Introduction to distributed algorithms

- definition
- relation to practice
- models
- main concepts
  - states and actions
  - computations and properties
  - 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&set)
    - note that unlike PRAM the order of operations on shared memory objects is non-deterministic

States and Actions

- 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 is an operation that cannot interleave with other operations
  - 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
  - fixpoint (quiescent state) – a state where none of the actions are enabled

Computations

- 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
  - a computation is either infinite or ends in a fixpoint
  - A computation is weakly fair if no action is enabled in infinitely many consequent states
  - prefix – a finite segment of a computation that starts in an initial state and ends in a state
  - suffix – a computation segment that starts in a state and is either infinite or ends in a fixpoint
    - a computation is obtained by joining a prefix and a suffix
  - can two separate computations share a prefix? a suffix? both?
Program Properties

- property – a set of correct computations
- safety – there is a set of prefixes none of which a correct computation may contain (something "bad" never happens)
- liveness – for every prefix there is suffix whose addition creates a correct computation (something "good" eventually happens)
- every property is an intersection of safety and liveness properties

Example:
- mutual exclusion problem (MX) – a set of processes alternate executing critical section (CS) of code and other code

- a solution to mutual exclusion problem has to satisfy three properties:
  - exclusivity – only one process at a time is allowed to enter CS
  - progress – some process eventually enters CS
  - lack of starvation – each process requesting CS is eventually allowed to enter

- which properties are liveness? safety? both? which are redundant?

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
- 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

Causality relationship

- An event is usually influenced by part of the state.
- two consecutive events influencing disjoint parts of the state are independent and can occur in reverse order

- this intuition is captured in the notion of causality relation <
- for message-passing systems:
  - if two events e and f are different events of the same process and e occurs before f then e < f
  - if s is a send event and r is a receive event then s < r
- 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 < b or b < a then a and b are concurrent: a||b

Ring algorithm

For the initiator:

```
begin send (tok) to Nextp; receive (tok); decide end
```

For non-initiators:

```
begin receive (tok); send (tok) to Nextp 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) - no root, edges are undirected (bi-directional)
- 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?
What are these processes?

How many messages are sent in the algorithm?
Is this a wave algorithm?

Polling algorithm

```
var rexx, for each q in Neigp : boolean init false; (* rexx[q] is true if p has received a message from q *)
begin while (q: rexx[q] is false) > 1 do
  (* Now there is one q with rexx[q] is false *)
  send (tok) from q; rexx[q] := true end;
2: receive (tok) from q; rexx[q] := true;
end;

(* Excludes other processes of decsion: forall q in Neigp, q != p do send (tok) to q *)
```

- 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?
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, when it gets tokens from all other neighbors it replies back
- How many processes decide?
- How many messages are exchanged in the algorithm
- Is this a wave algorithm?

```
var recv : integer ("Counts number of received messages")
father : P ("")

For the initiator:
begin forall q in S
begin send (talk) to q;
while recv = d [q] ;
decide
end

For non-initiators:
begin receive (talk) from neighbor q ;
recv[q] = d [q] ;
while recv[q] = d [q] ;
send (talk) to father_q
end
```

14