AGREEMENT IN A FAILURE-FREE SYSTEM

- In a failure-free system, consensus can be reached by collecting information from the different processes, arriving at a decision, and distributing this decision in the system.
- A distributed mechanism would have each process broadcast its values to others, andeach process computes the same function on the values received.
- The decision can be reached by using an application specific function.
- Algorithms to collect the initial values and then distribute the decision may be based on the token circulation on a logical ring, or the three-phase tree-based broadcast converge cast: broadcast, or direct communication with all nodes.
- In a synchronous system, this can be done simply in a constant number of rounds.
- Further, common knowledge of the decision value can be obtained using an additionalround.
- In an asynchronous system, consensus can similarly be reached in a constant numberof message hops.
- Further, concurrent common knowledge of the consensus value can also be attained.

AGREEMENT IN (MESSAGE-PASSING) SYNCHRONOUS SYSTEMS WITH FAILURES

Consensus algorithm for crash failures (synchronous system)

- Consensus algorithm for crash failures message passing synchronous system.
- The consensus algorithm for n processes where up to f processes where f < n may fail in a fail stop failure model.
- Here the consensus variable x is integer value; each process has initial value xi. If

up to f failures are to be tolerated than algorithm has f+1 rounds, in each round a process i sense the value of its variable xi to all other processes if that value has not been sent before.

- So, of all the values received within that round and its own value xi at that start of the round the process takes minimum and updates xi occur f + 1 rounds the local value xi guaranteed to be the consensus value.
- In one process is faulty, among three processes then f = 1. So the agreement requires f + 1 that is equal to two rounds.
- If it is faulty let us say it will send 0 to 1 process and 1 to another process i, j and k. Now, on receiving one on receiving 0 it will broadcast 0 over here and this particular process on receiving 1 it will broadcast 1 over here.
- So, this will complete one round in this one round and this particular process on receiving 1 it will send 1 over here and this on the receiving 0 it will send 0 over here.

(global constants)

integer: f; // maximum number of crash failures tolerated

(local variables)

Integer: x ← local value;

(1) Process P_i ($1 \leq i \leq n$) execute the consensus algorithm for up to f crash failures:

(1a) **for** round from 1 to f + 1 **do**

(1b) **if** the current value of x has not been broadcast **then**

(1c) **broadcast(x);**

(1d) $y_i \leftarrow$ value (if any) received from process j in this round;

(1e) $x \leftarrow \min_{\forall j}(x, y_j)$;

(1f) **output** x as the consensus value.

Fig : Consensus with up to f fail-stop processes in a system of n processes, n > f

- The agreement condition is satisfied because in the f+ 1 rounds, there must be at least one round in which no process failed.
- In this round, say round r, all the processes that have not failed so far succeed in broadcasting their values, and all these processes take the minimum of the values broadcast and received in that round.
- Thus, the local values at the end of the round are the same, say x^i_r for all non-failed processes.
- In further rounds, only this value may be sent by each process at most once, and no process i will update its value x^i_r.
- The validity condition is satisfied because processes do not send fictitious values inthis failure model.
- For all i, if the initial value is identical, then the only value sent by any process isthe value that has been agreed upon as per the agreement condition.
- The termination condition is seen to be satisfied.

Complexity

- The complexity of this particular algorithm is it requires f + 1 rounds where f < n and the number of messages is $O(n^2)$ in each round and each message has one integers hence the total number of messages is $O((f+1)\cdot n^2)$ is the total number ofrounds and in each round n^2 messages are required.

Lower bound on the number of rounds

- At least f + 1 rounds are required, where f < n.
- In the worst-case scenario, one process may fail in each round; with f + 1 rounds, there is at least one round in which no process fails. In that guaranteed failure-free round, all messages broadcast can be delivered reliably, and all processes that have not failed can compute the common function of the received values to reach an

agreement value.

Consensus algorithms for Byzantine failures (synchronous system)

