

1 Characteristics of Distributed Algorithms

Distributed algorithms are algorithms that execute on multiple processes. Distributed algorithms include a wide range of parallel algorithms including

- (1) shared memory algorithms
- (2) message passing or network algorithms
- (3) synchronous algorithms
- (4) asynchronous and partially synchronous algorithms
- (5) dataflow and database algorithms.

Unlike most parallel algorithms, which are characterized by *well coordinated computation* and *tightly coupled* processors, distributed algorithms have

- (1) *independence* of activities and *loosely coupled* processors
- (2) independent inputs with outputs at multiple locations
- (3) several programs executing simultaneously with separate control
- (4) processes starting at different times
- (5) failure of processors
- (6) processes running at different speeds
- (7) uncertain and asynchronous communication
- (8) uncertain *knowledge* of the system – each process is only aware of a part of the system
- (9) unknown number of processors etc.

It is hard to reason about distributed algorithms because of the uncertainty due to *asynchrony* and *failures*. We will study the distributed system by using various models of the system.

1. Synchronous and asynchronous processors

Synchronous processors take steps simultaneously. The times at which all these processors take steps are called *rounds*. Asynchronous processors take arbitrary time between steps.

2. Synchronous and asynchronous communication

This characterization is based on the amount of time taken for the message to be delivered to a processor. In synchronous communication system, the message sent will take a fixed time. In asynchronous communication system, messages can take arbitrary time and the time taken for a message to be delivered is not bounded.

3. Communication system

We model the communication between different processors using the communication system. The different processors communicate through a message passing system or shared memory. In message passing system, processors send messages to other processors through a communication

channel or a network, while in shared memory system the communication is achieved by writing and reading to a shared memory.

4. Partially Synchronous System

These are systems that lie between synchronous and asynchronous systems, where partial information about the timing of processor taking steps or for a message to be delivered.

5. Failure of processors

Failure in processors could be as *fail-stop faults* where processors stop participating according to the protocol, and ceases to take steps after a point, or as *omission faults* where processors intermittently fail to take some of its steps, but will take steps later, or *byzantine* where processors take steps which are not in accordance to their protocol. These are hardest to detect.

6. Failure of message system

Ideally, a message sent by a processor should be received by the receiving processor without any faults. However the message system can *drop, duplicate, or corrupt* some messages.

We will identify problems of major significance in distributed computing and define abstract versions of the problem for mathematical study. We will study various algorithms, prove correctness and analyze their complexity. We will prove lower bound and impossibility results. Various problems we will look at are

1. computing in a ring
2. mutual exclusion
3. consensus and renaming
4. set consensus

2 The Distributed Message Passing Model

We will start by talking about *message passing algorithms* for distributed systems. The communication system is described by a communication graph where

- *nodes* of the graph represent processors
- *edges* of graph represent communication channels between processors

Each processor has its own local memory and is running a local program. The local program consists of

- receiving messages on some edges
- some local computation and state change

- sending messages on some edges

The *communication graph* can be some structured graph – a ring, a tree, a complete graph (clique) or some arbitrary graph.

The degree of *synchrony* could vary in different systems. In a *synchronous system*, the computation proceeds in rounds. Each round consists of

- processors send messages
- processors wait to receive messages sent in that round
- processors do local computation based on messages received and changes state, and determine what messages to be sent in the next round

In an *asynchronous* system processors take a individual steps, send messages wait for messages to be received and then do a local computation steps. However,

- time between these individual steps may vary
- time for a message to be received by a processor may vary. Some messages take longer while others take shorter time to be delivered.

The degree of *symmetry* varies in systems. In an *anonymous* system, all processors are identical, without individual IDs. Therefore, all processors need to have identical local programs. On the other hand, in a system with *distinct* IDs, each processor has a unique name (usually an integer).