SPECIAL ISSUE

7th SA COMPUTER RESEARCH SYMPOSIUM
7th Southern African Computer Research Symposium

Karos Indaba Hotel, Johannesburg
1 July 1982

PROCEEDINGS

Guest Editor: Judy M Bishop

Organised by the SA Institute of Computer Scientists
in association with the Computer Society of SA

Sponsored by Persetel and the FRD
When the first SA Computer Symposium was held at the CSIR in the early eighties, it was unique. There was no other forum at the time for the presentation of research in computer science. In the intervening decade, conferences, symposia and workshops have sprung up in response to demand, and now there are several successful ventures, some into their third or fourth iteration. Each of these addresses a specific topic - for example, hypermedia, expert systems, parallel processing or formal aspects of computing - and attracts a specialised audience, well versed in the subject and eager to learn more. For the main part, the proceedings are informal, and certainly not archival.

SACRS, though, is still unique, in that it deliberately covers a broad spectrum of research in computing, and in addition, seeks to provide a lasting record of the proceedings. To achieve the second aim, we negotiated with the SA Institute of Computer Scientists for the proceedings to form a special issue of the SA Computer Journal, and the copy you have in front of you is the result. The collaboration between the symposium committee and the journal’s editorial board placed high standards on the refereeing and final presentation of the papers, to the symposium’s benefit, while we were still able to maintain a fresh, audience-oriented approach to the selection of papers.

This is SACJ’s first such special issue, and the largest issue (at 145 pages) to date. We hope that it is only the beginning of future such collaborations.

In all 29 papers were received, all were refereed twice, and 19 were chosen for presentation by the programme committee. All the papers were thoroughly revised by the authors on the basis of the referee’s comments, and the committee’s suggestions aimed at making the material more accessible to a broadly-based audience. Papers had to be new, and not to have been presented elsewhere, a requirement that is still unusual within the SA conference round.

A third goal of SACRS has been to invite keynote speakers, usually from overseas. This year, we are fortunate to present Dr Vinton Cerf, the father of the Internet and a world-renown expert on computer networks. Although his paper is not available for this special issue, it will appear later in SACJ. Through the good offices of Professor Chris Brink of UCT, we also have three other speakers from Germany, Canada and the US adding interest to the event, and two of their papers appear in this issue.

The programme committee originally devised a theme for the symposium - "Computing in the New South Africa". We received several queries as to the meaning of this theme, but unfortunately few papers that addressed it directly. One prospective author went as far as to enquire whether computer research would survive in the new South Africa. Another felt that his work was definitely not in the theme, as it was genuine, old world, basic, theoretical science! Nevertheless, there are two papers that consider one of South Africa’s key issues, that of language. Others look at the success we have achieved in applying technology to mining, and the future of low-cost operating systems. In all, the mix of papers represents a balance between the theoretical and the practical, the past and the future, all firmly based in the computing of the present.

Organising the symposium has involved the hard work of several people, and I would like to thank in particular

• Derrick Kourie, my co-organiser, and the editor of SACJ for his invaluable advice and hard work throughout the planning and implementation stages;
• Riel Smit, the production editor, for attaining such a high standard in such a short time for so many papers;
• Gerrit Prinsloo and the staff at the CSSA for their efficient and quite delightfully unfussy organisation;
• Persetel for their very generous sponsorship of R25000, and Tim Schumann for taking a genuine interest in our events;
• the Foundation for Research Development for sponsoring Vint Cerf’s visit;
• and finally the Department of Computer Science of the University of Pretoria for providing the ideal working conditions for undertaking ventures of this kind, and especially Roelf van den Heever for his unfailing encouragement and support.

Judy M Bishop
Organising Chairman, SACRS 1992
Guest Editor, SACJ Special Issue
Referees

The journal draws on a wide range of referees. The following were involved in the refereeing of the papers selected for this special issue. Their role in certifying the papers and their contribution to enhancing the quality of papers is sincerely appreciated.

John Barrow  
Ronnie Becker  
Danie Behr  
Sonia Berman  
Liesbeth Botha  
Theo Bothma  
Chris Brink  
Peter Clayton  
Ian Cloete  
Antony Cooper  
Elise Ehlers  
Quintin Gee  
Andy Gravell  
Wendy Hall  
David Harel  
Scott Hazelhurst  
Derrick Kourie  
Willems Labuschagne  
Doug Laing  
Dave Levy  
Graham McLeod  
Hans Messerschmidt  
Deon Oosthuizen  
Ben Oosthuysen  
Neil Pendock  
D Petkov  
Martin Rennhackkamp  
Cees Roon  
Jan Roos  
John Shochot  
Morris Sloman  
Riel Smit  
Pat Terry  
Walter van den Heever  
Lynn van der Vegte  
Herman Venter  
Herna Viktor  

