ISSN 0254–2757

QI QUÆSTIONES INFORMATICÆ

Volume 6 • Number 1

May 1988

The official journal of the Computer Society of South Africa and of the South African Institute of Computer Scientists

Die amptelike vaktydskrif van die Rekenaarvereeniging van Suid-Afrika en van die Suid-Afrikaanse Instituut van Rekenaarwetenskaplikes

QUÆSTIONES INFORMATICÆ

The official journal of the Computer Society of South Africa and of the South African Institute of Computer Scientists

Die amptelike vaktydskrif van die Rekenaarvereniging van Suid-Africa en van die Suid-Afrikaanse Instituut van Rekenaarwetenskaplikes

Wirth / Hoare / Dijkstra / Gries /
Bach
Kirchen
Kowalski

Liotta

Guest Editorial

Subscriptions

The annual subscription is

	SA	US	UK
Individuals	R20	$ 7	£ 5
Institutions	R30	$14	£10

to be sent to:
Computer Society of South Africa
Box 1714 Halfway House 1685

Quæstiones Informaticæ is prepared by the Computer Science Department of the University of the Witwatersrand and printed by Printed Matter, for the Computer Society of South Africa and the South African Institute of Computer Scientists.

Editorial

Volume six of QI heralds several changes. The most visible is the change in format. The black on red cover has been changed to a more readable blue on white, but we have retained the style of the old cover, for the sake of continuity. The papers are now set in a tighter format, using double columns, which will enable more papers to be published for the same cost.

For authors, the most significant change is that as from Volume 6 Number 2 (the next issue), a charge will be made for typesetting. The charge is quite modest – R20 per page – and will enable us to keep up the high standards that we have become used to with QI. It is worth recording that the alternative to this suggestion was that authors should present camera-ready typescript, as is done for *Quæstiones Mathematicæ*. Given that document preparation and electronic typesetting is one of the areas of computer science that we can feel proud of, it seemed right that our journal should use the most modern techniques available. Fortunately, the two controlling bodies, the CSSA and SAICS, eventually agreed to our proposal and the result is the professional journal you have in front of you now.

Supporters of QI may be interested in a few statistics that I compiled when I took over the editorship from Gerrit Wiechers in April this year. In the past two years (June 1985 to June 1988), 73 papers have been received. Of these 39 (53%) have appeared, 19 have been rejected or withdrawn (26%) and 15 (21%) are either with authors for changes or with referees. If we look at the complete picture for Volumes 4 and 5, we find the following:

Volume	Issues	Papers	Pages	Ave. pages per paper
5	3	27*	220	7.7
4	3	21	136	6.4

Although this issue contains one very long paper of 18 pages, the future policy of QI will be to restrict papers to 6 or 7 printed pages, and prospective authors are asked to bear this in mind when submitting papers.

For the future, we are hoping to move towards more special issues. Many of the papers being published at the moment were presented at the 4th SA Computer Symposium in 1987. Instead of continuing the policy of allowing such papers to be accepted by QI without further refereeing, we are hoping to negotiate with Conference organisers to produce special issues of QI. Thus the proceedings would *ab initio* be typeset by QI and all the papers would be in a single issue. Given the competitive charges of QI, there will be financial gains for both parties in such an arrangement.

As this is my first editorial, it is fitting that it should close with a tribute to the previous QI team. My predecessor as editor was Gerrit Wiechers. Gerrit took over the editorship in 1980 and served the journal well over the years. With his leadership, the number and quality of the papers increased to its present healthy state. I must also extend a big thank you to Conrad Mueller and the University of the Witwatersrand who pioneered desk top publishing of QI in August 1985, using the IBM mainframe and its laser writer. Without Conrad's diligence and the excellent facilities provided by the Wits Computer Centre and subsequently the Computer Science Department, QI would easily have degenerated into a second-rate magazine. Quintin Gee, also of the Wits Computer Science Department, has taken over from Conrad and has raised the production quality of QI to new heights, as this issue testifies.

I look forward to your help and support in the future. Long live QI!

Judy M Bishop
Editor
June 1988

Protocol Performance Using Image Protocols

P S Kritzinger

Department of Computer Science, University of Cape Town, Private Bag, Rondebosch, 7700

Abstract

Performance analyses of data communication systems do not always rely on a detailed analysis of the underlying protocols. Those analyses which do, usually rely on an analysis of the protocol state transition graph. These graphs tend to become very large for nontrivial protocols and the analyses correspondingly complex. Image protocols is a recent approach to reduce the complexity of communication protocol analysis. The method allows for the construction of an image protocol which is generally smaller than the original protocol and its analysis therefore less complex. An image protocol system is said to be faithful if it preserves the safety and liveness properties of the original protocol system. In this paper we show that an image protocol is also faithful as far as its performance is concerned, a result which simplifies performance studies of the protocol considerably. An example which illustrates the principles involved is included.
Keywords: *Protocol performance, image protocol, aggregation, state transition graph, finite state machine, multiclass queueing theory, queueing network, MVA-algorithm.*

Received July 1987, Accepted August 1987

1. Introduction

Most international standards for data communication networks are based upon a layered architecture. At each layer tasks or protocol entities provide and receive services from the adjacent layers and execute a peer to peer protocol between that layer and the corresponding layer at remote stations. These architectures are very complex and initial reports about their performance in terms of measures such as Data Unit throughput are not encouraging.

Performance analyses of data communication networks do not always include a detailed analysis of the protocol itself as it influences the execution of the protocol function. Those analyses which do, make use of a formal specification of the protocol and more specifically, the protocol state transition graph. As is the case for protocol validation methods, these graphs become very large for real-life protocols and the analyses correspondingly complex.

Despite the obvious advantages to be gained, protocol performance analysis is not a very active research area. The first work to make use of a formal specification is that of Bauerfeld [2] followed by Rudin [8]. Analyses based upon Petri-net descriptions are those by Molloy [6] and Razouk [9]. More recently the author has proposed a performance analysis technique [5] which applies the techniques of multiclass queuing network theory. The latter analysis is based upon the state transition graph of a protocol and although it can handle systems with many thousands of states the method can benefit from state reduction as indeed can all of the proposed methods.

