Chandy-Misra- Haas

- Edge chasing algorithm based on the AND model.
- A process Pj is dependent on Pk if there is a sequence Pj, Pi1….Pin,Pk such that all process but Pk are blocked, and each process except Pj has something that is needed by its predecessor.
  - Locally dependent
- If Pi is locally dependent on itself, then we have a deadlock. Otherwise
  - Forall Pj, Pk such that Pi locally depends on Pj and Pj is waiting(not locally) on Pk, send probe(i,j,k) to Pk.

- On receiving probe(i,j,k)
  - If (*Pk is deadlocked && ! dependent$_k$(i) && Pk has not replied to all requests of Pj*)
    - Dependent$_k$(i) = true.
    - If (k == i)
      - » **Then** Pi is deadlocked
      - » **Else** Forall Pm, Pn such that Pk locally depends on Pm and Pm is dependent (not locally) on Pn, send probe(i,m,n) to Pk.
- Sends 1 proble message on each edge of WFG, so m(n-1)/2 messages for a deadlock with m processes over n sites. Size is fixed, and detection time is linear in number of sites

Diffusion Based Algorithm

- Works for OR request model
- Initiation:
  - A blocked process i sends *query(i,i,j)* to all Pj in its dependent set; $num_i(i)$ = |DSi| , $wait_i(i)$ = true;
- When a blocked process Pk recvs query (i,j,k)
  - If this is engaging query, send query(i,k,m) to all processes in its dependent set, and set $num_k(i)$ and $wait_k(i)$
  - Else if $wait_k(i)$ then send reply(i,k,j)
- When Pk gets reply(i,j,k)
  - If $wait_k(i)$
    - Decrement $num_k(i)$, if it becomes 0 then
      - » If k == I then deadlock else reply(i,k,m) to the process which sent the engaging query.

Heirarchical Algorithms

- Menasce-Muntz
  - Resources are managed by nodes that form the “leaves” of a tree. They maintain TWF/WFGs corresponding to the resources they manage.
  - Several leaf controllers have a single parent, and so on in a tree fashion. Each non-leaf controller maintains WFG which is union of child WFGs. Changes are propagated upwards, and deadlocks detected on the way
- Hierarchical Ho-Ramamoorthy
  - Sites split into disjoint clusters.
  - Each cluster has its own control site. There is also a central control site.

Issues

- Formal methods to prove correctness
- Performance metrics
  - No of messages ? Message size? Time to detect ? Storage overhead ? Computation overhead ?
- Resolution – basically aborting a process
  - How does a process know which others are involved in a deadlock ?
  - Can two process detect the same deadlock simultaneously ?
  - Use Priorities!
  - Rollback – release resources, clean up graph
- Phantom Deadlocks.