Sistemi Ipermediali
Sistemi per media continui

Claudio E. Palazzi
Università degli Studi di Padova

Lucidi a cura di: Ombretta Gaggi, Claudio Palazzi

Sistema Operativo

- Il Sistema Operativo (SO) è quello strato di software che gestisce le risorse hardware del computer
- Essenzialmente, si può considerare il SO come quello strato di software che fornisce a programmi e applicazioni una macchina virtuale attraverso la quale accedere alle risorse hardware del computer

Sistema Operativo

- In sostanza, il SO virtualizza le risorse di una macchina reale e crea una macchina astratta che:
  - offre un ambiente di lavoro amichevole per gli utenti finali
  - alloca a programmi ed utenti specifici le risorse hardware e software disponibili, ottimizzandone l'utilizzo
  - controlla l'esecuzione dei programmi ed in particolare l'uso della CPU, memoria e dei dispositivi di I/O

Alcune definizioni

- **algoritmo:** sequenza di passi che consentono di risolvere un problema
- **programma:** descrizione di un algoritmo tramite un linguaggio che ne rende possibile l'esecuzione da parte di un processore
- **evento:** esecuzione di una delle istruzioni del processore
- **processo:** sequenza di eventi prodotti da un processore nell'esecuzione di un programma

Osservazioni

- Utente, tramite Applicazioni MM, deve percepire audio e video in maniera naturale, error-free
- Integrazione di multimedia di tipo discreto e continuo, richiede controllo da parte del SO
- Dati processati e varie risorse sono localmente sotto il controllo del SO

Media Continui

- Suoni o motion video dove il tempo è parte della semantica del media stesso
  - Le reti che trasportano media continui devono rispettare la dipendenza dal tempo
  - I dati persi o ritardati nella trasmissione sono un problema perché non possono essere ritrasmessi
  - È richiesta la dipendenza dal tempo quando i media vengono integrati
  - Richiede degli strumenti di authoring dipendenti dal tempo

Media Discreti

- Testo, grafica e immagini che sono visualizzati secondo un'ampia varietà di tempistiche e ordini
  - Quando i media discreti sono integrati con i media continui, la dipendenza dal tempo deve essere mantenuta anche per quelli discreti

Problemi Tipici

- Compiti tipici di un SO (in relazione a multimedia):
  - Real-Time
  - Gestione delle Risorse
  - Gestione dei Processi
  - File Systems
  - Sincronizzazione + Interprocess Communication

Real-Time

- *Def*: *Processo Real-Time*

  Processo che elabora e restituisce in output risultati ***entro*** un determinato Δt

- *Def*: *Sistema Real-Time*

  Sistema che riceve informazione dall'ambiente, la processa restituisce output ***entro*** certi vincoli temporali

Sistemi Real-Time

- Correttezza della Computazione
  - Errorless = computazione corretta
  - In termini temporali = entro Δt

- Failure
  - Software/Hardware failure
  - Computazione OK *ma* fuori da Δt

Def: Deadline

- Limite temporale per la presentazione in output di un risultato
  - *Soft Deadline*: limite non determinato esattamente, solo di riferimento, non troppo vincolante
  - *Hard Deadline*: limite ***non*** violabile

    `hard deadline violation = failure`

Multitasking

- I sistemi operativi che permettono di eseguire contemporaneamente più programmi (ovvero gestire più processi) sono detti sistemi multitasking
- In sistemi con un solo processore l'esecuzione contemporanea dei processi è virtuale, poiché la CPU esegue un'istruzione per volta
- Il meccanismo che si utilizza per poter tenere più programmi in esecuzione è la ripartizione del tempo (time sharing)

Multitasking

- La ripartizione del tempo di CPU fra tutti i processi attivi, avviene per porzioni di tempo (dette time slice) così piccole che l'utente ha la sensazione che i processi avanzino parallelamente
- La presenza di un insieme di risorse comuni a più applicazioni (e di conseguenza a più processi) ingenera competizione (o meglio, concorrenza)

Scheduler

- Il SO si fa carico della sincronizzazione dei processi che intendono utilizzare le risorse comuni che gestisce direttamente
- La componente del SO che si occupa di assegnare il processore ai processi è detto scheduler
- I processi dunque non sono tutti attivi contemporaneamente

Altre componenti

- Altre componenti del SO di cui discuteremo:
  - Il gestore della memoria che ha il compito di gestire la memoria in modo trasparente ed efficiente consentendo ad ogni programma di lavorare in un proprio spazio di indirizzamento (virtuale)
  - Il gestore dei file (file system), che si occupa di organizzare le informazioni, che vengono strutturate in contenitori logici (file) identificati mediante un nome logico (filename)
  - Il gestore delle periferiche che consente la concorrenza sulle periferiche e consente all'utente di operare mediante periferiche astratte

Sistemi Real Time

- Caratteristiche dei sistemi real time:
  - Capacità di rispondere in modo veloce e predicibile a eventi time-critical
  - Capacità di schedulare efficacemente le richieste per ottenere un alto livello di utilizzo dello risorse
  - Stabilità rispetto a momentanei overload
- Un processo real time è un processo che restituisce i risultati della computazione entro un determinato tempo

MM e Real Time

- Rispetto ai requisiti di real time, nel caso di multimedia:
  - C'è una certa tolleranza ai guasti e ai ritardi
  - Le operazioni critiche si ripetono con una periodicità predicibile (ESEMPIO: il sampling dell'audio)
  - Le richieste al sistema in alcuni casi possono essere modulate e ridotte (ESEMPIO: l'accesso alla scheda di rete può essere ridotto comprimendo)

SO Multimediale

- Rispetto ai sistemi operativi tradizionali:
  - Gestione Processi: deve tenere conto dei vincoli imposti dalle applicazioni multimediali tra cui:
    - ✓ real time
    - ✓ sincronizzazione
  - Riserva delle risorse: complicata dalla gestione dei media continui
  - Gestione della Memoria: deve garantire accesso efficiente ai dati multimediali
  - File system: deve offire spazi contigui ai dati multimediali

Stream

- I media continui hanno proprietà specifiche che influiscono sullo scheduling:
  - Richiesta periodica del processore
  - Deadline: necessità di essere processati entro un determinato istante
- Per fornire servizi multimediali il SO deve quindi incorporare:
  - Meccanismi di gestione della Qualità del Servizio (QoS)
  - Gestione delle priorità

Priorità

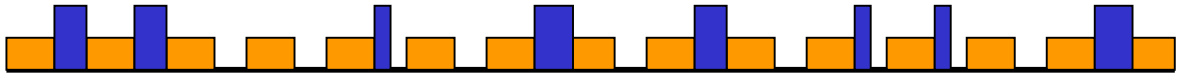
- Rispetto alla priorità si possono adottare due diverse strategie:
  - Preemptive Scheduling:
    - ✓ Il processo che sta usando la CPU (running) è rimpiazzato quando un processo con priorità più alta è pronto (ready)
    - ✓ Questo tipo di scheduling può provocare un overhead nel process switching
  - Non-preemptive Scheduling:
    - ✓ Processo con priorità più alta attende che il processo attualmente running rilasci la CPU
    - ✓ Può essere più efficace anche per i processi ad alta priorità se il time slice è piccolo perché non comporta process switching frequenti

Priorità

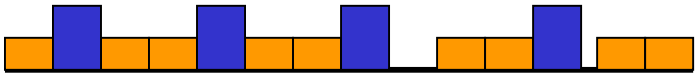
Processo 1 : Alta priorità

Processo 2 : Bassa priorità

Preemptive Scheduling



Non preemptive Scheduling



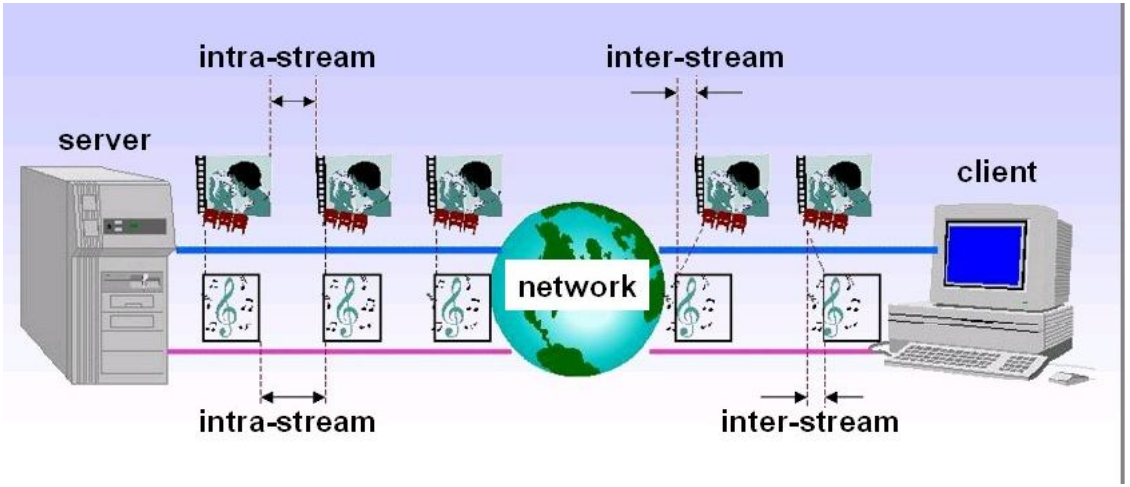
Sistemi per media continui

I sistemi tradizionali sono ottimizzati per

- accesso interattivo
- efficienza di memorizzazione
- gestione di trasmissione discontinua (*burst*)

Si devono gestire media con evoluzione temporale propria, quindi emerge il problema della sincronizzazione

- *intramedia*: mantenimento delle proprietà temporali di un medium
- *intermedia*: mantenimento delle relazioni temporali reciproche tra media diversi



Classificazione dei sistemi operativi

Sistemi operativi non real-time

- computazione senza errori

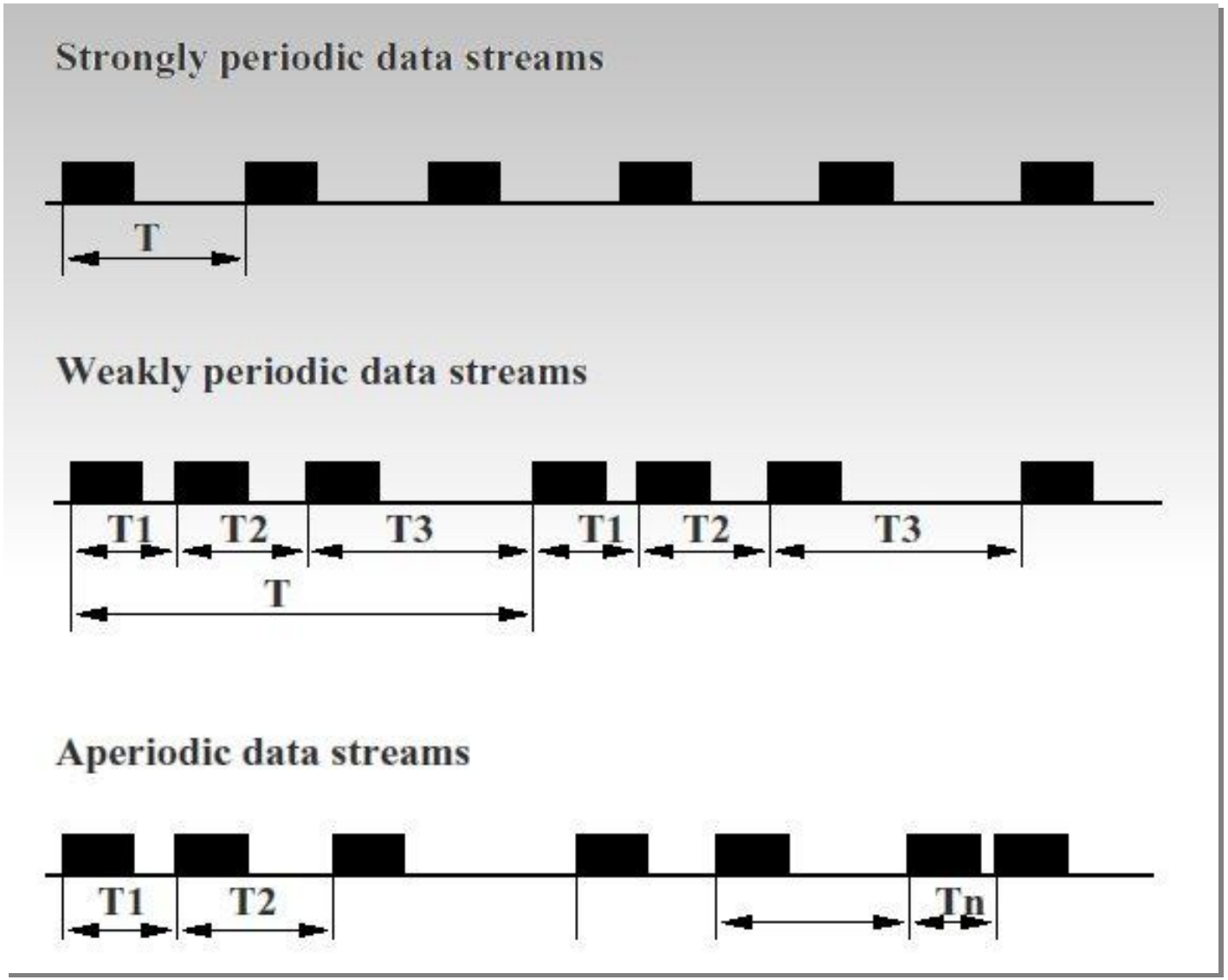
Sistemi operativi *hard* real-time

- computazione senza errori
- vincoli temporali stringenti (*catastrophic failure*)

