

Reti e Sistemi



Brock

<https://git.brockdev.it>



INTRODUZIONE ALLE RETI

PROTOCOLLO definisce il formato, l'ordine dei messaggi inviati e ricevuti in una rete tra le entità e le azioni intraprese su questi messaggi.

- standard servono per mettere in comunicazione diversi device

INTERNET AS A SERVICE è un'infrastruttura che fornisce servizi a applicazioni distribuite: web, video, streaming...

INVIO DI PACCHETTI DATI un messaggio non viene direttamente inviato, ma:

- viene diviso in pacchetti lunghi L bit
- trasmessi in una rete di banda R

Il tempo necessario al trasferimento è detto **delay di trasferimento** ed è $\frac{L}{R}$

NUCLEO DELLA RETE insieme di router interconnessi che:

- inoltrano (forwarding) i pacchetti in input verso l'output appropriato (livello locale)
- instradano (routing), ovvero determinano le strade che devono essere intraprese dai pacchetti

PACKET SWITCHING spezzettamento dei pacchetti e indirizzamento, attraverso vari router, fino al destinatario.

Esistono diverse tecniche:

- **STORE AND FORWARD** tutto il pacchetto deve essere ricevuto dal router prima che sia possibile l'inizio del trasferimento successivo
- **QUEUING** quando c'è un maggior carico in input rispetto alla velocità di trasferimento (**transmission rate**) i pacchetti vengono messi in coda, in attesa di essere trasmessi.

Se la coda non è in grado di gestire una grande mole di pacchetti (limiti di memoria) vengono persi.

CIRCUIT SWITCHING o reti a commutazione di circuito, sono un'alternativa al packet switching che non si basano sui pacchetti e sulla condivisione di risorse, infatti ogni dispositivo finale riserva risorse, banda e circuiti in cui avviene il trasferimento.

- i packet switching sono migliori per uno scambio di dati orientato alla condivisione di risorse, anche se d'altra parte ci sono i rischi di **congestione** della rete, che vengono comunque gestiti.
- Inoltre è possibile simulare il circuit switching.

RETE DI RETI

l'internet è costituito da:

- connessioni tra hosts e ISPs (internet service provider)
- l'interconnessione tra i vari ISP.

La connessione diretta tra ISP è infattibile, quindi ogni access ISP si connette a degli ISP GLOBALI che sono connessi attraverso IXP (internet exchange point).

A questi ISP globali sono connessi gli ISP regionali, che si fanno carico di una determinata zona geografica.

Attraverso questa rete, i Content Provider Network offrono i servizi agli utenti finali.

PROBLEMI DEL PT SW

Un primo problema sono i vari tipi di ritardo della rete (delay):

- ELABORAZIONE dei pacchetti, controllati con i bit di errore
- ACCODAMENTO dura tipicamente qualche millisecondo e dipende dal livello di congestione *
- TRASMISSIONE dipendente dal metodo di trasferimento
- PROPAGAZIONE dipendente dalla velocità di propagazione del mezzo fisico = $\frac{d}{s}$ [lunghezza] [velocità]

* dato a (average packet arrival rate), l'intensità del traffico è $\frac{L \cdot a}{R}$

THROUGHPUT frequenza (rate) alla quale i bit sono inviati dal mittente al destinatario.

È di due tipi:

- istantaneo rate in un dato punto in un dato tempo
- medio rate lungo un periodo di tempo.

Si calcola trovando il minimo tra le velocità di trasferimento nel collegamento interessato dal trasferimento.

ORGANIZZAZIONE DI INTERNET è diviso in layer, e ogni layer implementa un servizio basandosi sui layer inferiori. È così diviso per suddividere logicamente i compiti della rete e per permettere una più agevole manutenzione e aggiornamento.

I layer sono:

1. applicazione supporto protocolli di rete
2. trasporto trasferimento dei dati
3. rete routing dei dati
4. collegamento trasferimento tra nodi adiacenti
5. fisico supporto fisico

DNS domain name system, serve per associare un host name ad un indirizzo IP, sfruttando gerarchie di nameserver, che sono suddivisi in:

- **root name server** lista di server contenenti correlazioni IP/root domain, sono 13 replicati nel mondo.
- **Top-level-domain TLD** responsabili del top level (com..it,.net,...)
- **DNS di organizzazioni** come amazon,yahoo,unimib ...
- **locali**: cache di recenti collegamenti o name server locali

I domini vengono risolti attraverso delle query.

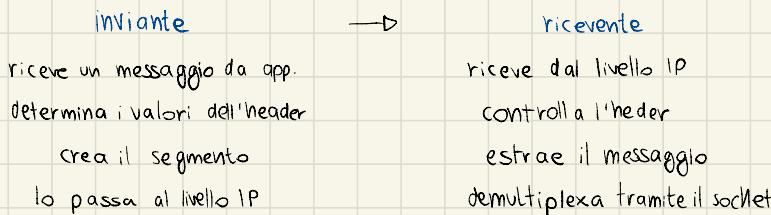
LIVELLO DI TRASPORTO

TRANSPORT LAYER fornisce comunicazione logica tra applicativi, ovvero ci permette di far comunicare processi su macchine diverse come se fossero sulla stessa macchina.

Le azioni intraprese in questo livello sono:

- **Sender** divide i messaggi in segmenti, passa al layer di rete (= comunicazione logica tra host)
- **receiver** riassumba i segmenti, passa al layer applicazione

SOCKET Un'interfaccia che permette di far comunicare direttamente i processi che inviano e ricevono messaggi tramite il socket stesso



MULTIPLEXING gestire dati da più socket, aggiungendo l'header di trasporto.

Lo fa il mittente

DEMULTIPLEXING usare l'header per consegnare i pacchetti ai socket corretti.

Lo fa il destinatario, usando indirizzi IP e porte per indirizzare correttamente i segmenti ai socket.

CONNESSIONE può esserci o non esserci:

- **connectionless** la creazione del socket avviene specificando IP:porta di destinazione
Quando l'host riceve un segmento, lo reindirizza verso la porta specificata
Tutte le comunicazioni arrivano allo stesso socket, indipendentemente dal mittente
- **connection-oriented** i socket sono identificati da srcIP:srcPort e dstIP:dstPort, in questo modo il ricevente può supportare più socket contemporaneamente (ogni socket è connesso con un client).

UDP user datagram protocol, è usato in applicazioni real-time non sensibili alla perdita di dati.

Ha diverse caratteristiche:

- best-effort non garantisce la consegna/consegna ordinata dei messaggi
- no congestion control le congestioni fanno perdere pacchetti
- connectionless non c'è handshake, garantendo setup più semplice e un header di minori dimensioni

Non sono garantiti servizi su ritardi e latenza.

PRINCIPI DI TRASFERIMENTO DATI AFFIDABILE

La complessità di un trasferimento dipende dalle caratteristiche del canale di comunicazione: mittente e destinatario devono comunicare per capire lo stato dell'uno e dell'altro.

Considerando che anche ack e nack sono soggetti ad errori, introduciamo dei protocolli di controllo dell'errore:

- **STOP-AND-WAIT** tutti i segmenti sono numerati, aggiungendo un timer per gestire le perdite (capire quanto tempo farlo durare è difficile, inoltre potrebbe essere troppo lento).
- **SLIDING WINDOW** vengono inviati più pacchetti per volta e si aspettano ack in sequenza, condizione per una trasmissione continua è che la finestra non chiuda prima del primo ack.

Quando un segmento viene perso:

- **Go-Back-N** trasmette un pacchetto e passa a quello successivo dopo ogni ricevimento di ack, se a tempo $S+T$ non arriva ack diamo pacchetto per perso, e il trasferimento ricomincia da dove il pacchetto è stato perso.
- **Protocollo Selective Repeat** uguale a quello sopra, ma viene reinviato solo il pacchetto perso in caso di errore.
- protocolli ibridi secondari

TCP transmission control protocol, usato in applicazioni poco sensibili ai ritardi, ma dove è importante che non ci sia perdita.

Ha diverse caratteristiche:

- point-to-point: un mittente, un destinatario
- affidabile, consegna in-order
- meccanismi di controllo congestione e di flusso
- necessario un handshake iniziale

TCP SEQUENCE NUMBERS

il controllo di sequenza avviene con due numeri:

- sequence number numero del primo byte nel segmento
- acknowledgements numero del prossimo byte atteso dall'altro lato della comunicazione

TCP ROUND TRIP TIME tempo che un pacchetto impiega per "viaggiare", utile per stabilire il valore di timeout che deve essere ponderato.

Per stimarlo si usano due parametri:

- SampleRTT tempo trascorso tra la trasmissione di un segmento e la ricezione dell'ACK
- EstimatedRTT = $(1-\alpha) \cdot \text{EstimatedRTT} + \alpha \cdot \text{SampleRTT}$
- TimeoutInterval = EstimatedRTT + 4 · DevRTT
- DevRTT = $(1-\beta) \cdot \text{DevRTT} + \beta \cdot |\text{SampleRTT} - \text{EstimatedRTT}|$

TCP FAST RETRANSMIT se il mittente riceve 3 ack addizionali per lo stesso data, rinvia un pacchetto unACKed con il più piccolo seq. number

FLOW CONTROL controlla la velocità di trasmissione e la impone di conseguenza, evitando di far andare in overflow il buffer del ricevente

Per farlo, il ricevente restituisce lo spazio libero nel buffer nel campo **rwnd**.

CONNECTION MANAGEMENT prima di iniziare lo scambio di dati i due host effettuano un **three-way handshake**, nel quale concordano come stabilire la connessione e i parametri.

- **CHIUSURA DELLA CONNESSIONE** client e server chiudono la connessione, impostando FINbit=1 e rispondendo con un ACK.

CONTROLLO DI CONGESTIONE la congestione avviene quando più mittenti inviano dati che la rete non riesce a gestire.

Più la rete si satura più si perdono pacchetti e aumenta il delay.

La congestione è gestita da:

- **end-to-end** gestita dai due host (scelta del TCP)
- **network-assisted** gestita da dei feedback del router

Una congestione viene gestita con:

- **AIMD additive increase multiplicative decrease**: in caso di trasferimento di successo l'RTT viene incrementato. In caso di loss, la dimensione viene dimezzata.
- **SLOW START** all'inizio della connessione il cwnd=1 MSS e aumenta esponenzialmente la quantità di dati trasmessa. Arrivato ad un valore prestabilito ssthresh viene impostato a $\frac{1}{2}$ del valore di cwnd.

Implementato in TCP tramite **Tcp Reno**: usa il Fast recovery dopo 3 acks duplicati: imposta cwnd a ssthresh+3 MSS.

CONTROLLO FLOW un mittente potrebbe inviare troppi dati che il destinatario non riesce a gestire.

In particolare:

- cwnd gestisce la congestione
- rwnd gestisce il flow

SISTEMI OPERATIVI: STRUTTURA E SERVIZI

SISTEMA OPERATIVO è un insieme di programmi che gestisce gli elementi fisici del computer. Fornisce una piattaforma di sviluppo per i programmi applicativi che permette loro di condividere ed astrarre le risorse hardware, inoltre agisce da intermediario tra utenti e computer.

