On Deadlocking in Queueing Networks

Geraint Ian Palmer

This report will define and discuss the properties and detection of deadlock in queueing networks. Throughout the section, when discussing queueing networks, it is assumed that the queueing network is open and connected. Open queueing networks are those networks that have at least one node to which customers arrive from the exterior, and least one node from which customers leave to the exterior.

1 Introduction

A simple representation of deadlock in the context of road traffic is shown in Figure 1. Here deadlock, or gridlock, is caused by the mutual blocking of vehicles. The blue vehicles are blocked from movement due to the yellow vehicles; the yellow vehicles are blocked from movement due to the green vehicles, the green vehicles are blocked from movement due to the red vehicles; and the red vehicles are blocked from movement due to the blue vehicles. This can be interpreted as the blue vehicles indirectly blocking themselves, and similarly for the other colours.

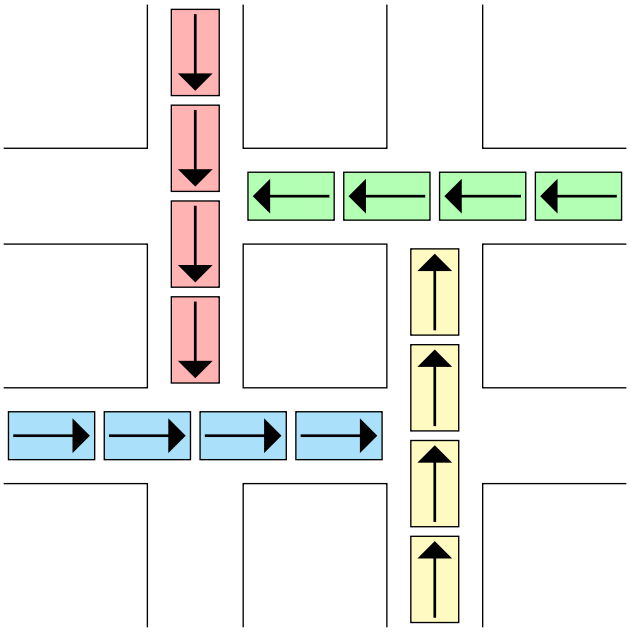


Figure 1: Gridlock, a representation of general deadlock.

This report is structured as follows:

- Section 2 gives an overview of the literature on both resticted queueing networks and general deadlocks.
- Section 3 discusses different types of deadlocks in queueing networks.
- Section 4 defines and explains the state digraph, and demonstrates how to use the state digraph to detect deadlock in queueing network simulations.
- Section 5 defines Markov chain models for five simple queueing networks, and solved using linear algebraic techniques.
- Section 6 gives a bound on the expected time to deadlock for one of the queueing network models given.

Throughout this report, for the ith node of a queueing network the following notation will be used:

- Λ_i denotes the arrival rate
- μ_i denotes the service rate
- r_{ij} denotes the transition probability from node i to node j
- n_i denotes the queueing capacity
- c_i denotes the number of parallel servers

Exponential service rates and Poisson arrivals are assumed.

In open queueing networks with at least one cycle containing all service stations with restricted queueing capacity, deadlock can be experienced. For the purposes of this report, deadlock is defined as follows.

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

In Figure 2 a simple two node queueing network is shown in a deadlocked state. The customer occupying server A_1 has finished service at node A, but remains there as there is not enough queueing space at node B to accept them. The customer at server A_1 is said to be blocked by the customer at server B_1, as they are waiting for that customer to be released. Similarly, the customer occupying server B_1 has finished service at node B, but remains there as there is not enough queueing space at node A, and so the customer at server B_1 is blocked by the customer at server A_1.

When there are multiple servers, individuals become blocked by all customers in service at the destination service station. Figure 3a shows two nodes in deadlock, the customer occupying server A_1 is blocked by customers at both B_1 and B_2, who are both blocked by the customer at A_1. However in 3b, customer at A_1 is blocked by customers at both B_1 and B_2, but the customer at B_2 isn't blocked, and so there is no deadlock.

Note that the whole queueing network need not be deadlocked, only a part of it. If one section of the network is in deadlock, then the system is deadlocked, even though customers may still be able to have services and

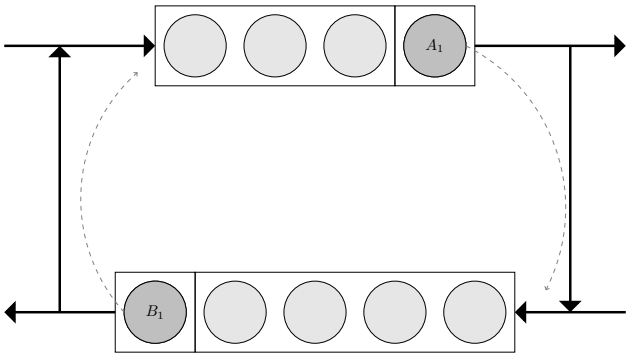
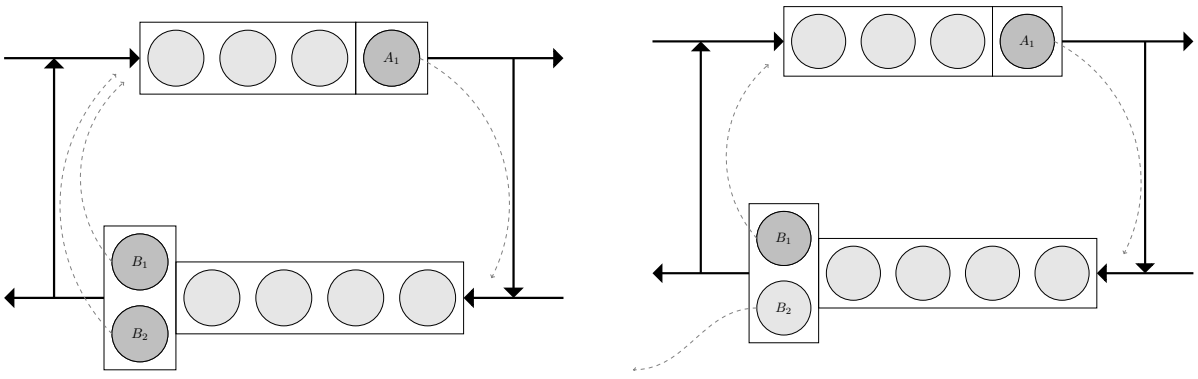


Figure 2: Two nodes in deadlock.



(a) Two nodes in deadlock.

(b) Two nodes not deadlocked

Figure 3: Two nodes: a) in deadlock and b) not in deadlock.

transitions in other areas of the network. An example is shown in Figure 5a. This idea is expanded on in Section 3.

2 Literature Review

Restricted queueing networks that exhibit blocking are well discussed in the literature, both exact [2, 3, 12, 14, 17] and approximate methods [15, 19, 21]. Discussion on restricted queueing networks with feedback loops, that may exhibit deadlock, are sparse however. In fact, the problem of deadlock in queueing networks has either been ignored, not studied, or assumed resolved in much of the literature [19].

General deadlock situations that are not specific to queueing networks are discussed in [8]. These deadlocks occur in flexible manufacturing systems and distributed communication systems. 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 [10, 13]:

- 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 under the blocked at service mechanism (BAS, Type I, transfer blocking). For closed networks of K customers with only one class of customer, [16] 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 [18], 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.

Further conditions on deadlock prvention in closed queueing networks are reviewed in [19], including closed networks under different blocking mechanisms such at blocked before service (BBS, Type II, service blocking) and repetitive services (RS-FD and RS-RD, Type III, rejection blocking).

2.2 Deadlock Avoidance

There are algorithms discussed in the literature for the dynamic avoidance of deadlock. In the Banker's Algorithm [9, 13], unsafe states, those that will lead to deadlock, are avoided by ensuring actions leading to these states are not carried out.

A common deadlock avoidance technique in flexible manufacturing systems is the use of dynamic resource allocation policies using petri net nodels of the system [11, 23]. Some resource allocations are disallowed if that allocaton will lead to deadlock.

Another deadlock avoidance algorithm in [4], for systems with one type of each resource, makes use of a directed resource allocation graph. Resource requests are not allocated resources if that allocation leads to a directed cycle in the resource allocation graph.

2.3 Deadlock Detection & Recovery

A popular method of detecting general deadlock is the use of wait-for graphs, state-graphs and their variants [6, 7, 8, 10]. These wait-for graphs, keep track of all circular wait relations between tasks.

In [8] 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 [7] 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.

In [1], deadlock detection and recovery is listed as one of the two possible soulutions for handling deadlock in queueing networks, although this isn't discussed further.

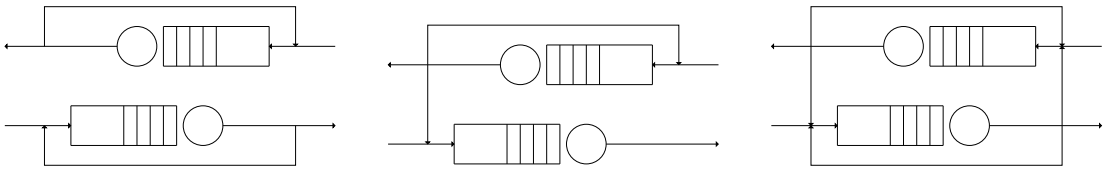
3 Types of Deadlock

In this report the term deadlocked is used to describe a queueing network in deadlock, a node that is part of the deadlock, and components of the network where all nodes are experiencing the same deadlock. For the purposes of this section a queueing network can be described as **connected** and **complete** according to the following definitions.

Definition 2. *A queueing network is connected if there is a possible path from every node to every other node regardless of direction.*

Definition 3. *A queueing network is complete if, from any node, there is a non zero probability of joining any other node of the network directly, including re-entering the same node.*

Figure 4 illustrates these concepts. Note that a complete queueing network is equivalent to the underlying network being a complete graph with loops. Also note that a complete queueing network is connected.



(a) Not connected. (b) Connected but not complete. (c) Complete.

Figure 4: Illustrating connectedness and completeness in queueing networks.

The different configurations of which nodes experience deadlock can be thought of as different types of deadlock.

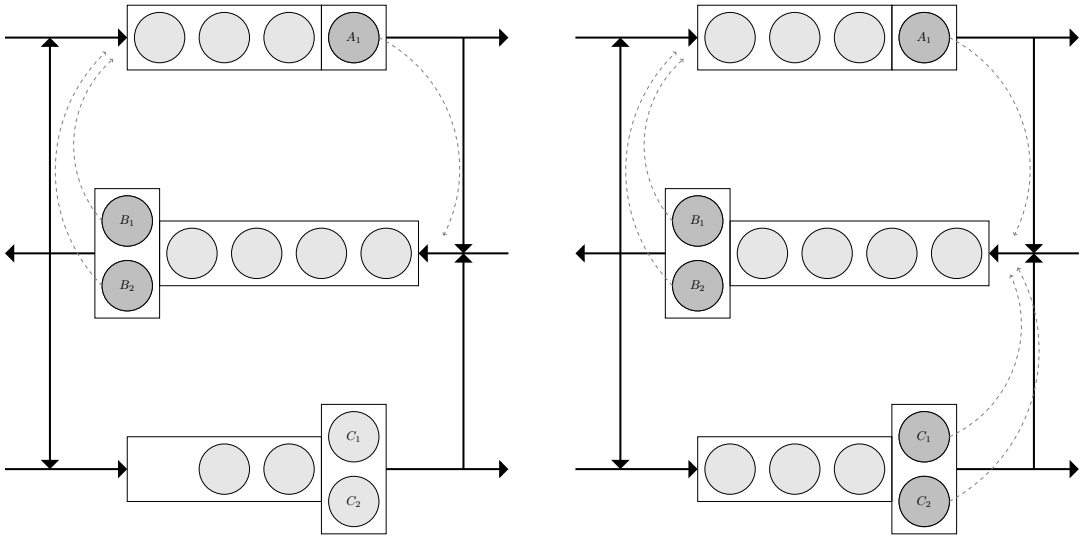
For connected queueing networks, these deadlocks can be classified into **transient deadlocked** states and **absorbing deadlocked** states.

Definition 4. *A transient deadlock state is when there are still some changes of state whilst a subgraph of the queueing network is itself in deadlock.*

Definition 5. *The absorbing deadlock state is when all subgraphs of the queueing network are in deadlock.*

Figure 5 shows a three node network in a transient deadlocked state, and an absorbing deadlocked state. In Figure 5a the occupants of servers B_1 and B_2 are blocked from entering node A; and the occupant of server A_1 is blocked from entering node B, and so these two nodes are in deadlock. However, node C can continue with regular services, until the occupants of every server of C attempt to join a deadlocked node. At which point, the whole system is deadlocked, and so has reached absorbing deadlock, shown in Figure 5b.

It should be clear that if the queueing network is connected, then there is a non-zero probability that once one part of the network is in deadlock, all nodes will fall into a deadlocked state, simply by the individuals in the non-deadlocked nodes attempting to transition into a deadlocked node, or those non-deadlocked nodes also becoming deadlocked themselves. That is, once a queueing network falls into one of the transient deadlocked states, it will eventually transition, either directly or through other transient deadlocked states, into an absorbing deadlocked state.



(a) Transient deadlock. (b) Absorbing deadlock.

Figure 5: Types of deadlock: a) transient and b) absorbing.

3.1 Deadlock types in Complete Queueing Network

Figure 6 illustrates the number of different types of deadlock configurations possible in a complete queueing network with four nodes.

Before proceeding, remember that a Bell number B_n is the number of ways a set of n elements can be partitioned into non-empty subsets [5]. Now the following theorem gives the number of possible deadlocked states in a complete queueing network.

Proposition 1. *If the queueing network Q with N nodes is complete, and every node has finite queueing capacity, then the number of possible deadlocked states is $B_{N+1} - 1$, where B_n denotes the nth Bell number.*

Proof.

- Define S_N as a set of N nodes.
- Define D_N as the set of possible deadlocks on a queueing network Q with N nodes.
- Define ξ as a dummy node.
- Define P_N as the set of all partitions of $S_N \cup \{\xi\}$ not including the set that counts all nodes in one partition.

By definition $|P_N| = B_{N+1} - 1$.

Consider the mapping $g : D_N \mapsto P_N$ defined by:

- Order the nodes of Q and S_N.
- The node $x \in Q$ maps to the corresponding node node $x' \in S_N$.
- Partition $S_N \cup \{\xi\}$ in such a way that:

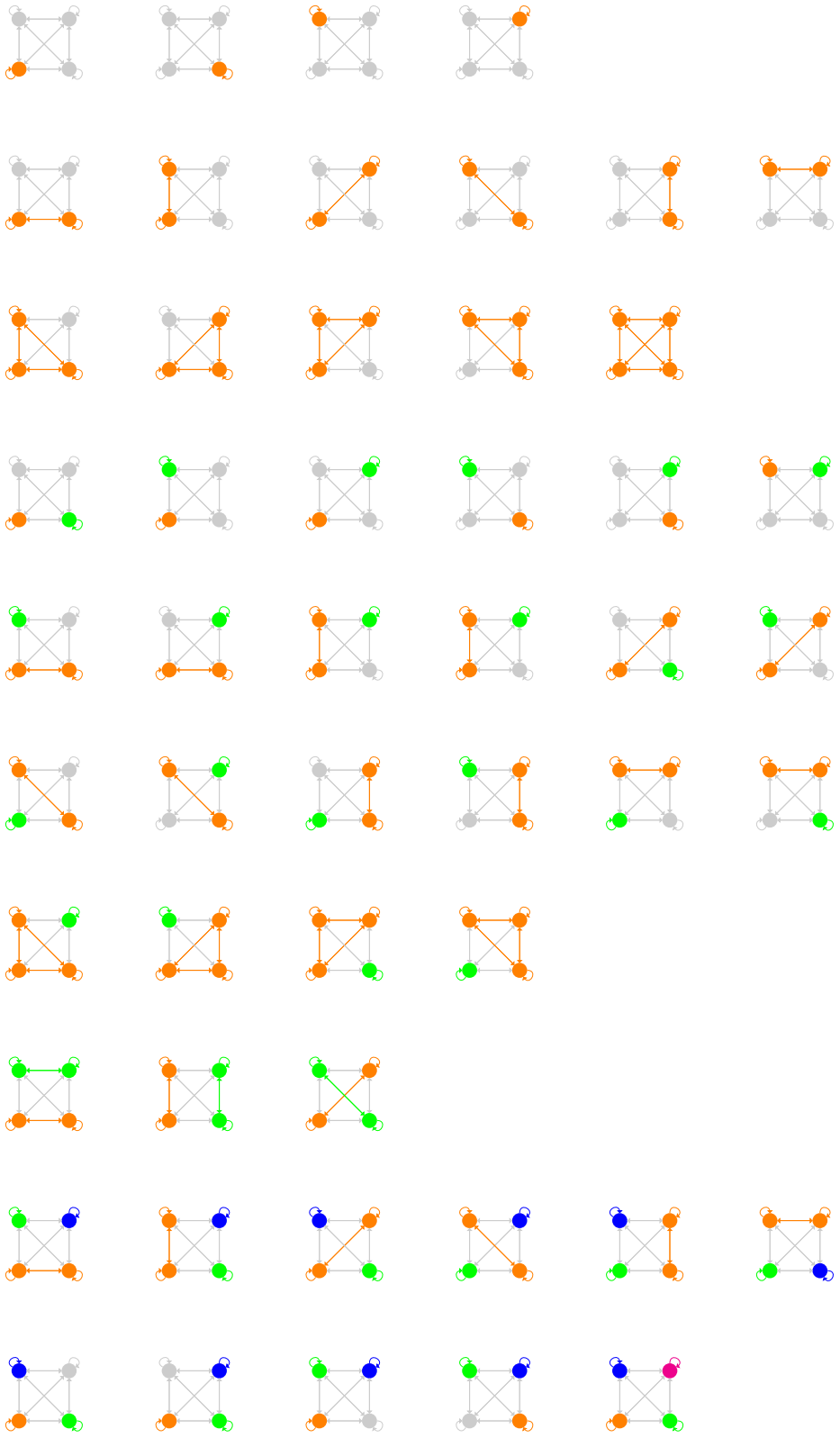


Figure 6: The $51 = B_5 - 1$ configurations of a 4 node complete queueing network.

- For all $x \in Q$ that are in the same deadlocked component, all corresponding $x' \in S_N$ are placed in the same partition of P_N.
- For all $x \in Q$ that are not deadlocked, all corresponding $x' \in S_N$ are placed in the same partition as ξ.

