Distributed Algorithms **M.Tech., CSE, 2016**

Lecture 1: Introduction to distributed Algorithms

Faculty: K.R. Chowdhary *: Professor of CS*

1.1 Syllabus

Models of synchronous and asynchronous distributed computing: Synchronous networks, asynchronous shared memory, asynchronous network.

Basic Algorithms for synchronous and asynchronous networks: leader selection, breadth-first search, shortest path, minimum spanning-tree.

Advanced asynchronous algorithms: distributed consensus with failures, commit protocols.

Asynchronous shared memory algorithms: mutual exclusion and consensus.

Relationship between shared memory and network models.

Asynchronous network with failures.

1.2 Introduction

Distributed algorithm is an interconnected collection of autonomous processes. These algorithms run on the hardware consisting of many interconnected processors, run concurrently and independently, and supposed to function properly even if processors or channels operate at different speeds, or even if some of them fails. These algorithms are useful for information exchange, resource sharing, and parallelization, to increase performance, replication to increase reliability, s well they are used in multicore programming.

Distributed v/s uni-process

- *Lack of knowledge of global state:* A process has no up-to-date knowledge of the local state of other processes. Hence, termination and deadlock becomes the major issues.
- *Lack of global time frame:* There is no total order on events by their temporal occurrence. In such situations the mutual exclusion becomes an issue.
- *Non-determinism:* Execution of processes is non-deterministic. Hence, running a system twice can give different results. The example is *race condition.*

Following are important applications of distributed algorithms:

- Telecommunications

- Distributed information processing
- Scientific computing
- Real-time process control
- Telephone systems
- Airline reservations
- Banking
- Weather prediction

The fundamental nature of concurrent algorithms is that there are many processors in distributed way over a large geographical area, against the conventional algorithms which run on a LAN, and in shared memory multi-processors.

Two important paradigm to capture communication in a distributed system are message passing and shared memory. However, the challenge with the distributed algorithms is that their design is extremely difficult.

Fundamentally, a communication can be *synchronous* or *asynchronous*, as defined below.

Definition 1.1 *Synchronous: Sending and receiving of a message are coordinated to form a single event, and a message is only allowed to be sent if its destination is ready to receive it. For example* handshake communication.

Definition 1.2 *Asynchronous: Communication means sending and receiving of a message are independent events. For example, sending postal letters, sending emails, etc.*

In a computer network, messages are transported through a medium, which may lose, duplicate or garbage these messages. A communication protocol detects such flaws during message passing. Example for this is sliding window protocol.

We make following assumptions in a communication network:

- A strongly connected network,
- Message passing asynchronous communication,
- Delay of messages in the channel are arbitrary, but finite,
- Channels are non-FIFO, and
- Processes do not crash.

Different types of distributed algorithms are:

- *IPC or Interprocess communication method:* Accessing shared memory, sending point-to-point or broadcast messages, and remote procedure calls.
- *The timing model:* Timing of events in systems, lockup in synchronously, or can be completely asynchronous, i.e., arbitrary speeds and arbitrary order. Others can be partially synchronous - processes can have access to approximately synchronized clocks.

- *Failure models:* Hardware is allowed to failure, or completely reliable. Failures might be processor failures: process stop with or without failures, might fail transiently, communication may fail with message loss or duplication.
- *Problem addressed:* Problems of resource allocation, communication, consensus among distributed processes, database concurrency control, deadlock detection, global snapshots, synchronization.

Following are some important features of distributed networks:

- Unknown number of processors
- Unknown hardware topology
- Independent inputs at different locations
- Several programs executing, starting at different times, and operating at different speeds,
- Uncertain message delivery times
- Unknown message ordering
- processor and communication failures

1.3 Synchronous network systems

The system of synchronous network is treated as a graph $G = (V, E)$, $|V| = n$, where nodes are logical software processors, and edges are communication paths. We use following terminology:

$out\text{-}nbrs_i$ is set of outgoing neighbors of node i

$in\text{-}nbrs_i$ is set of all incoming neighbors of of node i

$distance(i, j)$ is shortest path from node i to node j. If there is no path existing, then $distance(i, j) = \infty$.