STRUTTURA SO un S.O. comprende almeno:

- **KERNEL** si impadronisce dell'hardware, lo gestisce, ed offre ai programmi i servizi per poterlo usare in maniera condivisa e astratta.
- **MIDDLEWARE** servizi di alto livello che semplificano la programmazione di applicazioni.
- **PROGRAMMI DI SISTEMA** offrono ulteriori funzionalità di controllo.

CHIAMATE DI SISTEMA ED API

CHIAMATE DI SISTEMA Funzioni invocabili attraverso le quali il kernel offre i propri servizi.

- per usarle, occorre passare alla Kernel mode, la subroutine `System call interface` legge il numero identificativo della chiamata di sistema, genera una `trap` e, al termine dell'esecuzione, ripassa il controllo alla User mode

I parametri delle chiamate possono essere passati:

- nei registri del processore (rapido ma limitato)
- puntatore a dove sono memorizzati i dati
- push dei parametri sullo stack (lento e macchinoso, ma flessibile)

API application programming interface, sono delle librerie middleware implementate invocando le chiamate di sistema.

- sono fortemente legate con le librerie standard del linguaggio di implementazione.
- sono standardizzate
- sono stabili, al contrario delle chiamate di sistema che variano in base al s.o.
- offrono funzionalità più ad alto livello e più semplici da usare
e.g. Win32 su Windows, POSIX su Unix, Linux e MacOs

CATENA PROGRAMMATIVA un programma sorgente è compilato in un file oggetto che deve poter essere caricato a parlire da qualsiasi locazione di memoria fisica (=file oggetto rilocabile).

- **LINKER** combinano più file oggetto per formare un file eseguibile, anch'esso rilocabile
- **LOADER** caricano in memoria i file eseguibili nel momento in cui devono essere eseguibili e effettuano la rilocazione e il linking delle librerie dinamiche.

LIBRERIE DINAMICHE non vengono linkate compile time, ma quando il programma è caricato.

- possono essere condivise da più programmi
- possono essere modificate senza dover ricompilare i programmi che lo usano (purché non cambi l'ABI).
- snellisce gli eseguibili

MODI DI FUNZIONAMENTO permette all's.o. di proteggersi e proteggere i programmi in esecuzione.

- **USER MODE** protetta, la CPU non può accedere alla memoria del kernel.
- **KERNEL MODE** istruzioni privilegiate.

ABI application binary interface, insieme di convenzioni attraverso le quali il codice binario dell'applicazione si interfaccia con il codice delle librerie dinamiche delle API.

Un eseguibile binario può essere:

- portabile
- runtime portable
- compilato → serve un eseguibile per ogni S.O.

PROGRAMMI DI SISTEMA

Permettono agli utenti di avere un ambiente più conveniente per l'esecuzione dei programmi, il loro sviluppo e la gestione delle risorse del sistema.

- servizi in background, GUI, ambienti di supporto alla programmazione...
- sono implementati attraverso le chiamate di sistema.

INTERPRETE DEI COMANDI permette agli utenti di immettere in maniera testuale le istruzioni che il sistema operativo deve eseguire (Shell).

- possono essere Built-in o programmi di sistema.

IL KERNEL

SOTTOSISTEMI basati sulle categorie dei servizi offerti dal Kernel stesso

I principali sono:

- gestione dei processi e dei thread
- comunicazione tra processi e sincronizzazione
- gestione della memoria
- gestione dell'IO
- file system

ORGANIZZAZIONE è un programma complesso e di dimensioni elevate, che deve operare rapidamente e il cui malfunzionamento può provocare crash di sistema.

Esistono diversi modelli per progettarlo:

- **MONOLITICA** è un singolo file binario statico.
 - ⊕ elevate prestazioni
 - ⊖ complesso, fragile e non modulare
- **A STRATI** diviso in un insieme di livelli, lo strato più basso interagisce con l'hardware, e ogni strato interagisce con il successivo
 - ⊕ indipendenza degli strati, astrazione delle caratteristiche della macchina.
 - ⊖ introduce overhead, come suddividere in strati?
- **MICROKERNEL** sposta quanti più servizi possibili fuori dal kernel, ha quindi dimensioni ridotte.
 - ⊕ estensibile, affidabile e meno sensibile ai crash
 - ⊖ Overhead, una richiesta deve transitare diversi "strati"

- **A MODULI** strutturato in componenti: dinamicamente caricabili (moduli), che parlano tra di loro attraverso interfacce: il kernel carica dinamicamente i moduli, solo quando offre il servizio implementato dal modulo, e lo scarica quando non serve più.
 - ↪ tra strati e microkernel, con minore overhead e isolamento
- **IBRIDI** combinano diversi approcci allo scopo di ottenere sistemi indirizzati a una specifica prestazione.

POLITICA dice quando una certa operazione viene effettuata

- impattano profondamente sulle caratteristiche percepite del sistema di elaborazione

MECCANISMO spiega come una certa operazione è effettuata

- Sono più stabili delle politiche, che spesso cambiano in funzione delle caratteristiche che il sistema di elaborazione deve avere

PROCESSI E THREAD

- PROCESSO** è un'entità attiva astratta definita dal sistema operativo allo scopo di eseguire un programma
- è un'entità attiva, al contrario, un programma è un'entità passiva
 - uno stesso programma può dar vita a diversi processi.

Per mantenere impegnata la CPU il maggior tempo possibile e dare l'illusione che i processi evolvono contemporaneamente:

- **MULTIPROGRAMMAZIONE** impedire che un programma che non è in condizione di proseguire l'esecuzione mantenga la CPU
- **MULTITASKING** far sì che un programma interattivo reagisca agli input utente in un tempo accettabile

MULTIPROGRAMMAZIONE

- il S.O. mantiene in memoria i processi da eseguire
- se la CPU non è impegnata, viene assegnata ad un processo, quando un processo non può più evolvere la CPU viene assegnata ad un altro processo.
 - richiede che tutte le immagini di tutti i processi siano in memoria: lo **swapping** viene usato per spostare dentro/fuori dalla memoria le immagini (eq. **memoria virtuale**).

MULTITASKING

estensione della multiprogrammazione per sistemi interattivi: la CPU viene sottratta periodicamente al programma in esecuzione ed assegnata ad un altro programma

- tutti i programmi progrediscono in maniera continuativa nella propria esecuzione, in modo che nessun programma monopolizzi la CPU.

CREAZIONE DEI PROCESSI

solo un processo può creare un altro processo, all'avvio il sistema crea dei processi primordiali che generano gli altri.

- un processo padre può creare altri processi figli (albero di processi)

TERMINAZIONE DI PROCESSI

i processi richiedono esplicitamente la propria terminazione al sistema operativo

- un padre può attendere o meno la terminazione di un figlio, o forzare la terminazione di un figlio.
- alcuni S.O. non consentono ai processi figli di esistere dopo la terminazione del padre (terminazione a cascata)
 - se un processo termina, ma un padre non lo sta aspettando, il processo è detto **zombie**.
 - se un processo termina prima di un suo figlio, il figlio è **orfano**.

STRUTTURA DI UN PROCESSO

processi distinti hanno immagini distinte:

- stato dei registri del processore
- Stato della regione di memoria centrale usata (immagine).
- Stato del processo stesso
- le risorse dell'S.O. in uso

L'immagine è formata da:

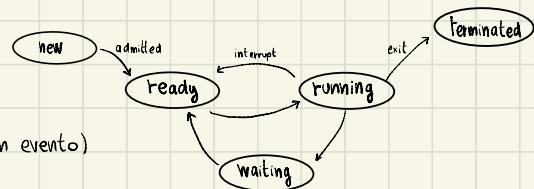
- text section, codice del programma
- data section, contenente le variabili globali
- stack delle chiamate
- Lo heap

} hanno dimensione costante
} variano durante la vita del processo

STATO DI UN PROCESSO

durante l'evoluzione può trovarsi in:

- NEW creato, ma non ancora ammesso all'esecuzione
- READY il processo è pronto (in attesa della CPU)
- RUNNING in esecuzione
- WAITING il processo non può essere eseguito (in attesa di un evento)
- TERMINATED il processo ha terminato l'esecuzione



PCB process control block, è la struttura dati del kernel che contiene tutte le informazioni relative ad un processo:

- process state
- process number
- program counter
- registers
- informazioni relative alla gestione della memoria
- informazioni sull'I/O
- informazioni di scheduling
- informazioni di accounting

SCHEDULING DEI PROCESSI

sceglie il prossimo processo da eseguire tra quelli in stato ready.

- READY QUEUE processi residenti in memoria e in stato ready
- WAIT QUEUES code per i processi che sono residenti in memoria e in stato wait; una coda diversa per ciascun diverso tipo di evento di attesa

I processi migrano da una coda all'altra a seconda dello stato del processo stesso.

COMMUTAZIONE DI CONTESTO context switching, viene effettuata dal dispatcher, che salva il contesto del processo da interrompere nel suo PCB, caricare il contesto del processo da eseguire dal suo PCB.

- il tempo necessario è overhead

COMUNICAZIONE INTERPROCESSO i processi possono essere indipendenti o cooperare.

- coopera se il suo comportamento influenza o è influenzato dal comportamento di uno o più altri processi
- per cooperare, il sistema operativo deve mettere a disposizione primitive di comunicazione interprocesso (IPC)

MEMORIA CONDIVISA viene stabilita una zona di memoria condivisa tra i processi che intendono comunicare.

- è controllata dai processi che comunicano, non dal sistema operativo (devono sincronizzarsi)

MESSAGE PASSING permettono ai processi sia di comunicare che di sincronizzarsi, attraverso la mediazione del sistema operativo

- devono stabilire un link di comunicazione tra di loro e scambiate usando send e receive

PIPE canali di comunicazione tra i processi:

- Named pipes bidirezionali
- Unix half-duplex
- Windows full-duplex

MULTITHREADING più istruzioni possono eseguire conconemente, e quindi il processo può avere più percorsi (**thread**) di esecuzione concorrenti

- i thread di uno stesso processo condividono la memoria globale, la memoria contenente il codice e le risorse ottenute dal sistema operativo. Ogni thread ha un proprio stack

LIBRERIE DI THREAD in Posix pthreads, in Windows i threads

IMPLEMENTAZIONE DEI THREADS

- A LIVELLO UTENTE i thread disponibili nello spazio utente dei processi, sono quelli offerti dalle librerie di thread ai processi
- A LIVELLO DEL KERNEL i thread implementati nativamente dal kernel, sono usati per strutturare il kernel stesso in maniera concorrente

MODELLI MULTITHREADING i thread a livello del kernel vengono utilizzati dalle librerie di thread per implementare i thread a livello utente di un certo processo

- **MOLTI-A-UNO** tutti i thread a livello utente sono implementati su un solo thread a livello kernel

- Usabile su ogni SO

- una chiamata bloccante blocca tutto, non sfrutta i core

- **UNO-A-UNO** ogni thread a livello utente è implementato su un singolo, distinto thread a livello del kernel

- permette un maggior grado di concorrenza, sfrutta il multicore

- minore performance, stress del kernel

- **MOLTI-A-MOLTI** i thread a livello utente di un certo processo sono implementati su un insieme di thread a livello del kernel, con un associazione dinamica stabilita da uno scheduler

- modello a due livelli: mapping uno-a-uno

- Complesso da implementare

THREAD CONTROL BLOCKS struttura analoga al PCB

LIGHTWEIGHT PROCESS è l'interfaccia offerta dal kernel alle librerie dei thread per usare i thread del kernel

- oggetto astratto associato staticamente ad un thread

ATTIVAZIONE DELLO SCHEDULER è un modello di collaborazione tra libreria dei thread e kernel

- il kernel comunica alla libreria attraverso upcall

- in risposta ad una upcall, la libreria può:

- aggiornare le proprie strutture dati

- decidere di effettuare operazioni di scheduling

SCHEDULING DELLA CPU

SCHEDULER a breve termine, seleziona un processo tra quelli nella ready queue ed alloca un core libero ad esso.

