Computing in presence of fault

Marco Buracchi

Università degli studi di Firenze

5 febbraio 2018

Introduzione

Guasti

Guasto

Si parla di guasto quando accade qualcosa nel sistema che devia dal comportamento atteso.

Scenario

- Total reliability impossibile da ottenere praticamente
- Un guasto prima o poi accadrà e dobbiamo essere in grado, se possibile, di terminare comunque la computazione

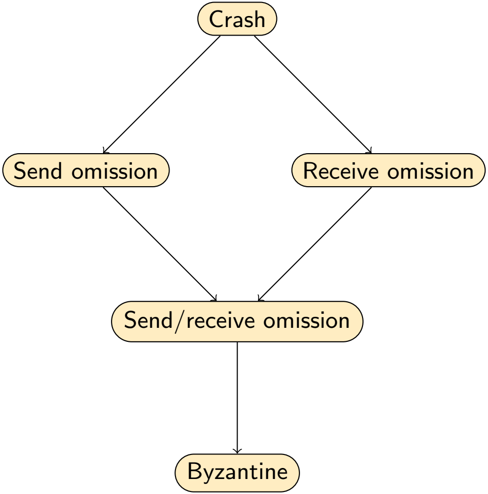
Guasti

Nessun protocollo può resistere ad un numero indefinito di guasti. Se tutto il sistema crolla, non può esistere un protocollo corretto.

Obiettivo

Costruire protocolli che resistano ad un certo numero di guasti di un determinato tipo.

Gerarchia dei guasti



Modelli di guasto

Modelli di guasto

- Modelli di guasto dei componenti
  - Entity
  - Link
  - Hybrid
- Modelli di guasto di comunicazione
  - Omissione
  - Aggiunta
  - Corruzione

Fattori topologici

Connessione

Definiamo:

- $C_{edge}(G)$ il minimo numero di archi la cui rimozione rende la rete G non fortemente connessa
- $C_{node}(G)$ il minimo numero di nodi la cui rimozione rende la rete G non fortemente connessa

Proprietà

$C_{\bullet}(G) = k \Rightarrow$ per ogni coppia di nodi x, y ci sono k cammini diversi che li uniscono.

Fattori topologici

Network G	Node Connectivity $c_{\text{node}}(G)$	Edge Connectivity $c_{\text{edge}}(G)$
Tree T	1	1
Ring R	2	2
Torus Tr	4	4
Hypercube H	$\log n$	$\log n$
Complete K	$n - 1$	$n - 1$

Entity failure

p-Consenso

Problema

- Ogni entità x ha un valore in input $v(x) \in \mathcal{I}$
- Ogni entità deve decidere un valore $d(x) \in \mathcal{O}$ in tempo finito
- Una volta scelto $d(x)$ non può più essere modificato
- Se tutti i valori $v(x)$ sono uguali, deve essere scelto quel valore
- Almeno p entità devono decidere lo stesso valore $d(x)$
  - $p = N \rightarrow$ *unanimità*
  - $p = \lceil \frac{N}{2} \rceil + 1 \rightarrow$ *maggioranza assoluta*

Consenso con guasti di tipo crash

Consideriamo solo guasti di tipo crash. Siano $\mathcal{S}$ l'insieme delle entità non guaste, $\mathcal{F}$ l'insieme delle entità guaste e $\mathcal{I} = \mathcal{O} = \{0, 1\}$

Problema

- Ogni entità $x \in \mathcal{S}$ ha un valore in input $v(x) \in \mathcal{I}$
- Ogni entità $x \in \mathcal{S}$ deve decidere un valore $d(x) \in \mathcal{O}$ in tempo finito
- Una volta scelto $d(x)$ non può più essere modificato
- Se tutti i valori $v(x)$ iniziali sono uguali, $d(x) = v(x)$ (*non-trivialità*)
- Vogliamo l'unanimità su tutti gli $x \in \mathcal{S}$