UNISA  
University of Cape Town  
University of Pretoria  
University of Cape Town  
University of Pretoria  
University of Pretoria  
University of Cape Town  
Rhodes University  
Stellenbosch University  
CSIR  
Rand Afrikaanse University  
DALSIG  
University of Southhampton  
University of Southhampton  
The Weizmann Institute of Science  
University of the Witwatersrand  
University of Pretoria  
UNISA  
ISM  
University of Natal  
University of Cape Town  
University of the Orange Free State  
University of Pretoria  
University of Pretoria  
University of the Witwatersrand  
University of Natal  
Stellenbosch University  
University of Pretoria  
University of Pretoria  
University of the Witwatersrand  
Imperial College, London  
University of Cape Town  
Rhodes University  
University of Pretoria  
University of Pretoria  
University of Fort Hare  
Stellenbosch University
The Multiserver Station with Dynamic Concurrency Constraints

Carel F Kriel*  Anthony E Krzesinski†

*Bureau for Systems Engineering, University of Stellenbosch, Stellenbosch 7600
†Institute for Applied Computer Science, University of Stellenbosch, Stellenbosch 7600

Abstract

This paper defines the Multiserver Station with Dynamic Concurrency Constraints. The MSDCC station consists of $B$ parallel identical exponential servers. The customers requesting service at the MSDCC station belong to $C$ types. Customers arriving at the MSDCC station are queued for service in FCFS order. A set of tokens ($B_1 \ldots B_C$) is associated with the MSDCC station where $B_c$ is the number of type $c$ tokens. The tokens are used to enforce a system of dynamic concurrency constraints which limits the number of customers of each type that may simultaneously be in service. The concurrency constraints operate as follows. A type $c$ customer may only enter into service if a server is free and a type $c$ token is available. When entering into service a type $c$ customer siezes a type $c$ token. The token is retained while the customer is in service. Upon completing service the customer releases the type $c$ token which, according to a Bernoulli trial, changes into a type $d$ token. This paper defines the MSDCC station and proves that the MSDCC station has a product form solution. Several examples of state dependent token switching functions are given. Finally, an efficient recursive expression is derived for the queue length distribution at the MSDCC station.

Keywords: Blocking, Concurrency Constraints, Product Form Solutions, Queueing Networks, Queueing Theory.

Computing Review Categories: C.4, D.4.8

Received March 1992, Accepted April 1992

1 Introduction

Queueing networks are often used as performance models of computer systems because they accurately describe how the performance of these systems is affected by queuing for access to resources. Product form networks are particularly useful as computer performance models due to the availability of efficient computational algorithms for their solution. Product form networks are based on a series of results, amongst which the BCMP [1] theorem, which gives sufficient conditions on the customer scheduling, service and routing processes at the network service stations so that the network steady state probability has an explicit product form. The BCMP theorem requires that the service stations in a product form network belong to one of the following four types: First Come First Served; Last Come First Served; Processor Sharing; Infinite Server.

Several investigations [5, 9] of multiple bus multiprocessor architectures suggested extensions to the BCMP theorem. A striking fact was that the local balance equation held for these multiple bus models, even though the models contained service stations that did not belong to the famous catalogue of BCMP service stations. Local balance is the key property that allows product form. The references cited above provided no analytical proofs that the local balance equation held for the multiple bus models under investigation. Only numerical proofs were presented, except for very specific simple cases. Some time later, the concept of the MSCCC station [7] was developed which placed these models in a general analytical framework.

The MSCCC station (Multiserver Station with Concurrent Classes of Customers) differs from the classical BCMP First Come First Served service station in that it includes a supplementary blocking. The blocking works as follows. Each customer has a type membership. Maximally one customer from each type can be in service at the MSCCC station. The major interest of the MSCCC station is that (1) it is an additional, unexpected member of the catalogue of BCMP stations that are the building blocks of product form networks, (2) the MULTIBUS algorithm [8] was developed to compute efficient solutions for closed Multiclass Queueing Networks (MQNs) containing MSCCC stations, and (3) the MSCCC station had an immediate application in the modelling of multiple bus multiprocessor architectures.

Several extensions to the MSCCC station have recently been reported. The bound on the number of type $k$ customers that could be served simultaneously was extended from one to $n$ [2] to $n_k$ [3, 4]. A generalized blocking scheme was reported in [6] which imposed restrictions on the numbers of customers of different types and sub-types that can be served simultaneously. The extended MSCCC station has been applied to model window flow control [4] and polling systems [10].

This paper presents an extension of the MSCCC station which we have named the MSDCC station (Multiserver Station with Dynamic Concurrency Constraints). The main results of this paper are (1) a more general definition for the MSDCC station is given where the concurrency bounds $n_k$ are random variables, (2) a product form solution for the MSDCC station is derived, and (3) efficient recursive expressions are derived for the joint probability distribution at the MSDCC station.

Some of the mathematical derivations related to the MSDCC analysis are complex. In order to keep the paper...
as readable as possible, only those derivations and proofs that are new are presented in the course of the text. Other proofs which are based upon extensions to previous results are not described here.