Consider the inverse mapping $g^{-1} : P_N \mapsto D_N$:

- Consider a set $Y \in P_N$:
  - If $\xi \in Y$ then all other elements of Y correspond to nodes in Q that are not deadlocked.
  - If $\xi \notin Y$ then all other elements of Y correspond to nodes in Q that are deadlocked together.

The mapping g is both 1-1 and onto, and so is a bijection. Therefore $|D_N| = |P_N| = B_{N+1} - 1$.

□

Figure 7 gives an example of the mapping g for $N = 2$ and $N = 3$.

Proposition 2. *If the queueing network Q with N nodes is complete, and every node has finite queueing capacity, then the number of possible absorbing deadlocked states is B_N, and the number of possible transient deadlocked states is $B_{N+1} - B_N - 1$.*

Proof. In an absorbing deadlocked state every node is involved in some configuration of deadlock. The number of deadlock configurations where there are no undeadlocked nodes is simply the number of partitions of N nodes, where each partition represents a separate deadlocked component. Therefore the number of absorbing states is B_N from the definition of a Bell number.

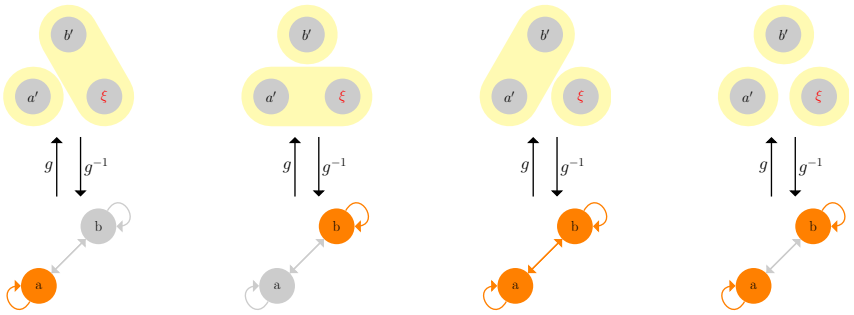
A deadlocked state is either absorbing or transient. If there are B_N absorbing states, and $B_{N+1} - 1$ in total, then there must be $B_{N+1} - BN - 1$ transient deadlocked states. □

Some deadlock configurations cannot be reached without first having gone through another deadlock state. For example, a deadlock configuration that consists of two separate components in deadlock must have been in a state with one deadlocked component prior to reaching that configuration.

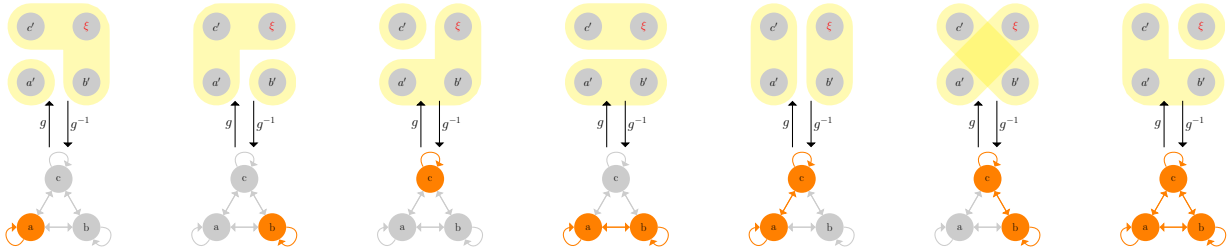
The remainder of this report will only be concerned with the first instance of deadlock, that is the deadlocked states that can be reached from a non-deadlocked state directly. That is deadlock configurations with only one deadlocked component. The number of different deadlocked states that can reached at first instance is equal to the number of directed cycles where every node has finite queueing capacity.

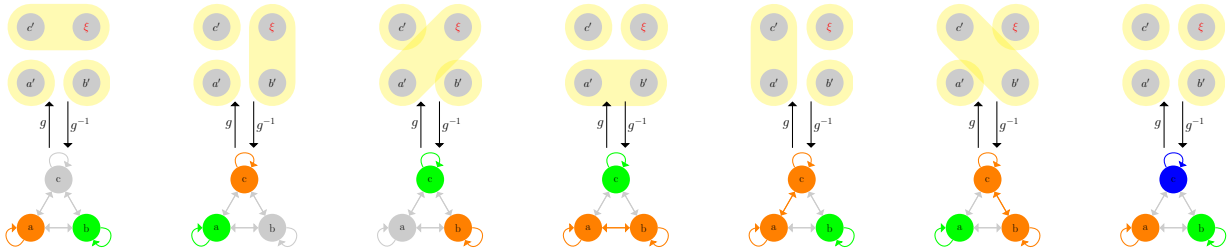
4 Detecting Deadlock

In the following subsections, a method will be presented to detect deadlock in discrete event simulations of queueing networks.



(a) Example of g for $N = 2$





(b) Example of g for $N = 3$

Figure 7: Example of g for $N = 2$ and $N = 3$.

4.1 State Digraph

A method for detecting when deadlock occurs in an open queueing network Q with N nodes, using a dynamic directed graph, the state digraph, is now proposed.

Let the number of servers in node i be denoted by c_i. Define $D(t) = (V(t), E(t))$ as the state digraph of Q at time t.

The vertices at time t, $V(t)$ correspond to servers in the queueing system. Thus, $|V(t)| = \sum_{i=1}^{N} c_i$ for all $t \geq 0$.

The edges at time t, $E(t)$ correspond to a blocking relationship. There is a directed edge at time t from vertex $X_a \in V(t)$ to vertex $X_b \in V(t)$ if and only if an individual occupying the server corresponding to vertex X_a is being blocked by an individual occupying the server corresponding to vertex X_b.

The state digraph $D(t)$ can be partitioned into N service-station subgraphs, $D(t) = \bigcup_{i=1}^{N} d_i(t)$, where the vertices of $d_i(t)$ represent the servers of node i. The vertex set of each subgraph is static over time, however their edge sets may change.

The state graph is dynamically built up as follows. 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 vertices of $d_j(t)$ are created in $D(t)$.

When an individual is released and another customer who wasn't blocked occupies their server, that servers out-edges are removed. When an individual is released and another customer who was previously blocked occupies their server, that server's out-edges are removed along with the in-edge from the server who that previously blocked customer occupied. When an individual is released and there isn't another customer to occupy that server, then all edges incident to that server are removed.

4.1.1 Adding Edges

This general process of building up the state digraph as the queueing network is simulated will now be shown. Customers are labelled (i, j, k) where i denotes the server that customer is occupying, j denotes that individuals i.d. number, and k denotes the service station that customer is waiting to enter. As an example, a customer labelled $(A_2, 10, C)$ would have an i.d. number of 2, is occupying server A_2 and is currently waiting to join node C. If a customer isn't occupying a server the notation $\emptyset$ is used. Similarly for customers occupying a server and still in service, their next destination is yet undecided, so $\emptyset$ is used.

Consider the simulation that starts with full queues, and every server occupied by a customer in service. This is shown in Figure 8.

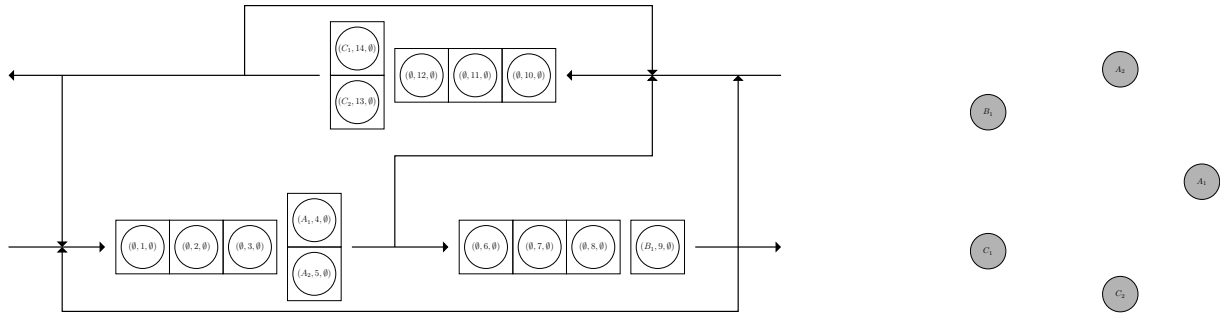


Figure 8

Customer 13 finishes service and is blocked from entering node A (Figure 9).

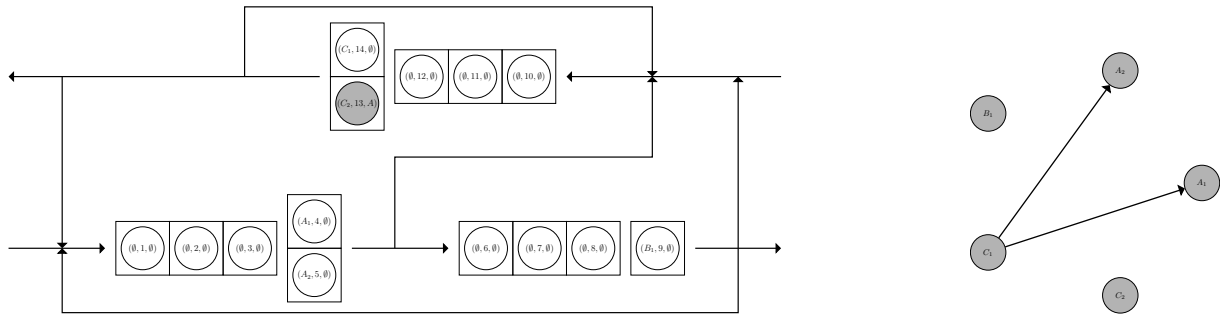


Figure 9

Then customer 4 finishes service and is blocked from entering node B (Figure 10).

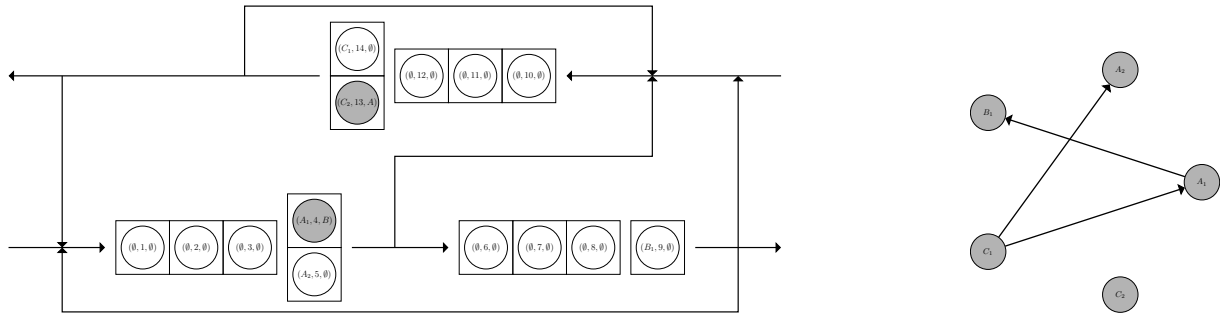


Figure 10

Then customer 9 finishes service and is blocked from entering node A (Figure 11).

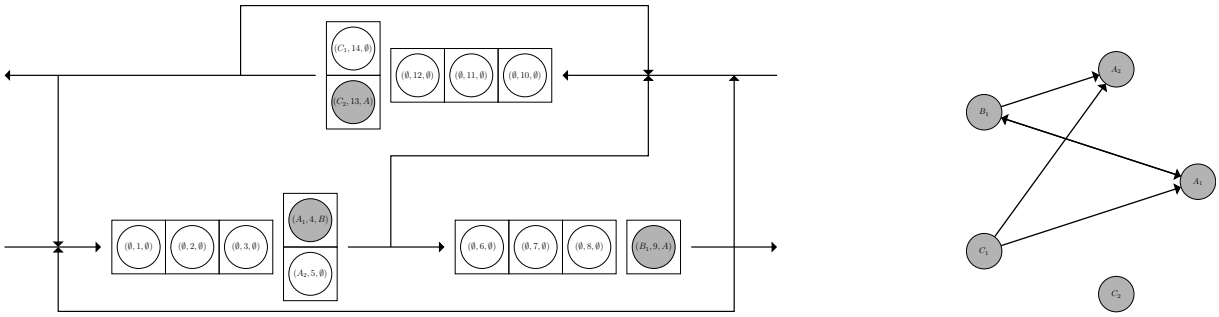


Figure 11

Finally, in Figure 12 customer 5 finishes service and wants to re-enter the queue for node A but is blocked. A deadlock situation arises as customer 5 is waiting for customer 4 to move, who is waiting for customer 9 to move, who is waiting for either customer 4 or 5 to move.

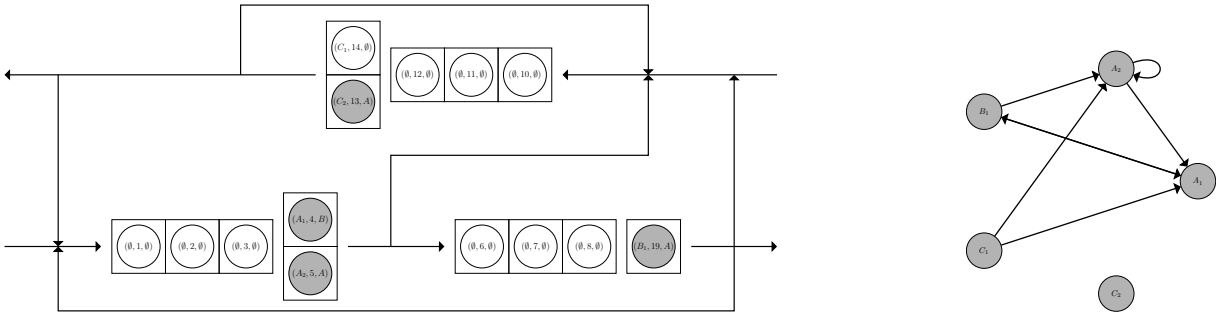


Figure 12

4.1.2 Removing Edges

The rules on how edges are removed from the state graph will now be shown. For illustrative purposes the queueing network here is a different queueing network than discussed above.

This simulation begins with four customers occupying servers; those at node A blocked to node B, the customer at node C blocked to node A, and the customer at node B still in service. This is shown in Figure 13.

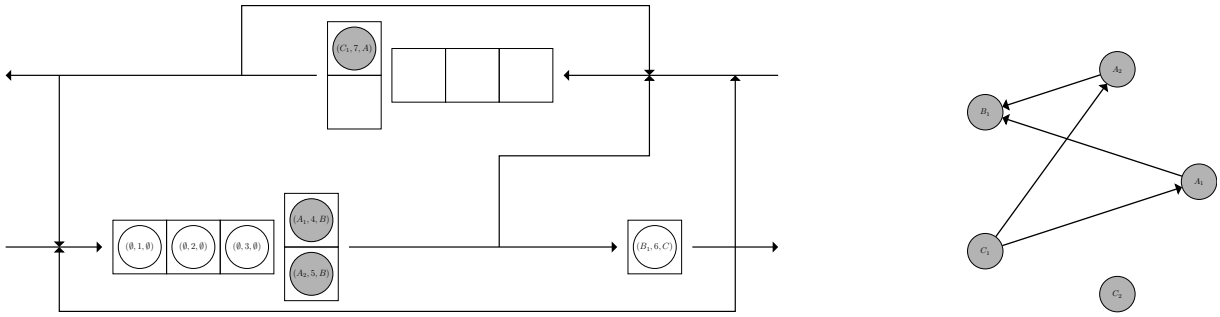


Figure 13

Customer 6 finishes service and immediately joins service at node C (Figure 14).

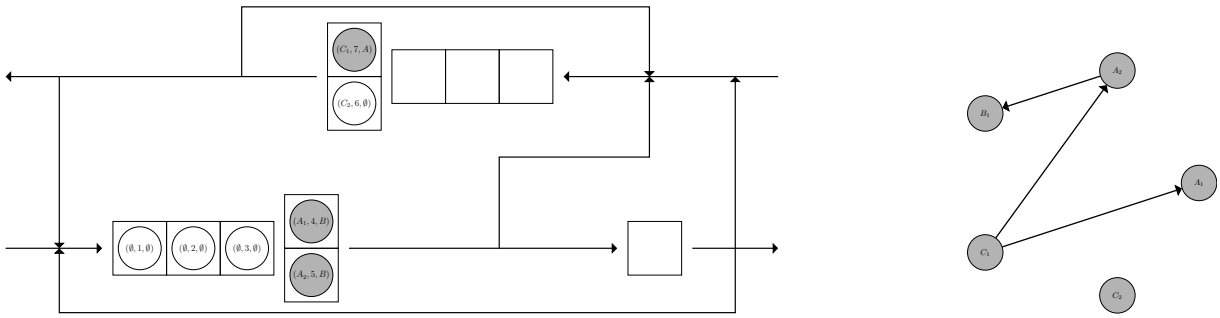


Figure 14

Now there is room for customer 4 to move into service at node B (Figure 15). Notice that the edge $A_2 \longrightarrow B_1$ remains in the state graph, as customer 5 is still blocked by that server.

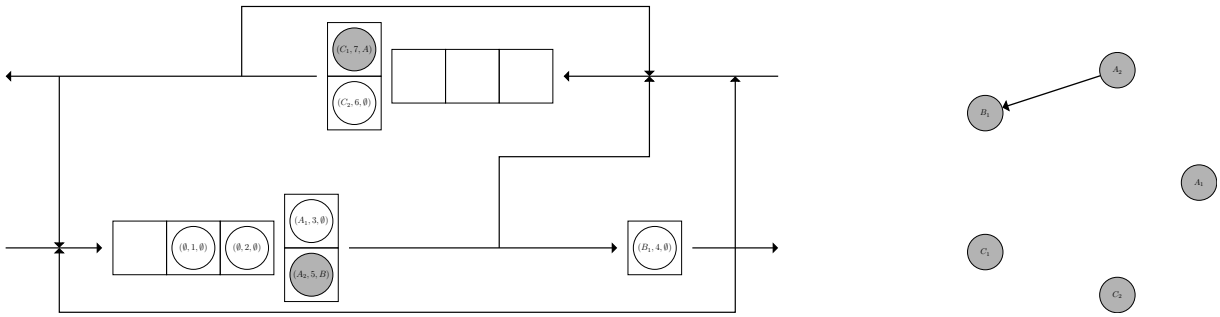


Figure 15

The customers queueing at node A move along the queue, with customer 3 beginning service. This leaves enough room for customer 7 to join the back of the queue at A (Figure 16).