Several techniques have been proposed in the literature for the reduction of the protocol state space with the objective of protocol validation in mind. Recent advances in this regard is the work on decomposition by Choi and Miller [3], that on phase reduction techniques by Chow et al. [4] and the use of image protocols by Lam and Shankar [10].

In this paper we show how the performance analysis technique proposed by the author [5] can be combined with the method of projections as proposed by Lam and Shankar to reduce the complexity of the performance analysis. We define the concept of faithful performance of an image protocol and show that it applies.

After first introducing the protocol system model of Lam and Shankar the performance analysis technique is described in terms of that model. A very brief overview is given of the construction of an image protocol and the paper ends with an analysis of the performance properties of an image protocol.

2. Protocol System Model

The reader is referred to Lam and Shankar [10] for a full description of the protocol system model used here. In this paper only those definitions from that paper which are required for an understanding of the performance model, described in the next section, are given.

Let there be I protocol entities $P_1, P_2, \ldots, P_I$ and K channels $C_1, C_2, \ldots, C_K$. Let S_i be the set of states of P_i and M_{ik} be the set of messages that P_i can send into C_k.

The dynamics of these protocol entities and channels are described by entity events and channel events.

Channel events specified for channel C_k are denoted by E_k and are used to model various types of channel errors. The reader is referred to [10] for a full description of channel event errors. The occurrence of a channel event in E_k depends on, and changes only the state of C_k; no change in the state of a protocol entity is involved and for that reason channel events are not considered any further in this paper.

There are three types of entity events:

- *Send Events:* Let $t_i(r,s,-m)$ denote the event of P_i sending message m into channel C_k where $m \in M_{ik}$ and C_k is in the outgoing channel set of P_i. The send event is enabled when P_i is in state r. After the event occurrence, P_i is in state s and m has been appended to the end of the message sequence in C_k.
- *Receive Events:* Let $t_i(r,s,+m)$ denote the event of P_i receiving message m from channel C_k where $m \in M_{ik}$ and C_k is in the incoming channel set of P_i. The receive event is enabled when P_i is in state r and m is the first message in channel C_k. After the event occurrence, P_i is in state s and m is deleted from the channel.
- *Internal Events:* Let $t_i(r,s,\alpha)$ denote an internal event of P_i where α is a special symbol indicating the absence of a message. This internal event is enabled when P_i is in state r. After the event occurrence, P_i is in state s. Internal events model timeout occurrences internal to P_i, as well as interactions between the entity and its local user.

Each send or receive event may alter the states of the entity and the channel involved in the event. Internal entity events do not affect the state of any channel. Following Lam and Shankar we use T_i to denote the set of events specified for P_i, $i = 1, 2, \ldots, I$.

In most cases, the behaviour of P_i would be nondeterministic. For example, we may specify T_i to contain both $t_i(r,s,x)$ and $t_i(r,u,x)$ where $s \neq u$ as well as both $t_i(r,s,x)$ and $t_i(r,s,y)$ where $x \neq y$. We denote the probability of an entity event $t_i(r,s,x)$ by $p_i(r,s,x)$. Note that unless otherwise specified, we will use the notation $t_i(r,s,x)$ to mean all of $x = +m$, $-m$ or α.

Each event corresponds to a set of transitions in the global state space G of the protocol system. The union of the sets of transitions over all events specified for the protocol system will be denoted by τ. The simultaneous occurrences of multiple events in the protocol system are treated as occurrences of the same events in some arbitrary order. The pair (G,τ) is also called the global state transition graph.

Figure 1 taken from [10] is an example of two protocol entities P_1 and P_2 modelled by two communicating finite-state machines.

3. Protocol Performance Model

Each event in the protocol system is specified by its enabling condition and its execution. The enabling condition of an event is a predicate in the components of the global state of the protocol system. We will not concern ourselves with the logical properties of a protocol system in this paper.