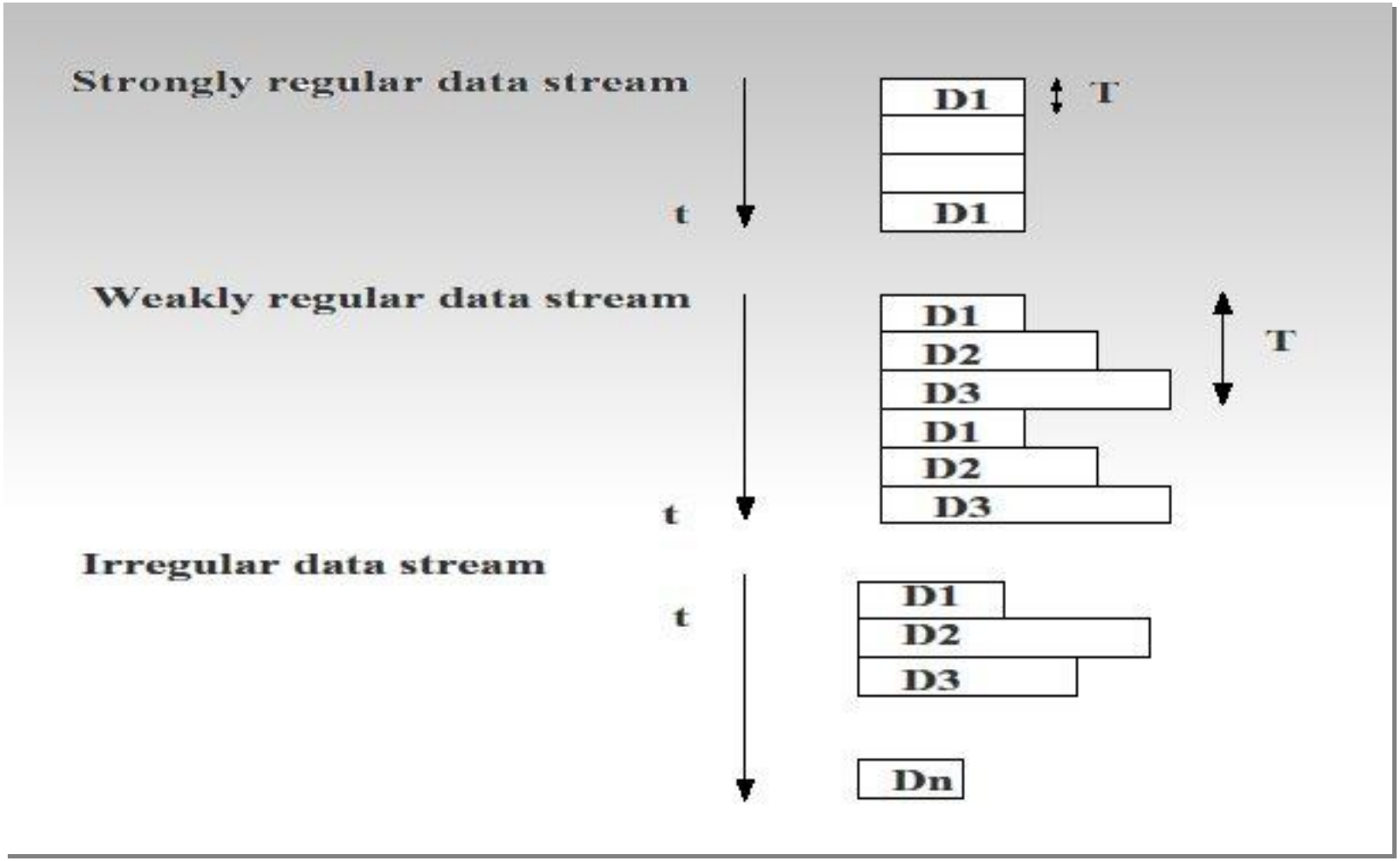
Sistemi operativi *soft* real-time (multimedia O.S.)

- vincoli temporali meno stringenti
- controllo di ammissione al servizio
- qualità di servizio (QoS, *Quality of Service*)

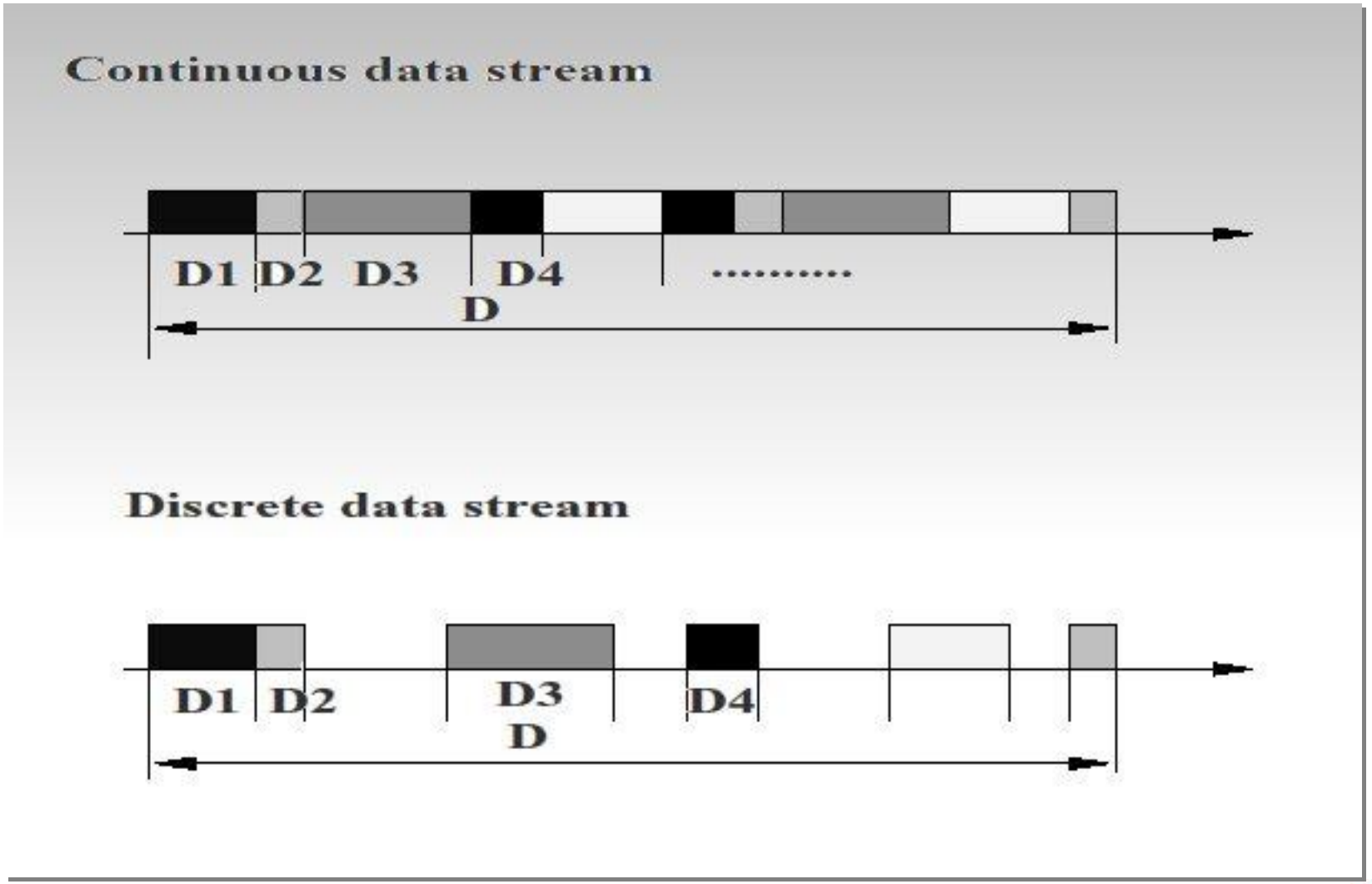
Proprietà degli stream di dati (1)



Proprietà degli stream di dati (2)



Proprietà degli stream di dati (3)



Sistemi operativi per media continui

Gestione risorse

- i dati multimediali competono per l'allocazione di componenti del sistema necessari alla loro elaborazione su un piano di continuità temporale piuttosto che a fronte di eventi asincroni

Gestione processi

- il rispetto di vincoli temporali ricorrenti richiede appropriate politiche di scheduling (le politiche real-time classiche non sono adeguate)

Gestione memoria

- i dati devono essere disponibili per l'elaborazione con ritardi massimi prevedibili e garantiti

Gestione file

- gli accessi ai dati memorizzati devono garantire il rispetto dei vincoli di tempo e l'efficacia rispetto alle dimensioni rilevanti

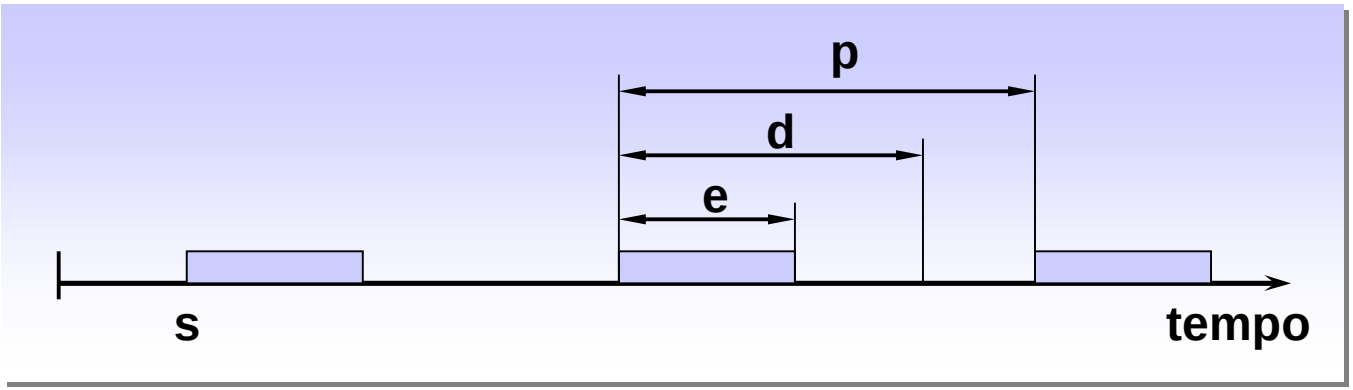
Proprietà dei dati temporizzati

		interval	
		predictable	unpredictable
delivery	guaranteed	nuclear plant control	user interface events
	not guaranteed	continuous media	shared drawing