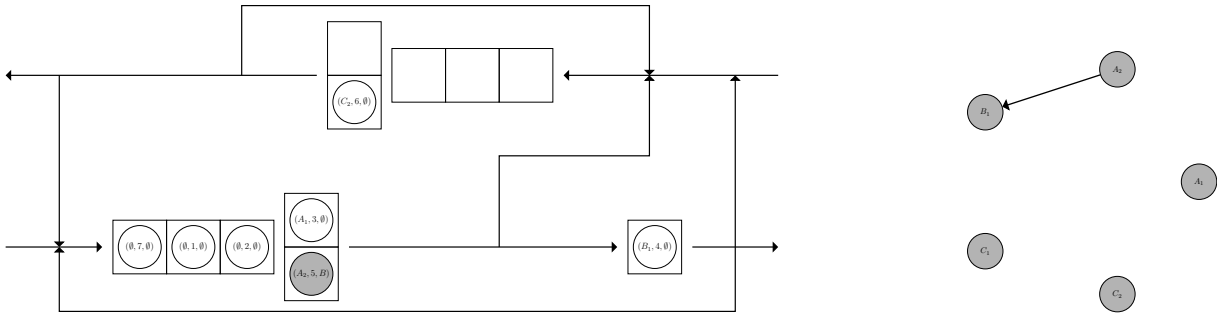


Figure 16

4.1.3 Observations

Consider one weakly connected component $G(t)$ of $D(t)$. Consider the node $X_a \in G(t)$. If X_a is unoccupied, then X_a has no incident edges. Next 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 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 $X_a \in G(t)$ 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 all of X_a's descendants are occupied by blocked individuals, then the system is deadlocked at time t. We also know that by definition all of X_a's ancestors are occupied by blocked individuals.

Also note that if a service-station subgraph $d_i(t)$ contains edges, then there is an individual in $X_a \in d_i(t)$ that is being blocked by himself. This does not necessarily mean there is deadlock.

4.2 Results on the State Digraph

The definition of a knot is given in the Appendix. The following result is used to identify deadlock:

Theorem 1. *A deadlocked state arises at time t if and only if $D(t)$ contains a knot.*

Proof. Consider one weakly connected component $G(t)$ of $D(t)$ at time t. All vertices of $G(t)$ are either descendants 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 $G(t)$ contains a vertex X such that $\deg^{\text{out}}(X) = 0$, and there is a path from every other non-sink vertex to X. This implies that X's occupant is not blocked and is a descendant of another vertex. Therefore Q is not deadlocked as there does not exist a vertex whose descendants are all blocked.

Now assume that we have deadlock. For a vertex X who is deadlocked, all descendants of X are are occupied by individuals who are blocked, and so must have out-degrees greater than 0. And so there is no path from X to a vertex with out-degree of 0.

□

For queueing networks with certain properties, this results can be simplified:

Proposition 3. *For queueing networks:*

1. *with one node*
2. *with two nodes, each with two or fewer parallel servers*
3. *with a finite amount of nodes, each with a single server*

a deadlocked state arises if and only if a weakly connected component without a sink node exists in $D(t)$.

Proof. 1. Consider a one node queueing network.

If there is deadlock, then all servers are occupied by blocked individuals, and so all servers have an out-edge.

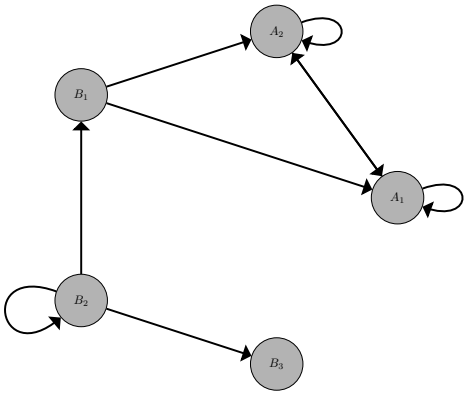
2. Consider a two node queueing network, each node with 2 or fewer parallel servers.

If both nodes are involved in the deadlock, so there is a customer in node 1 blocked from entering node 2, and a customer from node 2 blocked from entering node 1, then all servers in node 1 and node 2 in $D(t)$ will have out edges as they are occupied by a blocked individual. The servers of node 1 and 2 consist of the entirety of $D(t)$, and so there is no sink nodes.

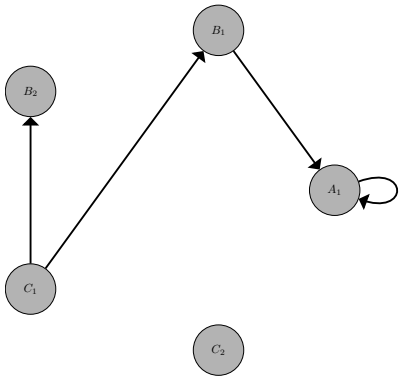
Now consider the case when only one node is involved in the deadlock. Without loss of generality, let's say that node 1 is in deadlock with itself, then the servers of node 1 have out-edges. For the servers of node 2 to be part of that weakly connected component, there either needs to be an edge from a server in node 1 to a server in node 2, or and edge from a server in node 2 to a server in node 1. An edge from a server in node 1 to a server in node 2 implies that a customer from node 1 is blocked from entering node 2, and so node 1 is not in deadlock with itself. An edge from a server in node 2 to a server in node 1 implies that a customer in node 2 is blocked from entering node 1. In this case one server in node 2 has an out-edge. Now either the other server of node two is still in server, and so isn't part of that weakly connected component, or the other server's customer is blocked and so has an out edge.

For the case of a queueing network with two nodes and more than 2 servers at node 2, consider the following counter-example:

Node A has two parallel servers, node B has three parallel sevrers. Begin with all servers occupied by customers in service and full queues. The customer at server A_1 is blocked to node A. The customer at server B_1 is blocked to node A. The customer at server B_2 is blocked to node B. The customer at server A_2 is blockec to node A. The resulting state digraph in Figure 17a has a weakly connected component with a sink.



(a) State Digraph of Counter-Example 1.



(b) State Digraph of Counter-Example 2.

3. Consider a queueing network with N nodes, each with a single server.

   If $1 \leq n \leq N$ nodes are involved in the deadlock, then each server in those n nodes has a blocked customer, and so has an out-edge. Of the other nodes, they can only be in the same weakly connected component if either they contain a indiviual blocked by those in deadlock, in which case they will have an out edge; or they contain a blocked individual blocked by those directly or indirectly blocked by those in deadlock, in which case they will have an out edge; or they are blocking someone who is blocked directly or indirectly by those in deadlock. However this last case cannot happen, as every node is single server each person can only be blocked by one other individual at a time.

   For the case of a queueing network with more than two nodes with multiple servers, the following counter-example proves the claim:

   Node A has one parallel server, node B has two parallel servers, and node C has three parallel servers. Begin with all servers occupied by customers in service and full queues. The customer at server B_1 is blocked from entering node A. Then the customer at server C_1 is blocked from entering node B. Then the customer at server A_1 is blocked from entering node A. The resulting state digraph in Figure 17b has a weakly connected component with a sink.

□

Absorbing deadlocked states can be found using a simpler result:

Proposition 4. *An absorbing deadlocked state arises at time t if $D(t)$ doesn't contain a sink vertex.*

Proof. A vertex with out-degree greater than zero represents an occupied server whose occupant has finished service and is blocked. If all vertices have out-degree greater than zero, then all servers are occupied by blocked individuals. A release at vertex X_a can only be triggered by one of X_a's descendants finishing service. As all servers are occupied by blocked individuals, no server can finish service, and so no server can release their occupant, implying an absorbing deadlocked state.

□

Finding knots is discussed in Section 4.3.

4.3 Finding Knots

By definition knots are strongly connected subgraphs where every member has descendants, and every member's descendants belong to that subgraph.

An implementation of an algorithm to find strongly connected components in the Python package NetworkX is used, in the following algorithm. This brute force algorithm was adapted from the NetworkX developer zone ticket #663 (`https://networkx.lanl.gov/trac/ticket/663`), is sufficient to identify knots in a directed graph.

Find the strongly connected subgraphs of $D(t)$
for *strongly connected subgraph SCS in list of strongly connected subgraphs of $D(t)$* **do**
    for *vertex v in SCS* **do**
        if *number of v's descendants > number of vertices in SCS* **then**
            Add SCS to list of knots
            break
        end
    end
end

---

5 Markov Chain Models

This section will build analytical models of systems' behaviour with regards to deadlock. The section is structured as follows:

- Section 5.1 presents a general Markov model of a deadlocking queueing system.
- Section 5.2 gives the Markov formulation for a one node queueing network with single servers.
- Section 5.3 gives the Markov formulation for a two node queueing network with single servers without self loops.
- Section 5.4 gives the Markov formulation for a two node complete queueing network with single servers.
- Section 5.5.1 gives the Markov formulation for a one node queueing network with multiple servers.
- Section 5.5.2 gives the Markov formulation for a two node queueing network with multiple servers without self loops.

5.1 Analysis of Markov Chain Models

This section will discuss the techniques used in subsequent sections to analyse specific Markov chain models of deadlock.

In general a continuous Markov chain model of a deadlocking queueing network is a series of states $s \in S$ and the transition rates between states q_{s_1,s_2}. Each state s uniquely defines a configuration of customers around the queueing network. Deadlocked states are also present, either denoted by that specific configuration of customers, or by negative numbers, for example -1.

In the Markov chain models discussed here, only the first instance of deadlock is considered. This may be a transient deadlocked state, however once this state is reached further transitions are not considered, and so all deadlocked states cannot transition to any other state, and so is an absorbing state of the Markov chain. Therefore any queueing network that can experience deadlock is guaranteed to experience deadlock, as absorbing Markov chains are guaranteed to enter one of its absorbing states.

The expected time until deadlock is reached is equivalent to the expected time to absorbtion of the Markov chain, which can easily be found [20]. The canonical form of an absorbing Markov chain is

$$P = \begin{pmatrix} T & U \\ 0 & I \end{pmatrix}$$

where I is the identity matrix.

Now the expected number of time steps until absorption starting from state i is the ith element of the vector

$$(I - T)^{-1} e \tag{1}$$

where e is a vector of 1s.

Therefore by discretising the continuous Markov chain and ensuring the correct order of states, the number expected of time steps to absorption, or deadlock can be found.

When there are more than one deadlock state, there is more than one absorbing state in the Markov chain. The probabilities of which absorbing state a Markov chain will reach are given by the $(i, j)^{\text{th}}$ element of the vector

$$(I - T)^{-1} U \tag{2}$$

which corresponds to the probability of reaching absorbing state j from transient state i.

5.2 One Node Network

Consider the one node network with feedback loop shown in Figure 18. This shows an $M/M/1$ queue with room for n customers to queue at any one time, customers arrive at a rate of Λ and served at a rate μ.

Once a customer has finished service they rejoin the queue with probability r_{11}, and so exit the system with probability $1 - r_{11}$.

Let this system be denoted by Ω_1 with parameter set (Λ, μ, n, r_{11}), and the time to deadlock of this system be denoted by ω_1.

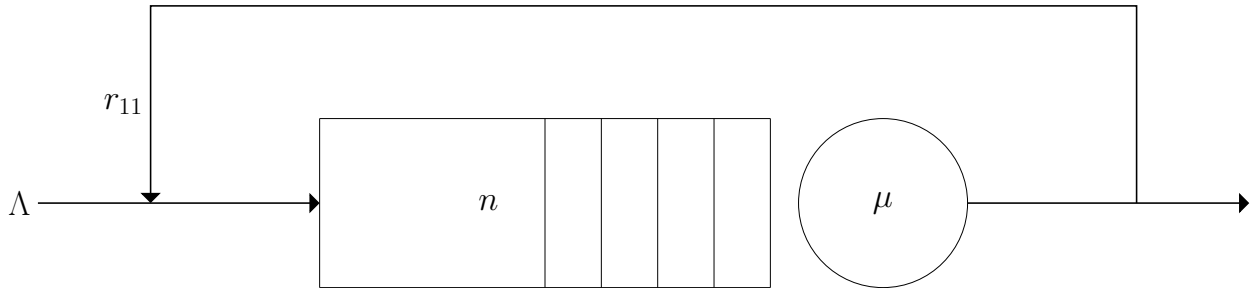


Figure 18: A one node queueing network.

State space:

$$S = \{i \in \mathbb{N} \mid 0 \leq i \leq n+1\} \cup \{-1\}$$

where i denotes the number of individuals in service or waiting, and -1 denotes the deadlocked state.

If we define $\delta = i_2 - i_1$ for all $i_k \geq 0$ then the transitions are given by:

$$q_{i_1,i_2} = \begin{cases} \left.\begin{matrix} \Lambda & \text{if } i < n+1 \\ 0 & \text{otherwise} \end{matrix}\right\} & \text{if } \delta = 1 \\ (1 - r_{11})\mu & \text{if } \delta = -1 \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

$$q_{i,-1} = \begin{cases} r_{11}\mu & \text{if } i = n+1 \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

and

$$q_{-1,i} = 0 \tag{5}$$

The Markov chain is shown in Figure 19.

Figure 20 shows the effect of varying the parameters of the queueing network on times to deadlock. Base parameters of $\Lambda = 10$, $n = 3$, $\mu = 5$ and $r_{11} = 0.25$ were used.

It can be seen that increasing the arrival rate Λ and the transition probability r_{11} results in reaching deadlock faster. This is intuitive as increasing these parameters results in the first node's queue filling up quicker. Increasing the queueing capacity n results in reaching deadlock slower. Again this is intuitive, as increasing the queueing capacity allows more customers in the system before becoming deadlock.

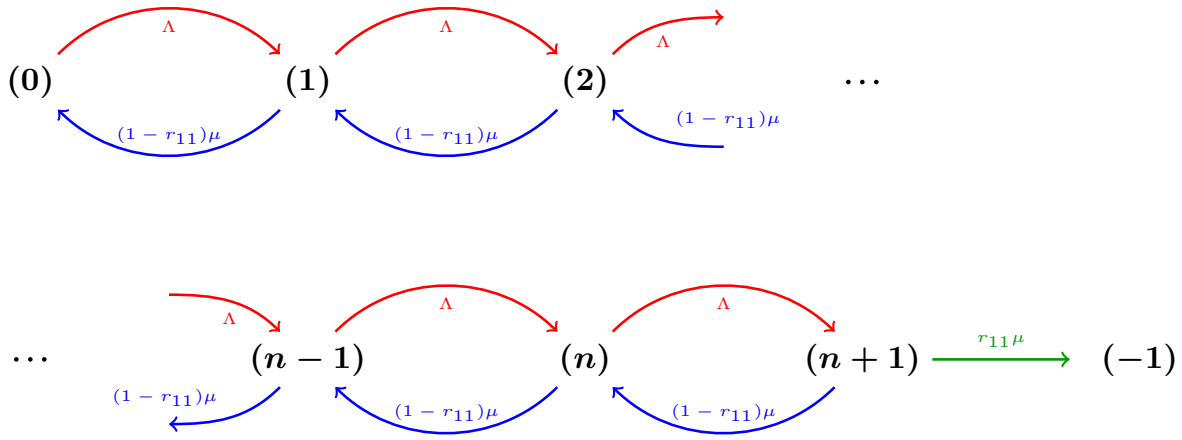
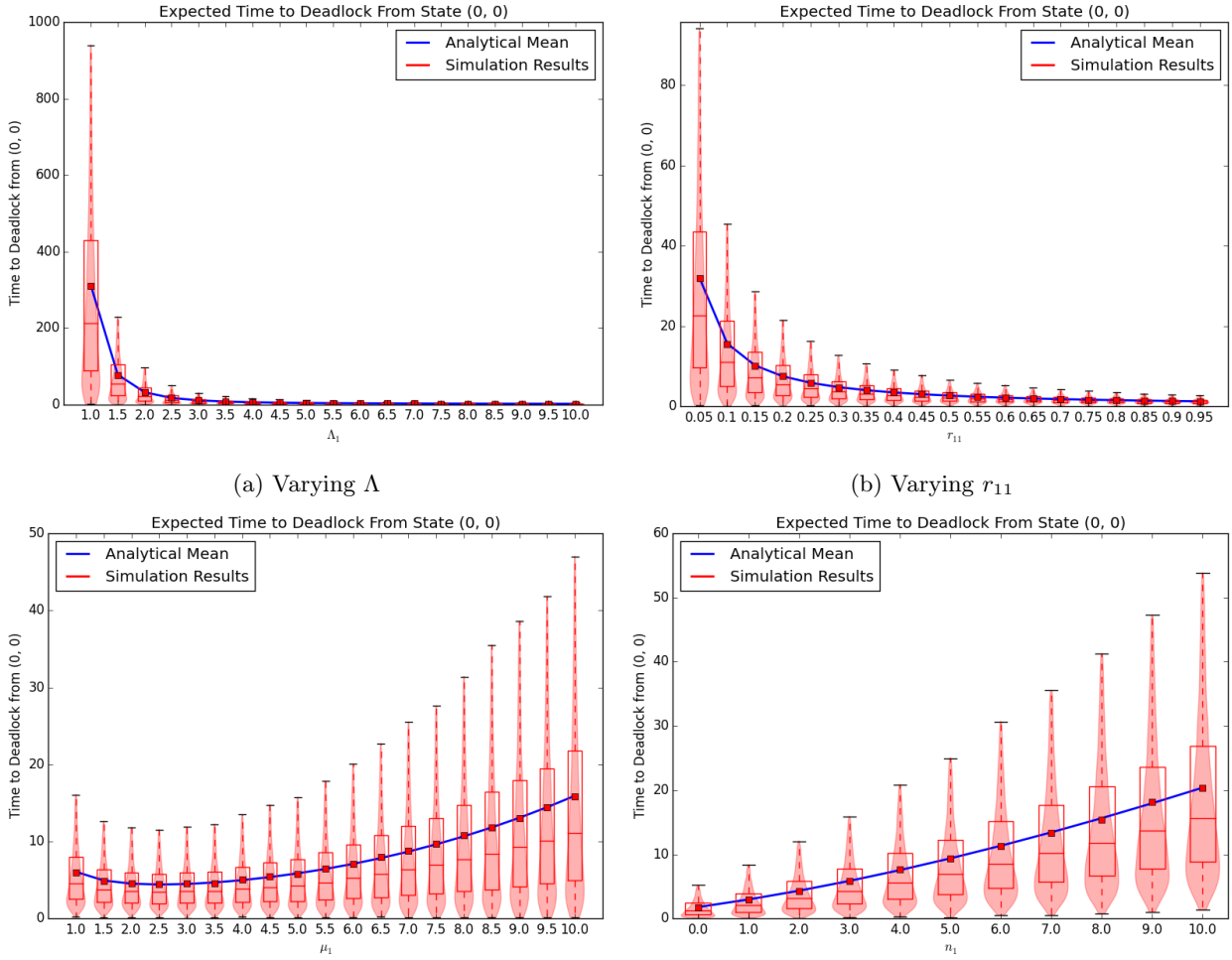


Figure 19: Diagrammatic representation of the Markov chain for Ω_1.



