Introduction to distributed computing

Paulo Sérgio Almeida

Distributed Systems Group
Departamento de Informática
Universidade do Minho

Distributed algorithms

- Algorithms that run on several nodes connected by network;
- Irrelevant whether WAN, LAN, . . .
- Broader definition can even include shared memory algorithms;
- Some key attributes:
  - Interprocess Communication (IPC) Model;
  - Timing model;
  - Failure Model;

Interprocess communication model

- Shared memory;
- Point-to-point messages;
- Broadcast messages;

Timing model

- Many possible timing assumptions;
- One extreme: completely synchronous

  communication and computation, in lock-step;
- Another: completely asynchronous

  arbitrary relative speeds, arbitrary order;
- In between: partial synchrony assumptions

  e.g. bounds on relative speeds or communication delay

Failure model

- Algorithms may be designed to tolerate some faults.
- Processsor failures:
  - processors stop;
  - transient failures;
  - Byzantine Failures, with arbitrary behavior;
- Communication failures:
  - message loss;
  - message duplication;

Working under uncertainty

- Many interesting algorithms do not make many assumptions regarding environment;
- Examples:
  - unknown number of processors;
  - unknown network topology;
  - programs starting at different times, operating at different speeds;
  - unknown message delivery times;
  - unknown message ordering;
  - processor and communication failures;

Understanding distributed algorithms

- Uncertainty leads to difficulty in understanding;
- Normal difficulties in concurrency (like arbitrary interleavings)
- Plus asynchrony and failures;
- Cannot predict what exactly will happen;

  many behaviors even for same inputs

- But understand properties;
  - correctness;
  - complexity and lower bounds;
  - impossibility results;
- Underlying this are mathematical models of distributed systems;

Approach

- Concentrate on essential problems;
- Organization according to system models;

Concentrate on essential problems

- Field is very large;
- But some fundamental problems recur in many applications;
- Examples:
  - leader election;
  - network searching;
  - spanning tree construction;
  - consensus;
  - mutual exclusion;
  - resource allocation;
  - global snapshots;
  - reliable communication;

Organization according to system models

- Consider same problems in different system models;
- What causes most impact are the timing models;
- Timing models used for top-level organization;
  - synchronous model;
  - asynchronous model;
  - partially synchronous model;

Synchronous model

- Execution proceeds in synchronous rounds;
- Simplest model to program and reason about;
- Provides insight to solve problems in asynchronous models;
- Strongest model; impossibility results apply to other models;

Asynchronous model

- Components take steps in arbitrary order at arbitrary speeds;
- Harder to program due to more uncertainty;
- Weakest model; more problems unsolvable;
- Algorithms are more general and work in other models;

Partially synchronous model

- Some assumptions can be made about relative timing of events;
- Most realistic model;
- Most difficult to program;
- Efficient but fragile algorithms if assumptions violated;

Course Outline

- Synchronous networks:
  - Formal model (lockstep rounds) and proof methods
  - Basic algorithms: Leader Election
  - Agreement with process and link failures
- Asynchronous networks:
  - Formal models (I/O automata) and proof methods
  - Basic algorithms: (revisited)
  - Logical time and State-machine simulation
- Agreement in asynchronous networks:
  - Impossibility of fault-tolerant consensus
  - Failure Detectors and Indulgence
  - Unreliable communication channels
  - Agreement problems: Distributed commit, Atomic broadcast
- Timed/Hybrid Asynchronous networks:
  - Formal model (timed I/O automata) and proof methods
  - Clock synchronization and Failure Detectors implementation
  - Timeliness and Real-time guarantees