SDRC 2013/2014 — Sistemi Distribuiti e Reti di Calcolatori — P. Crescenzi — Riccardo Bernini

1 Stabilizzazione

Un grande ramo della ricerca in calcolo distribuito si occupa della tolleranza ai guasti con l'obbiettivo di tollerare una frazione considerevole di fallimenti, come ad esempio rottura del singolo nodo o errori di comunicazione.
Affronteremo questo contesto, analizzando uno dei problemi fondamentali dei sistemi distribuiti che è quello del **consenso**. Tale problema risulta fondamentale in quanto permette di risolvere altri problemi come ad esempio quello dell'elezione del leader. Una delle sue applicazioni più importanti è al problema del *'commit'* in un database distribuito in cui più processori devono decidere se aggiornare o meno il valore di un record. Più precisamente si tratta di risolvere il problema di disaccordo tra processori sulla decisione d'aggiornamento. Se anche un solo processore non concorda nel fare il *commit* questo non deve essere fatto.
Supponendo di avere più processori dove ciascuno di essi ha un valore locale, che assumiamo essere booleano intuitivamente quindi l'obbiettivo è far convergere questi valori locali, possibilmente anche diversi, verso un unico di questi.
La formulazione più famosa di questo problema è quella dei *'Generali Bizantini'*. In tale formulazione vi sono due generali a capo ciascuno di una parte dell'armata i quali devono decidere se attaccare o meno un avversario A:

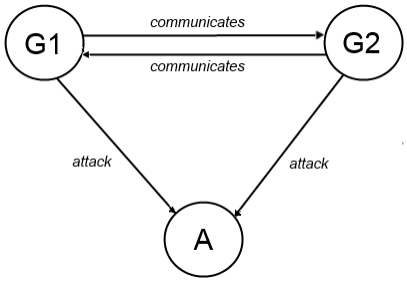

Figura 1: Generali Bizantini

L'esito dell'attacco è determinato dall'accordo tra i generali:

- se G_1 e G_2 attaccano contemporaneamente $\rightarrow$ l'avversario A è sconfitto,
- se G_1 attacca e G_2 non attacca o viceversa $\rightarrow$ l'avversario A vince.

È chiaro che ciascun generale sceglierà indipendentemente dall'altro quando e se attaccare, secondo una propria strategia e notificherà all'altro questa scelta.
Osserviamo che questo problema, ovviamente, è facilmente risolvibile in assenza di problemi di comunicazione tra i due generali, i quali possono quindi accordarsi sul momento d'attacco.
Non appena viene eliminata l'assunzione di assenza di fallimenti e vengono quindi permessi errori in comunicazione (un messaggio può non raggiungere la destinazione) il problema diventa irrisolvibile, cioè non vi è un modo sempre valido per far accordare i generali sulla strategia da adottare.
Questo problema si generalizza facilmente a qualsiasi numero di generali assumendo che il grafo della comunicazione sia un grafo completo e cioè che ogni generale possa comunicare con ogni altro generale mantenendo l'ipotesi che un messaggio può non raggiungere la destinazione.

1.1 Problema del consenso

Il problema appena descritto si formalizza e si generalizza a più nodi considerando un grafo $G = (V,E)$ con $E = V^2$ (ogni nodo può comunicare con ogni altro nodo), supponendo di lavorare in un ambiente sincrono e introducendo la possibilità che i collegamenti falliscano.
L'obbiettivo diviene la generazione di un algoritmo che presi i valori booleani iniziali di tutti i nodi permetta a ciascun nodo, attraverso lo scambio di messaggi, di accordarsi su un unico valore booleano finale. L'algoritmo oltre a rispettare questa condizione, chiamata di **consenso**, deve anche rispettare la condizione di **validità** più precisamente al termine dell'esecuzione devono essere soddisfatte entrambe le seguenti condizioni:

- **Consenso**: tutti i nodi al termine dell'esecuzione dell'algoritmo devono concordare su un unico valore booleano.
- **Validità**:
  - Se tutti i nodi iniziano con il valore *'0'* $\rightarrow$ Il consenso si deve avere sul valore booleano *'0'*.
  - Se tutti i nodi iniziano con *'1'* e non ci sono fallimenti nella comunicazione $\rightarrow$ Il consenso si deve avere sul valore booleano *'1'*.