(a) Varying Λ

(b) Varying r_{11}

(c) Varying μ

(d) Varying n

Figure 20: Time to deadlock in Ω_1, analytical & simulation results (10,000 iterations).

The behaviour as the service rate μ varies is not monotonic, as the service rate contributed towards both moving customers from the system and allowing customers to rejoin the queue, causing blockages and deadlock. This behaviour is described in the following remark.

Remark 1. *The function $\omega_1(\mu)$ that describes the time to deadlock of an Ω_1 system as the service rate μ varies, and all other parameters are fixed, has one critical point and is a local minimum for $\mu \in (0, \infty)$.*

This can be seen by interpretation. The behaviour of $\omega_1(\mu)$ can be interpreted as follows:

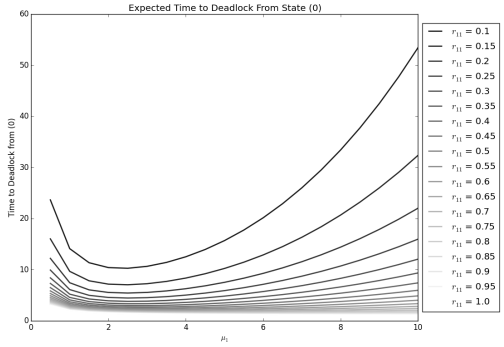
- At $\lim_{\mu \to 0} \omega_1(\mu)$ there is infinite service time, and so infinite time until deadlock.
- At $\lim_{\mu \to \infty} \omega_1(\mu)$ there is zero service time, the queue can never fill up, and so infinite time to deadlock.
- At low service rates below a certain threshold $\hat{\mu}$, the arrival rate is relatively large compared to the service rate, and we can assume a saturated system. At this point services where a customer exits the system does not have much of an effect, as we can assume another arrival immediately. However services where a customer wishes to rejoin the queue results in a blockage as the system is saturated. Therefore, increasing the service rate here increases the chance of a blockage, and so the chance of deadlock.
- Above $\hat{\mu}$ the service rate is large enough that we cannot assume a saturated system, and so services where the customer exits the system does have an affect on the number of customers in the system. Thus increasing the service rate removes people from the system, and as such there is less chance of getting blocked and deadlocked.

This effect is closely related to the transition rate r_{11}, as the rate at which the system enters deadlock from a full queue is $r_{11}\mu$. Figure 21a shows the effect of the transition rate on the behaviour of varying the service rate.

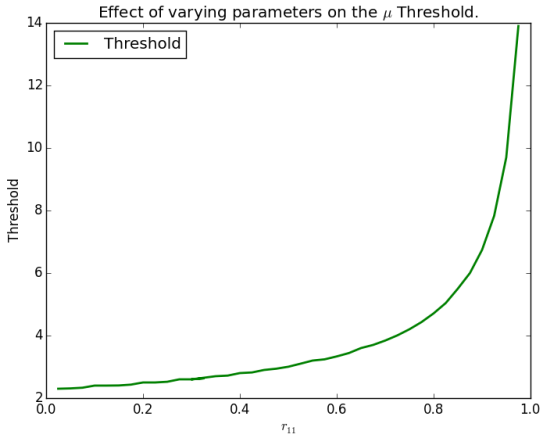
Figures 21b, 21c and 21d show the effect of varying r_{11}, Λ and n on this threshold $\hat{\mu}$ respectively. In Figure 21d it is shown that increasing the queueing capacity lowers the threshold; this is due to the system becoming saturated easier at lower queueing capacities and therefore requiring a larger service rate to escape this saturated zone. In Figure 21b it is shown that increasing the rejoining probability raises the threshold. Lower rejoining probabilities make the system escape the saturated zone quicker, and only a low service rate is required to escape; however at high rejoin probabilities a higher service rate fills the system up quicker, and so an even higher service rate is required to escape the saturated zone. Figure 21c implies a positive linear relationship between the arrival rate and the threshold. Again, at lower arrival rates the system does not fill up as easily, and so is easier to escape the saturated zone.

Remark 2. $\arg\max_{\mu \in [a,b]} \omega_1(\mu)$ *is either a or b.*

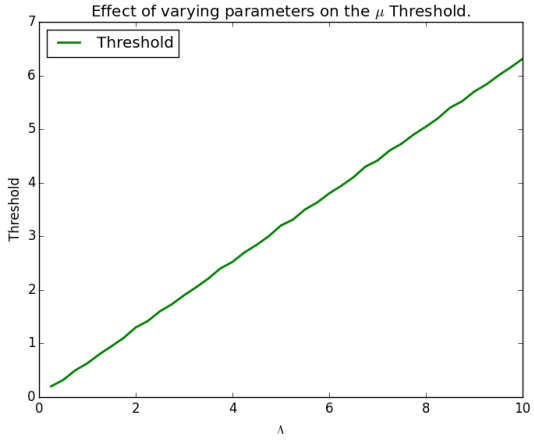
From the closed interval method of finding absolute maximum [22], the absolute maximum of $\omega_1(\mu)$ on the closed interal $[a, b]$ is either the critical points in (a, b), a or b. The only critical point in (a, b) is $\hat{\mu}$, and is a local minimum (from Remark 1), and so $\hat{\mu} \leq a$ and $\hat{\mu} \leq b$. Therefore $\omega_1(\mu)$ obtains its maximum at either a or b.



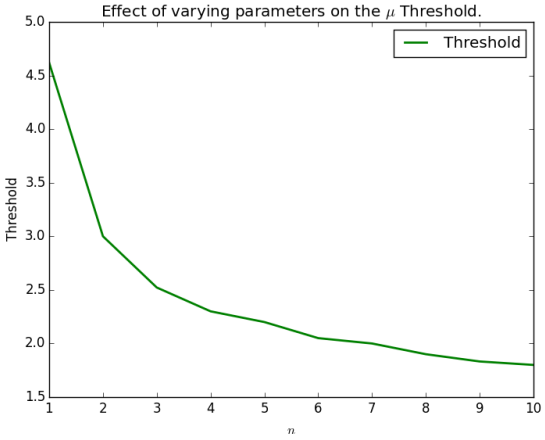
(a) The effect of r_{11} and μ on times to deadlock.



(b) The effect of r_{11} on $\hat{\mu}$.



(c) The effect of Λ on $\hat{\mu}$.



(d) The effect of n on $\hat{\mu}$.

Figure 21: Investigating the service rate threshold.

5.3 Two Node Network without Self Loops

Consider the queueing network shown in Figure 22. This shows two $M/M/1$ queues, with n_i queueing capacity at at each service station and service rates μ_i. Λ_i is the external arrival rates to each service station. All routing possibilities except self loops are possible, where the routing probability from node i to node j is denoted by r_{ij}.

Let this system be denoted by Ω_2 with parameter set (Λ_1, Λ_2, μ_1, μ_2, n_1, n_2, r_{12}, r_{21}).

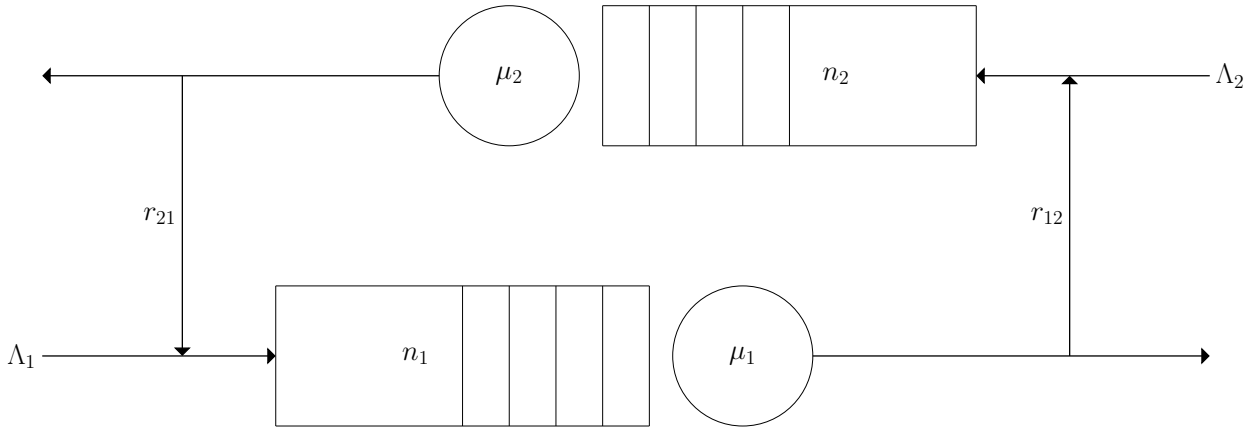


Figure 22: A two node queueing network.

- State space:

$$S = \{(i,j) \in \mathbb{N}^{(n_1+2\times n_2+2)} \mid 0 \leq i+j \leq n_1+n_2+2\} \cup \{(-1)\}$$

  where i denotes the number of individuals:

  - In service or waiting at the first node.
  - Occupying a server but having finished service at the second node waiting to join the first.

  where j denotes the number of individuals:

  - In service or waiting at the second node.
  - Occupying a server but having finished service at the first node waiting to join the second.

  and the state (-1) denotes the deadlocked state.

If we define $\delta = (i_2, j_2) - (i_1, j_1)$ for all $(i_k, j_k) \in S$, then the transitions are given by:

$$q_{(i_1,j_1),(i_2,j_2)} = \begin{cases} \left.\begin{array}{rr} \Lambda_1 & \text{if } i_1 < n_1 + 1 \\ 0 & \text{otherwise} \end{array}\right\} & \text{if } \delta = (1,0) \\ \left.\begin{array}{rr} \Lambda_2 & \text{if } j_1 < n_2 + 1 \\ 0 & \text{otherwise} \end{array}\right\} & \text{if } \delta = (0,1) \\ \left.\begin{array}{rr} (1-r_{12})\mu_1 & \text{if } j_1 < n_2 + 2 \\ 0 & \text{otherwise} \end{array}\right\} & \text{if } \delta = (-1,0) \\ \left.\begin{array}{rr} (1-r_{21})\mu_2 & \text{if } i_1 < n_1 + 2 \\ 0 & \text{otherwise} \end{array}\right\} & \text{if } \delta = (0,-1) \\ \left.\begin{array}{rr} r_{12}\mu_1 & \text{if } j_1 < n_2 + 2 \text{ and } (i_1,j_1) \neq (n_1+2,n_2) \\ 0 & \text{otherwise} \end{array}\right\} & \text{if } \delta = (-1,1) \\ \left.\begin{array}{rr} r_{21}\mu_2 & \text{if } i_1 < n_1 + 2 \text{ and } (i_1,j_1) \neq (n_1,n_2+2) \\ 0 & \text{otherwise} \end{array}\right\} & \text{if } \delta = (1,-1) \\ 0 \quad \text{otherwise} & \end{cases} \tag{6}$$

$$q_{(i_1,j_1),(-1)} = \begin{cases} r_{21}\mu_2 & \text{if } (i,j) = (n_1, n_2+2) \\ r_{12}\mu_1 & \text{if } (i,j) = (n_1+2, n_2) \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

and

$$q_{-1,s} = 0 \tag{8}$$

For $n_1 = 1$ and $n_2 = 2$, the resulting Markov chain is shown in Figure 23.

Figure 24 shows the effect of varying the parameters of the above Markov model. Base parameters of $\Lambda_1 = 4$, $\Lambda_2 = 5$, $n_1 = 3$, $n_2 = 2$, $\mu_1 = 10$, $\mu_2 = 8$, $r_{12} = 0.25$ and $r_{21} = 0.15$ were used. similar behaviour to Figure 20 can be seen.

5.4 Complete Two Node Network

Consider the queueing network shown in Figure 25. This shows two $M/M/1$ queues, with n_i queueing capacity at at each service station and service rates μ_i. Λ_i is the external arrival rates to each service station. All routing possibilities are possible, where the routing probability from node i to node j is denoted by r_{ij}.

Let this system be denoted by Ω with parameter set (Λ_1, Λ_2, μ_1, μ_2, n_1, n_2, r_{11}, r_{12}, r_{21}, r_{22}).

- State space:

$$S = \{(i,j) \in \mathbb{N}^{(n_1+2\times n_2+2)} \mid 0 \leq i + j \leq n_1 + n_2 + 2\} \cup \{(-1),(-2),(-3)\}$$

where i denotes the number of individuals:

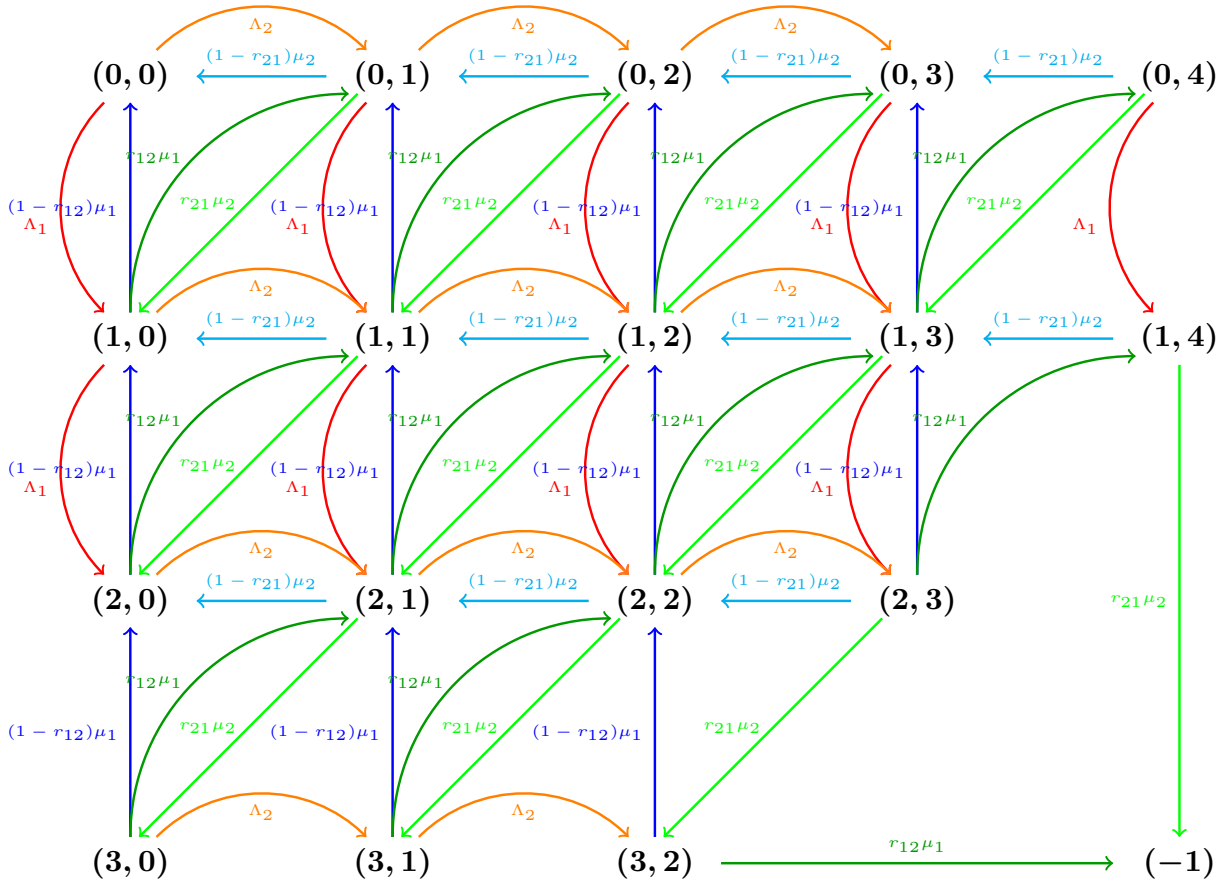
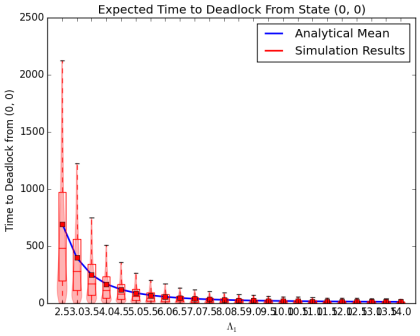
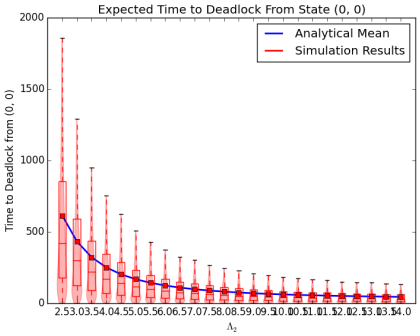


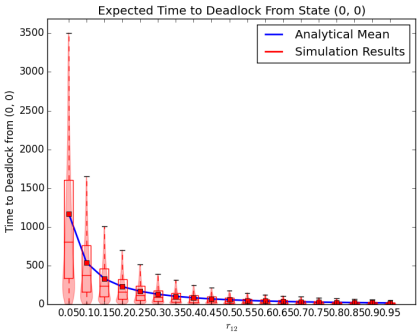
Figure 23: Diagrammatic representation of the Markov chain for Ω_2 with $n_1 = 1$ and $n_2 = 2$.