Consenso con guasti di tipo crash

Assunzioni

- La rete è un grafo completo
- Link bidirezionali
- Sincronia
  - Delay di comunicazione unitari
  - Clock sincronizzati
- Tutte le entità iniziano la computazione contemporaneamente
- L'unico tipo di guasti è CRASH

Protocollo TellZero-Crash

begin
    if $I_x = 0$ **then send** 0 **to** $N(x)$;
    for $t = 1, \ldots, f$ **do**
        compute $rep(x, t)$;
        if $(rep(x, t) = 0$ **and** $rep(x, t-1) = 1)$ **then send** 0 **to** $N(x)$;
    endfor
    $O_x := rep(x, f+1)$;
end

$$\text{rep}(x, t) = \begin{cases} v(x) & \text{if t} = 0 \\ \textbf{AND}(\text{rep}(x, t-1), M(x_1, t), \ldots, M(x_{n-1}, t)) & \text{otherwise} \end{cases}$$

Protocollo TellZero-Crash

Proprietà

- Se tutte le entità hanno come valore iniziale 1, tutte le entità $x \in \mathcal{S}$ decideranno 1
- Se un'entità $x \in \mathcal{S}$ ha ricevuto o riceve uno 0 al tempo $t < f$, allora tutte le entità $x \in \mathcal{S}$ riceveranno uno 0 al tempo $t + 1$
- Se un'entità $x \in \mathcal{S}$ ha ricevuto o riceve uno 0 durante l'esecuzione del protocollo, allora deciderà 0

Riassumendo, ogni entità $x \in \mathcal{S}$ deciderà 0 se almeno una di loro ha come valore iniziale 0 e deciderà 1 se tutte hanno come valore iniziale 1 (**AND**).

Guasti bizantini

Guasto bizantino

Un'entità affetta da guasto bizantino può mandare qualunque valore voglia (corretto o errato), in qualunque momento, a qualunque vicino o non mandare niente. Generalmente un guasto bizantino simula un attaccante che cerca di far fallire il protocollo mandando false informazioni o scartando messaggi corretti.

Assunzioni aggiuntive

- ID unico per ogni entità
- Ogni entità conosce l'ID dei suoi vicini

Guasti bizantini

Consenso

Con le precedenti assunzioni si riesce ad arrivare al consenso con un numero di entità bizantine al massimo pari a $(\frac{N}{3} - 1)$.

Iniziamo a vedere come raggiungere il consenso, inizialmente utilizzando solo valori booleani ($\mathcal{I} = \mathcal{O} = \{0, 1\}$)

Guasti bizantini

Formato messaggi

- $\{\bullet, 0, \text{id}(s), t\}$
- s è il mittente del messaggio
- id(s) il suo ID univoco
- t il passo corrispondente a quando è stato spedito il messaggio

Problemi con guasti bizantini

1. $z \in \mathcal{F}$ potrebbe mandare $\{\bullet, 0, \text{id}(z), t'\}$ a $x \in \mathcal{S}$ con $t' \neq t$
2. $z \in \mathcal{F}$ potrebbe mandare $\{\bullet, 0, \text{id}(y), t\}$ a $x \in \mathcal{S}$ con $y \neq z$
3. $z \in \mathcal{F}$ potrebbe mandare informazioni diverse a vicini diversi facendo decidere alcuni 0 e altri 1.

RegisteredMail

Algoritmo

Siano $x, y \in \mathcal{S}$, $v(x) = 0$ e sia $z \in \mathcal{F}$

- x manda inizialmente al tempo t {"init",0,id(x),t} a tutte le entità
    - Al tempo $t + 1$, y riceve {"init",0,id(x),t} e manda {"echo",0,id(x),t} a tutte le entità
    - Al tempo $t' \neq t + 1$, y riceve {"init",0,id(x),t} ed ignora il messaggio
- Al tempo $t' \geq t + 2$, y ha ricevuto {"echo",0,id(x),t} da almeno f +1 entità diverse e manda {"echo",0,id(x),t} (se non lo ha già fatto) al tempo t' a tutte le entità
- Al tempo $t' \geq t + 1$ y ha ricevuto {"echo",0,id(x),t} da almeno $n - f$ entità diverse e accetta il messaggio

