

# Detecting Deadlock in Queueing Network Simulations

Geraint Palmer

## 1 Explanation of Deadlock

Any open queueing network with feedback loops, at least one service station that has limited queueing capacity, and where individuals can be blocked from joining the queue at the next destination can experience deadlock. Deadlock occurs when a customer finishes service at node  $i$  and is blocked from transitioning to node  $j$ ; however the individuals in node  $j$  are all blocked, directly or indirectly, by the blocked individual in node  $i$ . That is, deadlock occurs if every individual blocking individual  $X$ , directly or indirectly, are also blocked.

In figure 1 a simple two node queueing network is shown in a deadlocked state. Customer  $e$  has finished service at node 1, but remains there as there is not enough queueing space at node 2 to accept him. We say he is blocked by customer  $i$ , as he is waiting for customer  $i$  to be released. Similarly, customer  $i$  has finished service at node 2, but remains there as there is not enough queueing space at node 1, customer  $i$  is blocked by customer  $e$ .

When there are multiple servers, individuals become blocked by all customers in service or blocked at their destination. Figure 2 (top) shows two nodes in deadlock, customer  $i$  is blocked by both  $d$  and  $e$ , who are both blocked by customer  $i$ . However in figure 2 (bottom), customer  $i$  is blocked by both  $d$  and  $e$ , and customer  $d$  isn't blocked, and so there is no deadlock.

NEED A LOT MORE HERE. DIAGRAMS TO EXPLAIN TOO.

## 2 Literature Review

Most of the literature on blocking conveniently assumes the networks are deadlock-free. For closed networks of  $K$  customers with only one class of customer, [1] proves the following condition to ensures no deadlock:

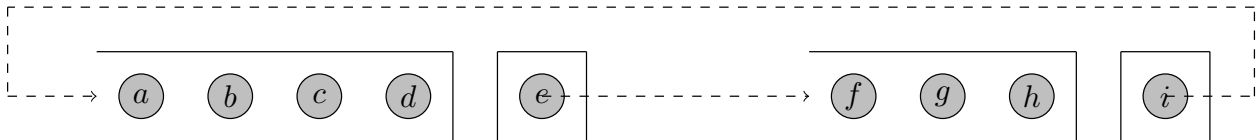


Figure 1: Two nodes in deadlock.

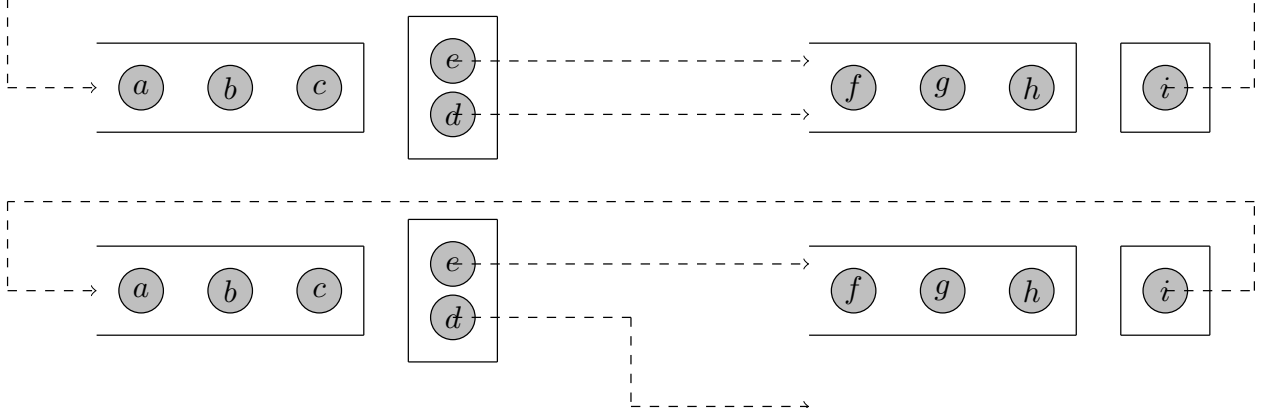


Figure 2: Two nodes in deadlock (top), and Two node not deadlocked (bottom).

for each minimum cycle  $C$ ,  $K < \sum_{j \in C} B_j$ , the total number of customers cannot exceed the total queueing capacity of each minimum subcycle of the network. The paper also presents algorithms for finding the minimum queueing space required to ensure deadlock never occurs, for closed cactus networks, where no two cycles have more than one node in common. This result is extended to multiple classes of customer in [2], with more restrictions such as single servers and each class having the same service time distribution. Here a integer linear program is formulated to find the minimum queueing space assignment that prevents deadlock. The literature does not discuss deadlock properties in open restricted queueing networks.