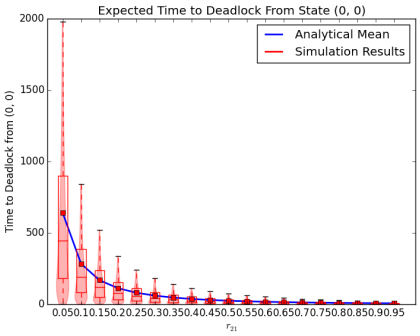
(a) Varying Λ_1



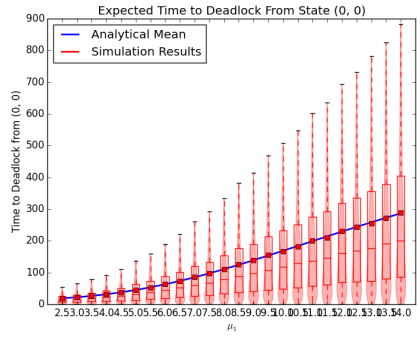
(b) Varying Λ_2



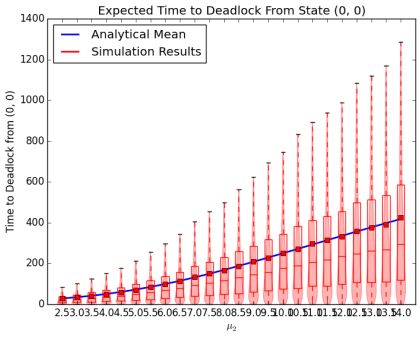
(c) Varying r_{12}



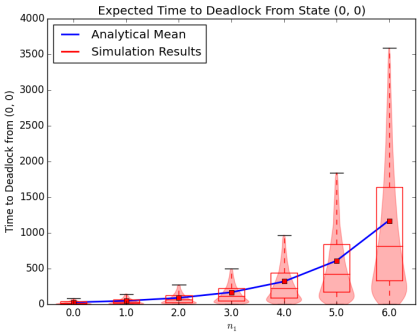
(d) Varying r_{21}



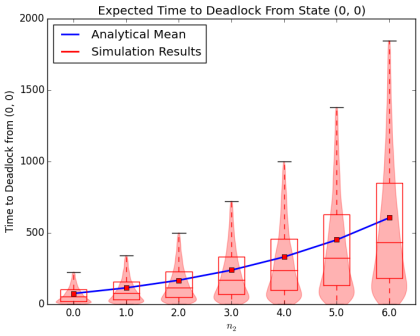
(e) Varying μ_1



(f) Varying μ_2



(g) Varying n_1



(h) Varying n_2

Figure 24: Time to deadlock in Ω_2, analytical & simulation results (10,000 iterations).

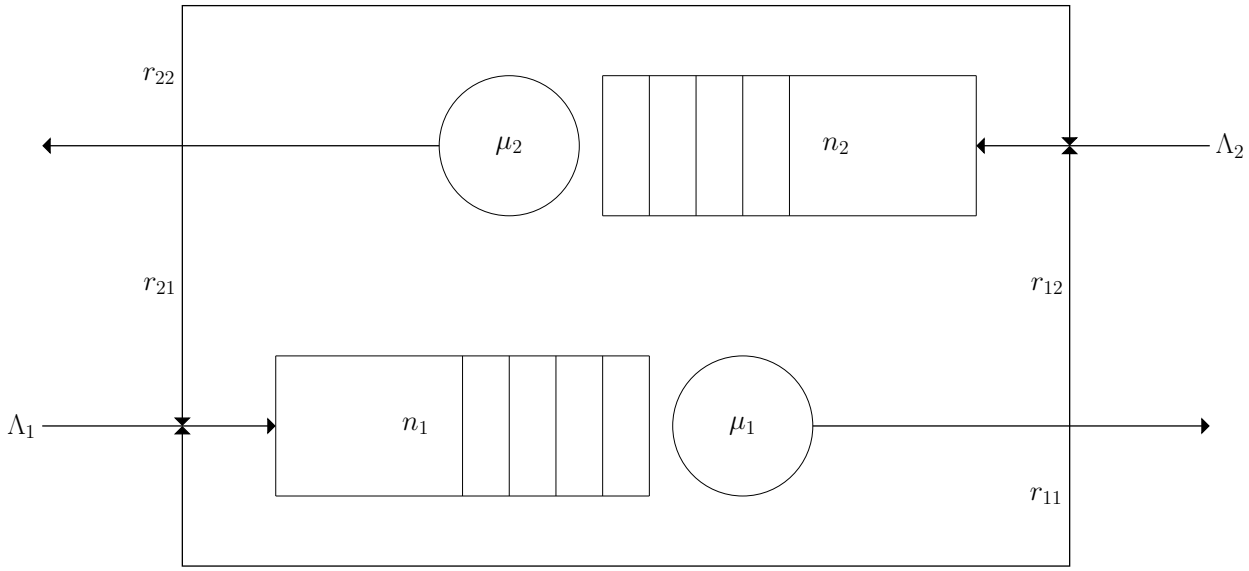


Figure 25: A complete two node queueing network.

- In service or waiting at the first node.
- Occupying a server but having finished service at the second node waiting to join the first.

where j denotes the number of individuals:

- In service or waiting at the second node.
- Occupying a server but having finished service at the first node waiting to join the second.

and the state (-3) denotes the deadlocked state cause by both nodes; (-1) denotes the deadlocked state caused by the first node only; and (-2) denotes the deadlocked state caused by the second node only. (Recall the different configurations of deadlock in Figure 6)

If we define $\delta = (i_2, j_2) - (i_1, j_1)$ for all $(i_k, j_k) \in S$, then the transitions are given by:

$$q_{(i_1,j_1),(i_2,j_2)} = \begin{cases} \left.\begin{array}{rl} \Lambda_1 & \text{if } i_1 < n_1 + 1 \\ 0 & \text{otherwise} \end{array}\right\} & \text{if } \delta = (1,0) \\ \left.\begin{array}{rl} \Lambda_2 & \text{if } j_1 < n_2 + 1 \\ 0 & \text{otherwise} \end{array}\right\} & \text{if } \delta = (0,1) \\ \left.\begin{array}{rl} (1 - r_{11} - r_{12})\mu_1 & \text{if } j_1 < n_2 + 2 \\ 0 & \text{otherwise} \end{array}\right\} & \text{if } \delta = (-1,0) \\ \left.\begin{array}{rl} (1 - r_{21} - r_{22})\mu_2 & \text{if } i_1 < n_1 + 2 \\ 0 & \text{otherwise} \end{array}\right\} & \text{if } \delta = (0,-1) \\ \left.\begin{array}{rl} r_{12}\mu_1 & \text{if } j_1 < n_2 + 2 \text{ and } (i_1,j_1) \neq (n_1+2,n_2) \\ 0 & \text{otherwise} \end{array}\right\} & \text{if } \delta = (-1,1) \\ \left.\begin{array}{rl} r_{21}\mu_2 & \text{if } i_1 < n_1 + 2 \text{ and } (i_1,j_1) \neq (n_1,n_2+2) \\ 0 & \text{otherwise} \end{array}\right\} & \text{if } \delta = (1,-1) \\ 0 \quad \text{otherwise} & \end{cases} \tag{9}$$

$$q_{(i_1,j_1),(-1)} = \begin{cases} r_{11}\mu_1 & \text{if } i > n_1 \text{ and } j < n_2 + 2 \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

$$q_{(i_1,j_1),(-2)} = \begin{cases} r_{22}\mu_2 & \text{if } j > n_2 \text{ and } i < n_1 + 2 \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

$$q_{(i_1,j_1),(-3)} = \begin{cases} r_{21}\mu_2 & \text{if } (i,j) = (n_1, n_2+2) \\ r_{12}\mu_1 & \text{if } (i,j) = (n_1+2, n_2) \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

and

$$q_{-1,s} = 0 \tag{13}$$

$$q_{-2,s} = 0 \tag{14}$$

$$q_{-3,s} = 0 \tag{15}$$

Note that the there are only two differences between this formulation and the formulation given in Subsection 5.3: the probabilities of leaving nodes 1 and 2 are now $(1 - r_{11} - r_{12})\mu_1$ and $(1 - r_{21} - r_{22})\mu_2$; and there are now two more ways to reach deadlock, Equation 10 and Equation 11.

For $n_1 = 1$ and $n_2 = 2$, the resulting Markov chain is shown in Figure 26.

Figure 27 shows the effect of varying the parameters of the above Markov model. Base parameters of $\Lambda_1 = 4$, $\Lambda_2 = 5$, $n_1 = 3$, $n_2 = 2$, $\mu_1 = 10$, $\mu_2 = 8$, $r_{11} = 0.1$, $r_{12} = 0.25$, $r_{21} = 0.15$ and $r_{22} = 0.1$ were used.

The heatmaps in Figure 28 illustrate how varying the two transition probabilities out of each node affects

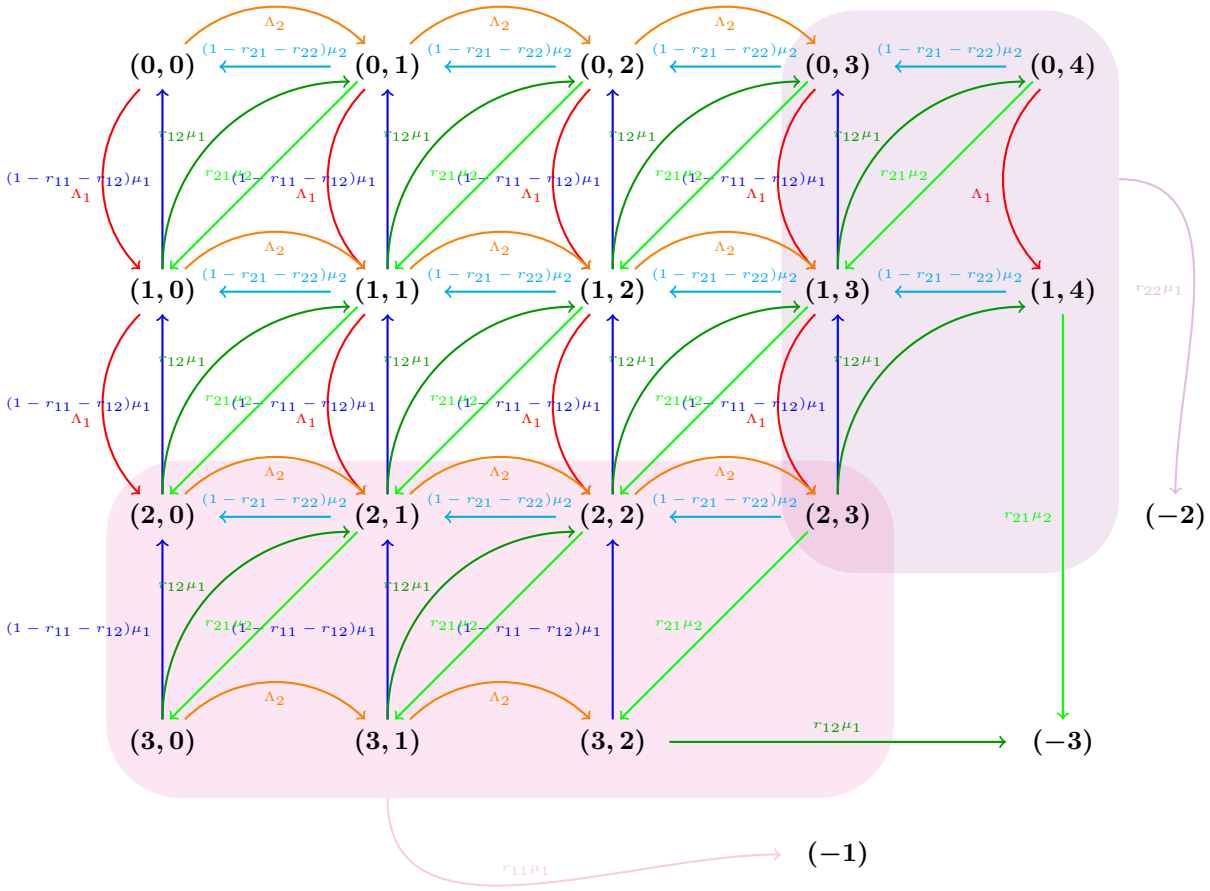
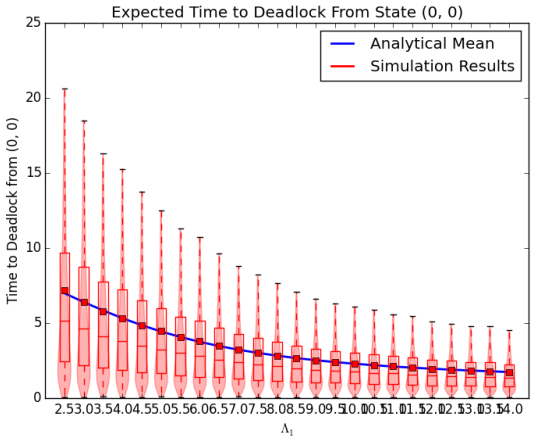
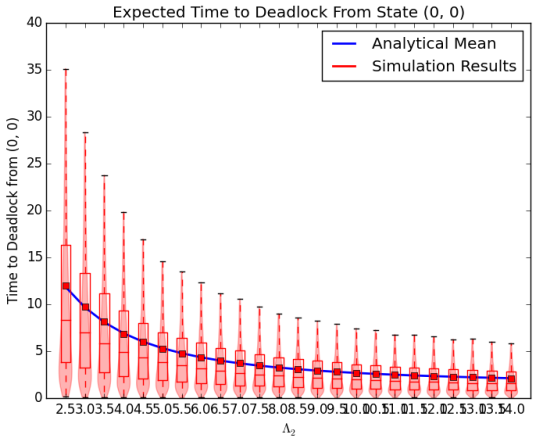


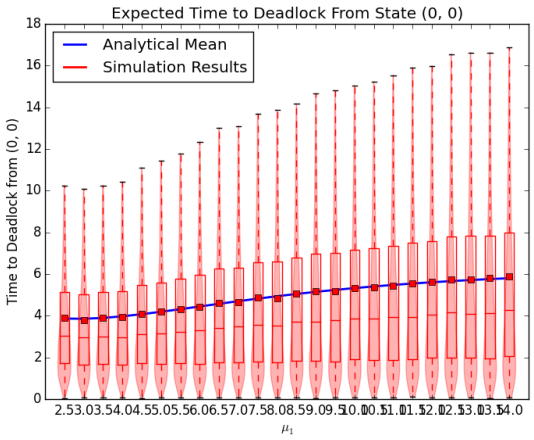
Figure 26: Diagrammatic representation of the Markov chain for Ω with $n_1 = 1$ and $n_2 = 2$.



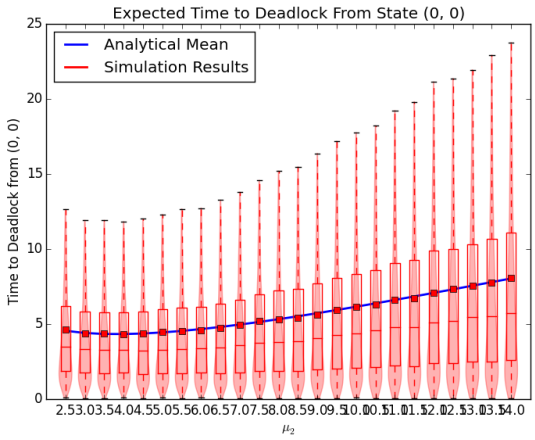
(a) Varying Λ_1



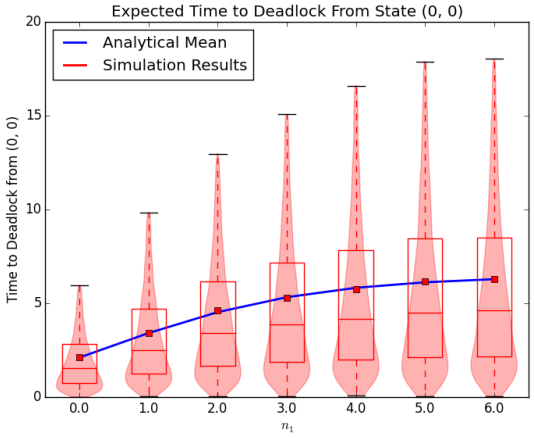
(b) Varying Λ_2



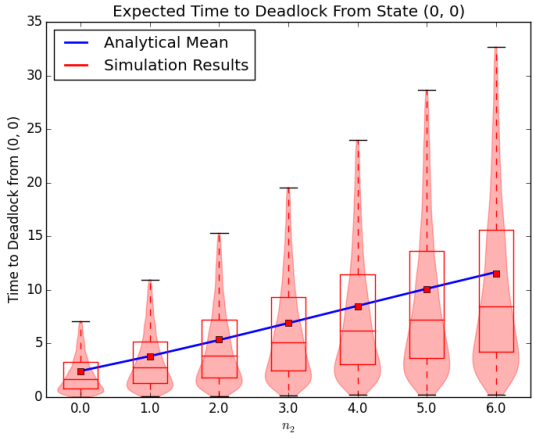
(c) Varying μ_1



(d) Varying μ_2



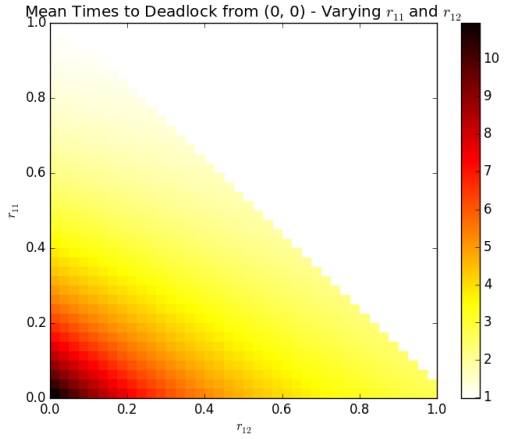
(e) Varying n_1



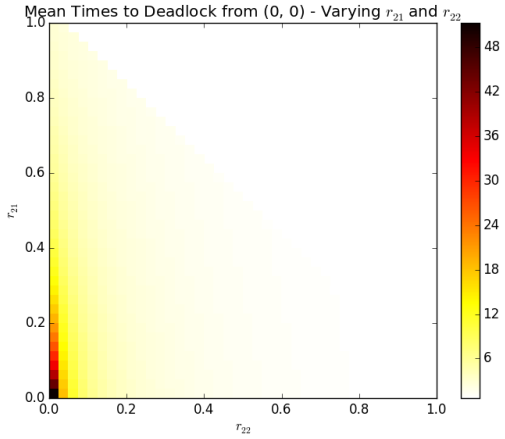
(f) Varying n_2

Figure 27: Time to deadlock in Ω, analytical & simulation results (10,000 iterations).

