Sincronizzazione

SCD

Anno accademico 2009/10
Sistemi Concorrenti e Distribuiti

Tullio Vardanega, tullio.vardanega@math.unipd.it

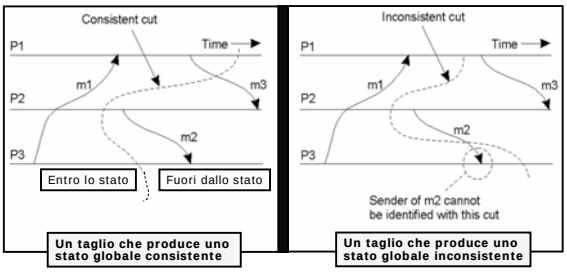
Stato del sistema – 1

- **Stato globale di un sistema distribuito**
  - **Lo stato locale di ciascun processo**
    - Non necessariamente tutto
      - Solo ciò che è importante ai fini dello stato globale
  - **L'insieme dei messaggi in transito**
- **Conoscere lo stato globale consente di**
  - **Verificare se il sistema è globalmente attivo oppure no**
    - Nessun messaggio in transito → nessuna attività globale
  - **Se no, quali cause: normale terminazione oppure stallo?**

Stato del sistema – 2

- **Istantanea distribuita (*distributed snapshot*)**
  - **Riflette uno stato globale consistente come potrebbe essere stato nel recente passato**
    - Esempio di **stato inconsistente**: P ha ricevuto un messaggio da Q il cui invio non risulta dallo stato globale
  - **Rappresenta un "taglio" (*cut*) nell'evoluzione temporale individuale dei processi del sistema**
    - Fissa ciò che appartiene allo stato globale e ciò che ne è escluso
    - Il "percorso" del taglio (causato dall'algoritmo usato) determina la consistenza dello stato
  - **Ideato da K. Chandy e L. Lamport**
    - **Distributed Snapshots: Determining Global States of Distributed Systems**. *ACM Transactions on Computer Systems*, 3(1):63-75, 1985

Stato del sistema – 3



Stato del sistema – 4

- **Situazioni pericolose**
  - **Messaggi inconsistenti**
    - Inviati dal processo M **dopo** aver salvato il proprio stato (***checkpoint***) ma ricevuti dal processo D **prima** di aver salvato il proprio stato
    - L'invio di un messaggio inconsistente rischia di essere duplicato in caso di ripristino dello stato
    - Se il suo effetto non è **idempotente** lo stato di D ne viene corrotto!
  - **Messaggi in transito (*in-flight*)**
    - Inviati da M **prima** di aver salvato il proprio stato
    - Ricevuti da D **dopo** aver salvato il proprio stato
- **Uno stato consistente non ammette messaggi inconsistenti**

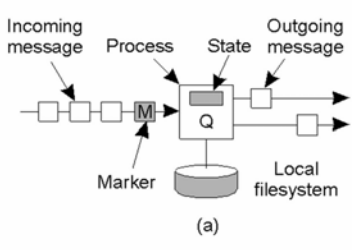
Stato del sistema – 4

- **Algoritmo della istantanea distribuita**
  - **Sistema visto come un insieme di processi connessi da canali di comunicazione punto a punto unidirezionali**
    - Concettualmente, un "overlay network" sulla topologia fisica
  - **Qualunque processo può iniziare la cattura dello stato**
    - Di conseguenza **più istantanee** possono trovarsi in corso nello stesso istante
  - **Il processo iniziatore salva il proprio stato e invia un *marker* su ogni suo canale in uscita richiedendo al destinatario di partecipare alla cattura dello stato**
    - Il *marker* identifica il processo iniziatore e la versione (p.es. progressiva) dell'istantanea

Stato del sistema – 5

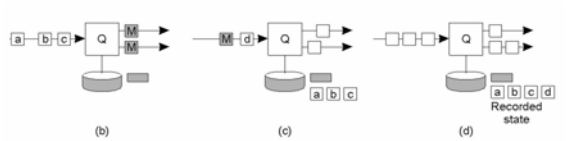
- **Istantanea distribuita**
  - **Il processo che riceve un *marker* da un suo canale di ingresso C**
    - Se non ha ancora salvato il suo stato locale lo salva e invia il *marker* su tutti i propri canali in uscita
    - Se già lo ha salvato inizia a salvare lo **stato del canale** C
      - L'insieme dei messaggi ricevuti su C a partire dall'ultimo salvataggio del sistema
  - **Un processo ha completato la sua parte dell'algoritmo quando abbia trattato tutti i *marker* (di inizio istantanea) ricevuti da tutti i suoi canali in ingresso**
  - **Quando tutti i processi coinvolti dall'iniziatore hanno completato l'iniziatore può raccogliere lo stato globale**

Stato del sistema – 6



Il processo Q riceve il *marker* M dal suo canale in ingresso ...