Upper bound on Byzantine processes

- In a system of n processes, the Byzantine agreement problem can be solved in a synchronous system only if the number of Byzantine processes f is such that

$$f \leq \left\lceil \frac{n-1}{3} \right\rceil$$

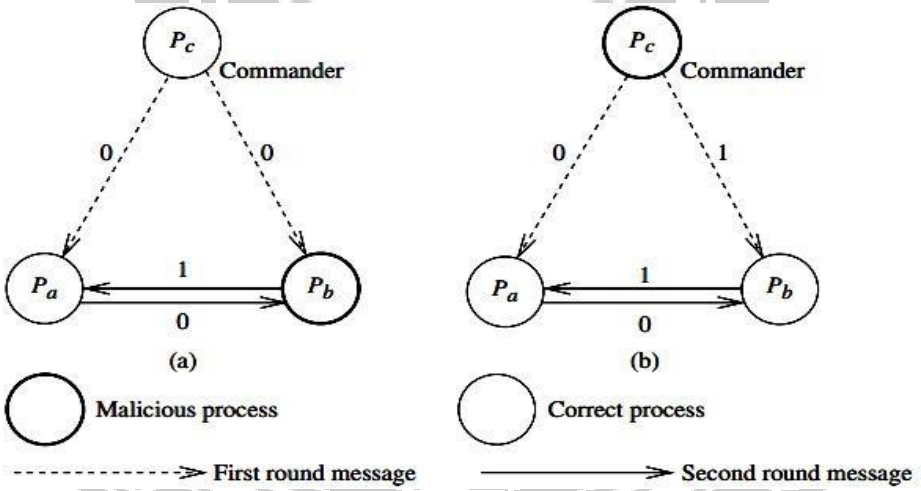

Fig: Impossibility of achieving Byzantine agreement with n = 3 processes and f = 1 malicious process

- The condition where f < (n – 1) / 2 is violated over here; that means, if f = 1 andn = 2 this particular assumption is violated
- (n – 1) / 2 is not 1 in that case, but we are assuming 1 so obviously, as per the previous condition agreement byzantine agreement is not possible.
- Here P 0 is faulty is non faulty and here P 0 is faulty so that means P 0 is the source, the source is faulty here in this case and source is non faulty in the other case.
- So, source is non faulty, but some other process is faulty let us say that P 2 is faulty. P 1 will send because it is non faulty same values to P 1 and P 2 and as far as the P 2s concerned it will send a different value because it is a faulty.

- Agreement is possible when f = 1 and the total number of processor is 4. So, agreement we can see how it is possible we can see about the commander P c.
- So, this is the source it will send the message 0 since it is faulty. It will send 0 to Pd 0 to P b, but 1 to pa in the first column. So, P a after receiving this one it will send one to both the neighbors, similarly P b after receiving 0 it will send 0 since itis not faulty.
- Similarity P d will send after receiving 0 at both the ends.
- If we take these values which will be received here it is 1 and basically it is 0 and this is also 0.
- So, the majority is basically 0 here in this case here also if you see the values 10 and 0. The majority is 0 and here also majority is 0.
- In this particular case even if the source is faulty, it will reach to an agreement, reach an agreement and that value will be agreed upon value or agreement variable will be equal to 0.

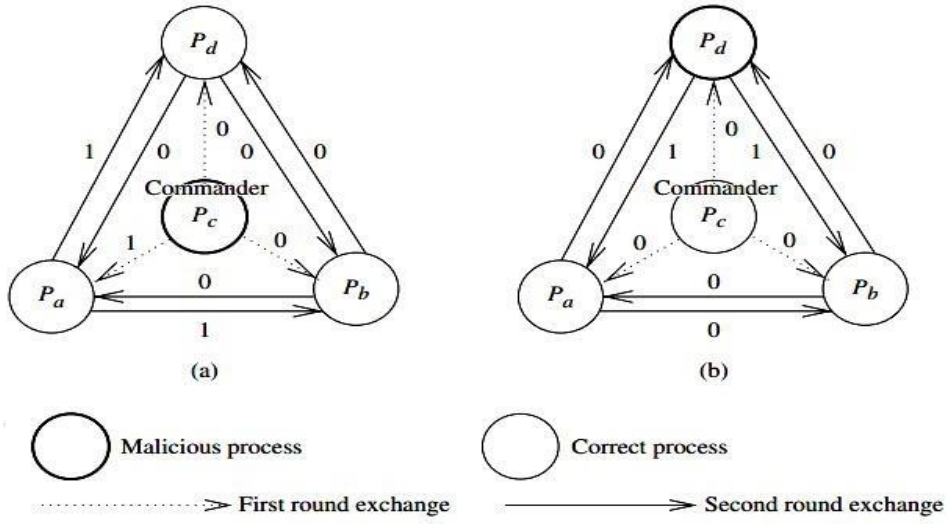

Fig : Achieving Byzantine agreement when n = 4 processes and f = 1

ROHINI COLLEGE OF ENGINEERING AND TECHNOLOGY
PALKULAM, KANYAKUMARI
OBSERVE OPTIMIZE OUTSPREAD