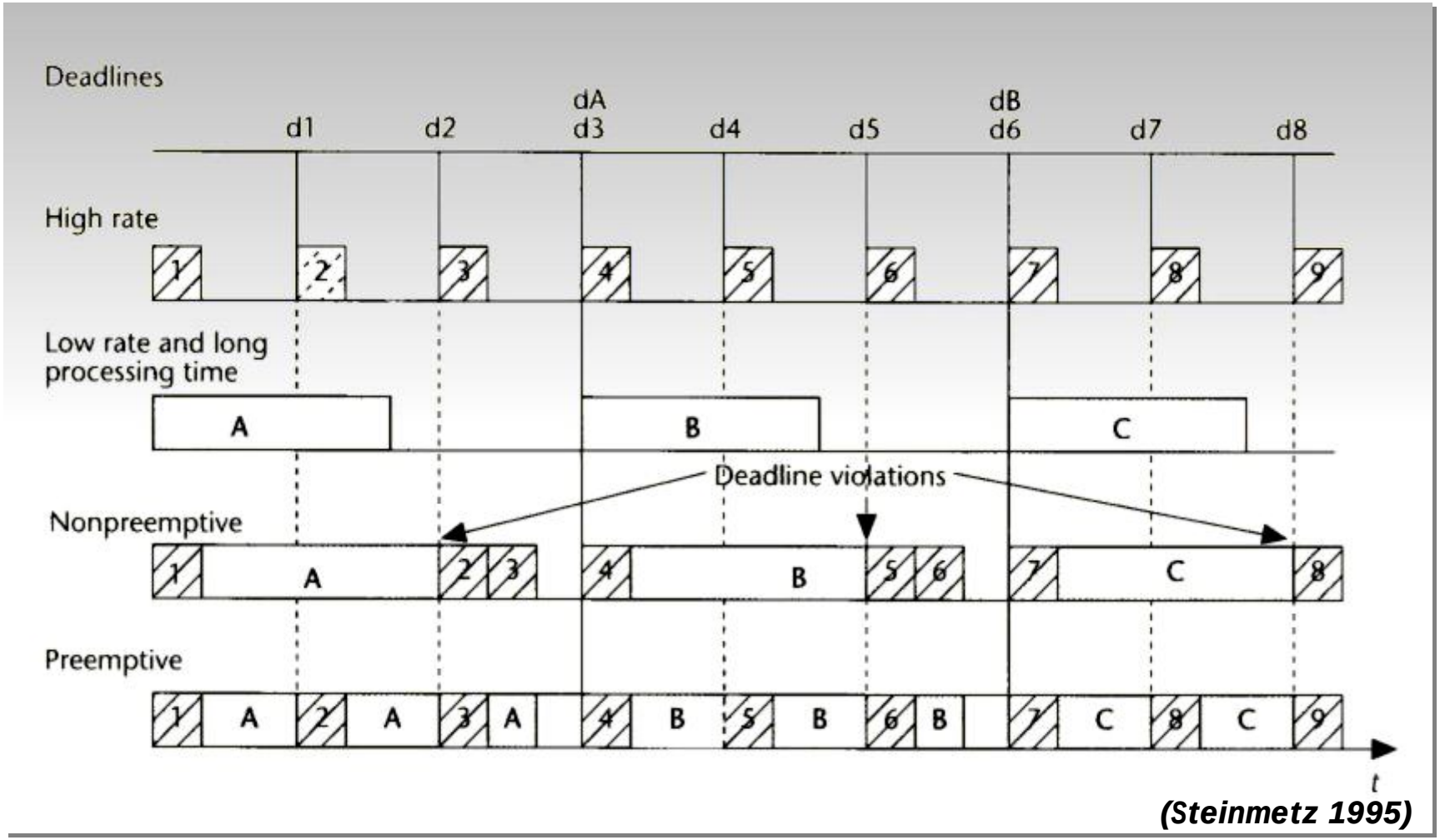
Scheduling in O.S. multimediali (1)

Un sistema operativo multimediale deve considerare i vincoli temporali di attività periodiche (*task*)

- s = tempo di inizio (start)
- e = tempo di elaborazione
- d = tempo in cui il risultato dell'attività deve essere disponibile (deadline)
- p = periodo di ripetizione dell'attività



Scheduling in O.S. multimediali (2)



(Steinmetz 1995)

Modello Semplice: Cyclic Executive

- L'applicazione consiste di un insieme fissato di processi periodici (ripetitivi) ed indipendenti con caratteristiche note
- Ciascun processo è suddiviso in una sequenza ordinata di procedure di durata massima nota
- L'ordinamento è costruito a tavolino come una sequenza di chiamate a procedure di processi fino al loro completamento
- Un ciclo detto maggiore (*major cycle*) racchiude l'invocazione di tutte le sequenze di tutti i processi
- Il ciclo maggiore è suddiviso in N cicli minori (*minor cycle*) di durata fissa che racchiude l'invocazione di specifiche sottosequenze

Esempio - *Modello semplice*

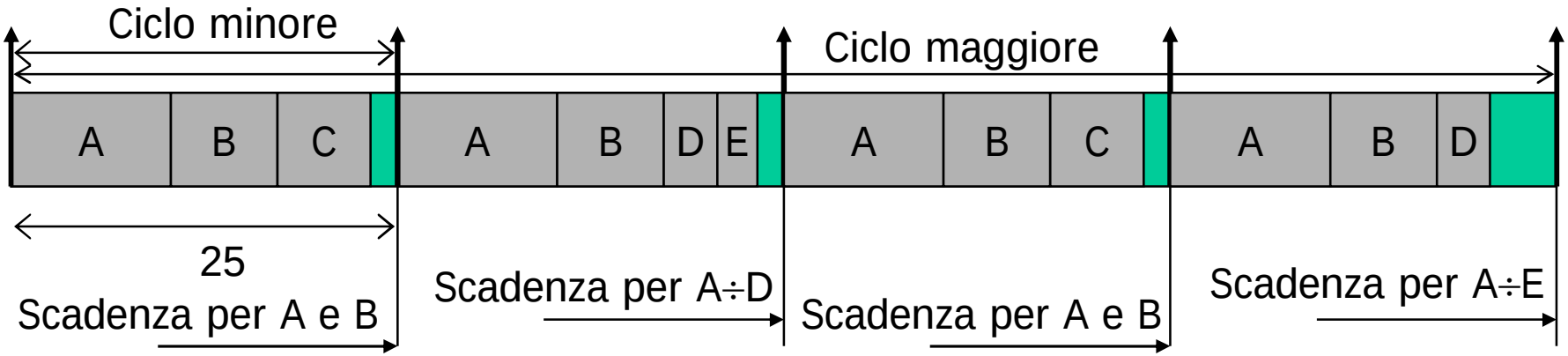
Processo	Periodo T	Durata C
A	25	10
B	25	8
C	50	5
D	50	4
E	100	2

$$U = \Sigma_i (C_i / T_i) = 46/50 = 0.92$$

Ciclo maggiore di durata 100 → MCM di tutti i periodi

Ciclo minore di durata 25 → periodo più breve

$$\Sigma_i (C_i / T_i) \leq 1 \text{ ???}$$



Rate Monotonic algorithm

Proposto da NASA, ESA per sistemi real-time

Gestisce task periodici indipendenti con scadenze e tempi di esecuzione costanti per ogni richiesta.

I task non periodici non hanno scadenza (*recovery*, *failure*)

Scheduling pre-emptive a priorità statica

- le priorità dei task sono assegnate in fase di inizializzazione, **i task più frequenti hanno priorità maggiore**
- è un algoritmo ottimo tra gli algoritmi con priorità statiche per il rispetto delle scadenze di scheduling

Svantaggi: non utilizza a pieno il processore

- efficienza media dell'80%, 69% nel caso pessimo
- richiede un numero maggiore di cambi di contesto (*context switch*)

Rate Monotonic: Ammissibilità

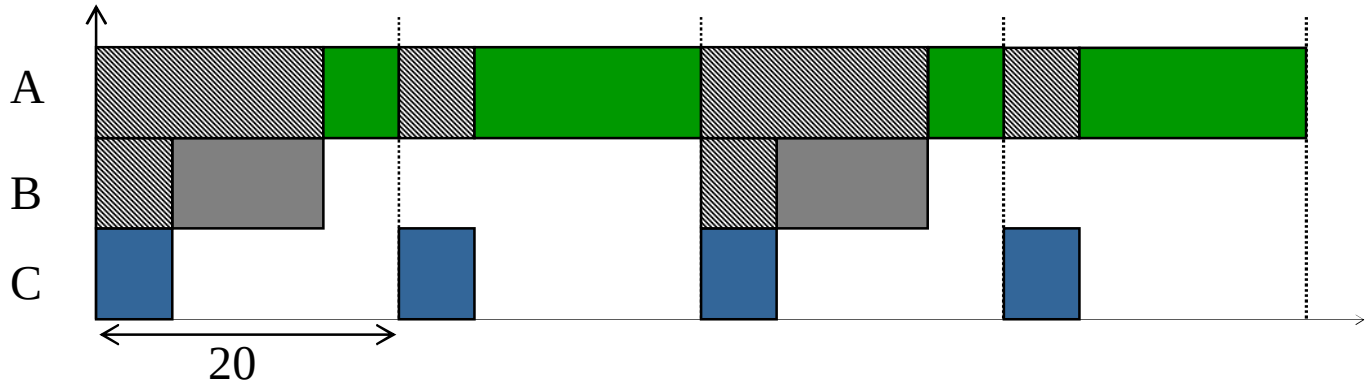
- Assegnazione di priorità secondo il periodo
  - ✓ Per scadenza uguale a periodo (D = T), priorità maggiore per periodo più breve
  - ✓ Test di ammissibilità sufficiente ma non necessario per *n* processi indipendenti (Liu & Layland, 1973)

$$U = \sum_{i:=1}^{n} \left(\frac{C_i}{T_i} \right) \leq f(n) = n(2^{1/n} - 1)$$

Esempio: Caso semplice ordinamento a priorità

Processo	Periodo T	Durata C	Priorità
A	80	40	1 ← Bassa
B	40	10	2
C	20	5	3 ← Alta

Il test di ammissibilità fallisce U = 1 > *f* (3) = 0,78
ma il sistema è ammissibile!



Earliest Deadline First algorithm

- Earliest Deadline First (EDF):
  - Scheduling Dinamico
  - Preemptive
  - Ogni volta che un nuovo stato entra tra i ready, lo scheduler seleziona (e mette in running) **il processo con deadline più vicina**
  - Produce uno scheduling che soddisfa tutte le deadline, qualora questo esista
- Una versione estesa (Time Driven Scheduler) gestisce le situazioni di overload cancellando i task con priorità più bassa tra quelli che non possono essere soddisfatti contemporaneamente

Earliest Deadline First algorithm

E' il miglior algoritmo conosciuto per scheduling real-time

Può gestire sia task periodici sia task con richieste arbitrarie e con scadenze variabili

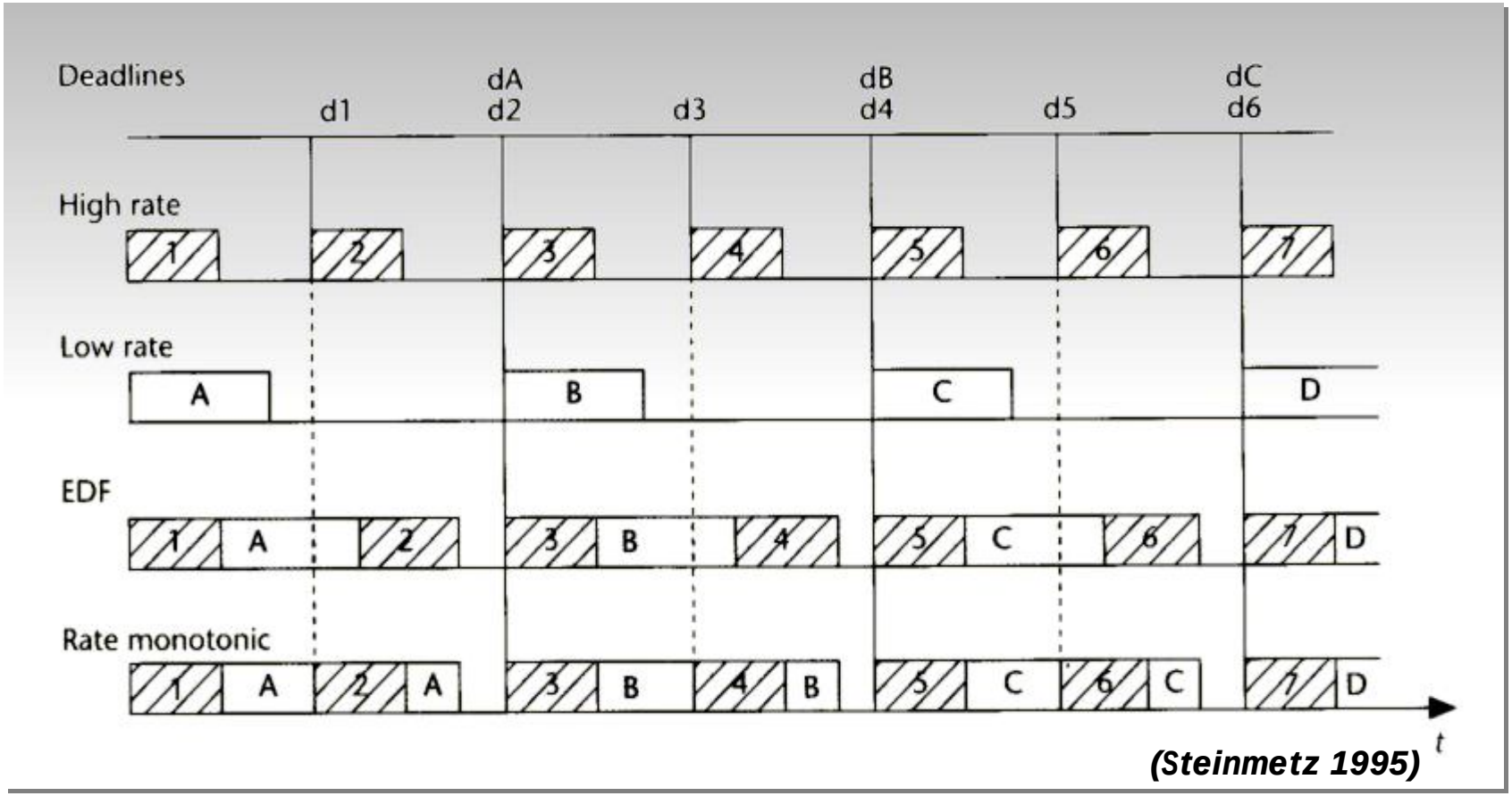
Scheduling pre-emptive a priorità dinamica

