# ANALYTICAL MODELS FOR MULTI-STAGE INTERCONNECTION NETWORKS

P HARRISON

**Rapporteur**: S Caughey

# ANALYTICAL MODELS FOR MULTI-STAGE INTERCONNECTION NETWORKS

Peter G Harrison
Department of Computing
Imperial College
London

## Abstract

Central to all parallel architectures is a *switching network* which facilitates the communication between a machine's components necessary to support their cooperation. Multi-stage interconnection networks (MINs) are classified and two queueing models for packet-switched MINs with unlimited buffer space are introduced. The first uses standard techniques and is exact with respect to its assumptions, hence providing a standard against which to assess approximate models. From this exact model, we can also obtain distributions of transmission times; previous work has either used simulation, which can be unreliable and is expensive to run, or produced only Laplace transforms. The second model has much milder assumptions, is more generally applicable and can be implemented more efficiently, but is approximate. However, it has been found to give accurate predictions for a wide range of traffic patterns and distributions of link transmission times. Established techniques can be integrated into our queueing-based methodology to model MINs with finite buffers and hence blocking.

## 1. Introduction

The complexity of large-scale parallel computer architectures is such that their effective design requires guidance from some form of predictive modelling prior to construction. Analytical models based on queueing network analysis provide a powerful tool for this purpose; see for example [HF86]. Central to all parallel architectures is a *switching network* which facilitates the communication between a machine's components necessary to support their cooperation. It is therefore crucial to develop efficient models of the performance of such networks and to integrate them into existing, accepted modelling methodologies.

Generally in multiprocessing systems, the processors can communicate and cooperate at different levels in solving a given problem, i.e. by sending messages or by sharing memory. If there is relatively little processor interaction via shared memory the parallel system is said to be *loosely coupled.* Thus in loosely coupled multiprocessors, the processors have large local memories from which most of its instructions and data are obtained. Communication between processors is achieved by sending messages across an *interconnection network* (IN), also often called a *switch* or *switching network.* In this case, full connectivity is not essential as the processors spend little time communicating with each other. Thus an IN with a fixed or *static* topology is suitable and conventional queueing models are suitable for performance analysis.

Parallel systems are said to be *tightly coupled* if there is a lot of processor interaction via shared memory. Thus in tightly coupled multiprocessors, the processors communicate through shared main memory and the IN provides *full* connectivity between processors and memory. Usually a small local memory or cache is attached to each processor to improve overall performance. The speed of the machine is restricted by the memory bandwidth, and hence by the IN topology, and an IN with a *dynamic* topology is required. ALICE [HR86], designed to execute functional languages in parallel [FH88], and the NYU Ultracomputer [Go83] are examples of tightly coupled multiprocessor systems. This homogeneous type of architecture is illustrated in Figure 1.1 in which processing elements (PEs) and memory modules (MMs) communicate through a dynamic IN topology. The dynamic IN can make connections from any PE to any MM.

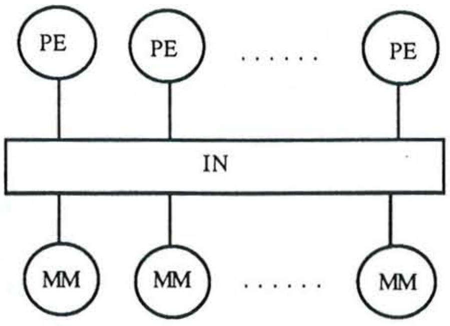


**Figure 1.1 IN in a tightly coupled architecture**

In this paper, we first consider the different types of interconnection networks and focus on the widely used delta network which is composed of several stages of simple switches. We then describe, in Section 3, a simple queueing model of a delta network under appropriate simplifying assumptions. This provides a standard against which to assess approximate methods in simple cases, and such an approximation is presented. In Section 4 we use the former model to determine characteristics of transmission times, using recent results on the density of passage times through networks of queues. We conclude in section 5 and outline some of the limitations of the techniques presented, for example the handling of circuit-switching (considered in the second lecture and [HP90]) and blocking.

## 2. A classification of interconnection networks

The simplest type of switch for communication in parallel computer architectures is the *common bus* which connects to every component: processing element, memory module or other. However, the bus derives its simplicity at the expense of increased complexity in each component that it connects. For example, bus request and receive pins are needed by each element, together with several others for control and synchronisation, and non-trivial logic is necessary to implement the bus's protocol. This increases the overall cost of a large-scale multiprocessor and, moreover, each additional processor that is added to the bus is faced with more contention and so delivers a diminishing increment in performance.

At the other end of the spectrum, the switch might be a full crossbar which provides parallel communication between any number of distinct pairs of processors connected through it. This facilitates, for example, parallel access to shared memory provided that there are no *memory conflicts*, i.e. two or more processors attempting to access the same memory address. An a×b crossbar can switch any of its a input pins to any of its b output pins on the appropriate clock-cycle, so that the only contention possible is for the destination processors, i.e. is internal in the switch for the output pins. We therefore have the configuration shown below in Figure 2.1 when a = b = 4.

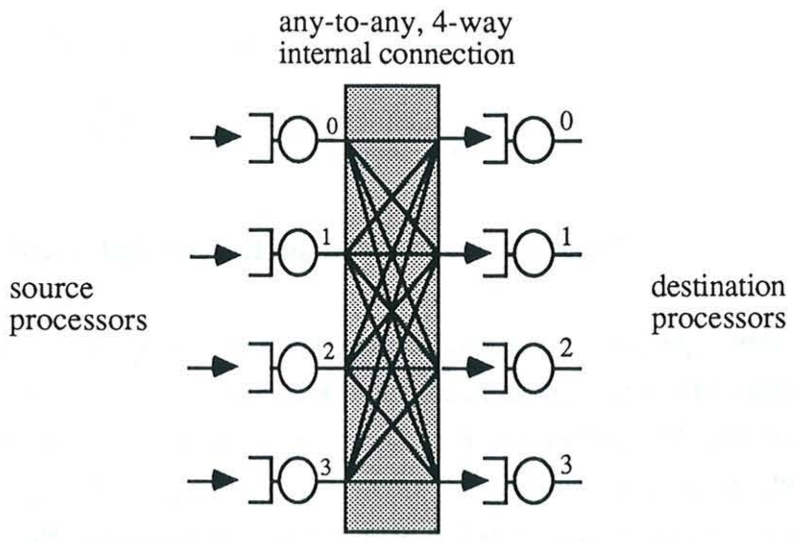


**Figure 2.1 The 4-way crossbar switch**

Each additional processor that is added to the crossbar requires a new parallel link to each of the output pins. Thus for a N-way crossbar the complexity and cost grow as $O(N^2)$ and the very large crossbars required for large-scale multiprocessors would become hopelessly complex and expensive. Moreover, large crossbars are physically infeasible to construct using contemporary electronic devices.

