

Principles of Cryptocurrency Design

Appunti ed Esercizi

Francesco Pasquale

4 aprile 2024

In questi appunti diamo una dimostrazione del teorema FLP [1]. Si vedano le Sezioni 1 e 2 degli appunti del 28 marzo per le definizioni di *Byzantine Agreement* e modello asincrono e per alcune proprietà di quest'ultimo. In aggiunta, in questi appunti useremo queste ulteriori definizioni.

Dato un protocollo deterministico Π per *Byzantine agreement*, una configurazione $C = (S, M)$ e uno *schedule* $\sigma = (m_1, \dots, m_k)$, diciamo che σ è *applicabile* a C se $m_1 \in M$ e se per ogni $i = 2, \dots, k$, il messaggio m_i appartiene alla *message pool* della configurazione $\sigma_{[1:i-1]}(C)$, dove con $\sigma_{[1:i-1]}$ intendiamo lo *schedule* formato dai primi $i - 1$ messaggi dello *schedule* σ .

Diciamo che una configurazione C è *0-valente* (rispettivamente *1-valente*) per il protocollo Π se, qualunque cosa faccia l'avversario, a partire dalla configurazione C il protocollo fa in modo che tutti i nodi onesti danno in output 0 (rispettivamente 1). Altrimenti diciamo che una configurazione C è *bivalente*.

Lemma 1. Se Π_{BA} è un protocollo deterministico per Byzantine agreement che termina e soddisfa *validity* e *consistency* allora esiste un'assegnazione degli input ai nodi tale che la configurazione iniziale è bivalente.

Dimostrazione. Per $i = 0, 1, \dots, n$, sia X_i la configurazione iniziale in cui gli input dei nodi minori o uguali a i sono 1 e gli input dei nodi maggiori o uguali a i sono 0. Siccome per ipotesi Π_{BA} soddisfa *validity* la configurazione X_0 (in cui tutti gli input sono 0) è 0-valente e la configurazione X_n (in cui tutti gli input sono 1) è 1-valente.

Sia i il più piccolo indice tale che X_{i-1} è 0-valente e X_i non è 0-valente. Mostriamo che X_i non può essere 1-valente, e quindi deve essere bivalente.

Siccome per ipotesi Π_{BA} termina anche in presenza di un nodo corrotto, deve esistere uno *schedule* σ applicabile a X_{i-1} tale che σ non contiene messaggi per il nodo i e $\sigma(X_{i-1})$ è una configurazione in cui tutti i nodi diversi da i che seguono il protocollo hanno deciso il loro output. Infatti, se un tale *schedule* non ci fosse, nel caso in cui i è un nodo corrotto che non fa nulla ogni volta che riceve un messaggio non ci sarebbe nessuno *schedule* che consenta agli altri $n - 1$ nodi di terminare.

Siccome X_{i-1} è una configurazione 0-valente, nella configurazione $\sigma(X_{i-1})$ tutti i nodi diversi da i devono dare in output 0. Siccome le configurazioni X_{i-1} e X_i differiscono solo per l'input del nodo i e lo *schedule* σ non contiene messaggi per il nodo i , anche nella configurazione $\sigma(X_i)$ tutti i nodi diversi da i devono aver deciso che il loro output è 0. Quindi la configurazione iniziale X_i non può essere 1-valente e siccome non è neanche 0-valente per costruzione, deve essere bivalente. $\square$

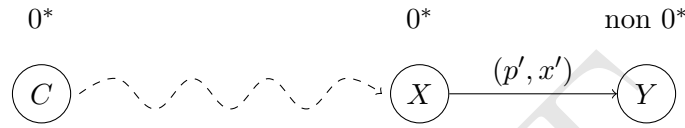
Lemma 2. Se Π_{BA} è un protocollo deterministico per *Byzantine agreement* che termina e soddisfa *validity* e *consistency*, $C = (S, M)$ è una configurazione bivalente per Π_{BA} e $m = (p, x)$ è un messaggio in M , allora esiste uno *schedule* $\sigma = (m_1 = (p_1, x_1), \dots, m_k = (p_k, x_k))$ applicabile a C tale che

1. $m_k = m$ e $p_i \neq p$ per ogni $i = 1, \dots, k - 1$;
2. $\sigma(C)$ è una configurazione bivalente per Π_{BA} .

