Principles of Cryptocurrency Design

Appunti ed Esercizi

Francesco Pasquale

16 marzo 2026

Nelle lezioni precedenti abbiamo studiato il problema Byzantine Broadcast (BB) nel modello sincrono, con e senza PKI setup.

In questa lezione introduciamo un problema leggermente diverso, chiamato *Byzantine Agreement (BA)* e il *modello asincrono*, nel quale non assumiamo che ci sia un *clock* globale né che i messaggi arrivano a destinazione nello stesso ordine nel quale sono partiti.

1 Byzantine Agreement

Nel problema *Byzantine Agreement (BA)* abbiamo n nodi di cui f sono corrotti. A ogni nodo i viene affidato in input un messaggio b_i. Vogliamo progettare un protocollo, che termini in un tempo finito, in cui ogni nodo onesto i dia in output un valore y_i tale che

- **Consistency**: Se i e j sono due nodi onesti, allora $y_i = y_j$;
- **Validity**: Se tutti i nodi onesti hanno in input lo stesso bit b, allora $y_i = b$ per ogni nodo onesto i.

Esercizio 1. Osservare che il problema Byzantine Agreement può essere risolto solo se il numero di nodi corrotti è $f < n/2$: Mostrare che dato un qualunque protocollo Π, se ci sono $f \geqslant n/2$ nodi corrotti questi possono sempre fare in modo che venga violata la condizione di *validity*.

Esercizio 2. Modificando opportunamente il protocollo di Dolev-Strong per il problema Byzantine Broadcast, mostrare che nel modello sincrono con PKI setup esiste un protocollo per Byzantine Agreement che soddisfa consistency e validity qualunque sia il numero di nodi corrotti $f < n/2$.

Esercizio 3. Modificando opportunamente la dimostrazione di impossibilità del problema Byzantine Broadcast con $n/3$ o più nodi corrotti, mostrare che senza PKI setup non esiste nessun protocollo per Byzantine Agreement che soddisfa consistency e validity in presenza di $n/3$ o più nodi corrotti.

2 Modello asincrono

Nel modello asincrono non abbiamo un *global clock* e non abbiamo un *bound* sul tempo che impiega un messaggio inviato da un nodo a raggiungere il nodo di destinazione.

Un *protocollo* nel modello asincrono è *event driven*, cioè specifica cosa fa un nodo nel momento in cui riceve un messaggio: il nodo può eseguire un numero arbitrario di operazioni locali, cambiare *stato* (lo *stato* di un nodo è costituito dai valori delle sue variabili locali, che possono dipendere dall'input e da tutti i messaggi ricevuti). Una configurazione C è una coppia $C = (S, M)$ dove S

rappresenta lo stato di ogni nodo e M è la *message pool*, ossia l'insieme di messaggi non ancora consegnati. Un messaggio m è una coppia $m = (p, x)$ dove p è il nodo destinatario del messaggio e x è il contenuto del messaggio.

Se Π è un protocollo deterministico nel modello asincrono, data una configurazione $C = (S, M)$ e un messaggio $m = (p, x) \in M$, indichiamo con $m(C)$ la configurazione $C' = (S', M')$ che si ottiene in base al protocollo quando il messaggio m viene consegnato, ossia l'insieme S' è quello che si ottiene da S aggiornando lo stato del nodo p dopo l'elaborazione delle informazioni contenute in x e M' è la *message pool* che si ottiene da M togliendo il messaggio appena consegnato m e aggiungendo tutti i messaggi inviati da p a seguito della ricezione del messaggio m.

Esercizio 4. Osservare che, dato un protocollo deterministico Π, una configurazione $C = (S, M)$ e due messaggi $m_1 = (p_1, x_1), m_2 = (p_1, x_2) \in M$, se $p_1 \neq p_2$ allora $m_2(m_1(C)) = m_1(m_2(C))$.

Uno *schedule* σ è una sequenza ordinata di messaggi $\sigma = \big((p_1, x_1), \ldots, (p_k, x_k)\big)$. Diciamo che uno schedule σ è *applicabile* a una configurazione C se $m_1 \in M$ e se per ogni $i = 2, \ldots, k$, 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. Dato un protocollo Π, una configurazione C e uno schedule σ, indichiamo con $\sigma(C)$ la configurazione che si ottiene partendo da C in base al protocollo Π dopo la consegna di tutti i messaggi in σ (nell'ordine stabilito dalla sequenza).

Esercizio 5. Osservare che, dato un protocollo deterministico Π, una configurazione $C = (S, M)$ e due schedule $\sigma_1 = \big((p_1, x_1), \ldots, (p_k, x_k)\big)$ e $\sigma_2 = \big((q_1, y_1), \ldots, (q_h, y_h)\big)$ applicabili a C, se gli insiemi di nodi destinatari nei due schedule sono disgiunti, $\{p_1, \ldots, p_k\} \cap \{q_1, \ldots, q_h\} = \emptyset$, allora $\sigma_2(\sigma_1(C)) = \sigma_1(\sigma_2(C))$.

Nella prossima lezione vedremo nei dettagli una dimostrazione del teorema seguente. Qui ci limitiamo a dare una idea della dimostrazione e a dimostrare uno dei due lemma chiave.

Teorema 1 (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.

Sketch of Proof. La dimostrazione procede dimostrando che se Π è un protocollo deterministico per Byzantine Agreement che soddisfa validity e consistency allora esiste una strategia dell'avversario che fa in modo che il protocollo non termini mai.

Se Π_{BA} è un protocollo per Byzantine Agreement che soddisfa validity e consistency, 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*.

Informalmente, la dimostrazione procede come segue:

1. Si dimostra che esiste una assegnazione degli input ai nodi tale che la configurazione iniziale C_0 è bivalente;

2. Si dimostra che se C è una configurazione bivalente, allora l'avversario può fare in modo che, partendo da C, il protocollo raggiunga una nuova configurazione bivalente C'.

Dai punti 1 e 2 segue che l'avversario può fare in modo che il protocollo Π_{BA} non termini mai. Dimostriamo il punto 1 nel Lemma 2. Vedremo la dimostrazione del punto 2 nella prossima lezione. □

Lemma 2. 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 $i = 0, 1, \ldots, n$, 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 di a i sono 0. Siccome per ipotesi Π_{BA} soddisfa *validity* la configurazione X_0 (in cui tutti gli input sono 0) deve essere 0-valente e la configurazione X_n (in cui tutti gli input sono 1) deve essere 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. □

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. `https://dl.acm.org/doi/pdf/10.1145/3149.214121`.