
che definiscono  
l'ind. delle  
sottorete

151.81.232.066 AND  
255.255.255.000 < maschera

Indicare X o la  
maschera è equivalente

la maschera di sottorete serve a distinguere  
l'host (serve per dividere la rete in sottorete, sottorete e host)

### Ampliamento indirizzo IP

Ci sono vari enti che si occupano dell'aspetto  
degli indirizzi

RIR - Continente

Sotto abbiamo i LIR (in Italia CARIN-LIR)  
che gestiscono a livello locale.

### Formato datagramm IPV4 Foto di de

- 0 bit : versione
- 4...7 : IHL (length) interno
- 8...15 : tot offset il tipo
- 16...31 length datagram
- D, flags offset : per frammentazione ; identifica il datagram
- TTL Contatore di salti (per fare round)
- Protocol : n indica il transport layer
- checksum : se c'è uno error si butta
- Source e dest IP
- Dati e opzioni

Matchera: fatto da 111...000 ci ha un  
molo bit. L'indirizzo di portare parte di  
rete, subnet e host.

Ella è applicata su l'AND, i ha il vantaggio di  
non dovrà avere per forza il bloccaggio del byte, si  
regionala bit a bit.

Frammentazione Si fa perché ci il MTU = Maximum transmission unit

A volte è necessario suddividere i pacchetti grandi.  
La frammentazione della sorgente consente il riassemblo alla destinazione. (La frammentazione è evitata se già partito)

Ogni frammento ha lo stesso ID, offset indica appunto da "distanza" dall'inizio del pacchetto.

Vivono i bit-flag DF, MF.

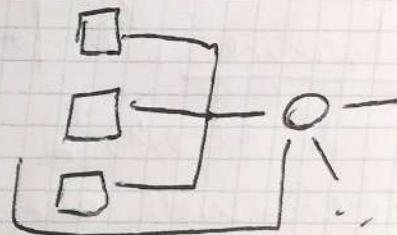
MF indica se è l'ultimo frammento.

DF indica se è il primo frammento \*

La matrice serve per riunire gli sketch di indirizzi

\* Identificare una rete

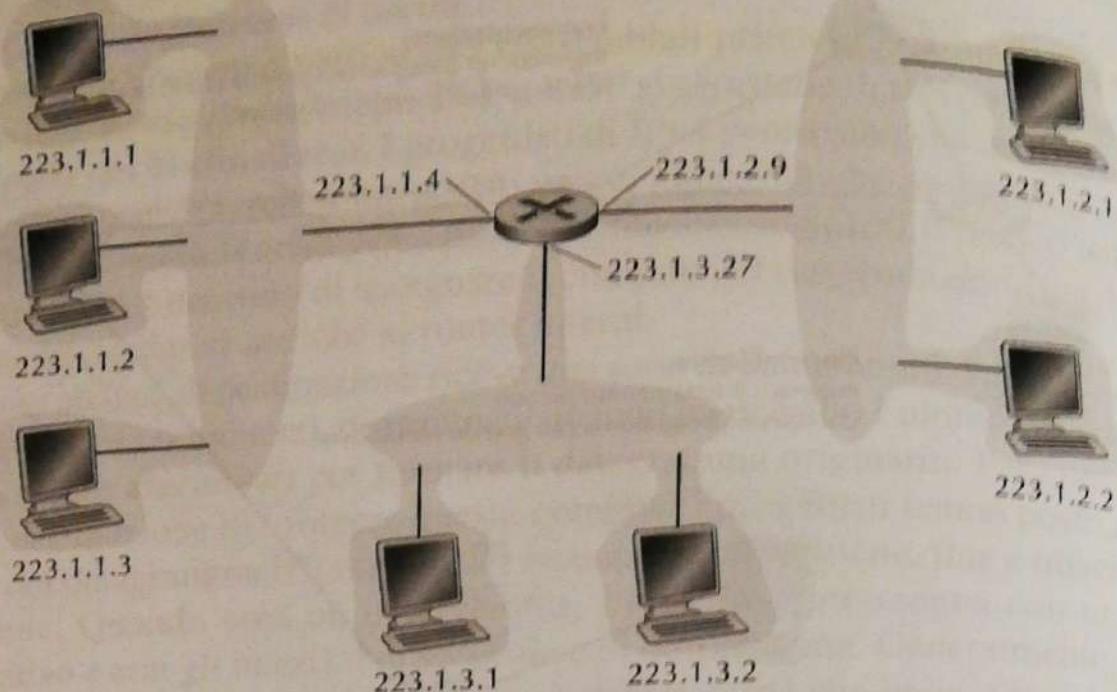
La rete a sinistra ha  
identificazione 223.1.1.0



Il router al centro ha 3 interfacce di rete,  
fa parte quindi di 3 reti.

Figura 4.11

mento con la rete; quando l'implementazione di IP dell'host vuole inviare un datagramma, lo fa su tale collegamento. Il confine tra host e collegamento fisico viene detto interfaccia. Invece, dato che il compito di un router è ricevere datagrammi da un collegamento e inoltrarli su un altro, questo deve necessariamente essere connesso ad almeno due collegamenti. Anche il confine tra un router e i suoi collegamenti è chiamato interfaccia. Il router presenta più interfacce, una su ciascuno dei suoi collegamenti. Dato che host e router sono in grado di inviare e ricevere datagrammi, IP richiede che tutte le interfacce abbiano un proprio indirizzo IP. Pertanto, l'indirizzo IP



**Figura 4.18** Indirizzi delle interfacce e sottoreti.

La Figura 4.18 mostra un router (con tre interfacce) che connette sette host. I tre a sinistra e l'interfaccia del router cui sono connessi hanno un indirizzo IP della forma 233.1.1.xxx: ossia, i 24 bit più a sinistra nell'indirizzo IP sono identici. Le quattro interfacce sono interconnesse da una rete che non contiene router. Se questa rete fosse, per esempio, una LAN Ethernet, le interfacce sarebbero interconnesse da uno switch Ethernet (Capitolo 6) o da un punto di accesso wireless (Capitolo 7). Per adesso rappresentiamo la rete priva di router che connette questi host come una nuvola.

Per IP, questa rete che interconnette tre interfacce di host e l'interfaccia di un router forma una **sottorete** [RFC 950]. Nella letteratura relativa a Internet le sottoreti sono anche chiamate reti IP o semplicemente *reti*. IP assegna a questa sottorete l'indirizzo 223.1.1.0/24, dove la notazione /24, detta anche **maschera di sottorete** (*subnet mask*), indica che i 24 bit più a sinistra dell'indirizzo definiscono l'indirizzo della sottorete. Di conseguenza, la sottorete 223.1.1.0/24 consiste di tre interfacce di host (223.1.1.1, 223.1.1.2, 223.1.1.3) e di un'interfaccia di router (223.1.1.4). Ogni altro host connesso alla sottorete 223.1.1.0/24 deve avere un indirizzo della forma 223.1.1.xxx. La Figura 4.19 riporta gli indirizzi delle tre sottoreti.

dove  $x$  indica il numero di bit nella prima parte dell'indirizzo.

Gli  $x$  bit più a sinistra di un indirizzo della forma  $a.b.c.d/x$  costituiscono la porzione di rete dell'indirizzo IP e sono spesso detti **prefisso** (di rete) dell'indirizzo. A un'organizzazione viene generalmente assegnato un blocco di indirizzi contigui con un prefisso comune (si veda al riguardo il Box 4.1) per tutti gli indirizzi IP dei dispositivi che si trovano al suo interno. Quando tratteremo il protocollo di instradamento BGP (Paragrafo 5.4) vedremo che i router esterni alla rete dell'organizzazione considerano solo gli  $x$  bit del prefisso, cioè, quando un router all'esterno dell'organizzazione inoltra un datagramma avente un indirizzo di destinazione che è interno, dovrà considerare solo i primi  $x$  bit dell'indirizzo. Questo riduce in modo considerevole la dimensione della tabella di inoltro dei router, dato che una sola riga della forma  $a.b.c.d/x$  è sufficiente per far pervenire i pacchetti all'organizzazione.

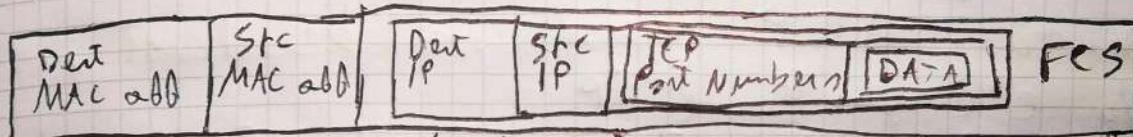
I rimanenti  $32-x$  bit di un indirizzo possono essere usati per distinguere i dispositivi interni dell'organizzazione, che hanno tutti lo stesso prefisso di rete. Saranno quindi i router della rete interna che utilizzeranno i restanti bit dell'indirizzo per indirizzarli al dispositivo destinatario. Tali bit potrebbero presentare un'aggiuntiva struttura di sottorete, come quella trattata precedentemente. Per esempio, supponiamo che

MAC-ADDRESS ogni scheda di rete ne ha una

Per ogni interfaccia di rete attivare un indirizzo IP e un MAC ADDRESS

Protocollo ARP per traduzione IP e MAC ADDRESS

Inviare tutti i pacchetti per creare quelli che viene opportunamente spedito (pacchetto ethernet)



Il MAC-ADDRESS ~~della~~ ~~macchina~~ serve solo per comunicare le schede sulla stessa rete

Il MAC ADDRESS si trova a livello LAN, è dato dal costruttore.

Quando una macchina vuole comunicare con un'altra deve avere in modo per passare da IP a MAC ADDRESS e riceverla

l'IP è legato alla posizione nella LAN, il MAC

<sup>mo.</sup>  
Si usa ARP che fa la "traduzione"  
(Le frane viaggiano sempre il MAC)

ARP significa Address Resolution Protocol

Supponiamo di conoscere l'IP del dest.

Puoi inviare ARP request in broadcast,  
nella info c'è scritto l'IP

E' una richiesta che sta a livello IP

Tutte le machine la analizzano ma solo  
una non la risponde

La reply verrà inviata solo a chi aveva  
fatto la request, essa contiene il MAC-address  
che il destinatario voleva ricevere.

Qui ci sarà un TTL grande

Problemi: pagine prima \*

- ARP <sup>52</sup> non provvede autenticazione
- Una reply può essere inviata senza richiesta
- Anche un host riceve pacchetti ARP aggiornati da altri Cache ARP

Conseguenza: il traffico IP è facilmente sfruttabile

## Mandare /31

Serve per un P2P. In questo caso non ha senso parlare di sottrete e broadcast (è uno dei 2 macchine). Si può mandare /32 ma solo da macchina.

- NB: quei sottoreti non possono avere indirizzo IP in comune.

## ARP ≠ Sicurezza

### RARP reverse ARP

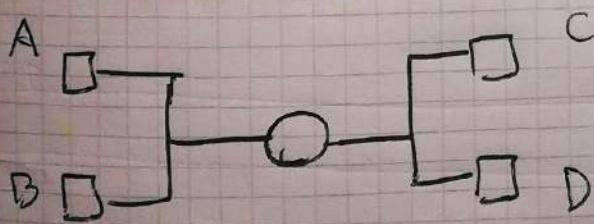
Ha bisogno di un server RARP.  
Sarà la macchina che ha il proprio IP.

L'Host invia un RARP fragment in broadcast, il server risponde con lo stesso IP e consegna il IP di risposta.  
Anche questi protocolli soffrono di vulnerabilità.

### BootP

Serve per fare bootstrapping della rete sulle macchine senza SO. Oggi nel processo di avvio dell'OS il server BootP assegna l'indirizzo IP.  
E' utile quindi per il bootstrapping della rete.

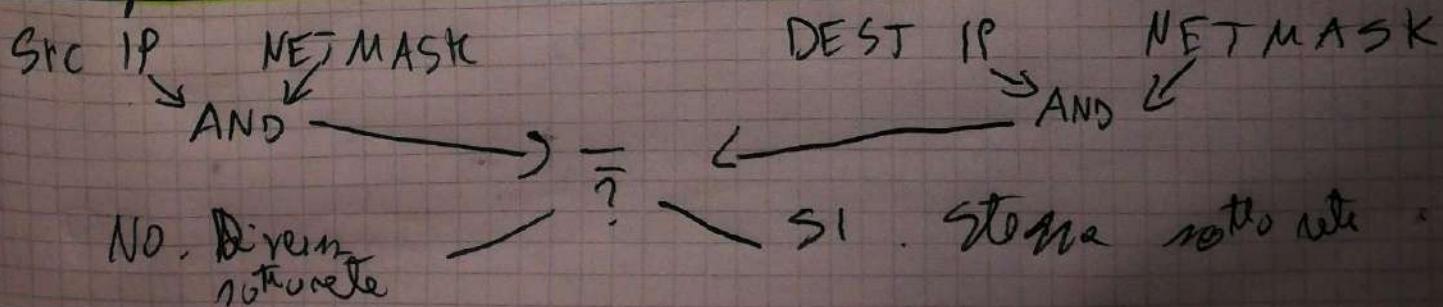
## LAN interconnessione



E se A deve parlare con B si usa ARP

Ma se A deve parlare con C non si può usare semplicemente ARP. Si deve copiare se si deve inviare nella stessa sottrete o "usare"

Come capire se è nella stessa sottrete?



Quando bisogna inviare frame dalla rete viene generato  
fatto ARP ma con il portale, ne bisogna comunicare  
con qualcuno nella stessa rete che ha l'IP ARP e porta

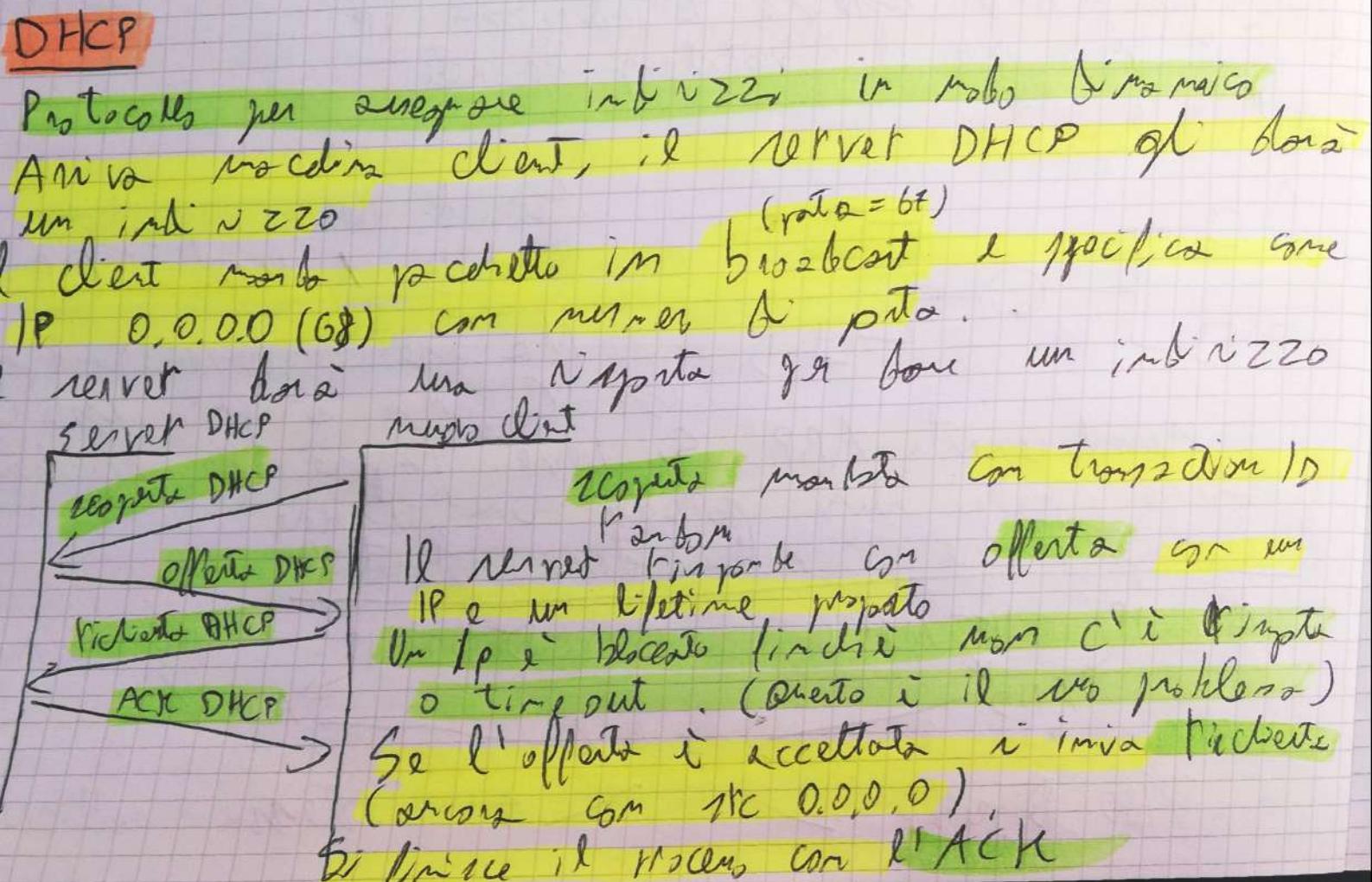
Quando Configura sua máquina na rede sempre queira 3 informações:

- IP
- Macenzie da sua rede
- IP de roteador/gateway

Despunt CW: Rauter bi deserte bi misioita  
Si maz solo per il rauter dopo la  
manutenzione

Tabella D testing Prende in tutti gli host  
E' un database che serve nelle di definizione.