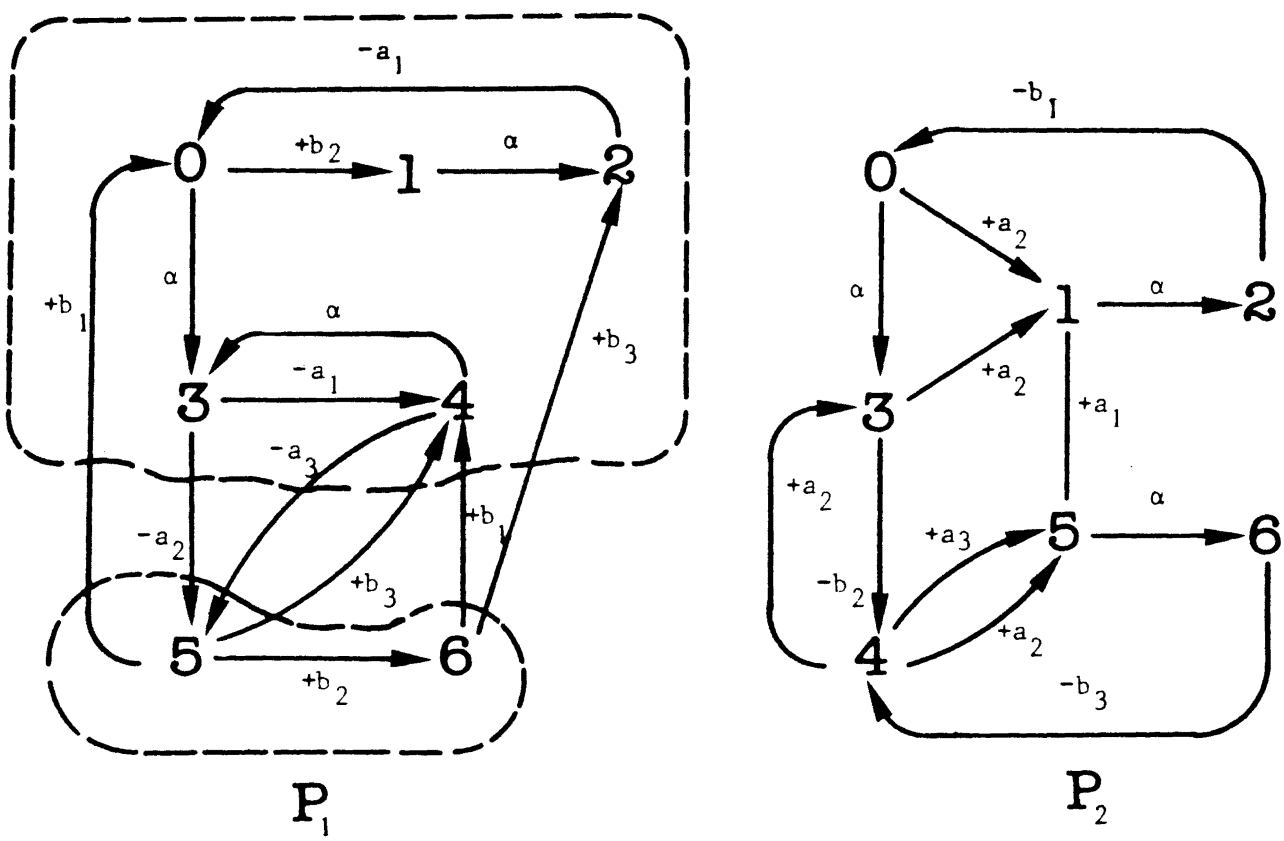

Figure 1. Two Communicating Finite-State Machines

The execution of an event however, specifies an update to the components of the global state and requires time from the processor on which the entity is executing.

The performance model first described in [5], assumes that there are l_i instances of entity P_i, $i = 1, 2, \ldots, I$ as would be the case with several endpoint connections in a real protocol system. It is moreover assumed that all protocol entities contend for service at a single processor. A measure of the performance of the protocol would typically be the rate at which a particular send event $t_i(r,s,-m)$ is executed or the throughput rate at the processor of all send events associated with entity P_i. These, and other performance quantities are computed by considering the protocol system as a closed multiclass queueing system with two service centres.

The one service centre is of the Processor Sharing type [1] and models the processor. The second service centre is a pure delay server and models the delays experienced in channels $C_1, C_2, \ldots, C_K$ associated with receive events.

Customers in the network are instances of the entities $P_1, P_2, \ldots, P_I$. A customer takes on a distinct class associated with each distinct entity event causing a state transition. Each distinct entity event therefore represents a distinct customer class. Each entity event is moreover classified to be either an *active event*, if the event requires processor time, or a *delay event* otherwise.

For example, all send events $t_i(r,s,-m)$ are active events, since they require the execution of a certain average number of instructions, which in turn requires a certain average amount of processor time. Receive events $t_i(r,s,+m)$ represent time delays in the corresponding channel. An internal event $t_i(r,s,x)$ could be either a delay event (denoted by $x = +\alpha$), representing a timeout event say, or an active event (denoted by $x = -\alpha$) when it represents an interaction between the entity and its local user requiring processor time.

Note that an event cannot be both an active event and a delay event; any such event would have to be decomposed into two events with a new intermediate state. If $t_i(r,s,\alpha)$ were such an event for example, it would be decomposed into an active event $t_i(r,s_1,-\alpha)$ and a delay event $t_i(s_1,s,+\alpha)$.

It follows from the above that we associate a time duration with every entity event. Denote the expected value of this time for event $t_i(r,s,x)$ by $\mu_i^{-1}(r,s,x)$.

In the model, a sequence of events will thus cause an entity to spend time in either one of the two servers, depending on whether the associated event is an active event or a delay event. For example, an entity in a transition due to active event $t_i(r,s,-m)$ having received a corresponding average service time $\mu_i^{-1}(r,s,-m)$ from the processor centre, will leave and either return to that same centre if the following event is again an active event, or would proceed to the delay centre if the following event is a delay event. In this way an entity would pass nondeterministically between the servers, depending on the sequence of events.

Figure 2 illustrates these concepts for the communicating finite state machine P_1 of Figure 1 where, arbitrarily, the internal events $t_i(r,s,\alpha)$ have been considered to be active events for all r, s and α.

Note that the only assumptions required to solve the model as a closed multiclass queueing network are that,

- the probabilities $p_i(r,s,x)$ are independent for all r, s and x, and
- the time $\mu_i^{-1}(r,s,x)$ associated with an event is independent of the time associated with any other event in the same protocol system.

4. Analysis of the Performance Model

As one event follows another, an entity P_i changes from one to the next, not necessarily different state in S_i. Let $\xi_i(r,,)$ denote the expected value of the number of times P_i is in state $r \in S_i$ between successive times that it is in some arbitrarily chosen state $r^* \in S_i$. In other words, $\xi_i(r,,)/\xi_i(r^*,,)$ is the relative number of times that P_i would be in state r compared to the number of times it would be in state r^*. The $\xi_i(r,,)$ satisfy the set of linear equations given by

$$\xi_i(r,,) = \sum_{s \in S_i} \sum_{x} \xi_i(s,,)p_i(s,r,x) \qquad (1)$$

where Σ_x denotes the summation over all $x \in M_{ik}$, $k = 1, 2, \ldots, K$, such that $t_i(s,r,x)$ takes entity P_i from state s to state r. It is not difficult to see that if $\xi_i(r,s,x)$ denotes the expected value of the relative number of times entity event $t_i(r,s,x)$ is executed, that

$$\xi_i(r,s,x) = p_i(r,s,x)\xi_i(r,,) \qquad (2)$$

Define the relative utilisation $v_{ia}(r,s,x)$ of the processor centre by instances of entity P_i in a transition due to active event $t_i(r,s,x)$ to be

$$v_{ia}(r,s,x) = \xi_i(r,s,x)/\mu_i(r,s,x) \qquad (3)$$

where $x = -m$ or $-\alpha$. Similarly we define the relative utilisation $v_{id}(r,s,y)$ of the delay centre by instances of entity P_i in a transition due to delay event $t_i(r,s,y)$ to be

$$v_{id}(r,s,y) = \xi_i(r,s,y)/\mu_i(r,s,y) \qquad (4)$$

where $y = +m$ or $+\alpha$.

