An Adaptive Buffer Management Algorithm for Enhancing Dependability and Performance in Mobile-Object-Based Real-time Computing

May T.W. Ip, Wilfred W.K. Lin, Allan K.Y. Wong, Tharam S. Dillon and Dianhui Wang
Department of Computing, Hong Kong Polytechnic University, Hunghom, Kolwoon
Hong Kong SAR, P.R. China, Email: cstwip@comp.polyu.edu.hk Fax: (852) 27740842

Abstract

In an ORC (object-oriented real-time computing) environment, it is important to reduce message retransmissions during message passing because these retransmissions cause significant time delay. And, this makes it difficult to achieve the necessary timeliness. A cause of retransmission is message loss due to buffer overflow at the reception side. Here a P+I+D (P for proportional, I for integral and D for derivative) adaptive buffer control algorithm is proposed to prevent such possible overflow. The I control depends on the Convergence Algorithm (CA), which is a stable and efficient IEPM (Internet End-to-End Performance Measurement) tool that predicts the mean message roundtrip time (RTT) of a communication channel quickly and accurately. The P+I+D algorithm was tested under different conditions in a mobile ORC (MORC) environment, where mobile agents collaborate freely over the Internet. The different test results confirm that the proposed P+I+D approach is indeed effective for preventing buffer overflow.

Keywords: MORC, buffer overflow, P+I+D control algorithm, adaptive, Convergence Algorithm

1. Introduction

In object-oriented real-time computing (ORC), program objects collaborate as a logical layer over a physically distributed network. If these objects communicate by message passing in an asynchronous manner, then the time delay in request/response protocols can seriously impede the performance and timeliness of the system. In fact, the success of an ORC system depends not only on the correct logical answer but also on the timeliness of the computation result. In hard real-time ORC systems, the expected results must be produced before the deadline to prevent catastrophic effects. In soft ORC systems, it is desirable to have results produced before the deadlines, but some deadline slippage in this case is not harmful. In firm ORC systems, a task is executed only if it could be completed before the deadline because the result produced after that is meaningless [1].

If collaborative sensors and controllers that feed essential data to the logical objects or the decision making algorithm do locate at the same site (e.g. within the compound of a factory), then a dedicated network would suffice the real-time processing requirement. If the information system is dispersed due to geography, as exemplified by a electrical power plant, the real-time requirements are quite different because local controllers and sensors at different sites are distant from each other, and yet the real-time information has to be manipulated centrally. One way to do this is to put in place a dedicated data communication system that brings the dispersed information together, but this is an expensive approach. One alternative is to make use of the Internet and in this case the important issue is to resolve the message passing time delay due to the random nature and sheer size of the underlying network. The solution then is to develop effective techniques to ensure achievement of the necessary timeliness for ORC success. In ORC systems, service users and service providers collaborate and interact in a client/server relationship [2], and the client invokes a service through message passing as depicted in Figure 1.

Figure 1. The ORC client/server relationship model

In the client/server relationship an object can act as a service user (client) and service provider (server) at the same time. As a server it provides service to many different clients in the network through asymmetric rendezvous (client/server interaction). For conceptual simplicity, it is assumed in our project that every logical
server in a host would maintain a buffer to house the input request queue. This phenomenon can be depicted by the M/M/I (M for Markov) model [3], in which different random streams of requests can be combined into a single logical random input stream of average rate \( \lambda_r \) by the rule of superposition; that is, \( \lambda_r = \lambda_1 + \lambda_2 + \lambda_r + \ldots \)

If the utilization \( \rho \) for the logical server is \( \rho = \frac{\lambda_r}{\mu} \), where \( \mu \) is the server's average service rate (capacity), then the average number of jobs in the server queue \( L \) can be approximated by:

\[
L = \frac{\rho}{1 - \rho} \frac{\lambda_r}{\mu - \lambda_r}
\]

leads to a shorter \( L \).