RegisteredMail

Proprietà

1. se x manda {"init",0,id(x),t} , allora il messaggio verrà accettato da tutte le entità non guaste al tempo t+2
2. se un messaggio è accettato da una qualunque entità non guasta al tempo $t' > t$, allora sarà accettato da tutte le entità non guaste al tempo $t' + 1$
3. se x non manda {"init",0,id(x),t} , allora un eventuale messaggio {"init",0,id(x),t} non sarà accettato dalle entità non guaste

TellZeroByz

Protocollo

1. Al tempo 0, ogni entità $x \in \mathcal{S}$ con $v(x) = 0$ avvia RegisteredMail (RM) mandando $\{$"init",0,id(x),t$\}$
2. Al tempo $2i$, $1 \leq i \leq f + 1$, una entità $x \in \mathcal{S}$ avvia RM mandando $\{$"init",0,id(x),2$i\}$ SSE ha accettato messaggi da almeno $f + i - 1$ entità differenti al tempo $2i$ e non ha ancora mandato un messaggio $\{$"init",0,id(x),t$\}$
3. Al tempo $2(f + 2)$, una entità $x \in \mathcal{S}$ decide 0 SSE fino a quel momento ha accettato messaggi da almeno $2f + 1$ entità diverse. Altrimenti decide 1.

TellZeroByz

Risultato

TellZeroByz risolve il problema del consenso con valori booleani in una rete sincrona e completa sotto le restrizioni dichiarate per ogni valore di $f \leq \frac{N}{3} - 1$ dopo $2(f+2)$ unità di tempo.

Per dimostrare questo risultato dobbiamo dimostrare la non-trivialità e il consenso.

Dimostrazione

Non-trivialità

- Se tutte le entità non guaste hanno valore iniziale 0, tutte faranno partire RM al tempo 0 e, per le proprietà del meccanismo, tutte accetteranno i rispettivi messaggi al tempo 2 e decideranno 0 al termine del protocollo.
- Se tutte le entità non guaste hanno valore iniziale 1, non faranno mai partire RM. Infatti, per farlo, una entità deve accettare almeno $f + 1$ messaggi ma solo le f entità guaste possono aver generato tali messaggi. Di conseguenza le entità non guaste decideranno 1 al termine del protocollo.

Dimostrazione

Se un'entità non guasta decide 0, tutte le altre unità non guaste decideranno 0. Supponiamo che $x \in \mathcal{S}$ abbia deciso 0.

Consenso caso tutti 0

- Al tempo $t = 2(f+2)$ x ha accettato messaggi da almeno $2(f+1)$ entità diverse. Sia $\mathcal{R}$ l'insieme delle entità non guaste tra queste.
- $|\mathcal{R}| \geq (2f+1) - f = f+1$

Dimostrazione

Consenso caso tutti 0

- Se tutte le entità in $\mathcal{R}$ hanno valore iniziale 0, ognuna fa partire RM al tempo 0.
- Proprietà 1 di RM $\Rightarrow$ tutte le unità non guaste accettano messaggi da $|\mathcal{R}| \geq f+1$ entità al tempo 2.
- Proprietà 2 di TZB $\Rightarrow$ ogni entità non guasta farà partire RM.
- Proprietà 1 di RM $\Rightarrow$ tutte le unità non guaste accetteranno questi messaggi al tempo 4.
- Al tempo $2(f+2) \geq 4$ il protocollo termina e tutte le unità non guaste decidono 0.

