

# OPEN QUEUEING NETWORKS WITH BLOCKING – A PERSONAL LOG

Harry G. Perros<sup>1</sup>

*Abstract*

*The analysis of queueing networks with blocking reached its peak in the 1980s and early 1990s. Several elegant decomposition-based approximation techniques were developed, and in this paper, the author gives a personal account of some of these techniques and also some of the players involved. The lessons learnt from analyzing such queueing networks have been useful in developing decomposition-based approximation algorithms for novel queueing networks which have been motivated by new networking technologies.*

## **1. A not so simple open queueing network with blocking**

I am very pleased to participate in the festivities for the 60<sup>th</sup> birthday of my good friend, colleague, and *almost* compatriot, Dr. Günter Haring.

*XRONIA POLLA, KAI NA TA EKATOSTHSEIS!*

I first met Günter back in the summer of 1986, the year of the Chernobyl nuclear disaster, during a conference in Vienna. I remember spending a pleasant evening with Günter and my colleague Billy Stewart at a *heuringen!* I also remembered how astonished I was when I discovered that he spoke fluent Greek!

At that time, there was still a lot of interest in the analysis of queueing networks, and also in the more specialized area of queueing networks with blocking. My first encounter with queueing networks with blocking was in 1972, when I started my PhD studies in Trinity College Dublin under the supervision of Professor F. G. Foster. Somewhere between Trinity College and my local pub, O'Neil's *The Crowing Cock*, I got the idea to model a set of IBM 2314 disks as an open queueing network with blocking. This is a queueing network of finite capacity nodes, where a customer upon completion of its service at a node gets blocked (i.e., stuck) if its destination node is full. The customer remains blocked until a space becomes available at the destination node. During the time the customer is blocked, the server that had just finished serving this customer is also blocked and it cannot serve any other customers. (This type of blocking should not be confused with the notion of a customer getting blocked in teletraffic, which means that the customer gets lost if it arrives at a node at a time when the node is full.)

---

<sup>1</sup> Department of Computer Science, North Carolina State University, Raleigh, NC 27695, USA

The simplest open queueing network with blocking is the one shown in figure 1. The first node has an infinite capacity and the second node has a finite capacity of size  $m$ . Service at each node is provided by a single exponential server. External arrivals join the first node in a Poisson fashion at the rate  $\lambda$ . Let  $\mu_1$ , and  $\mu_2$  be the service rate at the first and second server respectively. Customers in each node are served in a FIFO manner. We assume *blocking-after-service*. That is, the first server becomes blocked when a customer completes its service at a time the second node is full. The server remains blocked, and it cannot serve any other customer in its queue until a departure occurs from node 2.

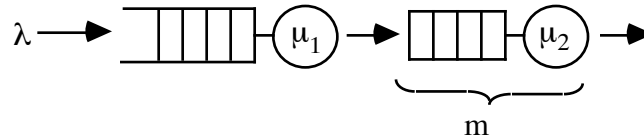


Figure 1: A two-node open queueing network with blocking

This problem looked like a piece of cake! I recall that about the time I was trying to analyze this simple queueing network, Isi Mitrani came over to give a talk. He and I spent quite a bit of time discussing this problem, and I also recalled that he introduced me to the art of rate diagrams, that by-passes the task of having to write down Kolmogorov's equations. However, all my efforts to solve this queueing network failed. I never managed to obtain the closed-form expression of the queue-length distribution in each node. In fact, except for special cases, this closed-form expression is still eluding human mankind!

Typically, in order to obtain the closed-form solution, one writes down the steady-state equations, which then transforms into a small set of equations by introducing generating functions. Following this approach, one will very quickly find out that in the case of general  $m$ , the number of unknowns is  $2m+3$ . This consists of  $m+2$  unknown generating functions and  $m+1$  unknown probabilities  $p(0,0), p(0,1), \dots, p(0,m)$ . The number of available independent equations is  $m+4$ , thus being  $m-1$  equations short. In view of this, with the exception of the case  $m=1$ , it is not possible to solve system of generating functions.