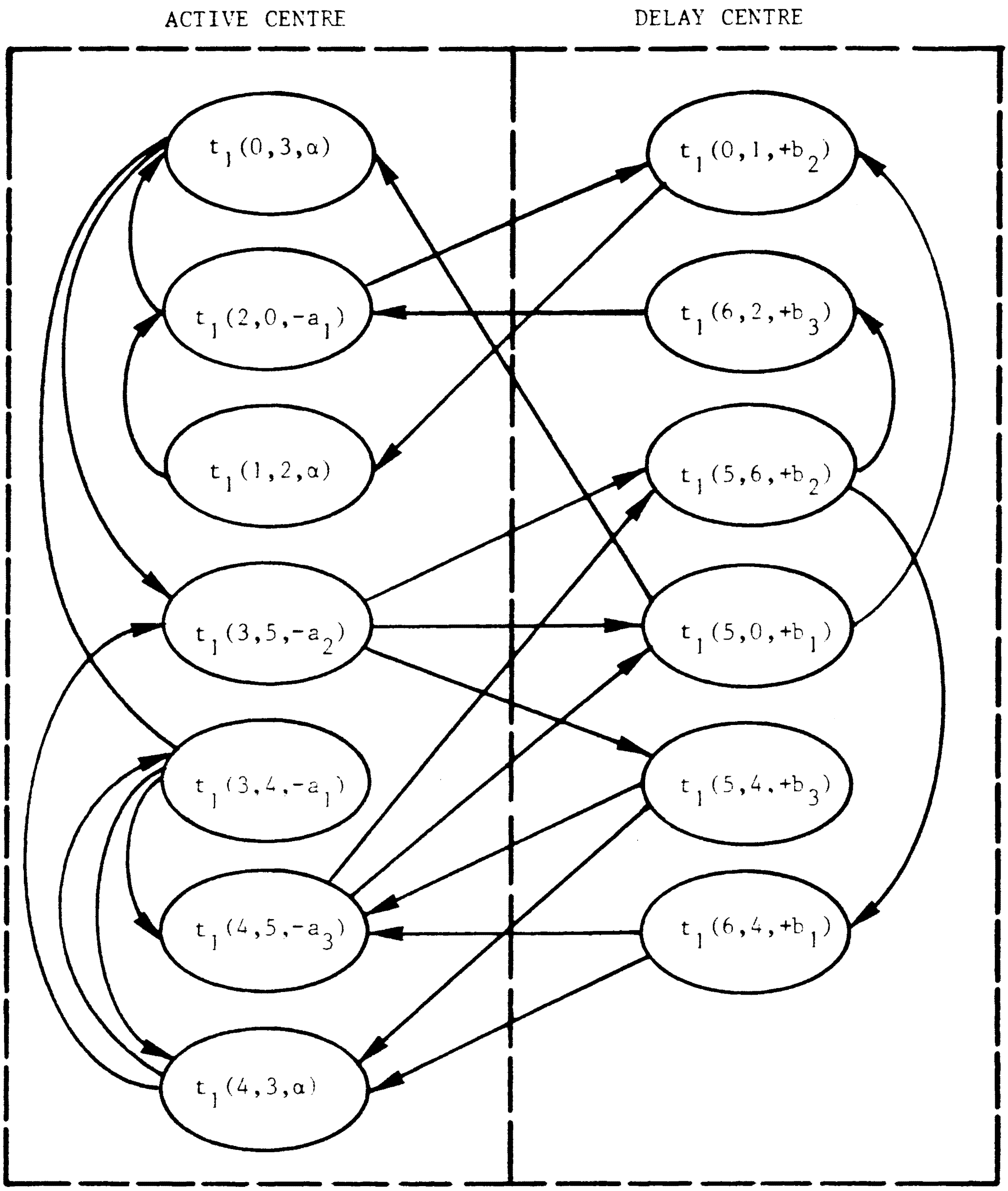

Figure 2. Event Sequences in the Two Servers for Entity P_1 of Fig. 1

Denote the total relative utilisation of the processor centre by P_i by

$$\rho_{ia} = \sum_{r,s \in S_i} \sum_{x} v_{ia}(r,s,x) \tag{5}$$

where $x = -m$ or $-\alpha$, and of the delay centre by

$$\rho_{id} = \sum_{r,s \in S_i} \sum_{y} v_{id}(r,s,y) \tag{6}$$

where $y = +m$ or $+\alpha$.

Let $\mathbf{L} = (l_1, l_2, \ldots, l_I)$ be the system population vector of the number of instances l_i of entity P_i, $i = 1, 2, \ldots, I$. Let $\chi_{ia}(r,s,x,\mathbf{L})$ denote the average rate at which active events $t_i(r,s,x)$ are completed at the processor or in other words, the expected throughput rate of the l_i instances of the entity P_i in a transition due to that event. It is known [11] that the value $\chi_{ia}(r,s,x,\mathbf{L})$ is given by

$$\chi_{ia}(r,s,x,\mathbf{L}) = \xi_i(r,s,x)\chi_i^*(\mathbf{L}) \tag{7}$$

where $x = -m$ or $-\alpha$. The average throughput rate $\chi_{ia}(\mathbf{L})$ of entity P_i due to any active event $t_i(r,s,x)$ where $r,s \in S_i$, $x = -m$ or $-\alpha$, is the sum of the throughputs given by (7) for individual events. That is,

$$\chi_{ia}(\mathbf{L}) = \Xi_{ia}\,\chi_i^*(\mathbf{L}) \tag{8}$$

with

$$\Xi_{ia} = \sum_{r,s \in S_i} \sum_{x} \xi_i(r,s,x) \qquad (9)$$

$\chi_i^*(\mathbf{L})$ in (7) and (8) is a mathematical quantity without direct physical interpretation and which is calculated in the Mean-Value-Analysis (MVA)-Algorithm [7] presented in Fig. 3.

