Modelling and Performance Analysis of Inter-Processor Messaging in Distributed Systems

David Manfield *
Telecom Australia Research Laboratories, Melbourne, Australia 3168

Phuoc Tran-Gia *
IBM Research Division, Zürich Research Laboratory, Säumerstrasse 4, CH-8803 Rüschlikon, Switzerland

Herbert Jans *
Faculty of Electrical Engineering, Technical College, Landshut, Fed. Rep. Germany

Received 6 September 1984
Revised 27 January 1987

This paper addresses some performance modelling issues arising from the inter-processor messaging requirements of distributed real-time processing systems. We consider two basic classes of message transfer protocols, namely clocked schemes where message transfer is initiated as a periodic timed task, and event-driven schemes where the transfer mechanism is triggered by the messaging requests themselves. The aim is to maximize system efficiency by passing messages in batches. It is shown how the classes of protocols may be modelled by a two-stage queueing systems, which is analysed using the theory of imbedded Markov chains and semi-Markov processes. The results are used to show how the important performance measures are derived, and how the protocol parameters should be chosen to optimize the overall message system performance. The methods are illustrated by numerical examples.

Keywords: Clocked Schedule, Inter-Processor Communications, Delay Analysis, Imbedded Markov Chain, Stability Analysis, Two-Level Queueing System.

1. Introduction

Currently, there is a trend in real-time computer systems towards decentralized multi-processor architectures in order to take advantage of functional divisions in the applications software, and to provide increased processor power in a modular way. Good examples are modern telecommunication switching systems, where there is often a clear division of tasks and the overall system has to be easily extendable. The concurrent application programs running in the distributed processors are synchronized by the passing of messages over some type of interconnection network (e.g., bus system) under the control of a messaging protocol. One price to be paid for such distributed processing is the real-time overhead which is expended to perform the inter-processor communication functions, and this can have a significant impact on the real-time performance. The idea here is to show how a class of inter-processor messaging protocols can be optimized from the performance point of view to minimize message delays, and minimize real-time overhead due to the communication function.

* This work was done while the authors were at the University of Siegen, Fed. Rep. Germany.

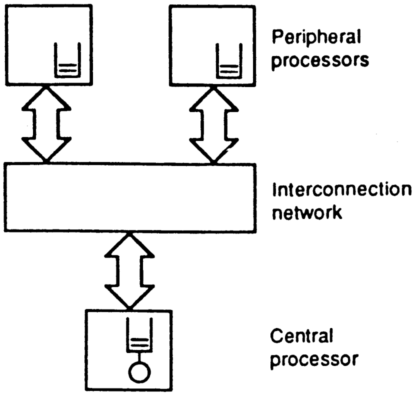

Fig. 1. Generic system architecture.

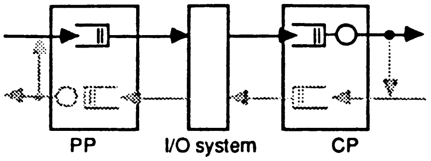

Fig. 2. Model building block.

In this paper we want to concentrate on the modelling steps for the performance evaluation of messaging in a distributed system with the generic architecture depicted in Fig. 1, where a number of peripheral processors have access to a central processor via an interconnection network such as a bus system. Tasks executing in the peripheral processors generate events which require further processing in the central processor (CP), and these events are passed as messages via the peripheral processor (PP) transmit queue. We assume that the peripheral queues can be reduced to a single logical queue (through

David Manfield completed his B.E. and Ph.D. degrees in Electrical Engineering at the University of Queensland, Australia, in 1975 and 1979 respectively. His doctoral thesis was on the subject of teletraffic engineering of telephone networks. He then spent one year as an Alexander von Humboldt post-doctoral fellow in the Department of Communications Engineering at the University of Siegen, Fed. Rep. Germany, working on the performance modelling of switching systems. In 1981 he joined Bell-Northern Research in Ottawa, Canada, where he worked on the traffic and real-time design of computer controlled switching systems, becoming the manager of product traffic design in 1984. Since March 1987, Dr. Manfield has been with the Telecom Australia Research Laboratories in Melbourne, Australia, where he is involved in future network planning.

Phuoc Tran-Gia received the M.S. degree (Dipl.-Ing.) from the Stuttgart University, Stuttgart, in 1977 and the Ph.D. degree (Dr.-Ing.) from the University of Siegen, Siegen, in 1982, both in Electrical Engineering. In 1977, he joined Standard Elektrik Lorenz (ITT), Stuttgart, where he was working on software development of digital switching systems. From 1979 until 1982 he was working as an assistant professor in the Department of Communications, University of Siegen. From 1983 to 1986, Dr. Tran-Gia was head of a research group at the Institute of Communications Switching and Data Technics, Stuttgart University, Fed. Rep. Germany. The research activity was in the field of environment simulations for communication systems and applied queueing theory, especially with focus on discrete-time methods. In 1985, he was appointed lecturer for computer networks at the University of Würzburg, Fed. Rep. Germany. He joined the IBM Zürich Research Laboratory in October 1986, where he is currently working on the architecture and performance evaluation of computer communication systems.

