Principi di progettazione di sistemi distribuiti

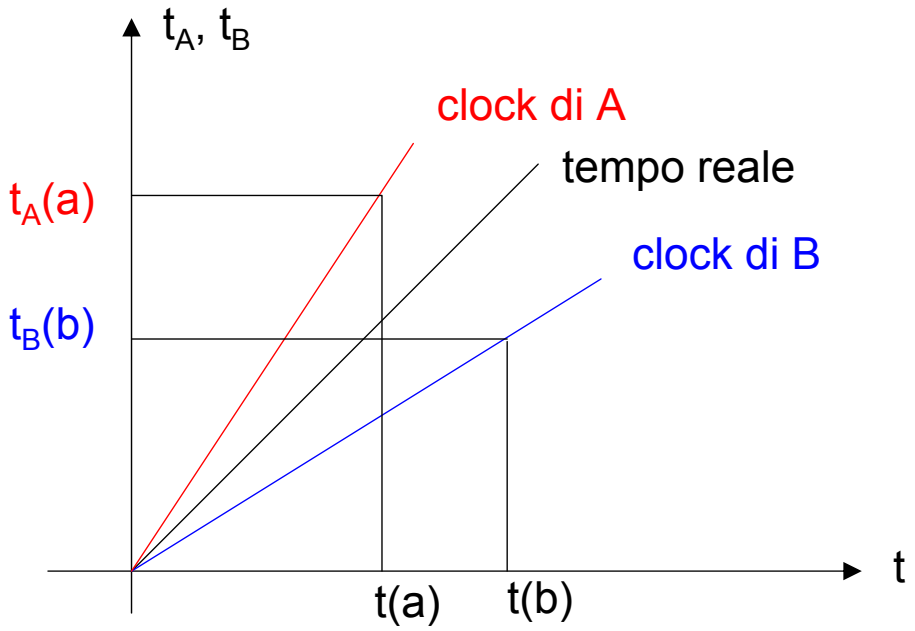
- ✓ i modelli sincrono ed asincrono
- ✓ il problema dell'agreement
- ✓ i guasti e loro rilevazione

Sistema distribuito sincrono

Un sistema distribuito viene detto **sincrono** quando **è possibile** stabilire sia un limite inferiore sia un **limite superiore** a

- **il tempo di esecuzione** di ciascun passo di un processo
- **il tempo di propagazione** di un messaggio in rete
- **la velocita di deriva** di un clock

Drift dei clock



Sistema distribuito sincrono

Un sistema distribuito **sincrono** può essere implementato purché

- **si individuino** le esigenze di ciascun processo in termini di risorse di calcolo e di comunicazione
- **si garantiscano** tali risorse a ciascun processo
- **si fornisca** a ciascun processo un clock con una velocità di deriva limitata

Sistema distribuito asincrono

Un sistema distribuito viene detto **asincrono** quando **non è possibile** stabilire un **limite superiore** a

- **il tempo di esecuzione** di ciascun passo di un processo
- **il tempo di propagazione** di un messaggio in rete
- **la velocita di deriva** di un clock

I modelli a confronto

- In un sistema sincrono certi problemi (coordinamento tra processi, rilevamento di un guasto) possono essere risolti “perfettamente”
- In un sistema asincrono certi problemi (coordinamento tra processi, rilevamento di un guasto) non possono essere risolti “perfettamente”
- Un algoritmo valido per il modello asincrono è valido anche per il modello sincrono