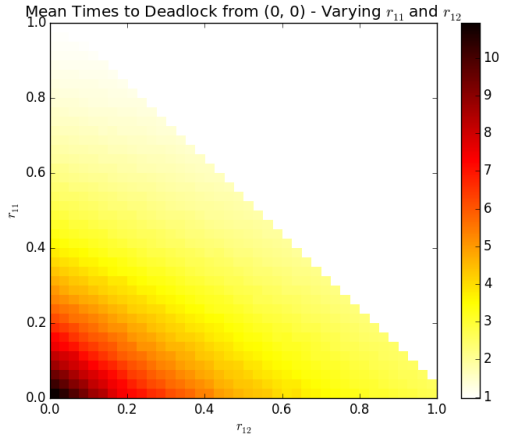
the time to deadlock. Note the shape of the heatmap, this is due to the restriction that $r_{11} + r_{12} \leq 1$ and $r_{21} + r_{22} \leq 1$. We can see for both nodes it is the rejoining probability (r_{11} and r_{22}) that has the most drastic effect on time to deadlock. This effect is greater for Node 2, the node that has the smaller queueing capacity.



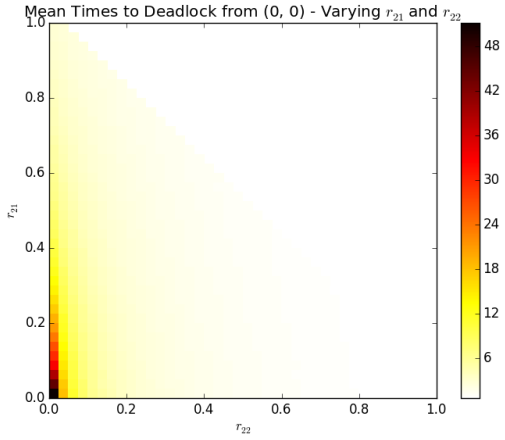
(a) Analytical: Varying transition probabilities at Node 1



(b) Analytical: Varying transition probabilities at Node 2



(c) Simulation: Varying transition probabilities at Node 1



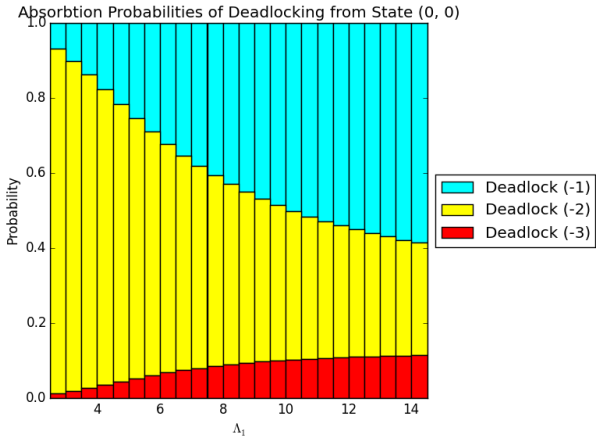
(d) Simulation: Varying transition probabilities at Node 2

Figure 28: Analytical & Simulation Results of Times to Deadlock, varying Transition Probabilities (10,000 iterations)

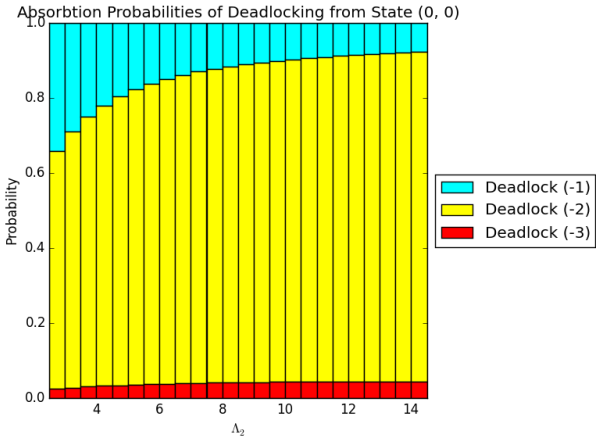
Times to deadlock in this case means the time to the first instance of deadlock. There is however three different deadlocked states which the queueing network can fall into, (-1), (-2) and (-3).

Figure 29 shows how varying the parameters of the queueing network affects the absorption probabilities.

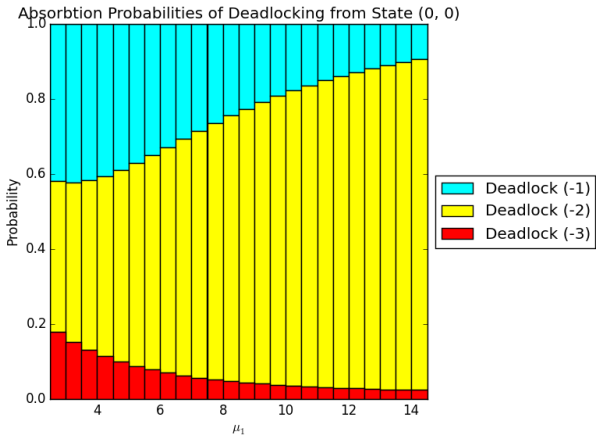
Figure 27e shows that increasing queueing capacity is no longer nearly linear. Increasing n_1 causes a longer time to deadlock up to a point, where increasing n_1 any more does not affect the time to deadlock. This is due to the fact that there are other ways to get to deadlock, and we are only interested in the first instance of any of these deadlocks. Getting to deadlocked state (-2) is not affected by n_1, and so once n_1 goes over



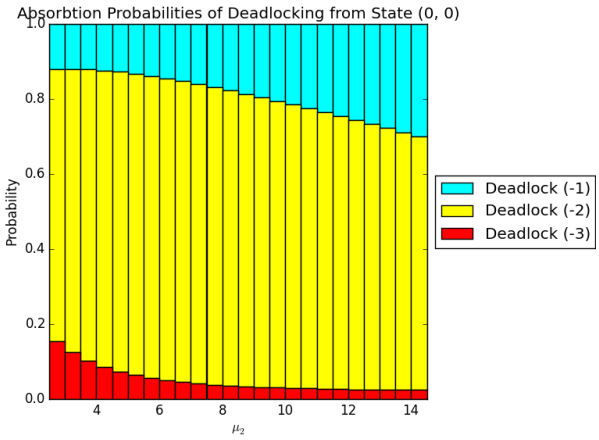
(a) Varying Λ_1



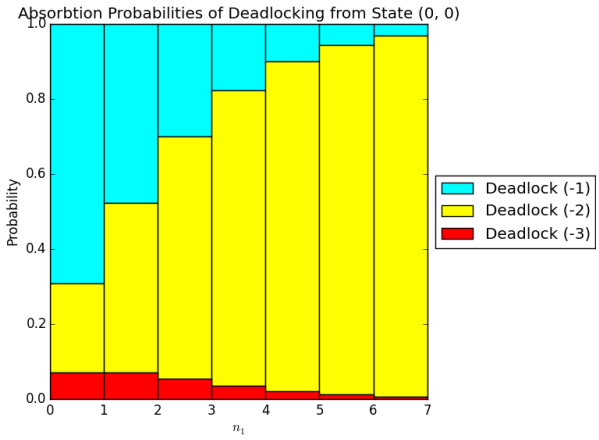
(b) Varying Λ_2



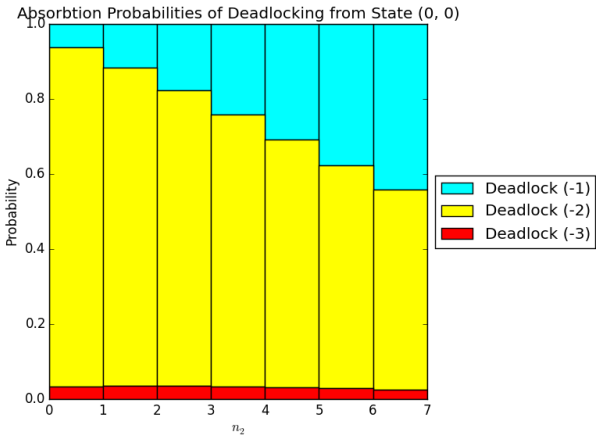
(c) Varying μ_1



(d) Varying μ_2



(e) Varying n_1



(f) Varying n_2

Figure 29: Probabilities of reaching each deadlocked state.

a certain threshold the system always reaches (-2) before any other deadlocked state, and so the time to deadlock is unaffected by increasing n_1 any further.

This behaviour is exhibited as n_2 increases too, however this threshold will take longer to reach as the base parameter for n_1 is larger than the base parameter for n_2.

This behaviour is highlighted in the heatmap shown in Figure 30, where the analytical mean time to deadlock is shown with combinations of n_1 and n_2. For each value of n_2, once a certain threshold of n_1 is reached no more change in time to deadlock occurs. Similarly for each value of n_1, once a certain threshold o n_2 is reached no more change in time to deadlock occurs. The behaviour is unsymmetrical due to the other parameters associated with Node 1 and Node 2 being unsymmetrical.

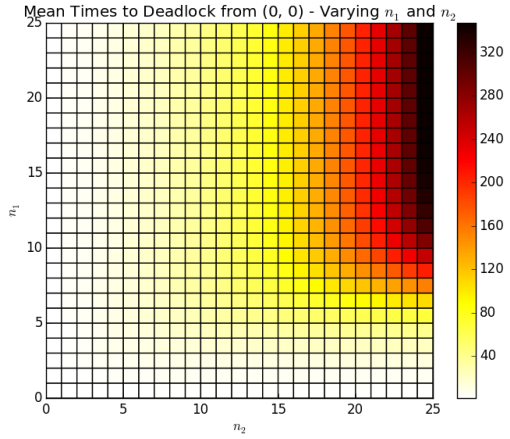
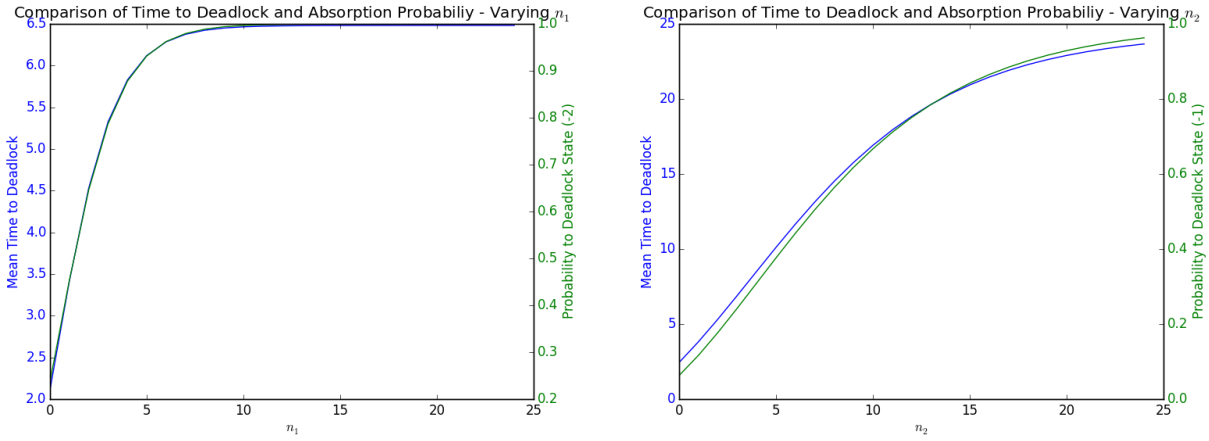


Figure 30: Deadlocking behaviour, varying n_1 and n_2



(a) Varying n_1

(b) Varying n_2

Figure 31: Comparing time to deadlock and absorption probabilities as queue capacities increase.

Figure 31 compares the deadlocking behaviour as n_1 and n_2 increase to absorption probabilities. Time to deadlock while varying n_1 is compared to the probability of absorption to state (-2), as n_1 plays no part in

getting to state (-2). Similarly time to deadlock while varying n_2 is compared to the probability of absorption to state (-1). Here it is shown that once it is almost certain that deadlock will be caused by the other node only, then increasing the queueing capacity at this node does not affect time to deadlock.

5.5 Multi-Server Networks

It is possible to model the networks discussed in Section 5.2 and Section 5.3 where each node has multiple parallel servers. These models are discussed in the next subsections.

5.5.1 Multi-Server: One Node Network

Consider the one node network with feedback loop discussed in Subsection 5.2, now with c parallel servers. This multi server Ω_1 system is shown in Figure 32.

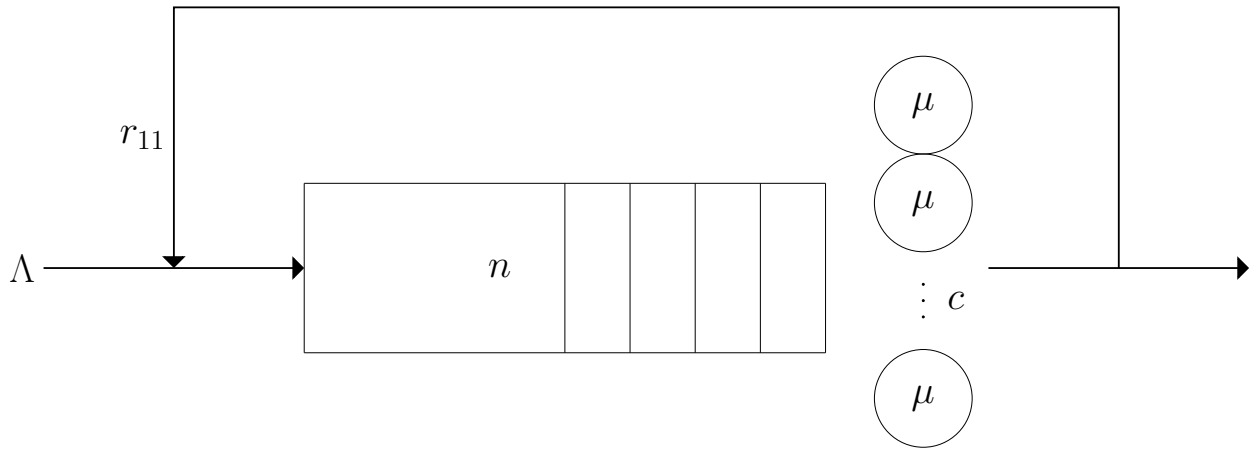


Figure 32: A one node multi-server queueing network.

State space:

$$S = \{i \in \mathbb{N} \mid 0 \leq i \leq n + 2c\}$$

where i denotes the number of individuals at the node plus the number of individuals blocked at that node. For example, $i = n + c + 2$ denotes a full system, $n + c$ individuals in the node, and 2 of those individuals are also blocked. The state $i = n + 2c$ denotes the deadlocked state.

If we define $\delta = i_2 - i_1$ for all $i_k \in S$, then the transitions are given by:

$$q_{i_1,i_2} = \left\{ \begin{array}{rl} \Lambda & \text{if } \delta = 1 \\ (1 - r_{11})\mu \min(i, c) & \text{if } \delta = -1 \\ 0 & \text{otherwise} \end{array} \right\} \quad \text{if } i_1 < n + c \tag{16}$$

$$q_{i_1,i_2} = \left\{ \begin{array}{rr} (c-b)r_{11}\mu & \text{if } \delta = 1 \\ (1-r_{11})(c-b)\mu & \text{if } \delta = -b-1 \\ 0 & \text{otherwise} \end{array} \right\} \quad \text{if } i_1 = n+c+b \quad \forall \quad 0 \leq b \leq c \tag{17}$$

where b denotes the number of blocked customers. The Markov chain is shown in Figure 33.

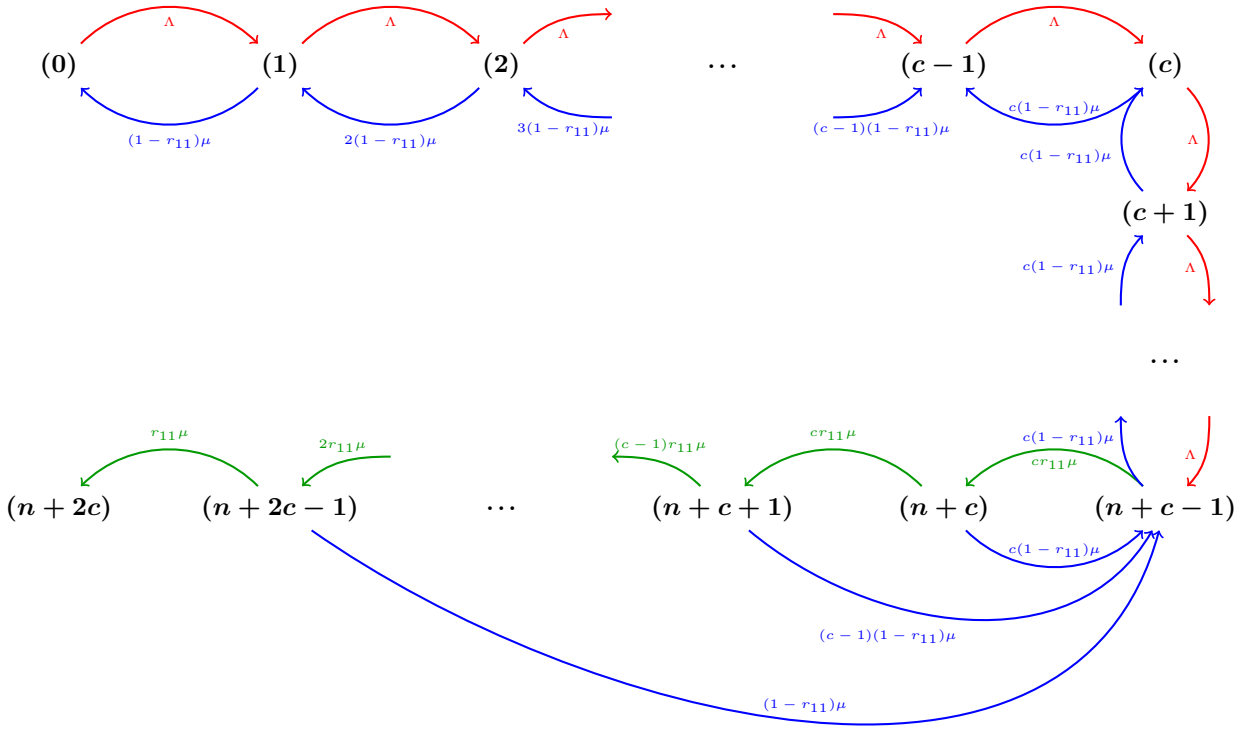
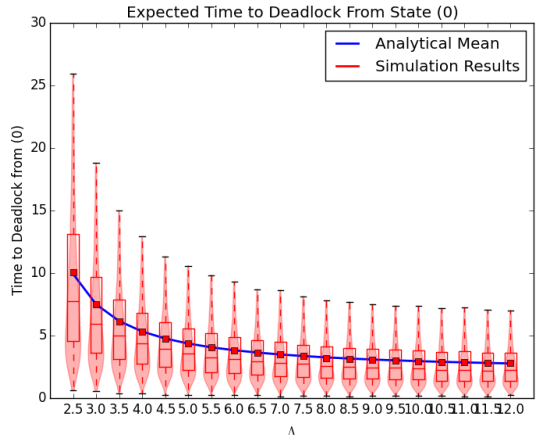


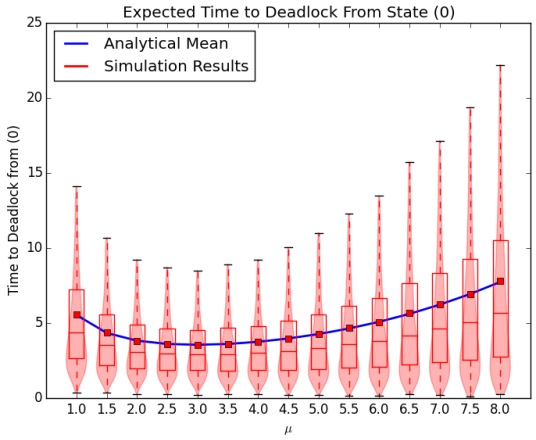
Figure 33: Diagrammatic representation of the Markov chain for a multi server Ω_1 system.