In a mobile ORC (MORC) environment, which is the main interest of the present research, client and server objects move around for various reasons. For example, an object may try to move out of an undependable server node for self-preservation. Making use of object mobility for system dependability is an attractive approach. We have addressed this issue in detail and elaborate its potential advantages in our previous work [4,5]. Object mobility affects the values of both \( \lambda_r \) and \( L \) because every location (host node) has a different traffic characterization [6]. When a logical server has moved to a new location, the local traffic pattern and channel/link quality can change \( \lambda_r \) and \( L \), and this leads to queue overflow and message loss. The resultant message retransmissions would yield poor system performance and unreliability [7]. If \( \sigma \) is assumed to be the error probability that causes retransmissions and message transmission is successful at the \( j^{th} \) trial, then the average number (expectation) of transmissions (ANT) to pass a message through a channel successfully is given by equation (1.1).

\[
\text{ANT} = \sum_{j=1}^{\infty} j \sigma^{j-1} (1 - \sigma) = \frac{1}{1 - \sigma} \quad (1.1)
\]

It means that if \( \sigma = 0 \), then the number of message transmissions to have a success is one (i.e. 100% channel dependability). The corollary is that a larger \( \sigma \) means more retransmissions and thus longer response times. This causes sluggish performance by the client because it takes longer to complete the I/O operations that are necessary for the overall computation. In the MORC sense, a large ANT implies that the client may not be able to produce the necessary final computation result before the deadline because of various delays in service responses by service providers. If this happens to many collaborating mobile objects in a MORC application, then the overall application is not dependable at all [7]. There are many possible factors that contribute to the value of \( \sigma \), including excessive routing time, message loss in the transmission process, message corruption, and queue overflow at the server side. Different methods have been proposed to address the above issues by different researchers (e.g. [8,9,10]).

In this paper, we concentrate mainly on the issue of how to adaptively manage the buffer size to support the queue of service requests at the server side in a MORC environment. The aim is to eliminate queue overflow that leads to frequent retransmissions caused by message loss.

2. Related Work

In the last few years, there has been active research in different ORC areas, including the following: a) distributed real-time Java (e.g. [11]), b) fault tolerance (e.g. [7,12]), c) deadline scheduling (e.g. [13]), and d) real-time firmware settings such as DCOM and CORBA. Gradually, as indicated by the nature of most of the research, ORC will also include the Internet, which inherently imposes many timeliness and reliability problems for real-time computing due to its sheer size and random nature.

\[
\begin{align*}
\text{If } (dQ/dt &> \text{prescribed_positive_threshold}) \text{ OR } \\
(dQ/dt &< \text{prescribed_negative_threshold}) \text{ AND } \\
\text{RQL &> prescribed_positive_threshold}) &) & \\
\text{then } L_{\text{min}} = L_{\text{max}} - B_0, \text{ } L_{\text{max}} \geq L_{\text{minimum}}
\end{align*}
\]

Else \( \text{if } (dQ/dt < \text{prescribed_negative_threshold}) \text{ OR } \\
(dQ/dt \text{ is_ negative}) \text{ AND } \\
\text{RQL < prescribed_positive_threshold}) \text{) then } \\
L_{\text{min}} = L_{\text{max}} - B_0, \text{ } L_{\text{max}} \geq L_{\text{minimum}}
\]

Figure 2. The original P+D algorithm

This research is based on our findings described in [7] and [10]. The main idea is that if one can adaptively manage the buffer at the receiver (server) side to prevent message loss due to buffer overflow, then the elimination of retransmissions by such message loss will enhance both system performance and dependability due to more reliable channels. This idea is important for MORC computing, in which objects or agents migrate for various reasons. The quality of the service, in terms of correctness, timeliness and reliability should remain independent of the traffic characterization of the location. In [7] and [10] the P+D (P for proportional control and D for derivative control) adaptive buffer management algorithm was introduced (Figure 2). In this algorithm the current server's buffer length \( L_{\text{max}} \) will be changed adaptively to prevent overflow. If \( dQ/dt \) (rate of change of the queue length for the derivative control) is consistently
larger than the prescribed threshold, then \( L_{\text{new}} \) is increased adaptively by \( B_a \), which is either a constant or a mathematical expression. Similarly, if \( dQ/dt \) is consistently smaller than the prescribed threshold, \( L_{\text{new}} \) is decreased adaptively by \( B_b \). The proportional control \( P \) is indicated by the use of the RQL ratio defined as "current queue length \( Q_{\text{new}} \) over current buffer length \( L_{\text{new}} \)." When the \( P+D \) algorithm was verified in our laboratory with a setup similar to Figure 1, where the client and the server were nodes within the same Intranet, it eliminated queue overflow in all the test cases. However, when the same algorithm was used over the Internet with distant Internet sites (e.g., LaTrobe University in Australia), overflows, as shown in Figure 3, were widespread. Our analysis of the test results from field trials led to the conclusion that these overflows were caused by the time delay in communication channels; the algorithm used out-dated data to predict the future buffer trend. This means more effective overflow anticipation is needed.