Il problema dell'agreement

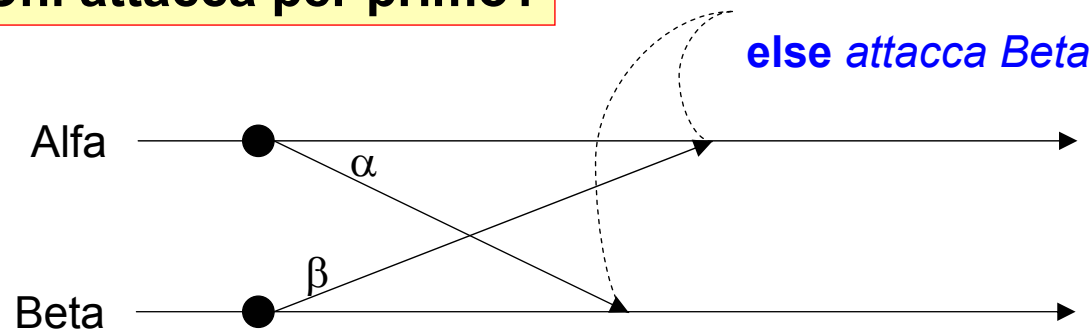
- Le divisioni Alfa e Beta dell'esercito Bianco sono accampate su due colline vicine
- La valle tra le due colline è in mano all'esercito Nero
- Le divisioni comunicano per mezzo di messaggeri che attraversano la valle (*comunicazione affidabile*)
- L'esercito Bianco vuole sferrare un attacco e le divisioni devono trovare un accordo su:
  - chi deve attaccare per prima (la divisione con più uomini)
  - quando sferrare l'attacco (la seconda divisione deve intervenire entro *D* unità di tempo dalla prima)

Il problema dell'agreement

Chi attacca per primo?

if ($\alpha > \beta$) *attacca Alfa*

else *attacca Beta*



α = numero di uomini di Alfa

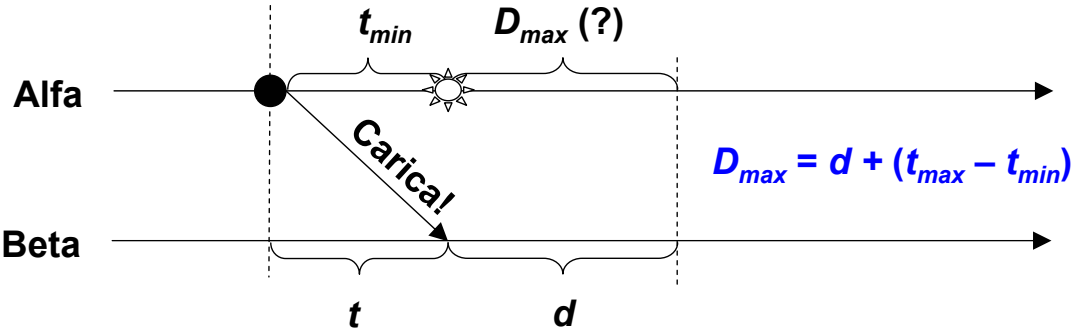
β = numero di uomini di Beta

L'algoritmo funziona correttamente in un sistema asincrono

Il problema dell'agreement

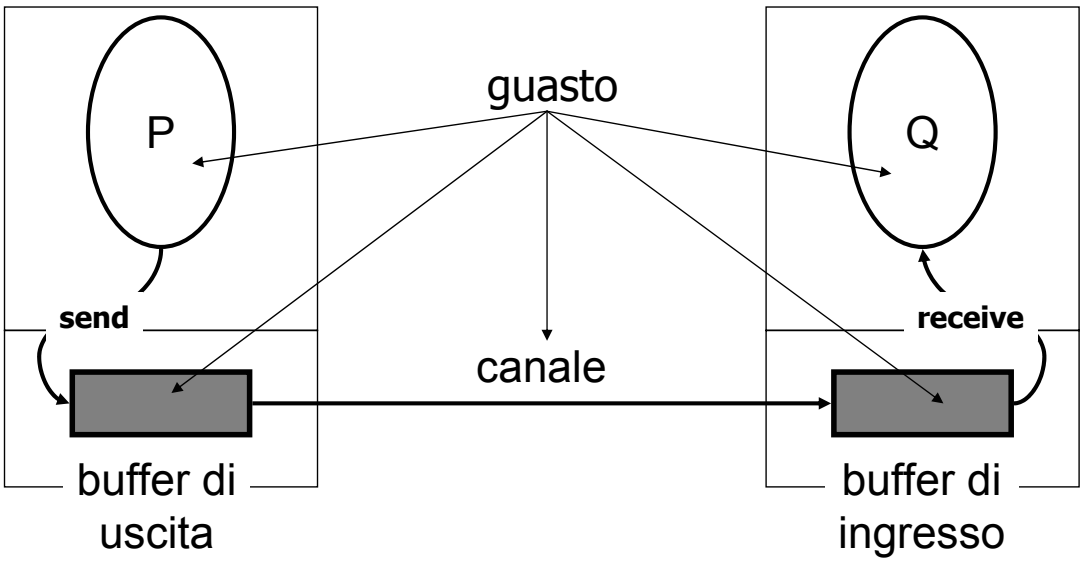
Quando attaccare?