The paper is organized as follows: Section 2 defines the MSDCC station. Section 3 presents the global balance equations for the MSDCC station. Section 4 partitions the global balance equations into partial balance equations and derives a product form solution for the partial balance equations. Section 5 computes the joint distribution of the concurrency bounds. Section 6 reduces the joint probability distribution at the MSDCC station to a computationally tractable form.

2 The MSDCC Station

Consider a service station serving customers of type \( c \) where \( c \in C \) and \( C \) is a finite set. Customers of type \( c \) arrive individually in independent Poisson streams with rate \( \lambda_c \) such that \( \sum_{c \in C} \lambda_c < \infty \). The customers, whether waiting or in service, form a queue in the order of their arrival. Thus arriving customers join the tail of the queue and the front of the queue is identified with position 1. Each customer presents a demand for service time which is exponentially distributed with mean \( 1/\mu \). The server gives service at certain positions of the queue. Service at each position served is non-interruptible and takes place at the positive rate \( \mu_n > 0 \) where \( n \) is the length of the queue. Upon entering service a customer is served to completion without interruption, whereupon the customer departs. When a service completion occurs, the corresponding customer departs, the gap in the queue is closed by the obvious shift, and the server scans the queue from the front searching for the first customer whose admission into service would not violate the following set of concurrency constraints. Firstly, maximally \( B \) customers can be served simultaneously where \( B \in \mathbb{N} \). The queue can thus be viewed as having \( B \) parallel servers. Next, maximally \( B \) type \( c \) customers are allowed to be in service simultaneously where \( c \in C \) and \( B_c \in \mathbb{N} \). The queue discipline can thus be described as FCFS subject to concurrency constraints.

Dynamic Concurrency Constraints

We associate a token pool with the MSDCC station as follows. Let \( \vec{B} = (B_c)_{c \in C} \) denote the state of the token pool where \( B_c \in \mathbb{N} \) is the number of type \( c \) tokens in the token pool. We require that there must always be at least one token of each type in the token pool, otherwise queue and token pool states can arise where all servers are idle and no enqueued customers can be served. Thus \( B_c > 0 \) for all \( c \in C \) so that the customer at the head of the queue is always in service.

Let \( \vec{C} = (c_n \ldots c_1) \) denote the state of a queue of length \( n \) where \( c_i \) denotes the type of the customer in position \( i \) where \( 1 \leq i \leq n, n \in \mathbb{N} \) and \( c_i \in C \). Let the token pool state \( \vec{B} + e - d \) be obtained from \( \vec{B} \) by substituting \( B_i + 1 \) for \( B_i \) and \( B_d - 1 \) for \( B_d \), the latter only being defined for \( B_d > 0 \). Consider an MSDCC station in the state \((\vec{C}, \vec{B} + e - d)\). The token pool is used to enforce a system of dynamic concurrency constraints which limits the number of customers of each type that may simultaneously be in service. The concurrency constraints operate as follows. Each token is in one of two states: the token is either available or it is in-use. A type \( c \) customer may only enter into service if a server is free and a type \( c \) token is available. When entering into service the type \( c \) customer (the tagged customer) siezes a type \( c \) token. The token changes its state from available to in-use. The token remains in-use while the tagged customer is in service. Upon completing service the tagged customer releases the type \( c \) token which, with probability \( q_{cd}(\vec{B} + e - d) \), changes into a type \( d \) token. The token changes its state from in-use to available and the state of the token pool becomes \( \vec{B} \). The token pool therefore has a state space

\[ B = \{ \vec{B} \mid B_c > 0 \text{ for all } c \in C \text{ and } |\vec{B}| = K \} \]

where \( K \geq C \) is the total number of tokens and \( C = |C| \) is the number of token (customer) types.

If \( \vec{B} \in B \) such that \( B_d > 1 \) for all \( d \in C \) then \( \vec{B} + e - d \in B \) for all \( c \in C \). However if \( B_d = 1 \) for any \( d \in C \) then \( \vec{B} + e - d \in B \) if and only if \( c = d \). Consider \( \vec{B} \in B \) where \( B_d = 1 \) for any \( d \in C \). Define

\[ q_{cd}(\vec{B} + e - d) = 1(c = d) \]

where \( 1 : \mathcal{P} \rightarrow \{0, 1\} \) such that

\[ 1(p) = \begin{cases} 1 & \text{if } p \text{ is true} \\ 0 & \text{if } p \text{ is false} \end{cases} \]

where \( \mathcal{P} \) denotes the set of all predicates. Clearly \( 0 \leq q_{cd}(\vec{B}) \leq 1 \) for any \( c, d \in C \) and \( \sum_{c \in C} q_{cd}(\vec{B}) = 1 \) for any \( \vec{B} \in B \).

