Achieving Agreement In Three Rounds With Bounded-Byzantine Faults

Mahyar Malekpour
NASA Langley Research Center
AIAA SciTech 2017, 7-14 January 2017
Grapevine, Texas

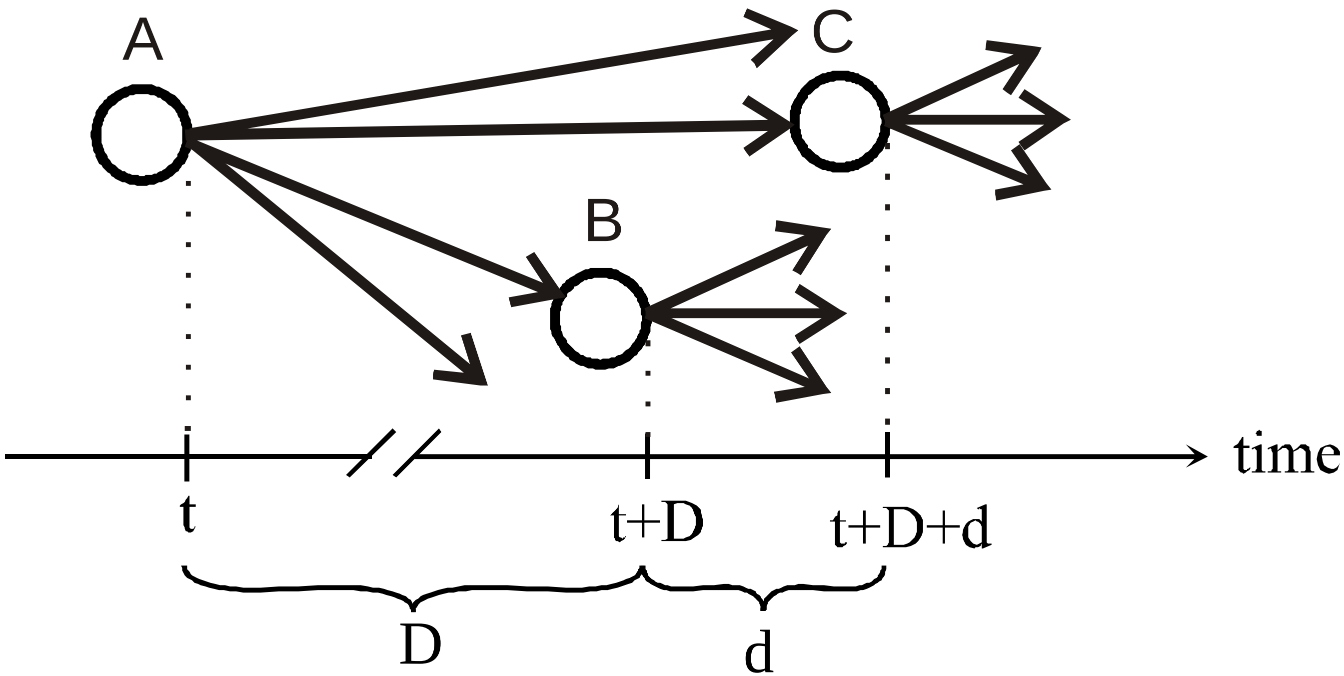
Communication And Synchronization

- Distributed systems are integral part of safety-critical computing applications, necessitating system designs that incorporate complex fault-tolerant resource management functions to provide globally coordinated operations with ultra-reliability.
- Distributed systems are modeled as graphs, nodes and edges, with wire/wireless communication links
- Robust clock synchronization is a required fundamental service
- Faults add complexity, various types from benign to arbitrary (Byzantine)

What Is Synchronization?