One way to obtain the additional equations is to exploit the fact that a generating function is analytic within the unit circle. In particular, let us consider a generating function  $g(z)$ , and let us assume that it can be written as a ratio of two polynomials, i.e.  $g(z)=f_1(z)/f_2(z)$ . Now, in order for  $g(z)$  to be analytic within the unit circle, all the zeroes of  $f_2(z)$  within the unit circle have to be zeroes of  $f_1(z)$  as well. For instance, let us assume that the zeroes of  $f_2(z)$  within the unit circle are  $\zeta_1, \zeta_2, \dots, \zeta_k$ . Then, we have that  $f_1(\zeta_i)=0$ ,  $i=1,2,\dots,k$ , which gives us  $k$  additional equations. This argument was used successfully by Konheim and Reiser [1] to analyze different queueing system. In general, the zeroes are obtained numerically, though in several cases, one can obtain their exact closed-form expression.

An alternative method is to use numerical techniques. Specifically, this simple queueing network has a matrix-geometric solution, see Neuts [2], as Marcel took great pleasure in

reminding me! The interested reader is referred to Perros [3] for further details on the analysis of this queueing network with blocking.

Since it was not possible to obtain the exact analytic closed-form expression for the above two-node open queueing network with blocking, the possibility of being able to analyze exactly large tandem configurations involving many nodes was nil. In view of this, a large number of different approximation algorithms were proposed, as described in the following section. I should mention that Billy Stewart always argued with me that large tandem configurations can be solved using numerical techniques. I do not believe that he ever managed to convince me all these years. By the way, speaking of Billy, I first met him at the banquet of the first Performance conference organized by Erol Gelenbe and colleagues in INRIA in the summer of 1974. Unfortunately, I do not recall our conversation, as we both had thoroughly availed of the free champagne.

## 2. Analyzing tandem open queueing networks with blocking

In 1975 I received my PhD, and before I knew it I was married, I had a young child, and I was teaching at the University of Illinois at Chicago. By that time, I had mastered the art of analyzing fairly accurately very small queueing networks with blocking, but my algorithms did not scale. I remember reading with awe papers on product-form queueing networks, where the authors analyzed huge queueing networks. I had decided to abandon my beloved queueing networks with blocking for better pastures, when two things happened. Firstly, I moved to North Carolina State University, and secondly my colleague Billy Stewart introduced me to Tayfur Altioek, who at that time was completing his PhD thesis on production lines under the direction of Sandy Stidham. Tayfur's problem was the opposite to mine! He had developed a way of analyzing approximately large tandem queueing networks with blocking, but his approximation algorithms did not have a good accuracy. In no time we put aside our national differences between Greece and Turkey, and set out to combine our methodologies. This collaboration led to many interesting publications that permitted us to analyze large tandem and feed-forward configurations of open queueing networks with blocking. The adrenaline rush when things go well is indeed an addiction that I have not yet forsaken!

There were too many researchers (friends and worthy adversaries!) to mention in this short paper who worked in this area at that time, tackling this problem from various angles and under various assumptions. The interested reader is referred to Perros [3] for details.

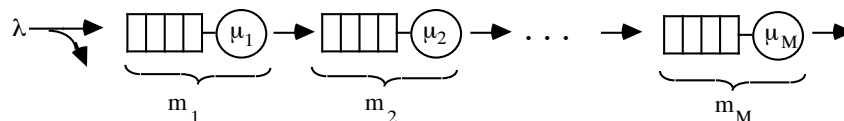


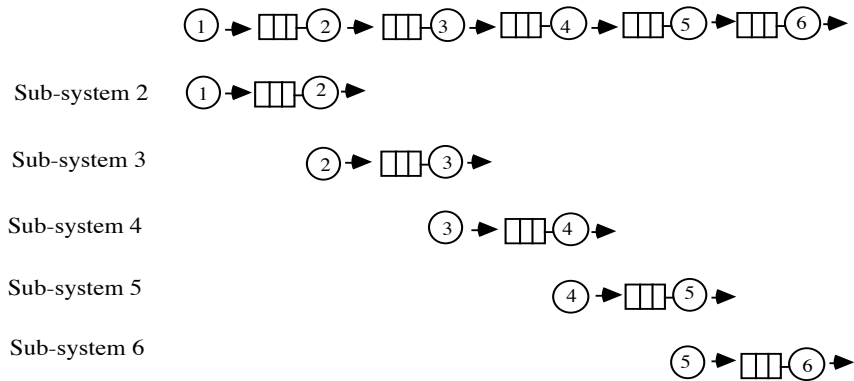
Figure 2: An open tandem queueing network with blocking