Herbert Jans received the M.S. degree (Dipl.-Ing.) in Electrical Engineering from the Technical University of Berlin, Germany, in 1975. From 1975 he was an Assistant Professor at the University of Siegen, Faculty of Electrical Engineering, and completed his Ph.D. degree in 1983. His research activities and doctorial thesis were in the field of queueing theory and its application in computer and communication systems. In 1983 he joined Siemens A.G. in Münich where he worked on the design of traffic and real-time problems of the Siemens computer controlled switching systems EWSD. In October 1986, Dr. Herbert Jans was appointed Professor at the Technical College (Fachhochschule) of Landshut, Fed. Rep. Germany. His main research interests are in the area of performance analysis of communication systems

the use of centrally queued agents or otherwise). The system we wish to concentrate on is depicted in Fig. 2. In principle, the message traffic is two-way, but our methodology is based on separating the flows in each direction assuming mutual independence. This is a good assumption if only a few messages generate correlated replies to the originating processor.

The message transfer is controlled by a communication task in the CP. Its job is to take messages from the PP queue and load them into the CP queue where they await processing by the CP. The communication task may be scheduled a number of different ways. In real-time systems, the level of message traffic may be very high, so significant gains may be made by passing messages in batches to minimize overhead. It is necessary to ensure as well that no message will be delayed too long. In the following we treat two classes of messaging protocol, the first being when the transfer is implemented by a clocked schedule, and the second when the transfer is event-driven (triggered by the events in the PP). Our objective is to produce queueing models for the message transfer protocols so we can make an optimal choice of the protocol parameters to achieve our delay and robustness criteria for the message system.

2. Queueing model

The queueing model which we use for the performance analysis of the model building block of Fig. 2 is depicted in Fig. 3. The generation of message event arrivals to the PP queue is assumed to be Poisson with rate λ, based on the observation that events are usually generated by a large number of sources (tasks, devices). The events accumulate in the PP queue until the next scheduled occurrence of the CP communication task when the messages may be transferred to the CP queue. The PP queue may be served FIFO or random order of service (ROS) depending on implementation. The switch in Fig. 3 represents the communication function. The CP queue is assumed to have a finite number of waiting places (S) with the CP system size $N = S + 1$. The work time for an arbitrary message in the CP is assumed to be exponentially distributed with mean value $1/\mu$. The overhead incurred by the communication task is assumed to be fixed and equal to T_0.

The batch transfer of messages can be done in several ways, and in this paper we consider the following schedules:

Version 1. Message transfer at clocked intervals, period T, implemented as a timed task within the CP.

Version 2. Event-driven transfer after n new messages have accumulated in the PP queue generating an external interrupt, but with the inter-transfer interval subject to a timeout, length T, on the occurrence of which a transfer is initiated anyway.

For either of the above versions, the message transfer protocol can be blocking or nonblocking, depending on how messages are treated when the CP queue is full. In the blocking case, messages finding the CP queue full are discarded and the originating task is left to recover by itself. In the nonblocking case, messages blocked from the CP queue are allowed to remain in the PP queue until the next transfer epoch, so that they ultimately find a place in the CP queue. Whether the transfer protocol is blocking or nonblocking has significant implications for the system design and provisioning, depending on the application.

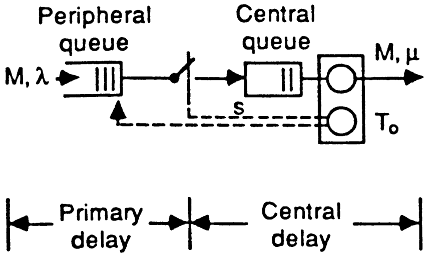

Fig. 3. Queueing model.

In the following sections we apply the queueing model of Fig. 3 to the different versions of the transfer protocol described above. The object is to highlight the performance issues by developing analytical methods for the determination of the key performance measures, and indicating how the protocol parameters need to be chosen to ensure good overall system operation. In Section 3 we look at the clocked schedules of Version 1, and in Section 4 we look at the event-driven schedules of Version 2.

3. Clocked schedules

For a clocked schedule, we assume that the communication task is a timed task in the CP with period T. The main question is how large T should be for best system performance. If it is too small, then many communication interrupts will take place, often with no messages waiting in the PP, so a lot of overhead will be incurred for no purpose. At the other extreme of large T, interrupts will come too seldom and messaging delays will be large because messages are forced to wait a long time in the PP queue. Further complications are induced by the finite size of the CP queue.

3.1. Imbedded Markov chain analysis

The operation of the messaging system is depicted in the time line diagram in Fig. 4. The message events occurring randomly in the PP are stored (temporarily) in the PP queue. At the next clock epoch, the total contents of the PP queue is presented as a batch to the CP queue and, depending on whether the transfer protocol is blocking or nonblocking, messages not finding a place in the CP queue are either discarded or remain in the PP queue respectively. During the active service period, the CP is available for event processing but is not necessarily busy, even if messages are waiting in the PP queue. The number in the CP queue is denoted by the random variable $X(t)$ and the number in the PP queue by $Y(t)$. In an arbitrary clock period, the process $X(t)$ will count down after an initial jump at the times $\{t_n^+\}$ as the messages are processed. $Y(t)$ will count up as messages arrive. At the clock period, the two queues are momentarily joined and an appropriate number of messages transferred to the CP queue, following the service discipline of the PP queue.

