Introduction to distributed algorithms

- definition
- relation to practice
- models
- main concepts
- causality relationships
- wave algorithms
- ring algorithm
- tree algorithm
- echo algorithm
- polling algorithm

What is distributed system

- Distributed system is a collection of independent processes that communicate with each other and cooperate achieving a common goal
- each process has its own independent set of instructions and can proceed with its own speed
- the only way for one process to coordinate with others is via communication
- thus the system consists of a set of processes connected by a network of communication links

Why distributed systems

- distributed system is a convenient abstraction to model various practical computer architectures:
  - multiprocess OS
  - multiprocessor computer architecture
  - local area network
  - Internet
  - VLSI chip
- the abstraction of the model allows designing algorithms that are correct irrespective of implementation details

Model classification

- By time
  - (fully) synchronous model - processes take steps simultaneously (execution proceeds in synchronous rounds
  - partially synchronous model - processes proceed independently but some timing information (such as the maximum difference between the slowest and fastest process) is known
  - fully asynchronous model - processes take steps in arbitrary order with arbitrary speeds
- By communication primitives
  - message-passing - processes do not share memory, connected by channels, communicate by sending and receiving messages
  - shared-memory - processes share registers to which they can apply read/write or more complex operations (like test/and set)
- note that unlike PRAM the order of operations on shared memory objects is non-deterministic

Uncertainty in distributed systems

- Algorithm designers may have to deal with some of these failures
  - unknown number of processes
  - unknown network topology
  - independent inputs at different locations
  - processes may start at different times and execute with different speeds
  - uncertain message delivery times, unknown message ordering
  - unknown order of operations on shared memory objects
  - processor and communication failures

Main Concepts

- Assignment of values to all variables in distributed system is a (global) state of distributed system. Let Z be a set of system's states
  - if the system is message passing the state includes an assignment of messages to communication channels
  - an atomic operation takes the system from one state to another
  - a distributed algorithm defines a system, a set of initial state and a set of possible atomic operations
  - an operation is enabled at a certain state if it can be executed at that state
  - the execution of an action is an event
  - a computation is a sequence of states such that the first state is an initial state and each consequent state is obtained by executing an enabled action at the preceding state
Main Concepts (cont.)

- A computation is **weakly fair** if no action is enabled in infinitely many consequent states.
- An assertion is a binary predicate that maps every state of the system to **true** or **false**.
- Properties of the system:
  - **Safety** - a certain assertion is true in every state of every computation of the algorithm; this assertion is called invariant.
  - **Liveness** - every computation contains a state where a certain assertion is true.
- Every algorithm’s property can be decomposed to safety and liveness.

Causality relationship

- An event is usually influenced by part of the state.
- Every computation contains a state where a certain assertion is true.
- Properties of the system:
  - **Safety** - a certain assertion is true in every state of every computation of the algorithm; this assertion is called invariant.
  - **Liveness** - every computation contains a state where a certain assertion is true.
- For shared memory systems:
  - A read and a write on the same data item are causally related.
  - Is transitive.
  - Is a partial order.
  - If not a \(a\) or b \(b\) then a and b are concurrent: a\(a\), b\(b\).

Wave Algorithms

- Wave algorithm satisfies the following three properties:
  - **Termination**: Each computation is finite.
  - **Decision**: Each computation contains at least one decide event.
  - **Dependence**: In each computation, each decide event is causally preceded by an event in each process.

  - **Initiator** (starter) - process that execution of its actions spontaneously begins.
  - **Non-initiator** (follower) - starts execution only when receives a message.

  - Wave algorithms differ in many respects, some features:
    - Centralized (single-source) - one initiator; decentralized (multi-source) - multiple initiators.
    - Topology: ring, tree, clique, etc.
    - Initial knowledge:
      - Each process knows its own unique name.
      - Each process knows the names of its neighbors.
    - Number of decisions to occur in each process.
    - Usually, wave algorithms exchange messages with no content - tokens.

Ring algorithm

For the initiator:

```
begin send (tok) to Next[0]; receive (tok) ; decide end
```

For non-initiators:

```
begin receive (tok) ; send (tok) to Next[0] end
```

Processes are arranged in a unidirectional ring (each process has a sense of direction or knowledge of one dedicated neighbor).

- Initiator send message 'tok' along the cycle.
- Each process passes it on.
- When it returns to initiator, initiator decides.

Theorem: Ring algorithm is a wave algorithm.

Tree algorithm

- Operates on tree network (can work on spanning tree of arbitrary network).
- Leaves of tree initiate the algorithm.
- If a process has received a message from all neighbors but one (initially true for leaves), the process sends a message to the remaining neighbor.
- If process gets messages from all neighbors - it decides.
- Excluding the **forall** statement, how many processes can decide?
- How many messages are exchanged in the algorithm?
- Is this a wave algorithm?

Chang’s Echo algorithm

- Works on networks of arbitrary topology.
- One initiator (centralized).
- Initiator sends messages to all neighbors.
- When non-initiator receives the first message, it forwards it to all other neighbors. If it gets tokens from all neighbors, it replies back.
- How many processes decide?
- How many messages are exchanged in the algorithm?
- What other topology can this algorithm be used in?
- Is this a wave algorithm?
Polling algorithm

```plaintext
var n0: integer := 0; (* for initiator only *)

For the initiator:
    begin forall q in N0, do send (tok) to q;
    while n0 < p do q, do begin receive (tok); n0 := n0 + 1 end;
    decide
    end

For non-initiators:
    begin receive (tok) from q; send (tok) to q end
```

- Works on cliques (complete networks)
- one initiator (centralized)
- how many processes decide?
- How many messages are exchanged in the algorithm
- what other topology can this algorithm be used in
- Is this a wave algorithm?