Modo di trasferimento a pacchetto

Servizio di rete:
- con connessione (circuito virtuale)
- Senza connessione (datagramma)

Multiplazione:
- dinamica

Commutazione:
- attraversamento a immmagazzinamento e rilancio

Architettura protocollare:
- multiplazione e commutazione a livello di rete

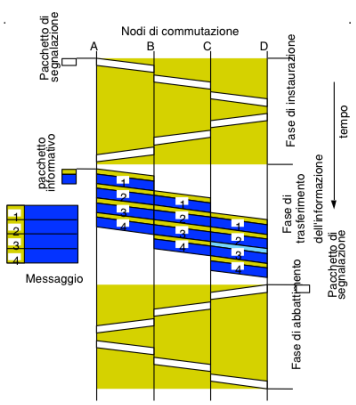
I pacchetti sono le PDU del livello di rete dove i protocolli hanno lo stesso “linguaggio”.

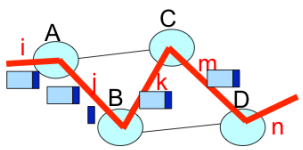
Pacchetto

Un modo di trasferire i dati è un insieme di scelte su come multiplarli sui rami e come commutarli sui nodi, questo si rispecchia sulle prestazioni di integrità e trasparenza temporale.
I pacchetti sono le unità di dati trasferite in rete e sono composte da:
- Un’**intestazione** (*header*), contenente informazioni di controllo della comunicazione
- Un **campo informativo** (*payload*), contenente le informazioni dell’utente (a livello di rete all’IP i dati arrivano dall’entità dello strato superiore che in questo caso si chiama trasporto)

Servizio di rete: con connessione

Immaginiamo due sistemi A e D che devono trasferire i pacchetti da A a D.
La rete attraverso la quale si trasferiscono i pacchetti è quella in alto a destra : è un grafo con nodi e archi (se non ci sono gli archi i pacchetti devono seguire un modo indiretto).





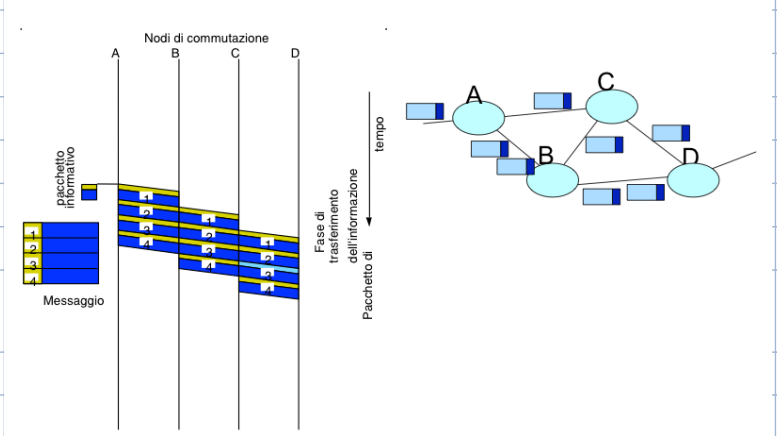
Per andare da A a D ci sarebbero più percorsi, quello scelto è quello in rosso, quindi si passa per i nodi intermedi B e C.

A sinistra c’è un diagramma spazio tempo, in orizzontale ci sono i nodi via via attraversati, in verticale il tempo.
In un servizio con connessione c’è interazione tra i vari nodi : fase di instaurazione, fase di trasferimento dell’informazione e fase di abbattimento.
Un pacchetto trasferito da A a D è un parallelogramma, l’altezza di questo è il tempo per trasmettere il pacchetto .

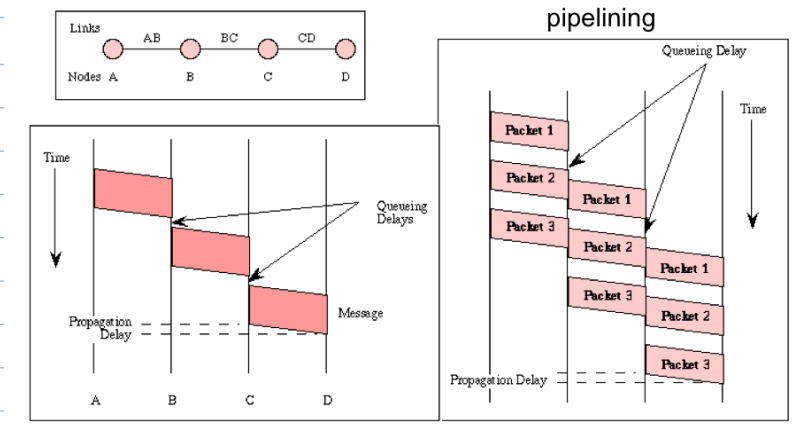
A emette i pacchetti uno dopo l'altro (back to back), evidentemente perchè ce li ha pronti. La figura dice che B inizia a lavorare sul pacchetto appena ha ricevuto tutto, se C a cui devo mandare i pacchetti sta lavorando i pacchetti di B si mettono in coda. Il tempo che intercorre tra B che riceve il pacchetto e il tempo in cui è pronto a trasmetterlo, secondo il diagramma passa 0. Nella realtà non è così, in questo caso viene trascurato il tempo di attesa in coda (in quanto è molto piccolo).