The instants $\{t_n^+\}$ are seen to form the regeneration points of an Imbedded Markov Chain (IMC) since, at these points, the state of the system can completely be described by just a single random variable being the total number of messages in the system. Note that, for the blocking protocol, $Y(t_n^+)$ is always zero, because extra messages are discarded. Let P_k be the probability of k messages being in the system

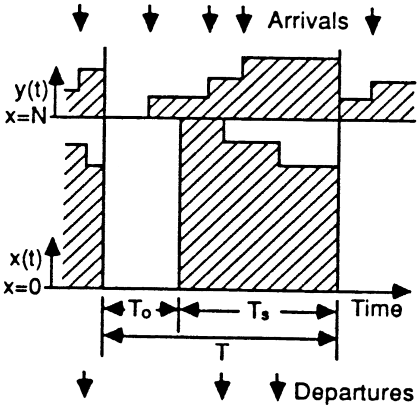

Fig. 4. Snapshot of queue fluctuations.

(CP + PP) at the imbedded instants, and let $\{q_{jk}\}$ represent the one-step transition probabilities. The state probabilities are calculated as follows:

$$P_k = \sum_{j=0}^{\infty} q_{jk} P_j, \qquad \sum_{k=0}^{\infty} P_k = 1. \tag{1}$$

The transition probabilities are determined jointly by the (Poisson) arrival rate and the service (departure) process during the previous clock period. Hence, we define

$$\begin{aligned} g_i &= \Pr\{i \text{ fresh arrivals in previous clock period}\} \\ &= (\lambda T)^i \mathrm{e}^{-\lambda T}/i!, \end{aligned} \tag{2}$$

$$d_r = \Pr\{r \text{ service completions in } (T_0, T)\} = (\mu T_S)^r \mathrm{e}^{-\mu T_S}/r! \quad (0 \leqslant r \leqslant N), \tag{3}$$

where $T_S = T - T_0$.

Note that, sometime during the clock period, the CP may become idle; equation (3) is implicitly conditioned on there being sufficient messages to serve. The significance of this is immediately apparent in the following analysis.

3.1.1. Nonblocking case

Here, there is no restriction on the total number in the system:

$$q_{jk} = \begin{cases} g_k \sum_{r=j}^{\infty} d_r + \sum_{r=j-k}^{j-1} d_r g_{k+r-j}, & 0 \leqslant j \leqslant N, \\ g_{k+N-j} \sum_{r=N}^{\infty} d_r + \sum_{r=j-k}^{N-1} d_r g_{k+r-j}, & j > N, \end{cases} \tag{4}$$

where terms with negative indices are set to zero, and summations where the upper limit is exceeded by the lower are also zero. The infinite summations of the departure probabilities $\{d_r\}$ represent cases where the server becomes idle during the clock period.

3.1.2. Blocking case

Here, the indices j and k cannot exceed N because excess messages are discarded at transfer epochs:

$$q_{jk} = \begin{cases} g_k \sum_{r=j}^{\infty} d_r + \sum_{r=j-k}^{j-1} d_r g_{k+r-j}, & k < N, \\ \sum_{r=j}^{\infty} d_r \sum_{i=N}^{\infty} g_i + \sum_{r=0}^{j-1} d_r \sum_{i=N-j+r}^{\infty} g_i, & k = N. \end{cases} \tag{5}$$

Again, the infinite sums of the departure probabilities represent cases where the server becomes idle.

For either case of the nonblocking or blocking protocol, the state probabilities are obtained from equation (1) with either (4) or (5) respectively, by the numerical technique of Gauss–Seidel iteration [5,7].

3.2. Delay analysis

The mean delays in the PP and CP queues are most easily derived through mean queue lengths and Little's law [5,7]. To do this it is necessary to know the state distribution at an arbitrary point in time, equivalent to a time average of the queue lengths. The method is most completely illustrated by looking at the nonblocking protocol, where we must consider the delays in both the PP and CP queues. We discuss the blocking protocol later.

3.2.1. Nonblocking case

Define

$$P_k(x) = \Pr\{X(t_n^+) = k;\ 0 \leqslant k \leqslant N\}, \qquad P_k(y) = \Pr\{Y(t_n^+) = k;\ k \geqslant 0\}. \tag{6}$$

These can be calculated by conditioning on the $\{P_k\}$ calculated in Section 3.1.

An outside observer sees the system at an arbitrary time, which will fall in the overhead period with probability $\pi_1 = T_0/T$, or into the active service period with probability $\pi_2 = 1 - \pi_1$. If the observation instant falls in the active service period, the d.f. of the time elapsed since the end of the previous overhead period is given by the backward recurrence time of the active service period, in this case a uniform distribution with mean $\frac{1}{2}T_S$.

