M³RT: An Internet End-to-End Performance Measurement Approach for Real-time Applications with Mobile Agents

Allan K.Y. Wong¹, May T.W. Ip¹ and Tharam S. Dillon²
¹Department of Computing, Hong Kong Polytechnic University, Hong Kong SAR, P.R. China, email:csalwong@comp.polyu.edu.hk Fax: (852) 27740842
²Department of Computer Science and Computer Engineering, LaTrobe University, Australia

Abstract
In this paper the M³RT (Micro Mean Message Roundtrip Time) IEPM (Internet End-to-End Performance Measurement) tool is proposed for supporting agent/object mobility in mobile-agent-based real-time applications. If an agent finds that the service roundtrip time (RTT) is too long, then it may try to cut it down by migrating to another idle node. Since migration is an expensive activity, the agent should decide correctly before making the move. One way of achieving sound decision making is to measure the mean RTT of the channel and to ensure that the RTT up-trend is not a transient phenomenon. The M³RT works on past history accumulated since the commencement of the channel operation. It never has computation overflow because of its integral nature. In the M³RT development process the corresponding time/stochastic Petri-Net model was built for exhaustive verification. The aim is to ensure that convergence stability is there under all conditions. The input waveforms used in the simulations mimic the dynamic reality of a sizeable network such as the Internet. Preliminary and limited validation tests in a controlled environment indicate that the M³RT works equally well as its macro predecessor, namely, the stable M²RT (Mean Message Roundtrip Time) package.

Keywords: M³RT, IEPM, Petri-Net, Internet, microscopic application, mobile agents

1. Introduction

The M³RT (Micro Mean Message Roundtrip Time) is an IEPM (Internet End-to-End Performance Measurement) tool for microscopic (micro) applications. Micro applications differ from macro ones in two aspects: a) they do not require user intervention, and b) they exist as logical service providing entities that can be invoked by message passing. The M³RT is adapted from its macro M²RT predecessor [1] to support mobile-agent based applications running over a controlled portion/subnet of the Internet. One example of a controlled portion is the “Public Intranet (PIN)” [2] for Internet-based real-time computing. A PIN is a subnet formed by dynamic clustering of voluntary Internet nodes. Any node can join and leave at will through a specially designed registration system. If a node is a PIN member, it can move around within the PIN boundary and share the pool of public resources. The PIN concept carves a subset out of the huge Internet for practical real-time applications because it is easier to control the timeliness of a smaller subnet.

Extant IEPM tools are macro by nature, which means that the user must install the tool at two nodes that represent the ends of a logical channel. The original goal of these macro tools is to predict and characterize channel performance by analyzing the data samples (e.g. roundtrip times). The results obtained by different previous approaches prove that the IEPM methodology is indeed effective [3,4,5,6,7]. Until this moment, however, no stable micro IEPM (MIEPM) tool can be identified. To be called micro, an IEPM tool must possess the following attributes:
a) It can be invoked asynchronously by running logical objects in a many-to-one fashion (asymmetric rendezvous client/server relationship) or a one-to-one dedicated fashion.
b) It should work independently of the network hardware.
c) It should measure/predict quickly and accurately to avoid any deleterious effects. If the prediction is too long, then with the result, the client may unknowingly respond to history rather than the reality. This kind of mistake is called deleterious effects in the M³RT context.

The usefulness of a MIEPM tool covers many areas. One example is when a client mobile agent is trying to seek quick service from the nearest cognate agent server with the PIN. Since these agents are constituent components of the same distributed program [8,9], they may collaborate in an itinerary of request/response steps that indicate the programmed agent/data dependency [10]. To show the impact of shorter itinerary paths, which consist of intermediate roundtrip times, namely, the RTTP, on the overall program execution time T_OPE, a scenario is...
created in Figure 1 for demonstration. This scenario is based on the recursive option [11] of the master/slaves (MS) paradigm. In this option slaves are nested in a client/server relationship to a considerable depth.