However, we can interconnect a matrix of crossbars to form a *multi-stage interconnection network* (MIN) which has a number of levels (or stages) such that the output pins of crossbars in one stage are connected to the input pins of crossbars in the next stage. This provides the same connectivity as a full crossbar (any input pin can be dynamically connected to any output pin if there is a free path) but introduces contention for internal links (*path conflicts*) on top of the memory conflicts. The cost of an $N$-way MIN grows almost linearly, $O(N\log N)$, compared with $O(N^2)$ for the equivalent crossbar. There are several types of MINs, many with similar structure and properties, but here we will consider a subset of *banyan networks*, named after a type of Indian fig tree with a similar structure. A banyan network is a MIN with a unique path from each input pin to each output pin and is said to be *layered* if its switches can be arranged as a number of distinct stages. A layered banyan comprising $J$ stages of crossbars in which outputs of one stage connect directly to the inputs of the next stage, is called a *J-level banyan*. There are two main types of $J$-level banyans: *regular banyans* and *irregular banyans*. Irregular banyans connect any $N$ inputs to $M$ outputs through $J$ stages of crossbars where each stage comprises a set of identical crossbars.

Regular banyans are constructed from a single type of basic crossbar throughout all stages.

## 2.1 Delta Networks

A well-known example of a regular banyan network is the delta network defined in [Pa81]. The definition of a delta network includes the additional property that routing in the network is *digit-controlled*, i.e. the choice of which output pin to select at a particular crossbar in the network can be determined by a single digit in the destination address of the packet. A rectangular delta network constructed from b-way crossbars with the same number of input pins as output pins is called a *delta-b* network. There are a number of different ways of connecting the outputs of one stage of crossbars to the inputs of the next so as to obtain the desired connectivity. The corresponding permutation functions define the network's topology. All these topologies are equivalent in that one can be obtained from another by permuting the switches in each stage of the network and so all have the same performance characteristics, such as equilibrium throughput, [WF80]. The following classification is taken from the PhD thesis of Naresh Patel [Pa89].

We may interpret each switching element in a MIN as a *node* (or vertex) and each link as an *edge* (or arc) in a *directed graph*. If a link connects two crossbars $A$ and $B$, then the corresponding edge in the graph has two labels: the first one corresponding to the output pin number of $A$ and the second to the input pin number of $B$. Although the edges are directed in the graph representation, the actual links may be bi-directional. For each graph $G$ representing a particular MIN, there is a reverse graph $G^R$ which is formed by reversing all the edges and swapping the input and output nodes. In general, the path from a given input node to some output node in a delta network can be described by a *path descriptor*. In a $J$-stage network, this is a string of $J$ digits where the $i^{th}$ digit indicates the output pin number of the switch in stage $i$ on the path ($1 \leq i \leq J$). If this path descriptor is independent of the input node then we can associate it with the output node address and no longer require to store path descriptors at each input node. In delta networks, this independence of source node is maintained for all nodes and usually the address or the reversed address of the output node is the path descriptor. Hence messages can be routed through the network by consuming the destination address string from one end.

It is useful to have this delta property in the reverse direction as well as the forwards direction because messages sent in one direction need acknowledgements in the other direction. Thus a delta network with the self-routing property in both directions is called a *bidelta network*. In graph terms, a delta network $G$ is a bidelta network if $G^R$ represents a delta network. Each node $A$ in a bidelta network is now associated with two types of path descriptors: forward (fpd($A$)) describing the routing to it from an input node and backward (bpd($A$)) describing the reverse route from an output node to node $A$. The way in which node $A$ is labelled using fpd($A$) and bpd($A$) determines the topology of the network. Each descriptor can be reversed and the two descriptors can be taken either way round. Hence there are eight possible topologies T given fixed "baseline" functions FPD, BPD and the operations reverse ( $^R$) and pairing ( , ):

- $T = (FPD(A), BPD(A))$ which gives the *baseline,* or *partial shuffle* network and the reversed topology, $T^R = (FPD(A), BPD(A))$;
- $T = (BPD^R(A), FPD(A))$ which gives the *omega,* or *perfect shuffle* network and the reversed topology, $T^R = (FPD(A), BPD^R(A))$;
- $T = (BPD(A), FPD^R(A))$ which gives the *binary n-cube* network and the reversed topology, $T^R = (FPD^R(A), BPD(A))$, often called a *flip* network;
- $T = (FPD^R(A), BPD^R(A))$ and $T^R = (BPD^R(A), FPD(A))$ which are uncommon.

These topologies are illustrated in figure 2.2 for a 3-stage, delta-2 network. For topology T with $T(A) = (t_1, t_2)$ for node $A$, the selector functions fpd and bpd are defined by fpd($A$) = $t_1$ and bpd($A$) = $t_2$. Suppose now that an edge labelled $(a,b)$ connects node $A$ in stage $j$ to node $B$ in stage $j+1$ with labels $\alpha_1,\alpha_2,...,\alpha_{J-1}$ and $\beta_1,\beta_2,...,\beta_{J-1}$ in a $J$-stage network. The label of a switch or pin in any stage is just its J-digit sequence number, with radix b, starting with 0 at the top. (Recall there are $b^{J-1}$ switches and $b^J$ pins in each stage.) In any of these topologies:

$$\text{fpd}(B) = \text{fpd}(A)<>[a] \text{ and } \text{bpd}(A) = \text{bpd}(B)<>[b]$$

where <> denotes the "append" or "concatenate" operation.

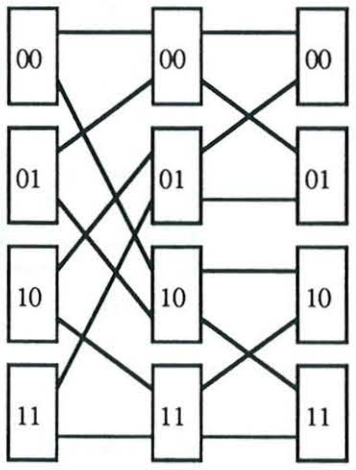
For the baseline network, fpd = FPD and bpd = BPD so that

$$\text{fpd}(A) = \alpha_1,\alpha_2,...,\alpha_{j-1} \text{ and } \text{bpd}(A) = \alpha_j,\alpha_{j+1},...,\alpha_{J-1}$$

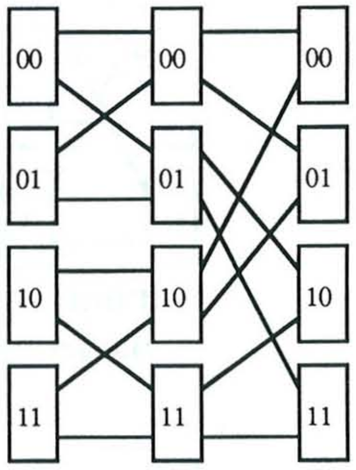
and likewise $\text{fpd}(B) = \beta_1,\beta_2,...,\beta_j$ and $\text{bpd}(B) = \beta_{j+1},\beta_{j+2},...,\beta_{J-1}$.