3 The Global Balance Equations

Consider a queue with state space \( W = \{0\} \cup Q \) where 0 denotes the empty queue and \( Q = \bigcup_{n=1}^{\infty} C^n \). Consider the combined system consisting of the queue and the token pool with state space \( S = W \times B \). For \( 1 \leq i \leq n \) define the set of functions \( 1_{i,n} : Q \times B \rightarrow \{0, 1\} \) such that

\[ 1_{i,n}(c_n \ldots c_1, \vec{B}) = \begin{cases} 1 & \text{if the customer in queue position } i \text{ is in service} \\ 0 & \text{if the customer in queue position } i \text{ is not in service} \end{cases} \]

Next define the function \( k : Q \times B \rightarrow \mathbb{N} \) such that

\[ k(c_n \ldots c_1, \vec{B}) = \sum_{i=1}^{n} 1_{i,n}(c_n \ldots c_1, \vec{B}) \geq 1 \]

so that \( k(c_n \ldots c_1, \vec{B}) \) is the number of customers being served in \( (c_n \ldots c_1, \vec{B}) \).

Let \( P(C, \vec{B}) \) denote the probability that the MSDCC station is in state \((\vec{C}, \vec{B}) \in S \). Define \( P(\vec{C}, \vec{B}) = 0 \)
if \((\tilde{C}, \tilde{B}) \notin S\). Consider first the empty queue. For the corresponding Markov process, which is obviously regular, the global balance equations are given by

\[
P(O, \tilde{B})\lambda = \mu_1 \sum_{c \in C} \sum_{d \in C} P(c, \tilde{B} + c - d) \times q_{cd}(\tilde{B} + c - d)
\]

(1)

where \(\lambda = \sum_{c \in C} \lambda_c\).

Consider next the nonempty queue. The global balance equations are given by

\[
P(c_n \cdot \cdot \cdot c_1, \tilde{B}) \left(\lambda + k(c_n \cdot \cdot \cdot c_1, \tilde{B})\mu_n\right) = \lambda_c n_{c_n} P(c_n-1 \cdot \cdot \cdot c_1, \tilde{B} + c - d) \times 1_{n+1} + \mu_n L L P(c_n \cdot \cdot \cdot c_1, \tilde{B} + c - d) \times q_{cd}(\tilde{B} + c - d)
\]

(2)

where the term \(P(c_n-1 \cdot \cdot \cdot c_1, \tilde{B})\) is replaced by \(P(O, \tilde{B})\) if \(n = 1\).

We rewrite the global balance equations (1) and (2) into a form where it can readily be demonstrated that the rewritten global balance equations have a product form solution. Define

\[X = \{(X_c)_{c \in C} | \exists d \in C \ni \tilde{X} + d \in B\}\]

One component of \(\tilde{X}\) may thus be zero. If one component of \(\tilde{X}\) is zero then \(\tilde{X}\) is not a valid token pool state. However, if all the components of \(\tilde{X}\) are positive then \(\tilde{X}\) is a valid token pool state for a MSDCC station with a total of \(K - 1\) tokens.

Let \(P(\tilde{C}, \tilde{X} + d)\) denote the probability that the MSDCC station is in state \((\tilde{C}, \tilde{X} + d)\) where \(d \in C\) and \((\tilde{C}, \tilde{X} + d) \in S\). Define \(P(\tilde{C}, \tilde{X} + d) = 0\) if \((\tilde{C}, \tilde{X} + d) \notin S\).

From equation (1)

\[
\sum_{d \in C} \sum_{\tilde{B} \in B} P(0, \tilde{B})\lambda_d = \mu_1 \sum_{d \in C} \sum_{\tilde{B} \in B} \sum_{c \in C} P(c, \tilde{B} + c - d) \times q_{cd}(\tilde{B} + c - d)
\]

Now

\[
\sum_{d \in C} \sum_{\tilde{B} \in B} P(0, \tilde{B})\lambda_d = \sum_{\tilde{B} \in B} \sum_{d \in C} P(0, \tilde{B} + d)\lambda_d
\]

and

\[
\sum_{d \in C} \sum_{\tilde{B} \in B} \sum_{c \in C} P(c, \tilde{B} + c - d) q_{cd}(\tilde{B} + c - d) = \sum_{d \in C} \sum_{\tilde{B} \in B} \sum_{c \in C} P(c, \tilde{X} + c) q_{cd}(\tilde{X} + c) = \sum_{c \in C} P(c, \tilde{X} + c)
\]

The global balance equations for the empty queue thus become

\[
\sum_{c \in C} P(0, \tilde{X} + c)\lambda_c = \mu_1 \sum_{c \in C} P(c, \tilde{X} + c)
\]

(3)

which is true for all values of \(\lambda_c\) and hence

\[
P(0, \tilde{B})\lambda_c = \mu_1 P(c, \tilde{B})
\]

(4)

for any \(c \in C\) and \(\tilde{B} \in B\). Note that equation (4) also applies to the MSCCC station where tokens cannot change their type. However, equation (3) only applies to MSDCC stations with token type changing.