- le priorità sono modificate ad ogni periodo di scheduling
- può utilizzare completamente il processore (efficienza fino al 100%)
- è un algoritmo ottimo tra quelli con priorità dinamica

Svantaggi: le priorità dei processi sono modificate frequentemente

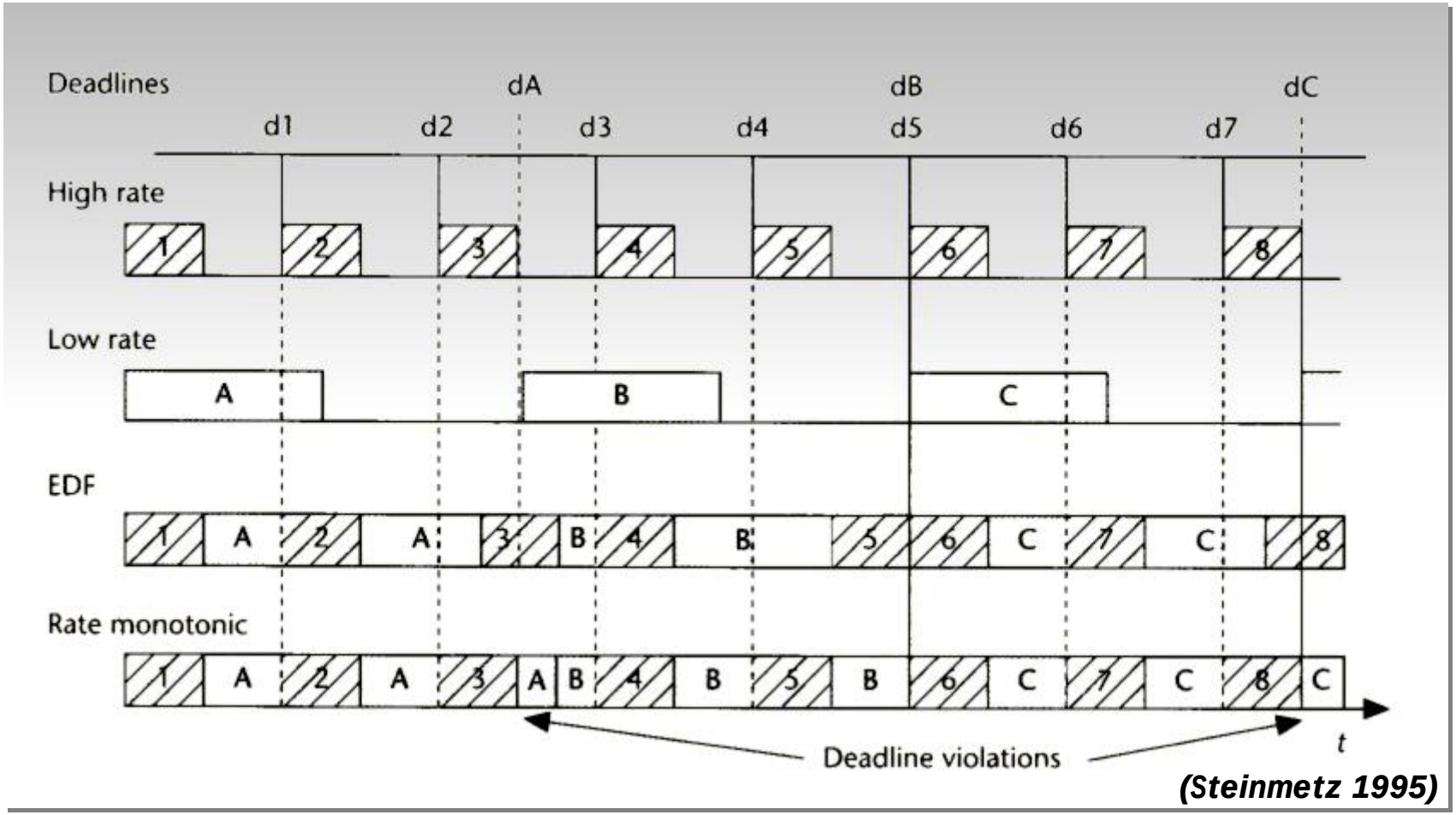
- nel caso peggiore tutte le priorità dei processi possono essere ricalcolate

EDF vs. RM (1)



(Steinmetz 1995)

EDF vs. RM (2)



(Steinmetz 1995)

Gestione memoria

L'occorrenza di numerosi page fault durante l'elaborazione di dati temporizzati può influenzare negativamente le prestazioni real-time

Può essere necessario bloccare un set di pagine in memoria

- svantaggio: le prestazioni complessive diminuiscono
- può non essere possibile con alcuni sistemi, es. IBM AIX (UNIX) non permette di bloccare più del 70% della memoria fisica

Molte operazioni sono di semplice copia senza reale elaborazione dei dati

- si possono copiare puntatori invece di contenuti

File system (1)

Un file system tradizionale può non garantire l'accesso continuo ai dati

- problemi hardware: latenza, tempo di accesso, caching
- problemi software: layout, buffering, concorrenza, scheduling

I problemi software possono essere risolti a livello di progettazione del S.O.

- *layout*: posizionamento dei blocchi di dati sui dischi
- *buffering*: requisiti minimi per stream / stream massimi per dimensionamento
- *concorrenza*: ammissibilità di ulteriori client per un file server
- *scheduling*: ordinamento delle richieste in funzione dei loro vincoli real-time

File system (2)

Obiettivo: assicurare la corretta riproduzione di uno o più stream sincronizzati

- evitare il blocco dei client
- minimizzare il ritardo iniziale
- minimizzare la dimensione dei buffer

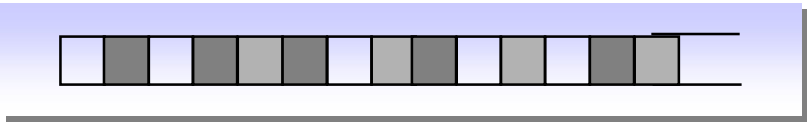
La gestione di stream multipli pone ulteriori problemi

- può riguardare uno o più file (es. video on demand)
- richiede una politica di scheduling dell'accesso ai dischi che rispetti le deadline di ogni richiesta
- presuppone la negoziazione di ulteriori richieste di connessione in base alla qualità del servizio
- può beneficiare di tecniche efficienti di allocazione dello spazio su disco

Metodi di allocazione (1)

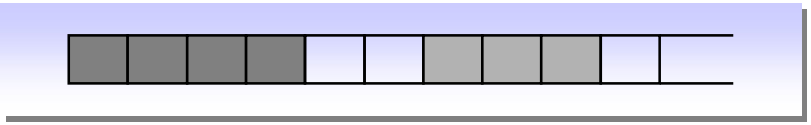
Allocazione non contigua (standard in file system tradizionali)

- minimizza il tempo medio di accesso
- evita la frammentazione del disco
- non garantisce tempi massimi di accesso e di latenza



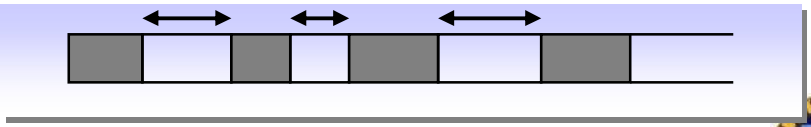
Allocazione contigua

- richiede una sola operazione di *seek*, poi l'accesso è continuo
- richiede un index file di dimensioni ridotte (minore overhead)
- modifiche al file molto costose
- genera frammentazione



Allocazione interallacciata a distanza vincolata (*constrained interleaved placement*)

- pone un limite alla distanza media tra qualunque sequenza di blocchi appartenenti allo stesso file
- limita la lunghezza di seek



Metodi di allocazione (2)

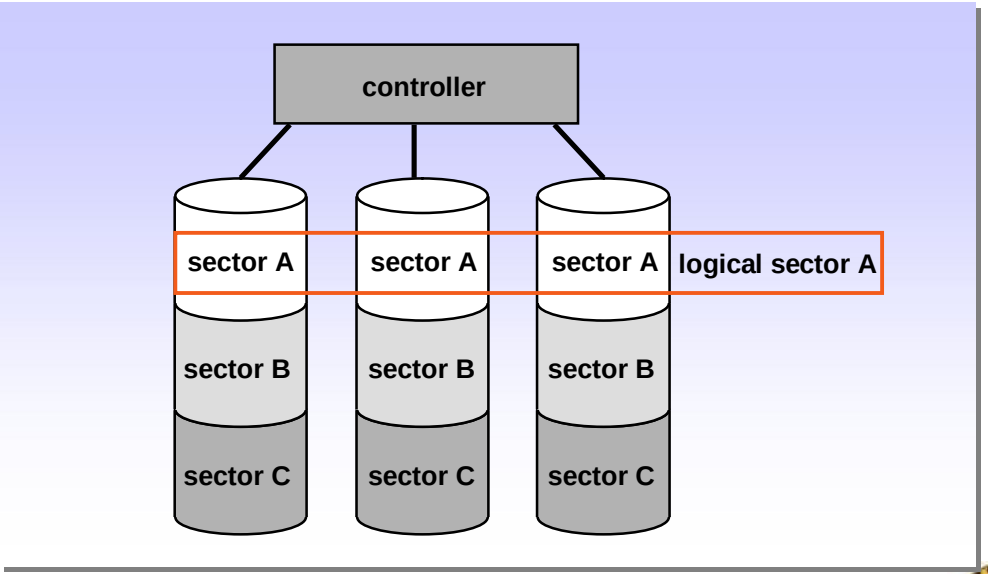
Le tecniche di *data striping* accelerano il trasferimento dei dati dividendo un settore logico su più dischi fisici

Soluzioni hardware

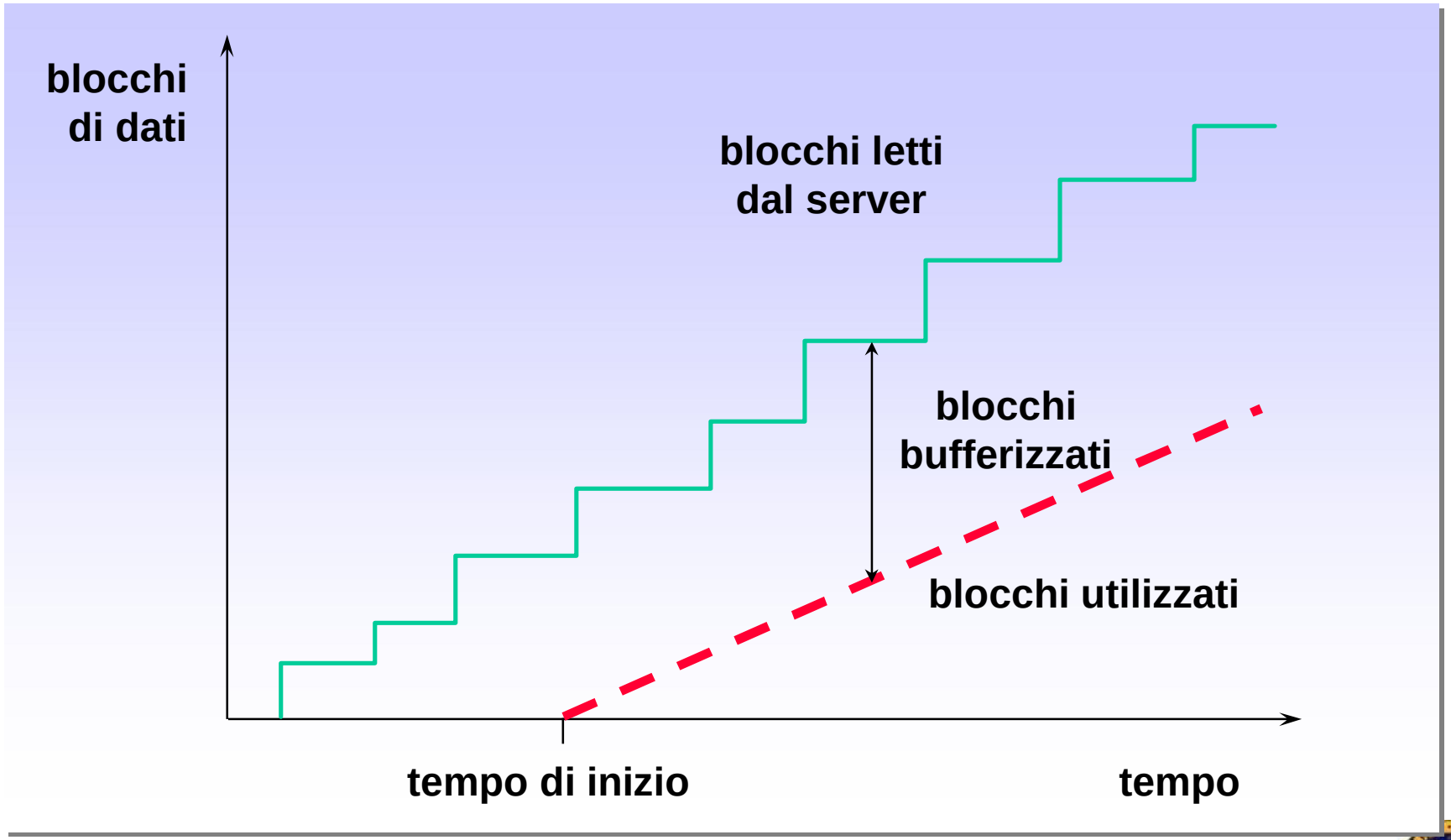
- accesso parallelo a un array di dischi sincronizzati meccanicamente
- il *transfer-rate* aumenta
- il tempo di seek e la latenza di rotazione non diminuiscono