One of the techniques used to analyze approximately an open queueing network with tandem was to decompose it to single nodes, and then analyze each node separately. The

results from each node were then combined through an iterative scheme. Specifically, let us consider the open queueing network with blocking shown in figure 2. Each node  $i$  has a capacity of  $m_i$  customers and it is served by an exponential server at the rate of  $\mu_i$ . We assume Poisson arrivals to the first node. This configuration can be analyzed by decomposing it to  $M$  individual nodes. Unlike, other decomposition algorithms proposed for general queueing networks, such as Raymond Marie's algorithm, in this case one not only had to characterize the arrival process to each node but also one had to modify the service time to reflect the blocking delays due to a downstream node becoming full. (By the way, speaking of Raymond Marie, I have lots of stories about him, but this will require an additional 10 pages!)

If the arrival process to the queueing network is Poisson, then it is not a bad assumption to consider that the arrival process to each decomposed node is also Poisson. However, if we assume a Coxian arrival process, then it is imperative to characterize the arrival process to each decomposed node as a Coxian distribution. This, however, makes the above single-node decomposition algorithm computationally complex.

The approximation algorithm described below avoids the task of having to characterize the arrival process to each node, by cleverly including in the sub-model of each server, the server of the previous node. This is probably the best algorithm for analyzing tandem open queueing networks with blocking, and it is attributed to Altıok and Ranjan, [4] and also to Gün and Makowski [5]. The modification of the service time is done using a simple recursive mechanism described in Perros and Altıok [6].



**Figure 2: The sub-systems in the two-server decomposition algorithm.**

The algorithm was developed for an open tandem configuration with blocking-after-service, saturated first node, and phase-type service times. It decomposes the tandem configuration into sub-systems consisting of two successive servers and the in-between buffer. In figure 3, we show these sub-systems for a tandem configuration consisting of 6 nodes. We note that sub-system  $i$  consists of servers  $i-1$  and  $i$ , and the in-between buffer. Each sub-system  $i$  is analyzed in isolation after the service time at the  $(i-1)$ st and  $i$ th servers are modified. In particular, we need to know if upon service completion at the  $i$ th server the customer will get blocked, and how long it would remain blocked. Also, we

need to know if upon service completion at the  $(i-1)$ st server, the server becomes idle and for how long it remains idle. Thus, in the  $i$ th sub-system, the modified services of the  $(i-1)$ st and  $i$ th servers represent the effective arrival process to the  $i$ th node and the effective service time of the  $i$ th node, respectively. The parameters of the effective service time of the  $(i-1)$ st server and of the  $i$ th server are obtained using information from the  $(i-1)$ st and the  $(i+1)$ st sub-system respectively. In view of this, the following iterative scheme is employed. The algorithm starts with the 2nd sub-system and proceeds forward until it analyses the  $M$ th sub-system. It then executes a backward pass starting from the  $(M-1)$ st sub-system back to the 2nd sub-system. In the forward pass, the parameters of the first server of each sub-system are updated, while in the backward pass the parameters of the second server of each sub-system are updated. The algorithm iterates backwards and forwards until the throughputs of all the sub-systems are sufficiently close to each other. It yields the queue-length distribution at each node. More details can be found in Perros [3]. As is the case with many such iterative schemes, it is very difficult to prove convergence of this algorithm, something that made the numerical analysis community very unhappy!

### 3. The early 1990s: Discrete-time queueing theory galore

The early 1990s were marked by an intense research and development of the *Asynchronous Transfer Mode* (ATM) networks. ATM gave rise to fascinating performance problems, many of which were addressed using discrete-time queueing theory. There is a vast literature on continuous-time queueing networks, but very little was known about discrete-time queueing networks in the late 1980s. The same, of course, applied to the more specialized area of queueing networks with blocking.

In ATM, there is no blocking as understood in queueing networks with blocking. There is blocking as understood in teletraffic, i.e. lost customers. However, designers of ATM switches often used a backpressure mechanism, as a way of avoiding dropping ATM cells within a switch fabric. This backpressure mechanism can be modelled using a queueing network with blocking.