When the \( P+D \) algorithm was verified in our laboratory with a setup similar to Figure 1, where the client and the server were nodes within the same Intranet, it eliminated queue overflow in all the test cases. However, when the same algorithm was used over the Internet with distant Internet sites (e.g., LaTrobe University in Australia), overflows, as shown in Figure 3, were widespread. Our analysis of the test results from field trials led to the conclusion that these overflows were caused by the time delay in communication channels; the algorithm used out-dated data to predict the future buffer trend. This means more effective overflow anticipation is needed.

The motivation of the present research is to improve the responsiveness of adaptive buffer management by introducing integral (I) control, that is, changing from \( P+D \) control to \( P+I+D \) control. Integral control means registering the history of buffer change and incorporating this history into the control process. If the recent history indicates that the queue length is on the up trend, then the buffer should be elongated. Likewise, if the queue length is on the downturn, the buffer length should be reduced to save memory resources. The \( P+I+D \) idea in this paper is borrowed from the traditional concept of control theory, but it differs by not emphasizing on the system gain due to the feedback loop. It concentrates mainly on how to use the \( P, D \) and \( I \) mechanisms to ensure that the buffer always covers the queue length.

Our previous experience in the IEPM (Internet End-to-End Performance Measurement) work [15] has suggested that we can adapt the IEPM approach to predict the queue behaviour dynamically. Therefore, the integral control mechanism in the proposed \( P+I+D \) control is based on the Convergence Algorithm (CA) IEPM approach [15]. We will use only the "micro" version adapted from original CA tool. In this version, the CA is invoked as another object for providing service in a real-time manner by other objects through message passing. This contrasts to the "macro" approach (e.g., PingER) in which the IEPM tool must be set up in nodes that represent the two ends of the end-to-end concept [16].

3. Background Theory

The two major elements in this research are, namely, a) utilization of the CA approach to predict the mean of a distribution quickly, accurately and dynamically, and b) the formulation of the \( P+I+D \) algorithm. Here, some relevant terms are defined as follows:

a) Typical queue length (TQL): It is the mode of the distribution of queue lengths over time.

b) Ratio for integral correction (RIC): It is the "ratio of current queue length fluctuation over the TQL". The RIC will be explained in detail later.

c) Ratio of queue length to buffer length (RQL): It is the ratio of "current queue length \( Q_{\text{new}} \) over current buffer length \( L_{\text{new}} \)" and is calculated by the instantaneous values.

3.1 The Convergence Algorithm

The Convergence Algorithm or CA is defined by the equations: (3.1) and (3.2). It incorporates the history or past prediction represented by \( M_{i,j} \), which is the average queue length predicted in the last cycle; \( i \) indicating the cycle. \( M_0 \) is the initial value when the prediction cycle starts and it is normal to set it to the first sampled value of \( m_{i,j} \). The other parameters in equation (3.1) are:

- \( m_{i,j} \) is the sampled queue length in the current prediction cycle marked by \( i \). The parameter \( j \) marks every queue-length sample used in the cycle.
- \( F \) is the flush limit, and \( (F-1) \) is the number of queue-length samples used in every \( M_i \), which is the predicted mean \( M \) at the \( i^{th} \) prediction cycle.