Stato del sistema – 7



... Salva il proprio stato locale e invia il *marker* su ciascuno dei suoi canali in uscita ...

... e comincia a salvare anche lo stato del suo canale in ingresso ...

... All'arrivo del successivo *marker* sul suo canale in ingresso ha completato il suo contributo alla cattura dello stato globale

Questo algoritmo produce stati consistenti?

Esempio d'uso: terminazione – 1

- **Il processo Q che ha ricevuto un *marker* per la prima volta ne considera il mittente M sul canale in ingresso come suo predecessore**
  - **Q diventa dunque il successore di M**
- **Quando Q completa il suo contributo invia a M il messaggio "FINITO"**
- **Lo stato globale per l'iniziatore M' è pronto quando ogni suo successore M abbia ricevuto messaggi "FINITO" da tutti i suoi successori Q'**
- **Nel caso generale lo stato globale potrebbe mostrare messaggi in transito**
  - **Segno di attività non completate**

Esempio d'uso: terminazione – 2

- **Il processo Q invia il messaggio "FINITO" sse**
  - **Tutti i suoi successori hanno inviato il messaggio "FINITO"**
  - **Q non ha ricevuto alcun messaggio successivo al completamento del suo stato**
- **Altrimenti Q invia il messaggio "CONTINUA" al suo predecessore**
  - **L'iniziatore allora invia un nuovo *marker* (inizia una nuova istantanea) finché non riceva solo messaggi "FINITO" dai suoi successori**

Esempio d'uso: terminazione – 3

- **Esiste moltissima documentazione su rete che applica, discute, ed estende l'algoritmo di Chandy & Lamport**
  - **Per un interessante esempio animato, si veda**
    - http://www.risc.uni-linz.ac.at/software/daj/snapshot/index.html
    - Perdonando l'inglese ☺
  - **Vale la pena fare qualche ricerca e qualche riflessione al riguardo!**

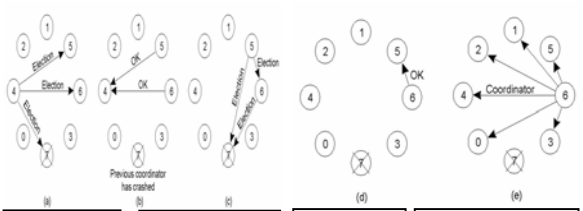
Elezione del coordinatore – 1

- **La presenza di un coordinatore facilita la costruzione di algoritmi distribuiti**
- **Eleggere un coordinatore richiede accordo distribuito**
  - **L'obiettivo dell'algoritmo di elezione è assicurarne la terminazione con l'accordo di tutti i partecipanti**
- **Prerequisiti**
  - **Un identificatore unico e ordinabile (maggiore, minore) è associato a ciascun processo del sistema (o del gruppo)**
  - **Ogni processo conosce gli identificatori di tutti gli altri processi del sistema (o del gruppo)**

Elezione del coordinatore – 2

- **Algoritmo del "bullo" (*bully*)**
  - **Il processo P che rilevi l'assenza del coordinatore promuove una nuova elezione**
    - P invia un messaggio "**ELEZIONE"** a tutti i processi di identificatore **maggiore**
    - Se nessuno risponde P si auto-proclama vincitore e diventa coordinatore
    - Un processo che riceva il messaggio "**ELEZIONE"** da un processo di identificatore minore risponde con il messaggio "**OK"** e rileva la gestione dell'elezione
    - Se P riceve un messaggio "**OK"** ha finito il suo lavoro
  - **Un processo appena (ri-)creato non conosce il coordinatore e dunque promuove una elezione**
  - **L'algoritmo designa sempre come coordinatore il processo in vita con identificatore maggiore**
    - Il vincente informa tutti i processi del sistema che hanno un nuovo coordinatore

Elezione del coordinatore – 3



Il processo **4** inizia una nuova elezione

I processi **5** e **6** rilevano l'elezione in parallelo

Il processo **6** rileva l'elezione del processo **5**

Il processo **6** non conosce processi di identificatore maggiore, quindi si proclama coordinatore