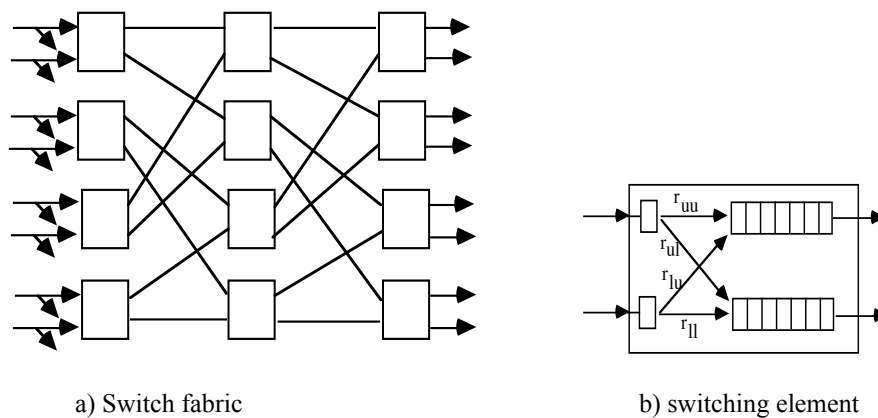


Figure 3: An 8X8 Banyan Network.

As an example, let us consider the 8x8 buffered Banyan switch shown in figure 3a, where each box represents a switching element. The function of the switch is to route cells from its input ports to its output ports. This is done by moving a cell forward from one switching element to another until the cell eventually departs from the switch. Each switching element is a 2x2 crossbar as shown in figure 3b. Each input port of the switching element has a buffer of length one, referred to as the *input buffer*, and each output port consists of a buffer of length  $m$  referred to as the *output buffer*.

The time it takes to transfer a cell from one buffer to another, within a switching element or between two adjacent switching elements, is constant and it is referred to as the *switch cycle*. All the switching elements are synchronized, so that all transfers of cells from one buffer to another begin and end at the same time. Cells arrive at the switch in an asynchronous manner but they do not enter the switch until the next switch cycle. The basic operation of the switch is as follows:

1. Upon arrival to the switch, a cell waits until the next switch cycle. At that time, if the corresponding input buffer is empty, the cell is forwarded to this buffer. If the buffer is full then the cell is lost.
2. A cell in an input buffer of a switching element is forwarded to its destination output buffer of the same switching element if there is a space. If there is no space, then the cell is blocked. That is, it is forced to wait in the input buffer for another switch cycle before it attempts to enter the output buffer. Blocking will also occur if both input buffers contain a cell destined to the same output buffer and the output buffer is either full or it has one free space. If there is only one free space at the output buffer, then only one cell, chosen randomly from the two input buffers, will be forwarded.
3. A cell at the head of the output buffer is forwarded to the next switching element if the corresponding input buffer is free. If the input buffer is not free, the cell is blocked. The cell will wait until the next switch cycle, and if at that time the input buffer is free, it will be forwarded to the next stage. Else, it will wait for another switch cycle, and so on. This *blocking-before-service* mechanism prevents cell loss between the switching elements.

All buffers function so that a full buffer will accept another cell if the cell at the head of the buffer is departing at the same time.

The arrival process to each input port of the switch is assumed to be bursty and it is modelled by an Interrupted Bernoulli Process (IBP). That is, the incoming link into an input port of the switch is slotted. Each slot is assumed to be equal to the switch cycle. A slot is long enough to contain one cell. An arriving slot may or may not contain a cell. In an IBP, we have a geometrically distributed period during which no arrivals occur (idle state), followed by a geometrically distributed period during which arrivals occur in a Bernoulli fashion (active state). Given that the process is in the active state respectively idle state at slot  $t$ , it will remain in the same state in the next slot  $t+1$  with probability  $p$  respectively  $q$ . During the active state, a slot contains a cell with probability  $\lambda$ . It can be shown that the average arrival rate is  $\rho = \lambda(1-q)/(2-p-q)$ , and the squared coefficient of variation of the time between successive arrivals is  $c^2 = 1 + \lambda(-1 + [(1-p)(p+q)/(2-p-q)^2])$ .