From the first equality fpd($B$) = fpd($A$)<>[$a$] we have: $\beta_i = \alpha_i$ $(1 \le i \le j-1)$ and $\beta_j = a$ and from bpd($A$) = bpd($B$)<>[$b$] we have: $\beta_i = \alpha_{i-1}$ $(j+1 \le i \le J-1)$ and $\alpha_{J-1} = b$. Thus any node with label $\alpha_1,\alpha_2,...,\alpha_{J-2},b$ in stage $j$ $(j=1,2,\ldots,J-1)$ connects to nodes in stage $j+1$ with labels $\alpha_1,\alpha_2,.\alpha_{j-1},a,\alpha_j,..,\alpha_{J-1}$ where $a \in \{0,1,...,\text{outdegree}(A)\}$.

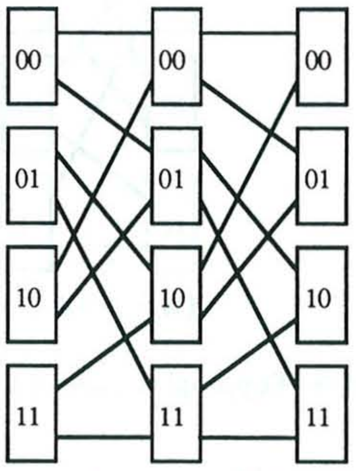
Connections for other topologies follow using a similar argument based on forward and backward path descriptors. It is shown in [WF80] that all eight topologies are isomorphic.



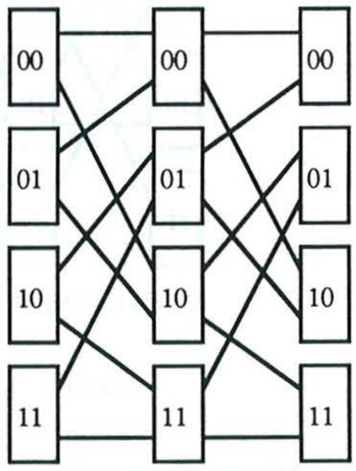
**(FPD($A$), BPD($A$))**
**(Partial shuffle)**



**(BPD($A$), FPD($A$))**
**(Reversed partial shuffle)**



**(BPD$^R$($A$), FPD($A$))**
**(Omega)**



**(FPD$^R$($A$), BPD($A$))**
**(Flip)**

**Figure 2.2 Delta Network Topologies (first part)**

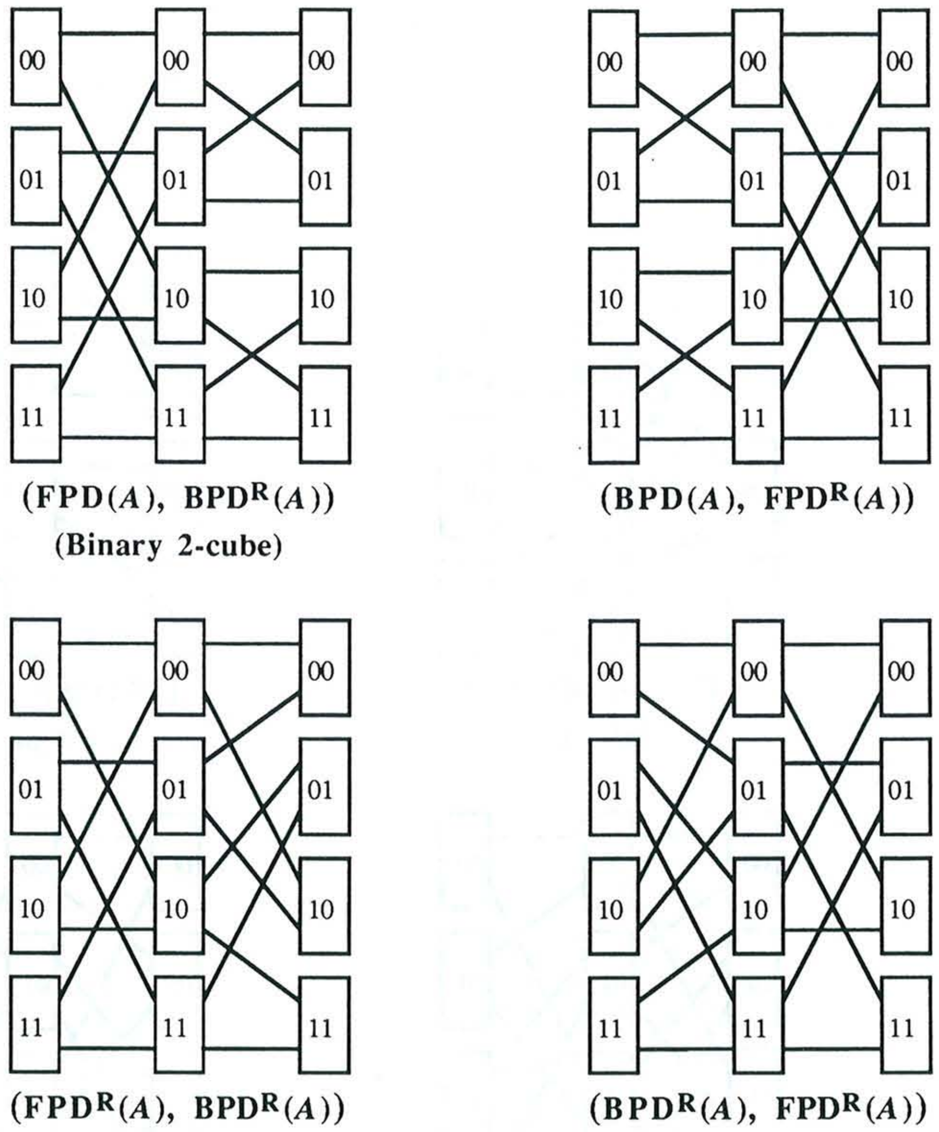


**Figure 2.2 Delta Network Topologies (second part)**

Notice that the mirror image topology is formed by swapping the fpd and the bpd, and that the topologies labelled with fpd$^R$ (i.e. a reversed forward path descriptor) require the destination address to be consumed from the least significant bit whereas in the other four topologies the address is consumed from the other (most significant) end.