→ Nel lab contiene reti di destinazione con mask,  
IP del router da inviare gli ormai verso la  
dest e una metrica che dice quanto è  
conveniente quel percorso



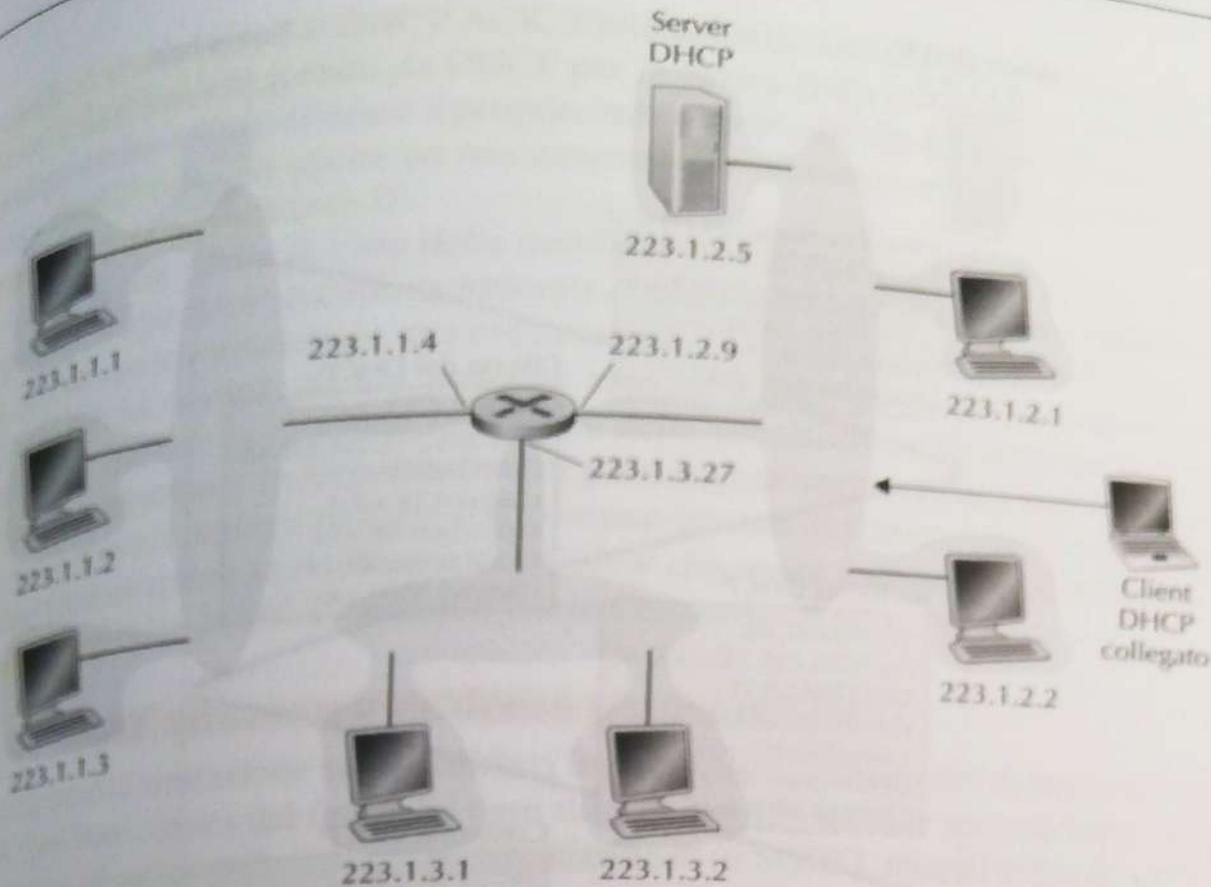


Figura 4.23 Scenario del protocollo DHCP client–server.

- Individuazione del server DHCP.** Il primo compito di un host appena collegato è l'identificazione del server DHCP con il quale interagire. Questa operazione è svolta utilizzando un messaggio **DHCP discover**, che un client invia in un pacchetto UDP attraverso la porta 67. Il pacchetto UDP è encapsulato in un datagramma IP. Ma a chi dovrebbe essere inviato questo datagramma? L'host non conosce ancora l'indirizzo IP della rete alla quale è collegato e ancor meno l'indirizzo di un server DHCP per quella rete. Detto ciò, il client DHCP crea un datagramma IP contenente il suo messaggio DHCP con l'indirizzo IP di destinazione broadcast di 255.255.255.255 e l'indirizzo IP sorgente di 0.0.0.0, cioè “questo host”. Il client DHCP inoltra il datagramma IP al suo livello di collegamento, che invia il frame in broadcast a tutti i nodi collegati alla sottorete. Vedremo i dettagli dell'invio in broadcast a livello di collegamento nel Paragrafo 6.4.
- Offerta del server DHCP.** Un server DHCP, che riceve un messaggio di identificazione, risponde al client con un messaggio **DHCP offer**, che viene inviato in broadcast a tutti i nodi della sottorete, usando di nuovo l'indirizzo IP broadcast 255.255.255.255 (dovreste chiedervi come mai anche la risposta del server deve essere in broadcast). Dato che in una sottorete possono essere presenti diversi server DHCP, il client dovrebbe trovarsi nell'invidiabile posizione di essere in condizione di scegliere tra le diverse “offerte” disponibili. Ciascun messaggio di offerta contiene un insieme di informazioni che definiscono le proprietà di un’intera IP.

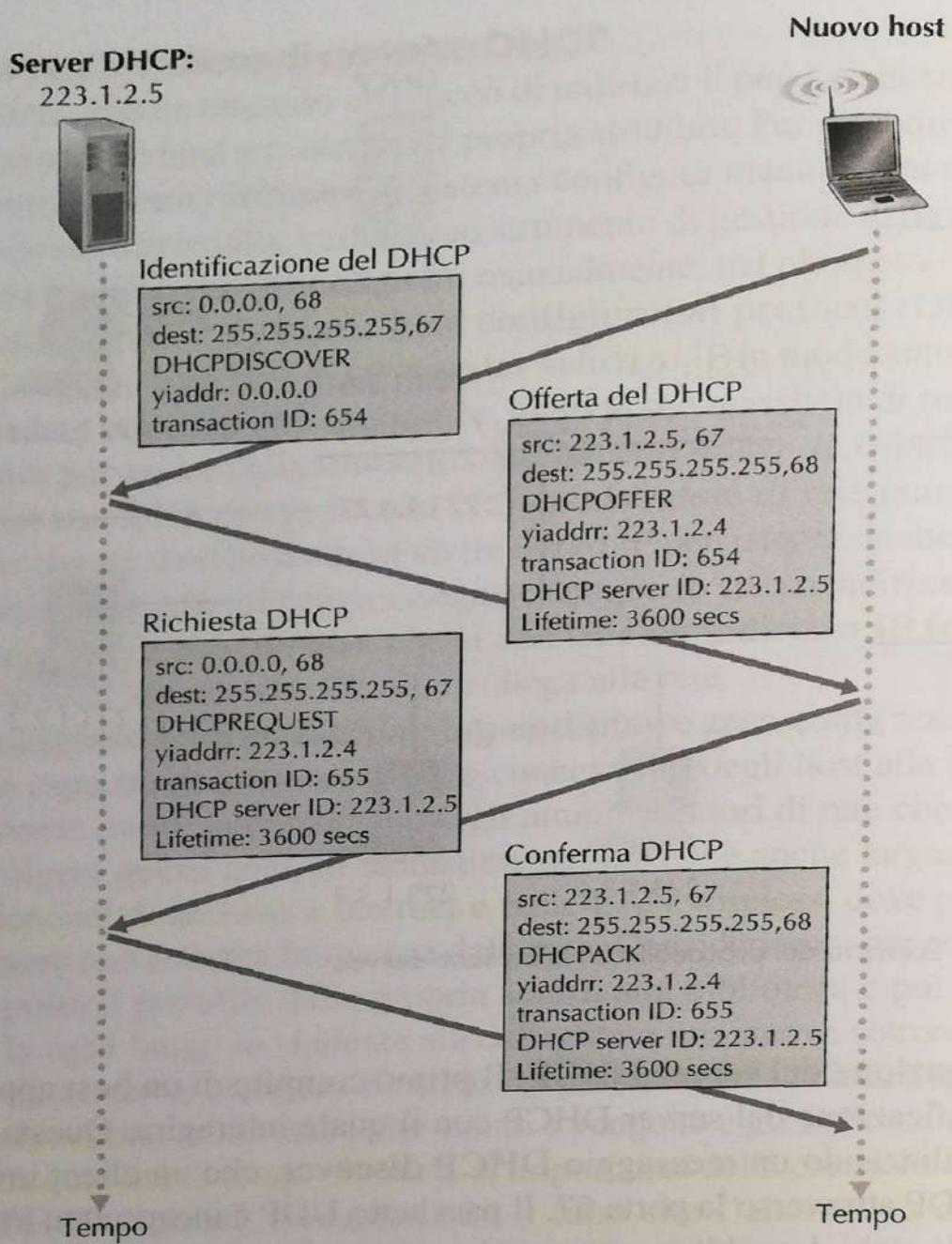


Figura 4.24 Interazione client–server DHCP

fetta server contiene l'ID di transazione del messaggio di identificazione ricevuto, l'indirizzo IP proposto al client, la maschera di sottorete e la durata della concessione (**lease time**) dell'indirizzo IP (il lasso di tempo durante il quale l'indirizzo IP sarà valido). Tale valore è comunemente dell'ordine delle ore o dei giorni [Droms 2002].

- **Richiesta DHCP**. Il client appena collegato sceglierà tra le offerte dei server e risponderà con un messaggio **DHCP request**, che riporta i parametri di configurazione.
- **Conferma DHCP**. Il server risponde al messaggio di richiesta DHCP con un messaggio **DHCP ACK**, che conferma i parametri richiesti.

Quando il client riceve il DHCP ACK, l'interazione è completata e il client può utilizzare l'indirizzo IP fornito da DHCP per la durata della concessione. Dato che un client potrebbe voler utilizzare il proprio indirizzo IP oltre la durata della sua concessione, DHCP fornisce anche un meccanismo che consente ai client di rinnovare la concessione di un indirizzo IP.

Se tra i due IP ci sono due port, entro interni che usano il MAC ADDRESS per evitare conflitti.

Se ci sono 2 diversi DHCP i più gettive negli IP sono di Backup.

C'è anche il necessario per inviare l'IP

## NAT

Fa avere la rete interna a cui assegnare indirizzi locali che con apertura traduzione mappante.

Viene fatto indirizzo valido al gw che fornisce alle macchine interne degli indirizzi locali.

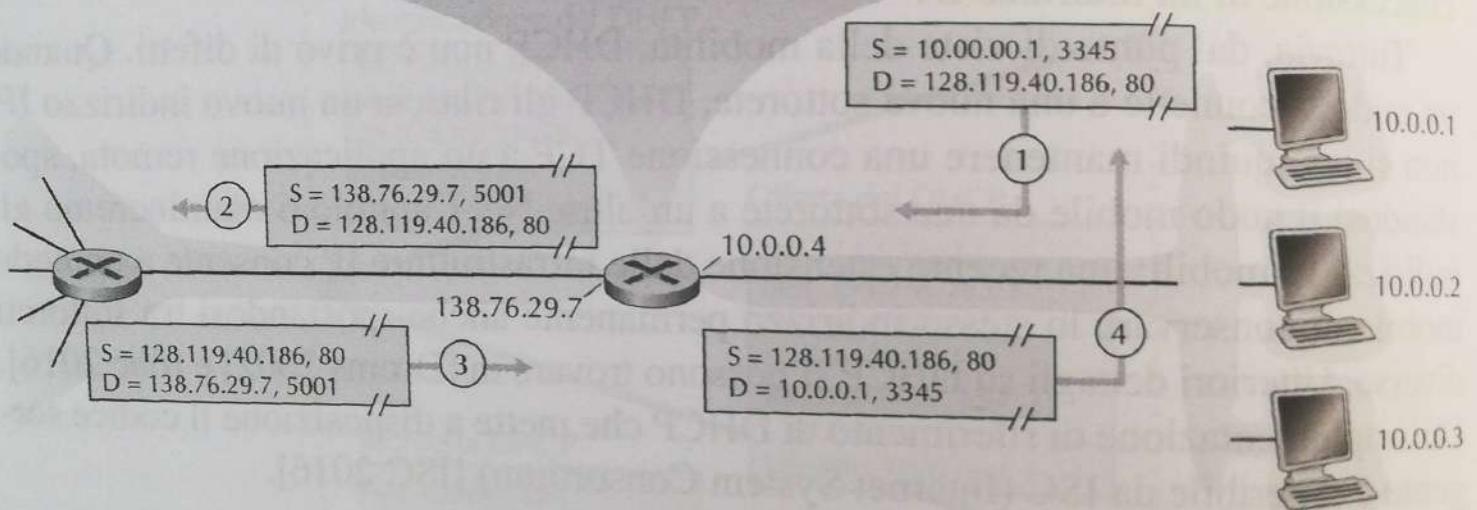
La traduzione avviene attraverso la tabella che trascorre lato WAN e LAN

Il NAT è leggermente nuovo ma che un pacchetto dall'esterno deve passare dal router

La traduzione si fa su un bb nel router che mappano gli WAN e LAN attraverso delle porte.

Appartamento indirizzi in NAT? DHCP

Tabella di traduzione NAT	
Lato WAN	Lato LAN
138.76.29.7, 5001	10.0.0.1, 3345
...	...