The entire switch can be viewed as a discrete-time queueing network with blocking, where each node represents a switching element. The switch can be analyzed, therefore, by decomposing it into the individual switching elements, and then analyzing each switching element independently of the other switching elements. In order to accurately analyze a switching element in isolation, we need to characterize the arrival process to each input buffer of a switching element. This is done using the departure process from the output buffer of the corresponding switching element in the previous stage. Since it takes one switch cycle to forward a cell from an output buffer to the next switching element, we can think of the output buffer as being served by a server which has a constant service equal to one switch cycle. Due to blocking, we distinguish between the *actual* and the *attempted* departure process. The actual departure process is associated with departure instants where a cell leaves the switching element. The attempted departure process involves all instants of service completion independent of whether the cell leaves or gets blocked and is forced to receive another service. In this study, we worked with the attempted departure process, which we approximated by an IBP. This departure process becomes the offered arrival process to the input buffer of the switching element in the next stage. Part of this process is lost due to cells arriving to a full input buffer.

In order to analyze a switching element in isolation we also require knowledge of the blocking probability  $\pi$ . This is the probability that a cell at the head of an output buffer is blocked at the beginning of a switch cycle due to the destination input buffer being full. This probability is used to represent the service at an output buffer by a geometric distribution. That is, a cell receives a service equal to a switch cycle and then with probability  $1-\pi$  it departs from the switching element, or with probability  $\pi$  it is blocked and it is forced to receive another service.

The details for calculating the attempted departure process from an output buffer and the blocking probability  $\pi$  can be found in Morris and Perros [7]. Given that the arrival process to each input buffer of a switching element and  $\pi$  are known, the switching element can be analyzed numerically by solving the underlying system of linear equations  $\mathbf{xP}=\mathbf{0}$ .

The entire switch is analyzed using the following iterative scheme. An iteration consists of visiting each switching element beginning at the first stage of the switch and continuing sequentially to the last stage of the switch. For the analysis of each switching element, we use the blocking probabilities from the previous iteration and the attempted arrival processes from the current iteration.

- Step 0:* For each switching element assume initial values for the blocking probability for each input buffer. Set  $i$  to the first stage of the switch ( $i=1$ ).
- Step 1:* Analyze numerically each switching element in stage  $i$ .
- Step 2:* For each switching element in stage  $i$  calculate the new attempted departure process from each output buffer and the new blocking probability for each input buffer using any of the three models described above. Note that the new attempted departure process becomes the input process to the next stage.

*Step 3* If stage  $i$  is not the last stage of the switch, then set  $i=i+1$  and go to step 1. Else, this iteration is completed. Go to step 4.

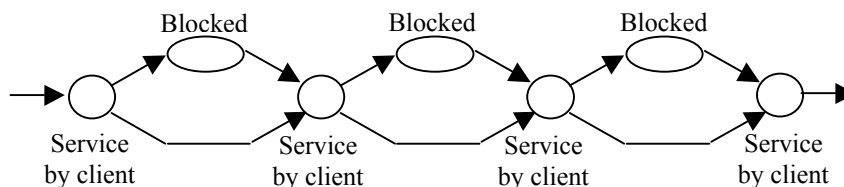
*Step 4:* The convergence test compares  $p(\mathbf{n})$  for each switching element from one iteration to the next. The tolerance we used was nine decimal place accuracy.

Stop, if convergence has occurred. Else set  $i=1$  and go to step 1.

The approximation algorithm gives the queue-length distribution within the buffers of each switching element. From this, important measures such as cell loss probability, mean time to traverse the switch, and throughput can be obtained.