Tali assegnamenti possono essere effettuati quando un processo cambia stato:

- running \rightarrow waiting
- running \rightarrow ready
- waiting \rightarrow ready
- termina

TIPI DI RIASSEGNAZIONE

- NONPREEMPTIVE un core è sempre e solo liberato da un processo che volontariamente rinuncia ad esso
- PREEMPTIVE un core può essere liberato forzatamente

DISPATCHER passa effettivamente il controllo della CPU al processo scelto dallo scheduler, genera una Latenza.

BURST

- della CPU sequenza di operazioni di CPU
- dell'I/O attesa completamento operazione di I/O

ALGORITMI DI SCHEDULING

IN ORDINE DI ARRIVO FCFS, la CPU viene assegnata al primo processo che lo richiede.

- implementazione molto semplice
- "effetto convoglio"

PER BREVITÀ

SJF La CPU viene assegnata al processo che ha il successivo CPU burst più breve

- minimizza il tempo di attesa medio
- non prevedibile (come si calcola il burst)

SRTF utilizza la prelazione nel caso in cui i processi non arrivano tutti nello stesso istante.

CIRCOLARE RR ogni processo ottiene una piccola quantità fissata di tempo di CPU (quanto di tempo), trascorso il quale viene interrotto in favore di un altro processo, con politica FIFO.

- ha un tempo di completamento medio più alto, ma un tempo di risposta medio più basso

CON PRIORITÀ ad ogni processo è associato un numero intero che indica la sua priorità, viene eseguito quello con priorità più alta.

- problema di **STARVATION** un processo a priorità troppo bassa potrebbe non essere mai schedulato
Risolto con l'**AGING** aumento di priorità all'aumentare del tempo di permanenza

CODE MULTILIVELLO code separate per ogni priorità

- **BASATA SUL PROCESSO** la priorità dipende dal tipo di processo
- **CON RETROAZIONE** la priorità può variare dinamicamente

ESERCIZIARIO - Sistemi operativi

1 P Burst

P₁ 21

P₂ 13

P₃ 2

P₄ 6

P₅ 4

Algoritmo: FCFS

- Tempo medio di attesa:

$$(0 + 21 + 34 + 36 + 42) / 5 = 26,6$$

- Tempo medio di completamento:

$$(21 + 34 + 36 + 42 + 46) / 5 = 35,8$$

2 P Burst

Algoritmo: SJF

P₁ 21

P₂ 13

P₃ 2

P₄ 6

P₅ 4

- Tempo medio di attesa:

$$(25 + 12 + 0 + 6 + 2) / 5 = 9$$

- Tempo medio di completamento:

$$(46 + 25 + 2 + 12 + 6) / 5 = 18,2$$

3 P Burst

Algoritmo: RR con quanto = 6

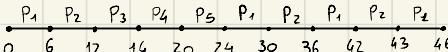
P₁ 21

P₂ 13

P₃ 2

P₄ 6

P₅ 4



- Tempo medio di attesa:

$$P_1: 46 - 21 = 25$$

$$P_2: 43 - 13 = 30$$

$$P_3: 14 - 2 = 12$$

$$P_4: 20 - 6 = 14$$

$$P_5: 24 - 4 = 20$$

- Tempo medio di completamento:

$$P_1: 46 - 0 = 46$$

$$P_2: 43 - 0 = 43$$

$$P_3: 14 - 0 = 14$$

$$P_4: 20 - 0 = 20$$

$$P_5: 24 - 0 = 24$$

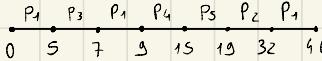
$$(25 + 30 + 12 + 14 + 20) / 5 = 20,2$$

$$(46 + 43 + 14 + 20 + 24) / 5 = 29,4$$

4

P	Burst	E
P ₁	21	0
P ₂	13	10
P ₃	2	5
P ₄	6	9
P ₅	4	12

Algoritmo: SRTF



- Tempo medio di attesa:

$$P_1: 46 - 21 = 25$$

$$P_2: 32 - 23 = 9$$

$$P_3: 7 - 7 = 0$$

$$P_4: 15 - 15 = 0$$

$$P_5: 19 - 16 = 3$$

- Tempo medio di completamento:

$$P_1: 46 - 0 = 46$$

$$P_2: 32 - 10 = 22$$

$$P_3: 7 - 5 = 2$$

$$P_4: 15 - 9 = 6$$

$$P_5: 19 - 12 = 7$$

$$(25 + 9 + 0 + 0 + 3) / 5 = 7,4$$

$$(46 + 22 + 2 + 6 + 7) / 5 = 16,6$$

5

P	Burst	P
P ₁	21	4
P ₂	13	5
P ₃	2	3
P ₄	6	2
P ₅	4	1

Algoritmo: PR

- Tempo medio di attesa:

$$(0 + 4 + 10 + 12 + 33) / 5 = 11,8$$

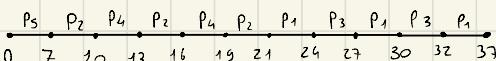
- Tempo medio di completamento:

$$(33 + 46 + 12 + 10 + 4) / 5 = 21$$

6

P	Burst	P
P ₁	11	3
P ₂	8	2
P ₃	5	3
P ₄	6	2
P ₅	7	1

Algoritmo: PR+RR con quanto=3



- Tempo medio di attesa:

$$P_1: 37 - 11 = 26$$

$$P_2: 21 - 8 = 13$$

$$P_3: 32 - 5 = 27$$

$$P_4: 19 - 6 = 13$$

$$P_5: 7 - 7 = 0$$

- Tempo medio di completamento:

$$P_1: 37$$

$$P_2: 21$$

$$P_3: 32$$

$$P_4: 19$$

$$P_5: 7$$

$$(26 + 13 + 27 + 13 + 0) / 5 = 15,8$$

$$(37 + 21 + 32 + 19 + 7) / 5 = 23,2$$

ESERCIZIARIO - Reti 1

1 $O = 100 \text{ KB}$

$S = 500 \text{ byte}$

$\text{RTT} = 100 \text{ ms}$

$W = ?$

$R = 1 \text{ Mbps}$

$$\text{Latenza minima} = \text{RTT} \cdot O = \frac{100 \text{ KB}}{R} = \frac{8 \cdot 10^5 \text{ b}}{1 \text{ Mbps}} = 0.8 \text{ s} + 0.2 \text{ s} = 1 \text{ s}$$

$$W = \frac{\text{RTT} \cdot R}{L} + 1 = \frac{100 \text{ ms} \cdot 1 \text{ Mbps}}{500 \text{ byte}} + 1 = \frac{1 \cdot 10^{-2} \cdot 1 \text{ bps} \cdot 10^6}{4 \cdot 10^3} + 1 = \frac{100}{4} + 1 = 26$$

2 $R = 1 \text{ Gbps}$

$\text{RTT} = 30 \text{ ms}$

$L = 1000 \text{ byte}$

nº pacchetti = ?

$$N = \frac{\text{RTT} \cdot R}{L} + 1 = \frac{30 \text{ ms} \cdot 1 \text{ Gbps}}{1000 \text{ byte}} + 1 = \frac{3 \cdot 10^{-2} \cdot 1 \text{ bps} \cdot 10^9}{8000 \text{ b}} + 1 = \frac{3 \cdot 10^7}{8 \cdot 10^3} + 1 =$$

$$= 3750 + 1 = 3751$$

3 carattere = 20

$$\text{ack} = \text{seq} + \text{dati} = 51 + 20 = 71$$

$\text{seq} = 51$

$\text{ack} = 60$

4 $T = 50 \text{ kBps}$

$\text{RTT} = 90 \text{ ms}$

$L = 500 \text{ byte}$

nº pacchetti = ?

$$T = \frac{3 \cdot N \cdot L}{4 \cdot \text{RTT}} \rightsquigarrow n = \frac{4 \cdot \text{RTT} \cdot T}{3L} = \frac{4 \cdot 90 \text{ ms} \cdot 50 \text{ kBps}}{3 \cdot 500 \text{ b}} =$$

$$= \frac{4 \cdot 9 \cdot 10^{-2} \cdot 4 \cdot 10^6 \text{ bps}}{8 \cdot 15 \cdot 10^3 \text{ b}} = \frac{4 \cdot 9 \cdot 4 \cdot 10}{8 \cdot 15} = 12$$

5 Finestra scorrevole

$W = ?$

tras. continua

$R = 12 \text{ Mbps}$

$\text{RTT} = 80 \text{ ms}$

$L = 1000 \text{ byte}$

$$W = \frac{\text{RTT} \cdot R}{L} + 2^* = \frac{80 \text{ ms} \cdot 12 \text{ Mbps}}{1000 \text{ byte}} + 2 = \frac{8 \cdot 10^{-2} \cdot 12 \cdot 10^6 \text{ bps}}{18000 \text{ b}} + 2 =$$

$$= \frac{8 \cdot 10^{-2} \cdot 12 \cdot 10^6 \text{ bps}}{18000 \text{ b}} + 2 = \frac{12 \cdot 10^4}{10^3} + 1 = 120 + 2 = 122$$

* = si aggiunge +2 e non +1 perché nella traccia è specificato "segmenti trasmessi [+1] e i messaggi di riscontro [+1]".

6 TCP Reno

SSThresh = 12 [partenza lenta]

Segmenti = 20

Max. Cwnd = ?

Seg. 18 è perso

	LOOP	CWND	SST	PACCHETTI
	1	1	12	1
	2	2	12	2-3
	3	4	12	4-5-6-7
	4	8	12	8-9-10-11-12-13-14-15
	5	16	12	16-17-[18]-19...

7 dim = 10000 byte

Ttrans = ?