- Alfa invia il messaggio, aspetta t_{min} e poi attacca
- Beta riceve il messaggio, si prepara per d e poi attacca
- L'attacco ha successo se Beta attacca entro D_{max} rispetto ad Alfa



In un **sistema asincrono** D_{max} può essere arbitrariamente lungo perché t_{max} (e d) non hanno estremo superiore

Guasti



Sia i processi sia il sistema di comunicazione possono guastarsi, possono cioè esibire un comportamento che si discosta dalle specifiche

Modello di guasto

- **OMISSION FAILURES**

  un processo o un canale non esegue le azioni previste dalla sua specifica

- **BYZANTINE FAILURES**

  un processo o un canale si discostano dalla specifica in modo arbitrario e (possibilmente) malizioso

- **TIMING FAILURES**

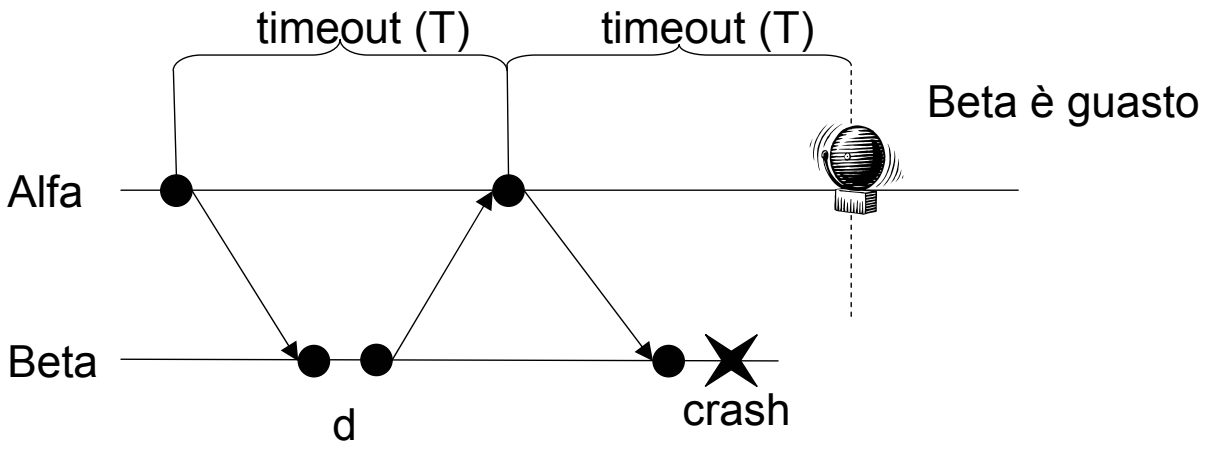
  si considerano nei sistemi sincroni quando i limiti sui tempi di esecuzione, di comunicazione (performance failure) e sulla velocità di deriva dei clock (clock failure) sono violati

Omission failures

- **PROCESS OMISSION FAILURES**
  - ✓ **Crash**: un processo sospende la sua esecuzione e non esegue più alcuna azione
  - ✓ **Fail-stop**: un processo sospende la sua esecuzione, non esegue più alcuna azione e gli altri processi possono rilevare lo stato del processo (!!!)
- **CHANNEL OMISSION FAILURES**
  - ✓ **Send omission failures**: un processo esegue la **send** ma il messaggio non è inserito nel buffer di uscita
  - ✓ **Receive omission failures**: un messaggio è posto nel buffer di ingresso di un processo ma questi non lo riceve mai
  - ✓ **Omission failure**: un messaggio è inserito nel buffer di uscita del mittente ma non raggiunge mai il buffer di ingresso del ricevente