#### 4. Queueing networks with blocking and simultaneous resource possession

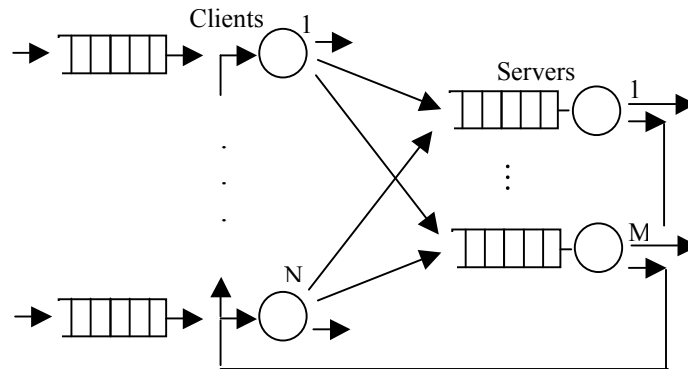
So far we have assumed that blocking occurs due a downstream node becoming full. In this section, we will consider a queueing network where blocking of a customer is due to simultaneous resource possession. This queueing network arises in client-server type of applications, where a group of software clients interacts with a group of software servers through *synchronous* and *asynchronous* messages. A client receives requests from local users which it processes one at a time. The processing of a request involves some local processing by the client followed by a *synchronous* or *asynchronous* message to a server. The client gets blocked if it sends a synchronous message. That is, it cannot do any processing until it receives a response from the server. The client does not get blocked if it sends an asynchronous message. The cycle consisting of some local processing by the client followed by a synchronous or asynchronous message to a server, repeats for a number of times, as shown in figure 4, until the request is completely processed. The number of times that the cycle repeats, the specific server that the client will send a synchronous or asynchronous message each time it completes some local processing, and the distribution of the local processing time, depend on the class of the request.



**Figure 4: The processing time of a request at a server**

Due to lack of space in this paper, we will only consider synchronous messages, one tier of servers, and a single class of customer. We will also assume that each client and each server runs on a separate CPU. The queueing network shown in figure 5, depicts  $N$  clients and a single tier of  $M$  servers. Each client receives requests which are queued in an input buffer with an infinite capacity. Requests arrive at a client following a two-phase Coxian distribution. The local processing time at each client (i.e., the service provided by a client in-between messages to the servers) and the service times at each server are all assumed to be exponentially distributed. The service pattern shown in figure 4 is the same for all requests at all clients since we assumed a single class of customers, and it follows an arbitrary pattern. A request is first processed by the client, and then a synchronous message is sent to a specific server  $i$ . During the time that the synchronous

message is queued up and processed by the  $i$ th server, the client is blocked and it cannot process any other requests. When the  $i$ th server completes processing the synchronous message, it sends back to the client a response, and the client can proceed with some more local processing at the end of which it will send another synchronous message to a server, and so on. Finally, the request receives a last round of local service by the client and then it departs. At that time, the client can start processing the next request queueing up in its queue.

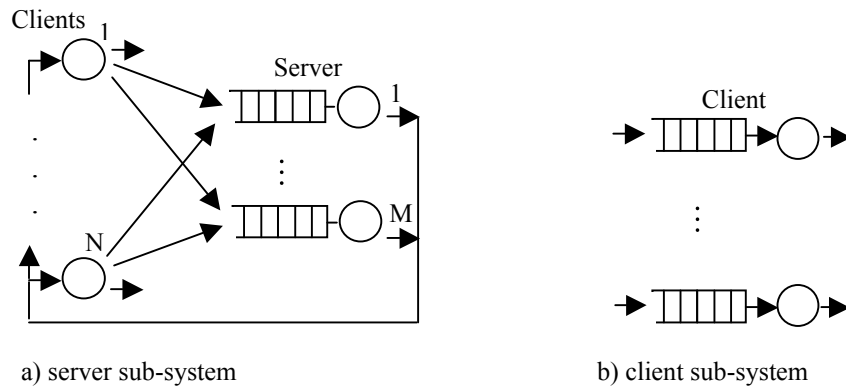


**Figure 5: The client-server queueing network**

This queueing network was proposed and analyzed by Ramesh and Perros [8] and [9]. The proposed approximation algorithm can be seen as an extension of the two-server decomposition algorithm presented above in section 2. The approximation algorithm decomposes the client-server queueing network to two sub-systems, namely, the *server* sub-system and the *client* sub-system. These two sub-systems are shown in figure 6. The server sub-system is a closed queueing network model that includes the  $M$  server queues and the  $N$  clients, but without the queue in front of each client. The service time of each client is modified to account for the lack of knowledge about its queue. The client subsystem consists of  $N$  independent nodes, one per client queue. The service time at each node is modified to reflect the blocking delays due to synchronous messages. The analysis of each sub-system requires information that can be obtained from the other sub-system. This is resolved using an iterative scheme.

