Agreement in (message-passing) synchronous systems with failures : –

Consensus algorithm for crash failures

Vidya Academy of Science & Technology, Thrissur
Sivadasan E T
Associate Professor
Department of CSE.

Consensus algorithm for crash failures

- A set of processes need to agree on a value (decision), after one or more processes have proposed what that value (decision) should be

- Examples:

  mutual exclusion, election, transactions

Consensus algorithm for crash failures

System model

- N processes $\{p_1, p_2, ..., p_N\}$
- Communication is reliable but processes may fail.
- At most f processes out of N may be *faulty.*
  - Crash failure.
  - Byzantine failure (arbitrary).

- The system is logically fully connected.
- A receiver process knows the identity of the sender process.

Authenticated & Non-authenticated messages

- To reach an agreement, processes have to exchange their values and relay the received values to other processes.

Authenticated & Non-authenticated messages

Authenticated or *signed* message system – A (faulty) process cannot forge a message or change the contents of a received message (before it relays the message to other).

Because a process can verify the authenticity of a received message.

Authenticated & Non-authenticated messages

Non-authenticated or *unsigned* or *oral* message

– A (faulty) process can forge a message and claimed to have received it from another process or change the contents of a received message before it relays the message to other.

A process has no way of verifying the authenticity of a received message.

Consensus problem:

- ‘N’ processes agree on a value (e.g. synchronized action – go / abort)

- Consensus may have to be reached in the presence of failure

  - Process failure – crash/fail-stop, arbitrary failure
  - Communication failure

Consensus algorithm for crash failures

(global constants)
integer: f; // maximum number of crash failures tolerated
(local variables)
integer: $x \longleftarrow$ local value;

(1) Process P_i ($1 \leq i \leq n$) executes 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) $y_j \longleftarrow$ value (if any) received from process j in this round;
(1e) $x \longleftarrow min_{\forall j}(x, y_j)$;
(1f) **output** x as the consensus value.

Algorithm 14.1 Consensus with up to f fail-stop processes in a system of n processes, $n > f$ [8]. Code shown is for process P_i, $1 \leq i \leq n$.

Consensus algorithm for crash failures

- The above algorithm is a consensus algorithm where *“Consensus with up to **f** fail-stop processes in a system of **n** processes, **n>f**”.*

Consensus algorithm for crash failures

- Here, the consensus variable ***x*** is integer-valued.
- Each process has an initial value x_i. If up to ***f*** failures are to be tolerated, then the algorithm has ***f +1*** rounds.

Consensus algorithm for crash failures

- In each round, a process ***i*** sends the value of its variable $\boldsymbol{x_i}$ to all other processes if that value has not been sent before.
- Of all the values received within the round and its own value $\boldsymbol{x_i}$ at the start of the round, the process takes the minimum, and updates $\boldsymbol{x_i}$.

Consensus algorithm for crash failures

- After ***f* *+1*** rounds, the local value $\boldsymbol{x_i}$ is guaranteed to be the consensus value.

Consensus problem:

- ***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.

Consensus problem:

- Thus, the local values at the end of the round are the same, say 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 $\boldsymbol{x_i^r}$.

Consensus problem:

- The validity condition is satisfied because processes do not send fictitious values in this failure model. (Thus, a process that crashes has sent only correct values until the crash.)
- For all ***i***, if the initial value is identical, then the only value sent by any process is the value that has been agreed upon as per the agreement condition.

THANK YOU !