Similarly the global balance equations for the nonempty queue become

\[
\sum_{c \in C} P(c_n \cdot \cdot \cdot c_1, \tilde{X} + c) \left(\lambda_c + \mu_n k(c_n \cdot \cdot \cdot c_1, \tilde{X} + c)\right)
\]

\[
= \lambda_c n_{c_n} P(c_n-1 \cdot \cdot \cdot c_1, \tilde{X} + c) + \mu_n+1 \sum_{i=0}^{n} \sum_{c \in C} P(c_n \cdot \cdot \cdot c_{i+1} \cdot c_1 \cdot \cdot \cdot c_1, \tilde{B} + c - d) \times 1_{n+1} + \mu_n L L P(c_n-1 \cdot \cdot \cdot c_1, \tilde{B} + c - d) \times q_{cd}(\tilde{B} + c - d)
\]

(5)

4 Product Form Solution

We wish to determine solutions for equation (5) which also satisfy the partial balance equations

\[
\mu_n \sum_{c \in C} P(c_n \cdot \cdot \cdot c_1, \tilde{X} + c) k(c_n \cdot \cdot \cdot c_1, \tilde{X} + c) = \lambda_c n_{c_n} \sum_{c \in C} P(c_n-1 \cdot \cdot \cdot c_1, \tilde{X} + c)
\]

(6)

Equation (6) has a solution

\[
P(c_n \cdot \cdot \cdot c_1, \tilde{B}) = P(0, \tilde{B}) \prod_{i=1}^{n} \frac{\lambda_c}{\mu_i k(c_i \cdot \cdot \cdot c_1, \tilde{B})}
\]

(7)

which can be verified by substituting equation (7) into equation (6).

In order for equation (7) to satisfy equation (5) it is necessary and sufficient that equation (7) satisfies the partial
balance equations
\[
\sum_{c \in C} P(c_1 \cdots c_t, X + c) \lambda_c \\
= \mu_{n+1} \sum_{i=0}^{n} P(c_1 \cdots c_i \cdot c_{i+1} \cdot c_{i+1} \cdots c_1, X + c) \\
\times 1_{i+1,n+1}(c_1 \cdots c_{i+1} \cdot c_{i+1} \cdots c_1, X + c) \quad (8)
\]
The above equation is true for all values of \( \lambda_c \). It is therefore necessary and sufficient that equation (7) satisfies the partial balance equations
\[
P(c_1 \cdots c_t, \bar{B}) \lambda_c \\
= \mu_{n+1} \sum_{i=0}^{n} P(c_1 \cdots c_i \cdot c_{i+1} \cdot c_{i+1} \cdots c_1, \bar{B}) \\
\times 1_{i+1,i+1}(c_{i+1} \cdots c_{i+1} \cdot c_{i+1} \cdots c_1, \bar{B}) \quad (9)
\]
for any \( c \in C \) and \( \bar{B} \in B \). Again note that whereas equation (9) also applies to the MSCCC station where tokens cannot change their type, equation (8) only applies to MS-DCC stations with token type changing.

REMARK 1 Note that for any \((\bar{C}, \bar{B}) \in S\) and any \( 1 \leq i \leq n \)
\[
i_{i,n}(c_1 \cdots c_t, \bar{B}) = i_{i,i}(c_1 \cdots c_t, \bar{B})
\]
which implies that whether queue position \( i \) is being served or not when the station is in the state \((c_1 \cdots c_t, \bar{B})\) depends solely on the composition of the queue up to position \( i \).

THEOREM 1 For the MS-DCC station
\[
k(c_1 \cdots c_t, \bar{B}) = k(c_{\sigma(n)} \cdots c_{\sigma(1)}, \bar{B})
\]
where \( \sigma \) denotes any permutation of \((1 \ldots n), (c_1 \cdots c_t) \in Q \) and \( \bar{B} \in B \). The total number of customers served in the queue thus remains unchanged under a permutation of the customers in the queue.

PROOF. This is a direct consequence of the MS-DCC service discipline. \( \square \)

THEOREM 2 Equation (7) is a solution to equation (9). PROOF. The proof is by induction. For any \((\bar{C}, \bar{B}) \in S\) and \( c \in C \) define
\[
\lambda_c P(c_1 \cdots c_t, \bar{B}) \\
= \mu_{n+1} \sum_{i=0}^{n} P(c_1 \cdots c_i \cdot c_{i+1} \cdot c_{i+1} \cdots c_1, \bar{B}) \\
\times 1_{i+1,i+1}(c_{i+1} \cdots c_{i+1} \cdot c_{i+1} \cdots c_1, \bar{B})
\]
For the empty queue
\[
\lambda_c P(0, \bar{B}) = \mu_1 P(c, \bar{B})
\]
From equation (7)
\[
\mu_1 P(c, \bar{B}) = \lambda_c P(0, \bar{B})
\]
so that \( P(0, \bar{B}) = P(0, \bar{B}) \). Equation (7) therefore satisfies the partial balance equations (9) in the case of the empty queue.

