Quick notes on deadlocks

Below is an illustration of deadlocking where each packet is dependent on the resource in front of it forming a cyclic dependency.

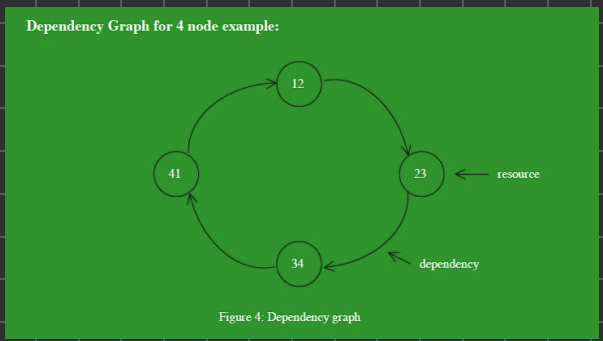


Figure 4: Dependency graph

Breaking a deadlock can occur with establishing routing relationship in which all cyclic dependencies are eliminated. This may be an extreme approach as it could conceivably eliminate minimum paths and cause channeling through key corridors.

We can still maintain a completely connected graph while eliminating the deadlock by adding another virtual channel on one leg shown below.

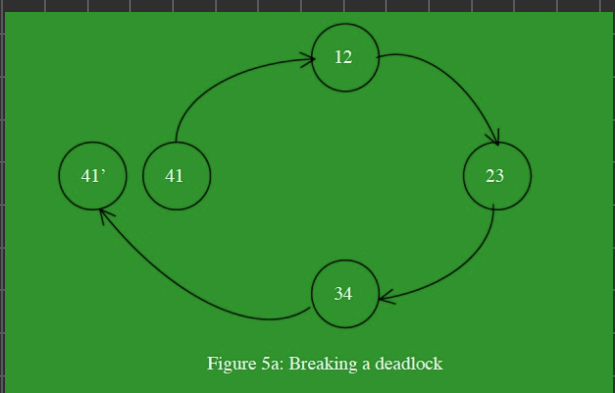


Figure 5a: Breaking a deadlock

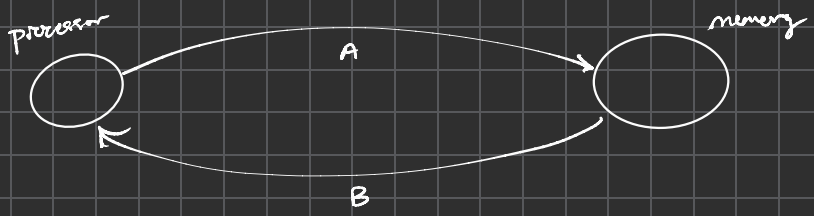
Generally speaking, deadlocks can be avoided or recovered from by doing the following:

To implement deadlock recovery we need to do two things:

1. Detect Deadlock: To detect real deadlock we need global knowledge, which is not available sitting at the router. It could possibly be done but would be very complex and expensive. An easier option is to approximate deadlock condition by using timeout. So if a particular channel has been unable to send a message for a large amount of time the router can assume that it is in a deadlock. The timeout can also be because of a bottleneck.

2. Recover from Deadlock:
• Drop the packet: Depends on the higher level protocol. In IP it may be acceptable to drop packets but it not feasible for processor-memory interconnect.
• Have an escape path: According to Prof. Dally, this is just like Duato's algorithm where we have a set of acyclic channels. The escape channels in this case can be looked upon as that set because they behave exactly like VCs (they have state, have buffering). Each node has n+1 channels, n channels are used for normal routing and 1 channel is restricted for escape. All the escape channels are linked in a hamiltonian cycle (a cycle that goes through all the nodes once and only once). You would still need to be careful as this is nothing but 'N' (where N is the number of nodes in the network) ring which itself is cyclic. Another problem with this approach is that with heavy traffic (when deadlocks are most likely to happen) many packets will timeout and all of them will get queued up on this escape path clogging it. It will then take a long time to get out of deadlock.
• Another method from last year's scribed lecture is to buffer the packet on the node itself and wait for the channels to become free. This is not possible because generally the memory bandwidth (and capacity) on the node is far less than the network bandwidth.

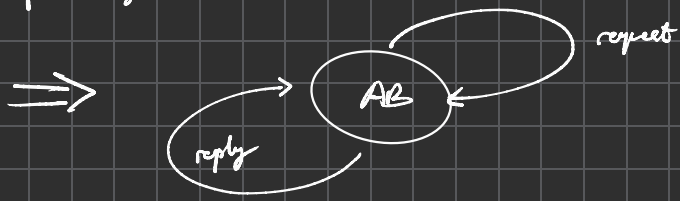
Finally consider an instance in which a higher level protocol may cause a deadlock in a network that was designed to be deadlock free
e.g. cache coherence protocol working over an interconnect network



- in this instance, the processor sends a request on channel A to which the memory is attempting to send a reply on channel B
- representing as a dependency graph:

↳ e.g. buffers that the memory is trying to use to send a reply with have already been allocated to the processor's request

- solution: use unidirectional VC's for request and reply respectively



or visualizing structurally