## 2.2 Operational Characteristics of Switching Networks

MINs have three main operational characteristics: control, timing and switching protocol. Typically, MINs have decentralised control with implicit digit-controlled routing as in delta networks. However, timing characteristics depend on the type of parallel machine that uses the MIN. For SIMD machines, the MIN is normally used in *synchronous* mode whereas in MIMD machines the MIN is used in *asynchronous* mode. As mentioned in the Introduction, there are two main types of switching protocols - *circuit-switching* and *packet-switching* - as well as hybrid protocols. In circuit switching, a complete path has to be established across the network before data can be transferred from the input buffer to the destination buffer. In the process of setting up this path there may be a required link already in use which causes the path to be blocked. The partial path already established may then be held until the link becomes free or may be released whereupon the source processor may have to retry after some time. In a packet-switched MIN, there are buffers at each crossbar switch and packets of data are transferred from one buffer to the next in a single hop fashion. This protocol incurs higher transfer delays than circuit switching but introduces greater potential for parallelism and higher throughput. Hybrid protocols can also be used in buffered MINs in which packets can by-pass many buffers by forming circuits across any number of stages when possible.

## 2.3 Models of MINs

The different types of models for MINs essentially correspond to the various different operational characteristics. We therefore classify models as either synchronous or asynchronous with either a circuit-switched or packet-switched protocol and, in the latter case, infinite or finite capacity buffers. (In the former case we can also distinguish protocols that hold partial paths from those that do not). Over the past ten years numerous synchronous MIN models have been proposed primarily as a result of interest in SIMD computers. Much of this work involves the *permutation capability* of MINs, i.e. the number of total, 1-1 functions mapping input ports to output ports that can be realised by the MIN without path conflicts. Other work in this area has concentrated on determining the effective bandwidth of the MIN when requests arrive at an input with a given probability at the beginning of the time slice. In particular, [Pa81] and [DJ81, Je83] consider synchronous circuit-switched and packet-switched networks respectively.

However, we will be concerned with asynchronous models. Circuit-switching will be considered in the next lecture, [HP90], and here we consider packet-switched networks with buffers that are assumed never to overflow - i.e. to have infinite capacity in modelling terminology. Under appropriately strong simplifying assumptions we will be able to obtain exact results, but more generally approximations are necessary and we present one such. We will not be concerned with the crucial problem of *blocking* - the suspension of one server's activity in a queueing network by another which has a full queue. However, existing approximate methods, such as [Ak87], can be incorporated into our methodology.

## 3. A queueing network model for packet-switched MINs

As we have seen, there are a number of different, but isomorphic, ways of connecting the outputs of one stage of crossbars to the inputs of the next so as to obtain the desired connectivity. We choose the *partial shuffle topology*, [Pa81], which is defined below. However, any topology would be equally appropriate for our analysis since it merely serves to provide an ennumeration for the queues in our queueing network model. The partial shuffle topology for an s-stage delta-2 network is defined inductively as follows:

(i) A one-stage network, $\Delta_1$, is the single 2-way crossbar

(ii) An s-stage network, $\Delta_s$, (s>0) consists of one stage (numbered s) of $2^{s-1}$ switches connected to the right of 2 sub-networks of type $\Delta_{s-1}$. The *i*th switch in stage s takes its top input and bottom input from the *i*th pin of the upper sub-network and the *i*th pin of the lower sub-network, respectively.

We consider packet-switched delta networks with sufficient buffer size that blocking effects can be neglected. We assume that message transmission times through any crossbar or other processor are exponentially distributed and that all queueing disciplines are first-come-first-served. We can therefore model the delta network as a Markovian queueing network [Ja63, GN67] which provides a benchmark against which to assess precisely more generally applicable approximate models such as [Ha86, HK87].

### 3.1 A queueing network model

If we were to consider an open queueing network model with independent Poisson arrivals at its inputs, by Burke's theorem the arrivals at *every* switch would be Poisson because of the fan-out structure of the delta network. We could then model the system as a collection of independent single server (M/M/1) queues which have simple solutions. However, such an open model is too simplistic to give a good representation of the real world and instead we consider a delta-2 MIN embedded in some computer architecture. This leads to a closed queueing network model in which each output queue in the constituent crossbars is represented by a server. In this closed model, independence is lost, arrival processes are not Poisson and so the "open" approach fails. For the sake of simplicity, the rest of the system, i.e. servers other than those in the delta network, is modelled by a single exponential server. However, the analysis applies equally well when the delta network is embedded in an arbitrary network of servers of the BCMP type [BCMP75]. In fact, we could short-circuit the whole of the rest of the system to derive a "flow equivalent server". This could then be used to replace the MIN in any encompassing queueing network in a decomposition-based analysis, [HP90].

We therefore consider a closed Markovian queueing network of M servers with population N, [Ja63, GN67]. Each server has first-come first-served (FCFS) queueing discipline and exponentially distributed service times with load-independent mean - i.e. we have *constant rate* servers. If only mean transmission times are required, milder assumptions suffice. Then, each server's rate may vary with its queue length, other disciplines can be modelled and multiple customer classes are allowed, see for example [BCMP75].

### 3.1.1 Notation and basic results

We will use the following notation:

- $\mu_i$ is the constant service rate specified for server i ($1 \le i \le M$).

- $e_i$ is the *visitation rate* of server i ($1 \le i \le M$). The vector **e** is any non-zero solution of the equations

  $$e_i = \sum_{j=1}^{M} e_j p_{ji} \qquad (1 \le i \le M)$$

  where $p_{ji}$ is the *routing probability* between servers j and i, i.e. the constant probability that, on completing service at server j, a customer next visits server i. Either the routing probabilities or the visitation rates directly are normally specified for a queueing network model.

- $x_i = \dfrac{e_i}{\mu_i}$.

- $S(N) = \left\{ \underline{n} \mid \sum_{i=1}^{M} n_i = N;\ n_i \ge 0,\ 1 \le i \le M \right\}$ is the state space of the queueing network.

- $P(\underline{n}) = \dfrac{1}{G(N)} \prod_{i=1}^{M} x_i^{n_i}$ is the *equilibrium probability distribution* of the state $\underline{n} \in S(N)$ by the result in [GN67], where G(N) is the *normalising constant* defined by:

- $G(N) = \sum_{\underline{n} \in S(N)} \prod_{i=1}^{M} x_i^{n_i}$

By a result in [Bu73], the normalising constant can be computed by the following simple recurrence: $G(N) = g(M,N)$ where

$$g(m,n) = g(m-1,n) + x_m g(m,n-1) \qquad (m,n \ge 1)$$

$$g(m,0) = 1 \quad (m \geq 0), \quad G(0,n) = 0 \quad (n > 0)$$

Finally, we give expressions for the mean queue length and throughput of a server in terms of the normalising function g which we will find invaluable in the next section. Let $G_j(n)$ be the normalising constant for the network with server j removed and population n, i.e.

$$G_j(N) = \sum_{\underline{n} \in S_j(N)} \prod_{\substack{i=1 \\ i \neq j}}^{M} x_i^{n_i}$$

where $S_j(N) = \left\{ (n_i) \mid \sum_{i \neq j} n_i = N;\ n_i \geq 0,\ 1 \leq i \neq j \leq M \right\}$.