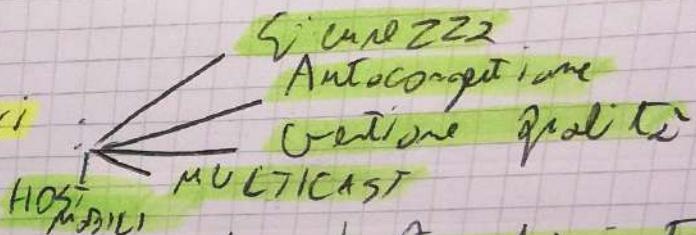


## Limi di IPv4

- Esaurimento indirizzi
- Mobilità routing
- Molti servizi

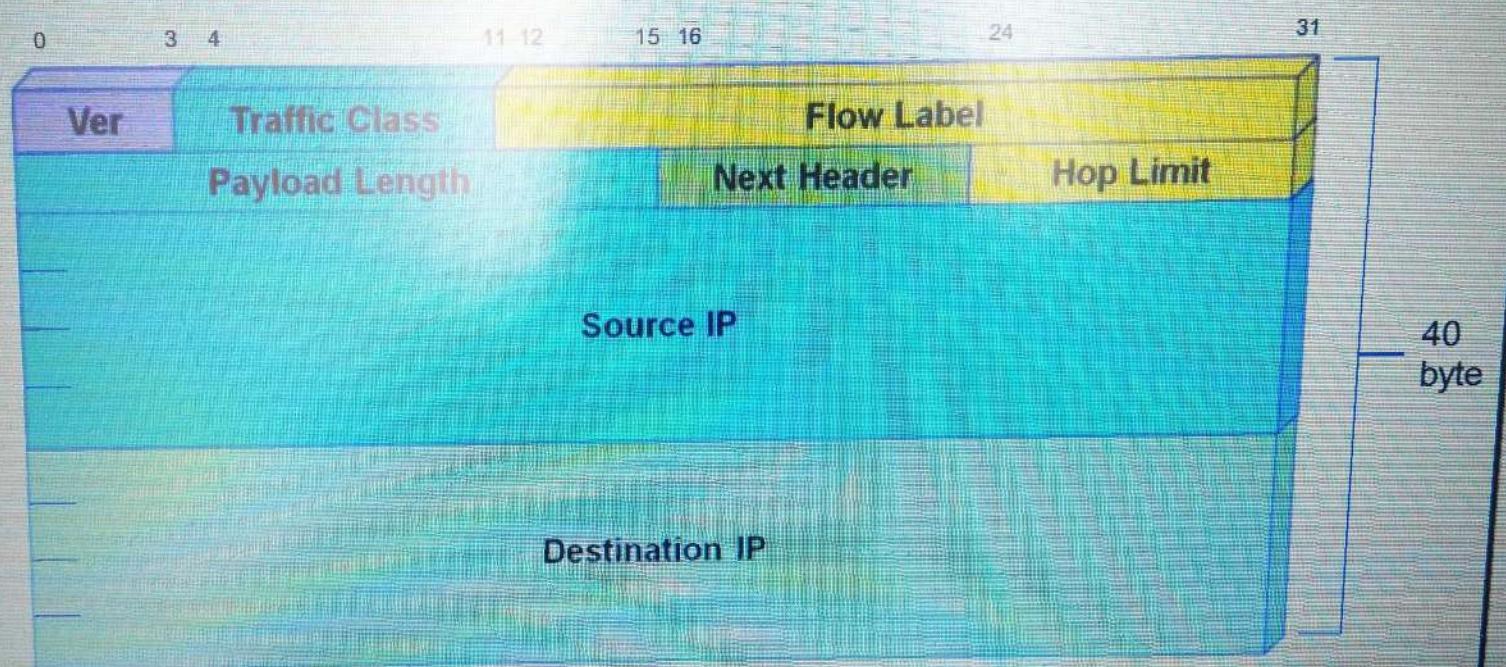
Il routing è stato fatto a caso, ci sono troppe complessità nel routing

IPv6 risolve per nuovi servizi nuovi clienti



Il pacchetto IPv6 pesa 50 byte di interazione  
Possibili indirizzi IPv6 =  $2^{128} = 3 \cdot 10^{38}$  mentre IPv4 =  $5 \cdot 10^9$

L'indirizzo è 16 byte



$$2^{128} = 340282366920938463463374607431768211456 \approx 3 \times 10^{38}$$

$$2^{32} = 4294967296 \approx 4 \times 10^9$$

Sup. terrestre  $\approx 5.1 \times 10^{14} \text{ m}^2 \Rightarrow 6.6 \times 10^{23} \text{ indirizzi IPv6 per m}^2$

$10^{24}$  stelle nell'universo  $\Rightarrow 10^{14}$  indirizzi per stella

~~Il pacchetto c'è~~ Nel pacchetto (lungo fino a 60 byte)

~~nel~~ Abbiamo ver, Traffic Class (conserva il  
~~traffic~~ type of service), Next header per il payload  
di trasporto e le informazioni.

Pay load length : lunghezza dei dati

Arché in IPv6 ci sono indirizzi particolari

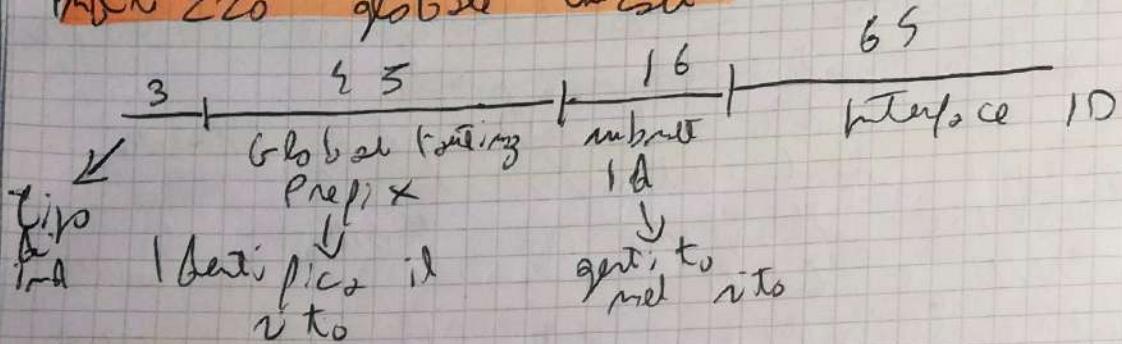
FOTO

Nel indirizzo :: già copie di tutti 0

Anche qui abbiamo la maschera

Si può scrivere un IPv4 in IPv6 :: 151.97.1.1  
:: 1 byte da 16 bit  
:: non specificato

Indirizzo globale uncast



Indirizzo locale /64 carico

FE80 :: 164 Si possono usare solo modelli dello stesso link

Indirizzo di rete

FE80 :: 168

Si possono usare solo tra nodi in uno stesso rete  
Sono deprecati

Assegnamento IPv6 da MAC ADDRESS

□ □ □ □ □ □ MAC ADD

↓

□ □ D Δ Δ C B □  
FF FF

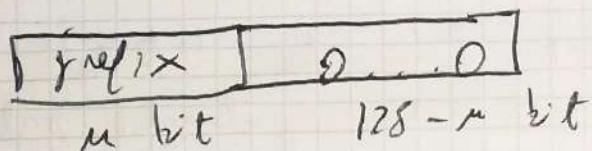
E' creata un IPv6  
valido che non può  
migrare in internet

2° bit res

specificativo 0 → 1

In windows non si usa questa tecnica, l'IPv6  
è generato da un grande numero di bit  
blancos FFFE al centro.

- IPv6 non bin in Per la indirizza sezione
- Unicast (global)  $\leftarrow 2^{xx} : \dots : 3xxx : \dots : FFxx : \dots$
- Multicast (già racchiuso da un gruppo)
- Anycast



Dato un gruppo prende una sola macchina (non importa quale) viene messo di conseguenza il già vicino.

Pure i pacchetti IPv6 sono incapsulati nelle frame Ethernet, come IPv4, cambia solo un campo EtherType

NDP usato da IPv6

Neighbour discovery protocol

Protocollo per scoprire i vicini e gli stranieri

Il protocollo prevede 3 messaggi

- Router solicitation
- / advertisement
- Neighbour solicitation
- / advertisement

Per implementare il protocollo ARP  
Sfrutta Multicast  
• Redirect

Classi di traffico: definiscono le priorità per i dati

Il campo classe di traffico nel pacchetto

definisce in un qualche modo una priorità  
gestisce il flusso dei dati attraverso questa costante

## Frammentazione

E' raggiunta, ma il mittente può frammentare  
tranne un fragment Header.  
Il max MTU è 1280 con 500 segmenti.

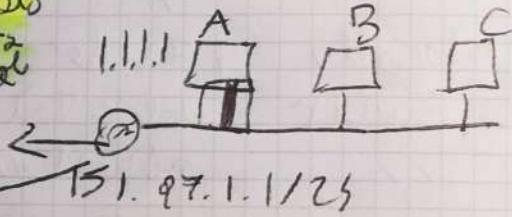
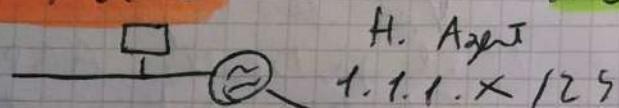
Il next header fa da "tasto" in una Città,  
degli Header optional contenuti nel payload.

Questi Header optional specificano varie cose tra cui  
anche frammentazione.

~~Si~~ Si possono specificare anche Header bivariati  
come authentication Header e per criptare il payload  
(autenticità e confidenzialità)

## Flow Label

Permette di gestire il flusso di dati. (invece di usare  
il porto) Ma questo campo serve indicazione il pacchetto  
in rete il campo è definito  
IP mobile Permette di usare il circuito virtuale



Si crea tunnel IP

Host A che ha connetto al F.A. il suo H.A.  
comunica su quale IP vuole connettersi a  
posta del suo H.A. che sposta lo spazio al F.A.

Nel zw della 151... c'è registrata una /già  
voce che permettono la comunicazione da B e C  
e dal resto del mondo con il tunnel e i  
F.A. e i H.A.

Migrazione IPv4 IPv6

Dual Stack = Macchina con entrambi

trunking: prendere il pacchetto e renderlo  
payable di un altro pacchetto.

In questo modo ~~avrà~~ avranno una sorta di  
compatibilità tra i pacchetti IPv4 e IPv6  
e possono presentare insieme le informazioni: IPv4 incapsula IPv6  
IPv6 incapsula IPv4

**ICMP** = protocollo per messaggi di controllo e livello basso  
(non usato le piste) vi è contro ping  
c'è n° vs che v6

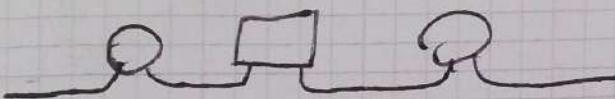
**DHCPv6**: molto simile a DHCPv4  
ma usare 5 messaggi, leggermente differenti

## Firewall

I bre, HW e SW sono completamente differenti

**Firewall SW**: filtro che decide se far passare o meno il pacchetto. Si comporta come un gateway interno. Lavora su basi indirizzi e porta a livello basso

**Firewall HW**



→ Viso che lavora al livello di frammeggio e dentro del LAN e un application gw (nord)

I due router e le LAN private fanno il tutto invisibile dall'esterno.

Il gateway guarda i contenuti dei pacchetti per filtrare. Anche i due router fanno un po' di filtraggio (per le indirizzi e porte) mentre il gw fa il filtraggio sul contenuto.  
Questo HW è uguale

## Algoritmo di routing

Servo a trovare la destinazione

L'algoritmo può essere

• statico

• Dinamico

• basale  
• globale

## • Floating (impostazione)

Il router non sapeva di trovarlo lo manda a tutti quelli collegati. La rete si intella

Bisogna migliorare il floating facendo base a elaborazione completa.

Potremo mettere identificatori nei pacetti e fare attenzione agli Hop limit.

Floating è efficace (recupera le strade uscite) ma non è efficiente.

Il Flooding è ottimo ma va migliorato per non intorpare la rete (abbiamo bloccato)

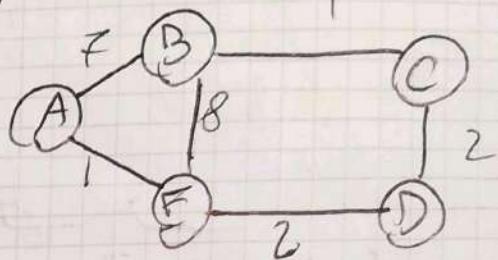
## Distance Vector (Protocollo Autobitato)

Soprattutto Bellman - Ford  
ogni nodo ha una tabella con i costi (miglia)

Colonne: Urte  
Righe: destinazioni

Nelle celle mettiamo i costi (miglia)

	A	B	D	Mia
Dest	1	12	5	
Tabella E	7	8	5	
	6	9	3	
	3	11	2	



Per raggiungere A urto come uscire B o il costo 15 che è il minimo

## Esempio Costruzione Tabella

Miglioramento zero: un zero vuole dire che non ci sono vie né nei vicini né nei lontani contatti immediati mentre i contatti si vicini le multe che i valori vengono aggiornati.

Esempio in stile

Miglioramento: più complicato Esempio in stile

Problema: Se la variazione tra il nuovo e vecchio zero è grande diventerà tutti i valori grandi che i corrispondenti potremo in certi casi riportare la tabella con le celle a 00

Quanto tempo è detto passare per essere

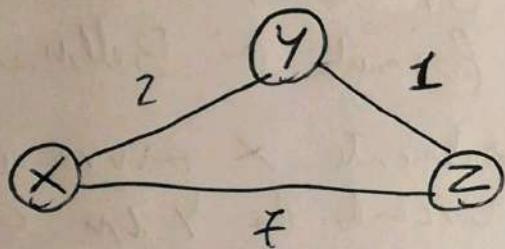


tabella X

X	Y	Z
Y	$\infty$	$\infty$
Z	$\infty$	$\infty$

Controllare la diagonale  
è  
basta

Esercizio DV vsc te

tabella  
A

Det

A	B	C
B	...	...
C	...	...

X	Y	Z
Y	2	$\infty$
Z	$\infty$	F

tabella Y

Y	X	Z
X	$\infty$	$\infty$
Z	$\infty$	$\infty$

tabella Z ...

Z	X	Y
X	F	$\infty$
Y	$\infty$	1

Controllare DV tabella a X: basta creare un vettore  
contenente Y, Z (le righe) e associare il valore  
minimo della rispettiva riga.

X	
Y	2
Z	F

Y	
X	2
Z	1

Z	
X	F
Y	1

Il fonte X invia il suo DV ai suoi vicini (Y e Z)  
così possono sfruttare le informazioni per migliorare  
la tabella.

Il router X prende come input i DV di Y e Z e fa i calcoli sfruttando la formula di BellmanFord

X	Y	Z
Y	$\infty$	
Z	$\infty$	7

Y	
X	2
Z	1

Z	
X	7
Y	1

Allora X arriva in Z uscendo da Y in 2  
Però raggiunge Y in 2 e Y raggiunge Z in 1

$2+1 < \infty$ ?  $\rightarrow$  TRUE: il valore 2+1 è minore del punto di  $\infty$  nella best Z uscita Y

X	Y	Z
Y	2	$\infty$
Z	3	7

Allora X arriva in Y uscendo da Z in  $\infty$  più Z raggiunge Y in 1 e X raggiunge Z in 7.

$7+1 < \infty$ ?  $\rightarrow$  TRUE  $7+1$  è minore del punto di  $\infty$  in best Y uscita Z.

X	Y	Z
Y	2	8
Z	3	7

Gli stessi ragionamenti sono fatti per le tabella Y che ha i DV di X e Z e per le tabella Z che ha i DV di X e Y.