The arbitrary time state probabilities seen by the outside observer are defined as $P_k^*(x) = \Pr\{X(t) = k\}$ and $P_k^*(y) = \Pr\{Y(t) = k\}$. In the nonblocking case, they can be calculated from

$$P_k^*(x) = \begin{cases} \pi_1 P_k(x) + \pi_2 \sum\limits_{j=k}^{N} P_j(x) d_{j-k}^*, & k > 0, \\ \pi_1 P_0(x) + \pi_2 \sum\limits_{j=0}^{N} P_j(x) \sum\limits_{r=j}^{\infty} d_r^*, & k = 0, \end{cases} \tag{7}$$

$$P_k^*(y) = \sum_{j=0}^{k} P_j(y) g_{k-j}^*, \tag{8}$$

where

$$d_r^* = \Pr\{r \text{ service completions in } (T_0, t)\}, \qquad g_i^* = \Pr\{i \text{ arrivals in } (0, t)\}.$$

Thus, the arbitrary time state probabilities can be calculated from the state probabilities of the IMC, allowing the calculation of the time-average means $EX = E[X(t)]$ and $EY = E[Y(t)]$. The mean queue delays EW_C and EW_P for the CP and PP queues respectively are then found from Little's law as

$$EW_C = EX/\lambda - 1/\mu, \qquad EW_P = EY/\lambda. \tag{9a, b}$$

The results of this sort of analysis are depicted in Fig. 5, where total mean delay is plotted as a function of the clock period T, with offered traffic intensity as a parameter. Clearly, the curves suggest an optimal choice of T. At small T, too much time is lost to overhead and, at large T, the CP is idle for part of the clock period because work cannot get in.

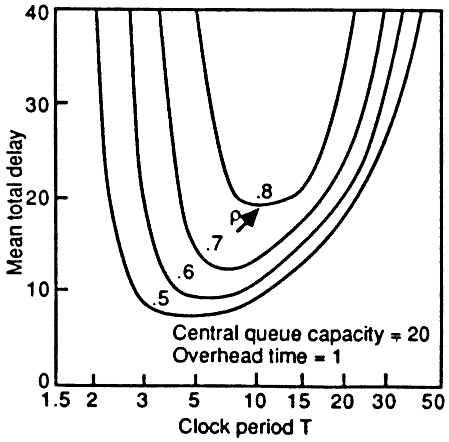

Fig. 5. Total mean delay vs. clock period—clock schedule (nonblocking).

3.2.2. Blocking case

The delay analysis of the blocking protocol is very similar, except that the delay in the peripheral queue is much more simple. The analysis for the delay in the CP is the same as for the nonblocking case, requiring the calculation of the state probabilities of the number in the system at an arbitrary time, and the application of Little's law. Numerically, the results are also similar, except that the total average delay in the system cannot grow without bound at high traffic (as we see in Fig. 5 for the nonblocking case). Instead, at high traffic, the bounded delay reflects the impact of discarding excess messages.

3.3. Delay distributions

Analysis of the distribution of delays is somewhat more difficult, and details of this can be found in [4,5,8]. The FIFO discipline is discussed in [5,7] and the ROS discipline in [4,7]. More complex models dealing with clocked transfers with priorities can be found in [2,3].

3.4. Stability analysis

In Fig. 5, upper and lower stability limits for the clock period T are apparent. These limits exist only in the case of nonblocking message transfer because, in the blocking case, the system is finite (size N) and by definition always stable.

For small T, the active service period must be large enough to process the expected number of arrivals in a clock period. Hence,

$$\lambda T_{\min} = \mu(T_{\min} - T_0), \qquad T_{\min} = T_0/(1 - \lambda/\mu).$$

For large T, the mean number of arrivals in the clock period cannot exceed the CP queue size,

$$T_{\max} = (S+1)/\lambda.$$

The system is then stable for $T_{\min} < T < T_{\max}$.

4. Event-driven schedules

The event-driven schedule is based on the idea that the event message generation process in the PP should be used to trigger a message transfer to the CP by means of an external interrupt. The rationale for this is that it is desirable to relate the frequency of message transfer attempts to the load generated by the PP. The event-driven schedule we consider here is that when n new messages have accumulated in the PP, a message transfer phase is initiated in the CP by interrupt. A major shortcoming of this strategy is that, at low traffic, delays can be very long [5]. To overcome this, it is necessary to protect the system messaging integrity by using a timeout, length T, after which a transfer is initiated anyway [6,9]. Protocols with timeouts are often very difficult to analyse from the performance point of view, and one of the major contributions of this work is that it does provide an analytical framework for handling a timeout.

4.1. Modelling assumptions

First we define the protocol operation and parameters and then give the modelling assumptions to be used in the subsequent performance analysis.

(a) Referring to the queueing model in Fig. 3, when n new messages have accumulated in the peripheral queue, or the timeout of duration T occurs, whichever comes first, the switch closes and messages are transferred to the central queue.

(b) The central queue is finite with S waiting places. The CP system size is $N = S + 1$.

(c) Whenever a transfer occurs, overhead time T_0 is incurred in the central processor as a result of the interrupt mechanism.

(d) If n messages accumulate in the peripheral queue before the last overhead period is ended, then this overhead period is curtailed and a new overhead period begun. This means that the transfer is 'gated', and this assumption is made for analytical tractability. In practical cases, it is apparent, however, that this assumption is not critical because there is only a remote possibility of n arrivals in a time period of length T_0.