SSThresh = 3

R = 820 Kbps

RTT = 40 ms

MSS = 2000 byte

$$TF1 = \frac{MSS}{R} + RTT = \frac{2000B}{820\text{Kbps}} + 4 \cdot 10^{-2}\text{s} = \frac{16000}{820000} + 4 \cdot 10^{-2} = 0,019 + 0,04 = 0,059$$

$$TF2 = TF1 = 0,059$$

$$TF3 = \frac{2 \cdot MSS}{R} = \frac{2 \cdot 16000\text{b}}{820000\text{bps}} = 0,038$$

$$T_{trans} = 59\text{ms} + 59\text{ms} + 38\text{ms} + 40\text{ms} = 196\text{ms}$$

8 R = 730 Kbps

RTT = 120 ms

MSS = 5000 bit

congestion avoidance

Cwnd alla perdita

C₁, C₂ connessioni

$$SSThreshold = \frac{R \cdot RTT}{L} = \frac{730\text{Kbps} \cdot 120\text{ms}}{5000\text{b}} = \frac{730000\text{bps} \cdot 12 \cdot 10^{-3}\text{ms}}{5000\text{b}} = 17$$

CWND1 CWND2 CWND1+CWND2

4 6 10

5 7 12

6 8 14

7 9 16

$\frac{1}{2} + 3$ { 8 10 [18] }
 { 7 8 15 }

8 9 17

9 10 [19]

8 8 16

9 9 [18]

9 Congestion avoidance

$$R = 3 \text{ Mbps}$$

MSS = 700 byte

RTT = 80 ms

a) Ampiezza massima

b) ampiezza media

c) velocità di trasmissione

$$\text{a)} \frac{R \cdot RTT}{L} = \frac{3 \text{ Mbps} \cdot 80 \text{ ms}}{700 \text{ byte}} = \frac{3000000 \cdot 10^{-2}}{700} = 42,85 \approx 42$$

$$\text{b)} \min = \frac{\max + 3}{2} = \frac{42 + 24}{2} = 33$$

$$\overline{Amp} = \frac{\max + \min}{2} = \frac{42 + 24}{2} = \frac{66}{2} = 33$$

$$\begin{aligned} 33 \cdot 700 \text{ b} &= \frac{33 \cdot 700}{10^{-2}} = 2310000 \text{ b} = \\ &= 2,31 \text{ Mbps} \end{aligned}$$

10

dim = 4122 byte

W = 2500 byte

MSS = 500 byte

R = 100 KBps =

$$= 100 \cdot 10^3 \text{ Bps}$$

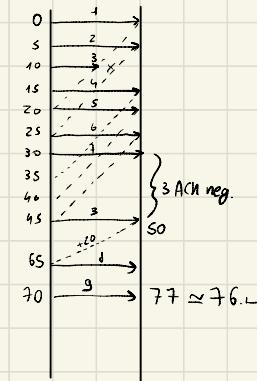
$$= 1 \cdot 10^5 \text{ Bps}$$

RTT = 15 ms

temp tot. = ?

$$T_{\text{prog}} = \frac{MSS}{R} = \frac{500 \text{ byte}}{100 \text{ KBps}} = 0,005 \text{ s}$$

$$\text{n. pack} = \frac{4122}{500} \approx 9 \text{ [per eccesso]}$$



ESERCIZIARIO - Reti 2

1 $V = 250 \text{ mln m/s}$

$$R = 25 \text{ kbps}$$

$$L = 100 \text{ bit}$$

$$d \text{ alla quale } d_{\text{prop}} = d_{\text{trans}}$$

$$d_{\text{trans}} = \frac{L}{R} = \frac{100 \text{ bit}}{25 \text{ kbps}} = \frac{100 \text{ bit}}{25 \cdot 10^3 \text{ bps}} = 0,004$$

$$0,004 \text{ s} = \frac{d}{250.000.000 \frac{\text{m}}{\text{s}}}$$

$$d = 0,004 \cdot 250.000.000 \frac{\text{m}}{\text{s}} = 1.000.000 \text{ m} = 1.000 \text{ km}$$

2 $L = 1500 \text{ byte}$

$$d_{\text{prop}} = 60 \text{ us}$$

$$R = ? \text{ per } d_{\text{prop}} = d_{\text{trans}}$$

$$60 \text{ us} = \frac{1500 \text{ byte}}{R}$$

$$\frac{1}{60 \text{ us}} = \frac{R}{1500 \text{ byte}}$$

$$R = \frac{1500 \cdot 8 \text{ b}}{60 \cdot 10^{-6} \text{ s}} = 200 \text{ Mbps}$$

3 $V = 300 \text{ mln m/s}$

$$R = 1 \text{ Gbps}$$

$$L = 500 \text{ byte}$$

$$d = ? \quad d_{\text{prop}} = d_{\text{trans}}$$

$$d_{\text{trans}} = \frac{L}{R} = \frac{500 \text{ byte}}{1 \text{ Gbps}} =$$

$$= \frac{500 \cdot 8 \text{ b}}{1 \cdot 10^9 \text{ bps}} = 0,000004 \text{ s}$$

$$0,000004 \text{ s} = \frac{d}{300 \cdot 10^6 \frac{\text{m}}{\text{s}}}$$

$$d = 0,000004 \cdot 300 \cdot 10^6 \frac{\text{m}}{\text{s}} = 1200 \text{ m}$$

4 $d_{\text{trans}} + d_{\text{prop}} = 13 \text{ us}$

$$L = 1500 \text{ byte}$$

$$d = 900 \text{ m}$$

$$V = 300.000 \text{ km/s}$$

$$R = ?$$

$$\frac{L}{R} + \frac{d}{V} = 13 \text{ us}$$

$$\frac{1500 \text{ byte}}{R} = 13 \cdot 10^{-6} \text{ s} - 3 \cdot 10^{-6} \text{ s}$$

$$\frac{R}{1500 \text{ byte}} = \frac{1}{10 \cdot 10^{-6} \text{ s}}$$

$$R = \frac{1500 \cdot 8 \text{ b}}{10 \cdot 10^{-6} \text{ s}} = 1,2 \cdot 10^9 \text{ bps}$$

5 $V = 300 \text{ m/s}$

$R = 12 \text{ Mbps}$

$L = 1500 \text{ byte}$

$d = ? \text{ in cui } d_{\text{prop}} = d_{\text{trans}}$

$$d_{\text{trans}} = \frac{L}{R} = \frac{1500 \text{ byte}}{12 \text{ Mbps}} = \\ = \frac{1500 \cdot 8 \text{ b}}{12 \cdot 10^6 \text{ bps}} = 0,001 \text{ s}$$

$0,001 \text{ s} = \frac{d}{300 \cdot 10^3 \frac{\text{m}}{\text{s}}}$

$$d = 0,001 \cdot 300 \cdot 10^3 \frac{\text{m}}{\text{s}} = 0,3 \cdot 10^6 \text{ m} = \\ = 300000 \text{ m} = \\ = 300 \text{ Km}$$

6 $R = 100 \text{ Mbps}$

$L = 1500 \text{ byte}$

$\text{n° pacchetti per secondo} = ? = a$

$d_{\text{queue}} > d_{\text{elab}} + d_{\text{prop}} + d_{\text{trans}}$

$\frac{L}{R} \cdot a \approx 1$

$a = \frac{R}{L} = \frac{100 \cdot 10^6 \text{ bps}}{1500 \cdot 8 \text{ b}} = 8333, \overline{3}$

7 Store and forward

$L = 7,5 \text{ Mbit}$

$4 \text{ router e 5 link (Q)}$

$R = 1,5 \text{ Mbps}$

$t_{\text{trasmissione}} = ?$

$t = \frac{L}{R} \cdot Q = \frac{7,5 \text{ Mb}}{1,5 \text{ Mbps}} \cdot 5 =$

$= 5 \cdot 5 = 25 \text{ s}$

8 Store and forward

$L = 7,5 \text{ Mbit}$

$4 \text{ router e 5 link}$

$R = 1,5 \text{ Mbps}$

$\text{Frammento} = 5000 \text{ da } 1500 \text{ b}$

$d_h = \frac{L}{R} + (Q - 1) \frac{L}{R} =$

$= \frac{7,5 \text{ Mbps}}{1,5 \text{ Mbps}} + (4) \frac{1500 \text{ b}}{1,5 \cdot 10^6 \text{ bps}} =$

$= 5 + 4 \cdot 0,001 = 5 + 0,004 \approx 5$

ESERCIZIARIO - Reti 3

1) $W = ?$

trasmissione continua

$V = 640 \text{ Kbps}$

$RTT = 40 \text{ ms}$

$\ell = 1000 \text{ b}$

$$\frac{N \cdot \frac{L}{R}}{RTT + \frac{L}{R}} = 1$$

$$N = \frac{RTT + \frac{L}{R}}{\frac{L}{R}}$$

$$N = \frac{40 \cdot 10^{-3} \text{ s} + 0,0125}{0,0125} = 4,2 \approx 6$$

$$\frac{L}{R} = \frac{8000 \text{ b}}{640 \cdot 10^3 \text{ bps}} = 0,0125$$

1.1 ℓ riscontro = 1000 byte

$$\frac{N \cdot \frac{L}{R}}{RTT + 2 \frac{L}{R}} = 1 \quad [\text{si moltiplica per due perché si considera anche ACK}]$$

$$N = \frac{RTT + 2 \frac{L}{R}}{\frac{L}{R}}$$

$$N = \frac{40 \cdot 10^{-3} \text{ s} + 0,025}{0,0125} = 5,2 \approx 6$$

2) $L = 14000 \text{ byte}$

$t_{trans} = ?$

$SSThresh = 4$

$R = 1 \text{ Mbps}$

$RTT = 30 \text{ ms}$

$MSS = 1000 \text{ byte}$

cwnd	dati inviati	tf
1	1000	38 ms
2	3000	38 ms
4	7000	38 ms
5	12000	40 ms
6	14000	16 ms

$$TF1 = \frac{MSS}{R} + RTT = \frac{8000 \text{ b}}{10^6 \text{ bps}} + 30 \text{ ms} = 38 \text{ ms}$$