NEED A LOT MORE HERE

### 3 Definitions

$|V(D)|$

The order of the directed graph  $D$  is its number of vertices.

Weakly connected component

A weakly connected component of a digraph containing  $X$  is the set of all nodes that can be reached from  $X$  if we ignore the direction of the edges.

Direct successor

If a directed graph contains an edge from  $X_i$  to  $X_j$ , then we say that  $X_j$  is a direct successor of  $X_i$ .

Ancestors

If a directed graph contains a path from  $X_i$  to  $X_j$ , then we say that  $X_i$  is an ancestor of  $X_j$ .

Decendents

If a directed graph contains a path from  $X_i$  to  $X_j$ , then we say that  $X_j$  is a descendant of  $X_i$ .

$\deg^{\text{out}}(X)$

The out-degree of  $X$  is the number of outgoing edges emanating from that vertex.

NEED CONSISTANT NOTATION, REFERENCES FOR DEFINITIONS.

## 4 Explaining Digraph

Here is a method of detecting when deadlock occurs in an open queueing network  $Q$  with  $N$  nodes. Let the number of servers in node  $i$  be denoted by  $c_i$ .

Define  $D$  as a directed graph associated with  $Q$ , where  $|V(D)| \leq \sum_{i=1}^N c_i$ . Each vertex of  $D$  is an occupied server, occupied by an individual who is either in service or blocked.

When an individual finishes service at node  $i$ , and this individual's next destination is node  $j$ , but there is not enough queueing capacity for  $j$  to accept that individual, then that individual remains at node  $i$  and becomes blocked. At this point  $c_j$  directed edges between this individual's server and the servers of each individual in service or blocked at node  $j$  are created in  $D$ .

When an individual is released and another customer occupies their server, that server's out-edges are removed. When an individual is released and there isn't another customer to occupy that server, then all edges incident to that server are removed.

CLEARER: LOTS OF DIAGRAMS. SUPGRAPHS THING.

## 5 Theorem

### Theorem

A deadlock situation arises if and only if there exists a weakly connected component of  $D$  containing no vertices with  $\deg^{\text{out}} = 0$ .

### Observations

Consider one weakly connected component  $G$  of  $D$ . Consider the node  $X_a \in G$ . If  $X_a$  is unoccupied, then  $X_a$  has no incident edges. Consider the case when  $X_a$  is occupied by individual  $a$ , whose next destination is node  $j$ . Then  $X_a$ 's direct successors are the servers occupied by individuals who are blocked or in service at node  $j$ . We can interpret all  $X_a$ 's descendants as the servers whose occupants are directly or indirectly blocking  $a$ , and we can interpret all  $X_a$ 's ancestors as those servers whose individuals who are being blocked directly or indirectly by  $a$ .

Note that the only possibilities for  $\deg^{\text{out}}(X_a)$  are being 0 or  $c_j$ . If  $\deg^{\text{out}}(X_a) = c_j$  then  $a$  is blocked by all its direct successors. The only other situation is that  $a$  is not blocked, and  $X_a \in G$  because  $a$  is in service at  $X_a$  and blocking other individuals, in which case  $\deg^{\text{out}}(X_a) = 0$ .

It is clear that if any of  $X_a$ 's descendants are occupied by individuals who are not blocked, then we do not have deadlock; and if all of  $X_a$ 's descendants are occupied by blocked individuals, then we have deadlock.

We also know that by definition all of  $X_a$ 's ancestors are occupied by blocked individuals.

## Proof

Consider one weakly connected component  $G$  of  $D$ . All vertices of  $G$  are either decedents of another vertex and so are occupied by an individual who is blocking someone; or are ancestors of another vertex, and so are occupied by someone who is blocked.

Assume that  $G$  contains a vertex  $X$  such that  $\deg^{\text{out}}(X) = 0$ . This implies that  $X$ 's occupant is not blocked, so  $X$  must be a descendent of another vertex. Therefore  $Q$  is not deadlocked as there exists a vertex whose decedents are not all blocked.

Now assume that we have deadlock. For a given vertex  $X$ , all decedents of  $X$  are occupied by individuals who are blocked, and so must have out-degrees greater than 0. However, as our choice of  $X$  was arbitrary, this must be true for all members of  $G$ , and no members of  $G$  have out-degree of 0.

## References

- [1] S. Kundu and I. Akyildiz. Deadlock buffer allocation in closed queueing networks. *Queueing systems*, 4(1):47–56, 1989.
- [2] J. Liebeherr and I. Akyildiz. Deadlock properties of queueing networks with finite capacities and multiple routing chains. *Queueing systems*, 20(3-4):409–431, 1995.