Queste due condizioni impediscono la formulazione di un algoritmo banale come ad esempio il semplice consenso su *'0'* (bizantini non attaccano), indipendentemente dai valori iniziali di ciascun nodo.
Si può osservare che la *condizione di validtà* risulta una condizioni piuttosto debole in quanto anche un solo fallimento consente all'algoritmo di convergere ad un valore di default indipendentemente dal valore iniziale.
Vediamo adesso però che, anche con un vincolo così lasco, non vi è comunque una soluzione al problema. Definiamo prima però, il concetto di '**equivalenza tra esecuzioni**' che ci sarà utile nella dimostrazione di questo risultato.

Supponiamo di avere un algoritmo che risolve il problema del consenso e di avere due esecuzioni di tale algoritmo α e β, in questo contesto definiamo il concetto di '**equivalenza tra esecuzioni**' nel seguente modo:

$$\alpha \sim^i \beta \begin{cases} \text{nodo } v_i Il \text{ ha lo stesso stato iniziale in } \alpha \text{ e } \beta. \\ Il \text{ nodo } v_i \text{ riceve gli stessi messaggi negli stessi round in } \alpha \text{ e } \beta. \end{cases} \quad \text{con } i = 1,2$$

Intuitivamente questa definizione afferma che le due esecuzioni devono essere identiche dal punto di vista dello stesso nodo mentre possono variare per nodi diversi, cioè un nodo deve inviare e ricevere gli stessi messaggi in entrambe le esecuzioni mentre questo non è richiesto per nodi diversi.

Dimostriamo quindi il seguente risultato:

Teorema 1.1: *Non esiste un algoritmo che soddisfa la condizione del consenso e la condizione di validità in presenza di un numero di fallimenti di comunicazione non limitati.*

Dimostrazione: Per dimostrare questa impossibilità di risoluzione consideriamo il caso semplice in cui $V = \{v_1, v_2\}$ ed $E = \{\{v_1, v_2\}\}$ (due nodi collegati da un arco bidirezionale).
Supponiamo adesso per assurdo che esista un algoritmo che risolva il consenso correttamente ed ancora che, per semplicità, ad ogni round vi sia l'invio di un messaggio dal nodo v_1 al nodo v_2 e l'invio di un messaggio dal nodo v_2 al nodo v_1 (se così non fosse possiamo supporre d'inviare un messaggio fittizio).
Consideriamo quindi le seguenti esecuzioni:

- α_1 in cui entrambi i nodi v_i iniziano con *'1'* e in cui non vi è alcun fallimento,
- α_2 identica ad α_1 con l'unica eccezione che l'ultimo messaggio che il nodo v_1 invia al nodo v_2 fallisce,
- α_3 identica ad α_2 con l'unica eccezione che l'ultimo messaggio che il nodo v_2 invia al nodo v_1 fallisce,

Osserviamo che l'esecuzione α_1 termina intuitivamente, per la condizione di validità, con il consenso in *'1'*.