Pipelining = è una tecnica in cui vengono inviati più pacchetti su una singola connessione senza aspettare le risposte corrispondenti.
È un meccanismo di parallelismo che ci permette di ottimizzare i tempi, mentre A trasmette a B , B trasmette a C. In una connessione c'è sempre un iniziatore (A) e un risponditore (B), chi decide di chiudere la connessione può farlo chiunque. Nella figura B.

Servizi di rete: senza connessione

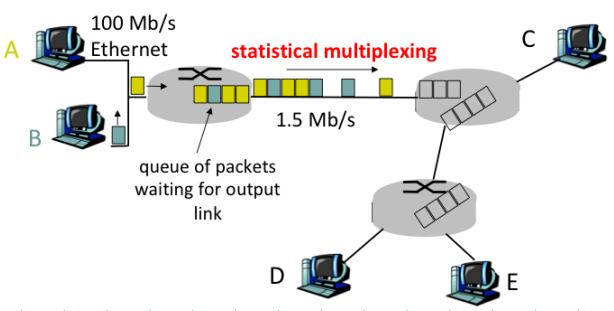


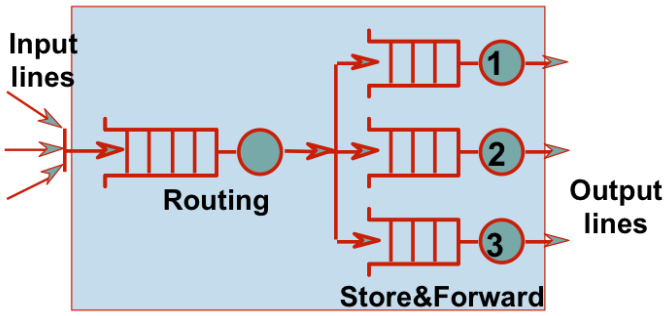
Se A vuole mandare pacchetti a D, prende e li manda. In ogni intestazione bisogna scrivere ogni info utile affinchè i nodi intermedi sappiano cosa farci. Non essendoci connessione ogni pacchetto fa storia a se.



A sinistra senza pipelining viene usata una connessione alla volta.
Non conviene con il pipelining ne mandare pacchetti troppo piccoli (fanno pesare troppo l'intestazione) ne troppo grandi perchè ci sarebbero dei ritardi più grandi.
È possibile trovare una lunghezza ottima del pacchetto.

La capacità di un ramo è di solito superiore a quella necessaria,quindi viene condivisa attraverso la commutazione.
I pacchetti arrivano da tanti ingressi quando arrivano messi in un buffer dove si accodano e quando devono uscire vengono presi da questa memoria.
Buffer= memoria del router dei pacchetti.
Non si può avere una memoria grande sia per i costi sia per il ritardo della queue





Non necessariamente escono nell'ordine in cui sono entrati.

Tutto il trasferimento dei pacchetti avviene nella rete non fisicamente.

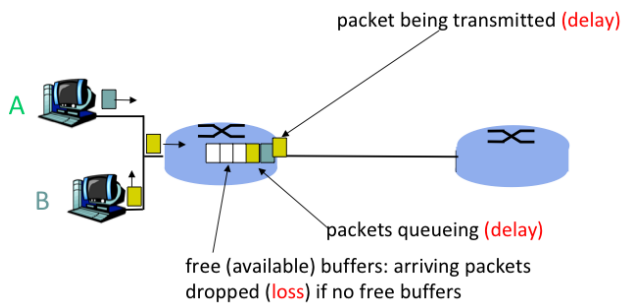
Principali metriche di prestazione

1) Ritardo
2) Perdita
3) Portata

Ritardo

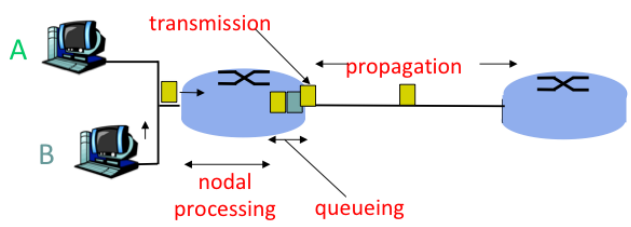
Le componenti del ritardo sono:

- la coda dei pacchetti
- La perdita nasce proprio perchè c'è un ritardo. Il motivo principale per cui i pacchetti vengono persi in internet è che il buffer è pieno, se la memoria è piena questo viene escluso (o ne sostituisce un altro che comunque viene perso)



Il buffer fa il così detto *overflow*, il trabocco.

