Csc72010

Parallel and Distributed Computation and Advanced Operating Systems

Lecture 1

January 26, 2006

Business

Introduce myself & research interests
Security seminar
Class roll

Go over syllabus & calendar

Every Thursday except April 13 and April 20, until May 11
Midterm exam on March 16;
Project plan on April 6;
Project due on May 18 (Reading Day)

The course is preparation for one of the sections of the first exam

Course Description

What's different about a distributed system?

Bank example

If we think about programming transactions coming into a centralized bank computer, we can assume it would look like this:

A withdraws $100 from 1 (refused)
B withdraws $100 from 2 (accepted)
C deposits $1000 in 1 (accepted)
D withdraws $50 from 2 (refused)

Not true at an ATM – let's assume constant communication between ATM's and banks (not necessarily true):

A begins transaction on account 1

B begins transaction on account 2
B requests withdrawal of $100

C begins transaction on account 1
C requests deposit of $1000

D begins transaction on account 2
D requests withdrawal of $50

Bank adds $1000 to 1
A requests withdrawal of $100
Bank checks that balance > $100
Bank subtracts $100 from 1
Bank dispenses $100 cash to A

Bank checks that balance > $100
Bank checks that balance > $50
Bank subtracts $100 from balance
Bank dispenses $100 cash to B
Bank subtracts $50 from balance
D's ATM fails!!!

Distributed System Problems:

The bank example illustrates all of the problems
Concurrency: multiple people acting on the same object at the same time – order of activities must be controlled
Partial failure: The bank subtracted the total withdrawals requested from account 2, but didn't dispense all of the money
Time: A's request is either accepted or rejected depending on how fast his transaction goes relative to C's. This makes correctness harder to state.

Papers:
Leslie Lamport, "Time, Clocks, and the Ordering of Events in a Distributed System," Communications of the ACM, July 1978, 21(7):558-565.
Network Time Protocol (Version 3) Specification, Implementation. D. Mills. March 1992.

Global state: Consider spreading the state around, so that the ATM's have the balances and don't have to go to a central site. This makes matters worse – we will learn later that in theory at least there is no guarantee that you can determine "The global state" – instead, there may be many possible global states consistent with a sequence of actions.

Michael J. Fischer, Nancy D. Griffeth, Nancy A. Lynch: Global States of a Distributed System. IEEE Transactions on Software Engineering, 8(3): 198-202 (1982).
K. Mani Chandy, Leslie Lamport: Distributed Snapshots: Determining Global States of Distributed Systems ACM Trans. Comput. Syst. 3(1): 63-75 (1985).

Waldo, Note on Distributed Computing

Solution is centralized – not what we'll be looking at.

Internet: no single node is in control (although we often end up selecting one).

Prototypical distributed problems

Network is a graph, communication links are edges in the graph, network devices are nodes, which we call processes.

Pick a leader (aka leader election): assume all processes identical – how can they select one to be the controlling process?

Broadcast communication: make sure everyone gets a message

*Routing: decide what routes messages should use in the network

Failure recovery/reliability:
    *make sure messages reach their destination
    one node fails, another takes over its function

Agreement (everybody does the same thing): commitment protocols

Resource allocation: make sure that a resource is given to at most one user, and a user requesting a resource gets one if it is available (fairness?)

Approach

Step 1. Observe how problems are solved by Internet protocols; consider the environment and the requirements, the design goals and the intuition behind the protocol.

Step 2. Model and prove properties of protocol. Make assumptions about environment; decide on algorithm requirements. (Sometimes) do some complexity analysis (number of messages, time).
    Language is I/O automata: simple procedural commands, composition, tools for simulation and proof

Go over syllabus again to see what we'll be doing in the course:
    Link layer, IP layer before midterm
    Finish IP layer, TCP, project after midterm

Environmental Assumptions

How does communication take place?
    Message-passing
Timing?
    Asynchronous: any time
Failures
    Processors: stopping or Byzantine (we'll do stopping only)
    Communication: lost messages
        Duplicate messages
        Out of order messages
        Channel failure
        Network partitions
We'll usually start with simplifying assumptions, solve the problem, then alter the assumptions.

Typical Requirements

Functional correctness

Atomicity
Correct resource allocation
Message delivery

Reliability

Guaranteed message delivery
No duplicates
In-order messages
Server uptime

Availability

Uptime/downtime

Maintainability

Network management
Network configuration
Network monitoring

Performance

Response time
Throughput
Utilization
Congestion
Usual approach: performance modeling, queuing theory