Se analiziamo l'esecuzioni α_1 e α_2 dal punto di vista del nodo v_1, queste due esecuzioni risultano identiche e cioè $\alpha_1 \sim^1 \alpha_2$ perché in α_2 l'unico fallimento si ha nella recezione dell'ultimo messaggio inviato dal nodo v_1 al nodo v_2. Quindi il nodo v_1 invia e riceve gli stessi messaggi in entrambe le esecuzioni.
Questa equivalenza implica che il nodo v_1 decide *'1'* in entrambe le esecuzioni, decidendo *'1'* in α_1 e per l'uso di un algoritmo del consenso abbiamo che anche il nodo v_2 in α_2 deciderà *'1'* perché appunto l'algoritmo soddisfa la condizione di consenso (tutti i nodi concordano sul valore booleano).
Analizzando adesso l'esecuzioni α_2 e α_3 si può osservare invece che le due esecuzioni risultano identiche dal punto di vista del nodo v_2 (invia e riceve stessi messaggi e non rileva il fallimento dell'ultima recezione nel nodo v_1 nell'esecuzione α_3) e cioè $\alpha_3 \sim^2 \alpha_2$. Quindi se per il nodo v_2, α_2 è identica ad α_3 e in α_2 il nodo v_2 decide *'1'*, per l'equivalenza dovrà decidere *'1'* anche in α_3 ma d'altra parte, come in precedenza, siccome questo è un algoritmo del consenso implica che anche il nodo v_1 dovrà decidere *'1'*.
Si può osservare che eseguendo in successione questo procedimento, cioè considerando iterativamente le esecuzioni della forma precedente andiamo man mano a eliminare i messaggi ricevuti, partendo dall'esecuzione iniziale dove tutti i messaggi inviati sono ricevuti fino ad arrivare all'esecuzione:

$$\alpha^* : \begin{cases} \text{I nodi } v_1, v_2 \text{ partono da } \textit{'1'}. \\ \text{Nessun messaggio è recapitato.} \\ \text{I nodi } v_1, v_2 \text{ decidono entrambi } \textit{'1'}. \end{cases}$$

dove cioè nessun messaggio inviato viene ricevuto.

Definiamo a questo punto l'esecuzione:

$$\alpha^{**} : \begin{cases} \text{Il nodo } v_1 \text{ parte da } \textit{'1'}. \\ \text{Il nodo } v_2 \text{ parte da } \textit{'0'}. \\ \text{Nessun messaggio è recapitato.} \end{cases}$$

e osserviamo che $\alpha^* \sim^1 \alpha^{**}$ in quanto il nodo v_1 ha come valore iniziale *'1'* e non ricevendo alcun messaggio decide *'1'* anche in questa esecuzione. Siccome l'algoritmo utilizzato è un algoritmo del consenso, anche il nodo v_2 in α^{**} decide *'1'*.
Come ultimo passo supponiamo di avere un'altra esecuzione α^{***} di questa forma:

$$\alpha^{***} : \begin{cases} \text{Il nodo } v_1 \text{ parte da } \textit{'0'}. \\ \text{Il nodo } v_2 \text{ parte da } \textit{'0'}. \\ \text{Nessun messaggio è recapitato.} \end{cases}$$

Osserviamo di nuovo che il nodo v_2 osserva il transito degli stessi messaggi sia in α^{**} che in α^{***} ed ha lo stesso valore iniziale *'0'*, quindi $\alpha^{***} \sim^2 \alpha^{**}$ da cui si ha che il nodo v_2 deciderà *'1'* perché per l'equivalenza deve decidere lo stesso valore deciso in α^{**} .
Quest'ultima osservazione porta all'assurdo perché essendo partiti entrambi i nodi da *'0'* al termine dell'algoritmo dovevano, per la condizione di validità, concordare in *'0'* e non decidere *'1'*. Da qui si può concludere che non esiste un algoritmo del consenso corretto.
□

Abbiamo appena dimostrato l'esistenza di uno dei limiti centrali del calcolo distribuito, ovvero supponendo che il numero di fallimenti non sia limitato, non può esistere un algoritmo del consenso deterministico corretto qualsiasi sia il numero di round utilizzabili.

1.2 Algoritmo del consenso randomizzato

Per superare questo limite intrinseco, accettiamo la possibilità di compiere errori o meglio accettiamo che l'algoritmo commetta errori entro una certa probabilità. Passiamo quindi da un algoritmo deterministico ad un algoritmo proba-

bilistico nel quale cercheremo appunto di limitare la probabilità che il risultato ottenuto sia diverso da quello che ci aspettiamo. Limitare la probabilità che ogni nodo non decida al termine dell'esecuzione dell'algoritmo lo stesso valore booleano.
In un algoritmo del consenso probabilistico, la probabilità entra in gioco sui nodi, sono quest'ultimi quelli che compiono scelte probabilistiche durante l'esecuzione dell'algoritmo mentre non associamo alcuna distribuzione di probabilità al fallimento dei collegamenti.
In altri termini, decidiamo quali sono i fallimenti che si verificano durante l'esecuzione dell'algoritmo e stimiamo la probabilità che rispetto a questa sequenza di fallimenti al termine dell'algoritmo i nodi concordino.
Definiamo quindi il concetto di *Avversario*, il quale cercherà di mettere in difficoltà l'algoritmo. Un avversario è specificato mediante:

- Il valore iniziale dei nodi.
- Una sequenza di triple *(i,j,k)*.

La sequenza di triple permette di specificare quali sono i messaggi che verranno recapitati correttamente, più precisamente ciascuna tripla identifica l'evento '*il messaggio da i a j al round k è stato recapitato*'.
Lo scopo di quest'avversario, come abbiamo detto, è mettere in difficoltà l'algoritmo e cioè portare al limite la probabilità d'errore di quest'ultimo. Tale probabilità varierà allora in funzione di questo, derivando così le due nuove condizioni che l'algoritmo del consenso probabilistico deve soddisfare:

- **Consenso**: Pr_A[Due nodi non concordano] $\leq \varepsilon$.
- **Validità**:
  - Se in A, tutti i nodi hanno valore zero $\rightarrow$ il consenso al termine dell'algoritmo deve essere su zero.
  - Se in A, tutti i nodi iniziano con uno e tutti i messaggi sono stati recapitati $\rightarrow$ Pr_A[Tutti i nodi decidono 1] $\geq L$.

Utilizzando queste due nuove condizioni accettiamo la possibilità che l'algoritmo non funzioni correttamente con l'obbiettivo però di minimizzare l'ε ed allo stesso tempo massimizzare L, questo perché intuitivamente vorremo che la probabilità che al termine dell'algoritmo i nodi non concordino sia la più piccola possibile qualsiasi siano i valori iniziali di tutti i nodi.
Questo problema quindi è risolvibile per specifici valori di ε ed $\mathbf{L}$, osserveremo inoltre che al crescere del numero di round permessi all'algoritmo riusciamo a diminuire ed aumentare rispettivamente ε e $\mathbf{L}$.
Descriviamo adesso l'algoritmo del consenso probabilistico, l'osserveremo nel caso semplice con soli due nodi v_1, v_2 collegati tra loro rispettivamente da un arco bidirezionale.
Fissiamo quindi r, come numero massimo di round e consideriamo i nodi v_1, v_2:

- Al primo round, il nodo v_1 sceglie una chiave a caso in $\{1,\ldots,\mathrm{r}\}$. Con questa scelta, v_1 valorizza una soglia che corrisponde al minimo numero di round necessari per poter concludere che l'algoritmo ha terminato con successo.
- $\forall$ round$\neq 1$: un nodo invia un messaggio con:
  - proprio valore inziale,
  - la chiave (solo v_1),
  - un colore tra verde o rosso (il colore rosso sta a significare che qualche messaggio si è perso).

Il colore viene scelto, ad esempio dal nodo v_1 per comunicare all'altro nodo v_2 l'avvenuta recezione o meno del messaggio precedente. Questo perché ad ogni round deve essere ricevuto un messaggio (sistema sincrono) e nel caso in cui il nodo v_1 non lo riceva, al round successivo invierà un messaggio al nodo v_2 di colore rosso.
Dopo il round r, ogni nodo utilizzerà la seguente regola di decisione per convergere ad uno stesso valore booleano:

$$\text{Il nodo decide } \textit{'1'} \text{ se: } \begin{cases} \text{è a conoscenza che almeno un nodo ha valore iniziale uno,} \\ \text{conosce la chiave,} \\ \text{tutti i messaggi sono ricevuti nei primi '}\textit{chiave}\text{' rounds e sono verdi.} \end{cases}$$