Soluzioni software

- blocchi consecutivi di file sono memorizzati su dischi diversi
- i dischi sono meccanicamente indipendenti (RAID)

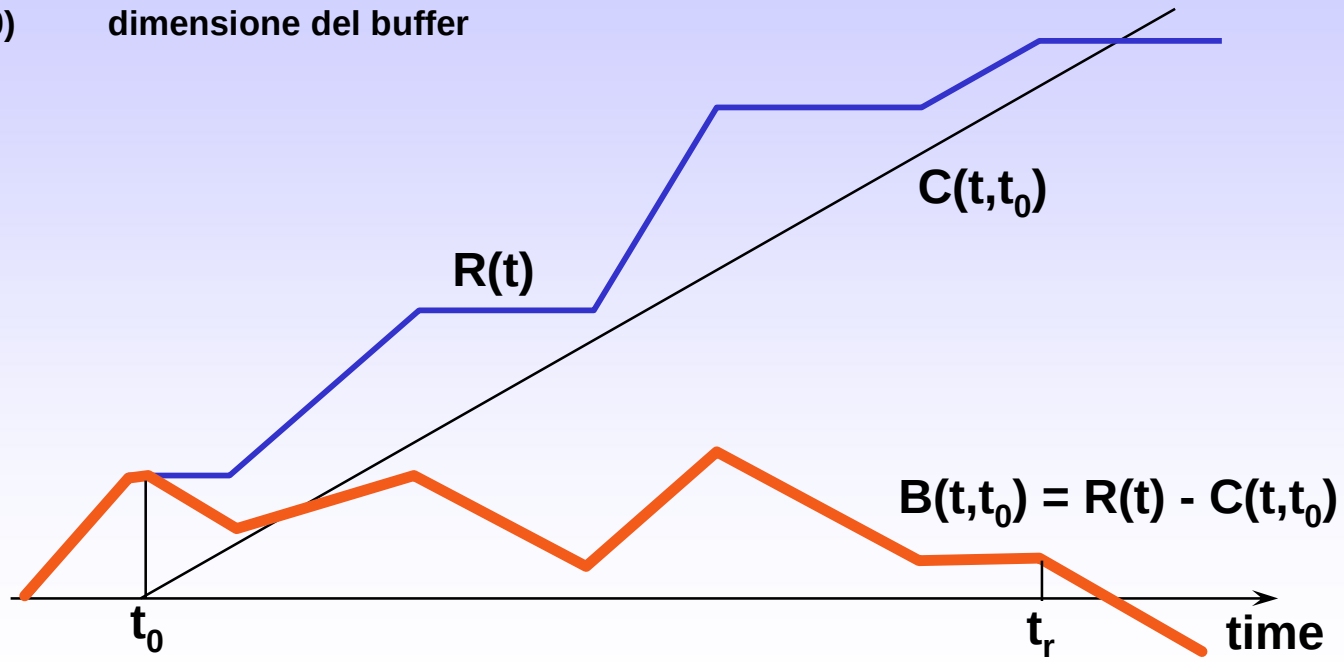


Gestione dei buffer (1)



Gestione dei buffer (2)

R(t)	n. di blocchi di dati letti da disco
C(t,t0)	n. di blocchi di dati necessari all'applicazione
B(t,t0)	dimensione del buffer

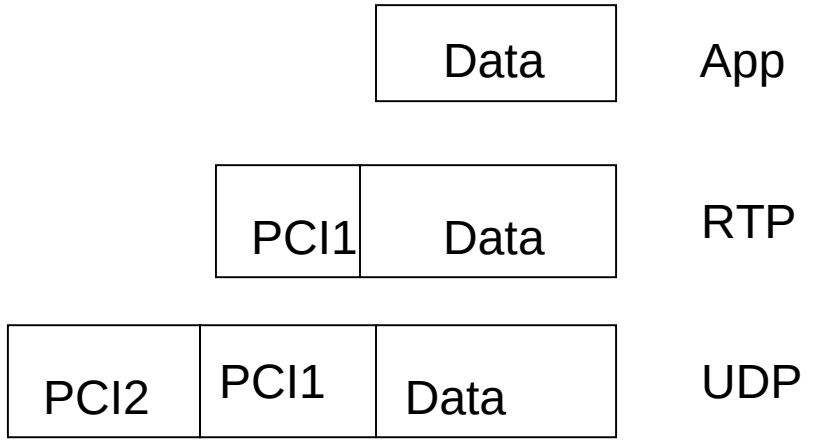


Buffer Management

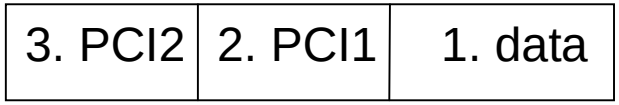
- Alcune problematiche nella gestione del buffer:
  - Virtual Memory vs. Real Memory: per le applicazioni multimediali è essenziale operare sulla memoria reale e non su sue estensioni su disco
  - Preferibili comunicazioni dirette adapter-adapter (evitare passaggi dalla memoria)
  - A volte inevitabile, allora copiare una **volta sola** in memoria
  - Per velocizzare, usare shared memory (tra user e kernel space)
  - Usare threads con upcalls per protocol processing di uno stream (si evitano context switch tra kernel e user mode)
  - Buffer Management
    - ✓ Data Copying
    - ✓ Offset Management
    - ✓ Scatter/Gather Scheme

Buffer Management Techniques

DATA COPYING



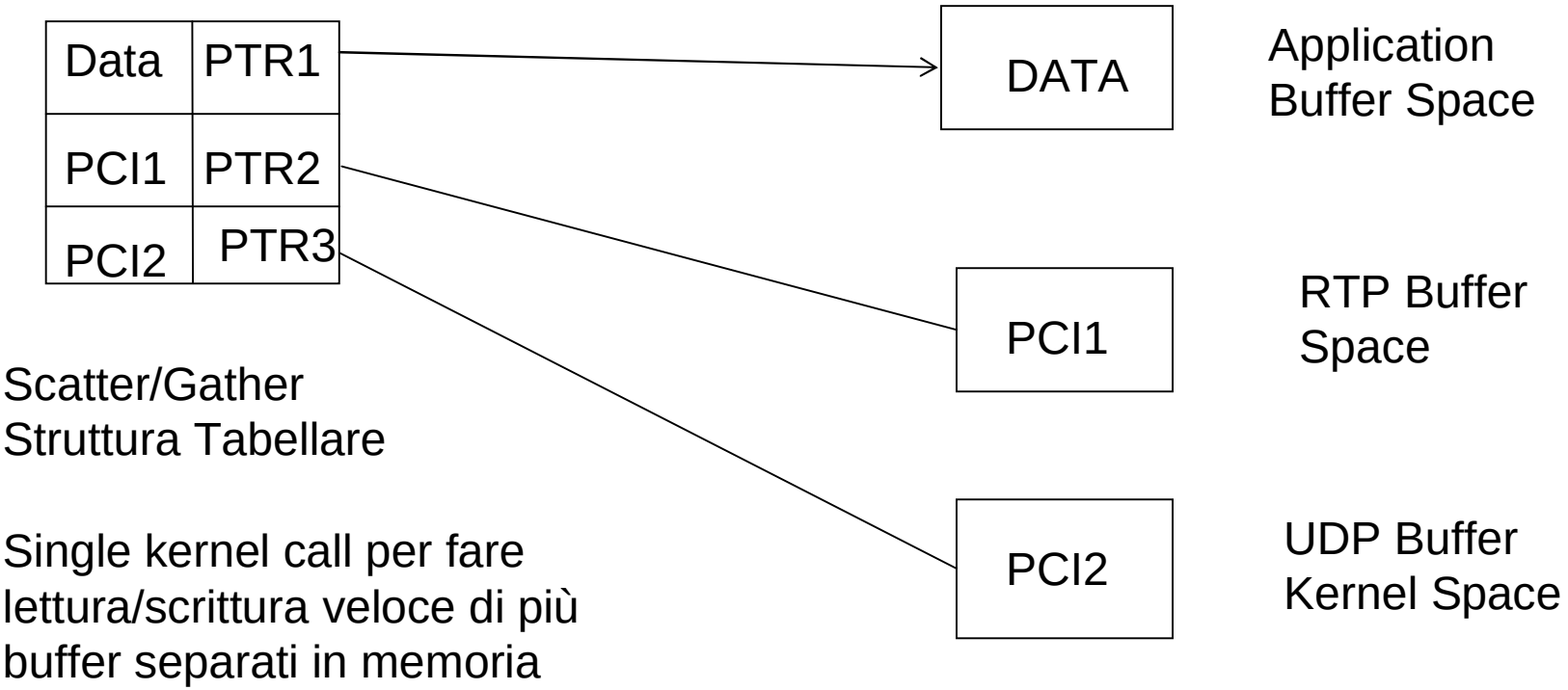
OFFSET MANAGMENT



1. app
2. PCI1 added by RTP
3. PCI2 added by UDP

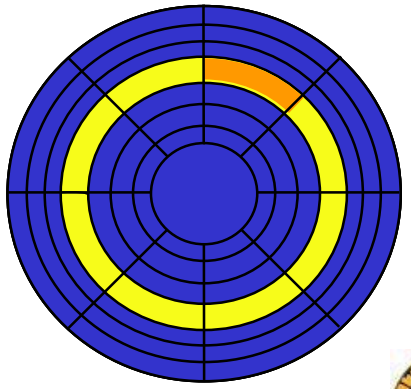
Buffer Management: Assegnare buffer larghi quanto dati + headers dei protocolli

Buffer Management Techniques (Scatter-Gather)



Disk Scheduling

- Altri limiti dei File System tradizionali riguardano i meccanismi di scheduling del disco
- Il tempo di lettura/scritura del file è composto da :
  - seek time (per trovare la traccia)
  - rotational latency (per trovare il blocco nella traccia)
  - transfer time