Then we have the following:

**Proposition 3.1**

The equilibrium probability that the queue length at server j is k is $\frac{G_j(N-k)}{G(N)} x_j^k$ $(1 \leq j \leq M,\ 0 \leq k \leq N)$.

**Proof**

The required probablity is equal to $\frac{1}{G(N)} \sum_{\substack{\underline{n} \in S(N) \\ n_j = k}} \prod_{i=1}^{M} x_i^{n_i} = \frac{x_j^k}{G(N)} \sum_{\underline{n} \in S_j(N\text{-}k)} \prod_{\substack{i=1 \\ i \neq j}}^{M} x_i^{n_i}$

□

**Proposition 3.2**

If server j has a fixed service rate, its equilibrium throughput is $\frac{G(N-1)}{G(N)} e_j$ $(1 \leq j \leq M)$.

**Proof**

The required throughput is equal to $\mu_j$ multiplied by the probability that the queue length at server j is non-zero, i.e. $\frac{\mu_j}{G(N)} \sum_{\substack{\underline{n} \in S(N) \\ n_j \geq 1}} \prod_{i=1}^{M} x_i^{n_i} = \frac{e_j}{G(N)} \sum_{\underline{n} \in S(N\text{-}1)} \prod_{i=1}^{M} x_i^{n_i}$

□

### 3.1.2 Model parameterisation

The packet-switched MIN is modelled as a Markovian network of stochastically identical queues with first-come-first-served queueing discipline, constant service rates and adequate buffer sizes to avoid blocking - i.e. at least as large as the population of the whole closed system. The queues are associated with the *output* pins of each switch, and logically this is where the buffers are placed. Thus, a delta network with m p×p switches in all has a total of m.p servers in its queueing model representation. We assume that the rest of the system is represented by a single additional exponential server with arbitrary fixed rate $\lambda$. In practice, there would typically be one buffer in each crossbar into which incoming messages from any input pin would be inserted as they arrived, assuming no overflow, and tagged with their required output pin number. Each output pin, when it became free after transmitting to the next crossbar, could then search the buffer for the first message addressed to it. This sharing of a single buffer delays the onset of blocking for as long as possible for a given total buffer space and is cheap to construct.

In our case the MIN is constructed from 2×2 crossbars and we assume that all inputs are utilised uniformly, i.e. have stochastically identical arrival processes. We number the pins in any stage of the network consecutively, starting at zero for the top pin. Now, the visitation rates of the output pins in the final stage are proportional to their selection probabilities which are specified. The rate for a pin in another stage is simply one half of the sum of the rates of the output pins reachable in the next stage from the said pin. These rates are determined from the previous iteration using the interconnection topology. The "half" factor arises since the inputs to each switch in stages after the first come from corresponding outputs in identical subnetworks; this is an immediate consequence of the partial shuffle topology and the uniformity of the inputs to the whole network.

In a S-stage MIN, let the visitation rate for pin number i in stage s be denoted by $e_{si}$ ($1 \le s \le S$, $0 \le i < 2^S$). Thus $\{e_{Si} \mid 0 \le i < 2^S\}$ is given and

$$e_{si} = \frac{e_{s+1,2j} + e_{s+1,2j+1}}{2} \quad \text{for } i = (k-1).2^s + j \quad \text{where } 0 \le j \le 2^s - 1,\ 1 \le k \le 2^{S-s},\ 1 \le s < S$$

The visitation rate of the other server, $e_M$ say, is of course $\sum_{i=0}^{2^S-1} e_{Si}$. We assume without loss of generality that all output pins have rate 1, i.e. the mean message transmission time between switches in adjacent stages is unity.

First, we consider a uniform network in which all outputs are selected with the same probability and so all visitation rates $e_{sj}$ are equal - to 1 say. All paths (of length S) through the MIN are then stochastically identical and so we have $M = 1 + S.2^S$, $e_i = 1$ for $1 \le i \le M-1$ and $e_M = 2^S$. We therefore have $G(N) = \sum_{k=0}^{N} \left[\frac{2^S}{\lambda}\right]^k \binom{M+N-k-1}{M-1}$ although we will not use this result, preferring the recurrence of section 3.1.1.

When there is a hot-spot, we set $e_{S0} = \rho$, the hot-spot selection probability and $e_{Si} = \frac{1-\rho}{2^S-1}$ for $1 \leq i \leq 2^S-1$. The remainder of the visitation rates are then computed successively as described above and the $e_{sj}$ are mapped onto the $e_i$ $(1 \leq i \leq M-1)$ according to some algorithm such as $e_{(s-1).2^s+j+1} = e_{sj}$ $(0 \leq j \leq 2^s-1,\ 1 \leq s \leq S)$

In this way, the embedded MIN is mapped into a conventional closed queueing network model. From this, standard resource-based measures such as throughput, utilisations and mean queue lengths, as well as mean transmission times, can be obtained using standard algorithms. So far, the structure of the physical network has been used *only* to determine the visitation rates of the servers. When we consider transmission time densities in the next section, we will find that we will need the feed-forward property of the delta network which ensures that a message cannot be overtaken by another message on any given path through it. However, the recursive structure of the partial shuffle topology is not exploited which contrasts with the analysis given for the circuit switching protocol in the next lecture, [HP90].

## 3.2 An approximate renewal model

An approach taken by Mitra and Cieslak [MC87] to solve for mean transmission times in an Omega network considers the arrival processes between stages. It is assumed that each link, i.e. switch output pin, behaves as an independent queue with arrivals from a stationary renewal process. Both interarrival time and service time distributions are arbitrary, i.e. a link is modelled by a GI/G/1 queue in Kendall's notation. The mean and coefficient of variation (i.e. the standard deviation divided by the mean) are given for the interarrival times of the external arrivals to the first stage and for switch service times. The mean and coefficient of variation of the interarrival times of the arrival processes at each stage are then computed iteratively. Standard results on GI/G/1 queues then yield the mean waiting time in each stage and hence the mean sojourn time in the whole network. The analysis uses properties derived elsewhere about the splitting and superposition of renewal processes, giving the mean and coefficient of variation of the component (respectively superposed) processes in terms of the one split (respectively its constituents).

In general, the topology of the network will determine which components of split departure processes must be combined to form an arrival process at the next stage. Thus, a convenient choice is the perfect shuffle since the routings between successive stages are all the same. Here, as in [MC87], we assume that the network is *uniformly utilised*, that is the traffic on every link between two given stages has the same renewal period distribution and that the external arrival processes to the inputs in stage one are identical. Let these arrival processes have a renewal period with mean $\lambda^{-1}$ (i.e. the arrival process have intensity $\lambda$) and coefficient of variation $c_0$, and let all link queues have the same service time distribution with mean 1 (without loss of generality) and coefficient of variation c. Thus, in equilibrium, the arrival process to every link has intensity $\lambda$ and the same applies to the departure processes. In this uniform case, the particular network topology is of no consequence, of course. The analysis now proceeds to compute the coefficients of variation of the interarrival times and interdeparture times at each stage i: $c_{ai}$ and $c_{di}$ respectively ($1 \le i \le J$).