Algorithm

Step 1. *initialisation*

set $Q_a(0, 0, \ldots, 0) = 0$

Step 2. *loop over the number of instances of every entity*

let $i = (i_1, i_2, \ldots, i_I)$

for $i_1 = 0, 1, \ldots, l_1$ $i_2; = 0, 1, \ldots, l_2; \ldots; i_I = 0, 1, \ldots, l_I$ *do*;

(i_1 *changes most rapidly*)

Step 3. *loop for every entity*

for $j = 1, 2, \ldots, I$ *do*;

$$R_{ja}(i) = \frac{\rho_{ja}}{\Xi_{ja}}[Q_a(i - \mathbf{e}_j) + 1]$$

$$R_{jd}(i) = \frac{\rho_{jd}}{\Xi_{jd}}$$

$$\chi_j^*(i) = i_j/(\rho_{ja}[Q_a(i - \mathbf{e}_j) + 1] + \rho_{jd})$$

end;

end of step 3.

$$Q_a(i) = \sum_{j=1}^{I} \Xi_{ja}\chi_j^*(i)R_{ja}(i)$$

end;

end of Step 2.

Figure 3. The MVA-Algorithm

The average queue length $Q_{ia}(r,s,x,\mathbf{L})$ of instances of entity P_i in transition due to event $t_i(r,s,x)$, $x = -m$ or $-\alpha$ is given by

$$Q_{ia}(r,s,x,\mathbf{L}) = \frac{\xi_{ia}(r,s,x)}{\rho_{ia}} \Xi_{ia}\chi_i^*(\mathbf{L})R_{ia}(\mathbf{L}) \qquad (10)$$

The quantity $R_{ia}(\mathbf{L})$ in the last equation is the average response time at the processor centre of instances of entity P_i in any transition. According to Little's formula the corresponding quantity $R_{ia}(r,s,x,\mathbf{L})$ for entity P_i in transition due to an active event $t_i(r,s,x)$, $x = -m$ or $-\alpha$, is given by

$$R_{ia}(r,s,x,\mathbf{L}) = Q_{ia}(r,s,x,\mathbf{L}) / \chi_{ia}(r,s,x,\mathbf{L}) \qquad (11)$$

The same statistical quantities can be computed for the delay centre by replacing the subscript "a" by the subscript "d" in equations (7) - (11) and computing the relevant quantities for delay events.

The vector quantity $\mathbf{e}_j$ is a unit vector in the j-th direction. The reader is referred to [5] for a full description of the computation of these quantities using the MVA-algorithm.

5. Constructing an Image Protocol System

An image protocol system is a partition of each of the sets

S_i, M_{ik}, T_i, E_k for all i and k

in the original protocol system specification. Protocol entity states, messages and events in each partition subset are treated as equivalent and are aggregated to form a single quantity, called their image, in the image protocol system. Since partition subsets are mutually exclusive and collectively exhaustive, quantities that are treated as equivalent have the same image; quantities not treated as equivalent have different images. Following Lam and Shankar we shall use x' to denote the image of a protocol quantity x.

Since an image protocol is obtained from the original protocol by aggregations, it captures only part of the logical behaviour of the original protocol system. First, global states of the image protocol system correspond to aggregations of global states of the original protocol system. Second, in the global state space of the image protocol system, the observable effect of different events in the original protocol system may be different or nil. Our objective however, is to show that the performance behaviour of the image protocol system will be faithful to that of the original protocol system. Before doing that however, and for the convenience of the reader, we describe the process used in [10] to construct an image protocol system.

5.1 Aggregation of Entity States

An image state s' of a set of states of the original state space S_i is constructed by partitioning S_i. All entity states in a partition subset are aggregated to the same image. Let S'_i denote the set of images of states in S_i i.e., $S'_i = \{s' : s \in S_i\}$. S'_i is called the image state space of P_i.

For example, Fig. 1 shows the partition {{0, 1, 2, 3, 4}, {5, 6}} of the state space S_1 of entity P_1. Let image state 0′ denote the image of states 0, 1, 2, 3 and 4 in S_1 and 5′ denote the image of states 5 and 6 in S_1. The image state space of P_1 is $S'_1 = \{0', 5'\}$. Similarly let image states 0′, 1′ and 2′ respectively denote the images of the states in the partitions {0, 3, 4}, {1, 5} and {2, 6} of S_2. In this case $S'_2 = \{0', 1', 2'\}$.

5.2 Aggregation of Messages

Only when the reception of two messages m and n cause identical state changes in the image state space of the receiver of C_k are they treated as equivalent. This equivalence relation partitions M_{ik}, and messages within the partition subset may be aggregated to form an image message m'. The image message sets are given by

$M'_{ik} = \{m' : m \in M_{ik}\}$ for all i and k.

In particular, messages in M_{ik} whose receptions do not cause any state change in the image state space of the receiver are said to have a null image.

The reader is referred to the example in [10] for the derivation of the image message sets $M' = \{a_2', a_3'\}$, and $M'_2 = \{b_1'\}$ where b_1' denotes the image of messages b_1 and b_3 and the image of a_1 and b_2 are both null. These transitions in the image state spaces are illustrated in Fig. 4.

5.3 Aggregation of Entity Events

Entity events which have the same observable effect in the image global state space are aggregated to the same image entity event. An image event whose occurrence does not have any observable effect in the image global state space is said to be a *null image event*. An image internal event $t_i(s',r',\alpha)$ is a null image event if $s' = r'$. Image send events involving null image messages and infinite buffer channels are treated as image internal events in T'_i; in this case the image event is represented as $t_i(s',r',\alpha)$ and it is a null image event if $s' = r'$. Finally, image receive events involving null image messages must have $s' = r'$ by definition; hence such receive events are null image events if the channel involved is an infinite buffer channel.

The set of image events for P_i is defined by $T'_i = \{t_i(s',r',x') : t_i(s,r,x) \in T_i\}$ where $x = +m$, $-m$ or α and the image of $t_i(s,r,x)$ is not a null image event.