Dimostrazione. Se $m(C)$ è bivalente, il lemma è dimostrato. Altrimenti $m(C)$ è 0-valente o 1-valente. Consideriamo il caso in cui $m(C)$ è 0-valente; il caso in cui è 1-valente è speculare.

Data una configurazione bivalente $\hat{C}$ per Π_{BA} , diciamo che $\hat{C}$ è di tipo 0^* per il nostro messaggio m se $m(\hat{C})$ è 0-valente. Analogamente diciamo che $\hat{C}$ è di tipo 1^* o bi^* , se $m(\hat{C})$ è 1-valente o bivalente, rispettivamente.

Siccome ci siamo posti, senza perdita di generalità, nel caso in cui $m(C)$ è 0-valente, con la notazione appena definita possiamo dire che C è una configurazione bivalente di tipo 0^* . Consideriamo tutte le configurazioni raggiungibili applicando a C degli *schedule* che non contengono il nostro messaggio $m = (p, x)$. Osservate che fra queste configurazioni ce ne deve essere qualcuna che non è di tipo 0^* , perché se fossero tutte di tipo 0^* allora la configurazione C non potrebbe che essere 0-valente, invece è bivalente per ipotesi. Indichiamo con Y la prima configurazione non 0^* raggiungibile applicando a C uno *schedule* che non contiene il messaggio m , indichiamo con X la configurazione di tipo 0^* che la precede e con $m' = (p', x')$ il messaggio dello *schedule* che secondo il protocollo porta dalla configurazione X alla configurazione Y



Il nostro messaggio $m = (p, x)$ è applicabile alla configurazione Y , perché stava nella *message pool* della configurazione C e non è mai stato consegnato, quindi sta anche nella *message pool* di Y . Sia allora $Z = m(Y)$ la configurazione che si ottiene applicando m a Y . Mostriamo che Z è la configurazione bivalente che stiamo cercando, concludendo così la dimostrazione del lemma.

La configurazione Z non può essere 0-valente, altrimenti Y sarebbe di tipo 0^* , e questo implica che deve essere $p = p'$. Infatti, se fosse $p \neq p'$, la configurazione Z , che si ottiene da X applicando prima (p', x') e poi (p, x) , sarebbe la stessa configurazione che si ottiene da X applicando prima (p, x) e poi (p', x') (vedi l'Esercizio 3 degli appunti precedenti), ma quest'ultima configurazione è sicuramente 0-valente, perché X è di tipo 0^* .

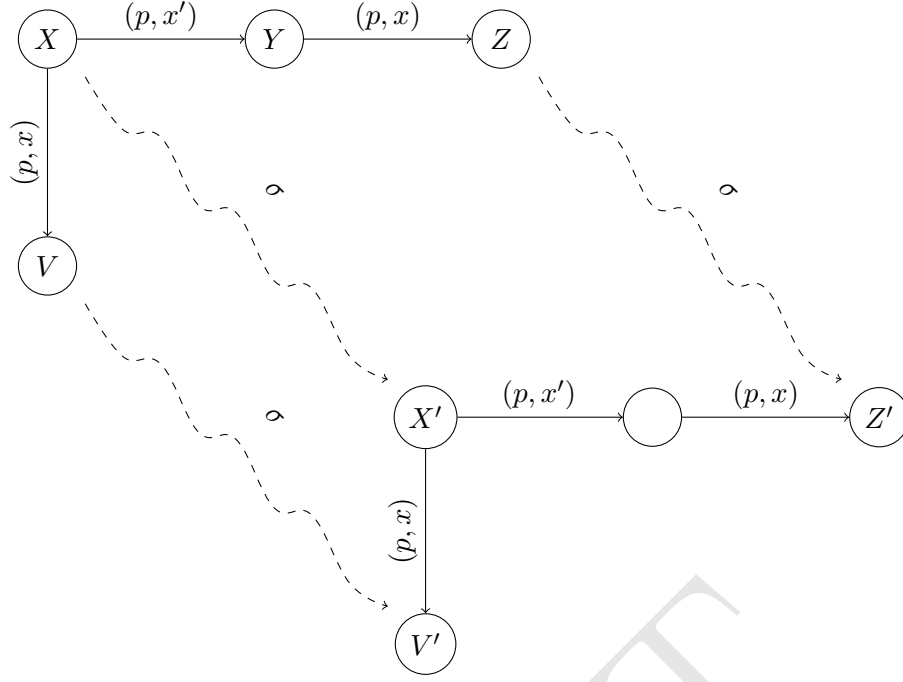
Ora mostriamo che Z non può essere neanche 1-valente esibendo uno *schedule* applicabile a Z che conduce a una configurazione in cui tutti i nodi onesti danno in output 0.