N.B. Se aggiornato una riga c'è un nuovo minimo il DV va aggiornato

X	
Y	2
Z	3

Pointer Y

Y	X	Z
X	2	$\infty$
Z	$\infty$	1

X
Y
Z
3

Z
X
Y
1

Y va in X visto Z in  $\infty$  quindi Y va in Z in  
 1 è Z va in X in 7  
 $7+1 < \infty$ ? TRUE ma non basta perché  $\infty$

Y	X	Z
X	2	8
Z	$\infty$	1

Y va in Z visto X in  $\infty$  visto  
 Y va in X in Z e X va in Z  
 in 3

$3+2 < \infty$ ? TRUE 5 è scritto al posto  
 di  $\infty$

Y	X	Z
X	2	8
Z	5	1

Dorsale: Il DV di Y dopo questi  
 calcoli rimane il medesimo,  
 deve comunque inviare se si sono  
 due vicini?

NO, implica  
 che non ci sono  
 miglioramenti;

Y
X
Z

Tortet =

$z$	$x$	$y$
$x$	$7$	$\infty$
$y$	$\infty$	$1$

$x$	
$y$	$2$
$z$	$3$

$y$	
$x$	$2$
$z$	$1$

$z$  va in  $x$  mentre  $y$  in  $\infty$  però  $z$  va in  $y$   
in 1 e  $y$  va in  $x$  in 2

$2+1 < \infty$ ? TRUE:  $2+1$  è meno del punto  $\infty$

$z$	$x$	$y$
$x$	$7$	$3$
$y$	$\infty$	$1$

$z$  va in  $y$  mentre  $x$  in  $\infty$  però  
va in  $x$  in 7 e  $x$  va in  $y$  in 2

$7+2 < \infty$ ? TRUE  $7+2$  è meno

del punto  $\infty$

$z$	$x$	$y$
$x$	$7$	$3$
$y$	$9$	$1$

DV di  $z$   $\rightarrow$

$z$
$x$
$y$

Questo DV va

da inviato ai tortet  $x$  e  $y$



Risultato X (2<sup>a</sup> volta, è stato inviato un DV da Z)

X	Y	Z
Y	2	8
Z	3	7

X	Y
X	2
Z	1

Z	
X	3
Y	1

X arriva in Y con M<sub>010</sub> e in 8, arriva in Z in 7 e Z va in Y in 1

T+1 < 8? NO nessuna sostituzione

X arriva in Z con M<sub>010</sub> e Y in 3 già va in Y in 2 e Y va in Z in 1

Z+1 < 3? NO nessuna sostituzione.

Il risultato X è inviato (il DV non è controllato)

Risultato Y (2<sup>a</sup> volta, è stato inviato un DV da Z)

Y	X	Z
X	2	8
Z	5	1

X
Y
Z
3

Z
X
3
Y
1

Y va in X con M<sub>010</sub> e Z in 8

Y va in Z in 1, Z va in X in 3

1+3 < 8? TRUE: 3+1 è minore di 8

Y	X	Z
X	2	5
Z	5	1

Y va in Z con X in 5

Y va in X in 2 e X va in Z in 3

3+2 < 5? NO nessuna sostituzione. Il DV rimane ed neaderemo. L'algoritmo è finito

Configurazioni finali

x	y	z
y	2	8
z	3	7

y	x	z
x	2	5
z	5	1

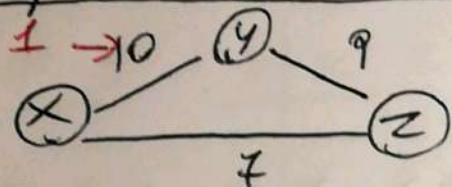
z	x	y
x	7	3
y	9	1

x		
y	2	
z	3	

y		
x	2	
z	1	

z		
x	3	
y	1	

Moltiplicazione di un zero



X	Y	Z
Y	10	16
Z	19	7

Y	X	Z
X	10	16
Z	17	9

Z	X	Y
X	7	19
Y	17	9

Il DVK deve dare zero 10 e 1

X e Y si rinviano subito a sinistra, tornano bene gli aggiornamenti nelle due tabella e iniziamo il loro DV ai vicini per continuare l'algoritmo.

X	Y	Z
Y	1	16
Z	19	7

Y	X	Z
X	1	16
Z	17	9

X	
Y	1
Z	7

Y	
X	1
Z	9

Z va in X uscendo da Y in 19 già Z va in Y in 9 e Y va in X in 1

$q+1 < 19$ ? TRUE viene scritto 10 al posto di 19

Z	X	Y
X	7	10
Y	17	9

Z va in Y uscendo da X in 17 già Z va in X in 7 e X va in Y in 1

$7+1 < 17$ ? TRUE: viene scritto 8 al posto di 17

Z	X	Y
X	7	10
Y	8	9

Z	
X	7
Y	8

Questo DV viene inviato ai vicini

<del>X</del>	Y	Z
Y	1	16
Z	19	7

	Y
X	1
Z	9

	Z
X	7
Z	8

$X \vee_2$  in Y  $\wedge_2$   $Z$  in 16  $\Rightarrow X \vee_2$  in Z in  
 F e Z  $\vee_2$  in Y in 8

$7+8 < 16$ ? TRUE

<del>X</del>	Y	Z
Y	1	15
Z	19	7

$X \vee_2$  in Z  $\wedge_2$   $Y$  in 19  $\Rightarrow$   
 $Y \vee_2$  in Y in 1 e  $Y \vee_2$  in Z  
 in 8.

$1+8 < 19$ ? TRUE

<del>X</del>	Y	Z
Y	1	15
Z	10	7

DV invertito

Y	X	Z
X	1	16
Z	17	9

	X
Y	1
Z	7

	Z
X	7
Y	8

$Y \vee_2$  in X  $\wedge_2$   $Z$  in 16  $\Rightarrow$   $Y \vee_2$  in Z in 9

e  $Z \vee_2$  in X in 7: nessuna variazione

$Y \vee_2$  in Z  $\wedge_2$   $X$  in 17  $\Rightarrow$   $Y \vee_2$  in X in 1  
 e  $X \vee_2$  in Z in 7;

$7+1 < 17$ ? TRUE

<del>X</del>	X	Z
X	1	16
Z	8	9

1) DV invertita

1) 10 in X  
 & non 9

	Y
X	1
Z	8

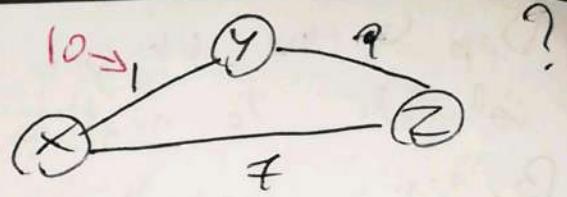
Viene inviato a  
 X e Z che riferiscono  
 i colori gessi non  
 ci salutano in questo  
 caso soluzioni

## Regruppamento di numeri

X	Y	Z
Y	10	16
Z	9	F

Y	X	Z
X	10	16
Z	8	9

Z	X	Y
X	F	10
Y	8	9



X e Y fanno aggiornamento del confronto e inviano il loro DV a Z.

Z	X	Y
X	F	10
Y	8	9

X	
Y	10
Z	7

Y	
X	10
Z	8

Azzingiamo a  $(10-1)$  a tutti i valori in cui partecipa la truzza.

Z	X	Y
X	F	9
Y	11	9

X	Y	Z
Y	10	16
Z	9	F

X	
Y	10
Z	7

Z	X	Y
X	10	16
Y	8	9

Y	
X	10
Z	8

Z	X	Y
X	7	10
Y	8	9

Z	
X	7
Y	8

X	Y	Z
Y	10	15
Z	18	F

Qui si nota che le formule

Y	X	Z
X	10	16
Z	17	9

Z	X	Y
X	7	19
Y	17	9

→ Si può vedere come un avvaggiato a tutte le voci che in cui partecipa l'una è la formula di Bellman Ford

Dopo di che vengono riuniti i DV e le loro  
Salgono la riga di classe

Bisogna fare in modo che venga indicato quale  
casi non sono rientri nelle tabelle

	X	Y	Z
Y	10	15	
Z	18	F	

	X
Y	10
Z	7

	X	Z
Y	10	16
Z	17	9

	X	Y
Z	7	19
X	7	19

	Z
X	X
Y	Y

	X	Y	Z
Y	10	16	
Z	19	7	

	Y	X	Z
X	10	16	
Z	17	9	

	Z	X	Y
X	7	19	
Y	17	9	

Il 15 e il  
18 erano già  
gli obiettivi  
(ma non Bellofford)

Facendo i  
calcoli per  
queste due tabelle

non ci sono  
contraddizioni

S	X	X
X	0	X
X	X	5

X	X	5
X	X	X
X	X	X

LINK State Routing

LINK

Questo protocollo è usato per trasmettere anche dati relativi al link.

Dopo che il controllo del link è stato inviato in uscita, l'informazione viene mandata a tutti i nodi transitanti il pacchetto. Dopo averlo ricevuto tutti i nodi conoscono il grafico.

A questo punto si può applicare Dijkstra (in tutti i nodi). Esempio su slide.

Nel LSRR non trasmette informazioni relative a come i propri vicini nulla sono ritenute le informazioni dei link.

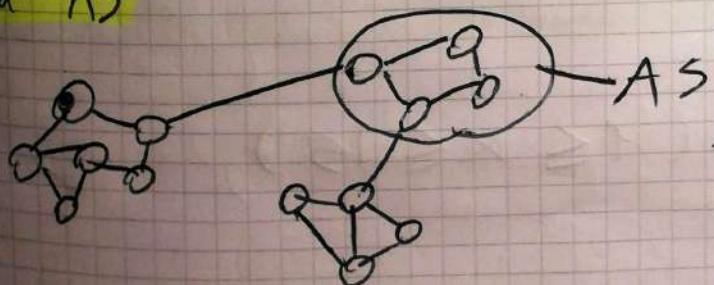
L'algoritmo oscilla poiché potrebbe tenere o rilasciare un link e lasciare uno vuoto.

## Confronto DV e LSR

DV	LSR
messaggi	Globale
messaggi 2	messaggi
vicini	broadcast
o sulle	info sui link
reti.	vicini
messaggi per variation	Messaggio traffico di messaggi

l'intero è organizzato in zone autonome (AS)

im unz AS C bsp. viele Maschine  
kontinuierl gr. mehr / alto o Tz AS o ~~die~~ fertig  
all' AS



I zw che colleziona  
sono sette libri

## RIPASSO DIJKSTRA

Dijkstra(s)

Initialize-single-source(s)

$Q := \text{Build-heap}(V, d)$

While  $Q \neq \emptyset$  do

$v := \text{Extract-min}(Q)$

SCAN(v)

Initialize-single-source(s)

for  $v \in V$

$d[v] = +\infty$

$\text{Pred}[v] = \text{NIL}$

$d[s] = 0$

SCAN(v)

for  $u \in \text{Adj}[v]$

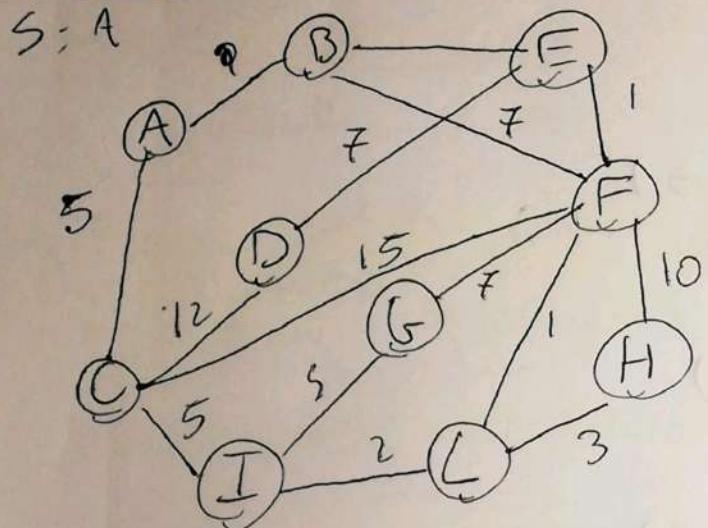
Relax(v, u)

Relax(v, u)

if  $d[u] > d[v] + w(v, u)$

$Q.\text{decreaseKey}(d[u], d[v] + w(u, v))$

$\text{Pred}[u] = v$



Si crea vettore dei Prod lungo  
 $|V|$  e vettore di n. d.  
 Prod

X	X	X	X	X	X	X	X	X	X	X
A	B	C	D	E	F	G	H	I	J	L

0	0	0	0	0	0	0	0	0	0	0
A	B	C	D	E	F	G	H	I	J	L

La inizializza mettere a tutto 0 infinito tranne  
 A = 0 e Prod tutto a NIL X

Si organizza una coda di priorità dei nodi che  
 si gestisce con Key = d

Nel ciclo relazionario (extraiamo) il minimo e  
 appliciamo in tutti i nodi adjacenti la formula

Extraiamo A (non c'è già nella coda)

$$\begin{array}{l} (B) \quad (A) \quad (W(A,B)) \\ Q > 0 + 0 \end{array} \quad \begin{array}{l} \text{TRUE : } A[B] \leftarrow 0 + a \\ Pnq[B] \leftarrow A \end{array}$$

Facciamo così anche per C

FALSE : nulla

e poi inizia la prima iterazione col nuovo min

Ottieni un nuovo grafo e  $S = A$

Prod

X	A	A	E	F	B	I	L	C	F
A	B	C	D	E	F	G	H	I	L

d

0	5	9	20	13	12	18	16	15	13
A	B	C	D	E	F	G	H	I	L

G: ACIG

D: ABFED

B:	AB
C:	AC
F:	ABF
L:	ABFL
E:	ABFE
I:	ACT
H:	ABFLH

## RIP (Routing Information Protocol)

Bottino in DN e una come metrica HOP, i salti.

Max HOP = 15 sono limiti massimi del grafo

sotto all'AS

Oltre tale valore il router è considerato un magnete.

Le routing table sono aggiornate inviando i vettori

Dopo 30 secondi

Se un pacchetto per un router non è stato aggiornato

entro 180 s la distanza è più a

Dopo altri 120 s il router è eliminato dalla RT

dal altro router

Request: inviare info nei reti vicine

Response: inviare info di routing

## Informazioni nello RT

- indirizzo destinazione
- distanza della destinazione in HOP
- Next hop (router successivo o via quale packet)
- timeout
- garbage-collection timer

## Datagrammi RTPV1

Hanno un lungo variabile fino a 512 byte (max 25 reti di destinazione)

## Nel pacchetto:

- Command: 1 = richiesta, 2 = aggiornamento
- Version: versione protocollo
- Address family id: 2 per id IP
- IP address

metric: hop count ( $1 \leq x \leq 15$ )

## RIP v2

- broadcastato con mostre
- autenticazione messaggi
- specifica next hop
- next horizon, poison reverse

## Diagramma RIP v2

Si aggiungono i campi Autentication, Authentication type, Route tag, subnet mask e next hop

RIPng protocollo aggiunto a IPv6 del livello utente, manca di campi autenticazione

OSPF Open Shortest Path First usa Link State Routing

- Usato per sostituire RIP
- è basato sul LSR
- è buono per reti grandi
- autenticazione in messaggi
- permette di usare altre metriche

OSPF si compone di 3 parti

- Hello: scoperta e verifica vicini
- Exchange: sincronizzazione iniziale DB
- Flooding: aggiornamento DB

Il giro nel ~~link~~ è dato dal tempo di attesa ritardo tra due nodi

Ogni 30 minuti i forni le operazioni per aggiornare il DB, il flooding è bloccato tranne 10%

**BGP**: protocollo di interconnessione tra AS che non tiene conto di metriche ma ha "politiche"

C'è tra i router N border: router dell'AS al confine

Hanno BGP ← Router speaker: interni all'"esterno" (sono connessi quale N border)

**BGP neighbors**: coppia di speakers che

condividono info di interconnamento inter-AS

Se due AS comunicano lo fanno tramite il BGP  
che evita e risolve problemi di compatibilità,  
le due AS potrebbero avere protocolli interni  
diversi

## DLL

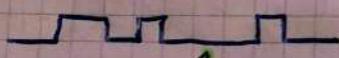
Si occupa di fornire al livello di Rete un servizio  
di trasmissione di frame a bit

Compiti principali:

- raggruppa bit per formare pacchetti (framing)
- gestisce l'accesso al mezzo fisico
- fornisce un recepto affidabile
- gestisce gli errori avvistati ed corrende
- regola il flusso di dati traffic e limita

Nel caso lo connette o c'è o non c'è.

