On Deadlocking in Queueing Networks

Geraint Ian Palmer, Paul Harper, Vincent Knight

September 29, 2015

Abstract

In this paper a deadlock in open restricted queueing networks is investigated. A method to detect when deadlock occurs in simulations of these networks in presented, using a state digraph. Markov models of five deadlocking queueing networks are given, and their properties on expected time to deadlock explored. These networks are a one node single server, one node multi server, two node single server without loops, two node multi server without loops, and two node single server with loops. These properties are compared to results obtained using the simulation model. Finally a bound on the time to deadlock of the two node single server with loops network is derived.

1 Introduction

Restricted open queueing networks that can experience deadlock are under-discussed in the literature. This paper addresses this by investigating deadlock properties in these queueing networks. A method of detecting deadlock in discrete event simulations of queueing networks is presented, and markovian models of five open restricted queueing networks are analysed, and the properties of their time to deadlock investigated. Finally a bound on the time to deadlock is given for one of these queueing networks.

Central to the study of deadlock in restricted queueing networks is the concept of blocking. Given two queues in tandem such that customers leaving the first service station enter the second, and the second queue has limited queueing capacity. If the second queue is full, and a customer finishes service at the first queue, that customer cannot join the next queue due to lack of capacity. This customer remains with the server, blocking other customers from beginning service with that server, until space becomes available at the second queue. This is referred to as blocking.

Throughout this paper service centers will be referred to as nodes, and an open unrestricted queueing network will use the following notation for the ith node:

- Λ_i denotes the external arrival rate.
- μ_i denotes the service rate.
- c_i denotes the number of parallel servers.
- n_i denotes the queueing capacity.

- r_{ij} denotes the routing probability from node i to node j upon completion of service at node i.

Exponential service times and Poisson arrivals are assumed.

For the purposes of this paper, deadlock is defined as follows.

Definition 1. *When a simulation is in a situation where at least one service station, despite having arrivals, ceases to begin or finish any more services due to recursive upstream blocking, the system is said to be in deadlock.*

Figure 1 shows a three node queueing network in a deadlocked state. The customers occupying servers B_1 and B_2 are blocked from entering the top node, while the customer occupying server A_1 is blocked from entering the middle node. Due to mutual blocking, these customers are preventing any more natural movement in these two nodes. Note however that on part of the network need be in deadlock, as the bottom node is free to continue services as normal.

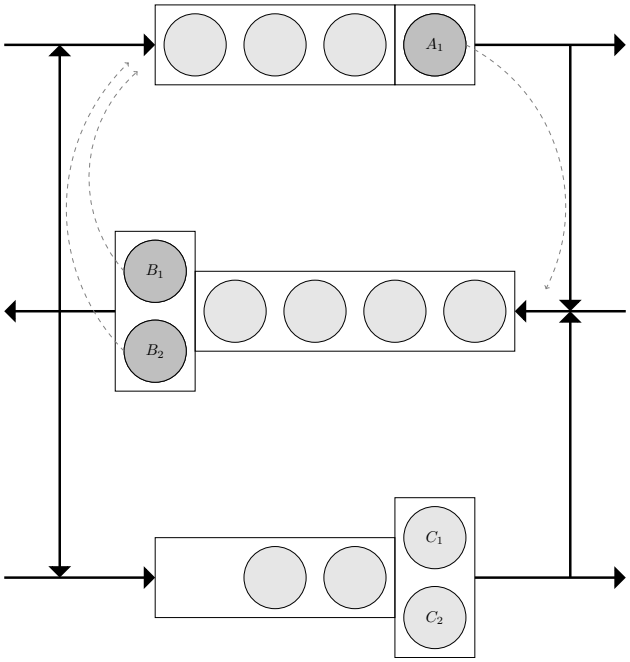

Figure 1: A three node queueing network in deadlock.

This paper is structured as follows: Section 2 discusses the exsisting literature on deadlock and deadlock strategies. Section 3 presents a method of detecting deadlock in discrete event simulations of queueing networks. Section 4 briefly describes the discrete event simulation model used to obtain the results in this paper. Section 5 presents Markov models of five deadlocking queueing networks, derives expected time to deadlock, and compares with resutls obtain through the simulation model. Finally Section 6 derives a bound for the expected time to deadlock for one of the queueing networks discussed.

2 Literature Review

Restricted queueing networks that exhibit blocking are well discussed in the literature [1, 2, 8, 10, 11, 13, 15]. Discussion on restricted queueing networks with feedback loops, that may exhibit deadlock, are sparse however.