There are  $N$  customers in the server sub-system, and each customer  $n$  represents the  $n$ th client,  $n=1,2,\dots,N$ . Each customer is associated with a different chain consisting of  $K$  different classes, where  $K$  is the total number of synchronous messages a client issues for each request. A new request beginning its service at client  $n$  is represented in the server sub-system by the  $n$ th customer starting its service at the  $n$ th client. At that moment, the  $n$ th customer is in class 1. When the  $n$ th client issues the first synchronous message to a server, say server  $i$ , this is represented by the  $n$ th customer joining the queue of server  $i$ . After the service completion at server  $i$ , the  $n$ th customer returns to the  $n$ th client and changes class from 1 to 2. In general, each time the  $n$ th customer returns to the  $n$ th client, after it has gone through a server, it changes class from  $k$  to  $k+1$ ,  $k=1,2,\dots,K-1$ . The request leaves the queueing network after it has been fully serviced. This is represented in the server sub-system by the  $n$ th customer going back to the  $n$ th client as class 1.

Associating each cycle of the processing of a request with a different class, permits the introduction of different routing probabilities and client service times for each cycle. The service time at the  $n$ th client when the  $n$ th customer is in class 1 is augmented to allow the case where the  $n$ th client becomes idle after a request departs from the queueing network. The information necessary for augmenting this service time is obtained from the client sub-system. This network has a product-form solution, which drastically simplifies its solution



**Figure 6: The two sub-systems**

The client sub-system consists of  $N$  independent nodes, one per client queue. Each node is analyzed separately, after its service time has been augmented. Specifically, we calculate the sojourn time for each server queue from the server sub-system, which is then reduced to a two-phase Coxian distribution. These distributions are incorporated into the service mechanism of each client node. The arrival process to each client node is not modified.

As mentioned above the algorithm iterates between the two sub-systems until convergence. This approximation method has been extended to multiple tiers of servers, which are accessed either in a hierarchical or non-hierarchical fashion (see [8] and [9]).

## 5. Is there life after queueing networks with blocking?

The lessons learnt from analyzing open queueing networks with blocking have been useful in developing decomposition-based approximation algorithms for novel queueing networks motivated by new networking technologies. Of interest is a new type of queueing network with simultaneous resource possession where the resources used simultaneously by a customer change dynamically, as the customer moves through the network. Such queueing networks arise when modelling satellite constellations where a phone call may use a set of different channels as the satellites move around their orbits. Also, the same type of queueing network arises in a new transport scheme for optical networks, known as optical burst switching, where a burst may occupy simultaneously several links along the path.

## References

- [1] KONHEIM, A.G. and REISER, M., A queueing model with finite waiting room and blocking, *J.ACM* **23** (1976) 328-341.
- [2] NEUTS, M.F., Two-queues in series with a finite, intermediate waiting room, *J. Appl. Probab.* **5** (1968) 123-142.
- [3] PERROS, H.G., Queueing networks with blocking: exact and approximate solutions, Oxford University Press, 1994.
- [4] ALTIOK, T. and RANJAN, R., Analysis of production lines with general service times and finite buffers: A two-node decomposition approach, *Engineering Costs and Production Economics* **17** (1988) 155-165.
- [5] GUN, L. and MAKOWSKI, A.M., An approximation method for general tandem queueing systems subject to blocking, in Perros and Altiok, (Eds.), *Proc. First International Workshop on Queueing Networks with Blocking*, (North Holland, Amsterdam, 1989).
- [6] PERROS, H.G. and ALTIOK, T., Approximate analysis of open networks of queues with blocking: tandem configurations, *IEEE Trans. Soft. Eng.* **12** (1986) 450-461.
- [7] MORRIS, T.D. and PERROS, H.G., Performance modelling of a multi-buffered Banyan switch under bursty traffic, *INFOCOM '92*, Florence, Italy, May 1992, 436-445.
- [8] RAMESH, S. and PERROS, H.G., A multi-layered queueing network model of a client-server system with synchronous and asynchronous messages", *IEEE Trans. Software Engineering*. Vol 26, 1086-1100, Nov. 2000.
- [9] RAMESH, S., and PERROS, H.G., A multi-layer client-server queueing network model with synchronous and asynchronous non-hierarchical messages", *Performance Evaluation J*, Vol. 45 (4) (2001) pp. 223-256.