Abbiamo 2 stati:



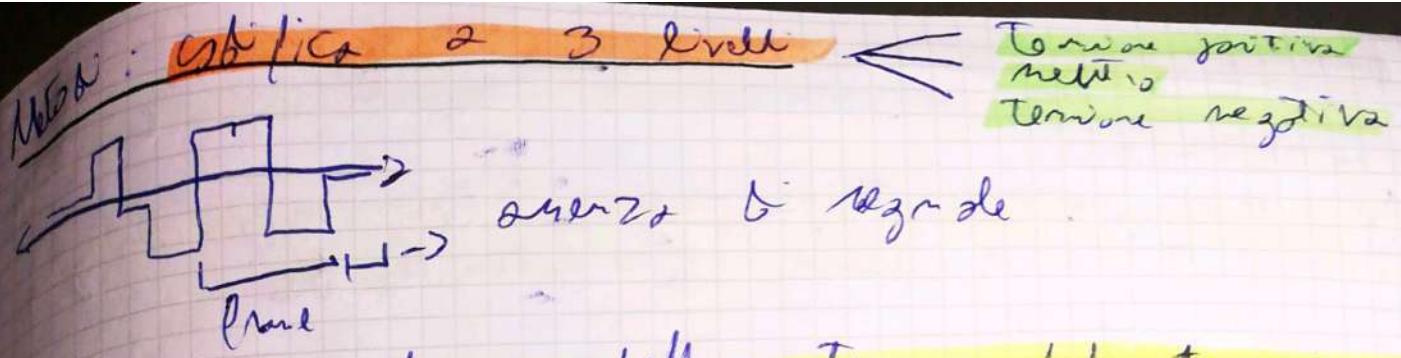
Se c'è: bit 1

Se non c'è: bit 0

Stati 0 o assenza di comunicazione?

"Soluzione": creare un intervallo in cui si è obbligati a trasmettere sempre (intervallo minimo)

Nel intervallo minimo: ho il problema legato a "quando inizia" e "quando finisce"



Si cerca un terzo filo potenza del terzo filo  
potere fare un intera a 5 livelli e quindi  
un quinto della potenza del mezzo

Codifica 5B5B

5 bit del DLL corrispondono a 5 bit nel filo

Nella tavola ci sono anche le reg. per definire  
l'inizio e lo fine di frame

Caso: per 5 bit ne ho 5 = 20% di synco  
Norma per codifica tutti 0 nel filo per l'antiguità

Codifica 8B10B

8 bit DLL corrispondono con 10 fili  
Abbiamo numero di bit non pari poiché ovviamente  
dai bit a 0 e metà a 1 la funzione meglio

il sistema  
Sai che è una codifica elicoidale  
dato che il numero di 1 è il numero di 0  
vengono tenuti uguali  
lo ng di 8 bit è diviso in

5 bit → 6 fili  
3 bit → 6 fili

Il sistema cerca di comporre il ~~numero~~ il numero  
di fili overo lunghe (N<sub>filo</sub>) ~~le filo~~ il numero  
di bit a 1 uguali al numero di bit a 0  
Questi numeri compongono synco

Quindi conviene usare reg di 5 bit in 6 fili overo  
2 opzioni e solo quelle che rende il tutto giusto  
possibile in base al valore/raggruppamento reg di 3 bit

## Rilevazione e lo collegamento ECRN

Troviamo una sequenza di segnali e ci sono interferenze

La rilevazione si ottiene tramite impianti più ribombanti che fanno da controllori.

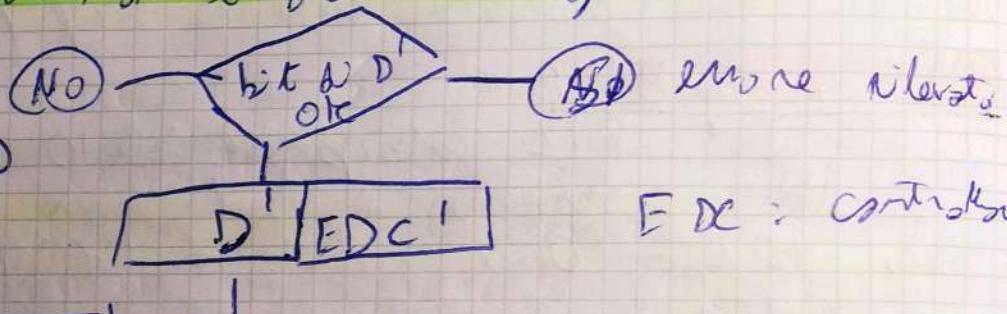
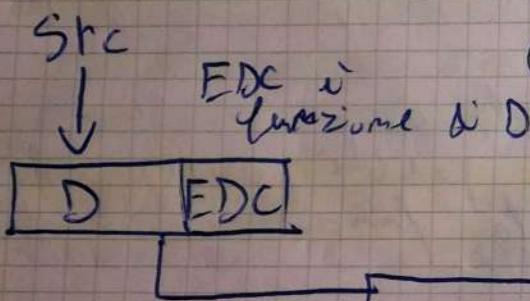
### Controllo superficie

#### Porta

Si usa nei codi che hanno margine di errore positivo a 0

Il controllo di porta dice solo se i bit vanno bene o no.

E dice che i bit sono sbagliati oppure i bit sono giusti non è fatto in questo



EDC: controllo

#### Porta a 2 binarie

Scriviamo stampe e creiamo una matrice

$a_{1,1} \dots a_{1,j}$  | bit controllo righe

$a_{2,1} \dots a_{2,j}$

$a_{i,1} \dots a_{i,j}$

bit controllo colonne

Se c'è un solo errore la riga e la colonna a trave e anche a congegno

Con anche 2 errori il interz. non funziona

CRC codice di controllo di parità  
bitribarazzi codice di controllo di parità implementazione

D: dati da memorare      R: r bit CRC

Funzione i bit e li scriviamo con un polinomio  
esempio:

$$\begin{array}{cccccc} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ x^7 + & x^5 + & x^2 + x^1 + 1 \\ x^6 + x^4 + x^3 + x^2 + x + 1 & & & x^6 + x^5 + x^3 + x + 1 \end{array}$$

Norma:  $(x^6)$   $x^7 + x^6 + x^3 + x^2 \rightarrow$  Segno positivo

Norma:  $(x^6)$   $x^{13} + x^{11} + x^8 + x^7 + x^5 + x^3 + 1$

### Esercizio scommesse $\mathbb{Z}^2$

$$x^7 + x^5 + x^2 + x + 1 \quad x^6 + x^5 + x^3 + x + 1$$

Si fa lo XOR tra i monomi e si sommano

$$x^7 : 1 \otimes 0 = 1$$

$$x^6 : 0 \otimes 1 = 1$$

$$x^5 : 0 \otimes 0 = 0$$

$$x^4 : 1 \otimes 1 = 0 \rightarrow x^7 + x^6 + x^3 + x^2$$

$$x^3 : 0 \otimes 1 = 1$$

$$x^2 : 1 \otimes 0 = 1$$

$$x : 1 \otimes 1 = 0$$

$$1(x^0) : 1 \otimes 1 = 0$$

### Esercizio problema $\mathbb{Z}^2$

Stesi polinomi di gradi 2.  $x^F$  modifica tutti gli  $a$  di  $A$  e solo via

$$x^7 : x^{13} + x^{11} + x^{10} + x^8 + x^7 +$$

$$x^6 : x^{10} + x^8 + x^7 + x^5 + x^3 +$$

$$x^2 : x^8 + x^6 + x^5 + x^3 + x^2 +$$

$$x : x^7 + x^5 + x^3 + x^2 + x +$$

$$1 : x^6 + x^5 + x^3 + x + 1$$

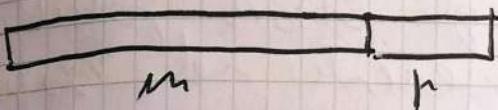
Ottiene questo binomio lungo e procede con degli XOR  
 $(x^m + x^n = 0 \times^m = //)$

Dunque i monomi che sono bipari (di  $x^8$  ce sono 3)  
 riformano rimane uno, quelli pari 0

$$R_1 : x^{13} + x^{11} + x^8 + x^7 + x^5 + x^3 + 1$$

~~D invia~~ polinomio  $M(x)$ : polinomio da inviare  
~~D riceve~~ polinomio generatore  $G(x)$

$R(x) = x^r M(x) \bmod G(x)$  Il trattante usa  $R(x)$  come risposta



Questo è quello che viene fatto  $x^r M(x)$ . Abbiamo aggiunto un "0"

Se le parole da testo diversa da 0 c'è stato un errore (non so dove)

Il  $G(x)$  è usato con degli standardi

del CRC (risparmio 2 int. ridurre già erri, formano ~~controllare~~ "controllare" i bit).

Facciamo nella pratica lo XOR della sequenza dei bit entranti con il polinomio generatore

$$\begin{array}{r} 11010 \\ 10011 \\ \hline 01001 \end{array} \quad \begin{array}{r} 11011110 \\ 10011 \\ \hline 01001110 \\ 10011 \\ \hline 000000 \end{array}$$

Regola: Se il 1° bit non è 0 allora reg è 1  $\rightarrow$  XOR con b e shift  $\dots$  di 1 a b

Se il 1° bit è 0 faccio solo shift

Se alla fine il reg è tutti 0 non ci sono errori

Nella pratica è un simbolo registrato o scritto

### Correzione errori

Distanza di Hamming: Distanza tra due codeword

10001100 XOR

$$11000100 = 0\textcircled{1}00\textcircled{1}000 \quad d=2$$

La distanza è 2 dato che per generare da un codeword altri altri servono 2 cambi di bit

Vocabolario: insieme di codeword

- 1) 000 000 Questo è un codeword da 3 codeword
- 2) 000 111 Distanza del vocabolario: calcolare tutte le distanze nel vocabolario, la distanza
- 3) 111 000
- 4) 111 111 Il vocabolario è la minima.

Questo è un vocabolario da 3 codeword e  $d=3$

L'idea è che il trasmettente invia una parola del vocabolario, ma per gli errori potrebbe arrivare una parola non presente nel vocabolario.

La correzione si basa sul fatto che

$$P(e=1) \gg P(e=2) \gg P(e=3) \dots$$

Ovvero la probabilità che ci sia un solo errore è molto maggiore della probabilità di 2 errori e così via.

Dunque se arriva un input/codeword non presente nel vocabolario gioiamo sul fatto che la probabilità che ci sia stato un errore e che la parola originale è quella con la distanza minore (distanza con la codeword in input)

Per esempio se è vero che un vocabolario con

$$d = 2e + 1$$

per  $d$  valori si ha un vocabolario con  $d = e + 1$

la connessione sarà fatta in base probabilistica

In un vocabolario per  $m$  bit di dati, se  $t$  bit di controllo

$$\text{pari} \quad m+t = n$$

I comb possibili sono  $2^m$ , quelle valide sono  $2^{m-t}$

Parlano di 10001, è ~~ottenuto~~ una codeword valida

quelle a distanza 1 (sono  $m$ ) le restano non valide

$$\forall d=3 \quad \exists m \quad (m+1)2^m \leq 2^n$$

$$\rightarrow m+1 \leq 2^{t-f}$$

10 vocabolari deve avere questa lunghezza

E se  $d=5$ ?

le formule non va bene

abbiamo un codice  $001100\cancel{0}000$

$\cancel{0}$  bit controlli

Dati originali  $10010001100$

$$\begin{array}{ccccccccc} \times & \times & | & \times & 001 & \times & 0001100 \\ 1 & 2 & 3 & 4 & & & 8 \end{array}$$

$$b_1 = 3 \otimes 5 \otimes 7 \otimes 9 \otimes 11 \otimes 13 \otimes 15$$

$$b_2 = 3 \otimes 6 \otimes 7 \otimes 10 \otimes 11 \otimes 15 \otimes 15$$

$$b_3 = 5 \otimes 6 \otimes 7 \otimes 12 \otimes 13 \otimes 15 \otimes 15$$

$$b_8 = 9 \otimes 10 \otimes 11 \otimes 12 \otimes 13 \otimes 15 \otimes 15$$

(sono bit mille per-potenze 2)

$$\text{Troriamo } b_1 = 1 \quad b_5 = 1$$

$$b_2 = 0 \quad b_8 = 0$$

key:  $101100100001100$

Sinistra combit moltiplicati  
bit di bit e  
controlli sono moltiplicati  
(mille per-potenze del 2)

Scrittura in  
base 2

$$3 = 1 + 2$$

$$5 = 1 + 4$$

$$6 = 2 + 4$$

$$7 = 1 + 2 + 4$$

$$9 = 1 + 8$$

$$10 = 2 + 8$$

$$11 = 1 + 2 + 8$$

$$12 = 4 + 8$$

$$13 = 1 + 4 + 8$$

$$15 = 2 + 4 + 8$$

$$15 = 1 + 2 + 4 + 8$$

Supponiamo di ricevere il messaggio con un errore

$101100100101100$

Il ricevente sa quindi che i bit si controllano ( $1, 2, 3, 8$ ) e i bit 1 e 8 sono errati perché non coincidono con le parole (messaggio corretto) che il ricevente ha ricevuto (sono errori).

$$b_1 = 1 \otimes 0 \otimes 1 \otimes 0 \otimes 0 \otimes 1 \otimes 0 = 1$$

$$b_2 = 1 \otimes 0 \otimes 1 \otimes 1 \otimes 0 \otimes 0 \otimes 0 = 1 \quad \text{non coincide con } b_0 \text{ in } 2$$

$$b_3 = 0 \otimes 0 \otimes 1 \otimes 1 \otimes 1 \otimes 0 \otimes 0 = 1$$

$$b_8 = 0 \otimes 1 \otimes 0 \otimes 1 \otimes 1 \otimes 0 \otimes 0 = 1 \quad \text{non coincide con } b_0 \text{ in } 8$$

Il ricevente vede che  $b_2$  e  $b_8$  non coincidono con  $b_0$  e  $b_1$ , ma non ha nessun problema.

Entro 1 bit, uno è uno sbaglio, che influenza 3 elementi; bit entro 2 e 8?

Sì, comunque la tabella delle scorrizioni in potenza due si rende conto che il bit 10 influenza 3 bit 2 e 8.

Il ricevente è tenuto ad accorgersi che il bit 10 è errato.