Timeout & failure detection

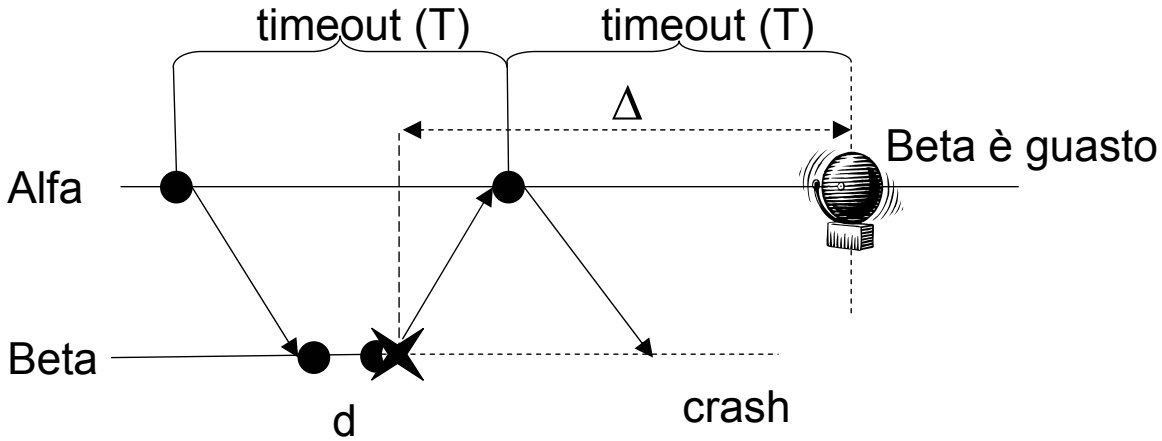
Sistema sincrono



$$T = R + d \leq 2 \times t_{max} + d$$

Timeout & failure detection

Sistema sincrono: tempo massimo di rilevazione di un crash in caso di comunicazione affidabile

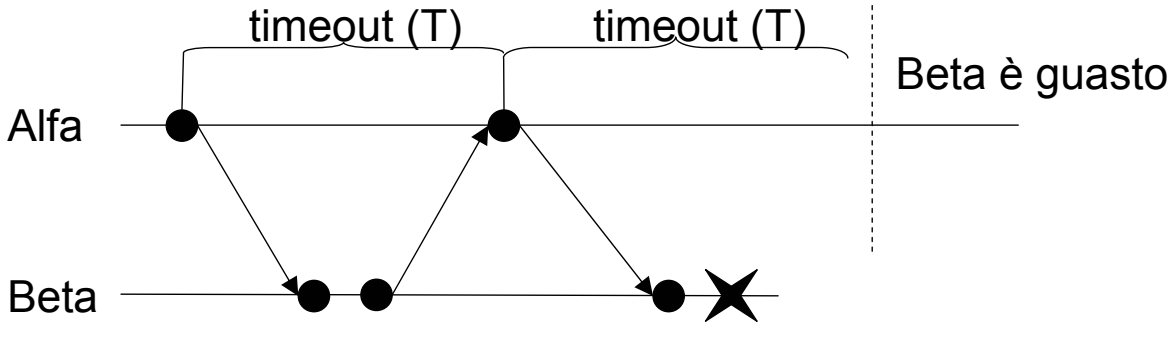


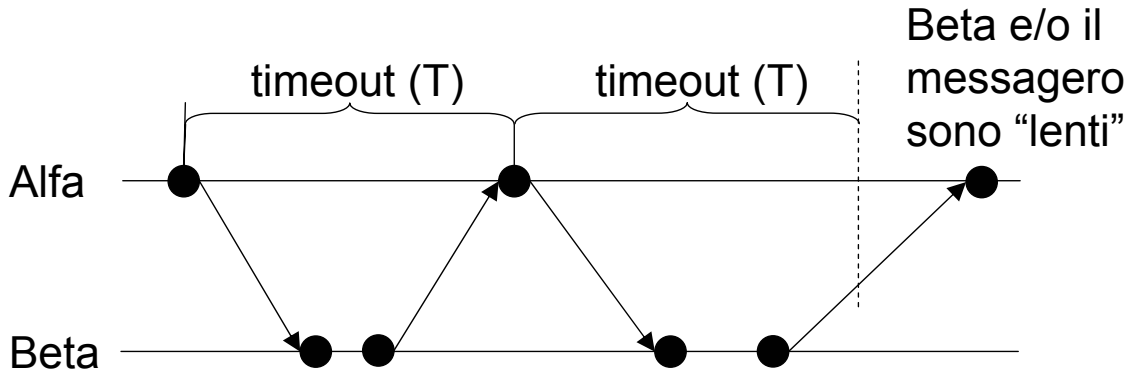
Il processo Beta risponde e “crasha” subito dopo