Next assume that equation (7) satisfies equation (9) for \( n-1 \geq 0 \) so that
\[
\lambda_c F(c_{n-1} \cdots c_t, \bar{B}) \\
= \mu_{n+1} \sum_{i=0}^{n-1} P(c_{n-1} \cdots c_i \cdot c_{i+1} \cdot c_{i+1} \cdots c_t, \bar{B}) \\
\times 1_{i+1,i+1}(c_{i+1} \cdots c_{i+1} \cdot c_{i+1} \cdots c_t, \bar{B}) \\
= \lambda_c P(c_{n-1} \cdots c_t, \bar{B}) \quad (10)
\]
Let \( \bar{C} = (c_{n-1} \cdots c_t) \). Now
\[
\lambda_c F(\bar{C}, \bar{B}) \\
= \mu_{n+1} \sum_{i=0}^{n-1} P(c_{n-1} \cdots c_i \cdot c_{i+1} \cdot c_{i+1} \cdots c_t, \bar{B}) \\
\times 1_{i+1,i+1}(c_{i+1} \cdots c_{i+1} \cdot c_{i+1} \cdots c_t, \bar{B}) \\
+ \mu_{n+1} P(\bar{C}, \bar{B}) 1_{n+1,n+1}(\bar{C}, \bar{B})
\]
We first apply equation (7) to the above equation and then apply theorem 1 which established the invariance of \( k(\bar{C}) \) under a permutation of the customers in the queue, yielding
\[
\lambda_c F(\bar{C}, \bar{B}) k(c_{n-1} \cdots c_t, \bar{B}) \\
= \lambda_c \sum_{i=0}^{n-1} P(c_{n-1} \cdots c_i \cdot c_{i+1} \cdot c_{i+1} \cdots c_t, \bar{B}) \\
\times 1_{i+1,i+1}(c_{i+1} \cdots c_{i+1} \cdot c_{i+1} \cdots c_t, \bar{B}) \\
+ \lambda_c P(\bar{C}, \bar{B}) 1_{n+1,n+1}(\bar{C}, \bar{B})
\]
Application of the inductive assumption equation (10) yields
\[
F(\bar{C}, \bar{B}) k(c_{n-1} \cdots c_t, \bar{B}) \\
= \lambda_c \sum_{i=0}^{n-1} P(c_{n-1} \cdots c_i \cdot c_{i+1} \cdot c_{i+1} \cdots c_t, \bar{B}) \\
\times 1_{i+1,i+1}(c_{i+1} \cdots c_{i+1} \cdot c_{i+1} \cdots c_t, \bar{B}) \\
+ \lambda_c P(\bar{C}, \bar{B}) 1_{n+1,n+1}(c_{n-1} \cdots c_t, \bar{B})
\]
From the first partial balance equation
\[
\lambda_c P(c_{n-1} \cdots c_t, \bar{B}) = \mu_n P(\bar{C}, \bar{B}) k(\bar{C}, \bar{B})
\]
so that
\[
F(\bar{C}, \bar{B}) k(c_{n-1} \cdots c_t, \bar{B}) \\
= P(\bar{C}, \bar{B}) \left( k(\bar{C}, \bar{B}) + 1_{n+1,n+1}(c_{n-1} \cdots c_t, \bar{B}) \right)
\]
and
\[
k(c_{n-1} \cdots c_t, \bar{B}) = \begin{cases} k(\bar{C}, \bar{B}) & \text{if } 1_{n+1,n+1}(c_{n-1} \cdots c_t, \bar{B}) = 0 \\ k(\bar{C}, \bar{B}) + 1 & \text{if } 1_{n+1,n+1}(c_{n-1} \cdots c_t, \bar{B}) = 1 \end{cases}
\]
which completes the proof of theorem 2. \( \square \)
5 Distribution of the Token Pool

The distribution \( P(\bar{B}) \) of the token pool is defined as

\[
P(\bar{B}) = \sum_{\bar{C} \in W} P(\bar{C}, \bar{B})
\]

**Lemma 1** For any \( d \in C \) and \((\bar{C}, \bar{B}) \in S\)

\[
\lambda_d P(\bar{C}, \bar{B}) = \sum_{c \in C} \lambda_c P(\bar{C}, \bar{B} + c - d) \times q_{ed}(\bar{B} + c - d)
\]

(11)

**Proof.** If \( B_d = 1 \) then \( c = d \) and \( q_{ed}(\bar{B} + c - d) = 1 \) so that equation (11) is identically true.

If \( B_d > 1 \) then there exists an \( \bar{X} \in X \) and \( \bar{B} \in B \) such that \( \bar{X} = \bar{B} - d \). Now consider

\[
\sum_{\bar{C} \in C} \lambda_d P(\bar{C}, \bar{X} + d)
\]

\[
= \sum_{\bar{C} \in C} \sum_{\bar{X} \in X} \lambda_c P(\bar{C}, \bar{X} + c) \times q_{ed}(\bar{X} + c)
\]

(12)