General deadlock situations that are not specific to queueing networks are discussed in [5]. Conditions for this type of deadlock, also referred to as deadly embraces, to potentially occur are given:

- Mutual exclusion: Tasks have exclusive control over resources.
- Wait for: Tasks do not release resources while waiting for other resources.
- No preemption: Resources cannot be removed until they have been used to completion.
- Circular wait: A circular chain of tasks exists, where each task requests a resource from another task in the chain.

In open restricted queueing networks the mutual exclusion condition is satisfied as customers cannot share servers; the wait for condition is satisfied due to the blocking rules defined previously; the no preemption condition is satified in networks that have no or non-preemptive priority (this report will only look at networks with no priority); and the circular wait condition is satisfied if the queueing network contains a cycle where all nodes have limited queueing capacity, that is feedback loops.

In general there are three strategies for dealing with deadlock [7, 9]:

- Prevention, in which the system cannot possibly deadlock in the first place.
- Avoidance, in which decisions are made as time unfolds to avoid reaching deadlock.
- Detection and recovery.

2.1 Deadlock Prevention

Deadlock prevention has been discussed in queueing networks. For closed networks of K customers with only one class of customer, [12] proves the following condition to ensures no deadlock: 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 [14], 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.

2.2 Deadlock Avoidance

There are algorithms discussed in the literature for the dynamic avoidance of deadlock. In the Banker's Algorithm [6, 9], unsafe states, those that will lead to deadlock, are avoided by ensuring actions leading to these states are not carried out.

2.3 Deadlock Detection & Recovery

General deadlock detection in systems unspecific to queueing networks are discussed in [5]. A popular method of detecting general deadlock is the use of wait-for graphs, state-graphs and their variants [3, 4, 5, 7]. These wait-for graphs, keep track of all circular wait relations between tasks.

In [5] dynamic state-graphs are defined with resources as vertices and requests as edges. For scenarios where there is only one type of each resource, deadlock arises if and only if the state-graph contains a cylce. In [4] the vertices and edges of the state graph are given labels in relation to a reference node. Using these labels *simple bounded circuits* are defined whose existence within the state graph is sufficient to detect deadlock.

3 Detecting Deadlock

4 Simulation Model

5 Markovian Models of Deadlocking Queueing Networks

5.1 One Node

5.2 Two Node without Self-Loops

5.3 Two Node with Self-Loops

5.4 One Node Multi-Server

5.5 Two Node Multi-Server without Self-Loops

6 A Bound on the Time to Deadlock

References

[1] B. Avi-Itzhak and M. Yadin. A sequence of two servers with no intermediate queue. *Management science*, 11(5):553–564, 1965.

[2] J. Baber. *Queues in series with blocking.* PhD thesis, Cardiff University, 2008.

[3] J. Cheng. Task-wait-for graphs and their application to handling tasking deadlocks. In *Proceedings of the conference on TRI-ADA'90*, pages 376–390. ACM, 1990.

[4] H. Cho, T. Kumaran, and R. Wysk. Graph-theoretic deadlock detection and resolution for flexible manufacturing systems. *IEEE transactions on robotics and automation*, 11(3):413–421, 1995.

[5] E. Coffman and M. Elphick. System deadlocks. *Computing surveys*, 3(2):67–78, 1971.

[6] Edsger W Dijkstra. The mathematics behind the bankers algorithm. In *Selected Writings on Computing: A Personal Perspective*, pages 308–312. Springer, 1982.

[7] A. Elmagarmid. A survey of distributed deadlock detection algorithms. *ACM Sigmod Record*, 15(3):37–45, 1986.

[8] G. Hunt. Sequential arrays of waiting lines. *Operations research*, 4(6):674–683, 1956.

[9] P. Kawadkar, S. Prasad, and A.D. Dwivedi. Deadlock avoidance based on bankers algorithm for waiting state processes. *International journal of innovative science and modern engineering*, 2(12), 2014.

[10] N. Koizumi, E. Kuno, and T.E. Smith. Modeling patient flows using a queueing network with blocking. *Health care management science*, 8(1):49–60, 2005.

[11] R. Korporaal, A. Ridder, P. Kloprogge, and R. Dekker. An analytic model for capacity planning of prisons in the netherlands. *The journal of the operational research society*, 51(11):1228–1237, 2000.

[12] S. Kundu and I. Akyildiz. Deadlock buffer allocation in closed queueing networks. *Queueing systems*, 4(1):47–56, 1989.

[13] G. Latouche and M. Neuts. Efficient algorithmic solutions to exponential tandem queues with blocking. *SIAM journal on algebraic discrete methods*, 1(1):93–106, 1980.

[14] 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.

[15] Y. Takahashi, H. Miyahara, and T. Hasegawa. An approximation method for open restricted queueing networks. *Operations research*, 28(3):594–602, 1980.