Si può osservare come la condizione di validità è un'immediata conseguenza dell'esecuzione dell'algoritmo; infatti se tutti i nodi hanno come valore iniziale *'0'* nessuno di essi potrà terminare decidendo *'1'* e lo stesso vale per il viceversa. Se tutti i nodi hanno come valore iniziale *'1'* e tutti i messaggi sono recapitati e quindi verdi, nessuno di essi potrà decidere *'0'*.
La situazione di trattazione non banale è invece quella del consenso, cioè il caso in cui i nodi non concordino nel valore iniziale (il primo ad esempio inizia con valore *'0'* e l'altro con valore *'1'*). Assumiamo anche in questo caso la possibilità che alcuni messaggi non siano recapitati questo perché, banalmente, se non introduciamo la possibilità di fallimenti nella comunicazione il consenso è facilmente ottenibile attraverso lo scambio dei messaggi.
Diamo adesso un risultato importante che ci permetterà di valorizzare facilmente sia ε che L.

Teorema 1.2: *Ammettendo la possibilità di perdere dei messaggi e sia k il primo round in cui questo avviene, allora l'algoritmo del consenso probabilistico non raggiunge il consenso se k = 'chiave'.*

Dimostrazione: Dimostriamo questo risultato lavorando sui possibili valori che k può assumere, restringendoci al caso semplice con soli due nodi collegati da un arco bidirezionale:

- Se k $<$ 'chiave', cioè il caso in cui un messaggio si perde in un round precedente a quello scelto da v_1. Ad esempio, supponiamo sia il nodo v_1 a non ricevere il messaggio al round k, nel round successivo $k+1 \leq$ '*chiave*' il nodo v_1 invierà un messaggio di colore rosso.
  Sempre al round k+1 l'altro nodo, v_2, può non riceve il messaggio oppure se lo riceve quest'ultimo è di colore rosso. Quindi entrambi i nodi non soddisfano il terzo punto della regola di decisione descritta in precedenza, più precisamente abbiamo che '*non tutti i messaggi sono ricevuti nei primi chiave rounds e sono verdi*' trovando così il consenso in *'0'*.
- Se k$>$'chiave', per definizione fino al round 'chiave' tutti i messaggi sono stati consegnati e quindi sono verdi. Siccome tutti i messaggi sono stati recapitati, ciascun nodo è a conoscenza del valore iniziale dell'altro, rendendo possibile l'applicazione del primo punto della regola di decisione e trovando il consenso, a seconda di tali valori iniziali, in *'0'* o in *'1'* (se ad esempio uno dei nodi aveva valore iniziale *'1'*, decidono *'1'*).

Il caso sfavorevole, dove non si raggiunge il consenso, è la situazione in cui l'avversario assegna valori diversi a i nodi e producendo la lista dei messaggi consegnati, il nodo v_1 ha sfortunatamente scelto come chiave il round dove viene perso il primo messaggio. Formalmente:

$$Pr_A[\text{nodi non raggiungono il consenso}] = Pr_A[\text{'chiave' = min k : (i,j,k)} \notin \text{A}]$$

Da qui si ha che la probabilità di selezionare proprio quel k come *'chiave'* e quindi non raggiungere il consenso è $\frac{1}{r}$, questo perché l'unica possibilità, come abbiamo appena osservato, è che 'chiave' = k, cioè più precisamente che il primo errore di comunicazione sia avvenuto proprio al round *'chiave'*. Tale probabilità corrisponde all'ε della condizione di consenso.
□

Abbiamo così prodotto, nel caso di due nodi, un protocollo che ha $\varepsilon = \frac{1}{r}$ e $L = 1$ perché la validità è soddisfatta con probabilità 1 per la struttura dell'algoritmo.
Per generalizzare questo risultato ad un grafo arbitrario, introducendo cioè più di due nodi, è necessario prima definire il concetto di *'conoscenza'*, cioè quanto un nodo sa degli altri nodi con il passare dei round.
Fissato un avversario A definiamo $H_A(i,k)$, dove i è l'indice del nodo e k è il numero del round, come:

$$H_A(i,k): \begin{cases} 0 \in H_A(i,k) \text{ in ogni caso.} \\ 1 \in H_A(i,k) \text{ se al round k, il nodo } i \text{ è a conoscenza che un altro nodo ha come valore inziale 1.} \\ h > 1 \in H_A(i,k) \text{ se al round k, il nodo i è a conoscenza che } h-1 \in H_A(j,l)\ \forall j \neq \text{ i per qualche l e round.} \end{cases}$$