M is set of message alphabet

null is absence of message

Associated with each $i \in V$, there is a process having following components:

- $states_i$: set of sates of process i
- $start_i$: start state or initial state of process i
- $msgs_i$: Message generation function for process i

$$msgs_i : start_i \times out - nbrs_i \rightarrow M \cup \{null\} \tag{1.1}$$

which is a mapping from $staes_i \times out - nbrs_i$ to elements of $M \cup \}$. Consider that $trans_i$ is a state transition mapping of $states_i$ and vectors (indexed by $in - nbrs_i$) of$M \cup \{null\}$ to $states_i$, i.e.,

$$trans_i : states_i \times M \cup \{null\} \rightarrow states_i \tag{1.2}$$

We define halting states as those not accepting, like FA. There may be variable start times of processes, and failures may be process failure, link failure, of stopping failure. In addition, there is a Byzantine failure - where next state and message generated are in arbitrary. The I/Os are assumed to be placed in designated variables.

Definition 1.3 *Rounds: This is sequence of message generation, transition, message generation, transition, and so on. This happens with set of states in arbitrary start state.*

Definition 1.4 *Execution of a system: To reason about a synchronous network system, we need formal notion of execution as:*

$$C_0, M_1, N_1, C_1, M_2, N_2, \ldots, C_r, M_{r+1}N_{r+1}$$

where C_0 is start state, M_1 is sent message, N_1 is received message, C_r is state after r rounds. The M_r and N_r may be different due to lossy network.

Let α and α' be two executions of a system. Then $\alpha \sim^i \alpha'$, if α is indistinguishable with respect to process i, if i has same states (c_i), M_i, N_i. This may be for r rounds.

Definition 1.5 *Randomization: We add component $rand_i$ to each process state S, so that $rand_i(S)$ is probability distribution on some subsets of $states_i$. So, $rand_i$ picks up new state, then $msgs_i$, $trans_i$ functions are applied. Execution is:*

$$C_0, D_1, M_1, N_1, C_2, \ldots,$$

where C_r, D_r state names, M_r, N_r are message names, and D_r is probability of process after r rounds of random counts.

1.4 Proof Methods

There are two approaches:

1. In variant assertions: A state which is true in every execution, after every round. The invariance assertions are proved by induction method for r rounds, starting with $r = 0$.
2. Simulation: A synchronous algorithm A "implements" another syn. algorithm B in the sense of providing same i/o behavior. It is simulation relation.

1.5 Complexity measures

Definition 1.6 *Time complexity: It is number of rounds until required output is produced or until the process halts.*

Definition 1.7 *Communication complexity: It is total number of non-null messages sent or (no. of bits sent).*

The communication complexity, causes congestion & effects other processes. We try to minimize the number of messages generated by induction algorithm. Following are the Complexity measures:

- *message complexity:* Total number of messages exchanged
- *Bit complexity:* total number of bits exchanged. This complexity measure is used only when messages are too long.
- *Time complexity:* It is assumed that event processing takes no time. Message is received at most one unit after it is sent.
- *Space complexity:* Amount of space used for processes.

Exercises

1. What are the basic communication approaches? Suggest suitable domains of their application (3 in each case).
2. What type of communication is suitable in distributed algorithms - synchronous or asynchronous? Justify.
3. What type of communication is used in the following cases:
   (a) Hard-disk to RAM data transfer
   (b) Computer to printer
   (c) CPU to monitor
4. Give a brief comparison of distributed vs. uni-process.
5. What are the criteria for determining the complexities of distributed systems?

References

[1] Nancy A. Lynch, and Boaz Patt-Shamir, "Distributed Algorithms," *Pearson Education, India*, 2002.

[2] Allen B. Tucker, Jr., "The computer Science and Engineering Handbook," *CRC Press*, 1997.