

CS 262

Introduction to Distributed Computing

Jim Waldo
Ian Rose

Administrivia

- Assignments
 - Don't ask for acknowledgement
 - Ack will be getting the paper back
 - Email is reliable (enough)
 - Programming assignment 0 up
 - Assignment 1 is coming soon
- Schedule update
 - May not be a class 2/20
 - Looking for a guest lecturer

Replication Intuitions

Reliability Through Redundancy

- Multiple copies of every part of the system
 - All parts exactly the same
 - If one fails, others can take over
 - How many replicas depends on type of failure protection
 - Explored through Isis, Horus, ...
- Strong, peer replication
 - Master/slave replication
 - Weak consistency

t-fault Tolerance

- Any system will fail
 - Meteors hit the earth
 - The universe will decay
- Design for tolerance to some number of faults
 - If less than this number fails, the system is fine
 - Otherwise, all bets are off

Cost of Fault Tolerance

- Reliability has a cost
 - Depending on the kind of fault
 - Failstop - cheapest
 - Byzantine - most expensive
- Cost to be t fault tolerant
 - Failstop - $t + 1$ machines
 - Byzantine - $2t + 1$ machines (or more)

Replica Insurance

- Each replica is an implementation of the same state machine
 - The externally visible state is determined by
 - The initial state
 - The inputs
- The implementations could be different
 - To avoid software faults, they should be

Memory

```
memory : state_machine  
  var store : array [0..n ] of word  
  read : command(loc : 0..n )  
    send store [loc ] to client  
  end read ;  
  write : command(loc : 0..n , value : word)  
    store [loc ] := value  
  end write  
end memory
```

Equivalently...

```
interface RemoteMemory{
    /**
     * returns the array of bytes stored at index indicated by
     * position
     */
    byte[] read(int position);
    /**
     * writes a copy of contents at location position
     */
    void write (int position, byte[] contents);
}
```

Semantics Digression

- **Meaning of a method**
 - Hard to specify in isolation
 - Often requires reference to other methods
- **Interface as a semantic unit**
 - Shouldn't have to refer to other interfaces for meaning
 - Should have to refer to other methods in the interface
- **Especially in distributed/concurrent systems**

Network Guarantees

- If all state machines are the same, replication requires
 - The same messages to all replicas
 - In the same order
- Reliable Multicast
 - How to order the messages
 - How to insure they all get everywhere

Network Ordering

- Hold delivery until all previous messages delivered
 - Received/delivered distinction
 - Requires ordering all messages
- Total ordering unnecessary
 - Causal ordering sufficient
 - Even that can be relaxed
 - Idempotent
 - Commute

Logical Clocks

- A logical clock is a counter
 - Every process keeps its own
- Value of logical clock is sent with every message
- For any event, current clock value is
 - $\max(\text{myValue}, \text{ValueOnMessage}) + 1$
- If e_1 is causally related to e_2 , then either
 - $LC(e_1) < LC(e_2)$
 - $LC(e_1) > LC(e_2)$
 - $e_1 = e_2$

When to Deliver

- Need to know
- No missing messages
 - Wait for timeout
 - Get a message with a higher value from all processes
 - Assumes no loss in channels, just delays
- Chatty, but it works
- But it doesn't scale

Order by Master

- Send all messages to one machine
 - That machine will order the messages
 - Send to everyone else
- If that machine fails
 - Find a new leader
 - Tell everyone
 - Continue
- Leader election
 - Lots of uses

Simple Leader Election

- Everyone has a Unique ID
- Asynchronous mechanism
 - Everyone send to everyone else their UID
 - Lowest UID is the leader
- Synchronous version
 - Wait UID time units for a nomination
 - If you receive a nomination, accept it
 - If not, send a nomination of yourself

Replication Costs

- Networking
 - Asynchronous
 - Lots of traffic
 - Synchronous
 - Slowest link
- Machines
 - Gated by the slowest processor/implementation
 - Reads are fast, writes are slow

Weak Replication

- Weakly replicated systems
 - Update sent to any replica
 - Update time-stamped, sent to other replicas
 - Goal of eventual consistency
 - Requires idempotent/commuting operations
- Highly efficient for mostly read systems
- Difficult to administer
- Can lead to puzzling behavior

Data Replication

- Databases generally have replicated versions
 - Much slower
 - Harder to administer
 - Not generally used
- Replication for disaster recovery
 - Geographically distributed
 - Slows update
 - Often lag between master and slave
- Continued hard problem

Service Replication

- Services available on the network
 - Data sent to the service
 - Service performs some computation
 - Results returned
 - Service essentially stateless
- Natural for the web, highly available
 - Amazon
 - Orbitz
- Allows other features
 - Versioning

Wednesday

- Birrell and Nelson, Implementing RPC
- Programming Assignment 0
- Programming Assignment 1