$$38 \text{ ms} < \frac{cwnd \cdot mss}{R} < 1 \cdot 0,008 \text{ no}$$

$$TF2 = 38 \text{ ms}$$

$$38 \text{ ms} < 0,016 \text{ no}$$

$$TF3 = 38 \text{ ms}$$

$$38 \text{ ms} < 0,032 \text{ no}$$

$$TF4 = 5 \cdot \frac{MSS}{R} = 40 \text{ ms}$$

$$38 \text{ ms} < 0,040 \text{ si}$$

$$TFS = 2 \cdot \frac{mss}{R} = 16 \text{ ms}$$

$$t_{trans} = 38 + 38 + 38 + 40 + 16 + 30 = 180 \text{ ms}$$

3) $L = 6500 \text{ byte}$

$W = 4000 \text{ byte}$

$MSS = 1000 \text{ byte}$

$R = 800 \text{ Kbps}$

$RTT = 30 \text{ ms}$

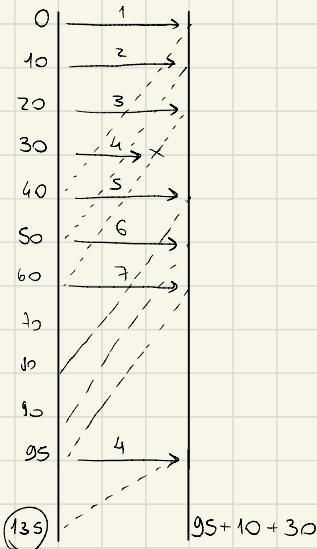
4° segmento perduto

t totale

TCP con finestra scorrevole

$$\text{n. invii} = \frac{6500}{1000} \approx 7$$

$$t_{\text{prop}} = \frac{MSS}{R} = \frac{8000}{800 \cdot 10^3} = 0,01$$



4) $\text{seq} = 112$

$$\text{ack} = \text{seq} + L = 112 + 30 = 142$$

$\text{ack} = 115$

$$\text{seq} = \text{ack} - 1 = 115$$

$L = 30 \text{ byte}$

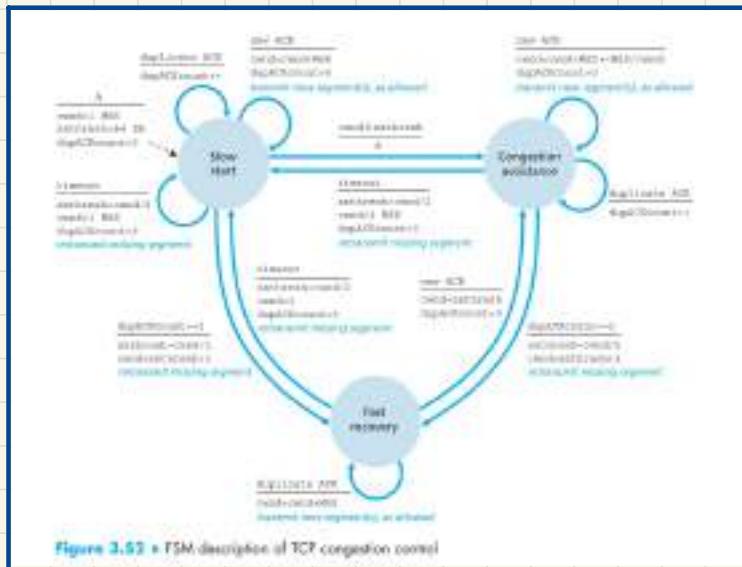
RISOLUZIONE ESERCIZI TCP

perché tipo sono troppi e se non schematizzo non arrivo a 21 anni: ma muoio prima di ansia e tristezza

TCP RENO

Si deve usare l'automa:

- congestion avoidance: aumenti di 1, quando si perde il pacchetto: $ssthreshold = \frac{cwnd}{2}$
- slow start: raddoppia ogni volta: $cwnd = \frac{cwnd}{2}$



- In sintesi:

La cwnd viene sempre duplicata fino a che non si verificano 3 ack o scade il fo.

$$cwnd = \frac{cwnd}{2} + 3 \text{ e viene } ssthresh = cwnd \text{ se } cwnd > ssthresh$$

$$\begin{cases} \text{timeout} \\ cwnd \text{ diventa } \frac{1}{2} \end{cases}$$

TCP CON FINESTRA SCORREVOLE

Fai lo schema con la comunicazione, segnando i pacchetti mancanti.

Quelli persi vengono rinvati, dopo 3 ack consecutivi.

Il tempo finale è $t_{\text{propagazione}} + \frac{\text{RTT}}{2} + \text{ultimo invio}$.

FORMULARIO

Trasmissione e Pacchetti

$$d_{trans} = \frac{L}{R} \quad d_{prop} = \frac{d}{v}$$

$$\text{intensità del traffico} = \frac{L \cdot a}{R} \approx 1 \text{ [max]}$$

$$\text{numero di pacchetti} = \frac{L}{\ell}$$

L : lunghezza del pacchetto

R : banda, transmission rate

d : distanza tra due nodi

v : velocità di propagazione

a : tasso medio di arrivo dei pacchetti

ℓ : dimensione del singolo pacchetto

Store and Forward

$$d_n (\text{no frammentazione}) = Q \cdot \frac{L}{R}$$

Q : numero di link

ℓ : grandezza singolo pacchetto

$$d_n (\text{frammentazione}) = \frac{L}{R} + (Q-1) \frac{\ell}{R}$$

Comunicazione e canali

$$\text{utilizzo canale} = \frac{N \cdot \frac{L}{R}}{RTT + \frac{L}{R}} \approx 1 \text{ (continuo)}$$

N : dimensione della finestra

RTT : round trip time

O : dimensione oggetto

MSS : massima dimensione segmento

$$\text{latenza} = 2 \cdot RTT + \frac{O}{R}$$

$$\text{dim. media finestra} = 1 + \frac{R \cdot RTT}{MSS}$$

$$\text{tasso perdita pacchetti} = \frac{1}{\frac{3}{8} \cdot N^2}$$

Congestione della finestra

$$\text{massima} = \left\lceil \frac{RTT \cdot R_{max}}{MSS} \right\rceil$$

$$\text{minima} = \left\lceil \frac{\max}{2} \right\rceil + 3$$

$$\text{media} = \left\lceil \frac{\max + \min}{2} \right\rceil$$

$$\text{tempo invio (slow start)} = \frac{L}{R} + RTT$$

Secondo compitino

GESTIONE DELLA MEMORIA

PREREQUISITI i processi usano la memoria centrale e i registri.

Le operazioni sulla memoria richiedono molti cicli di clock, causando uno **stall** del processore.

Per aumentare l'efficienza degli accessi, vengono usati diversi livelli di memorie cache.

Si pone il problema di allocare porzioni di memoria per i diversi processi.

ALLOCAZIONE CONTIGUA La più semplice, la memoria centrale è partizionata in due zone,

- Sistema operativo
- processi utente

Ogni processo utente occupa un'area contigua di memoria, e qui viene caricata la sua immagine.

PROTEZIONE CON REGISTRI BASE E LIMITE metodo di protezione

utilizzabile con l'allocazione contigua

Il processo possiede due registri:

- **base** contiene il più piccolo indirizzo della memoria fisica che il processo ha il permesso di accedere
- **limite** determina la dimensione dell'intervallo ammesso.

Possono essere impostati solo in modalità sistema, mentre:

- in mod utente il processore proibisce tutte le operazioni fuori dall'intervallo, generando nel caso un'eccezione
- in mod sistema può accedere a tutta la memoria.

ASSOCIAZIONE DEGLI INDIRIZZI il s.o. carica uno stesso programma, in momenti diversi, in diverse aree di memoria non contigue

Ci sono diverse rappresentazioni:

- si possono non usare (codice sorgente)
 - indirizzi rilocabili (file oggetto)
 - indirizzi assoluti (linker e loader)
- } operazioni di associazioni
binding

TIPI DI BINDING può essere fatto:

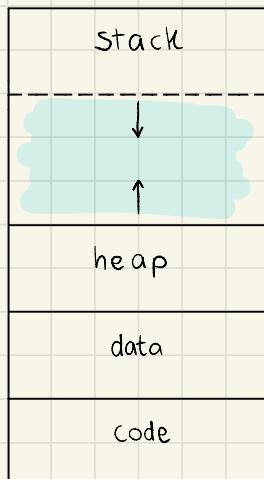
- Si conosce l'indirizzo: **in compilazione** il compilatore effettua il binding e genera codice assoluto, cambia indirizzo \Rightarrow ricompilazione
- Non si conosce l'indirizzo e non viene spostato in memoria: **in caricamento** il compilatore genera codice rilocabile e il loader effettua il binding al momento dell'esecuzione
- Non si conosce l'indirizzo e può essere spostato in memoria: **in esecuzione** viene richiesto il supporto hardware

INDIRIZZI LOGICI E FISICI

- indirizzi logici indirizzi generati dalla CPU
 - indirizzi fisici indirizzi che arrivano alla memoria centrale
- } differiscono solo in fase di esecuzione

I logici vengono tradotti in fisici dalla **memory management unit**, che interviene solo in modalità utente, infatti in mod kernel sono già fisici.

SPAZIO DI INDIRIZZAMENTO VIRTUALE è lo spazio di indirizzi logici di un processo, che si estende da un indirizzo logico (0) ad uno massimo, solitamente organizzato in:



MMU CON REGISTRO DI RILOCAZIONE

è una variazione dello schema con registri base e limite, in cui il registro base è il registro di rilocazione.

- indirizzo fisico = indirizzo logico + registro di rilocazione
- nel registro base viene caricato l'indirizzo più basso dell'area contigua assegnata
- nel registro limite la dimensione della memoria → se viene superato, viene generata un'eccezione e di solito viene interrotto il processo

ALLOCAZIONE CONTIGUA A PARTIZIONI VARIABILI

Ogni processo ottiene una partizione di memoria distinta, con uno schema a partizioni variabili: partizione di dimensione pari alla memoria necessaria al processo.

- buco regione di memoria libera contigua. Buchi adiacenti sono uniti.

Il SO mantiene una lista dei buchi disponibili sparsi nella memoria centrale.

Alla creazione di un processo, l'SO deve decidere quale buco usare, secondo la strategia:

- First-fit sceglie il primo sufficientemente grande
 - Best-fit sceglie il buco più piccolo
 - Worst-fit sceglie il buco più grande
- } migliori in tempo ed efficienza

FRAMMENTAZIONE

- ESTERNA La memoria libera è sufficiente, ma è sparsa tra buchi non contigui troppo piccoli

- INTERNA se la memoria allocata è maggiore di quella necessaria, una partizione può contenere memoria inutilizzata.

Lo spazio inutilizzabile tende ad essere circa il 50% di quello utilizzato

↓
rimedio

COMPATTAZIONE

spostare le partizioni in maniera da poter unire buchi separati.

- onerosa (serve la rilocazione dinamica)
- se effetta I/O, il processo non può essere rilocato

PAGINAZIONE

Permettere una allocazione di memoria non contigua ai processi

- La memoria fisica viene divisa in **FRAMES** di dimensione fissa
- Lo spazio di indirizzi logico è diviso in **PAGINE**, di dimensione pari ai frame
- **TABELLA DELLE PAGINE** associa le pagine ai frames e permette alla MMU la traduzione

• VANTAGGI:

- non vi è frammentazione esterna
- gli spazi logici e fisici sono separati

SVANTAGGI frammentazione interna

TRADUZIONE DEGLI INDIRIZZI un indirizzo logico è diviso in:

- numero di pagina indice nella tabella delle pagine
- offset di pagina offset all'interno della pagina
- Con 2^m indirizzi logici e page grandi $2^n \rightarrow$ n° di pagine: 2^{m-n} .

La MMU traduce un indirizzo logico in fisico nel seguente modo:

- 1 estrae il numero di pagina dall'indirizzo logico
- 2 lo utilizza per ricavare dalla tabella delle pagine il corrispondente numero di frame
- 3 concatena il numero di frame all'offset di pagina e ottiene l'indirizzo fisico

FRAMMENTAZIONE E DIMENSIONE DELLA PAGINA nel caso medio, la frammentazione

è di mezzo frame per processo

- per ridurre lo spreco di memoria \rightarrow fare pagine di dimensione ridotte

\hookrightarrow ma aumenta il numero totale di pagine e la dimensione della tabella

Tendenzialmente si hanno pagine di dimensioni maggiori

SUPPORTO SO il so deve:

- mantenere la tabella delle pagine di ciascun processo
- mantenere una tabella dei frame, che indica lo stato di ogni frame
- traduzione degli indirizzi senza MMU se deve accedere alla memoria di un processo.

SUPPORTO HIN la tabella delle pagine è contenuta in memoria centrale.