Mutua esclusione – 1

- **Algoritmo centralizzato: facile ma fragile**
  - **Le richieste di accesso ("ENTER") a risorse in mutua esclusione vengono inviate a un coordinatore centrale**
  - **Se la risorsa è libera il coordinatore informa il mittente che l'accesso è consentito ("GRANTED")**
  - **Altrimenti il coordinatore accoda la richiesta con politica FIFO e informa il mittente che l'accesso non è consentito ("DENIED")**
    - Oppure nessuna risposta per richieste sincrone
  - **L'utente che rilascia la risorsa ne informa il coordinatore ("RELEASED")**
  - **Il coordinatore allora preleva la prima richiesta in attesa e invia "GRANTED" al suo mittente**
  - **Il coordinatore è il *Single Point of Failure* (SPF) dell'algoritmo oltre che il suo collo di bottiglia**

Mutua esclusione – 2

- **Algoritmo distribuito**
  - **Richiede infrastruttura di comunicazione affidabile**
  - **Il processo P che voglia accesso esclusivo a una risorsa R costruisce un messaggio M ("GRANT?") contenente <P, R, C> e lo invia a tutti i processi del sistema**
    - C = ora locale di P
  - **Il processo che riceva M**
    - Se non sta usando R e non vuole (ancora) usarla risponde "**OK"**
    - Se invece sta usando R **non risponde** e accoda M presso di sé
    - Se vuole R ma non la ha ancora ottenuta confronta C con la sua ora locale: il valore più basso vince
      - Gli orologi devono essere coerenti! ← **Fattore critico**

Mutua esclusione – 3

- **Algoritmo distribuito**
  - **Per procedere P aspetta di aver ricevuto "OK" di tutti i processi**
    - Quando ciò avviene P accede a R in mutua esclusione
    - L'accesso avviene con garanzia di assenza sia di *deadlock* che di *starvation*
  - **Quando P rilascia R risponde OK a tutti i processi mittenti di richieste accodate presso di sé e le rimuove dalla coda**
    - A quel punto prevale il richiedente con C "minore"
- **Da 1 SPF (coordinatore) a N SPF (ogni processo)**
  - **I processi devono sempre rispondere a ogni richiesta**
  - **Mancata risposta (*time-out*) interpretata come "occupato"**
  - **Alla ricezione del primo "occupato" il richiedente deve bloccarsi in attesa del primo "OK" successivo**

Mutua esclusione – 4

- **Algoritmo a gettone circolante (*token ring*)**
  - **Processi collegati in sequenza circolare ordinata e punto a punto lungo la quale deve transitare un gettone**
  - **Il processo in posizione 0 riceve per primo il gettone**
  - **Il possesso del gettone consente al processo di accedere 1 risorsa in mutua esclusione**
    - Per poi passare il gettone al suo vicino
  - **Se il processo non ha immediato bisogno di risorse passa subito il gettone al vicino**
    - Il vicino conferma la ricezione altrimenti viene rimosso dalla sequenza
  - **Nel caso peggiore un processo richiedente aspetta una intera rotazione del gettone**
- **Il gettone è 1 SPF → se perso, va rigenerato**

Mutua esclusione – 5

- **I 3 algoritmi possono essere raffrontati in relazione a 3 criteri fondamentali**
  - **Numero di messaggi necessari al processo per poter operare sulla risorsa richiesta (ingresso e uscita)**
  - **Il tempo necessario perché la richiesta abbia successo**
  - **Le debolezze dell'algoritmo**
- **Questi 3 criteri possono essere applicati a varie classi di algoritmi distribuiti**

Mutua esclusione – 6

Algoritmo	# Messaggi per accesso e rilascio risorsa	Max attesa di accesso spesa per invio messaggi (costo >> lavoro)	Punti deboli (SPF)
Centralizzato	**3** (ENTER, GRANTED, RELEASED)	**2** (ENTER, GRANTED)	**Guasto del coordinatore**
Distribuito	**2 (n – 1)** (GRANT?, RELEASED) da uno a tutti gli altri	**2 (n – 1)**	**Guasto di qualsiasi processo**
Gettone circolante	**1 .. ∞** (se tutti [1] o nessuno [∞] sono interessati alla risorsa)	**0 .. n – 1** (gettone in possesso, gettone all'altro capo)	**Gettone perso Guasto di processo**

Raffronto prestazionale tra gli algoritmi

Argomenti non trattati

- **Argomenti importanti per la problematica di questa lezione non trattati per limiti temporali del corso**
  - **Sincronizzazione degli orologi fisici**
    - Il *middleware* di ogni nodo del sistema distribuito aggiusta il valore del suo orologio fisico in modo coerente con quello degli altri
  - **Sincronizzazione degli orologi logici**
    - Leslie Lamport ha mostrato come l'accordo degli orologi fisici non sia necessario ma lo sia solo l'ordinamento degli eventi (relazione "precede")
  - **Transazioni distribuite**
    - Come ottenere mutua esclusione e **operazioni atomiche** su dati condivisi (***A**tomicity – **C**onsistency – **I**solation – **D**urability*)