Increasing the amount of servers has a similar effect to increasing the queueing capacity, there are now more transient states to go through before reaching the deadlocked state. Varying the amount of servers has a greater effect on the time to deadlock however, as any states in which customers are blocked ($i \in [n+c+1, n+2c]$) can jump back to state $i = n+c-1$ simply with a service and an exit. Increasing the amount of servers also increases the rate at which i are reduced for most states, but not the rates at which i is increased.

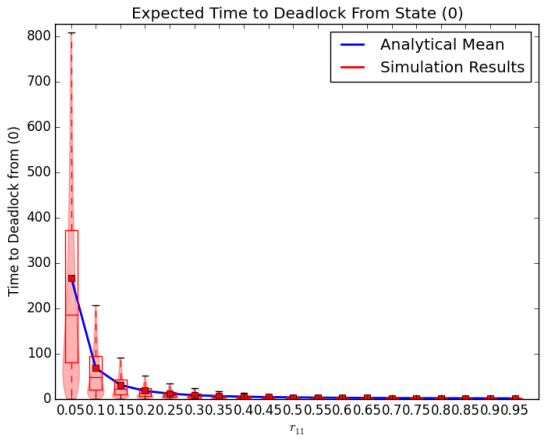
Figure 34 shows the effect of varying the parameters of the above Markov model. Base parameters of $\Lambda = 6$, $n = 3$, $\mu = 2$, $r_{11} = 0.5$ and $c = 2$ were used.



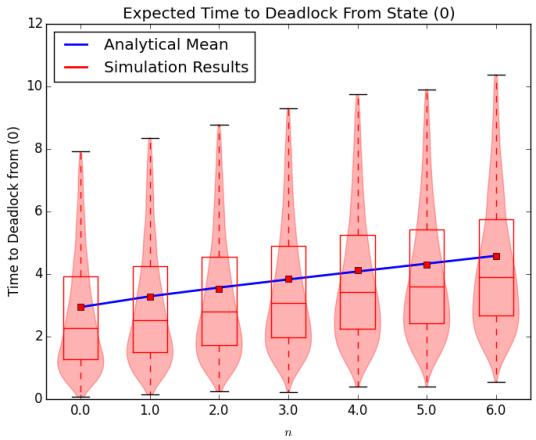
(a) Varying Λ



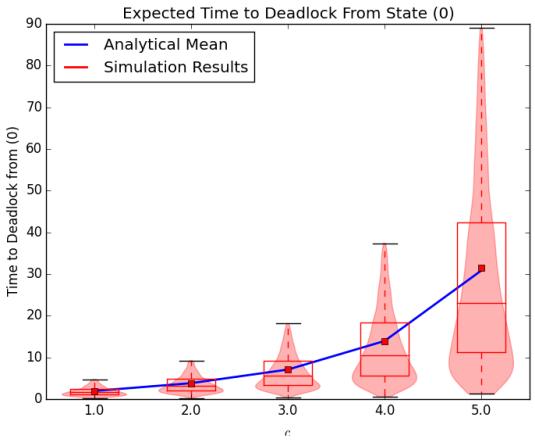
(b) Varying μ



(c) Varying r_{11}



(d) Varying n



(e) Varying c

Figure 34: Time to deadlock in multi-server Ω_1, analytical & simulation results (10,000 iterations).

5.5.2 Multi-Server: Two Node Network without Self Loops

Consider the two node network with feedback loops discussed in Subsection 5.3, now with c_1 parallel servers at the first node, and c_2 parallel servers at the second node. This is shown in Figure 35

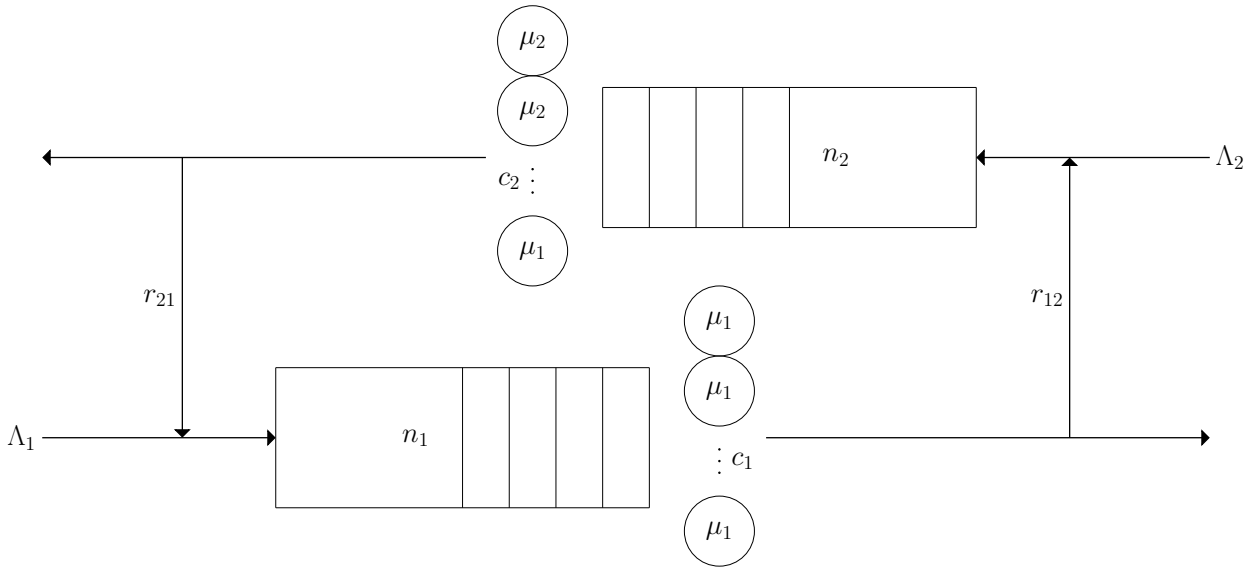


Figure 35: A two node multi-server queueing network.

State space:

$$S = \{(i,j) \in \mathbb{N}^{(n_1+c_1+c_2)\times(n_2+c_2+c_1)} \mid i \leq n_1 + c_1 + j,\ j \leq n_2 + c_2 + i\}$$

where i denotes the number of individuals at Node 1 plus the number of individuals blocked waiting to enter Node 1, and j denotes the number of individuals at Node 2 plus the number of individuals blocked waiting to enter Node 2. For example, $(i,j) = (n_1 + c_1 + 2, n_2 + c_2 + 1)$ denotes a full system, $n_1 + c_1$ individuals at Node 1, two of whom are blocked waiting to enter Node 2; $n_2 + c_2$ individuals at Node 2, one of whom is blocked waiting to enter Node 1. The state $(i,j) = (n_1 + c_1 + c_2, n_2 + c_2 + c_1)$ denotes the deadlocked state.

The Markov chain is shown in Figure 36.

Define $\delta = (i_2, j_2) - (i_1, j_1)$, $b_1 = \max(0, i_1 - n_1 - c_1)$, $b_2 = \max(0, i_2 - n_2 - c_2)$, $s_1 = \min(i_1, c_1) - b_2$ and $s_2 = \min(i_2, c_2) - b_1$,then the transitions $q_{(i_1,j_1),(i_2,j_2)}$ are given by Table 1.

The values b_1 and b_2 correspond to the number of people blocked to Node 1 and Node 2 respectively. The values s_1 and s_2 correspond to the amount of people currently in service at Node 1 and Node 2 respectively.

This formulation of the Markov chain makes use of the following proposition:

Proposition 5. *In the two node queueing network described, if there are b_1 customers blocked to Node 1, and b_2 customers blocked to Node 2, where $b_1, b_2 > 0$, then:*

i. A customer finishing service and leaving the system at Node 1 yields a difference in state of $\delta = (-\min(b_1 + 1, b_2 + 1), -\min(b_1, b_2 + 1))$.

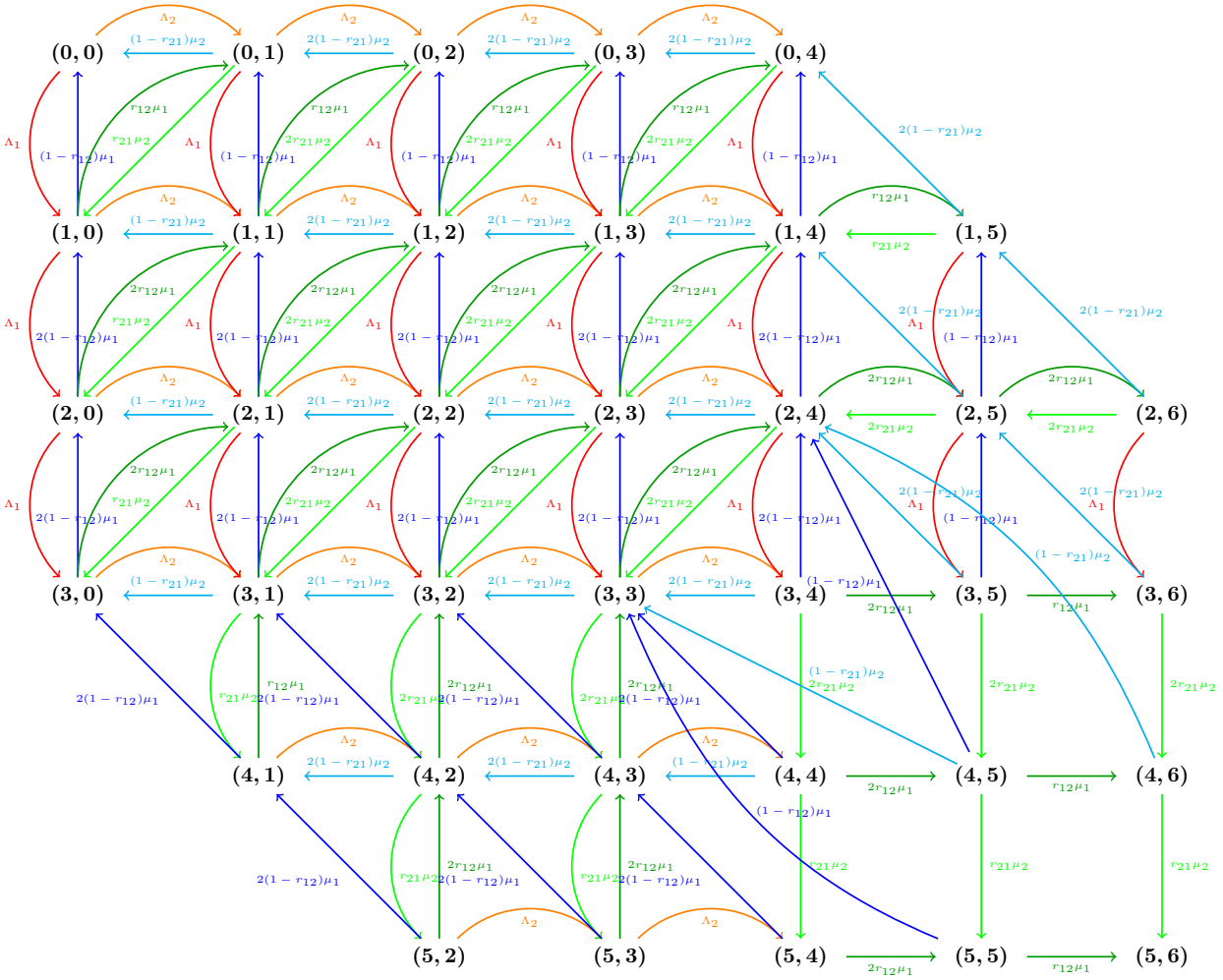


Figure 36: Diagrammatic representation of the Markov chain for a multi server Ω_2 system with $n_1 = 1$, $n_2 = c_1 = c_2 = 2$. The deadlocked state is $(5, 6)$.

	$j_1 < n_2 + c_2$	$j_1 = n_2 + c_2$	$j_1 > n_2 + c_2$
$i_1 < n_1 + c_1$	Λ_1 if $\delta = (1,0)$ Λ_2 if $\delta = (0,1)$ $r_{12}s_1\mu_1$ if $\delta = (-1,1)$ $r_{21}s_2\mu_2$ if $\delta = (1,-1)$ $(1-r_{12})s_1\mu_1$ if $\delta = (-1,0)$ $(1-r_{21})s_2\mu_2$ if $\delta = (0,-1)$	Λ_1 if $\delta = (1,0)$ $r_{12}s_1\mu_1$ if $\delta = (0,1)$ $r_{21}s_2\mu_2$ if $\delta = (1,-1)$ $(1-r_{12})s_1\mu_1$ if $\delta = (-1,0)$ $(1-r_{21})s_2\mu_2$ if $\delta = (0,-1)$	Λ_1 if $\delta = (1,0)$ $r_{12}s_1\mu_1$ if $\delta = (0,1)$ $r_{21}s_2\mu_2$ if $\delta = (0,-1)$ $(1-r_{12})s_1\mu_1$ if $\delta = (-1,0)$ $(1-r_{21})s_2\mu_2$ if $\delta = (-1,-1)$
$i_1 = n_1 + c_1$	Λ_2 if $\delta = (0,1)$ $r_{12}s_1\mu_1$ if $\delta = (-1,1)$ $r_{21}s_2\mu_2$ if $\delta = (1,0)$ $(1-r_{12})s_1\mu_1$ if $\delta = (-1,0)$ $(1-r_{21})s_2\mu_2$ if $\delta = (0,-1)$	$r_{12}s_1\mu_1$ if $\delta = (0,1)$ $r_{21}s_2\mu_2$ if $\delta = (1,0)$ $(1-r_{12})s_1\mu_1$ if $\delta = (-1,0)$ $(1-r_{21})s_2\mu_2$ if $\delta = (0,-1)$	$r_{12}s_1\mu_1$ if $\delta = (0,1)$ $r_{21}s_2\mu_2$ if $\delta = (1,0)$ $(1-r_{12})s_1\mu_1$ if $\delta = (-1,0)$ $(1-r_{21})s_2\mu_2$ if $\delta = (-1,-1)$
$i_1 > n_1 + c_1$	Λ_2 if $\delta = (0,1)$ $r_{12}s_1\mu_1$ if $\delta = (-1,0)$ $r_{21}s_2\mu_2$ if $\delta = (1,0)$ $(1-r_{12})s_1\mu_1$ if $\delta = (-1,-1)$ $(1-r_{21})s_2\mu_2$ if $\delta = (0,-1)$	$r_{12}s_1\mu_1$ if $\delta = (0,1)$ $r_{21}s_2\mu_2$ if $\delta = (1,0)$ $(1-r_{12})s_1\mu_1$ if $\delta = (-1,-1)$ $(1-r_{21})s_2\mu_2$ if $\delta = (0,-1)$	$r_{12}s_1\mu_1$ if $\delta = (0,1)$ $r_{21}s_2\mu_2$ if $\delta = (1,0)$ $(1-r_{12})s_1\mu_1$ if $\delta = (-\min(b_1+1, b_2+1), -\min(b_1, b_2+1))$ $(1-r_{21})s_2\mu_2$ if $\delta = (-\min(b_1+1, b_2), -\min(b_1+1, b_2+1))$

Table 1: Table of transitions for a multi server two node network.

ii. A customer finishing service and leaving the system at Node 2 yields a difference in state of $\delta = (-\min(b_1+1, b_2), -\min(b_1+1, b_2+1))$.

Proof. Each new blocked individual contributes $(1,1)$ to the state space. A blocked customer is counted as existing in the Node they are occupying, but also has a blocked part counted at the other node.

Now an unblocking where a customer transitions from being blocked at Node 1 to being at Node 2 yields $\delta = (-1, 0)$; the customer's blocked part is no longer blocked, while they themselves transition to Node 2. This type of unblocking leaves space at Node 1 for more potential unblockings.

Similarly an unblocking where a customer transitions from being blocked at Node 2 to being at Node 1 yields $\delta = (0, -1)$; the customer's blocked part is no longer blocked, while they themselves transition to Node 1. This type of unblocking leaves space at Node 2 for more potential unblockings.

Consider a customer finishing service and exiting the system at Node 1. Breaking down the overall unblocking process into subprocesses, we can break down the overall difference δ into sub-differences δ_1, δ_2, δ_3, etc, where $\delta = \sum_{i \in I} \delta_i$.

The set I is the number of sub-differences that happen until no more unblockings can happen.

An unblockage at Node 1 always occurs after space at Node 2 is created, by an unblockage at Node 2. Therefore the maximium number of times a customer can be unblocked from Node 1 is either until there are no more blocked customers at Node 1, or no more unblockages at Node 2, as there are no more customers at Node 2 to unblock. Therefore $\min(b_1, b_2)$ unblockages of this kind.

An unblockage at Node 2 occurs after an unblockage at Node 1 if there are enough blocked customers to do so. An unblockage at Node 2 causes an unblockage at Node 1, and so directly proceeds an unblockage at Node 1. If $b_2 > b_1$ there are b_1 unblockages at Node 1, and so b_1 unblockages at Node 2. If $b_1 > b_2$ there are b_2 unblockages at Node 1, and so $b_2 + 1$ unblockages at Node 2. Therefore $\min(b_1, b_2 + 1)$ unblockages of this kind.