La MMU utilizza quindi due registri:

- page table base register indirizzo fisico dell'inizio della tabella
- page table length register dimensione della tabella

Richiede 2 accessi in memoria ed è troppo oneroso, si usa la cache **translation Lookaside buffer**

TRANSLATION LOOKASIDE BUFFER è di solito piccolo e si effettua il suo flush ad ogni cambio di contesto

- address space identifier identificano univocamente uno spazio di indirizzi, così da poter mantenere le entry di più tabelle delle pagine. Vengono usati anche per proteggere la memoria

TEMPO EFFETTIVO DI ACCESSO ALLA MEMORIA

- TLB HIT numero di pagina trovato
- TLB MISS numero di pagina non trovato
- Effective Access Time = $ma \cdot hr + 2ma(1-hr) = ma(2-hr)$

$$\text{Effective Access Time} = \underbrace{ma}_{\text{memory access}} \cdot \underbrace{hr}_{\text{hit ratio}}$$

POLITICHE DI SOSTITUZIONE se capita una TLB miss ma il TLB è pieno, occorre sostituire

- una entry con una strategia:
- last recently used
 - Round-Robin
 - Casuale

Il SO potrebbe:

- con un'interrupt, partecipare alla decisione
- vincolare delle TLB entry che non vogliono siano mai sostituite

PROTEZIONE DELLA MEMORIA il PTLR permette di "tagliare" la tabella in maniera da ridurre gli indirizzi logici disponibili al processo, quindi si può ridurre la dimensione della tabella delle pagine

- alternativamente, mettere in ogni entry un bit di validità, con una maggiore flessibilità

PAGINE CONDIVISE è facile condividere memoria fisica tra più processi, basta che i frame siano nelle tabelle delle pagine dei processi che condividono

- condividere il codice (rientrante), realizzare una comunicazione interprocesso

STRUTTURA DELLE TABELLE DELLE PAGINE

esistono varie soluzioni per strutturarle in maniera che siano sufficientemente piccole ed efficienti:

- **TABELLE DELLE PAGINE GERARCHICHE** servono per evitare di allocare una tabella delle pagine in una regione di memoria contigua, in pratica si pagina la tabella delle pagine, arrivando a una tabella a due (o più) livelli, in cui ogni n° di pagina è suddiviso in un n° di pagina e un offset.

↳ supporta contemporaneamente pagine di dimensioni diverse

✗ aumenta il numero di accessi in memoria.

- **DI TIPO HASH** organizzata come una tabella hash (con gestione delle collisioni)

- **INVERTITE** un entry per ogni frame, anziché per ogni pagina

↳ una sola tabella per ogni processo

✗ maggiore tempo di accesso, non c'è condivisione

SWAPPING

È una tecnica che permette di eseguire più processi di quanti la memoria fisica ne riesca a contenere.

- Spostare un processo pronto o in attesa in una memoria secondaria, permettendo ad un altro processo di andare in esecuzione

SWAPPING STANDARD un intero processo viene spostato dalla memoria centrale alla backing store (processo di **snap out**), insieme alle relative strutture dati.

↪ è molto oneroso

SWAPPING CON PAGINAZIONE sposta solo un sottoinsieme delle pagine di un processo fino a quando si libera abbastanza spazio per il nuovo processo (operazioni di **page out** e **page in**)

MEMORIA VIRTUALE

È la completa separazione tra memoria logica e memoria fisica di un programma.

- un programma può essere eseguito anche se solo una parte di esso è in memoria fisica.

PAGINAZIONE SU RICHIESTA demand paging è basata sull'hardware di paginazione, quindi non si porta tutto il processo in memoria, ma solo le pagine necessarie

- **page Fault** interruzione se si tenta di accedere ad una pagina con bit di validità a 0, gestita dal SO che porta la pagina in memoria
- **swap space** memorizza le pagine fuori memoria nel backing store.

GESTIONE DI UNA PAGE FAULT il SO deve decidere se la pagina è non valida oppure non in memoria.

- nel primo caso: abort del processo
- nel secondo: caricamento della pagina dal backing store:
 - 1 il SO trova un frame libero
 - 2 schedula l'operazione di caricamento
 - 3 possibile cambio di contesto nell'attesa
 - 4 aggiorna la tabella delle pagine con il frame caricato
 - 5 quindi ritorna dall'interruzione di page fault, e il processore riesegue l'operazione che era stata interrotta.

POLITICHE DI CARICA/MENTO PAGINA

- su richiesta pura le pagine vengono caricate solo quando servono
- prefetching vengono caricate anche altre pagine in un intorno della pagina richiesta

MODELLO DI LOCALITÀ La paginazione su richiesta è efficiente se i page fault sono pochi rispetto alle istruzioni eseguite, secondo la località un processo, in un certo momento, opera su una certa località: i page fault sono pochi se ogni processo si mantiene la sua località.

GESTIONE DEI FRAME LIBERI

Sono gestiti con una lista. Vanno azzerati (zero-fill) prima di essere assegnati ad un processo e vanno mantenuti sopra una certa soglia, quindi la lista deve essere ripopolata.

COPY-ON-WRITE permette ad un processo figlio di condividere tutte le pagine con il processo padre, utile per la fork.

Se uno dei processi effettua una modifica, la pagina viene copiata.

GRADO DI MULTIPROGRAMMAZIONE

Numero massimo di processi che lo scheduler di breve termine può mandare in esecuzione. Determina quindi l'occupazione totale di memoria fisica da parte dei processi (senza memoria virtuale, il grado è uguale al numero di immagini in memoria).

- i processi in memoria sono ready + running + waiting
- più è alto il grado, maggiore è l'utilizzo dei processori

CONTROLLO DEL GRADO

- lo scheduler a lungo termine (admission scheduler) decide se ammettere un processo nella ready queue
- lo scheduler a medio termine sospende e riprende processi dalla swapping memory
- lo scheduler di breve termine (della cpu) non influenza sul grado

ALLOCAZIONE E SOSTITUZIONE DELLE PAGINE

SOSTITUZIONE quando vengono esauriti i frame (**surreallocazione**) si effettua una sostituzione, ovvero si libera un frame inutilizzato.

- si usa l'algoritmo di sostituzione per trovare il frame vittima
- si effettua il page out
- si aggiornano le tabelle

OTTIMIZZAZIONI

- le pagine read-only non hanno bisogno di page out
- le pagine con bit di modifica a 0 non necessitano di page out (non sono mai state modificate)

ALGORITMI DI ALLOCAZIONE determina quanti frame assegnate ad ogni processo.

- Scelto tra un numero minimo hardware e uno massimo vincolato alla quantità di memoria fisica
- l'allocazione può essere:
 - **FISSA** il numero di frame non cambia durante l'esecuzione
 - **VARIABILE** il numero di frame può cambiare durante l'esecuzione

UNIFORME con m frame e n processi, ogni frame riceve m/n frame, con $m \leq n$ frame usati come buffer

- ↪ semplice
- ↪ processi diversi potrebbero avere esigenze diverse
- ↪ all'aumentare del grado, diminuisce il numero di frame per processo

PROPORZIONALE con m frame, al processo i con richiesta s_i di memoria virtuale, ottiene un numero di frame $a_i = s_i / \sum_{i=1}^n s_i \cdot m$

ALGORITMI DI SOSTITUZIONE determina quale è il frame vittima

Ogni S.O. ha il suo, vengono confrontati in base al numero di page Fault

- Può essere:
 - **GLOBALE** selezionato tra tutti i frame di tutti i processi (più usato)
 - ↪ utilizzo efficiente della memoria
 - ↪ perdita di pagine a causa di altri processi
 - **LOCALE** selezionato tra i frame del processo che ha generato page fault
 - ↪ stesso numero di pagine nel tempo per ogni processo
 - ↪ possibile sottoutilizzo della memoria

POLITICA DI RECUPERO PAGINE

politica di sostituzione globale, con l'obiettivo di mantenere il numero di frame liberi in un certo intervallo

- quando scendono sotto una scelta soglia, il kernel avvia la routine reaper che recupera pagine da tutti i processi

THRASHING

Quando uno o più processi hanno un numero troppo basso di frame in memoria diventa elevata la probabilità di un page fault, ma d'altra parte se non ci sono frame liberi si rischia di sostituire una pagina che servirà poco dopo.

- Quando un processo trascorre più tempo nella gestione dei suoi page fault che non nell'esecuzione del suo programma si dice che il processo è in thrashing, e se la strategia di sostituzione di pagina è globale, potrebbe far andare altri processi in thrashing

LOCALITÀ E THRASHING il thrashing avviene perché ad un certo punto $\sum_{i=1}^n \text{loc}_i > \text{mem}$, quindi ci sono dei processi in memoria che non hanno la propria località completamente in memoria.

Un processo singolo ci va se:

- la sua località è più grande della memoria fisica, oppure
- se è più grande della memoria che gli è stata allocata

ANATOMIA se in un certo istante il grado di multiprogrammazione è elevato, diminuisce la memoria disponibile per ciascun processo: alcuni processi potrebbero non avere tutta la loro località in memoria, generando dei page fault.

Con pochi frame liberi o inutili, si sostituiscono frame appartenenti a qualche località

↳ in tal modo i processi in memoria continuano a sostituire frame utili, generando molti page fault in cascata

↳ inoltre, i processi in page fault entrano in stato di attesa per il dispositivo di paginazione e diminuisce l'uso della CPU

↳ uno scheduler che non individua la situazione, potrebbe portare processi in memoria per aumentare l'utilizzo della CPU.

Ma questo riduce ulteriormente i frame disponibili per ciascun processo

RIMEDI

- individuare quando si verifica e
- ridurre il grado di multiprogrammazione, scegliendo dei processi vittima da sconfiggere

MODELLO DEL WORKING SET

- tecnica per calcolare la località approssimata di un processo
- si definisce a priori Δ finestra del working set, come un numero fisso di riferimenti in memoria
 - il working set è l'insieme dei riferimenti alla memoria che vengono effettuati nella finestra:
 - $\Delta \ll$ w.s. non include l'intera località
 - $\Delta \gg$ sovrappone più località
 - $\Delta \rightarrow \infty$ w.s. include tutte le pagine usate dal processo
 - se calcoliamo per ogni processo i la dimensione del working set wssi, otteniamo che la domanda totale di frame D è $\sum_{i=1}^n WSS_i$
 - ↳ se $D > m$ vi è thrashing (m numero totale di frame)

FREQUENZA DEI PAGE FAULT

alternativo al working set e più diretto, basato sul calcolo della frequenza dei page fault dei processi

- Si stabilisce un range accettabile nella frequenza dei page fault (page fault frequency)
 - $PFF < \text{soglia minima}$ il processo perde frame
 - $PFF > \text{soglia massima}$ il processo acquista frame, se non ci sono abbastanza frame liberi, occorre sospendere uno o più processi per liberare frame

LIVELLO DI RETE - Piano dei dati

DATA PLANE Locale, funzione interna al router: determina come i datagrammi arrivati in input al router debbano essere inoltrati alle porte in output.

CONTROL PLANE network-wide, determina come i datagrammi viaggiano da router a router, dall'host di partenza fino a quello di destinazione.