$101000100001100$   
 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

Scary.

$$3 = 1 + 2$$

Errore nel  
Bit di  
controllo

$$b_1 = 3 \times 5 \times 7 \times 9 \times 11 \times 13 \times 15$$

$$5 = 1 + 4$$

$$b_2 = 3 \times 6 \times 7 \times 10 \times 11 \times 13 \times 15$$

$$6 = 2 + 4$$

$$b_3 = 5 \times 7 \times 12 \times 13 \times 15 \times 15$$

$$7 = 1 + 2 + 4$$

$$b_8 = 9 \times 10 \times 11 \times 12 \times 13 \times 15 \times 15$$

$$9 = 1 + 8$$

XOR:  $0 \oplus 0 = 0$

$$10 = 2 + 8$$

Avendo -1

$$11 = 1 + 2 + 8$$

$$12 = 3 + 8$$

$$13 = 1 + 2 + 8$$

$$14 = 2 + 3 + 8$$

$$15 = 1 + 2 + 3 + 8$$

$$b_1 = 1 \times 0 \times 1 \times 0 \times 0 \times 1 \times 0 = 1$$

$$b_2 = 1 \times 0 \times 1 \times 0 \times 0 \times 0 \times 0 = 0$$

$$b_3 = 0 \times 0 \times 1 \times 1 \times 1 \times 0 \times 0 = 1$$

$$b_8 = 0 \times 0 \times 0 \times 1 \times 1 \times 0 \times 0 = 0$$

Le tre N bs non corrispondono ai bit di controllo

Erre 1 bit; Uno è uno, ma che influenza sul b3? NO (vedi Tossicyn.)

Allora b3 è errato.

**Come si fanno le file**

Pretendiamo una reg. di codeword da trasmettere, da  
scriviamo per colonne e trasmettiamo in questo modo  
in ricezione (facendo tutte tecniche di buffering e ribacchiare)  
se una codeword arriva con tutti errori la fila non  
c'è problema.

Infatti si ritroviamo la colonna e righe e vediamo  
che ogni riga avrà un solo errore forse  
comenzando da una fila

la fila è spalmarne tutti meno di file in più corrispondenti  
per correggerli

## Codici di Hamming

xx1x110  
xx0x011  
xx0x110  
xx1x000  
xx0x011  
xx1x100  
xx0x001  
xx0x110  
xx1x011  
xx1x110  
xx1x001  
xx0x011

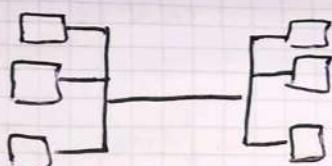
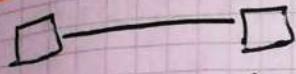
xxxxxxxxxxxxxx100101001110xxxxxxxxxxxx101001  
010100111010011101010010101011

xxxxxxxxxxxxxx100101001110xxxxxxxxxxxx101001  
01010000010111101010010101011

xxxxxxxxxxxxxx100101001110xxxxxxxxxxxxx101001  
010100000101111101010010101011

xx1x100  
xx0x001  
xx0x100  
xx1x010  
xx0x001  
xx1x110  
xx0x011  
xx0x110  
xx1x011  
xx1x110  
xx1x001  
xx0x011

## Access ai link



access multiuso

Un'ente cogne se la pone in direttrice o esce dalla trasmissione di rete stesso

Protocollo Aloha: è un protocollo di accesso esclusivo

- enti di comunicazione che ne sfruttano uno al "caso" che fa ripetizione. Le onde di comunicazione creano un canale e l'estensione del canale "trasmette" canale

Idea: trasmetti, se va bene va bene, se non va bene ti riporta.

Il gane Aloha risulta al giorno il 18,5% delle sue performance.

Cercano di migliorare le prestazioni cercando lo stesso Aloha.

- statistiche che la pone in linea fino a fine degli slot in cui inserire la pone per l'invio (slot temporal)

Con questo accorgimento le prestazioni fanno salto

N.B.: ogni ente gress solo si prova slot

L'entità comune non si trova perché una sola è portata (lo una gressa a tutti "inizia slot")

CSMA: per evitare i mali il canale tra troppe "stazioni"

1-persistent trasmissione appena ricevute le portate (di porta)

P-persistent si ride il termine della precedente trasmissione, si trasmette con probabilità p

Non-persistent stazione si mette un tempo random e ripete a modo il canale

CSMA/CD - molto rapido (Ethernet) (CD = collision detection)

Risolvono i problemi legati a collisione

Fibra: Entrambe le c'è pulsanti, se c'è segnali

Si prevedono "slot"

L'idea di CSMA è vedere se il canale è  
vettivo vuoto e in caso non di tempo le istruzioni  
Per le istruzioni il ricezione rientra avere il messaggio  
Il CSMA CD vuole interrompere le collisioni per evitare  
e non intuire il canale di collisioni.  
Quanto è la probabilità di Ethernet  
Avere i nodi rubare la collisione liberano il canale  
Per riguadagnare la comunicazione dopo questo periodo di  
transmissione in ogni slot il contention period,  
ogni due macchine comunicanti si contendono di slot  
Una macchina vincerà la contesa e trasmetterà, l'altra  
non trasmetterà.

Non sono che i tentativi aumentano, aumentando gli slot per aumentare le probabilità che qualcuno vinca la contesa.

La dimensione degli slot è pre determinata  
Nel CSMA CD dovunque c'è uno i periodi di transmissione  
periodi di collisione / contesa

La contesa è fatta generando numeri.

A gess 0 o 1, B gess

0: ~~prova~~ <sup>dopo t=0</sup>  $t = \frac{1}{2} \cdot 2^0$  se entrambi 0 o 1 lo rifiutiamo

In caso abbiamo solo 1 slot

l'algoritmo di ~~quello~~ binario esponenziale

Alla riga i regole tra 0 e 1, alla riga t+1  
0, 1, 2, 3, ... (n è 3 soggetti t=3)

Si fa max 10 volte (0, 1, 2, ..., 1023)

t: tempo per inviare 512 bit

## Protocollo a mappa bit entro le collisioni

Ricorda le macchine e le macchine, già in macchina ci sono i bit.

Le macchine ti ti premota lo slot e le macchine

Ogni slot è 1 bit.

Fatto la prenotazione tutti sanno chi deve parlare e con quale ordine

Poi dopo la comunicazione (regenerazione) c'è di nuovo il periodo di prenotazione

La distribuzione non è equa: macchine a numeri alti sono privilegiate nel basso traffico.

Se ho progettato per 10 macchine e invia l'offerta come riorganizzo il tutto? Se vorrei comunicare a tutti, non è semplice

## Conteggio binario e ritorno CANBUS entro le collisioni

0: assenza di segnale Se due macchine dicono 1, dicono lo stesso cosa (no prob)

1: segnale Se dicono 0 non c'è segnale

Se una ha 0 e l'altra 1, vince chi

dice 1. Res: sul canale c'è solo 1.

Idea: momento le macchine che inseriscono il proprio id sulla rete

Come si posso avere il MAC Address

Se una macchina grande fa contesa lascia momentaneamente il canale

Il ritorno non è equo: chi ha molti 1 è molto vantaggioso

## Protocollo a token

entra le collisioni

## Token bus

può essere usato nelle centrali

Cresce il latency e un token che fa il  
gioco di forbire, il token già per forza  
è tutto il gioco di forbire.

Problema: il token è un messaggio e ha tutte le  
sue problematiche

Se arriva un nuovo ento nel canale come fa a  
decidere di forbire per inviare senza il token?

Se una macchina è spenta e gli "arriva" il token  
il token è perso.

Se il token si perde ci deve essere qualcuno che lo  
genera, i più gestire tramite timer e altre tecniche

Il token bus crea bisogno di trovare un  
coordinamento che è già molto complicato.

ESMA CD / ethernet ~~contatta~~ prova, se va bene ok

Cablaggi:  $(10 - 10 \text{ Mbps})$

10 base 5: cavo coassiale gommo molto rigido (percorso di lunghezza) trazione isolante e calza metallica per protezione.

C'è anche la spina esterna

Segmento: 300 m. Max 5 pezzi (2,5 km). 100 m di

Punto di interconnessione normale = Presa a Vano

• Per l'isolante e per contatto col cane per rendere normale

• Con un cavo fondono il nastro poi si fa l'onda vidi

10 base 2: cavo coassiale flessibile.

Segmento: 185 m con 2m600 max 3 pezzi, 300 m

10 base T Twisted pair. E' un cavo a coppie intrecciate

Le coppie sono 5, permettono un segmento di 100 m

Lo schema di una rete in hub centrale che crea collegamenti diretti verso le macchine. Si può immaginare come una stella.

10 base F: una fibra ottica, arriva fino a 2 km e si usa per le punto-punto come nel Twisted

Entro i vari tipi di cavi intrecciati

Cat 5 100 Mbps 100 MHz

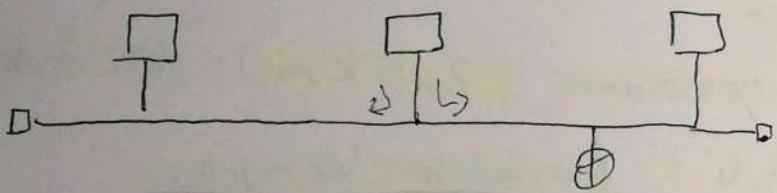
Cat 5c 100 Mbps / 1 Gbps 100 MHz

Cat 6 1 Gb / 10Gb 250 MHz

Cat 6a 1 Gb / 10Gb 500 MHz

## Problemi di token size 2

Vorremo meccanismi per i terminatori per evitare  
che bloccare il regnale ed entrare rimanga in giro.



Terminatore: non fa rimbalzo  
il regnale

Domino di collisione: se ci sono in collisione il regnale  
rimane dove vi siamo intaccato in questo

## Frame Ethernet

Protocollo	Dest MAC	Src MAC	EtherType/Size	Payload	CRC
8	6	6	2	...	3

(lunghezza in byte)

EtherType / Size: indica lunghezza campo dati se > 1500  
valore minimo 1500, indica il tipo di frame se invece  
abbiamo valori maggiori

Dunque se  $\geq 1500 \rightarrow$  EtherType,  ~~$\geq 1500 \rightarrow$~~  Size  
CRC occupa 3 byte

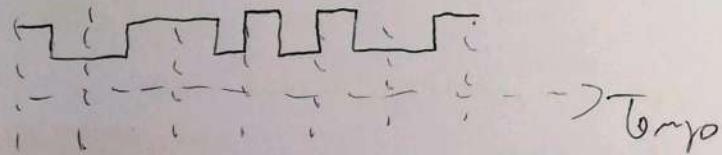
## Cab/ice Mandriller

$$\text{alto-basso} = 1 \quad \text{basso-alto} = 0$$

Viene definito da varie tempozzi, l'obbligo tra le due macchine deve essere perfezionamente sincronizzato.

Faccio così perché "Notare una variazione è già reattivo e notare i bit nel regolare modo"

La sincronizzazione è nota nel preambolo



Le variazioni da segnale agli estremi dei varie temporali non sono nulla

## Fast Ethernet

100base TS	Twisted pair	segmento
100 base TX	--	100 m
100 base FX	fiber optic	2000 m

Sceggiamo i cavi → goc offribili (bornei oltre 640 byte h  
lone, Oceanis)

Se voglio percorrere le colline; la lung. max del cavo  
individua la lunghezza minima della lone da inviare  
per corso

Usa le 5 coppie intrecciate: 1 in alto, 1 in basso e  
oltre due alternativi

Ho  $33 \cdot 3$  (già E) Mbps in una linea e 33 mbps

La cab/ice Mandriller viene abbassata

TX: 3b5b (v. ms CATS)

TS: 8BGT (CAT3)

8bit  $\downarrow$  subfisi in 6 Tristate

100base FX	Fibra	2 Km	
gi-Hub	per Ethernet	modo (impostato):	last switch
<u>gigabit</u>			
1000 base SX	fibra	550 m	STP copper schermata
1000 base LX	fibra	5000 m	UTP copper non schermata
1000 base CX	2 pairs STP	25 m	
1000 base T	2 pairs UTP	100 m	

### Come si arrivano a 1000 base T (CAT5) ?

- numero HB5B 100 → 125
- uti di cavo cat5e/cat6e 125 → 500
- full duplex → 500 full duplex
- 5 livelli impacchetti 3 500 → 1 Gbps full duplex
- i max max FEC per encl (comezione error e burst)

### Autore a testi: firsti - gli organizzatori legale

Fatto per trovare lezioni e generare da ~~ufficio~~ ~~ufficio~~ la rete e proprie regole di link da imparare

~~TESTA~~ VITERBI  
codifica decodifica standard 802.326

~~perché non si usa il codice?~~

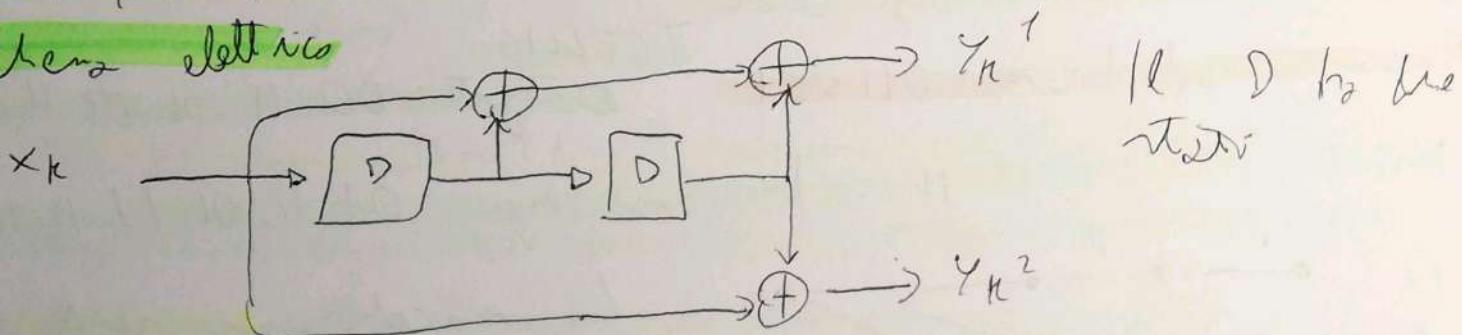
Eliano transmette a 2 Gbps (fidelità ~ 99.99%)

## Ribosom 22 mit 1000 base

Stiamo trattandosi di 26 bps (100% fibbanza)

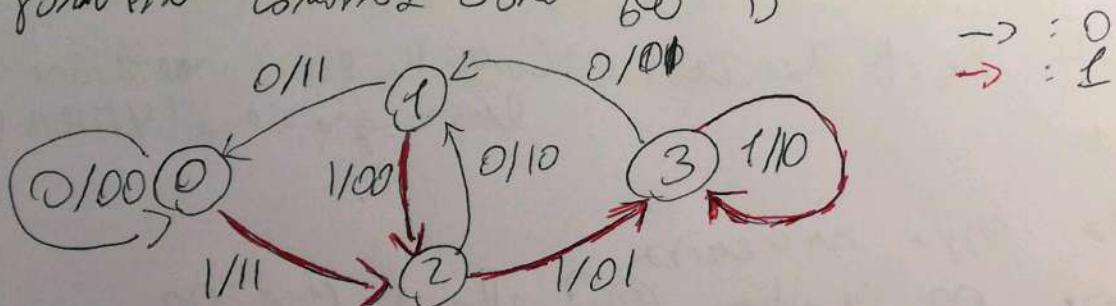
Generano i codici da inviare nel file con la **cabifica**  
di Trax e controllano il codice (del ricevente) con la  
**decabifica** di ritorno

## Schemi elettrici



Dalbit  $y_k$  gewuso  $y_{k+1}$ ,  $y_{k+2}$

A Urello tonico lo vediamo come una modulazione  
2 noti finti con 3 noti (1, 2, 3, 4) ovvero le  
possibili combinazioni dei D



Lo stato iniziale è 0. Se deve salpicare 0 (bit) lo stato rimane 0 e 0 (bit) lo trova in 00 se deve salpicare 1, la traduzione è 11 e n passa allo stato 1

- Stato 0 e 3 sono simmetrici tra loro mentre lo stato 1 e 2 conservano rispetto alla variazione
  - 0 e 3, 1 e 2 sono simmetrici
  - stadi 0 e 3: antisimmetrici

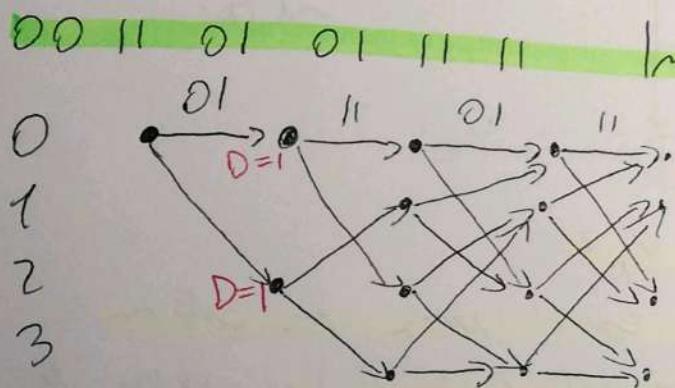
Unità di 1 o 2 versi D e 3: 2 distanze  
 frecce uscite in D=1: 0 (input)  
 frecce uscite in 2 e 3: 1 (input)

Ciò consente di individuare ogni blocco e le distanze sono  
 minime. Ci permette la decodifica esatta con la  
 forza bruta comezione degli errori.