Dimostrazione

Consenso caso almeno un 1

- Se una entità y in $\mathcal{R}$ ha valore iniziale 1, non farà partire RM al tempo 0.
- x ha accettato il suo messaggio $\Rightarrow$ y deve aver fatto partire RM a qualche tempo $2i$ con $1 \leq i \leq f + 1$
- Proprietà 2 di TZB $\Rightarrow$ se y fa partire RM al tempo $2i$ $\Rightarrow$ ha accettato almeno $f + i - 1$ messaggi differenti.
- Proprietà 2 di RM $\Rightarrow$ gli $f + i - 1$ messaggi sono accettati da tutte le unità non guaste al tempo $2i + 1$.
- Proprietà 1 di RM $\Rightarrow$ il messaggio originato da y al tempo $2i$ viene accettato da tutte le unità non guaste al tempo $2i + 2$.
- Ogni unità non guasta accetta almeno $(f + i - 1) + 1 = f + i$ messaggi al tempo $2i + 2$

Dimostrazione

Consenso caso almeno un 1

- $i \leq f \Rightarrow$ tutte le unità non guaste che non hanno fatto ancora partire RM, lo faranno al tempo $2i + 2$.
  - Al tempo $2i + 4 \leq 2f + 4 = 2(f + 2)$ ogni entità non guasta avrà accettato almeno $n - f \geq 2f + 1$ differenti messaggi.
  - Ogni entità non guasta deciderà 0 al termine del protocollo.
- $i = f + 1 \Rightarrow$ tutte le unità non guaste hanno accettato $f + i \geq 2f + 1$ differenti messaggi al tempo $2(f + 1) + 2 = 2(f + 2)$
  - Tutte le unità non guaste decideranno 0 al termine del protocollo (adesso).

Complessità

Tempo

Il protocollo termina sempre in $2(f+2)$ unità di tempo.

Numero di messaggi

- Ogni $x \in \mathcal{S}$ fa partire RM al più una volta creando $n-1$ messaggi di "init" e al più $n(n-1)$ messaggi di echo.
- Ogni $z \in \mathcal{F}$ può mandare messaggi a tutti i vicini ad ogni istante di tempo $\Rightarrow 2(f+2)(n-1))$.
- Un tempo pari, può essere usato da z per far partire RM generando quindi ulteriori $n(n-1)$ messaggi.
- Il numero di messaggi totale è quindi $\mathcal{M} \leq (2f^2+4f+n+n^2-fn+n-f)(n-1) = \mathcal{O}(n^3)$

Oltre TZB

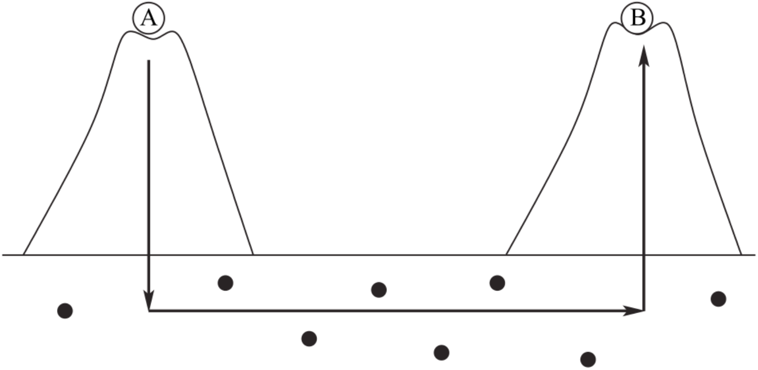
Generalizzazione

Esistono algoritmi che generalizzano questi risultati:

- L'algoritmo *FromBoolean(TZB)* generalizza l'insieme di valori in input da Booleano ad arbitrario.
  - Complessità messaggi $\leq 2n(n-1) + TZB$
  - Complessità tempo $= 2f + 6$
- L'algoritmo *ByzComm(TZB)* risolve il problema in una rete sincrona con $C_{node}(G) \geq 2f + 1$, $\forall f \leq \frac{n}{3} - 1$ ma ogni entità deve conoscere tutta la topologia della rete.
  - Complessità messaggi $= \mathcal{O}(f\, n^4 \log n)$
  - Complessità tempo $= \mathcal{O}(f\, n)$

Link failure

Il problema dei generali sincronizzati



I generali sincronizzati

Teorema

Il problema dei generali sincronizzati è irrisolvibile. Per raggiungere la *common knowledge*, supponendo che F link possano guastarsi, si ha bisogno che la rete G abbia $C_{edge}(G) \geq F + 1$.