Definiamo ancora il concetto di '**livello**' come:

$$level_A(i) = max\ v : v \in H_A(i,k)$$

Intuitivamente il valore di *level* identifica il concetto di distanza di diffusione dell'informazione, precisamente la massima profondità di conoscenza che un nodo possiede. Riportiamo un paio di esempi per cercare di chiarificare il concetto:

- per $level_A(i) = 2$ si ha che il nodo i è a conoscenza, in un qualche round, che un altro nodo ha iniziato l'esecuzione con valore iniziale *'1'*,
- per $level_A(i) = 3$ si ha che il nodo i è a conoscenza che ogni altro nodo è a conoscenza, in un qualche round, che un nodo ha iniziato l'esecuzione con valore iniziale *'1'*,
- per $level_A(i) = n$, che corrisponde alla massima diffusione dell'informazione, si ha intuitivamente che ogni nodo è a conoscenza, in qualche round, di un eventuale valore iniziale *'1'* di un qualsiasi altro nodo.

Utiliziamo adesso questi due concetti per dimostrare che vi è un legame tra ε, L e il numero di round:

Teorema 1.3: *Se un algoritmo del consenso con r round soddifa la condizione di consenso con probabilità ε e la condizione di validità con probabilità L allora $L \leq \varepsilon(r+1)$.*

Dimostrazione: Per dimostrare questa proprietà vediamo è sufficiente provare che Pr_A[i decide 1]$\leq \varepsilon level_A$(i,r):

$$Pr_A[\text{consenso su 1}] \leq Pr_A[\text{i decide 1}] \leq \varepsilon level_A(\text{i,r}) \leq \varepsilon(\text{r+1})$$

L'ultima maggiorazione deriva dal fatto che il livello, al massimo è *r+1*, osserviamo inoltre che: Pr_A[i decide 1]≥ 1.
Dimostriamo quindi che Pr_A[i decide 1]$\leq \varepsilon level_A$(i,r), per farlo generiamo un avversario *A'* diverso da *A* che mantenga l'esecuzione dell'algoritmo identica dal punto di vista del nodo *i*.
Immaginiamo adesso di mandare in esecuzione l'algoritmo e costruire un grafo di esecuzione in cui ogni nodo del grafo è una coppia (i,k) e supponendo (i,k) e (j,k+1) nodi del grafo aggiungeremo un arco tra i due se *i* invia un messaggio a *j* al round k e *j* riceve un messaggio da *i* al round k+1.

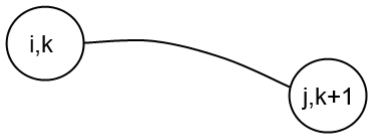

Figura 2: Grafo di esecuzione

Osservando l'esecuzione dal punto di vista di *i*, vi è la possibilità di eliminare tutti gli archi che raggiungono un nodo diverso da *i*, i cui archi raggiungono ancora nodi diversi da *i*.
Eliminiamo quindi tutti gli archi che appartengono a cammini che non condugono a nodi che coinvolgono *i*. Possiamo eliminare tali archi perché corrispondono a scambi di messaggi che non influenzano l'esecuzione di *i*. Possiamo poi modificare il valore iniziale dei nodi che non fanno parte di tale cammini in quanto qualsiasi sia il valore iniziale di quest'ultimi non influenzano il comportamento di *i*, non potendo i propri messaggi raggiungere *i*. Forziamo a *'0'* i valori iniziali di tali nodi.

Il grafo di esecuzione è determinato dall'avversario, perché è lui che specifica quali sono gli stati iniziali e quali sono i messaggi consegnati. Cancellando tutti gli archi inutili per *i* e trasformando tutti i nodi che non possono raggiungere tale nodo(forzando il valore iniziale a *'0'*) l'esecuzione rimane identica. Creiamo quindi il nuovo avversario in cui eliminiamo messaggi consegnati e modifichiamo alcuni valori iniziali.

