CSCI 251: Concepts of Parallel and Distributed Systems

Alvin Lin

November 1st, 2017

Agreement and Consensus

Topics:

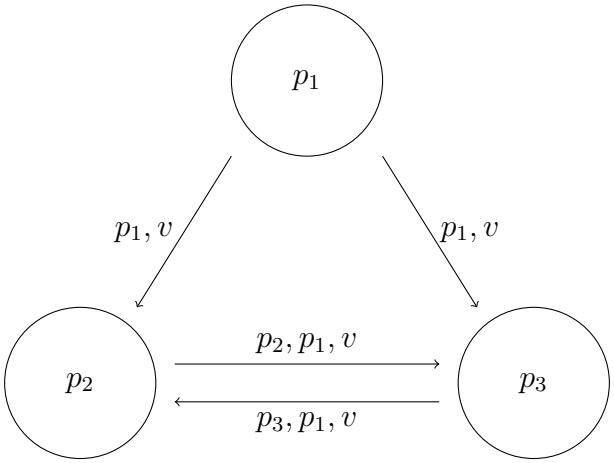
- Agreement Algorithms:
  - Consensus
  - Byzantine Generals
  - Interactive Consistence
- Transactions

Consensus

Suppose there are decision variables. At the termination of the agreement algorithm, all the processes should agree on the decision variable. For processes p_1 to p_n, each process proposes a value v_1 to v_n. At the end of the consensus algorithm, all the processes should agree on one decision value. This can be done as either the majority value, the max value, the min value, or the average.

Byzantine General's Problem

There is a distinguished process which decides the decision variable and every process sets their decision value to the one suggested by the distinguished process.



In the case where a propagated decision variable is faulty or corrupted, this degrades into making a decision via consensus instead. It is impossible to identify the number of faulty processes f out of n processes if:

$$n \leq 3f$$

Interactive Consistency

In interactive consistency, each process proposes a value, which is its contribution to the vector of decision values. At the termination of the algorithm, all processes agree on the vector of values.

Transactions

Transactions are series of a read/write operations on a server. A transaction has to satisfy four properties.

- Atomicity - all or nothing, either the operation completes or it does not complete, without a middle state.
- Consistency - the transaction process moves from one consistent state to another consistent state.
- Isolation - there should be no interference from other transactions.
- Durability - transaction objects must be recoverable and stored in permanent memory.

Example

Banking Accounts A, B, C:

$$A = \$100; \quad B = \$200; \quad C = \$300$$

Transaction T	**Transaction U**
`balance = b.getBalance();` $200	
	`balance = b.getBalance();` $200
	`b.setBalance(balance * 1.1);` $220
`b.setBalance(balance * 1.1);` $220	

A situation like this is problematic without ACID since transaction U's update is lost.

$$A = B = \$200;$$

Transaction V	**Transaction W**
`a.withdraw(100);` $100	
	`total = a.getBalance();` $100
	`total += b.getBalance();` $300
`b.deposit(100);` $300	

In this situation, W does not see the deposit done by V and only sees the withdrawal action. These issues are solved by:

- Serialization - each transaction should appear as if it is the only transaction on the server.
- Locking - Inefficient and creates a bottleneck.
- Optimistic Concurrency - Most efficient, uses multiple phases to resolve transactions.
- Timestamping - Resolves transactions in time stamp order.

Reminders

Check MyCourses for details on Project 2.
Professor Mohan Kumar:
`mjkvcs@rit.edu`
`https://cs.rit.edu/~mjk`

Rahul Dashora (TA):
rd5476@mail.rit.edu

You can find all my notes at http://omgimanerd.tech/notes. If you have any questions, comments, or concerns, please contact me at alvin@omgimanerd.tech