DISCLAIMER

Il materiale contenuto nel drive è stato raccolto e richiesto tramite autorizzazione ai ragazzi frequentanti il corso di studi di Informatica dell'Università degli Studi di Salerno. Gli appunti e gli esercizi nascono da un uso e consumo degli autori che li hanno creati e risistemati per tanto non ci assumiamo la responsabilità di eventuali mancanze o difetti all'interno del materiale pubblicato.

Il materiale sarà modificato aggiungendo il logo dell'associazione, in tal caso questo possa recare problemi ad alcuni autori di materiale pubblicato, tale persona può contattarci in privato ed elimineremo o modificheremo il materiale in base alle sue preferenze.

Ringraziamo eventuali segnalazioni di errori così da poter modificare e fornire il miglior materiale possibile a supporto degli studenti.

CoScienze
Associazione

Consenso di Paxos

Paxos è un protocollo che serve a raggiungere il consenso su un valore, e a mantenerlo finché il protocollo è in esecuzione. I nodi che partecipano a Paxos possono avere ruoli diversi, e l'esecuzione del protocollo è suddivisa in turni.

Proprietà del consenso di Paxos:
Liveness:

1. Uno tra i valori proposti prima o poi viene scelto.
2. Se un valore viene scelto, ogni processo prima o poi apprenderà tale scelta.

Safety:

1. Un valore può essere scelto solo tra quelli proposti.
2. Il valore scelto deve essere unico.
3. Un processo non deve mai apprendere che un valore è stato scelto a meno che esso non sia stato effettivamente scelto.

Ruoli dei processi

Abbiamo tre tipi di nodi:

- **Proposer**: sceglie un valore da votare e lo propone a tutti.
- **Acceptor**: riceve la proposta dal proposer, la vota oppure no.
- **Learner**: memorizza il valore votato dalla maggioranza.

Esecuzione del protocollo

Paxos funziona così:

1. Un **proposer** avvia un turno di votazione, inviando a tutti un messaggio che chiamiamo prepare.
2. Gli **acceptor** che lo ricevono, possono rispondere con una promise, impegnandosi quindi a votare per quel turno.
3. Il **proposer** sceglie un valore e lo manda a tutti con un accept.
4. Gli **acceptor** dicono ai **learner** di imparare quel valore con un learn.

Per spiegare meglio Paxos è possibile prendere in esempio una richiesta di update, in cui *V* è il valore da registrare. La richiesta di update viene effettuata da un *client* verso il *proposer*.

Questo a sua volta la comunica agli *acceptor* inviando un messaggio *prepare*. Il messaggio di *prepare* è identificato da un numero. Gli *acceptor* rispondono al *proposer* promettendo che non accetteranno altre *prepare* con identificatori inferiori (messaggio *promise*). Nel momento in cui il proposer rileva un «numero sufficiente» di promesse, invia agli *acceptor* il messaggio di *accept*. Questi rispondono con il messaggio *accepted* al proposer, ed inoltrandolo anche ai *learner*, ovvero ai processi che devono fisicamente eseguire le registrazione. A loro volta i learner, quando ricevono un «numero sufficiente» di messaggi *accepted*, eseguono l'operazione richiesta.

Effettuata la registrazione, i *learner* inviano al *client* il messaggio di *ok*.

Questo è un protocollo completamente distribuito, in quanto la sua ossatura è formata dagli acceptor (che spesso sono collassati con i learner). Paxos non è usato solo per mantenere consistenza tra le repliche, ma soprattutto per la gestione del ***fault tolerant***.

Infatti, Paxos funziona fino al guasto del 50% degli acceptor. Aumentare il numero di acceptor aumenta i costi (più macchine) a fronte di un aumento di resistenza ai guasti.

Il tipo di fallimento che Paxos prevede è il fault con stop. Ovvero la macchina guasta non dà ulteriori comunicazioni. Se si devono considerare guasti che prevedono l'invio di comunicazioni scorrette, Paxos non è utilizzabile; bisogna utilizzare Byzantine.