$\Delta = t_{max} + T = 3 \times t_{max} + d$

Timeout & failure detection

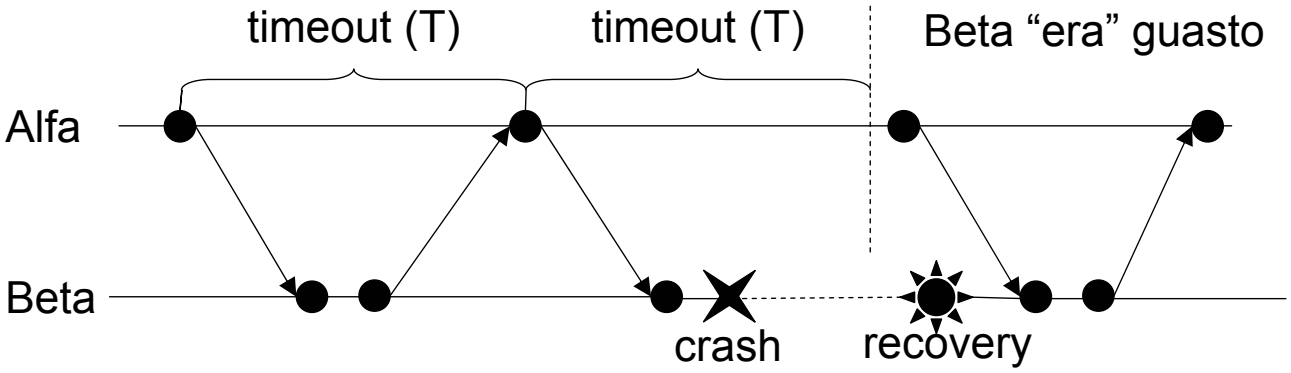
Sistema asincrono





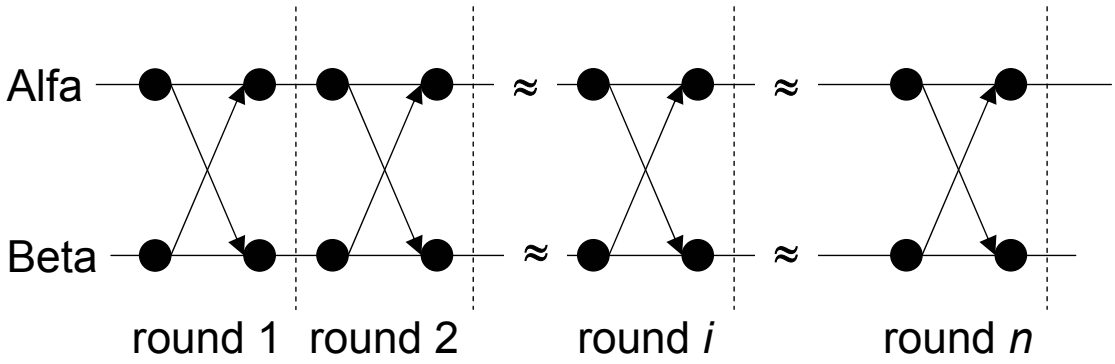
Timeout & failure detection

Sistema asincrono



Il processo Beta inserisce nel messaggio di risposta l’istante dell’ultimo recovery

Agreement in presenza di perdita di messaggi



- I processi devono accordarsi sul valore di una variabile
- I processi adottano un protocollo $\mathcal{P}$ costituito da *n* round
- Se il sistema di comunicazione è soggetto ad omission failure, l'agreement non è possibile