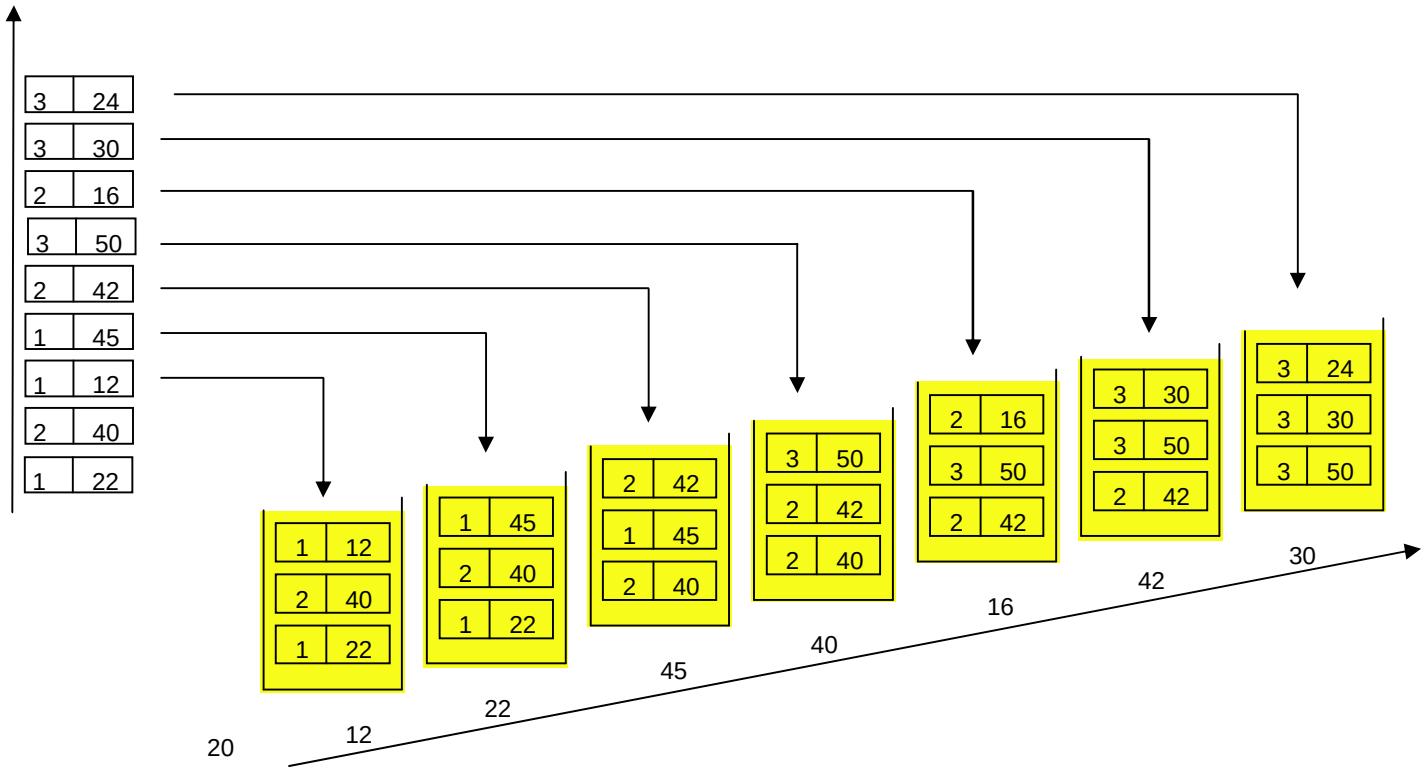
Disk Scheduling

- I sistemi di scheduling del disco tradizionali mirano tutto al più a minimizzare il tempo di latenza:
  - First Come First Serve
  - Shortest Seek Time First
  - SCAN
  - C-SCAN
- Nei sistemi multimediali occorre considerare che anche in fase di scrittura il processo ha una deadline da rispettare (ESEMPIO: nel playout di un video)

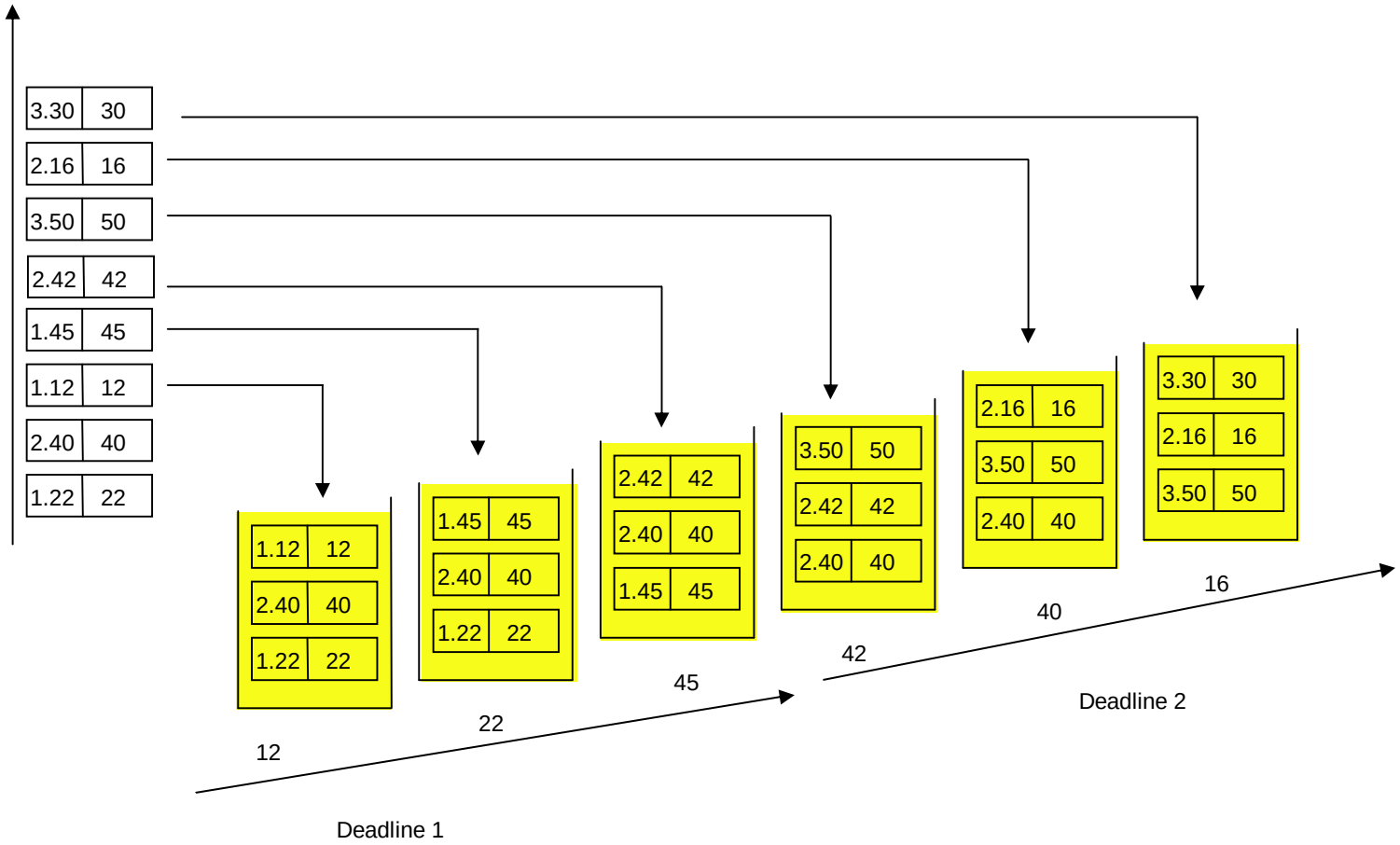
MM Disk scheduling

- EDF: come per lo scheduling della CPU
  - Non è molto adatto perchè riduce poco il tempo di seek
  - L'algoritmo è preemptive quindi la lettura/scrittura è spesso interrotta e c'è molto overhead
- SCAN EDF: combina SCAN con EDF:
  - Usa EDF per definire che la prima richiesta servita sarà quella con deadline più vicina
  - Tra richieste con la stessa deadline usa SCAN per scegliere la direzione

EDF Scheduling Algorithm



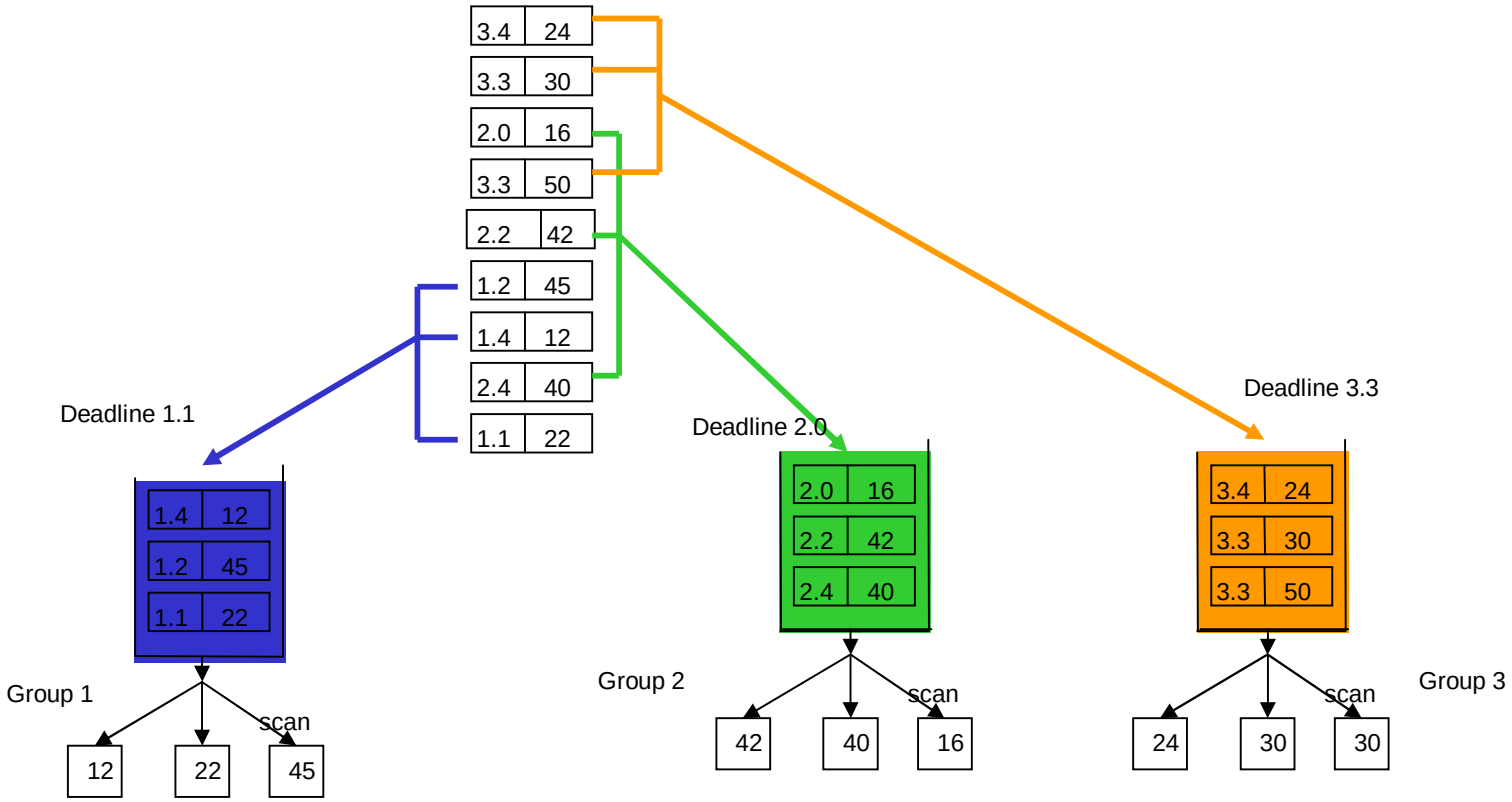
SCAN-EDF


3.30 30
2.16 16
3.50 50
2.42 42
1.45 45
1.12 12
2.40 40
1.22 22
1.12 12
2.40 40
1.22 22
1.45 45
2.40 40
1.22 22
2.42 42
2.40 40
1.45 45
3.50 50
2.42 42
2.40 40
2.16 16
3.50 50
2.40 40
3.30 30
2.16 16
3.50 50
12
22
45
42
40
16
Deadline 1
Deadline 2

Group Sweeping

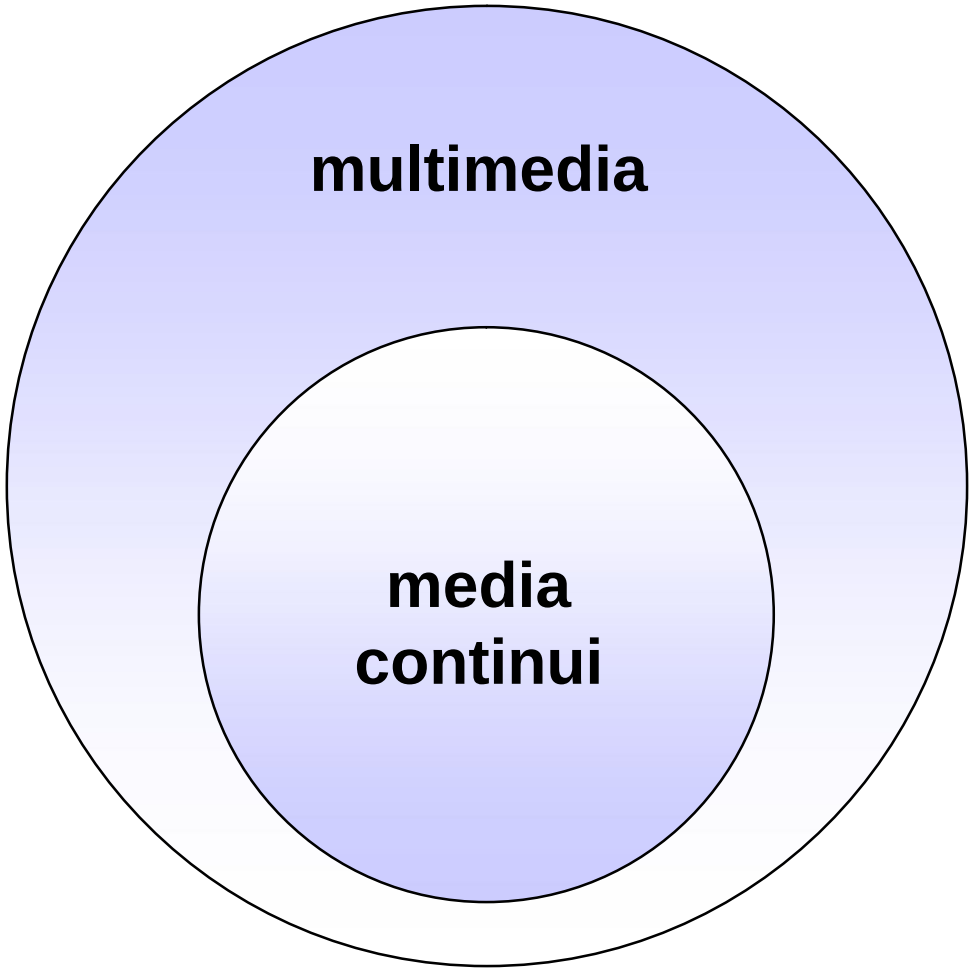
- Group Sweeping Scheduling (GSS):
  - Esamina le richieste ciclicamente (come un sistema round robin)
  - Per ridurre i movimenti del braccio del disco, l'insieme degli *n* stream è diviso in *g* gruppi con deadline vicine
  - All'interno del gruppo è applicato un algoritmo di scan
  - Può richiedere buffer di supporto per assicurare continuità ai flussi