Siccome $p = p'$ e siccome per ipotesi Π_{BA} termina anche in presenza di un nodo corrotto, deve esistere uno *schedule* σ applicabile a X tale che σ non contiene messaggi per il nodo p e $X' := \sigma(X)$ è una configurazione in cui tutti i nodi diversi da p che seguono il protocollo hanno deciso il loro output. Infatti, se un tale *schedule* non ci fosse, nel caso in cui p è un nodo corrotto che non fa nulla ogni volta che riceve un messaggio nessuno *schedule* consentirebbe agli altri $n - 1$ nodi di terminare.

Lo *schedule* σ non contiene messaggi per il nodo p quindi è applicabile anche alla configurazione $m(X)$ e, siccome $p' = p$, anche alla configurazione Z :

1. Siccome $\sigma(X)$ è una configurazione in cui tutti i nodi diversi da p devono aver deciso il loro output, anche $m(\sigma(X))$ lo è. Ma quindi l'output deciso dai nodi deve essere 0, perché $m(\sigma(X)) = \sigma(m(X))$ e $m(X)$ è una configurazione 0-valente dato che X è di tipo 0^* .
2. La configurazione $\sigma(Z) = \sigma(m(m'(X))) = m(m'(\sigma(X)))$ e siccome $\sigma(X)$ è una configurazione in cui tutti i nodi diversi da p hanno deciso che il loro output è 0, anche $\sigma(Z)$ lo è.

Quindi abbiamo trovato uno *schedule* σ applicabile a Z tale che i nodi danno in output 0. Quindi Z non può essere neanche 1-valente e pertanto deve essere bivalente.



□

Teorema 3 (Fischer, Lynch, Paterson [1]). Nel modello asincrono nessun protocollo deterministico per *Byzantine Agreement* che termina in un tempo finito può soddisfare *consistency* e *validity*, se c'è almeno un nodo corrotto.

Dimostrazione. Supponiamo per assurdo che esista un protocollo deterministico Π_{BA} che termina e soddisfa *validity* e *consistency* nel modello asincrono anche in presenza di un nodo corrotto. Per il Lemma 1 deve esistere una configurazione iniziale bivalente $C_0 = (S_0, M_0)$ per Π_{BA} . Per ogni $i \geq 1$, sia m_i il messaggio “più vecchio” presente in M_{i-1} (rompendo la simmetria arbitrariamente in caso di più messaggi), sia σ_i lo *schedule* che termina con m_i la cui esistenza è garantita dal Lemma 2 e sia $C_i = \sigma_i(C_{i-1})$ la configurazione bivalente risultante.

Lo *schedule* $(\sigma_1, \sigma_2, \sigma_3, \dots)$ è quindi uno *schedule* infinito che garantisce che ogni messaggio inserito nella message pool viene prima o poi consegnato e consente di rimanere sempre in configurazioni bivalenti. □

Riferimenti bibliografici

- [1] Michael J. Fischer, Nancy A. Lynch, and Michael S. Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM (JACM)*, 32(2):374–382, 1985.