4 sorgenti del ritardo dei pacchetti



1) tempo necessario per l'elaborazione
2)ritardo di accodamento
3) ritardo di trasmissione
4) ritardo di propagazione (segnale elettromagnetico)
Questi per ogni singolo hop che il pacchetto fa.

Non tutti i pacchetti hanno stessi ritardi.

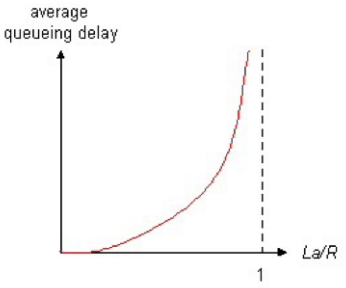
$$d_{nodal} = d_{proc} + d_{queue} + d_{trans} + d_{prop}$$

- d_{proc} = processing delay
  - typically a few microsecs or less
- d_{queue} = queuing delay
  - depends on congestion
- d_{trans} = transmission delay
  - = L/R, significant for low-speed links
- d_{prop} = propagation delay
  - a few microsecs to hundreds of msecs

Il ritardo cresce tutto insieme quando raggiunge una zona critica

- R=link bandwidth (bps)
- L=packet length (bits)
- a=average packet arrival rate

traffic intensity = La/R

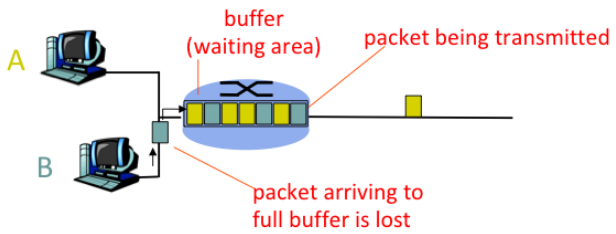


- La/R ~ 0: average queueing delay small
- La/R -> 1: delays become large
- La/R > 1: more "work" arriving than can be serviced, average delay infinite!

Perdita

Una perdita può essere dovuta anche al fatto che ci potrebbero essere degli errori e quindi vengono scartati oppure perchè si sbaglia l'instradamento.

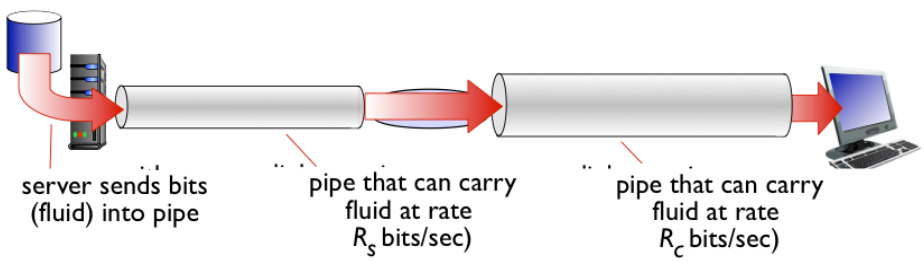
- queue (aka buffer) preceding link in buffer has finite capacity
- packet arriving to full queue dropped (aka lost)
- lost packet may be retransmitted by previous node, by source end system, or not at all



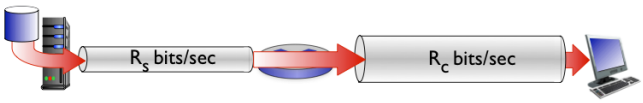
La portata

Quanti bit trasferisco per unità di tempo .

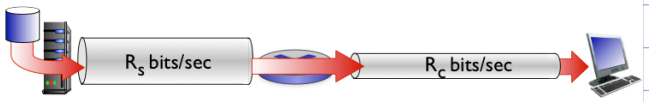
Se abbiamo una sequenza di collegamenti la capacità è quella minima (collo di bottiglia)



- $R_s < R_c$ What is average end-end throughput?



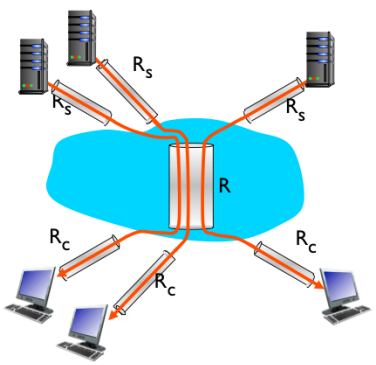
- $R_s > R_c$ What is average end-end throughput?



bottleneck link

link on end-end path that constrains end-end throughput

- per-connection end-end throughput: $min(R_c,R_s,R/10)$
- in practice: R_c or R_s is often bottleneck



10 connections (fairly) share backbone bottleneck link *R* bits/sec