

CS 262

Introduction to Distributed Computing

Jim Waldo
Ian Rose

Administrivia

- Assignments
 - Will appear on the web site
 - If there is nothing on the web site, there is no assignment
- Readings update
 - There is now a reading for Wednesday

Some Vocabulary

- Synchronous vs. Asynchronous (systems)
 - In a synchronous system, there are bounds on
 - Time of computation
 - Time of message deliver
 - An asynchronous system is one that is not synchronous
- Synchronous systems have extra failure modes
- Everything that can be proved of an asynchronous system is true in a synchronous system

More Vocabulary

- Synchronous vs. Asynchronous (calls)
 - In a synchronous call, you wait for the call to return before continuing execution
 - In an asynchronous call, you continue your thread of execution and are informed of the call result later

Safety and Liveness

- Safety is a relation between a specification and an implementation
 - An implementation is safe if no state of the implementation is not also a state of the specification
 - Nothing bad happens
- Liveness is a property of an implementation
 - An implementation is live if it makes progress
 - Something good happens
- Safety can be proved. Liveness can't

Theory vs. Practice

- Theory concentrates on
 - Asynchronous systems
 - Proving safety
 - Proof = a formal derivation
- Actual systems
 - Have some form or synchrony
 - Worry about liveness
 - Rarely have well-defined specifications
 - Proof = running system without (known) bugs

Value of Specifications

- Basis of proofs
 - Proof requires great detail
 - Good for parts of a large system
- As documentation
 - Where alternate implementations are possible
 - Abstract the essence of the system
- Design tools
 - Specs are easier to change than code

Levels of Specification

- Highly formal, very detailed
 - Next Monday
- Interface level
 - All interfaces specified, along with their interactions
 - Semantics of the interface described (informally)
 - Pre- and post-conditions described (informally)
- Informal
 - A description of the system
- Goal: be able to translate from level to level

Not Specifications

- Source code
 - No way to distinguish
 - What is essential
 - What is accidental
- User documentation
- Programmer documentation

Failure

- Computers fail
 - Failstop
 - Crash
- Networks Fail
 - Send omission
 - Receive omission
 - General omission
- Synchronous systems take too long
 - Timing failures

Failure II

- Software fails
 - Byzantine failure
- Failure hierarchy
 - Crash, Failstop : simplest
 - Send, Receive, General omission : harder
 - Timing failures : hard, but generally ignored
 - Byzantine : hardest
- All but byzantine considered benign

New Failure Models

- Gaming
 - Using the rules to your advantage
- Selfishness
 - Taking, but not giving back
- Reflect a move from
 - Mutually trusting systems
 - Built by one person or group
- to
 - Untrusting aggregations
 - Built on the fly

Programming

- Four key problems
 - Latency
 - Data sharing/passing
 - Concurrency
 - Partial Failure

Latency

- **c is the law**
 - Would that we were only bound by c
 - Software gets in the way
 - Topology gets in the way
 - Latency is variable
- **Partitioning systems**
 - Minimize calls
 - Move to where things are needed
 - Move data or computation?
- **Latency overwhelms other operations**

Data Sharing

- **Difficult to pass a reference**
 - Assume all data is distributed
 - Puts latency in the path of everything
- **Copy semantics is**
 - Natural
 - Leads to interesting programs

Concurrency

- Servers need to be concurrent
- Everyone is a server
- Everyone needs to be concurrent
- Concurrency is difficult

Partial Failure

- How to deal with a failed call
 - Idempotent calls
 - Replicas
 - Often application specific
- Asynchronous calls make this harder
 - When does the failure happen?
 - How to unwind to failure point
- We will stick to synchronous calls

Convention

- Rules of what to do when communication fails
 - Phone calls
 - Meeting someone
 - Classes
- Conventions in system design
 - Timing
 - Retry
 - Others...

Historical Precedent: Trains

- Latency
 - Trains don't always run on time
- Data sharing
 - How to insure orders were received
- Concurrency
 - Multiple trains per track
- Partial Failure
 - One train failing did not stop others

Train Conventions

Timing Conventions

- Coordinating time
 - Time zones
 - Synchronizing watches
- Timeouts
 - 12 hour schedule slush
 - 5 minute station rule

Safety Conventions

- Flagmen
- Time tables
- Priorities
- Balancing safety and liveness

Communication Conventions

- Written vs. verbal communication
- Idempotent instructions
 - No cancellations
- Rules for when a reply is needed

Others?

Wednesday Reading

- Birrell, *An Introduction to Programming with Threads*
 - Describes Modula-3 threads
 - Translates directly to Java
- Wednesday we will talk about building a distributed system
 - Look at Java sockets