Group Sweeping



Multimedia vs. media continui

I media continui coinvolgono relazioni temporali tra l'emittente e il destinatario di una trasmissione multimediale



Dipendenze temporali nei dati multimediali

Dipendenze *naturali* (o implicite): sono presenti nel dato al momento dell'acquisizione

- suono, filmato

Dipendenze *sintetiche*: sono introdotte al momento della presentazione, e possono riguardare sia dati statici sia dati dinamici

- slide show
- montaggio di scene di film

Dati persistenti vs. dati transienti

Dati *persistenti*: esistono indipendentemente dalla loro riproduzione

- immagini memorizzate
- video, audio memorizzato
- algoritmi di animazione

Dati *transienti*: sono generati dinamicamente ed eliminati quando obsoleti

- immagini catturate in real-time
- video real-time

Proprietà dei media continui (1)

I flussi di dati (*stream*) sono formati dai valori codificati dei media continui, variabili nel tempo

- campioni (audio), fotogrammi (video)
- ogni unità di dati deve essere presentata all'utente con una specifica scadenza temporale (*entro* un tempo determinato, ma anche *non prima di* un tempo determinato)

Real-time / interattività

- le relazioni temporali tra emittente e ricevente sono più stringenti che in una trasmissione unidirezionale
- es. il ritardo in una conversazione audio dovrebbe essere < 40 ms (accettabile fino a 150 ms)

Riproduzione

- il ritardo accettabile è limitato soprattutto dal dimensionamento dei buffer di ricezione e dalle reazioni dell'utente

Proprietà dei media continui (2)

Trasmissione *asincrona*

- comunicazione senza restrizioni temporali

Trasmissione *sincrona*

- esiste un ritardo massimo *end-to-end* per ogni pacchetto di uno *stream*

Trasmissione *isocrona*

- esiste un ritardo massimo e un ritardo minimo *end-to-end* per ogni pacchetto di uno *stream* (= la frequenza di trasmissione è costante entro il limite di tolleranza)

Gestione risorse

Una risorsa è un'entità di sistema che i processi richiedono per elaborare i dati multimediali

- riguarda sia componenti interni sia quelli esterni (rete)

La gestione risorse deve permettere di fornire servizi con vincoli di tempo e di qualità

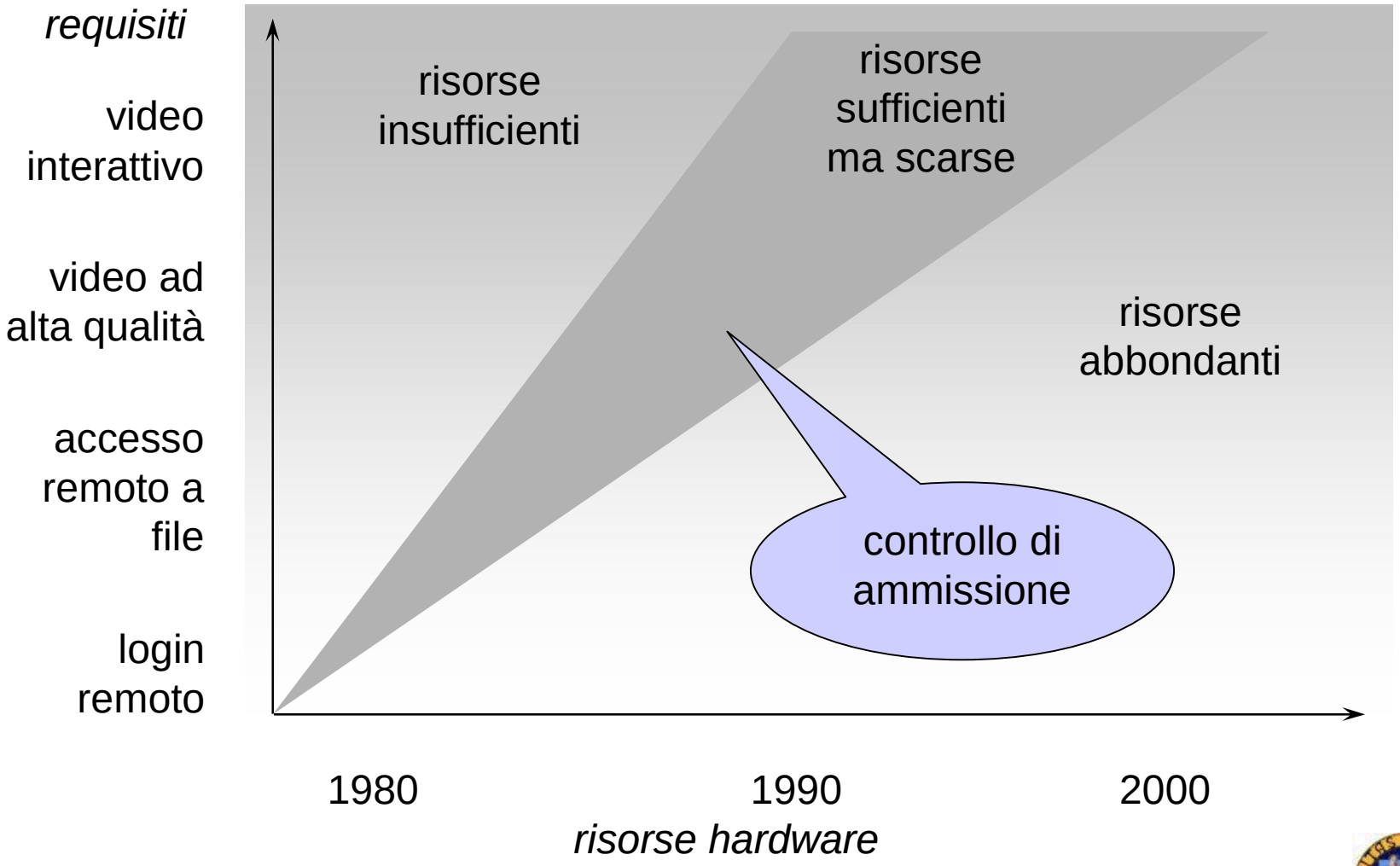
Le risorse devono essere allocate con un processo di negoziazione che si basa su

- disponibilità delle risorse
- priorità delle richieste
- livelli di qualità

Sono necessarie politiche di allocazione basate sul controllo di ammissione delle richieste verso le risorse

- garantito (approccio *worst case*)
- statistico (approccio *best case*)

Requisiti vs. risorse



Gestione della sincronizzazione

La sincronizzazione è un problema noto nei sistemi concorrenti

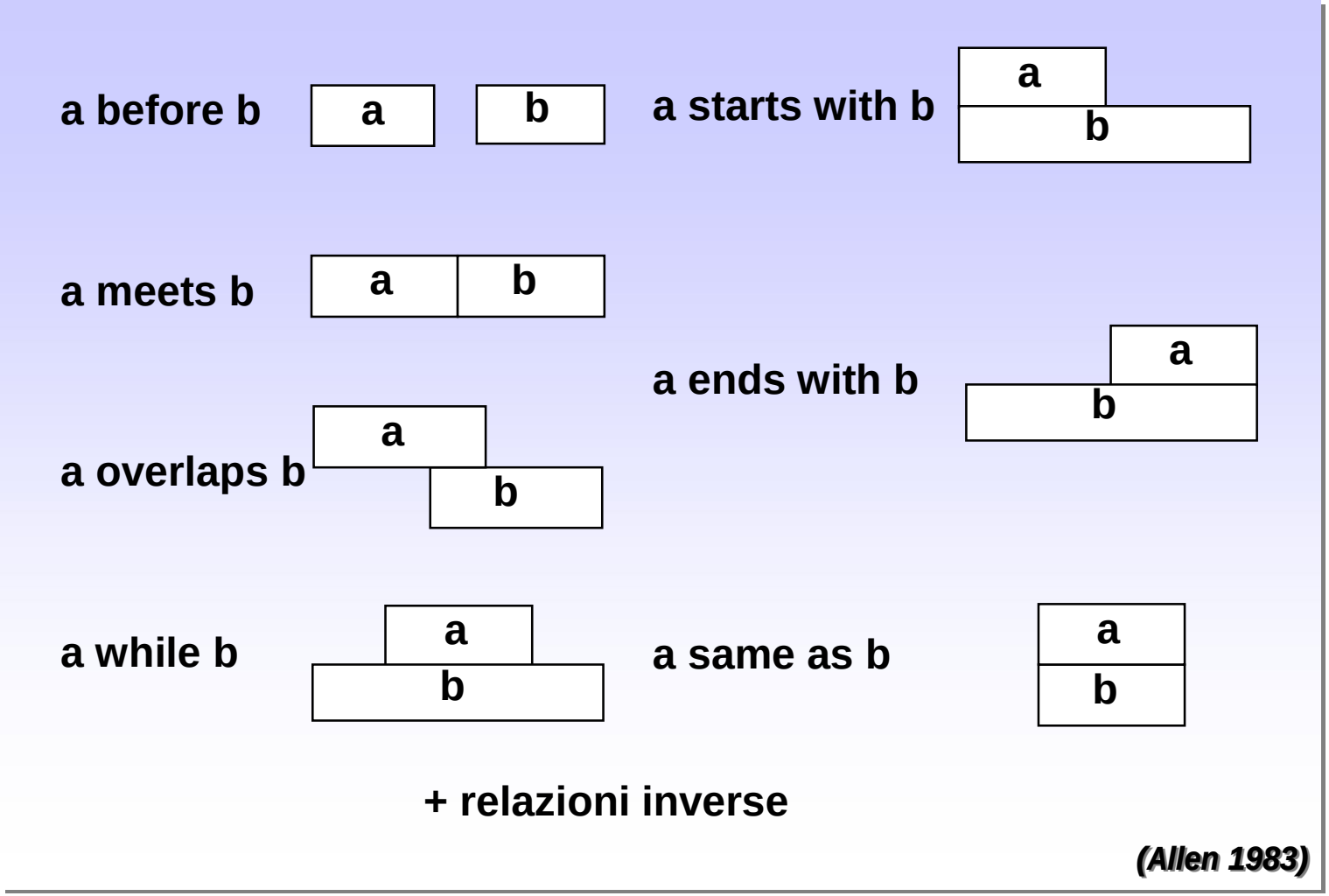
- la correttezza può essere garantita con modelli di sincronizzazione appropriati e opportune tecniche implementative

I media continui presentano problemi differenti

- la sincronizzazione coinvolge eventi real-time e attività prevedibili e ripetitive
- la presentazione di un elemento soffre di ritardi, dovuti a reperimento, generazione, trasmissione, elaborazione, entro un massimo *Dmax*
- una presentazione prevista al tempo *Tp* deve essere quindi iniziata al tempo *Tp - Dmax*
- ritardo medio < ritardo massimo: gli elementi devono essere bufferizzati
- gli errori di accesso o trasmissione possono non essere correggibili nel tempo a disposizione

Le variazioni nei ritardi e le percentuali di errori accettabili dipendono dall'applicazione e dallo specifico elemento mediale