4.2. Imbedded semi-Markov chain analysis

The extra difficulty imposed by the event-driven transfer schedule is that the inter-transfer intervals are a function of both the arrival process and a deterministic value (the timeout). Remaining with our standard notation, the state of the system at the instants $\{t_n^+\}$ are seen to constitute an IMC and, moreover, the sequence of inter-transfer interval types (as determined by the occurrence or nonoccurrence of the timeout) is seen to constitute an imbedded semi-Markov process.

By choosing the imbedded points just after the transfer epochs, the relationship between inter-transfer interval type and the size of the message batch is preserved. This is the most important observation of the modelling. The length of the inter-transfer interval and hence the batch size clearly depends on whether the n messages or the timeout triggers the transfer. Consideration of the inter-transfer interval type is the key to the following analysis.

As before, define the state probabilities $\{P_k\}$ of the number of messages in the system at the instants $\{t_n^+\}$ and the associated one-step transition probabilities $\{q_{jk}\}$. To determine the transition probabilities, it is necessary to distinguish among three types of message transfer intervals.

Case 1: *Timeout*. The inter-transfer interval is of length T and less than n messages are transferred.

Case 2: n messages in the interval (T_0, T), n transferred.

Case 3: n messages in $(0, T_0)$, n transferred.

Define γ_i to be probability of a transfer interval of type i. Conditioning on interval type, we have

$$q_{jk} = \sum_{i=1}^{3} \gamma_i q_{jk}^{(i)}, \tag{10}$$

where $q_{jk}^{(i)} = \{k \text{ at } t_{n+1}^+ \mid j \text{ in system at } t_n^+; \text{ interval type } i\}$ and

$$\gamma_1 = \bar{E}_n(T) = 1 - E_n(T), \qquad \gamma_2 = \bar{E}_n(T_0) - \gamma_1, \qquad \gamma_3 = E_n(T_0), \tag{11}$$

and $E_n(\cdot)$ is the special Erlang d.f. of order n with parameter λ and density $e_n(\cdot)$. The three types of interval have probability density functions (p.d.f.) $f_i(\cdot)$ and d.f. $F_i(\cdot)$, where

$$f_1(t) = \delta(t - T), \qquad f_2(t) = \begin{cases} e_n(t)/\gamma_2, & T_0 \leqslant t < T, \\ 0, & \text{otherwise}, \end{cases} \tag{12}$$

$$f_3(t) = \begin{cases} e_n(t)/\gamma_3, & 0 \leqslant t < T_0, \\ 0, & \text{otherwise}. \end{cases}$$

The number of service completions between transfers is given by the conditional departure probabilities $\{d_m^{(i)}\}$, where

$$d_m^{(i)} = \Pr\{m \text{ service completions} \mid \text{interval type } i, 0 \leqslant m \leqslant N\},$$

and are determined as follows:

Case 1—*Timeout*:

$$d_m^{(1)} = (\mu T_S)^m \, e^{-\mu T_S}/m!, \quad T_S = T - T_0. \tag{13a}$$

Case 2—n messages in (T_0, T):

$$d_m^{(2)} = \int_0^{T_S} \frac{(\mu x)^m \, e^{-\mu x}}{m!} f_2(x + T_0) \, dx$$

$$= \frac{\mu^m \lambda^n \, e^{-\lambda T_0}}{\gamma_2 m!(n-1)!} \sum_{i=0}^{n-1} \binom{n-1}{i} T_0^{n-i-1} \int_0^{T_S} x^{m+i} \, e^{-(\lambda+\mu)x} \, dx. \tag{13b}$$

The type of integral in this equation occurs many times in the subsequent analysis, and can be put into a tractable computational form by partial integration.

Case 3—n messages in $(0, T_0)$:

$$d_m^{(3)} = \begin{cases} 1, & m = 0, \\ 0, & \text{otherwise}. \end{cases} \tag{13c}$$

The generation of the transition probabilities is a somewhat tedious process but is not theoretically difficult. The set of possible transition paths between two imbedded points primarily depends on the interval type between them. This determines the distribution of the number of departures (service completions) represented by the coefficients $d_m^{(i)}$, and the number of arrivals. For inter-transfer intervals of types 2 and 3, the number of arrivals is exactly n. For an interval of type 1, the distribution of the number of arrivals is given by the probabilities $\{g_i\}$.

Define

$$\begin{aligned} g_i &= \Pr\{\text{arrival batch size } i \mid \text{interval of type } 1\} \\ &= (\lambda T)^i \, e^{-\lambda T}/Ki!, \quad 0 \leqslant i \leqslant n-1, \end{aligned} \tag{14}$$

where

$$K = \sum_{i=0}^{n-1} (\lambda T)^i \, e^{-\lambda T}/i!.$$

The transition probabilities also depend, of course, on whether the protocol is blocking or nonblocking. They follow from the same general principles as in Section 3.1, except that now it is necessary to distinguish between the transfer interval types as well.