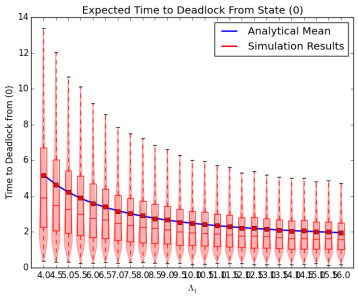
Now:

$$\begin{aligned}
\delta &= \sum_{i \in I} \delta_i \\
&= (-1, 0) + (0, -1) + (-1, 0) + (0, -1) + \dots \\
&= (-1, 0) + \min(b_1, b_2 + 1) \times (0, -1) + \min(b_2, b_1) \times (-1, 0) \\
&= (-\min(b_1 + 1, b_2 + 1), -\min(b_1, b_2 + 1))
\end{aligned}$$

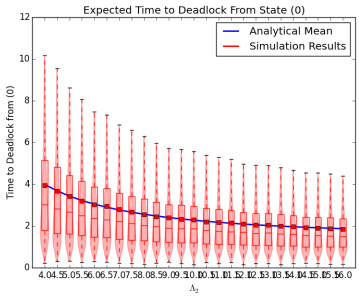
A similar argument yields $\delta = (-\min(b_1 + 1, b_2), -\min(b_1 + 1, b_2 + 1))$ for the situation where a customer finishes service at Node 2 and exits the system.

□

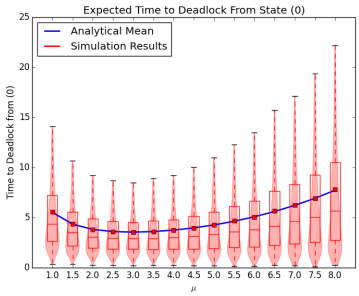
Figure 37 shows the effect of varying the parameters of the above Markov model. Base parameters of $\Lambda_1 = 9$, $\Lambda_2 = 7.5$, $n_1 = 2$, $n_2 = 1$, $\mu_1 = 5.5$, $\mu_2 = 6.5$, $r_{12} = 0.7$, $r_{21} = 0.6$, $c_1 = 2$ and $c_2 = 2$ were used.



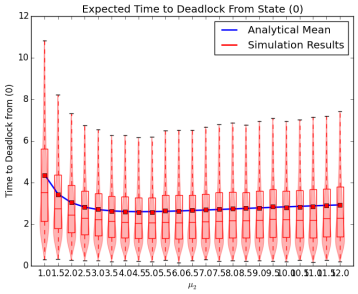
(a) Varying Λ_1



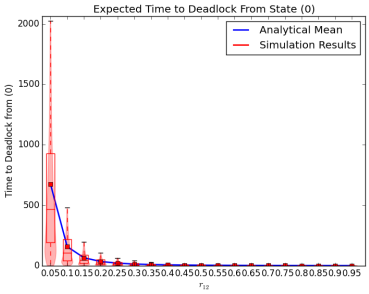
(b) Varying $\Lambda_@$



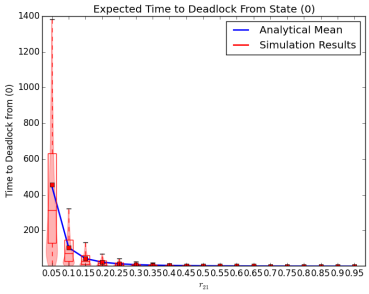
(c) Varying μ_1



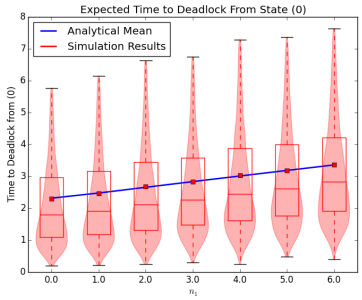
(d) Varying μ_2



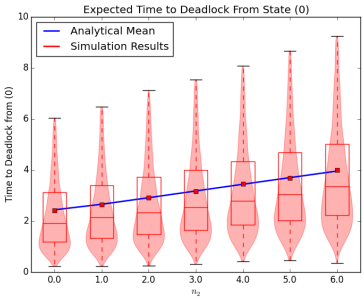
(e) Varying r_{12}



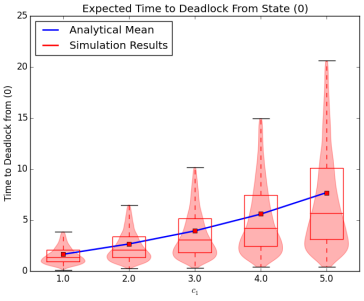
(f) Varying r_{21}



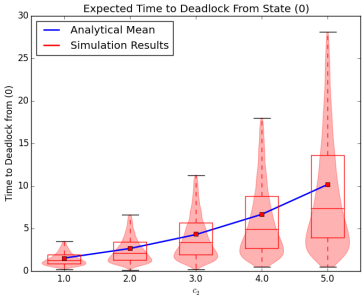
(g) Varying n_1



(h) Varying n_2



(i) Varying c_1



(j) Varying c_2

Figure 37: Time to deadlock in multi-server Ω_2, analytical & simulation results (10,000 iterations).

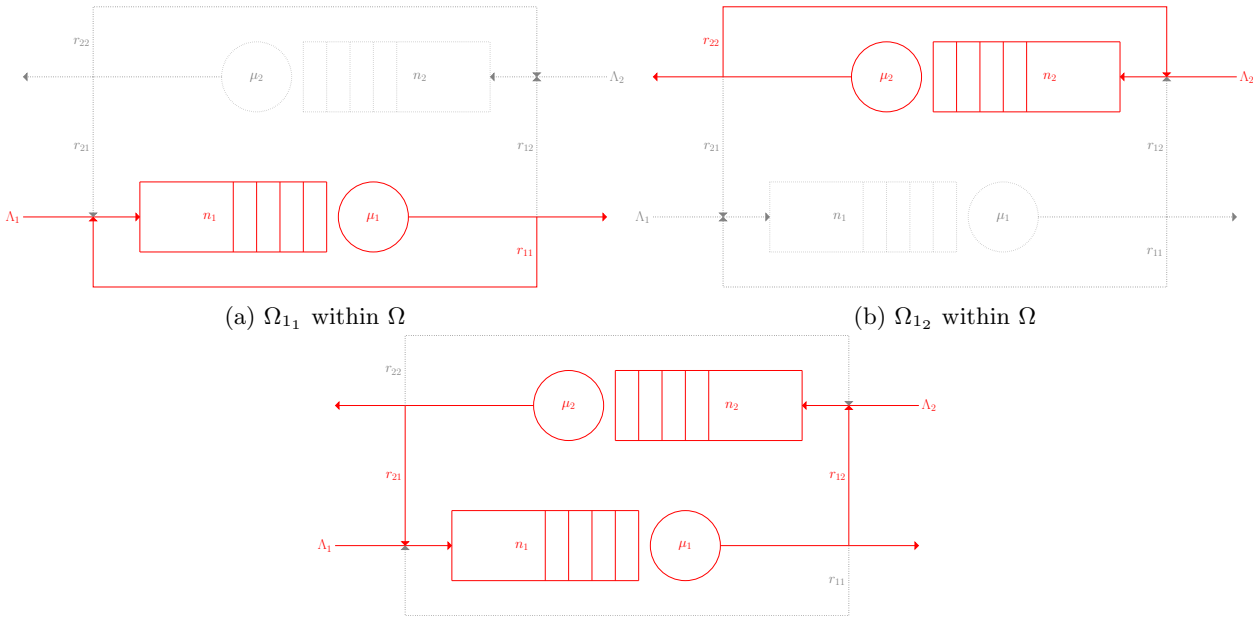
6 A Bound on the Mean Time to Deadlock

This section will derive a bound for the mean time to deadlock of the Ω system. In order to do this, six deadlocking queueing networks are defined as follows:

- Define $\Omega^{\star}_{1_1}$ as the 1 node queueing network described in Subsection 5.2 with the parameter set (Λ_1, μ_1, n_1, r_{11}). Let its mean time to deadlock be denoted by $\omega^{\star}_{1_1}$.
- Define $\Omega^{\star\star}_{1_1}$ as the 1 node queueing network described in Subsection 5.2 with the parameter set (Λ_1, m_1, n_1, r_{11}). Let its mean time to deadlock be denoted by $\omega^{\star\star}_{1_1}$.
- Define $\Omega^{\star}_{1_2}$ as the 1 node queueing network described in Subsection 5.2 with the parameter set (Λ_2, μ_2, n_2, r_{22}). Let its mean time to deadlock be denoted by $\omega^{\star}_{1_2}$.
- Define $\Omega^{\star\star}_{1_2}$ as the 1 node queueing network described in Subsection 5.2 with the parameter set (Λ_2, m_2, n_2, r_{22}). Let its mean time to deadlock be denoted by $\omega^{\star\star}_{1_2}$.
- Define Ω_2 as the 2 node queueing network described in Subsection 5.3 with the parameter set (Λ_1, Λ_2, μ_1, μ_2, n_1, n_2, r_{12}, r_{21}). Let its mean time to deadlock be denoted by ω_2.
- Define Ω as the 2 node queueing network described in Subsection 5.4 with the parameter set (Λ_1, Λ_2, μ_1, μ_2, n_1, n_2, r_{11}, r_{12}, r_{21}, r_{22}). Let its mean time to deadlock be denoted by ω.

where $m_1 = \frac{\mu_2}{3+2\frac{\mu_2}{\mu_1}+\frac{\mu_1}{\mu_2}}$, and $m_2 = \frac{\mu_1}{3+2\frac{\mu_1}{\mu_2}+\frac{\mu_2}{\mu_1}}$.

Figure 38 shows how Ω contains, and is made up by, Ω_{1_1}, Ω_{1_2} and Ω_2.



(a) Ω_{1_1} within Ω

(b) Ω_{1_2} within Ω

(c) Ω_2 within Ω

Figure 38: Decomposition of Ω into Ω_{1_1}, Ω_{1_2} and Ω_2

Defining $\omega_{1_j} = \max(\omega^{\star}_{1_j}, \omega^{\star\star}_{1_j})$ we get the following bound:

Theorem 2. *For any parameter sets the following inequality holds:* $\omega \leq \min(\omega_{1_1}, \omega_{1_2}, \omega_2)$

Proof. First, define the following systems:

- Let $\widetilde{\Omega}_{1_1}$ denote the Ω_{1_1} system embedded within Ω. Let $\widetilde{\omega}_{1_1}$ denote the mean time to deadlock of $\widetilde{\Omega}_{1_1}$.
- Let $\widetilde{\Omega}_{1_2}$ denote the Ω_{1_2} system embedded within Ω. Let $\widetilde{\omega}_{1_2}$ denote the mean time to deadlock of $\widetilde{\Omega}_{1_2}$.
- Let $\widetilde{\Omega}_2$ denote the Ω_2 system embedded within Ω. Let $\widetilde{\omega}_2$ denote the mean time to deadlock of $\widetilde{\Omega}_2$.

It follows that $\widetilde{\omega}_{1_1}$ is the mean time to state (-1) in Ω, $\widetilde{\omega}_{1_2}$ is the mean time to state (-2) in Ω and $\widetilde{\omega}_2$ is the mean time to state (-3) in Ω. Therefore $\omega = \min(\widetilde{\omega}_{1_1}, \widetilde{\omega}_{1_2}, \widetilde{\omega}_2)$, as the mean time to deadlock in Ω is the expected time it takes to reach either (-1), (-2) or (-3).

This proof compares each embedded system with its respective non-embedded counterpart.

1. Consider $\widetilde{\Omega}_2$. The effective arrival rate to Node 1 in $\widetilde{\Omega}_2$ is greater than or equal to the effective arrival rate to Node 1 in Ω_2, due to the extra customers who are rejoining the queue after service. Similarly the effective arrival rate to Node 2 in $\widetilde{\Omega}_2$ is greater than or equal to the effective arrival rate to Node 2 in Ω_2. As an increase in the arrival rate causes the mean time to deadlock to decrease, we can conclude $\widetilde{\omega}_2 \leq \omega_2$
2. Consider $\widetilde{\Omega}_{1_1}$. Both the service rate and arrival rate differ in the embedded system to the non-embedded system.
   - Consider the expected effective service time, the time that $\widetilde{\Omega}_{1_1}$'s state does not change due to services or outside factors.
     - The lower bound on the effective service time is $\frac{1}{\mu_1}$, corresponding to when neither Node 1 nor Node 2 are full. Therefore the upper bound on the effective service rate is μ_1.
     - The upper bound on the effective service time corrseponds to the following cycle of events:
       * A customer, a, begins service at Node 1: $\frac{1}{\mu_1}$
       * Customer a finishes service at Node 1, but is blocked from transitioning to Node 2: $\frac{1}{\mu_2}$
       * A customer, b, finishes service at Node 2 and transitions to Node 1. Now the a moves to Node 2, and another the customer begins service at Node 1: $\frac{1}{\mu_1}$

       This cycle is repeated with probability P_{repeat}. As all rates are Markovian then whatever point in b's service a gets blocked, a's expected wait is $\frac{1}{\mu_2}$. Therefore the upper bound for the effective service time is:

$$\begin{aligned}\frac{1}{m_1} &= \frac{2}{\mu_1} + \frac{1}{\mu_2} + P_{\text{repeat}}\left(\frac{2}{\mu_1} + \frac{1}{\mu_2} + P_{\text{repeat}}\left(\frac{2}{\mu_1} + \frac{1}{\mu_2} + P_{\text{repeat}}\Big(\cdots\right.\right.\right.\\ &= \left(\frac{2}{\mu_1} + \frac{1}{\mu_2}\right) \times \left(1 + P_{\text{repeat}} + P_{\text{repeat}}^2 + P_{\text{repeat}}^3 + \dots\right)\\ &= \left(\frac{2}{\mu_1} + \frac{1}{\mu_2}\right) \times \left(\frac{1}{1 - P_{\text{repeat}}}\right)\end{aligned}$$

If S_1 is the time a spends in service, and S_2 is the time b spends in service, then $S_1 \sim \text{Exp}(\mu_1)$ and $S_2 \sim \text{Exp}(\mu_2)$. Now $P_{\text{repeat}} = P(S_1 < S_2) = \frac{\mu_1}{\mu_1+\mu_2}$.

Therefore the lower bound on the effective service rate is:

$$\begin{aligned}m_1 &= \frac{1}{\left(\frac{2}{\mu_1} + \frac{1}{\mu_2}\right)\left(\frac{1}{1-P_{\text{repeat}}}\right)}\\ &= \frac{1}{\left(\frac{2}{\mu_1} + \frac{1}{\mu_2}\right)\left(\frac{1}{1-\frac{\mu_1}{\mu_1+\mu_2}}\right)}\\ &= \frac{1}{\left(\frac{2}{\mu_1} + \frac{1}{\mu_2}\right)\left(\frac{\mu_1+\mu_2}{\mu_2}\right)}\\ &= \frac{\mu_2}{\left(\frac{2}{\mu_1} + \frac{1}{\mu_2}\right)(\mu_1+\mu_2)}\\ &= \frac{\mu_2}{3 + 2\frac{\mu_2}{\mu_1} + \frac{\mu_1}{\mu_2}}\end{aligned}$$

Now the actual effective service rate $\widetilde{\mu}_1$ for $\widetilde{\Omega}_{1_1}$ must lie in the interval (μ_1, m_1). Using Remarks 1 and 2, we can conclude that $\widetilde{\omega}_{1_1} \leq \max(\omega_{1_1}^{\star}, \omega_{1_1}^{\star\star})$ when all other parameters are fixed.

- Consider the effective arrival rate of $\widetilde{\Omega}_{1_1}$. The effective arrival rate in $\widetilde{\Omega}_{1_1}$ is greater than or equal to the effective arrival rate in an Ω_{1_1} system; this is due to the extra customers who have transitioned from the Node 2 to Node 1. An increase in the arrival rate causes the mean time to deadlock to decrease.

Combining the above, it is concluded that $\widetilde{\omega}_{1_1} \leq \omega_{1_1}$.

3. A similar argument yields $\widetilde{\omega}_{1_2} \leq \omega_{1_2}$.

Therefore

$$\min(\widetilde{\omega}_{1_1}, \widetilde{\omega}_{1_2}, \widetilde{\omega}_2) \leq \min(\omega_{1_1}, \omega_{1_2}, \omega_2)$$
$$\omega \leq \min(\omega_{1_1}, \omega_{1_2}, \omega_2)$$

□

References

[1] I. Akyildiz. Product form approximations for queueing networks with multiple servers and blocking. *IEEE Transactions on Computers*, 38(1):99–114, 1989.

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

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

[4] F. Belik. An efficient deadlock avoidance technique. *IEEE Transactions on Computers*, 39(7):882–888, 1990.

[5] P. Cameron. *Combinatorics: topics, techniques, algorithms*. Cambridge University Press, 1994.

[6] 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.

[7] 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.

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

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

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

[11] J. Ezpeleta, F. Tricas, F. Garca-Valls, and J.M. Colom. A banker's solution for deadlock avoidance in fms with flexible routing and multiresource states. *IEEE Transactions on Robotics and Automation*, 18(4):621–625, 2002.

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

[13] 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.

[14] 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.

[15] 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.

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

[17] 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.

[18] 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.

[19] R. Onvural. Survey of closed queueing networks with blocking. *ACM Computing Surveys*, 22(2):83–121, 1990.

[20] W. Stewart. *Probability, markov chains, queues, and simulation.* Princeton university press, 2009.

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

[22] S.T. Tan. *Calculus.* Brooks/Cole/Cengage Learning, 2009.

[23] N. Viswanadham, Y. Narahari, and T.L. Johnson. Deadlock prevention and deadlock avoidance in flexible manufacturing systems using petri net models. *IEEE Transactions on Robotics and Automation*, 6(6):713–723, 1990.

Appendix

Some definitions from graph theory that have been used in this report:

- **Order**: The order of the directed graph D is its number of vertices, denoted by $|V(D)|$.
- **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.
- **Descendants**: 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.
- $\mathbf{deg^{out}}(X)$: The out-degree of X is the number of outgoing edges emanating from that vertex.
- **Subgraph**: A subgraph H of a graph G is a graph whose vertices are a subset of the vertex set of G, and whose edges are a subset of the edge set of G.
- **Sink vertex**: A sink vertex is a vertex in a directed graph that has out-degree of zero.

- **Knot**: In a directed graph, a knot is a set of vertices with out-edges that are inescapable by traversing the edges.