Let \( c \bar{C} = (cc_2 \ldots c) \). From equation (7)

\[
\mu_{n+1} k(c\bar{C}, \bar{X} + c) P(c\bar{C}, \bar{X} + c) = \lambda_c P(c\bar{C}, \bar{X} + c)
\]

is a solution to equation (12). Therefore

\[
\sum_{\bar{C} \in C} \lambda_d P(\bar{C}, \bar{X} + d)
\]

\[
= \sum_{\bar{C} \in C} \mu_{n+1} k(c\bar{C}, \bar{X} + c) P(c\bar{C}, \bar{X} + c) \times q_{ed}(\bar{X} + c)
\]

which is true for all values of \( \lambda_d \) so that

\[
\lambda_d P(\bar{C}, \bar{X} + d)
\]

\[
= \sum_{c \in C} \mu_{n+1} k(c\bar{C}, \bar{X} + c) P(c\bar{C}, \bar{X} + c) q_{ed}(\bar{X} + c)
\]

Substituting \( \bar{X} = \bar{B} - d \) into the above equation completes the proof of lemma 1. \( \square \)

From equation (11) for any \( d \in C \) and \( \bar{B} \in B \)

\[
\lambda_d P(\bar{B}) = \sum_{c \in C} \lambda_c P(\bar{B} + c - d) q_{ed}(\bar{B} + c - d).
\]

(13)

Equation (13) and the closure condition \( \sum_{\bar{B} \in B} P(\bar{B}) = 1 \) define a set of \( |B| + 1 \) linear equations which determine the values of \( P(\bar{B}) \). However, \( |B| = \binom{K-1}{C-1} \) so that equation (13) is only useful for computing values for \( P(\bar{B}) \) for small token pools.

6 Aggregation

The steady state distribution \( P(\bar{C}, \bar{B}) \) is too detailed to be of practical use when computing the performance measures of the MSDCC station. Aggregation is therefore used to reduce \( P(\bar{C}, \bar{B}) \) to a computationally feasible form. Theorem (1) is the key for obtaining results concerning aggregated states. Define

\[
Q_m = \{ \bar{C} \in W : m_\bar{C}(\bar{C}) = m, c \in C \}
\]

where \( m_\bar{C}(\bar{C}) \) is the number of type \( c \) customers present when the queue is in state \( \bar{C} \) and \( m = (m_c)_{c \in C} \) where \( m_c \geq 0 \).

Theorem 1 allows us to introduce an abuse of notation for the MSDCC station, namely for \( \bar{C} \in Q_m \) to write \( k(\bar{m}, \bar{B}) = k(\bar{C}, \bar{B}) \) to denote the total number of customers served in a state from \( (Q_m, \bar{B}) \).

**Theorem 3** For the MSDCC station

\[
k(\bar{m}, \bar{B}) \mu_{|\bar{m}|} P(Q_m, \bar{B}) = \sum_{c \in C} \lambda_c P(Q_{m-c}, \bar{B})
\]

(14)

where \( \bar{m} - c \) is obtained from \( \bar{m} \) by substituting \( m_c - 1 \) for \( m_c \), this being defined only for \( m_c > 0 \).

**Proof.** The proof is given in [6].
Equation (14) permits the recursive calculation of \( P(Q_n, \bar{B}) \). However, simpler recursions are available in the domain where \( k(\bar{n}, \bar{B}) < B \). These recursions require lemma 2. First define

\[
k_c(c_n \cdots c_1, \bar{B}) = \sum_{i=1}^{n} i_{i}(c_i \cdots c_1, \bar{B}) \ 1(c_i = c)
\]

for \( c \in C \) so that \( k_c(\bar{C}, \bar{B}) \) is the number of type \( c \) customers in service when the station is in the state \( (\bar{C}, \bar{B}) \).

**Lemma 2** For \( (\bar{C}, \bar{B}) \in S \) such that \( k(\bar{C}, \bar{B}) < B \)

\[
k_c(\bar{C}, \bar{B}) = m_c \land B_c \quad (15)
\]

for any \( c \in C \) where \( p \land q \) is the smaller of the two integers \( p \) and \( q \).

**Proof.** This is a direct consequence of the MSDCC service discipline.

Lemma 2 implies that for \( \bar{n} \) and \( \bar{B} \) such that \( k(\bar{n}, \bar{B}) < B \) we can allow a further abuse of notation for the MSDCC station, namely for \( \bar{C} \in Q_n \) to write \( k_c(m_c, B_c) = k_c(\bar{C}, \bar{B}) \) to denote the number of type \( c \) customers served in a state from \( Q_n \).

**Lemma 3** For \( m \) such that \( k(m, B) < B \)

\[
k_c(m_c, B_c) \mu_{|m|} P(Q_n, \bar{B}) = \lambda_c P(Q_{n-1}, \bar{B}) \quad (16)
\]

for all \( c \in C \) such that \( m_c > 0 \).

**Proof.** The proof is given in [6].
Notes for Contributors

The prime purpose of the journal is to publish original research papers in the fields of Computer Science and Information Systems, as well as shorter technical research papers. However, non-refereed review and exploratory articles of interest to the journal’s readers will be considered for publication under sections marked as Communications or Viewpoints. While English is the preferred language of the journal, papers in Afrikaans will also be accepted. Typed manuscripts for review should be submitted in triplicate to the editor.