Esempio: codificazione 011001

### TRELLIS

Output: 00 11 01 01 11 11



Invece errori 01 11 01 11 11 11

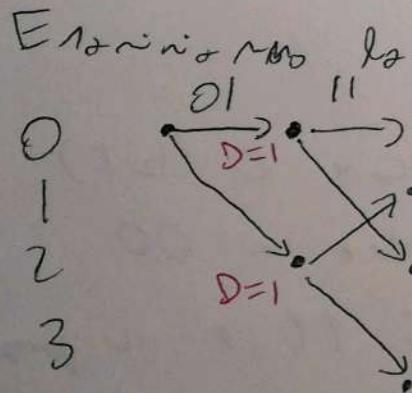
La modifica si nota  
 come c'è un errore nella  
 prima coppia (non dovrà  
 essere questo in nessun modo).

Nota: se l'errore è il 1° o 2° bit

Tiene conto delle distanze

per cui  $D=1$

Controlla e correzione  
 errori grazie a VITERBI



corregge macchina

00... se abitano ①② allora la 1<sup>a</sup> è 00

11... = ① → ② allora la 1<sup>a</sup> è 11

00... //

11... //

↓

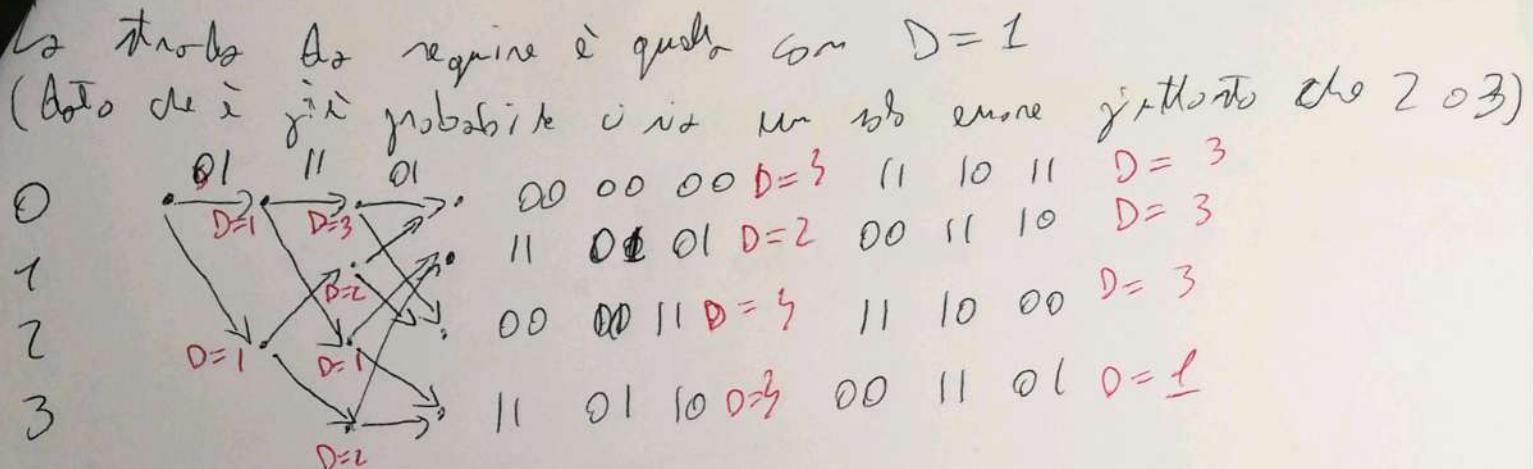
$D=3$  00 00 ①② 2 volte

$D=2$  11 10 ① → ② ... ② → ①

$D=1$  00 11 ①② ... ① → ②

$D=2$  11 01 ① → ② ② → ③

Distanza  
 della  
 codifica  
 in input



A destra, per ogni jettato finale abbiamo 2 strutture possibili.  
 Eseguendo l'algoritmo in tutte le copie del fine, il

jettato corretto è quello che ha  $D$  minima  
 (il jettato corretto grazie alla macchina C permette di generare lo stesso jettato)

Dunque naturalmente la conseguenza sarà essere (d'ogni G)  
 che i due bit eratici sono vicini.

Il CRC ci aiuta a identificare (ma non a sbloccare  
 comunque)

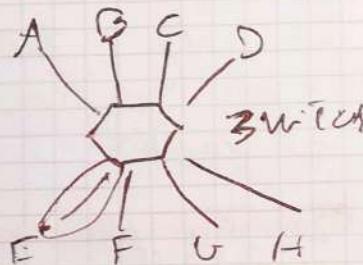
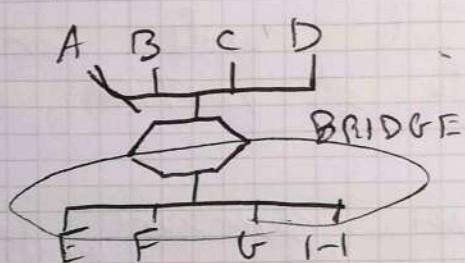
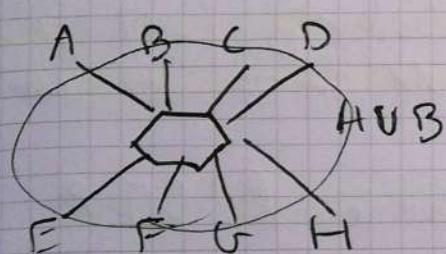
## Connessione di LAN tra di loro

Repeater: amplifica il segnale. Lavora solo su livello fino senza ~~ogni~~ controllo i bit.

Potenzia forse il segnale: infatti lo standard għiex max k-fogħġi

HUB: è un dispositivo għiex konċentra tutto; il segnale de tħalli b'porti e l-impresja se tħalli b'porti

Bridge: serve a connetter due LAN separate da tipi differenti. Neku minn conversione. Il kompożiżiun i-komplimenti connetter due ethernet



Hub: non intelligente

Bridge: intelligente

Switch: intelligente

Q: Benviato & Collisione

Switch (risolvi colliżjoni) raccorre per ethernet ed è portato, ha una scheda per ogni porta, il benviato & collisione è solo porto ~~ogni~~ porto.

Per questo motivo LAN n. 1 non causa fast collisione

Gli switch hanno porte opposte per essere collegati ad altri switch. Normalmente i porti sono banditi għid luu in uplink

Routier: lavora su livello di rete, interpreta i pacchetti IP

Ogni switch ha una tabella che memorizza le info nelle posizioni dei ponti (tavola di flooding)

Nelle tabelle ci sono 

Indirizzo	Interface	Tempo
-----------	-----------	-------

Il flooding viene eseguito contenendo il grafo come un albero (non c'è uno cicl.)

Si potrebbe creare un protocollo che fa un grafo circolare in modo possibile per l'intervento

Il protocollo STP

Quando un switch trova il DB ~~presente~~ ha le informazioni per fare forwarding.

Le info vengono inviate con flooding (che non ha intuito gli switch fino che non è in un albero)

Se ho un grafo ma voglio che avrei l'STP  
de determinare un albero del grafo

Comodo di LAN: tutto il rettangolo di BMS contiene  
tutte le connessioni disponibili

## VLAN

Unica **stessa** linea tra **non** vede **linee** diverse  
vedono tutti.  
E.g.: **injettare l'accesso** a diversi **linee** (con  
solo software).

Criano le **LAN virtuali**:

Dispositivo Port based VLAN: gli **switch** che le  
macchine da 1 a 5 non **vedono** le altre **LAN**  
(se 1 ha broad cast, le vederà 1, 2, 3, 4 e 5)  
Un dispositivo esterno non dovrebbe vedere la **LAN**  
tra 1, 2, 3, 4 e 5  
Sostanzialmente lo switch (il dispositivo è quieto) è  
partizionato e negando le singole porte a ogni **VLAN**.

Vengono applicate anche le tagghe & contiene all'interno  
di questo switch per far comunicare macchine di VLAN 1  
con macchine di VLAN 2 (sestamente)

### Bisogno etichettare le frame ethernet

IEEE 802.1Q: vengono inseriti 3 byte nella frame  
per identificare la frame (identificano il tipo di  
protocollo e il VLAN ID)

VLAN Tagged: c'è il Tag

VLAN Untagged: non c'è il tag

Le tagghe permettono di avere stessi indirizzi fisi  
in LAN & Network, si identificano le LAN con dei  
campi opposti nella frame ethernet

Primi 2 byte: protocollo VLAN

Secondi 2 byte: Priority, CFI, VLAN-ID

Nelle Tagged viamo una porta dello switch per  
fare l'uplink

Nelle Untagged se non è previsto un core che  
collega le LAN o un qualche tipo di software  
nello switch stesso le LAN fra di loro sono  
interconnesse, one porta di uplink basterà usare  
una per VLAN mentre nello tagged una è basta

### Scuze VLAN

- Risparmio
- Flessibilità
- Prestazioni
- Sicurezza

Livello fisico si occupa di trasportare segnali

Per trasportare informazioni da un posto all'altro

- Serve:
- canale trasmissivo (a rete)
  - operatore di rete
  - filizzazione di rete

Dare trasporto nulla maniera più comoda possibile  
i segnali

Segnale: informazione del rapporto

Onda che si propaga in un reticollo: la informazione arriverà

In questo caso mettiamo: subtratta

Molecole e costante di segnali bene non trasmettere informazioni

Mezzi: wireless / ottici ecc...

Nei mezzi ci sono conduttori elettrici, onde wireless o onde "miche"

Analisi di Fourier

Un segnale di periodo finito si può rappresentare con una somma di infinite onde sinusoidali di frequenza ed ampiezza opportune

$$g(t) = \frac{1}{2}c + \sum_{m=1}^{\infty} a_m \sin(2\pi m f t) + \sum_{m=1}^{\infty} b_m \cos(2\pi m f t)$$

$f = \frac{1}{T}$  è la frequenza del segnale

Un moto per l'operatore non  
è quale in moto  
"bambini"  
(interstiziale)

Cresce la commissione

$$a \sin(x) + b \cos(x)$$

$$\sqrt{a^2 + b^2} = \sqrt{0,4^2 + 0,916^2} = 1$$

a e b direttamente per le rese e coseno (teorema)

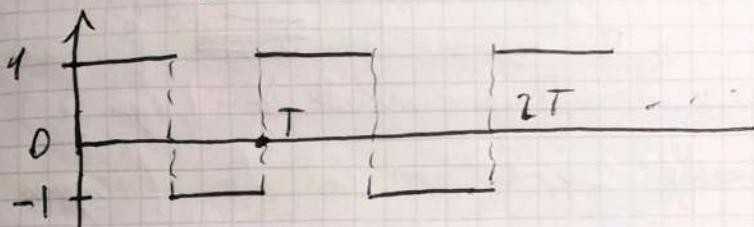
Come calcolare  $a_m$  e  $b_m$ ? (e c)

$$\begin{aligned} a_m &= \frac{2}{T} \int_0^T g(t) \sin(2\pi m \nu t) dt \\ b_m &= \frac{2}{T} \int_0^T g(t) \cos(2\pi m \nu t) dt \\ c &= \frac{2}{T} \int_0^T g(t) dt \end{aligned}$$

Risolvendo gli integrali abbiamo i coefficienti sufficienti della funzione

### Signde Onda quadra

$$g(t) = \begin{cases} 1 & \text{se } 0 \leq t < \frac{1+2m}{2} T \\ -1 & \text{altrimenti} \end{cases}$$



Caratteristiche  
onda  
tempo t  
Ampiezza  
frequenza

Applicando Fourier

- $a_m = 0$  se  $m$  pari
- $b_m = 0$

$a_m = 0$  se  $m$  pari

$a_m = \frac{4}{m\pi} \text{ se } m \text{ dispari}$

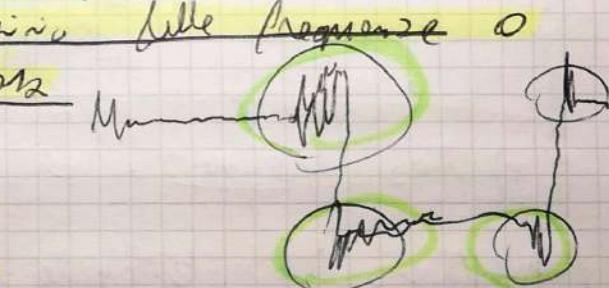
Si calcolano solo ampiezze e



poi si sommano e ottengono

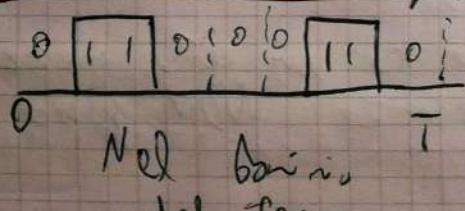
un grafico simile a un'onda quadrata (Pieno di ondette gradi giri in avvicinamento all'onda quadrata). Gran parte delle frequenze sono al tempo e la stessa cosa

Notare che ci sono zone di cancellamento



Nelle zone di cancellamento ci sono dei picchi

### Premesso un segnale limite



Nel dominio delle frequenze ↑ ↘ nel segnale

frequenze alte cancellate nel segnale

Con  $a_m$  e  $b_m$  si ricava lo spettro del segnale.

Si dice a copiare come viene deformato un segnale quando

$$8 \text{ Mbps} \rightarrow 1 \text{ bit ogni } \frac{1}{8 \cdot 10^6} \rightarrow$$

1 bit parso = 1 MHz

256 bit parso = 256 MHz

### Condizioni

- perfetti: non hanno attenuazioni o ritardi nella propagazione
- ideali: hanno ritardo costante nella propagazione
- reali: causano attenuazioni e ritardi in funzione della frequenza dei segnali  
Tutti i mezzi hanno ~~una~~ zone dove le ~~alcune~~ frequenze "non passano"  
(zone si intende intervalli di frequenze)  
Ci interessa lavorare nelle zone in cui le frequenze ~~non passano~~ sono ~~una~~ zone (ritmo soggetto banda del canale)  
La zona ~~debole~~ lunghezza di banda del canale  
Ese: nella fibra ottica le frequenze che passano bene sono i colori
- dimezziamo il tempo di transizione:
  - bit rate doppia
  - se la frequenza si allarga al doppio

Aumentiamo il bit rate le frequenze che riescono a passare nel canale diminuiscono.