![Diagram of an itinerary execution scenario based on recursive MS](image)

Figure 1. An itinerary execution scenario based on recursive MS

When this MS program is running on a mobile-agent platform over the Internet (e.g. Aglets [10]), slave agents collaborate for specific itinerary paths. For example, the agents: Master, Slave A and Slave B make up such a path. The intrinsic agent mobility supported by the platform enables agents to migrate to idle nodes for shorter service RTT by load balancing. If RTTP is the average interval for every “hop” between two agents, then the total roundtrip time for an itinerary path RTT_{ip} is

\[ \text{RTT}_{ip} = \sum_{j=1}^{M} \text{RTT}_{p_j} \]

where N represents the depth of the path (number of constituent slaves/agents). The overall program execution time T_{OPE} for the MS program becomes

\[ T_{OPE} = [\frac{1}{M} \sum_{k=1}^{M} \text{RTT}_{q_k}]^\beta M^{-\beta} \]

or

\[ T_{OPE} = [\sum_{k=1}^{M} \text{RTT}_{q_k}]^\beta M^{-(1+\beta)} \]

where \beta is the “degree of overlapped parallelism among the itinerary paths” [1,12]. When \beta is very large, indicating high degree of overlap then T_{OPE} converges to

\[ T_{OPE} = [\sum_{k=1}^{M} \text{RTT}_{q_k}] \]

where the number of itinerary paths in the MS program, and \beta is the degree of overlapped parallelism among the itinerary paths” [1,12]. When \beta is very large, indicating high degree of overlap then T_{OPE} converges to

There are many issues to be considered for cost/effective object migration such as the cost function, finding the nearest idle node, and answering the question of whether the RTT_{p} is persistently becoming longer. A MIEPM tool, which is invoked by other objects/agents for service in a real-time manner by message passing, can help determine the nearest idle node. The determination can be based on whether the RTT_{p} up-trend is sharp and persistent. It is important, however, for the prediction to be dependable in the sense that it should optimally reflect the physical reality on an integral basis. A migration decision on current data alone may lead to spurious responses to random noise, resulting in detrimental performance degradation. The concept of integral basis is the cornerstone in the quest for a stable MIEPM tool. Our literature search shows that the only extant stable integral macro IEPM tool is the M^\prime RT [1], which is the improved version of the macro Convergence Algorithm (CA) for Internet applications. Therefore, we propose to derive and develop the M^\prime RT from the integral M^\prime RT, for microscopic application in mobile-agent-based real-time computing over the Internet.

2. Related Work

Since the M^\prime RT is based on the Central Limit Theorem, its ability to predict the mean is insensitive to the type of distribution. The earliest CA work [13] concentrated on the conceptual framework of how to attain possible quick and accurate convergence. The focus in the later development of the M^\prime RT [1] is the feasibility of deploying the CA framework as a macro IEPM tool over the open Internet. The aim is to characterize the channel behavior over a long period by measuring its mean message roundtrip time [14, 5]. The CA and the M^\prime RT, however, were not tested exhaustively. The CA tests were limited to a few empirical simulations with known distributions. The empirical tests of the M^\prime RT were carried out only with a few channel data sets collected from several Internet sites. There is no way to know whether these data sets are representative of the channel behavior. Since the convergence stability for M^\prime RT is of utmost importance, the proposed tool must be verified exhaustively. The step to achieve this is to build the corresponding M^\prime RT Petri-Net model and then verify it vigorously in the credible Alpha/Sim time/stochastic Petri-Net environment. The Alpha/Sim by ALPHATECH INC. (version 1.0.10) is an industrial standard that satisfies the DoD requirements.

The essence of CA is represented by equations: (2.2) and (2.3). The mathematical average of i samples (RTT) can be computed theoretically by equation (2.1). In reality equation (2.1) has a serious hidden problem when working with a large volume of data samples, especially for samples with large values; that is, possible computation overflow. To resolve this problem, the CA algorithm (2.2), which is based on the Central Limit Theorem, was proposed. The parameter M_0 is the first
sample when the CA is started for the first time. When the CA was deployed in a continuous one-week test run for an extremely busy Internet channel, no overflow occurred (7 million RTT samples), despite frequent oscillations in the convergence computation process.

\[
M_i = \frac{\sum_{j=1}^{i-1} m_j}{i} \quad (2.1)
\]

\[
M_i = \frac{\sum_{j=1}^{i} m_j}{iF} \quad (2.2), i \geq 1
\]

\[
M_{i-1} = \frac{\sum_{j=1}^{i} m_j}{i} \quad (2.3)
\]