4.2.1. Nonblocking case

Case 1—*Timeout*:

$$q_{jk}^{(1)} = \begin{cases} g_k \sum_{r=j}^{\infty} d_r^{(1)} + \sum_{r=j-k}^{j-1} d_r^{(1)} g_{k+r-j}, & j < N, \\ g_{k+N-j} \sum_{r=N}^{\infty} d_r^{(1)} + \sum_{r=j-k}^{N-1} d_r^{(1)} g_{k+r-j}, & j \geqslant N. \end{cases} \tag{15a}$$

Case 2—n messages in (T_0, T):

$$q_{jk}^{(2)} = \begin{cases} \sum_{r=j}^{\infty} d_r^{(2)}, & k = j + n - N, \\ d_{j+n-k}^{(2)}, & k > n. \end{cases} \tag{15b}$$

Case 3—n messages in $(0, T_0)$:

$$q_{jk}^{(3)} = 1, \quad k = j + n. \tag{15c}$$

The transition probabilities are used in equation (10) in conjunction with the power method to obtain a numerical solution to the state probabilities. While the state space is theoretically infinite, it is truncated to a suitable large value for purposes of numerical computation, such that the probability of the largest system state is small. It is not possible to use a more efficient Gauss–Seidel method because of the existence of transient states.

4.2.2. Blocking case

In the case of the blocking protocol, the state transition probabilities are expressed by terms very similar to those in equations (15a, b, c). The chief difference is that we must consider the transitions where the batch transfer overflows the CP queue and messages are discarded [9]. Note that the number of messages in the system at the epochs of the imbedded Markov chain is at most N. As in the nonblocking case, the state probabilities are found numerically by the power method, except that here the state space is finite and truncation is not necessary.

A performance measure of interest here is the probability of message discard on transfer. The state probabilities at the imbedded points can be used to calculate the probability of a message being lost, by conditioning on interval type [9].

4.3. Delay analysis

4.3.1. Central queue delay

The mean delay analysis for the CP queue builds on the method used in Section 3.2, through use of the arbitrary time state probabilities which give the distribution of the number of messages in the system at an arbitrary observation instant. This distribution is useful in providing a quick calculation of mean delay in the CP queue through the use of Little's law. Define $\{P_k^*; 0 \leqslant k \leqslant N\}$ to be the arbitrary time state probabilities. To calculate these it is necessary to know what type of transfer interval is seen by the outside observer at an arbitrary instant. Let π_i be the probability of type i. From the properties of the imbedded semi-Markov process we have

$$\pi_i = \gamma_i T_i / \sum_{j=1}^{3} \gamma_j T_j, \quad \text{where} \quad T_i = \int_0^\infty t f_i(t)\, \mathrm{d}t. \tag{16}$$

The arbitrary time state probabilities require a new set of departure probabilities denoted $\{d_m^{(i)*}; i = 1, 2, 3\}$

Case 1: *Timeout.* Given that the observer is in a type 1 interval, he is in the overhead segment with probability π_{11} and the "active" service (nonoverhead) segment with the complimentary probability π_{12} where

$$\pi_{11} = T_0/T = 1 - \pi_{12}.$$

The backward recurrence time for the active service segment has p.d.f.

$$\bar{F}_1(x)/T_S = 1/T_S, \quad 0 \leqslant x < T_S,$$

so that

$$d_m^{(1)*} = \frac{1}{T_S}\int_0^{T_S} \frac{(\mu x)^m \, \mathrm{e}^{-\mu x}}{m!}\, \mathrm{d}x = \frac{\mu^m}{T_S m!}\int_0^{T_S} x^m \, \mathrm{e}^{-\mu x}\, \mathrm{d}x. \tag{17a}$$

Case 2. Given that the observer is in a type 2 interval, he is in the overhead segment with probability π_{21} and the active service segment with the complementary probability π_{22}, where

$$\pi_{21} = T_0/T_2 = 1 - \pi_{22}.$$

Denote by $T_2' = T_2 - T_0$ the mean length of the active service segment of a type 2 interval,

$$\begin{aligned} T_2' &= \int_0^{T_S} x f_2(x + T_0)\, \mathrm{d}x \\ &= \frac{\lambda^n \, \mathrm{e}^{-\lambda T_0}}{\gamma_2 (n-1)!} \sum_{i=0}^{n-1} \binom{n-1}{i} T_0^{n-i-1} \int_0^{T_S} x^{i+1} \, \mathrm{e}^{-\lambda x}\, \mathrm{d}x. \end{aligned}$$

Finally,

$$d_m^{(2)*} = (1/T_2') \int_0^{T_S} \frac{(\mu x)^m \, \mathrm{e}^{-\mu x}}{m!} \bar{F}_2(x + T_0)\, \mathrm{d}x,$$

which after some manipulation comes to

$$d_m^{(2)*} = \frac{\mu^m}{T_2' m!} \sum_{i=0}^{n-1} \binom{n-1}{i} T_0^{n-i-1} \int_0^{T_S} t^i \, \mathrm{e}^{-\lambda t} \, \mathrm{d}t$$

$$- \sum_{i=0}^{m} \mu^i / i! \sum_{j=0}^{n-1} \binom{n-1}{j} T_0^{n-j-1} \int_0^{T_S} t^{j+i} \, \mathrm{e}^{-(\lambda+\mu)} \, \mathrm{d}t. \tag{17b}$$

Case 3. Here there is no possibility for departures.