Questo nuovo avversario definisce un nuovo grafo di esecuzione in cui abbiamo semplicemente eliminato il transito di messaggi che non coinvolgono i producendo l'equivalenza: $A' \sim^i A$.
Dimostriamo quindi la tesi per induzione sul valore di $level_A$, utilizzando il contesto appena descritto:

- **Caso base**. $level_A$(i,r)=0: se $level_A$=0 significa che i non è a conoscenza di alcun nodo che ha come valore iniziale *'1'*. Questo implica quindi che tutti i nodi che iniziano con *'1'* non raggiungono i, il quale altrimenti sarebbe stato avvisato. Per come è definito l'avversario *A'* per tutti questi nodi, che non raggiungono i, viene forzato il valore iniziale *'0'*.
  Riassumendo quindi in *A'* tutti i nodi hanno come valore iniziale *'0'* sia che raggiungono i (i è raggiunto da i soli nodi con valore iniziale *'0'*) sia che non lo facciano (il valore iniziale di tali nodi viene forzato a 0).
  Concludiamo, seguendo le precedenti osservazioni, affermando che i decide *'0'* in A perché $A \sim^i A'$ in cui tutti i nodi decidono *'0'* per la condizione di validità (avendo tutti i nodi valore iniziale *'0'*):

$$Pr_A[\text{i decide 1}]=0 \leq \varepsilon level_A(\text{i,r})$$

- **Passo induttivo**. Supponiamo che $leve_A(i,r) = l > 0$ e che la proprietà sia vera $\forall level'_A(i,r) < l$.
  Siccome $level_A = l$, per la sua definizione deve $\exists j : level_{A'}(j,r) \leq l-1$, altrimenti se
  $\forall j, level_A(j,r) > l-1 \rightarrow level_{A'}(i,r) \geq l+1$.
  Se $level_{A'}(j,r) \leq l-1 \rightarrow Pr_{A'}$[j decide 1]$\leq \varepsilon$(l-1), questo per l'ipotesi induttiva.
  Consideriamo adesso $Pr_{A'}$[i decide 1] ed osserviamo che l'esecuzione può produrre due casi:
  - I nodi convergono, da cui si deriva che anche i decide *'1'* ovvero $Pr_{A'}$[i decide 1]$\leq Pr_{A'}$[j decide 1].
  - I nodi non convergono, da cui si deriva che $Pr_{A'}[\text{i decide 1}] \leq \varepsilon$.

Concludiamo quindi che:

$$Pr_{A'}[\text{i decide 1}] \leq Pr_{A'}[\text{j decide 1}] + \varepsilon \leq \varepsilon(l-1)+\varepsilon = \varepsilon l = \varepsilon level_{A'}(i,r) = \varepsilon level_A(i,r)$$

L'ultima ugualianza discende dall'equivalenza rispetto ad i delle due esecuzioni prodotte dagli avversari A e *A'*.

□

Intuitivamente quello che abbiamo appena dimostrato è che se richiediamo contemporaneamente: sia che la probabilità di non arrivare al consenso sia bassa (ε piccolo), sia di mantenere alta la probabilità di soddisfare la condizione di validità (L grande), è necessario eseguire un numero di round elevato.

Riassumendo in questo capitolo abbiamo trattato il problema del consenso a fronte di possibili fallimenti sui collegamenti, cioè abbiamo accettato che dei messaggi possano non essere consegnati a causa di fallimenti nella comunicazione. Il primo risultato prodotto è l'irrisolvibilità del problema dal punto di vista deterministico da cui poi siamo partiti per produrne una soluzione probabilistica. Per tale soluzione abbiamo poi dimostrato l'esistenza di una relazione tra la probabilità ε di soddisfare la condizione di consenso e la probabilità L di soddisfare la condizione di validità che coinvolge il numero di round eseguiti.