Communication and resource deadlock

A deadlock occurs if there is a cycle of processes waiting until:

- another process on the cycle sends some input (communication deadlock)
- or resources held by other processes on the cycle are released (resource deadlock)

Both types of deadlock are captured by the N-out-of-M model:

A process can wait for N grants out of M requests.

Examples:

- A process is waiting for one message from a group of processes: $N = 1$
- A database transaction first needs to lock several files: $N = M$.

Wait-for graph

A (non-blocked) process can issue a request to M other processes, and becomes blocked until N of these requests have been granted.

Then it informs the remaining $M - N$ processes that the request can be dismissed.

Only non-blocked processes can grant a request.

A (directed) wait-for graph captures dependencies between processes.

There is an edge from node p to node q if p sent a request to q that wasn't yet dismissed by p or granted by q.

Wait-for graph - Example 1

Suppose process p must wait for a message from process q.

In the wait-for graph, node p sends a request to node q.

Then edge pq is created in the wait-for graph, and p becomes blocked.

When q sends a message to p, the request of p is granted.

Then edge pq is removed from the wait-for graph, and p becomes unblocked.

Wait-for graph - Example 2

Suppose two processes p and q want to claim a resource.

In the wait-for graph, nodes u, v representing p, q send a request to node w representing the resource. Edges uw and vw are created.

Since the resource is free, the resource is given to say p.
So w sends a grant to u. Edge uw is removed.

The basic (mutual exclusion) algorithm requires that the resource must be released by p before q can claim it.
So w sends a request to u, creating edge wu in the wait-for graph.

After p releases the resource, u grants the request of w.
Edge wu is removed.

The resource is given to q. Hence w grants the request from v.
Edge vw is removed and edge wv is created.