Form of Manuscript

Manuscripts for review should be prepared according to the following guidelines:

- Use wide margins and 1 ½ or double spacing.
- The first page should include:
  - title (as brief as possible);
  - author’s initials and surname;
  - author’s affiliation and address;
  - an abstract of less than 200 words;
  - an appropriate keyword list;
  - a list of relevant Computing Review Categories.
- Tables and figures should be numbered and titled. Figures should be submitted as original line drawings/printouts, and not photocopies.
- References should be listed at the end of the text in alphabetic order of the (first) author’s surname, and should be cited in the text in square brackets [1, 2, 3]. References should take the form shown at the end of these notes.

Manuscripts accepted for publication should comply with the above guidelines (except for the spacing requirements), and may be provided in one of the following formats (listed in order of preference):

1. As (a) \LaTeX file(s), either on a diskette, or via e-mail/ftp – a \LaTeX style file is available from the production editor;
2. In camera-ready format – a detailed page specification is available from the production editor;
3. As an ASCII file accompanied by a hard-copy showing formatting intentions:
   - Tables and figures should be on separate sheets of paper, clearly numbered on the back and ready for cutting and pasting. Figure titles should appear in the text where the figures are to be placed.
   - Mathematical and other symbols may be either handwritten or typed. Greek letters and unusual symbols should be identified in the margin, if they are not clear in the text.

Further instructions on how to reduce page charges can be obtained from the production editor.

4. In a typed form, suitable for scanning.

Charges

Charges per final page will be levied on papers accepted for publication. They will be scaled to reflect scanning, typesetting, reproduction and other costs. Currently, the minimum rate is R20-00 per final page for \LaTeX or camera-ready contributions and the maximum is R100-00 per page for contributions in typed format.

These charges may be waived upon request of the author and at the discretion of the editor.

Proofs

Proofs of accepted papers in categories 3 and 4 above will be sent to the author to ensure that typesetting is correct, and not for addition of new material or major amendments to the text. Corrected proofs should be returned to the production editor within three days.

Note that, in the case of camera-ready submissions, it is the author’s responsibility to ensure that such submissions are error-free. However, the editor may recommend minor typesetting changes to be made before publication.

Letters and Communications

Letters to the editor are welcomed. They should be signed, and should be limited to less than about 500 words.

Announcements and communications of interest to the readership will be considered for publication in a separate section of the journal. Communications may also reflect minor research contributions. However, such communications will not be refereed and will not be deemed as fully-fledged publications for state subsidy purposes.

Book reviews

Contributions in this regard will be welcomed. Views and opinions expressed in such reviews should, however, be regarded as those of the reviewer alone.

Advertisement

Placement of advertisements at R1000-00 per full page per issue and R500-00 per half page per issue will be considered. These charges exclude specialized production costs which will be borne by the advertiser. Enquiries should be directed to the editor.

References

## Contents

1. Forward .......................................................... i
2. Referees .................................................................. ii
3. Machine Translation from African Languages to English
   DG Kourie, WJ vd Heever and GD Oosthuizen ......................................................... 1
   PZ Theron and I Cloete ............................................................... 9
5. A Lattice-Theoretic Model for Relational Database Security
   A Melton and S Shenoi .......................................................... 15
6. Network Partitions in Distributed Databases
   HL Viktor and MH Rennhackkamp ......................................................... 22
7. A Model for Object-Oriented Databases
   MM Brand and PT Wood .......................................................... 27
8. Quantifier Elimination in Second Order Predicate Logic
   D Gabbay and HJ Ohlbach ......................................................... 35
9. Animating Neural Network Training
   E van der Poel and I Cloete ......................................................... 44
10. HiLOG - a Higher Order Logic Programming Language
    RA Paterson-Jones and PT Wood ................................................ 53
11. ESML - A Validation Language for Concurrent Systems
    PJA de Villiers and WC Visser ................................................ 59
12. Semantic Constructs for a Persistent Programming Language
    SB Sparg and S Berman .......................................................... 65
13. The Multiserver Station with Dynamic Concurrency Constraints
    CF Kriel and AE Krzesinski ...................................................... 75
14. Mechanizing Execution Sequence Semantics in HOL
    G Tredoux ........................................................................ 81
15. Statenets - an Alternative Modelling Mechanism for Performance Analysis
    L Lewis .............................................................................. 87
    N Pendock ........................................................................ 95
17. Galileo: Experimenting with Graphical User Interfaces
    R Apteker and JM Bishop ......................................................... 99
18. Placing Processes in a Transputer-based Linda Programming Environment
19. Accessing Subroutine Libraries on a Network
    PH Greenwood and PH Nash ................................................... 117
20. A Multi-Tasking Operating System Above MS-DOS
    R Foss, GM Rehmet and RC Watkins ............................................. 122
21. Using Information Systems Methodology to Design an Instructional System
    BC O'Donovan ...................................................................... 126
22. Managing Methods Creatively
    G McLeod ........................................................................ 131
23. A General Building Block for Distributed System Management
    P Putter and JD Roos ......................................................... 141