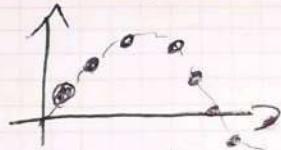
Ricostituire il segnale è compito

Questo è il motivo per cui se ho un certo mezzo che funziona con certe frequenze NON posso andare più veloce di lui

## Quantizzazione

(fatto in modo trascurabile non fa differenza)

Dobbiamo quantizzare il segnale



Prendiamo i quanti: numeri compioni che descrivono approssimativamente la curva

Più quanti prendi più meglio approssima il segnale

Nyquist: legge della lunghezza di banda di un segnale con la quantità di info trasportabili

H: lunghezza di banda del segnale

(in Hz)

V: numero circoli presenti nel segnale

H: quantizzazione  
gradi angolare

$$\text{Max bit rate} = 2H \log_2 V \text{ bits}$$

V: verticale (circoli)

In realtà potrei sottrarre H e V (bigrande poi vengono generato e ricevuto da b form)

Rumore: segnale che si sovrappone alla comunicazione  
di disturbo

"In una situazione di rumore senti più lentamente"

Abbiori il bit rate.

Il rumore è qualcosa che si mette al segnale e lo deforma

Non puoi quantizzare troppo, se ho quanti vicini al rumore potrebbe far diventare un quanti vicino a un quanti vicino.

Rumore Tono: Non sono a Ora di Temp.  
caso nei canali oscillatori di disturbo \*

## Shannon

H: lunghezza banda segnale

$$\text{Max bit rate} = H \log_2 (1 + S/N)$$

S: potenza segnale

N: rumore canale

definire un SNR che non venga negativo (altrimenti il ricavato non capirebbe nulla)

\*  $P = S/TBW$ : Potenza tipica da T, t (contatto) e BW (frequenze)  
Il rumore c'è a tutte le frequenze ma in alcune non ci disturba.  
Più è lungo il canale più fluttuazioni (dovute al P. tonico) ci sarà

## Aumentazione

Si stabilisce come si fa l'ammontare tecnico:  
proporzionale alla lunghezza del cavo.

All'aumentare della distanza il segnale viene attenuato.

Il rapporto segnale su rumore  $\rightarrow S/N$  se ne sente  
cavale telefono: 10 - 3000 Hz

Rapporto segnale rumore 30 dB

Come Shannon  $\rightarrow 30000 \text{ bps}$   
decibel  $r_{dB} = 10 \log_{10} \frac{A_1}{A_2}$

Per Shannon massimo  
contiene; Cattura (V) max

abb il rapporto segnale  
rumore abbai il bit rate  
ma nei già insieme di rumore)

## In genere

- qui sotto è il bit rate  $\rightarrow$  qui è sotto la base  
del segnale ( $\rightarrow S/N$  critico)
- qui lungo è il cavale  $\rightarrow$  maggiore è il rumore,  
diminuire capacità canale

Per ogni regnale abbiamo avere un canale ideale

In certi casi abbiamo per modulazione e portante una regnale  
a frequenza opportuna

$H_o$  in regnale  $S(t)$

$H_o$  portante  $p(t) = A \sin(\omega t + \phi)$  Frequenza portante

Creiamo  $g(t)$  che "incide"  $S(t)$  e  $p(t)$   $\downarrow$   
Dove trasmettere  $S(t)$

$$g(t) = p(S(t), \rho(t))$$

Ci sono vari modi per ottimizzare  $f$

• orizzontale

• frequenza  $\geq$  tipi di modulazione

• lunga

$$\text{Anziozz: } g(t) = A \sin(S(t) \sin(\omega t + \phi))$$

$$\text{frequenza: } g(t) = A \sin((\omega t + k S(t)) t + \phi)$$

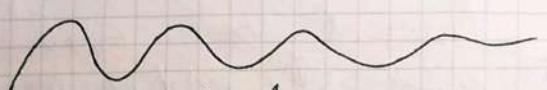
$$\text{fase: } g(t) = A \sin(\omega t + k S(t) + \phi)$$

Aumentazione poi "diluisce" la portante

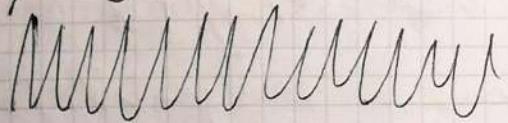
## Modulazione di Ampezzza AM

$$g(t) = A \sin((\omega + k s(t)) t + \phi)$$

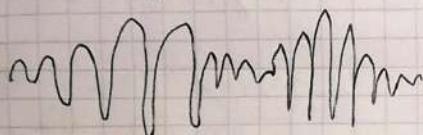
$$g(t) = A K s(t) \sin(\omega t + \phi)$$



Orba



potente



AM ( $g(t)$ )

La tensione ha una forma sinile e una ampiezza modulata dalla frequenza.

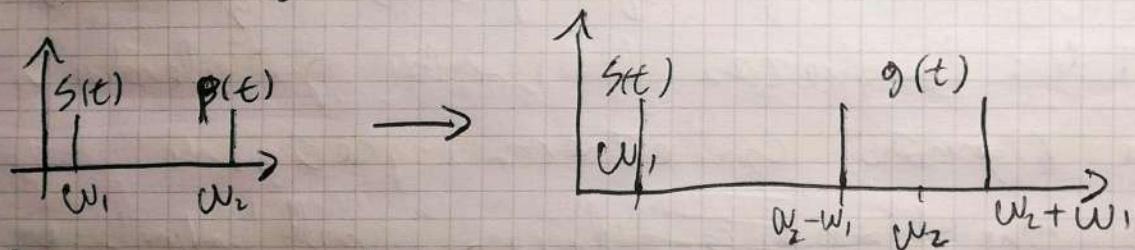
Metodo analitico per la 1

$$s(t) = \sin(\omega_1 t) \quad p(t) = \sin(\omega_2 t)$$

$$g(t) = s(t) * p(t) = \sin(\omega_1 t) * \sin(\omega_2 t)$$

Per le formule di Werner ...

$$g(t) = \frac{1}{2} * [\cos((\omega_1 - \omega_2)t) - \cos((\omega_1 + \omega_2)t)]$$

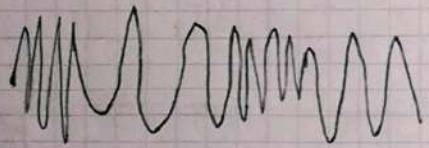


Da  $s(t)$  che era molto bassa nella frequenza  $\omega_1$  (che sarebbe quella ideale) e non avvicinò a  $\omega_2$  con le formule.

Trasmettendo  $g(t)$  si ricava facilmente il prodotto matematico inverso per ottenere  $s(t)$

## Modulazione di frequenza FM

$$x(t) = A \sin((\omega + k s(t)) t + \phi)$$



Modulando la frequenza dell'onda portante l'effetto è conservazione sulle zone positive (dell'onda originale) e distorsione sulle zone negative.

## Modulazione di fase PM

$$x(t) = A \sin(\omega t + k s(t) + \phi)$$

Si ride a quella di frequenza. Non è un caso si può dimostrare matematicamente che gli effetti di FM e PM sono legati da operazioni di integrazione/derivazione.

## Analogico o digitale

Grandezza analogica: più simile vera soluzione di continuità

Grandezza digitale: più simile solo valori ben precisi in un numero finito

Esempio analogico: lunghezza di banda già limitata, subisce distorsioni per ogni processo rigenerativo

Esempio digitale: lunghezza di banda già elevata, può essere rigenerato con estrema precisione

Si può dire che l'analogico lavora nell'infinito e il digitale nel finito.

Analogico e digitale sono due modi di vedere una trasmissione, non ci sono (trasmettitore che lavora con infiniti livelli?)

Il segnale analogico ha un enorme numero di stati binari (la macchina non tiene quasi 2 distinguibili)

Il numero di digitale se il numero di stati è finito.

In questo modo gli stati sono distanti e il rumore non diventa un gran problema

### Ricognizione

Analogico: tantissimi stati (il rumore può avere valori)

Digitale: pochi stati già facilmente distinguibili  
(anche in presenza di rumore)

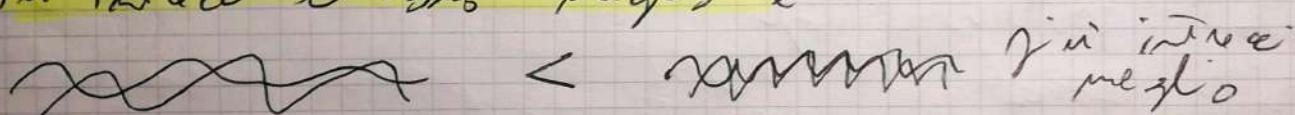
### Mezzi trasmessi

Oggetto telefonico: coppia di cari intrecciati ad elica (caneando)

Avrei intrecci fatti male provo il fenomeno di spieze  
e ci sono correnti interne con interferenze

Le spieze negli intrecci fatti bene tendono ad annullarsi  
tra di loro  $\rightarrow$  meno interferenze

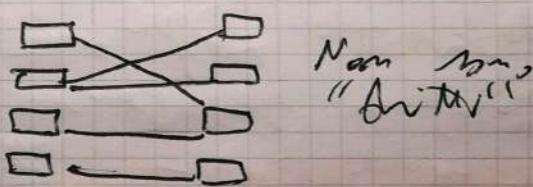
Più intrecci ci sono meglio e



Non avrei intrecci per formare un'antenna  
coi due cari (ovvero turbare la propagazione delle onde  
elettromagnetiche)

### Oggetto intreccio

Ingresso



### Cavo Coassiale



Sigl. babbuciate coassiale = intrecciato

### Cavalli

Un cavaliello d'aria circola nel corpo come un'antenna.  
Il cavaliello limita l'assorbimento di energia dell'esterno  
Questo tubo serve di circuito eletromagnetico  
Ha il buco verso la cassa riflettore (dove si concentrano i interni)

### Doppioni intrecciati

I cavalli vengono intrecciati annualmente fra di loro i binabbi elettrici (non per intervento)  
L'intreccio permette i cavalli genuine sartorie (a condizione che le distanze fra i cavalli sia infinitamente e neglige) riducendo distanze e aumentando le probabilità → corse migliori  
CATGA: 4 coppie intrecciate che a loro volta si intrecciano regalando un movimento circolare fatto da un "girino" che infatti (ogni coppia ricorda da un insieme)

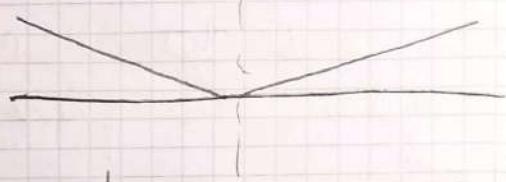
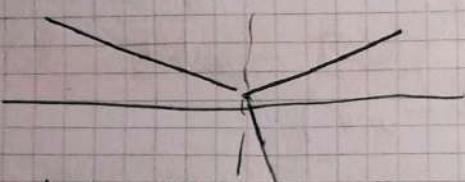
Si possono fare regole già accorgimenti (cavalli) per avere interazioni già limitate → velocità maggiore

## Fibre ottiche

Perciò i dati sono skinori, ente termico  
(non soffre di interruzione) (potentissimo)

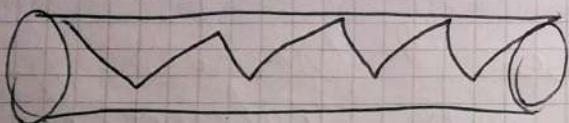
Un fascio luce solo nell'interfaccia di due mezzi  
differenti si rompe in due: riflessi o rifratti

Se si oltrepassa un angolo di incidenza ~~il fascio~~  
le componenti riflette e rifratta



Nelle fibre ottiche si fa rimbombare continuamente  
il segnale in modo opportuno

Rimbalzo  
di  
grado  
a 10<sup>14</sup>/15 Hz  
di  
breve  
per  
basso



## Fibre

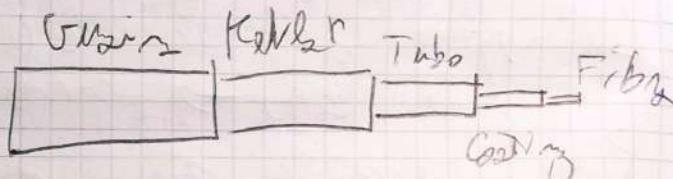
Ceramica: fibra

Kevlar: protezione tagli

Tubo: fibra

Coating: rinforzo meccanico

Fibra (vetro)



Collegamenti: nodi e puntelli si connettono (tubazione)

Dispositivo di fusione: corri: ho ricorso a questo  
tempo che li fonda

Due tipi di Fibre

- Multimodale: prendi più raggi riflessi contemporaneamente

(ogni tipo di riflessi e rimbombi)

- Monomodale: luce avanza singolarmente verso

riflessioni (le riflessioni ci sono ma  
in un solo modo)

Sia contro piacemente tra corri? Lo sono di per sé  
rimbalzo è già giusto nel monomodale

Nella fibra c'è uno van modi per avere simboli  
avanti (grado, rettivo & retro totale, inserzione...)

### Reti wireless

E' possibile / utile / necessario fare connessioni senza  
cavi

Pretoriano → radio  
(e bocce qualsiasi  
interferenze)

Da fare se è l'unica  
alternativa

Modem fanno anche comunicare di dati in tempo reale  
che già hanno le linee

Nel g. le reti telefoniche

I provider tendenzialmente hanno un filtro passante  
per tutte le linee tutte le freq.  $> 3000$  Hz  
tasso errore RET telefoniche =  $1/10^5$

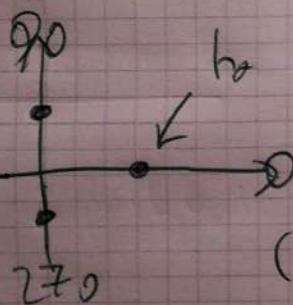
tasso errore:  $1/10^{13}$

Come uscire dalle linee telefoniche? → Modem  $\rightarrow$  trasmetti  
prende da digitale  
ad analogico e  
viceversa

Il modern modula la portata in  
in base alle specifiche e si aumenta  
il throughput

I moderni spesso usano lo collaborazione: ho un g. solo  
ogni punto porta un certo numero di bit  
per il tutto

Due moderni che comunicano devono avere lo stesso  
controllore. (ogni moderno ha il suo standard)



ha un'angolazione e una fase (angolo)  
(d'intorno a  $(0,0)$ )

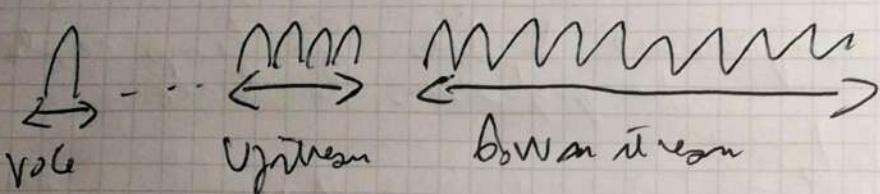
confronto lo standard ha un  
(potendo usare il migliore possibile  
per le prove) standard comune e

Oggi con le tecnologie ADSL non le prestazioni sono efficienti.

## DSL

Il canale ha subito una modifica: stato fatto il braccio nello stesso canale per evitare i disturbi che le frequenze che vogliono uscire per internet.

Nel DSL da 256 kHz canali



Frequenze dedicate  
a voce poi più...  
e poi frequenze im-  
pulsive e downstream

Nella DSL ci sono margini di rumore SNR e linee di attenuazione.

$$\text{Distanza cavo (metri)} = (1000 * \text{Max-attenuation}) / 13.81$$

Si è pensati così a VDSL che aumenta molto le prestazioni.

I sistemi DSL e VDSL sono usati da cabinet a casa (es: FTTC).

FTTH: Fiber To The Home

Ricordiamo che DSL serve per usare le vecchie linee telefoniche già avviate anche accesso a internet (usando freq. già esiste).

DSL e VDSL sono poi pensate per essere usate nel collegamento cabinet - casa.