The distribution of the number of messages in the CP queue at the last imbedded point is given by the probabilities $\{P_k'\}$, where

$$P_k' = \begin{cases} P_k, & 0 \leqslant k < N, \\ \sum_{r=N}^{\infty} P_k, & k = N. \end{cases}$$

(Note that in the blocking case we simply have $P_k' = P_k$ for all k.) The arbitrary-time state probabilities are found from

$$P_k^* = \pi_1 \left(\pi_{11} P_k' + \pi_{12} \sum_{j=k}^{N} d_{j-k}^{(1)*} P_j' \right) + \pi_2 \left(\pi_{21} P_k' + \pi_{22} \sum_{j=k}^{N} d_{j-k}^{(2)*} P_j' \right)$$

$$+ \pi_3 P_k', \quad 0 < k \leqslant N,$$

$$P_0^* = \pi_1 \left(\pi_{11} P_0' + \pi_{12} \sum_{j=0}^{N} P_j' \sum_{r=j}^{\infty} d_r^{(1)*} \right) + \pi_2 \left(\pi_{21} P_0' + \pi_{22} \sum_{j=0}^{N} P_j' \sum_{r=j}^{\infty} d_r^{(2)*} \right)$$

$$+ \pi_3 P_0' \quad (k = 0). \tag{18}$$

Finally, the expected delay in the CP queue EW_C is found from Little's law as

$$EW_C = W - 1/\mu, \tag{19}$$

where $W = L/\lambda$ and $L = \Sigma_{k=0}^{N} k P_k^*$.

4.3.2. *Peripheral queue delay*

The mean PP queue delay (transfer delay) is made up of two components: the mean delay from message arrival until the next transfer epoch denoted by the expectation W_{P1}, and the mean delay from the first transfer epoch until that transfer epoch the message is put into the CP queue denoted by the expectation W_{P2}. This decomposition is possible because, except for fresh arrivals, the number of messages in the peripheral buffers remains constant between transfer epochs.

Here it is necessary to distinguish between the blocking and nonblocking protocols. In the blocking case, only the term W_{P1} has any significance.

4.3.2.1. *Nonblocking case*

From Little's law,

$$W_{P2} = \sum_{k=N+1}^{\infty} (k - N) P_k / \lambda. \tag{20}$$

The other component depends on the interval type. Given there is no blocking of messages, we have

$$W_{P1} = \sum_{i=1}^{3} a_i W_{P1}^{(i)}, \tag{21}$$

where

$$a_i = \Pr\{\text{arbitrary message arrives in interval type } i\}$$
$$= \gamma_i S_i / \sum_{j=1}^{3} \gamma_j S_j,$$
$$S_i = \begin{cases} \sum_{k=0}^{n-1} k g_k, & i = 1, \\ n, & i = 2, 3. \end{cases}$$

The probabilities $\{a_i\}$ are based on the weighted sum of the expected number of arrivals in each interval type, according to the imbedded semi-Markov process.

$$W_{\mathrm{P1}}^{(i)} = \begin{cases} \tfrac{1}{2}T, & i = 1, \\ (n-1)/2\lambda, & i = 2, 3. \end{cases}$$

The delay for interval types 2 and 3 follows from assumption (d) in Section 4.1. Finally,

$$EW_{\mathrm{P}} = W_{\mathrm{P1}} + W_{\mathrm{P2}}. \tag{22}$$

4.3.2.2. Blocking case

Here, only the delay until the first transfer is of interest. However, the calculation of mean delay is a little complicated because it is most useful to find the mean peripheral delay conditional on acceptance in the CP queue. Details of this calculation may be found in [9].

4.3.3. Numerical results

Fig. 6 depicts the total mean delay versus the timeout period for the case of the nonblocking protocol. Results are shown for two different values of n, and the optimizing effect of the timeout is apparent. For large timeout, the delay flattens out since it becomes limited by the batch size n. Note the relative insensitivity of the delay to the choice of timeout (for reasonable n) as compared to the clocked transfer. This illustrates the improved performance that can be obtained by more complex event-dependent transfer protocols, which exhibit some adaptivity to changing traffic conditions. Comparing the clocked scheme with the event-driven one through the results in Figs. 5 and 6, we can see the sensitivity of the clocked

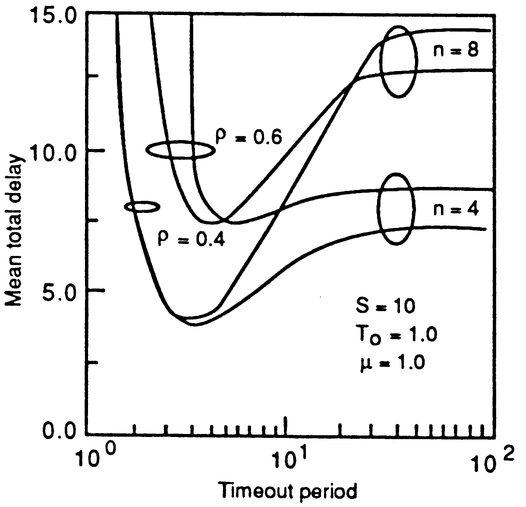

Fig. 6. Total mean delay vs. timeout period—event-driven schedule (nonblocking).