Esistono due approcci:

- **TRADITIONAL ROUTING ALGORITHMS** implementati nei router, ognuno comunica con gli altri nel control plane
- **SOFTWARE-DEFINED NETWORKING (SDN)** remote controller computers, che installano le forwarding tables nei routers

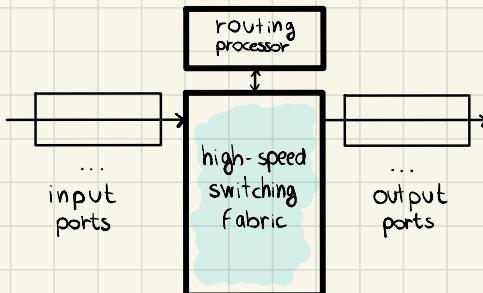
NETWORK SERVICE MODEL definisce le caratteristiche del trasporto dei dati:

- individual datagram services: garantire la consegna, non superare una soglia di delay...
- flow of datagram services: consegna ordinata, banda minima, restrizioni sui cambiamenti...

NETWORK-LAYER SERVICE MODEL caratteristiche del livello di rete:

- architettura di rete: internet
- service model: best effort → NESSUNA garanzia su:
- QoS:
 - Bandwidth: nessuna
 - Loss: no
 - Order: no
 - Timing: no
 - arrivo a destinazione dei pacchetti
 - consegna in tempo e ordinata
 - banda disponibile

OVERVIEW ARCHITETTURA ROUTER rappresentazione ad alto livello



FUNZIONI DELLE PORTE IN INPUT

- line termination ricezione bit-level
- link layer e.g. ethernet...
- switching decentralizzato usando i valori negli header, trova la porta di output in base alla forwarding table
 - se i datagrammi arrivano troppo velocemente, vengono accodati (input port queuing)
 - destination based forward in base al solo indirizzo ip di destinazione

SWITCHING FABRICS

trasferisce i pacchetti da un input link verso l'appropriato output link.

- switching rate frequenza con la quale i pacchetti sono trasferiti

INPUT PORT QUEUING

se lo switch fabric è più lento delle porte in input, si possono creare delle code.

- head-of-the-line HOL blocking i datagrammi in testa alla coda impediscono alla coda di avanzare

OUTPUT PORT QUEUING

coda in output

- buffering si usa quando i datagrammi arrivano dal fabric più velocemente del link transmission rate.
I datagrammi possono essere persi (drop policy)
- scheduling discipline scelta del datagramma da trasmettere (priority scheduling)

IP DATAGRAM FORMAT

le caratteristiche principali sono:

- dimensione massima di 65KBytes
- dimensione tipica di 1500 Bytes (o meno)
- introduzione di overhead:
 - 20 Bytes TCP
 - 20 Bytes IP
 - app Layer TCP+IP

INTERFACCIA connessione tra host/router e i collegamenti fisici. I router possiedono tipicamente diverse interfacce, gli host una o due.

IP ADDRESS è un identificatore a 32bit associato ad ogni interfaccia.

- rappresentato con la dotted decimal notation.

SUBNET interfacce che possono raggiungersi fisicamente senza passare per un router

In particolare, ogni indirizzo IP è strutturato in: SUBNET PART + HOST PART

- permette di istituire sottoreti isolate

CLASSI gli IP, in base al tipo di subnet, si possono classificare in due modi:

- **Classless InterDomain Routing** porzioni subnet di lunghezza arbitraria, nel formato a.b.c.d./x, dove la x sono il numero di bit dedicati alla subnet (ovvero la **subnet mask**)
- **Classful** non c'è bisogno della subnet mask, ma gli indirizzi sono divisi in classi. È un modello rigido e non più utilizzato.

INDIRIZZI SPECIALI identificano un gruppo di hosts:

- **subnet address** l'host part è tutto a zero, identifica la rete
- **direct broadcast address** l'host part è tutto a uno, identifica tutti gli host
- **Limited broadcast address** 255.255.255.255, broadcast nella stessa rete, che non viene inoltrato dai router.

OTTENIMENTO IP l'indirizzo si ottiene con:

- **Statico** specificato in configurazione
- **DHCP dynamic host configuration protocol**, gli host ottengono dinamicamente l'indirizzo da un server quando accedono alla rete.

Per richiedere un indirizzo:

- 1 **DHCP DISCOVER** cerca un DHCP server
- 2 **DHCP OFFER** il server risponde con un indirizzo IP
- 3 **DHCP REQUEST** l'host accetta l'indirizzo
- 4 **DHCP ACK** il server conferma l'assegnazione

NETWORK ADDRESS TRANSLATION tutti gli host in una rete locale condividono un solo indirizzo per comunicare con l'esterno.

- gli host hanno indirizzi privati → un solo indirizzo pubblico

L'implementazione avviene come segue:

- ai datagrammi in uscita viene cambiato il source IP e la porta con il NAT IP e una nuova porta
- la traduzione viene salvata grazie alla **nat translation table**
- ai datagrammi in entrata si applica il processo inverso (grazie alla table).

LIVELLO DI RETE - Piano di controllo

PROTOCOLLO DI ROUTING determina il percorso migliore da mittente a destinatario, attraverso la rete di router.

GRAFO DI ASTRAZIONE una rete può essere astratta in un grafo, in cui ogni link diretto ha un costo (∞ se non esiste un link diretto)

CLASSIFICAZIONE gli algoritmi vengono classificati in:

- **GLOBALI** tutti i router conoscono la rete e i costi dei link (*link state*)
- **DECENTRAZZATI** iterativo, scambio di informazioni solo con i vicini (*distance vector*)
- **STATICI** le route cambiano poco nel tempo
- **DINAMICI** le route cambiano periodicamente

} opposti
} opposti

DISTANCE VECTOR implementazione dell'equazione di Bellman-Ford:

$$\text{dist}(A, X) = \min \{ \text{dist}(V, X) + c(A, V) \}$$

L'algoritmo che la implementa:

- di volta in volta, ogni nodo invia la sua stima del distance vector ai vicini
- quando un router riceve una stima DV, aggiorna anche la propria
- la distanza stimata converge al costo minore

È un algoritmo:

- **iterativo** lo fa ogni nodo
- **asincrono** non deve essere sincronizzato
- **distribuito** è in ogni router
- **terminante** se non ci sono aggiornamenti, nessuno fa nulla

- LINK STATE** distribuzione in "selective flooding" delle informazioni relative alla topologia di rete (ogni router deve comunicare in broadcast il proprio link state agli altri router)
- utilizza l'algoritmo di Dijkstra
 - centralizzato tutti i nodi hanno le stesse informazioni
 - si basa su una Forwarding table
 - iterativo: dopo le iterazioni, conosce il costo minore per le destinazioni

DIFFERENZE fra i due algoritmi:

DISTANCE VECTOR

- semplice e intuitivo
- pochi messaggi scambiati
- lenta convergenza
- protocollo BGP

LINK STATE

- selective flooding
- molti messaggi scambiati
- robusto
- convergenza
- OSPF, IS-IS

SCALABILITÀ DELLA RETE aggregazione della rete in regioni chiamate "autonomous systems" (detti anche domini), in due tipi:

- intra-AS routing nella stessa AS, esiste un gateway router per comunicare con altre AS
- inter-AS routing tra AS, comunicazione tra i gateway router

INTER-AS ROUTING il router propaga agli altri le informazioni sulle reti raggiungibili

INTRA-AS ROUTING utilizza i protocolli detti interior gateway protocol, e possono essere:

- RIP (routing information protocol)
- EIGRP
- OSPF

OPEN SHORTEST PATH FIRST ogni router propaga il proprio link state agli altri router e conosce lo stato della rete

HIERARCHICAL OSPF basato su due livelli local area e backbone, ogni nodo conosce la direzione per raggiungere altre destinazioni

BORDER GATEWAY PROTOCOL consente alle subnet di fare conoscere la propria esistenza e le destinazioni raggiungibili al resto della rete

- eBGP ottiene le informazioni dalle vicine AS
- iBGP propaga le informazioni ottenute ai router interni alla AS

DIFFERENZE DI PROTOCOLLO i protocolli usati nelle intra e nelle inter sono differenti sulla base di:

• **PRIVACY**

inter l'admin ha il controllo sul traffico e su chi lo genera

intra policy più "leggere"

- **SCALE** riducono il traffico di update e la grandezza delle tabelle

• **PERFORMANCE**

inter la policy ha la meglio sulle performance

intra si può concentrare sulle performance

INTERNET CONTROL MESSAGE PROTOCOL usato per la comunicazione di informazioni a livello di rete

- report di errori, echo di richieste, repliche (e.g. ping)

TRACEROUTE source invia un set di segmenti UDP alla destinazione, ogni router incontrato sul percorso elimina il datagramma e risponde con un ICMP type 11, code 0.

La destinazione raggiunta risponde con un ICMP type 3, code 3 - "port unreachable".

La source registra il RTT di ogni pacchetto le risposte possono contenere informazioni come il nome del router che risponde e il suo indirizzo IP.

LIVELLO DI COLLEGAMENTO e RETI LOCALI

SERVIZI DEL LIVELLO

- incapsula i datagrammi in frame, aggiungendo l'header
- controlla l'accesso ai canali
- consegna affidabile tra nodi vicini (link wireless hanno un rate di errore maggiore)
- controllo di flusso
- controllo e correzione degli errori
- comunicazione half-duplex o full-duplex

TIPI DI LINK

- point-to-point (collegamento diretto)
- broadcast (link condiviso)

MULTIPLE ACCESS PROTOCOLS con un singolo canale broadcast condiviso, due o più trasmissioni temporanee possono generare interferenze e collisioni.

Bisogna idealmente verificare quando un nodo può trasmettere:

- 1 un nodo trasmette con rate R
- 2 M nodi con rate R/M
- 3 decentralizzato, senza nodi che controllano la trasmissione
- 4 Semplice

TIPI DI BROADCAST

ci sono 3 diversi tipi:

- channel partitioning partizione del canale ad uso esclusivo di un singolo nodo
- random access nessun controllo, le collisioni vengono risolte
- a turni ogni nodo attende il proprio turno per poter comunicare

RANDOM ACCESS PROTOCOLS

Ogni nodo trasmette senza coordinazione, controllando e correggendo le collisioni.

Lo sono i seguenti protocolli:

SLOTTED ALOHA

Ogni nodo ha un eguale slot di tempo nel quale può trasmettere ed è sincronizzato agli altri nodi.

Se si presenta una collisione, il frame viene ritrasmesso fino al successo (probabilità p)

- trasmissione continua
- decentralizzato
- collisioni fanno perdere slot
- slot in stallo
- clock di sincronizzazione

L'efficienza è del 37%

PURE ALOHA

Senza slot e sincronizzazione, aumentano le collisioni ma è più semplice.

L'efficienza è del 18%

CARRIER SENSE MULTIPLE ACCESS

Un nodo ascolta prima di trasmettere: se il canale è occupato, avvia una trasmissione differita, altrimenti invia l'intero frame

CARRIER SENSE MULTIPLE ACCESS / COLLISION DETECTION

CSMA con controllo delle collisioni: se avviene, le trasmissioni coinvolte vengono abortite, riducendo lo spreco dei canali.

L'algoritmo è come segue:

- 1 controllo dello stato del canale e inizio della trasmissione
- 2 se tutto viene trasmesso, l'operazione è terminata
- 3 se collide, abortisce e invia un Jam signal
- 4 arretra nella trasmissione (**backoff**)

Migliore dell'ALOHA.