First we have $c_{a1} = c_0$ by definition. Next, we use an approximate result of [KL76] that yields, for $1 \le i \le J$,

$$c_{di}^2 = c_{ai}^2 + 2\lambda^2c^2 - \lambda^2(c_{ai}^2+c^2)\, g(\lambda,c_{ai},c)$$

$$\text{where } g(\lambda,x,y) = \begin{cases} \exp\left\{\dfrac{-2(1-\lambda)}{3\lambda}\dfrac{(1-x)^2}{x+y}\right\} & (x \le 1) \\[2ex] \exp\left\{-(1-\lambda)\dfrac{x-1}{x+4y}\right\} & (x > 1) \end{cases}$$

We obtain $c_{ai}$ from the splitting of the departure processes from stage (i–1) into two followed by their superposition. Results from [Ku79] and [Wh83] then give, for the splitting and superposition respectively,

$$c_{ai}^2 - 1 = \frac{c_{d,i-1}^2 - 1}{2(1 + 4(1-\lambda)^2)}$$

These two equations, together with the initial condition $c_{a1} = c_0$, now give $c_{ai}$ for each stage and a result of [Ma68] gives an expression for the mean queueing time in stage i, $Q_i$ (which excludes the service time):

$$Q_i = \frac{c_{ai}^2 + 2\lambda^2c^2 - c_{di}^2}{2\lambda(1-\lambda)}$$

Mean transmission time through the network is then

$$J + \sum_{i=1}^{J} Q_i$$

This result has proved to be a good approximation for heavy traffic, but poor for light traffic when link transmission times are constant.

## 4. Analysis of transmission times

Whilst quantities such as throughput and mean queue lengths can certainly characterise well overall network behaviour, the probability *distribution* of transmission time through a MIN is also important to predict various reliability measures. These include variability in response times and the probability that network latency will exceed a given value - a quantile of the distribution. Unfortunately, time delay distributions are notoriously difficult to derive analytically whilst simulation is inefficient and yields estimates which can be unreliable, especially in the often crucial tail region. In this section we derive an explicit formula for the probability density function of transmission time through the packet-switched delta network considered in Section 3. From this we obtain all of its moments and, in particular, its variance. We first consider mean transmission time for which we have already obtained enough results.

## 4.1 Mean transmission times

A message's transmission time along a given path in a network is the sum of its sojourn times at the servers comprising that path. Therefore, *mean* transmission time is the sum of the mean sojourn times at each server. This is true whether or not the sojourn times are independent. Here, of course, we have a closed network in which the sojourn times are dependent.

Now, in the steady state, Little's result [Li61] implies that the mean sojourn time of a message at any server is the ratio of the server's mean queue length and throughput. Mean transmission time then follows from Propositions 3.1 and 3.2. However, we can exploit the fact that all service rates on a path through a regular banyan are the same, leading to a more efficient algorithm. We therefore have the following:

**Proposition 4.1**

Mean transmission time on path 1,...,m in the closed Markovian queueing network under discussion, with service rates $\mu_i=\mu$ for $1\le i\le m$, is

$$\frac{m}{\mu} + \frac{\sum_{p=0}^{N-1} p\, G_m(N-p-1)\, G_{\overline{m}}(p)}{G(N-1)\mu}$$

where for $k\ge 0$, $G_{\overline{m}}(k)$ is the normalising constant for the subnetwork comprising servers 1,...,m with population k, defined by

$$G_{\overline{m}}(k) = \sum_{\substack{\sum_{i=1}^{m} n_i=k \\ n_i\ge 0}} \prod_{i=1}^{m} \left[\frac{e_i}{\mu_i}\right]^{n_i}$$

and $G_m(k)$ is the normalising constant of the whole network with servers 1,...,m removed and population k, defined by

$$G_m(k) = \sum_{\substack{\sum_{i=m+1}^{M} n_i = k \\ n_i \geq 0}} \prod_{i=m+1}^{M} \left[\frac{e_i}{\mu_i}\right]^{n_i}$$

**Proof**

Mean transmission time is equal to the sum of the mean sojourn times at each server on the given path. But by Little's result, each mean sojourn time is the ratio of mean queue length to the server's throughput which reduces to

$$\sum_{j=1}^{m} \frac{1}{\mu_j} + \sum_{j=1}^{m} \sum_{k=0}^{N-1} \frac{G_j(N-1-k).k.x_j^k}{G(N-1)\mu_j}$$

where the servers in the path are numbered (arbitrarily) 1, ..., m and each $\mu_j=1$. It is therefore sufficient to prove that

$$\sum_{j=1}^{m} \sum_{k=0}^{n} G_j(n-k).k.x_j^k = \sum_{k=0}^{n} G_m(n-k)G_{\overline{m}}(k).k \quad \text{for all } n \geq 0$$

The left hand side of this equation is the sum of the means of the queue length random variables at servers 1,...,m. This is equal to the mean of the sum of these same random variables which is the expression on the right hand side. □

It is therefore simple to compute mean transmission times via the appropriate normalising constants. In section 4.3, we study the variation of mean transmission time along a number of paths through a delta network as the intensity of a hot-spot increases. However, to obtain the variance and higher moments, and certainly densities, requires a deeper analysis.

## 4.2 Transmission time densities

The problem of finding passage time densities in queueing networks has proved a difficult problem which can be solved in closed form only for a small class of Markovian networks; see for example [HP91]. These must possess the *non-overtaking property*, i.e. be such that no customer behind the customer being timed can influence this customer's progress through the network in any way. The delta networks considered here have a feed forward connection topology and crossbar switches with fixed transmission rates. They therefore satisfy the non-overtaking property and so we can apply the following result of [Da82]:

**Proposition 4.2**

The Laplace transform of the passage time density for the overtake-free path of servers numbered (arbitrarily) 1,...,m in a closed Markovian network of M servers with FCFS queueing disciplines and population N is

$$\frac{1}{G(N-1)} \sum_{\underline{n} \in S(N-1)} \prod_{i=1}^{M} \left(\frac{e_i}{\mu_i}\right)^{n_i} \prod_{j=1}^{m} \left\{\frac{\mu_j}{s+\mu_j}\right\}^{n_j+1}$$

□

Now, for paths in a homogeneous delta network such as ours, all the rates $\mu_i$ are the same and the Laplace transform is a mixed sum of terms of the form $\left(\frac{\mu}{s+\mu}\right)^n$ which can be inverted by inspection to give a corresponding mixture of Erlangians for the transmission time density. We therefore have, with straightforward manipulation:

**Proposition 4.3**

If the centres in overtake-free path (1,2,...,m) in the Markovian network of proposition 4.2 all have service rate $\mu$, the path's transmission time density function is

$$\frac{\mu^m e^{-\mu t}}{G(N-1)} \sum_{p=0}^{N-1} G_m(N-p-1)\, G_{\overline{m}}(p)\, \mu^p \frac{t^{p+m-1}}{(p+m-1)!}$$

From this result we can immediately obtain formulae for moments higher than the mean of transmission time. In particular, we use the first two moments to find its variance in our case study in the next section.