The reader is referred to the example in [10] for the derivation of

$$T'_1 = \{t_1(0',5',-a'_2), \{t_1(0',5',-a'_3), t_1(5',0',+b'_1)\}$$

and

$$T'_2 = \{t_2(0',0',+a'_2), t_2(0',5',-a'_3), t_2(0',1',+a'_3), t_2(1',2',\alpha), t_2(2',0',-b'_1)\}$$

in Fig. 5 which illustrates the image protocol constructed from the original protocol illustrated in Fig. 1.

6. Properties of Image Protocols

Lam and Shankar [10] discuss various logical properties of image protocols in relation to the logical properties of the original protocol.

In [5] the question was posed whether, depending typically on the constituent functions of the protocol, it would be possible to decompose a protocol state transition graph into its constituent

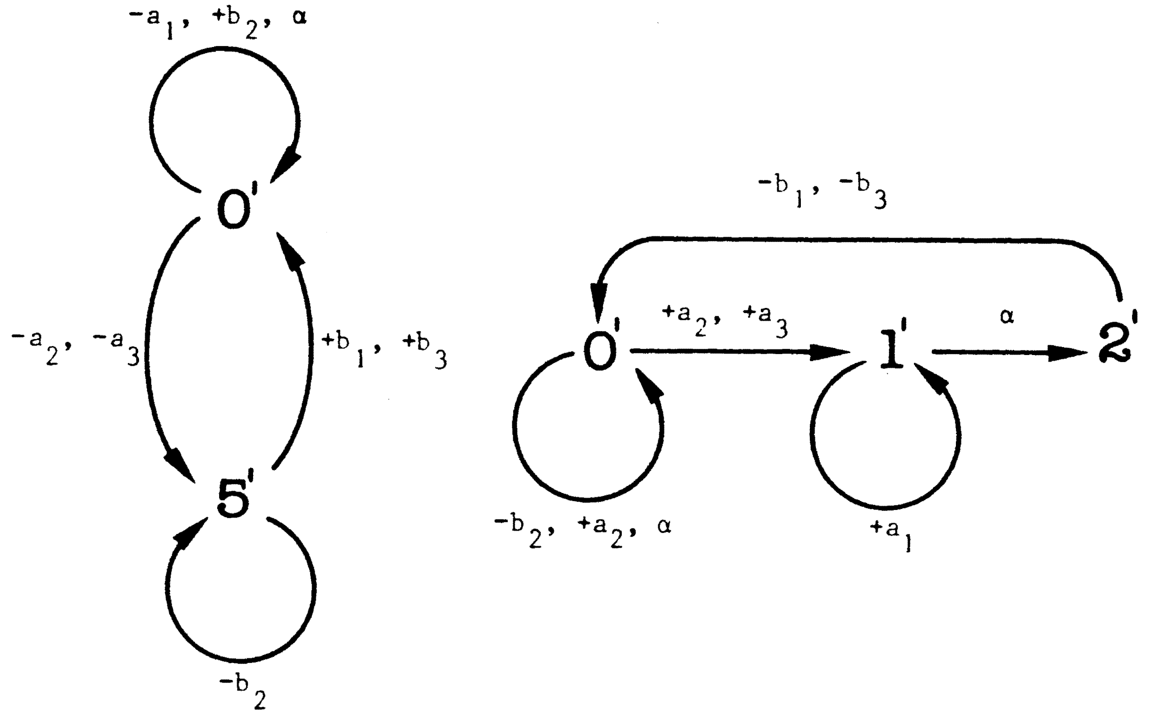

Figure 4. Transitions in the Image State Spaces

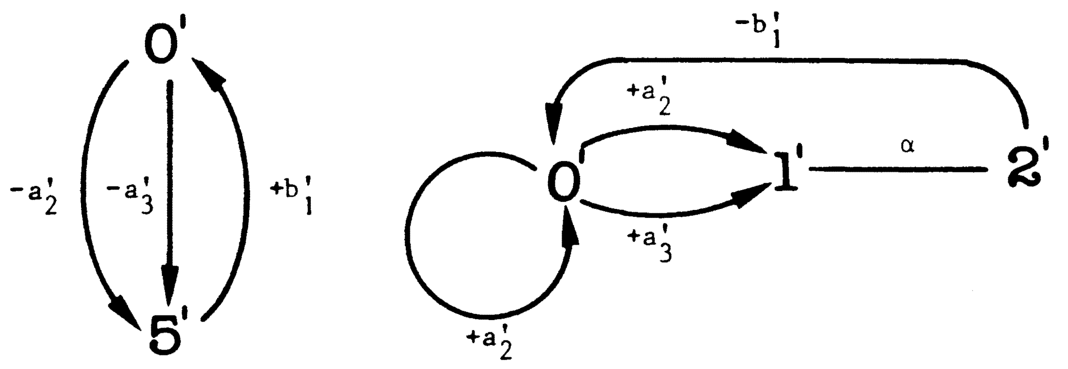

Figure 5. Image Protocol Constructed from the Original Protocol in Fig. 1

subgraphs, aggregate each subgraph and still have a system whose performance would be faithful to the original system. In this section we show that this is indeed possible.

6.1 Performance Properties

Let $t_i(r',s',x')$ denote the image entity event of a subset of entity send events in T_i. Define the relative utilisation of the processor centre by all instances of entity P_i in a transition due to event $t_i(r',s',x')$ to be

$$\nu_{ia}(r',s',x') = \sum_{(r',s',x')} \xi_i(r,s,x)/\mu_i(r,s,x) \tag{12}$$

Here $\Sigma_{(r',s',x')}$ denotes the summation over all send events $t_i(r,s,x)$ in the original protocol system with image $t_i(r',s',x')$.

Similarly, let $t_i(r',s',y')$ denote the image entity event of a subset of entity receive events T_i. Define the relative utilisation of the delay centre by all instances of entity P_i in a transition due to event $t_i(r',s',y')$ to be

$$\nu_{id}(r',s',y') = \sum_{(r',s',y')} \xi_i(r,s,y)/\mu_i(r,s,y) \tag{13}$$

where $\Sigma_{(r',s',y')}$ denotes the summation over all receive events $t_i(r,s,y)$ in the original protocol system with image $t_i(r',s',y')$.

