3.10 CHANDY–MISRA–HAAS ALGORITHM FOR THE AND MODEL

- This is considered an edge-chasing, probe-based algorithm.
- It is also considered one of the best deadlock detection algorithms for distributed systems.
- If a process makes a request for a resource which fails or times out, the process generates a probe message and sends it to each of the processes holding one or more of its requested resources.
- This algorithm uses a special message called probe, which is a triplet (i, j,k), denoting that it belongs to a deadlock detection initiated for process Pi andit is being sent by the home site of process Pj to the home site of process Pk.
- Each probe message contains the following information:
    - the id of the process that is blocked (the one that initiates the probe message);
    - the id of the process is sending this particular version of the probe message;
    - the id of the process that should receive this probe message.
- A probe message travels along the edges of the global WFG graph, and a deadlock is detected when a probe message returns to the process that initiated it.
- A process P_j is said to be dependent on another process P_k if there exists a sequence of processes P_j, P_{i1} , P_{i2} , . . . , P_{im}, $P_k$such that each process except $P_k$in the sequence is blocked and each process, except the P_j, holds a resource for which the previous process in the sequence is waiting.
- Process P_j is said to be locally dependent upon process P_k if P_j is dependent upon $P_k$and both the processes are on the same site.
- When a process receives a probe message,it checks to see if it is also waiting for resources
- If not, it is currently using the needed resource and will eventually finish and release the resource.
- If it is waiting for resources, it passes on the probe message to all processes it knows to be holding resources it has itself requested.
- The process first modifies the probe message, changing the sender and receiver ids.
- If a process receives a probe message that it recognizes as having initiated,it knows there is a cycle in the system and thus, deadlock.

Data structures

Each process Pi maintains a boolean array, dependenti, where dependent(j) is true only if Pi knows that Pj is dependent on it. Initially, dependenti (j) is false for all i and j.

if P_i is locally dependent on itself
  then declare a deadlock
  else for all P_j and P_k such that
    (a) P_i is locally dependent upon P_j, and
    (b) P_j is waiting on P_k, and
    (c) P_j and P_k are on different sites,
  send a probe (i, j, k) to the home site of P_k

On the receipt of a probe (i, j, k), the site takes
  the following actions:

if
    (d) P_k is blocked, and
    (e) $dependent_k(i)$ is false, and
    (f) P_k has not replied to all requests P_j,
    then
      begin
        $dependent_k(i) = true$;
        if $k = i$
          then declare that P_i is deadlocked
        else for all P_m and P_n such that
          (a′) P_k is locally dependent upon P_m, and
          (b′) P_m is waiting on P_n, and
          (c′) P_m and P_n are on different sites,
        send a probe (i, m, n) to the home site of P_n
      end.

Fig 3.11: Chandy–Misra–Haas algorithm for the AND model

Performance analysis

- In the algorithm, one probe message is sent on every edge of the WFG which connects processes on two sites.
- The algorithm exchanges at most m(n − 1)/2 messages to detect a deadlock that involves m processes and spans over n sites.
- The size of messages is fixed and is very small (only three integer words).
- The delay in detecting a deadlock is O(n).

Advantages:

- It is easy to implement.
- Each probe message is of fixed length.
- There is very little computation.
- There is very little overhead.
- There is no need to construct a graph, nor to pass graph information to other sites.
- This algorithm does not find false (phantom) deadlock.
- There is no need for special data structures.

3.11 CHANDY–MISRA–HAAS ALGORITHM FOR THE OR MODEL

- A blocked process determines if it is deadlocked by initiating a diffusion computation.
- Two types of messages are used in a diffusion computation:
  - query(i, j, k)
  - reply(i, j, k)

denoting that they belong to a diffusion computation initiated by a process p_i and are being sent from process p_j to process p_k.

- A blocked process initiates deadlock detection by sending query messages to all processes in its dependent set.
- If an active process receives a query or reply message, it discards it.
- When a blocked process Pk receives a query(i, j, k) message, it takes the following actions:
  1. If this is the first query message received by Pk for the deadlock detection initiated by Pi, then it propagates the query to all the processes in its dependent set and sets a local variable num_k (i) to the number of query messages sent.
  2. If this is not the engaging query, then Pk returns a reply message to it immediately provided Pk has been continuously blocked since it received the corresponding engaging query. Otherwise, it discards the query.
  - Process Pk maintains a boolean variable $wait_k$(i) that denotes the fact that it has been continuously blocked since it received the last engaging query from process Pi.
  - When a blocked process Pk receives a reply(i, j, k) message, it decrements num_k(i) only if $wait_k$(i) holds.
  - A process sends a reply message in response to an engaging query only after it has received a reply to every query message it has sent out for this engaging query.
  - The initiator process detects a deadlock when it has received reply messages to all the query messages it has sent out.

Initiate a diffusion computation for a blocked process P_i:
send $query(i, i, j)$ to all processes P_j in the dependent set DS_i of P_i;
$num_i(i) := |DS_i|$; $wait_i(i) := true$;

When a blocked process P_k receives a *query*(i, j, k):
if this is the engaging *query* for process P_i then
send $query(i, k, m)$ to all P_m in its dependent set DS_k;
$num_k(i) := |DS_k|$; $wait_k(i) := true$
else if $wait_k(i)$ then send a $reply(i, k, j)$ to P_j.

When a process P_k receives a *reply*(i, j, k):
if $wait_k(i)$ then
$num_k(i) := num_k(i) - 1$;
if $num_k(i) = 0$ then
if $i = k$ then **declare a deadlock**
else send $reply(i, k, m)$ to the process P_m
which sent the engaging query.

Fig 3.12: Chandy–Misra–Haas algorithm for the OR model

Performance analysis

- For every deadlock detection, the algorithm exchanges e query messages ande reply messages, where e = n(n – 1) is the number of edges.