**Corollary 4.4**

For a path of equal rate servers, message transmission time has kth moment equal to

$$\frac{1}{\mu^k G(N-1)} \sum_{p=0}^{N-1} G_m(N-p-1)\, G_{\overline{m}}(p)\, (p+m)\ldots(p+m-k+1)$$

□

We consider numerically a 16-way network, i.e. one having 4 stages of 8 crossbars. Thus our queueing network model has 65 servers in total and we set the population to 100. Notice that this by no means represents saturation since on average each queue would contain about 1.5 messages. Numerical results were computed for hot-spot ratios in the range 1, corresponding to a uniform delta network, to 8 and for four different types of path. (The hot-spot ratio is defined as the ratio of the selection probabilities of the hot-pin and of any cold pin). The path types lead to the hot-spot, to the pin adjacent to the hot-spot, to the pin adjacent to that pin (two away from the hot-spot) and to the coldest pin (furthest from the hot-spot). The mean and standard deviation of transmission time are plotted in Figure 4.1 for hot-spot ratios 1, 1.1, 1.2, 1.5, 1.8, 2, 3, 4, 8. Transmission time densities were also plotted but are not given here.

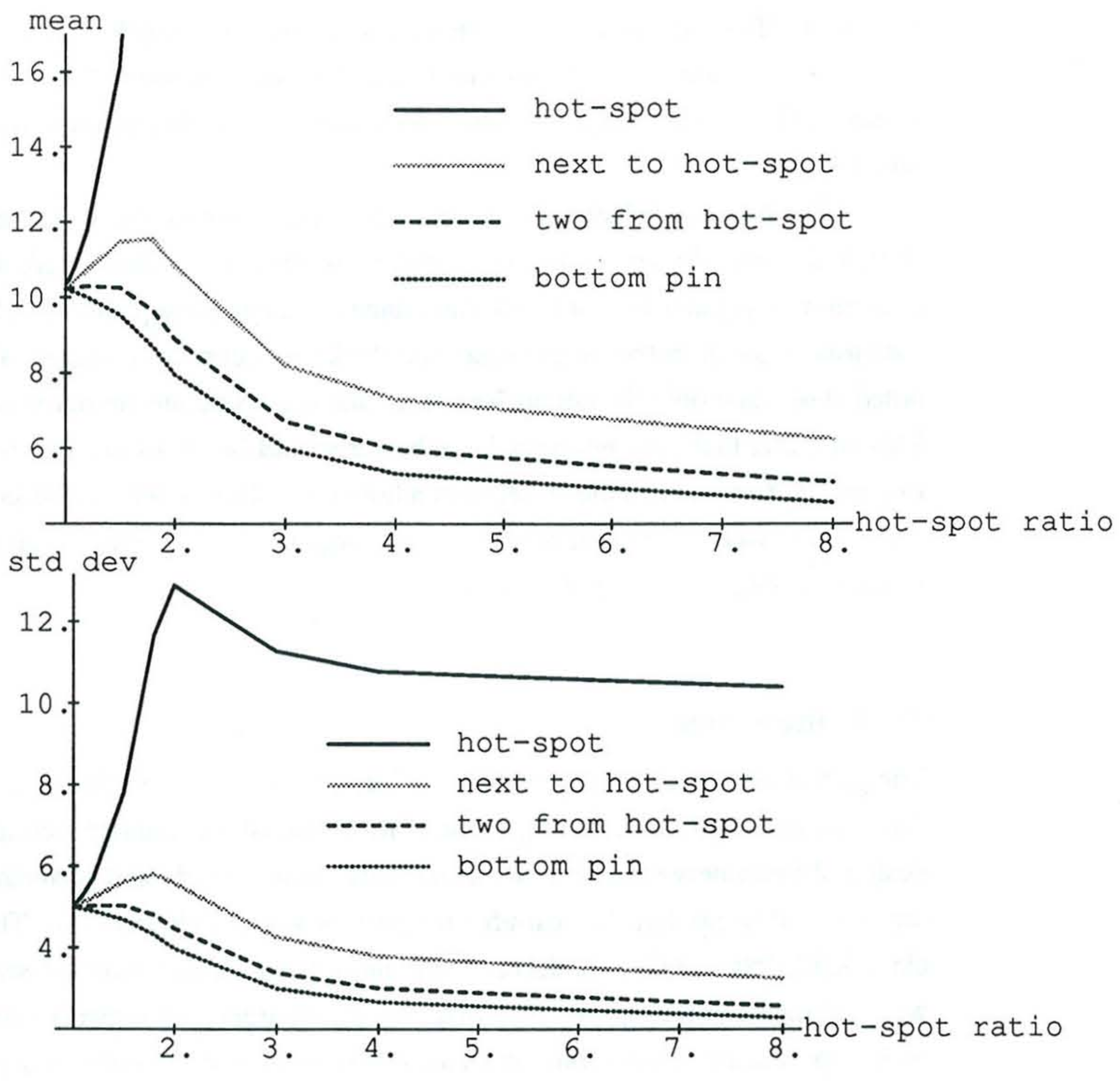


**Figure 4.1 Mean and standard deviation of transmission time**

This shows that the standard deviation on hot paths has a maximum near the ratio 2, a result which is not obvious *a priori*. The suggestion is that as the traffic in the network increases, the hot pin begins to dominate and the network's overall behaviour becomes more predictable. However, a deeper explanation follows from the observation that the hot pin's server in our model is the single *bottleneck* whenever the ratio is greater than 1. Thus for large populations, the model approaches an open system with Poisson arrivals at unit rate from the always busy hot pin to the "other server" (outside the delta network). Messages that select the hot pin in the final stage now depart the network. The interesting implication of this is that even paths to the hot-spot do not saturate until the very last stage, the hot-spot itself. This is because the arrival rates to *all* internal buffers are less than their service rates and will actually *decrease* as the hot-spot ratio increases - since more network output pins will become idle. Asymptotically, the arrival rates will approach those of the open model above and may be computed easily from its traffic equations. Notice that these arrivals are non-Poisson in view of the feedback which still exists for non-hot paths. It is the decrease in the utilisations of the internal links which eventually causes message transmission times to decrease on all non-hot paths.

The hot-spot behaviour observed here is very different from that seen in circuit-switched networks or packet switched networks with small buffers. In these, saturation propagates back from the final stage since complete paths are held throughout data transmission in the former case and blocking occurs in the latter. However, we noted above that only the hot buffer in the final stage saturates in our idealised model. This suggests that only one very large buffer would be necessary in a real system to improve performance in the presence of a hot-spot. More generally, the same would be true if there were more than one hot-spot: a large buffer need only be given to each of the equal hottest network output pins.