- Local oscillators/hardware clocks operate at slightly different rates, thus, they drift apart over time
- Local logical clocks, i.e., timers/counters, may start at different initial values
- The synchronization problem is to adjust the values of the local logical clocks so that nodes achieve synchrony and remain synchronized despite the drift of their local oscillators
- Application – Wherever there is a distributed system

Communication Parameters: *D, d*



Assumptions:
- Wired/wireless communication links
- $D \geq 1$ clock tick
- $d \geq 0$ clock tick
- D and d are bounded

What Is A Fault

- A defect/flaw in a system component resulting in an incorrect state
- Manifestation of an unexpected behavior

Fault Models

Node-Fault Model – traditional, Lamport 1982

- Faults are associated with the source node
- All count as a single fault, ex. Byzantine faulty node

Link-Fault Model – perception based, Schmid 1990

- Fault is associated with communication means connecting source to destination node
- All nodes are assumed to be good
- Invalid message at receiving node is counted as a single fault for the input link

Solving Clock Synchronization Problem

- *Direct* approach relies solely on local (node level) detection and filtering of faults
  - Limited to detecting timing and/or value faults of a node's incoming messages

- *Indirect* approach relies on the network level detection and filtering of faults independent of, and in addition to, local detection and filtering of faults
  - Requires coordination at the network level
    ➔ assumption of initial synchrony

Fault Management

- Authentication does not work, e.g., using CRC
- Driscoll: "It is not possible to prove such assumptions analytically for systems with failure probability requirements near 10^{-9}/hr."
- Other methods may not be verifiable, e.g., using
  - Self-checking pair at the node level
  - Central guardians at the system level

We believe, to be generally useful, algorithms that guarantee agreement must be able to handle non-authenticated messages.

System Overview

- Synchronous message passing
- Fully connected graph with $m < n/3$ nodes
- m = max number of simultaneous faults in the network
  - Note: *OM()* uses n and m, *3ROM()* uses K and F

Communication

- *Sync* message, i.e., {1, 0}
- Messages arrive within time interval $[t+D, t+D+d]$.

Oral Message (*OM*) Algorithm, Lamport *et al.* 1982

Let X = some arbitrary, but fixed, value
m = max number of faults

OM(0)

1. The transmitter sends its value to every receiver.
2. Each receiver uses value obtained from transmitter, otherwise X

OM(m), $m > 0$

1. The transmitter sends its value to every receiver.
2. For each p, let v_p be the value receiver p obtains from the transmitter, otherwise X. Each receiver p acts as the transmitter in OM(m - 1) to communicate its value v_p to n - 2 other receivers.
3. For each p, and each $q \neq p$, let v_q be the value receiver p obtained from receiver q in step (2) (using OM(m - 1)), otherwise X. Each receiver p calculates the majority value among all values v_q it receives, and uses that as the transmitter's value (otherwise X).

OM Algorithm

- Recursive $m + 1$ rounds of exchanges
- Reaches agreement
- Does not require initial synchrony
- Message complexity = $O(n^m)$ for wired network
- Number of exchanged messages grows exponentially as m grows linearly
- Impractical for $m > 2$
- A number of *shortcuts*, ex. early-stopping algorithm, overcome excessive rounds and growing message size and complexity

3-Round *OM* (*3ROM*) Algorithm

Assumptions:

- A good node experiences no more than *F* faults
  - Given - there are max *F* faulty nodes
- A faulty node induces no more than *F* faults
  - We assumed max *F* faults

Round 1 – The source node broadcasts *Sync* message

Round 2 – Each node receiving *Sync* broadcasts *Relay* message

Round 3 – Each node broadcasts its vector of received messages

Process & Vote –
Each node processes received messages and then votes

3ROM Algorithm

- Not recursive, only 3 rounds of exchanges
- Reaches agreement
- Does not require initial synchrony
- Message Complexity = $O(K^3)$ for wired network
- Message Complexity = $O(K^2)$ for wireless network
- Number of exchanged messages grows linearly with F
- Unlike *OM* alg. if a node does not receive a message, it does not broadcast a message

Model Checking

- Symbolic Model Verifier (SMV)
- SMV's language description and modeling capability provide relatively easy translation from the pseudo-code
- SMV semantics are synchronous composition, where all assignments are executed in parallel and synchronously

- Verified correctness of our formal proof of the algorithm
- Results confirmed claims of determinism and independence of the *3ROM* algorithm from *F*
- A number of cases for each fault model were model checked
- Node-Fault model, with $F = 0..3$ and $K = 4..10$, weaker assumptions: $\sum c_j \geq F+1$ and $\sum X_i \geq F+2$
- Link-Fault model, $F = 2$, $K = 7$, and $F = 3$, $K = 10$
- http://shemesh.larc.nasa.gov/people/mrm/publications.htm

Questions?