The CA prediction is insensitive to the nature of the distribution because the algorithm is based on the Central Limit Theorem. It provides optimal predictions because its simplicity means less computation time (i.e., less delay) so that a prediction actually reflects the physical reality.
In order to provide some insight into how the CA predicts with a distribution, the result from one of our previous IEPM experiments is shown in Figure 4. The CA in this case yielded two results from the message roundtrip time (RTT) of an Internet channel. The Estimated Mean is the predicted mean RTT and the Typical Mean is the mode of the RTT distribution. The Normal Mean and the Overall Mean are for control purposes in the experiment. The Estimated Mean converged quickly to the Typical Mean (800 µs) after the perturbations.

\[ M_{i-1} + \sum_{j=1}^{F-1} m_i^j = M_i \] \hspace{1cm} (3.1)

\[ M_0 = m_i^{j=0} \] \hspace{1cm} (3.2); \ i \geq 1

3.2 Incorporation of Integral Control

The logic of the P+I+D control mechanism is shown in the pseudo-program in Figure 5. The addition and subtraction of the buffer length are determined by the overall value currently accumulated (integration) by the ICM (Integral Control Measure).

If \((dQ/dt > \text{prescribed positive threshold}) \ \text{OR} \ \ [(dQ/dt \text{ is positive}) \ \text{AND} \ \ (RQL > \text{prescribed positive threshold})]\) then \(L_{now} = L_{now} + ICM; \ L_{now} \geq L_{minimum}\)

Else If \((dQ/dt < \text{prescribed negative threshold}) \ \text{OR} \ \ [(dQ/dt \text{ is negative}) \ \text{AND} \ \ (RQL < \text{prescribed positive threshold})]\) then \(L_{now} = L_{now} - ICM; \ L_{now} \geq L_{minimum}\)

Figure 5. The P+I+D control algorithm

3.2.1 The Integral Corrective Measure. The RIC, which is used for formulating the ICM, is a ratio defined by equation (3.3). The ICM integration process is shown in equation (3.4), where \(B_o\) is the unit for addition to or subtraction from the buffer length in an adaptive manner as depicted in Figure 5.

\[ RIC_i = \frac{\text{Queue}_{\text{CA, estimated}} - \text{Queue}_{\text{typical}}}{\text{Queue}_{\text{typical}}} \] \hspace{1cm} (3.3)

\[ ICM = B_o \sum_{i=1}^{N} RIC_i \] \hspace{1cm} (3.4)

Theoretically, the RIC can be viewed as a sinusoidal signal and the summation of all function or \(y(t)\) values in a cycle in a symmetric case should be zero. This sinusoidal signal however may be shifted upward or downward from the ideal case of \(y=0\) in reality, and this would yield a non-zero sum for every cycle. Equation (3.3) may cause a “divide by zero” problem when \(\text{Queue}_{\text{typical}}\) happens to assume the zero value. For most of the time, the ICM would oscillate according to the RIC values.

3.2.2 The Revised ICM. From the different experimental results obtained by pinging the distant LaTrobe University Internet site in Australia, we observed that the mode of the queue length distribution could assume a zero value. This is clearly demonstrated by the two occasions shown in Figure 6 and Figure 7, which are plotted with data collected from the same channel in the same experiment but at two different periods, namely, \(T_1\) and \(T_2\). This phenomenon implies infinite RIC value that leads to system instability. To get rid of this undesirable phenomenon, we revise equation (3.3) to the form of equation (3.5) by replacing \(\text{Queue}_{\text{typical}}\) by \(\text{Queue}_{\text{average}}\), which is the mathematical average of the queue lengths (marked Overall Mean in Figure 4). The argument for using the mathematical average arises from the fact that the queue length of any active channel would always assume some non-zero values. This means that the chance for \(\text{Queue}_{\text{average}}\) of an operational channel to assume a zero value is very unlikely.

Figure 6. Mode of the queue length is zero for period \(T_1\)
Results from different experiments with the LaTrobe site have concluded that the RIC values by equation (3.5) indeed follow the trend of the queue length closely, as shown in Figures 8 and 9. These graphs are plotted with data from the same experiment but collected in two different periods (T1 and T2). The aim is to demonstrate that the RIC trend always matches with that of the queue length in all test cases.

Figure 7. Mode of queue length is zero for period T2

\[
RIC = \frac{\text{Queue}_{\text{ca, revised}} - \text{Queue}_{\text{average}}}{\text{Queue}_{\text{average}}} \quad (3.5)
\]

As demonstrated in Figure 10, the CA senses and follows responsively the dramatic rises and drops in queue length perturbations, but the mathematical average always lags behind. The responses to the sudden rises and drops are however uneven. For example, in Figure 10, for a sudden rise in the current queue length, the CA estimate would respond more dramatically than the mathematical average. For a sudden drop, however, the difference between the responses by the average queue length and the CA estimate is much milder. These uneven responses leads to a skew phenomenon (Figure 11) in which the \( \sum_{\text{t}} RIC_{\text{t}} \) term yields skew large positive or negative values (Figure 11) leading to an uncontrollable ICM.