## 5. Conclusion

The performance provided by multi-stage interconnection networks is paramount since they constitute the central component of many parallel computer architectures. The design of these networks, as well as their cost, varies widely and performance models are essential to predict the cost-effectiveness of various alternatives. This paper has classified MINs and their models and introduced two models for packet-switched MINs with unlimited buffer space. The first, a standard queueing network model, is exact with respect to its assumptions which are rather restrictive. It provides on the one hand a standard by which to assess aproximations and has the added benefit that it can predict distributions of transmission times. The latter is an advance over previous work which has either used simulation, which can be unreliable and is expensive to run, or produced only Laplace transforms. Of more immediate use, the asymptotic analysis of section 4.3 suggested an inexpensive way of reducing the degradation introduced by hot-spots, by assigning large buffers to the hottest output pin(s) only.

Secondly, we considered a generally applicable approximate model with much milder assumptions and which can be implemented more efficiently. This has been found to give accurate predictions for a wide range of traffic patterns and distributions of link transmission times. However, it too is restricted to the case of unlimited buffer space and other methods are necessary to model MINs with finite buffers and hence blocking. Since our approach is based on standard queueing network theory, existing techniques for approximating blocking situations can be incorporated; see [Ak87], the references therein, and the Special Issue of *Performance Evaluation Journal* on blocking, 1990. The alternative protocol of circuit-switching is similar to packet-switching with zero buffer space in that whole paths through a MIN can effectively be held by a single message. However, messages can be interleaved through any switch and the dependence between switches is less strong. An alternative approach is therefore necessary and we consider the case of circuit-switching with partial paths held in the next lecture, [HP90].

## References

[Ak87] I.F. Akyildiz "General Closed Queueing Networks with Blocking", *Proc. of the 12th Annual International Symposium on Computer Performance Modelling*, December 1987, Brussels, Belgium.

[BCMP75] F. Baskett, K. M. Chandy, R.R. Muntz, F.G. Palacios, "Open, Closed and Mixed Networks of Queues with Different Classes of Customers", *Journal of the ACM* 22, 2 (1975)

[Bu73] J.P. Buzen, 'Computational algorithms for closed queueing networks with exponential servers', C. ACM 16, 9, 1973

[Da82] H. Daduna, 'Passage times for overtake-free paths in Gordon Newell networks', Adv. Appl. Prob. 14, 1982

[DJ81] D.M. Dias, J.R. Jump, "Analysis and Simulation of Buffered Delta Networks", *IEEE Trans on Computers*, Vol. C-30, No. 4, April 1981, pp. 273-282.

[FH88] A.J. Field, P.G. Harrison, *Functional Programming*, Addison-Wesley 1988.

[GN67] W.J. Gordon, G.F. Newell, 'Closed queueing systems with exponential servers', Oper. Res. 15, 254 - 265, 1967.

[Go83] A. Gottlieb, et al, "The NYU Ultracomputer – Designing an MIMD shared memory parallel computer", *IEEE Transactions on Computers*, C-32, 2 (1983), pp. 173-189.

[Ha86] P.G. Harrison, 'An enhanced approximation by pair-wise analysis of servers for time delay distributions in queueing networks', IEEE Trans. Comp. 35-1, 1986

[HF86] P.G. Harrison, A.J. Field "Performance Modelling of Parallel Computer Architectures", *Proc. of Performance '86 and ACM Sigmetrics '86 Conference*, May 1986, pp. 18-27.

[HK87] S.D. Hohl, P.J. Kuehn, 'Approximate analysis of flow and cycle times in queuing networks', in Proc. 3rd Int. Conf. on Data Communication Systems and their Performance, Rio de Janeiro, (N.Holland), 1987

[HP90] P.G. Harrison, N.M. Patel, 'The Representation of multi-stage interconnection networks in queueing models of parallel systems', accepted for publication in J. ACM.

[HP91] P.G. Harrison, N.M. Patel, 'Introduction to computer network performance modelling', *Addison-Wesley*, to appear.

[HR86] P. G. Harrison, M. J. Reeve, "The Parallel Graph Reduction Machine, ALICE", *Proc. Workshop on Graph Reduction*, Santa Fe, September 1986, to be published in LNCS series, Springer-Verlag.

[Ja63] J. Jackson, 'Jobshop-like queueing systems', Man. Sci. , 1963

[Je83] T. Jenq, "Performance of a Packet Switch Based on Single Buffered Banyan Networks", *IEEE Journal on Selected Areas in Communications*, December 1983, pp. 273-282.

[KL76] W. Kraemer, M. Langenbach-Belz, "Approximate formulae for the delay in the queueing system GI/G/1", *Proc. 8th International Teletraffic Congress*, Melbourne, Australia, 1976.

[Ku79] P.J. Kuehn, "Approximate Analysis of General Queueing Networks by Decomposition", *IEEE Transactions on Communications*, COM-27, No.1, 1979.

[Li61] J.D.C. Little, (1961), "A Proof of the Queueing Formula L = $\lambda$W", *Operations Research* 9 (3), pp. 383-387.

[Ma68] K.T. Marshall, "Some inequalities in queueing", *Oper. Res.* 16, 1968.

[MC87] D. Mitra, R. Cieslak, "Randomized Parallel Communications on an Extension of the Omega Network", *Journal of the ACM* Vol 34 No.4, October 1987, pp. 802-824.

[Pa81] J.H. Patel, "Performance of Processor-Memory Interconnections for Multiprocessors", *IEEE Transactions on Computers*, October 1981, pp. 771-780.

[Pa89] N.M. Patel, "Performance modelling of switching networks", PhD Thesis, Imperial College, University of London, 1989.

[WF80] C. Wu, T. Feng, "On a Class of Multistage Interconnection Networks", *IEEE Transactions on Computers*, Vol. C-29, No.8, August 1980, pp. 694-702.

[Wh83] W. Whitt, "The queueing network analyzer", *Bell System Technical Journal*, 62, 9, Part 1, 1983.

## DISCUSSION

**Rapporteur** : Steve Caughey

After the lecture Professor R. N. Ibbett asked about the value of simulation of this type of model. Dr. Harrison suggested that given the very large number of switching elements involved, simulation would be very difficult to achieve. He had not carried out a simulation of this particular model as the method is an exact one. However he did agree with Professor Ibbett that simulation of models using approximate methods was necessary to validate those models.

Professor B. Randell asked Dr. Harrison if he envisaged designers playing an active role in the modelling of their designs. What, he asked, did Dr. Harrison see as the designer / modeller interface ? Dr. Harrison said that the designer could use generic models if they existed but thought that changes in technology might force the designer to take decisions about the validity of the model's assumptions - a risky process for someone without a sound understanding of modelling.