Also let $\xi_i(r',s',x')$ denote the relative frequency of execution of the image event $t_i(r',s',x')$. Clearly

$$\xi_i(r',s',x') = \sum_{(r',s',x')} \xi_i(r,s,x) \tag{14}$$

where $\Sigma_{(r',s',x')}$ has the same meaning as above. The quantities $\nu_{ia}(r',s',x')$, $\nu_{id}(r',s',y')$, and $\xi_i(r',s',x')$, are respectively the image protocol quantities corresponding to $\nu_{ia}(r,s,x)$, $\nu_{id}(r,s,y)$, and $\xi_i(r,s,x)$ of the original protocol defined in (2), (3) and (4).

Although all entity events $t_i(r,s,x)$ whose image is a null image event are eliminated in the construction of the protocol image system, they cannot be ignored in the analysis of the image protocol system. The effect of these events have to be taken into account in the computation of the performance quantities ρ_{ia} and ρ_{id} defined in (5) and (6). Let $T_i^{n-} \subseteq T_i$ denote the set of send events in P_i which have null images. Define

$$\rho_{ia}^{n-} = \sum_{T_i^{n-}} \nu_{ia}(r,s,x) \tag{15}$$

where $x = -m$ or $-\alpha$. Similarly, let $T_i^{n+} \subseteq T_i$ denote the set of receive events in P_i which have null images. Define

$$\rho_{ia}^{n+} = \sum_{T_i^{n+}} \nu_{id}(r,s,x) \tag{16}$$

where $x = +m$ or $+\alpha$.

Rewrite (5) and (6) in terms of image entity events,

$$\rho'_{ia} = \rho_{ia}^{n-} + \sum_{r',s' \in S'_i} \sum_{x'} \nu_{ia}(r',s',x') \tag{17}$$

where $x' = -m'$ or $-\alpha$, and

$$\rho'_{ia} = \rho_{ia}^{n+} + \sum_{r',s' \in S'_i} \sum_{x'} \nu_{id}(r',s',y') \tag{18}$$

where $y' = +m'$ or $+\alpha$, and

Definition: An image protocol system is said to have faithful performance if the value of a performance statistic of the entity P_i in a transition due to an image event $t_i(r',s',x')$ is the sum of the performance statistics of the entity events whose image is $t_i(r',s',x')$.

Note that if an image event is the image of a single event in the original protocol, that the statistics for the two events will be identical.

Theorem: The performance of an image protocol system constructed as explained above, is faithful.

Proof: The proof follows from the MVA-algorithm using the quantities ρ'_{ia} and ρ'_{id} defined in (17) and (18) instead of ρ_{ia} and ρ_{id} as well as the equivalent

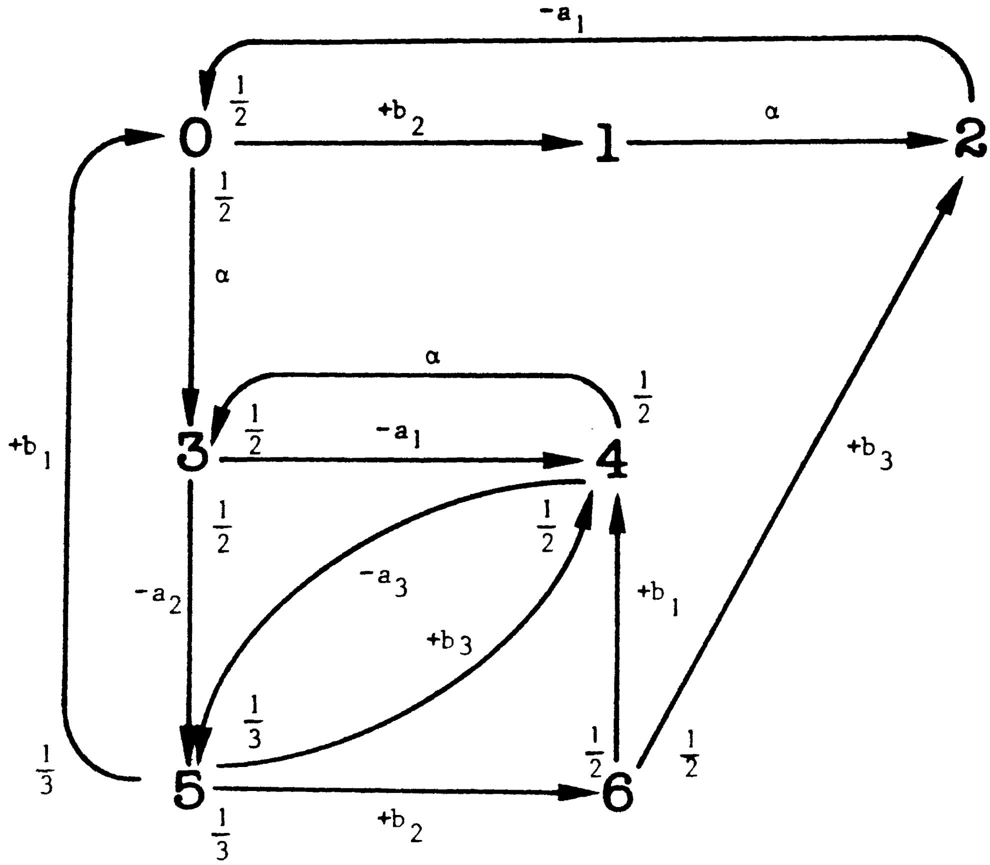

Figure 6. Finite State Machine with the Probability Values Indicated.

image protocol quantities defined in (12), (13) and (14) and substituting in the appropriate equations.

As an example, consider the throughput rate $\chi_{1d}(5',0',+b_1')$ at the delay centre of entity P_1 in transition due to image entity event $t_i(5',0',+b_1')$. Assume a single instance of P_1, the expected value of all times to be unity and the probabilities of the various events as illustrated in Fig. 6.

