## **Vincent Fernnadez**

**Operating System**

## **Reactions Paper**

**Deadlock Detection And Recovery**

If resources have single instance:  
In this case for Deadlock detection we can run an algorithm to check for cycle in the Resource Allocation Graph. Presence of cycle in the graph is the sufficient condition for deadlock.

If there are multiple instances of resources:  
Detection of the cycle is necessary but not sufficient condition for deadlock detection, in this case, the system may or may not be in deadlock varies according to different situations.

**Deadlock Recovery**  
A traditional operating system such as Windows doesn’t deal with deadlock recovery as it is time and space consuming process. Real-time operating systems use Deadlock recovery.

**Recovery method**

1. **Killing the process:** killing all the process involved in the deadlock. Killing process one by one. After killing each process check for deadlock again keep repeating the process till system recover from deadlock.
2. **Resource Preemption:** Resources are preempted from the processes involved in the deadlock, preempted resources are allocated to other processes so that there is a possibility of recovering the system from deadlock. In this case, the system goes into starvation.

And base on the video they say that there are two methods.

First would be abort all the deadlocks process, Aborting all the process will certainly break the deadlock but with a great expenses,

And the second one would be abort one process at a time, until the deadlock is eliminated from the system.

How do they detected?

Distributed deadlocks can occur in distributed systems when distributed transactions or concurrency control is being used. Distributed deadlocks can be detected either by constructing a global wait-for graph from local wait-for graphs at a deadlock detector or by a distributed algorithm like edge chasing.