3.2.3 Addition of the Window Mechanism. For maintaining a reasonable RIC value, a “window mechanism (WM)” is added to equation (3.5) to resolve the skew phenomenon and this addition produces equation (3.6). The main idea is to compute the average RIC values, namely, the meanRIC within a scrolling window of chosen size Wsize. The meanRIC is the foundation for calculating the revisedICM or RICM as shown in equation (3.7).
meanRIC = \sum_{n=0}^{Wsize} RIC / Wsize \quad (3.6)

RevisedICM = \begin{cases} B_a * \text{meanRIC, if } \text{meanRIC} > I \\ B_a, \text{ otherwise} \end{cases} \quad (3.7)

3.2.4 Validation of the Final P+I+D Control. The validation of the new P+I+D control mechanism that works with the RICM was performed with the remote LaTrobe Internet site. The results from two of the experiments are shown in Figure 14 and Figure 15. In the experiments the final P+I+D buffer control mechanism always adaptively and physically covered the queue lengths. The parameters used in all the tests are threshold = 0.003, L = 20, and B_a = L/3.

As shown in Figure 14 (Case 1), the buffer length by P+I+D has less chance of overflow than the P+D control because the anticipative power of I based on the past history of the responses. This power comes from the fact that the recent history indicates the trend of change, either upward or downward. For example, Figure 11 indicates a positive upward trend and therefore proactive measures must be taken to prevent buffer overflow. The lack of past history in the P+D approach makes it a less anticipative mechanism. Take point E on the graph in Figure 14 as an example, the buffer length by P+D cuts more dangerously close to the actual queue length than that by the P+I+D control. A similar phenomenon is also observed in Figure 15 (Case 2) at point F. The validation exercise has confirmed that the P+I+D is indeed a better adaptive buffer management solution for preventing message loss due to overflow.
4. Conclusion

It is not always possible to have a dedicated network to support an ORC system. For example, the dispersed nature of an electrical power plant may need the support of a sizeable network such as the Internet. It is natural to have collaborating distributed logical objects to monitor the sensors and controllers in parallel. In this case sophisticated techniques are needed to reduce time delay so as to guarantee the essential timeliness of the system. In certain ORC applications it is more advantageous to let objects migrate to where the large volume of data are, and this is the basis of mobile ORC or MORC systems. For objects in a MORC system that runs over the Internet to interact by message passing, buffer overflow at the reception side can be frequent due to the time delay, which is intrinsic to the sheer size of the network. This time delay would cause message loss and thus widespread message retransmissions, which in turn causes longer ANT for communication channels and overall sluggish system performance. From our experience, the P+D adaptive buffer management algorithm, which was developed previously, is not responsive enough for many real-life situations. This has motivated our inclusion of the integral (I) control mechanism (ICM) in this project to convert the P+D approach into the novel P+I+D mechanism proposed in this paper. From the validation exercise we conclude that the RICM, which is based on meanRIC that works with a scrolling window of predefined size $W_{size}$, is undoubtedly an effective basis for formulating the final P+I+D algorithm. This algorithm adaptively and responsively prevents buffer overflow for all test cases over the Aglets mobile agent platform, which provides a stable MORC testing environment. In the validation exercise all the parameters such as $B_0$ and $W_{size}$ were set to predefined values. Although this parameter setting approach satisfies the present experimental needs, it is not effective for solving real-life problems because the operation conditions are dynamic. It is necessary to adjust these parameters automatically and continuously to cope with the dynamics of a large network within the predefined safety margin. Therefore, the next step in this on-going research is to investigate how soft computing and neural controller can be utilized to deal with the issues of automatic adjustment of parameter values and maintenance of a safety margin between the queue length and the buffer length in a proactive fashion.

5. Acknowledgement

We thank the Hong Kong PolyU for funding the research (grants: GW110, GW057, HZJ91 and APB89).

6. References