In pratica abbiamo bisogno della certezza che ci sia almeno un link tra qualunque coppia di nodi che sicuramente non si guasti.

Broadcasting is the way

Condizioni

1. Al più k link possono guastarsi
2. $C_{edge}(G) \geq k + 1$
3. ID univoci (opzionale)

Soluzione

- 1 e 2 $\Rightarrow$ broadcast (ad esempio con flood) in sicurezza $\Rightarrow$ possiamo computare funzioni base (AND, OR, Min, Max)
- 1,2 e 3 $\Rightarrow$ leader election

Numero di guasti

Costo

Broadcast in un grafo completo senza guasti è banale e costa $(n-1)$ messaggi.
Consideriamo di avere sugli $\frac{n(n-1)}{2}$ collegamenti, $f < n-1$ guasti

- Non conosciamo f
  - Con Flood risolviamo il problema ma ci costa $(n-1)^2$ messaggi
- Conosciamo f a priori
  - Con *TwoSteps* possiamo abbassare questo costo.

TwoSteps

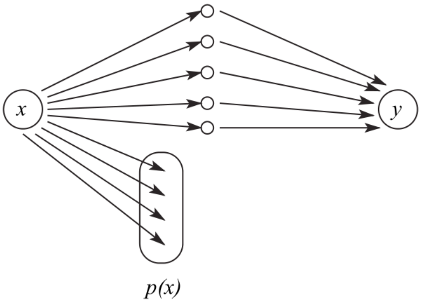
Algoritmo

1. x vuole far arrivare a tutti il valore *v* e manda un messaggio { Info, v } a $f+1$ vicini
2. Una entità y che riceve un messaggio { Info, v } da x, manda { Echo, v } a tutti i suoi vicini

TwoSteps

Correttezza

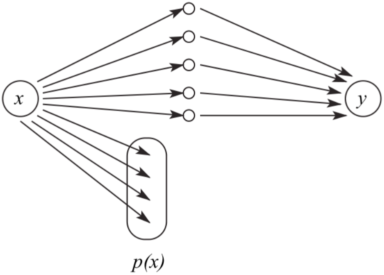
- $y \neq x$ non riceve il messaggio Info da x a causa del link (x, y) guasto
- Sia $p(x) \leq k$ il numero di link guasti incidenti x.
- Al primo passo, almeno $n - 1 - p(x)$ vicini riceveranno il messaggio Info di x



TwoSteps

Correttezza

- Al secondo passo, tutti loro manderanno il messaggio Echo a tutti i loro vicini, incluso y
- Almeno $n-1-p(x)$ messaggi di Echo verranno mandati a y e al più $f - p(x)$ link saranno guasti
- $n-1 > f \Rightarrow n-1-p(x) > f - p(x) \Rightarrow$ almeno un Echo arriva ad y



TwoSteps

Costo

- Al primo passo x manda $f+1$ messaggi
- Al secondo passo, al più $f+1$ entità mandano ognuna $n-2$ messaggi
- $(f+1)+(f+1)(n-2)=(f+1)(1+n-2)=(f+1)(n-1)$

Dato che $f \leq n-2$ il numero di messaggi scambiato da TwoSteps è minore di Flood.

Leader election

Supponiamo sempre $f < n - 1$ e che ogni entità abbia un ID unico.

FT-BcastElect

1. Ogni entità x manda in broadcast con un protocollo fault tolerant (ad esempio con TwoSteps) il suo valore id(x)
2. Una volta che x ha ricevuto tutti i valori di tutte le altre entità, x diventa leader SSE il suo id è il minimo

Costo

Tutti gli n nodi eseguono TwoSteps quindi il numero di messaggi scambiato è $\leq n(f+1)(n-1)$

Conclusioni

Difficile ma non impossibile

- I guasti accadono, dobbiamo conviverci
- Gli algoritmi funzionanti in presenza di guasti sono corretti ma spesso costosi (sia in termine di messaggi che di infrastrutture necessarie)
- Esistono casi particolari nei quali, sotto opportune condizioni, si riescono a raggiungere buoni risultati con costi contenuti


That's all Folks!