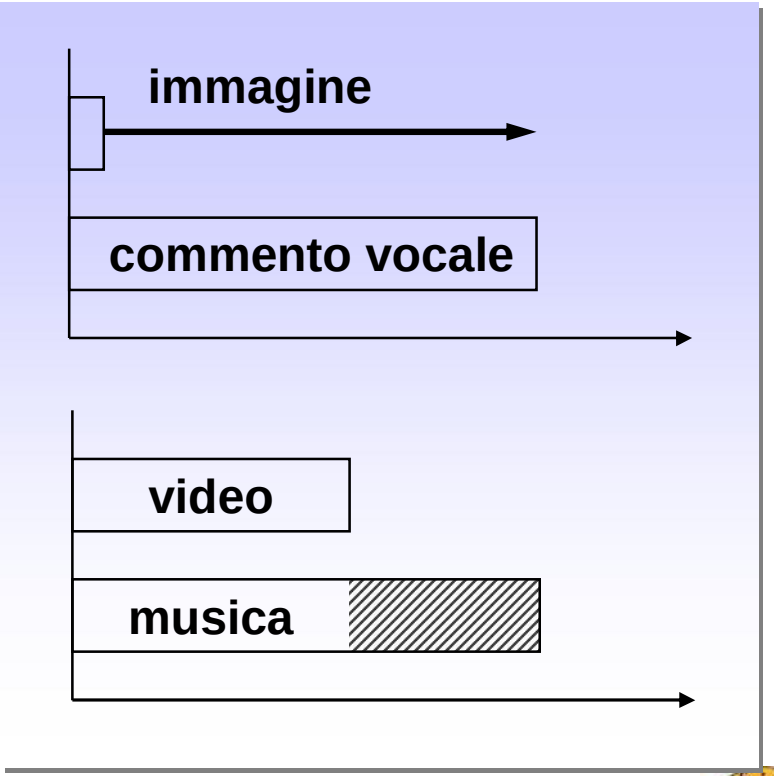
Relazioni temporali



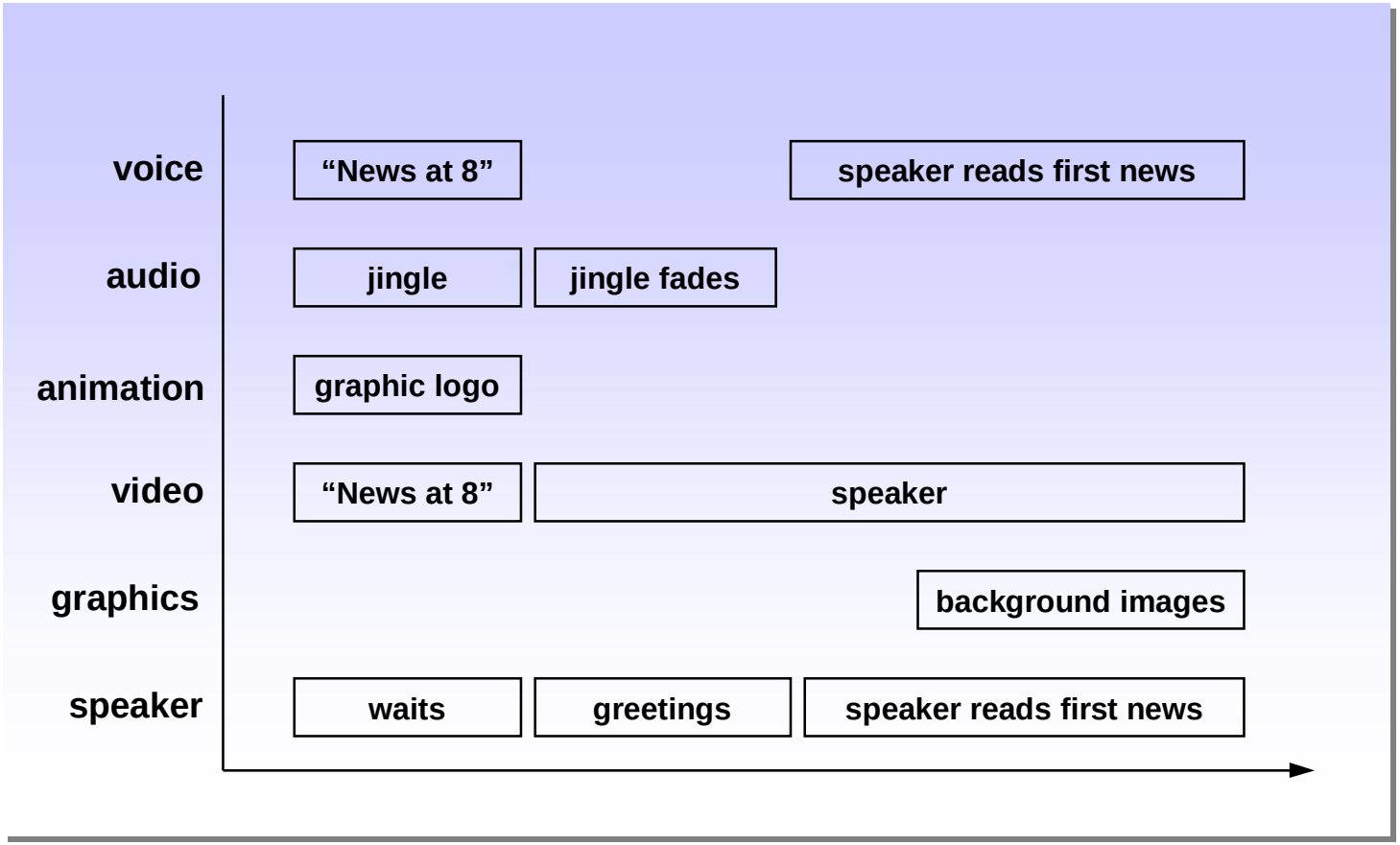
Specifica incompleta del tempo

In alcuni casi si introducono specifiche temporali incomplete che richiedono ad alcuni media di “adattarsi” alle richieste temporali di altri

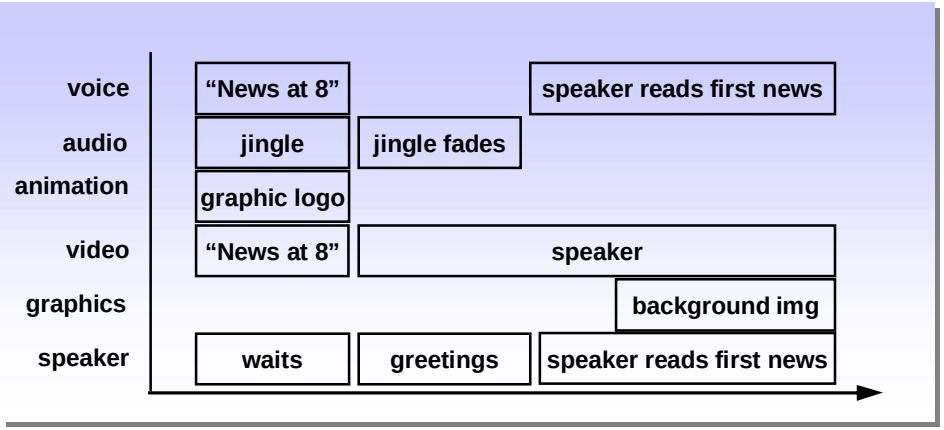
- immagine statica con commento vocale: l'immagine ha una durata pari alla lunghezza del commento
- video con musica di sottofondo: la musica cessa alla fine del video

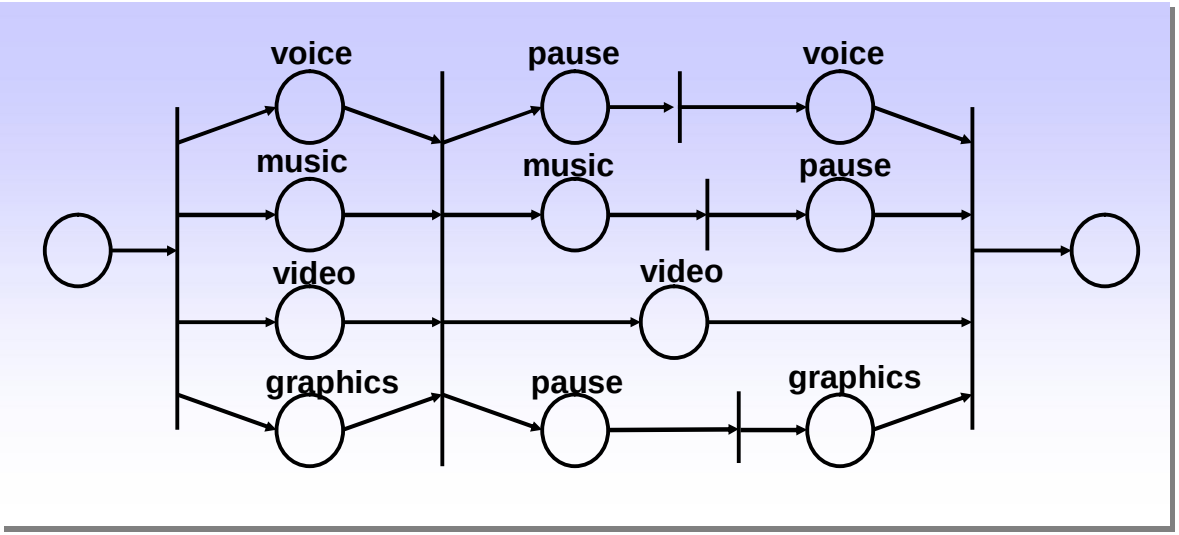


Specifica della sincronizzazione (1)

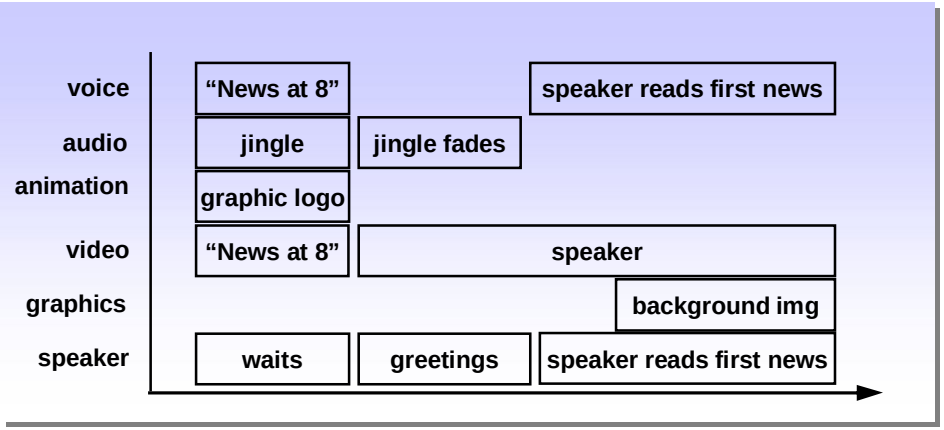


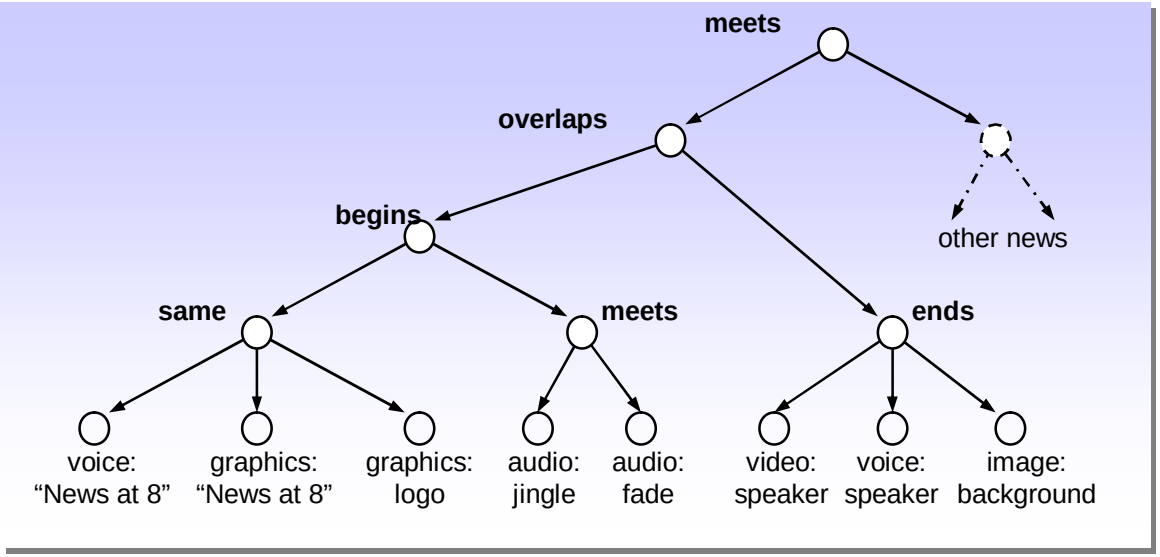
Specifica della sincronizzazione (2)





Specifica della sincronizzazione (3)





Qualità di servizio (QoS)

Il termine *QoS* (*Quality of Service*) è usato per descrivere i requisiti di una applicazione verso una certa risorsa, ad esempio:

- max/min risoluzione (video), fedeltà (audio)
- ritardi
- errori

Alcune applicazioni richiedono livelli minimi garantiti, altre richiedono qualità in termini statistici

- es. elaborazione remota di immagini mediche vs. video entertainment