Computing the quantities given by equations (12) - (18) for the image protocol and applying the MVA-algorithm, it can be shown that $\chi_{1d}(5',0',+b_1') =$ 0.18182 per unit time from the image protocol equivalent equation (7). However, $t_1(5',0',+b_1')$ is the image of entity events $t_1(5,0,+b_1)$, $t_1(5,4,+b_3)$, $t_1(6,4,+b_1)$ and $t_1(6,2,+b_3)$. The equivalent throughputs are $\chi_{1d}(5,0,+b_1) = 0.06061$ and $\chi_{1d}(5,4,+b_3) = 0.06061$ and $\chi_{1d}(6,4,+b_1) = 0.03030$ and $\chi_{1d}(6,2,+b_3) = 0.03030$ per unit time. The result follows.

7. Conclusion

Previous work by other authors have shown how to construct image protocols for individual protocol functions of a complex protocol. The complexity of the image protocol system is less, and under certain conditions it is faithful to the original protocol system in all its safety and liveness properties.

Image protocols have a further advantage in that they reduce the complexity of analysing the performance of the protocol. In this paper it is shown that the performance of an image protocol system is also faithful to that of the original protocol system. It is therefore possible to do a performance study of the reduced image protocol system, where the results obtained would be the same as when the analysis were applied to the original protocol system.

References

1. F. Baskett, K.M. Chandy, R.R. Muntz and F.A. Palacios, [1975], Open, closed and mixed networks of queues with different classes of customers, *J. ACM*, **22** (2), 248-260.
2. W.L. Bauerfeld, [1983], Protocol performance prediction, *Proc Intnl Conf on Comm*, IEEE, Boston, 1311-1315, June.
3. T.Y. Choi and R.E. Miller, [1983], A decomposition method for the analysis and design of finite state protocols, *ACM SIGCOMM '83 Symposium on Communication Architectures and Protocols*, 167-176, October.
4. C.H. Chow, M.G. Gouda and S.S. Lam, [1985], A discipline for constructing multiphase communication protocols, *ACM Trans. on Computer Syst.*, **3** (4), 315-343, November.

5. P.S. Kritzinger, [1986], A performance model of the OSI communications architecture, *IEEE Trans. Commun.*, **COM-43**, 554-563.
6. M.K. Molloy, [1982], Performance analysis using stochastic Petri nets, *IEEE Trans. Comp.*, **C-31** (9), 913-917.
7. M. Reiser and S.S. Lavenberg, [1980], Mean-value analysis of closed multichain queueing networks, *J. ACM*, **27**, 313-322.
8. H. Rudin, [1984], An improved algorithm for estimating protocol performance, *Proc. IVth Workshop on Protocol Specification, Testing, and Verification*, Pennsylvania, June.
9. R.R. Razouk and C.V. Phelps, [1984], Performance analysis using timed Petri nets, *Proc. IVth Workshop on Protocol Specification, Testing, and Verification*, Pennsylvania, June.
10. S.S. Lam and A.U. Shankar, [1984], Protocol verification via Projections, *IEEE Trans. Software Engineering*, **SE-10** (4), 325-342, July.
11. S.S. Lavenberg, Ed., [1983], ch. 3.5 in *Computer Performance Modeling Handbook.*, New York: Academic Press Inc.

This paper first appeared in the Proceedings of the 4th South African Computer Symposium.

NOTES FOR CONTRIBUTORS

The purpose of the journal will be to publish original papers in any field of computing. Papers submitted may be research articles, review articles and exploratory articles of general interest to readers of the journal. The preferred languages of the journal will be the congress languages of IFIP although papers in other languages will not be precluded.

Manuscripts should be submitted in triplicate to:

Professor J M Bishop
Department of Computer Science
University of the Witwatersrand
Johannesburg
Wits
2050

Form of manuscript

Manuscripts should be in double-space typing on one side only of sheets of A4 size with wide margins.

The first page should include the article title (which should be brief), the author's name and affiliation and address. Each paper must be accompanied by an abstract less than 200 words which will be printed at the beginning of the paper, together with an appropriate key word list and a list of relevant Computing Review Categories.

Manuscripts may be provided on disc using any Apple Macintosh package or in ASCII format.

For authors wishing to provide camera-ready copy, a page specification is freely available on request from the Editor.

Tables and figures

Tables and figures should not be included in the text, although tables and figures should be referred to in the printed text. Tables should be typed on separate sheets and should be numbered consecutively and titled.

Figures should also be supplied on separate sheets, and each should be clearly identified on the back in pencil with the authors name and figure number. Original line drawings (not photocopies) should be submitted and should include all the relevant details. Photographs as illustrations should be avoided if possible. If this cannot be avoided, glossy bromide prints are required.

Symbols

Mathematical and other symbols may be either handwritten or typewritten. Greek letters and unusual symbols should be identified in the margin. Distinction should be made between capital and lower case letters; between the letter O and zero; between the letter I, the number one and prime; between K and kappa.

References

References should be listed at the end of the manuscript in alphabetic order of the author's name, and cited in the text in square brackets. Journal references should be arranged thus:

[1] E. Ashcroft and Z. Manna, [1972], The Translation of 'GOTO' Programs to 'WHILE' programs, *Proceedings of IFIP Congress 71,* North-Holland, Amsterdam, 250-255.

[2] C. Bohm and G. Jacopini, [1966], Flow Diagrams, Turing Machines and Languages with only Two Formation Rules, *Comm. ACM,* **9**, 366-371.

[3] S. Ginsburg, [1966], *Mathematical Theory of Context-free Languages*, McGraw Hill, New York.

Proofs

Proofs will be sent to the author to ensure that the papers have been correctly typeset and *not* for the addition of new material or major amendment to the texts. Excessive alterations may be disallowed. Corrected proofs must be returned to the production manager within three days to minimise the risk of the author's contribution having to be held over to a later issue.

Only original papers will be accepted, and copyright in published papers will be vested in the publisher.

Letters

A section of "Letters to the Editor" (each limited to about 500 words) will provide a forum for discussion of recent problems.