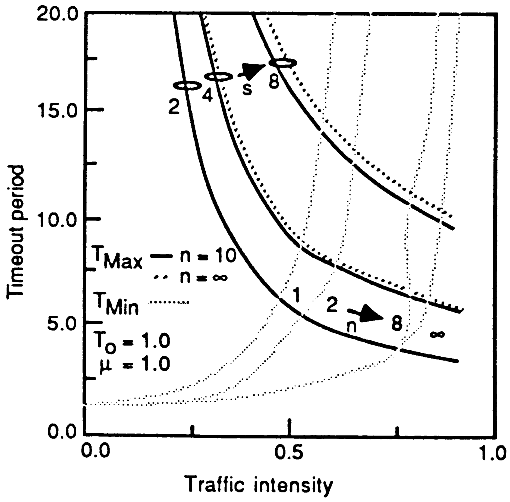

Fig. 7. Stability limits for event-driven schedule (nonblocking).

transfer to the choice of the clock interval, whereas the 'adaptive' event-driven scheme is reasonable over a broad range of timeout (the equivalent parameter). This tolerance has particular importance in real-time systems where the traffic conditions can vary widely with time.

Numerical results for message delay for the blocking protocol are qualitatively very similar to those for the nonblocking one where, depending on the choice of parameters, an optimum value of timeout is perceived. From the delay performance comparison, there is no compelling reason to distinguish between the blocking and nonblocking protocols, and the choice can be based on implementation considerations.

4.4. System stability

There are two conditions deciding system stability. First, the expected number of fresh arrivals in an arbitrary inter-transfer interval may not exceed the expected number of service completions. Hence,

$$\gamma_1\lambda T + n(\gamma_2 + \gamma_3) < \gamma_1\mu T_S + \gamma_2\mu T_2. \tag{23}$$

Second, the expected number of fresh arrivals cannot exceed the CP system size N. Hence,

$$\gamma_1\lambda T + n(\gamma_2 + \gamma_1) < N. \tag{24}$$

Equations (23) and (24) describe functional relations which must iteratively be solved for the protocol parameters (n, T). For a given value of n, equation (23) and (24) define minimum and maximum values of T, labelled $T_{\min}$ and $T_{\max}$ respectively. If n becomes large, the stability limits for the clocked schedule, as given in Section 3.3, apply. Also note that, for $n < S$, $T_{\max}$ goes to infinity, and in practice this will usually be the case. Results for the stability limits are depicted in Fig. 7.

5. Conclusions

This paper has introduced a set of queueing models and analysis techniques for the performance analysis of two classes of inter-processor message transfer protocols in distributed systems. Numerically tractable methods have been developed for the calculation of the important performance indices so that different design options can be evaluated and the protocol parameters tuned for optimum system performance (e.g., minimum message delay). The key results are the improvement in system performance which can be obtained by passing messages in batches (to minimize the overhead associated with the message transfer mechanism), and the improved delay performance which results by ensuring a maximum time between successive transfer epochs. The numerical techniques presented here may be applicable to a broader range of problems, such as the performance analysis of TDMA transmission systems.

Acknowledgment

The authors would like to thank Prof. P.J. Kuehn for useful discussions about this work, and M. Weixler for programming support.

References

[1] P.J. Burke, Delays in single server queues with batch input, *Bell System Tech. J.* **23** (1975) 830–833.

[2] H. Jans, Traffic analysis of switching system control structures with clocked I/O and priorities, *37th Report on Studies in Congestion Theory* (Inst. of Switching and Data Technics, Univ. Stuttgart, 1983).

[3] H. Jans, On queueing systems with clocked operation and priorities, *10th Internat. Teletraffic Conf.*, Montréal (1983) paper 4.4a.4.

[4] D. Manfield, Scanning processes and dial tone delay in SPC systems, *Proc. 2nd GI/NTG Technical Conf. on Modelling and Performance of Computer Systems*, Stuttgart (1983) 178–188.

[5] D. Manfield and P. Tran-Gia, Queueing analysis of scheduled communications phases in distributed processing systems, in: F.J. Kylstra, ed., *Performance '81* (North-Holland, Amsterdam, 1981) 233–250.

[6] D. Manfield and P. Tran-Gia, Queueing analysis of an arrival-driven message transfer protocol, *10th Internat. Teletraffic Conf.*, Montréal (1983) paper 4.1.4.

[7] P. Tran-Gia and H. Jans, Clocked event transfer protocol in distributed processing systems—a performance analysis, *Proc. ICCC*, London (1982) 769–774.

[8] P. Tran-Gia and H. Jans, Delay analysis of clock-driven message transfer in distributed processing systems, *Arch. Elektron. Übertragungstech.* **39** (1985) 285–292.

[9] P. Tran-Gia and D. Manfield, Performance analysis of communication delay optimization in distributed processing environments, *Proc. 2nd Internat. Symp. on Performance of Computer Communication Systems*, Zürich (1984) 259–274.