A TURNI uso esclusivo del canale lo soho i seguenti protocolli.

POLLING

Un nodo master gestisce la comunicazione di nodi "dumb"

- ▷ overhead
- ▷ Latenza
- ▷ Single point of failure sul server

TOKEN PASSING

Un token è passato tra i nodi per consentire l'uso del canale.

- ▷ overhead
- ▷ Latenza
- ▷ single point of failure sul token

MEDIUM ACCESS CONTROL ADDRESS

 usato localmente per trasmettere tra interfacce fisicamente collegate

- indirizzo a 48 bit associato all'interfaccia fisica (è univoco e assegnato dai produttori attraverso l'IEEE).

ADDRESS RESOLUTION PROTOCOL risolve indirizzi IP in indirizzi MAC grazie a tabelle di corrispondenza dette arp table, rinnovata circa ogni 20 minuti e propria di ogni dispositivo di rete.

ETHERNET

 tecnologia LAN dominante, semplice ed economica.

- connectionless non c'è un handshake
- unreliable non ci sono segnali di ACK

TOPOLOGIA DI RETE

- **BUS** tutti i nodi comunicano su un unico collegamento, con collisioni
- **SWITCHED** la rete è governata dagli switch

FRAME STRUCTURE i datagrammi IP vengono incapsulati in frame ethernet composti da:

- **preamble** usato per sincronizzare mittente, destinatario e il clock
- **addresses** indirizzi MAC mittente e destinatario
- **type** indica il protocollo di livello superiore (IP, AppleTalk)
- **CRC cyclic redundancy check**
- **payload** data

SWITCH dispositivo del livello di collegamento che svolge un ruolo attivo:

- store & forward dei frame
- inoltra i pacchetti in base all'indirizzo MAC
- è trasparente agli host
- non ha bisogno di configurazione (plug & play), e possono comunicare tra loro
- grazie allo switch, gli host possono comunicare simultaneamente senza collisioni in full duplex

SWITCH TABLE lo switch corrisponde le proprie interfacce agli indirizzi MAC degli host

- viene costruita dallo switch leggendo il MAC di origine dei pacchetti che gestisce

SWITCHING nella pratica, l'algoritmo è:

- 1 registra il MAC
- 2 interroga la switch table
- 3 se trova l'indirizzo, inoltra il frame (drop se corrisponde al mittente)
- 31 altrimenti lo inoltra a tutti i collegamenti

SWITCH

collegamento

vs

LIVELLO

ROUTER

rete

costruita leggendo i
pacchetti

FORWARDING TABLE

costruite con algoritmi
di routing

FILE E OPERAZIONI SU FILE

FILE SYSTEM è il modo attraverso il quale il sistema operativo memorizza in linea i dati e i programmi:

- insieme di file
- struttura delle directory

FILE è una unità di memorizzazione logica, un insieme di informazioni correlate, registrate in memoria secondaria, alle quali è stato dato un nome.

- i suoi attributi: nome, identificatore, tipo, locazione, dimensione, protezione, ora, data; sono memorizzati nella directory.
- creazione, lettura, scrittura, seek, cancellazione, truncamento, apertura, chiusura.

TABELLE DEI FILE APERTI Un file può essere aperto contemporaneamente da più processi, il SO mantiene una tabella dei file aperti per ogni processo: ogni entry punta ad una corrispondente tabella dei file aperti di sistema, che mantiene un contatore delle aperture per ogni file.

- se il contatore arriva a zero, la entry viene liberata dalla tabella di sistema.
 - ad ogni file aperto è associato:
 - puntatore alla posizione (nella tabella del processo)
 - contatore delle aperture
 - locazione del file
 - diritti di lettura/scrittura
- } (nella tabella di sistema)

LOCK DEI FILE associa ai file dei lock per coordinare più processi

- CONDIVISO più processi possono acquisirlo
- ESCLUSIVO solo un processo alla volta può acquisirlo
- OBBLIGATORI / MANDATORY il SO impedisce l'accesso al file se il lock è acquisito da qualche processo
- CONSULTIVI / ADVISORY il SO non regola l'accesso al file

STRUTTURA DEI FILE

- NESSUNA STRUTTURA un file è una sequenza di byte
- SEQUENZA DI RECORD di testo o binari
- STANDARD come per i file eseguibili

IMPACCAMENTO stabilisce come un certo numero di record logici di un file è memorizzato in un blocco fisico del dispositivo di memoria secondaria

- La tecnica determina quanti record sono memorizzabili in un blocco fisico e la dimensione totale del file il grado di frammentazione interna della memoria secondaria.

METODI DI ACCESSO

- **SEQUENZIALE** il file è una sequenza di record a lunghezza fissa, si legge /scrive il successivo record dalla posizione corrente
- **DIRETTO** si può accedere anche all'n-esimo record
- **INDICIZZATO** accesso basato su una chiave , il SO mantiene un indice

DIRECTORY

È una tabella di simboli che traduce il nome di un file negli elementi in essa contenuti; e risiede, come i file, su disco

- operazioni di ricerca, creazione, cancellazione, ridenominazione, elenco, attraversamento

STRUTTURA

- **AD UN LIVELLO** una sola directory per tutti i file

- ↳ Semplice

- ↳ naming, raggruppamento

- **A DUE LIVELLI** la dir principale contiene una sottodir per ogni utente , i file vengono identificati univocamente da una path name . I file di sistema sono in directory speciali

- **AD ALBERO** ogni directory contiene ricorsivamente files e altre directory. Si possono specificare path relativi

- **A GRAFO ACICLICO** permette l'aliasing, ovvero assegnare più di un nome allo stesso file

- ↳ problema di attraversamento

- ↳ se si cancella un file con alias:

- **hard links** file cancellato sse $\exists 0$ riferimenti

- Link simbolici i link sono riferimenti che restano dangling

- **A GRAFO GENERICO** viene introdotto un garbage collector

PROTEZIONE

Le informazioni devono essere protette dai danni fisici e dagli accessi impropri.

Un sistema multiutente permette un accesso controllato ai file degli utenti

LISTE DI CONTROLLO DEGLI ACCESSI ad ogni file / directory è associata un' ACL che specifica gli utenti che possono accedere al f./dir., con i relativi permessi di accesso

↳ Possono diventare molto lunghe

• si raggruppano gli utenti in classi distinte:

- Proprietario l'utente che possiede
- Gruppo di utenti che condivide
- Pubblico tutti gli altri utenti

ORGANIZZAZIONE DEL FILE SYSTEM

ORG. STRATIFICATA il file system è suddiviso in:

• fs Logico gestisce i metadati ed è responsabile di protezione e sicurezza.

• Modulo di organizzazione dei file traduce gli indirizzi dei blocchi logici in indirizzi dei blocchi fisici e gestisce lo spazio libero

• fs di base legge e scrive blocchi fisici dalle unità disco, mantiene buffer e cache

↳ minima duplicazione di codice

↳ all'aumentare degli strati, diminuiscono le prestazioni

REALIZZAZIONE DELLE OPERAZIONI

STRUTTURE DATI servono per realizzare operazioni, sono su disco o nel SO:

Su disco:

- **BOOT CONTROL BLOCK** informazioni necessarie al sistema per effettuare l'avvio del SO
- **VOLUME CONTROL BLOCK** informazioni su volumi e partizioni del disco
- **STRUTTURA DELLE DIRECTORY** una per file system, informazioni relative alla organizzazione dei file
- **FILE CONTROL BLOCK** uno per file, informazioni relative ad un singolo file

In memoria:

- Tabella di montaggio info relative a ciascun volume montato
- Cache della struttura della dir info delle dir usate più recentemente
- Tabella di sistema dei file aperti: file aperti dai processi
- Tabella dei file aperti + processo + tabella dei file aperti dal processo: puntatori a quella di sistema
- Buffer di lettura /scrittura

OPEN il SO verifica se nella tabella di sistema dei file aperti esiste una entry del file e aggiorna la tabella del processo, altrimenti aggiunge il file alla tabella.

Ritorna un indice alla tabella dei file aperti del processo.

CLOSE elimina l'elemento nella tabella dei file aperti del processo e decrementa il contatore in quella di sistema.

Se è zero, la entry viene cancellata.

REALIZZAZIONE DELLE DIRECTORY

LISTA LINEARE lista di nomi di file, con puntatori ai blocchi dei dati

- ↳ onerosa in termini di tempo
- ↳ si ottimizza con: caching, ordinamento, bilanciamento

TABELLA HASH funzione hash per calcolare l'indice nella lista a partire dal nome del file

- ↳ collisioni e dimensione della lista, risolvibili con liste e dimensione fissa.

METODI DI ALLOCAZIONE

Sono i metodi che i file system usano per assegnare i blocchi liberi ai file.

ALLOCAZIONE CONTIGUA

ad ogni file è associato un insieme contiguo di blocchi

- ottime performance
- semplice
- ▷ occorre sapere la dimensione del file
- ▷ occorre trovare sufficiente spazio contiguo
- ▷ frammentazione esterna

EXNET un file ha più exnet (= porzione contigua) in caso in cui il file debba crescere

ALLOCAZIONE CONCATENATA

ogni file è composto da una lista concatenata di blocchi

- non si specifica una dimensione
- no limite di crescita
- no frammentazione esterna
- ▷ accesso diretto inefficiente
- ▷ i puntatori sovrappossono spazio
- ▷ affidabilità

Si potrebbe invece operare su cluster

FAT : FILE ALLOCATION TABLE

simile al precedente, ma i puntatori ai blocchi sono tutti raccolti in una tabella (file allocation table) in una sezione del volume

- necessita di caching
- miglior performance di accesso diretto
- replica della FAT per una migliore affidabilità

ALLOCAZIONE INDICIZZATA

ogni file ha un blocco indice, strutturato come un array di indirizzi di blocchi del disco

- no frammentazione esterna
- no dimensione del file
- efficiente accesso diretto
- ▷ overhead blocco indice con file di dimensioni ridotte

SCHEMI PER BLOCCO INDICE

- **concatenato** l'ultima parola del blocco indice punta ad un successivo blocco indice
- **a più livelli** i puntatori del blocco indice puntano a blocchi indice di secondo livello e così via per N livelli
- **combinato** tipo UNIX-like ma non ho capito una sega, me spiace d'esser anna così

PRESTAZIONI

dipende dalla modalità di accesso

- contigua → accesso sequenziale e diretto
- concatenata → accesso sequenziale
- indicizzata → dipende dal blocco indice

GESTIONE DELLO SPAZIO LIBERO

Il file system mantiene un elenco dei blocchi liberi, La **Lista dello spazio libero**, che può essere organizzata in diversi modi.

VETTORI DI BIT

ogni blocco è rappresentato da un bit

- semplice
- efficiente solo se tutto il vettore è in memoria (si ottimizza con il **clustering**)

LISTE CONCATENATE

approccio simile all'allocazione concatenata

- può non essere semplice ottenere spazio contiguo

VARIANTI DELLE LISTE

- **RAGGRUPPAMENTO** memorizzate in ogni blocco libero gli indirizzi di n altri blocchi liberi
- **CONTEGGIO** memorizzate in un blocco libero l'indirizzo del primo blocco libero + numero di successivi blocchi contigui liberi, nel caso ve ne siano