Equation (2.2) has resolved the computation overflow problem in equation (2.1) with the following measures: a) replacing \( i \) in the denominator of (2.1) by \( F \), known as the flush limit, and (b) making the \( M_i \) convergence iterative, where \( i (i \geq 1) \) denotes the current cycle in the convergence computation. The \( M_{i-1} \) is the predicted value in last cycle, and it is included in the next cycle for integral purpose. The parameter \( j \) marks the \( j^{th} \) sample mean in the current convergence cycle. Unfortunately, all the changes above cannot get rid of the serious oscillations in the convergence process. Since these oscillations may lead to system instability and erroneous performance, they have to be damped. This leads to the incorporation of proportional (P) control and the resultant M^3RT model (equation (2.4)).

\[
PM_{i-1} + \sum_{j=1}^{i} m_j \quad (2.4)
\]

\[
M_i = \frac{\sum_{j=1}^{i} m_j}{P + f} \quad (2.4)
\]

3. The M^3RT Petri-Net Model

Figure 2 is the proposed M^3RT Petri-net model, which has three basic parts: Data Generator (DG), Queue, and the M^3RT Body (CA_B). In the DG birth process, the Client place is initialized with one token for starting and perpetuating a simulation. In a simulation, when the Generate transition fires, two tokens are generated and one of these (representing a request) goes to Queue (the service queue) and the other goes back to Client to start another simulation cycle. The transition Generate is fired by the chosen waveform/distribution of a given mean. In the actual simulations, different types of waveforms were used, but only the results from those Gamma distributions in Table 1 are presented here for demonstration. The Gamma function of a wanted shape is created by setting the parameters \( \alpha \) and \( \beta \). By varying the shape, the birth rate \( \lambda \) can be controlled at will in different tests. The mean queue length of Queue is estimated by the by M^3RT mechanism, declared as the CA_B part in the Petri-Net model. If the M^3RT (CA_B) prediction works correctly, the mean of the input Gamma function that drives \( \lambda \), should converge to the given value.

![Figure 2. The Time Petri-Net (Alpha/Sim) model for the M^3RT](image)

The requests in the Queue are consumed by the CA_B one by one, where the CA_Feedback place is initialized with one token, but M_{i-1} is initialized to zero. The Passing transition fires to represent the death/service rate \( \mu \) by the M^3RT, and the last M_{i-1} drives the CA_Feedback place. The CA_Calculation transition, which embeds equation (2.4), fires when there is a token in CA_Input. In the simulations, the parameters in equation (2.4) are set to the following optimal values suggested by [9]: a) \( P = 10 \) (P for proportional control), and \( F = 14 \) (F for the flush limit). The M_i computed at the \( i^{th} \) iteration in the M^3RT convergence process is stored in CA_EstMean, and it also serves as the M_{i-1} input for the next ((i+1)^{th}) convergence cycle. Since the transitions in Petri-Net model fire in an indiscriminate manner, the M^3RT behavior in a simulation is determined by the instantaneous firing sequence by the transitions. For example, the consuming rate \( \mu \) by the CA_B is decided by the two transitions within, namely, Passing and CA_Calculation. It is, however, imperative to ensure that the overall system utilization \( \delta = \lambda / \mu \) in the model always satisfies the constraint of \( \delta \leq 1 \) for the sake of system stability.

4. Verification Results

The verification of the M^3RT is carried out by simulation with its time Petri-Net model in the stable Alpha/Sim environment. The aim is to ensure that the M^3RT always converges to the mean of the chosen input waveform that drives \( \lambda \). The simulation results presented in this section are selected test outputs for the set of Gamma input waveforms listed in Table 1. These waveforms, which mimic different raw RTT distributions (RawIAT), are also plotted as graphs in Figures 3-5. The
corresponding convergence behaviors by the M3RT to the tabulated waveforms are plotted in Figures 6-8.

<table>
<thead>
<tr>
<th>α</th>
<th>β</th>
<th>Raw mean (mathematical)</th>
<th>Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>12</td>
<td>8</td>
</tr>
</tbody>
</table>

**Table 1. Different Gamma functions with defined mean values**

The y-axis (Time) on the graphs (Figures 6-8) represents the raw RTT for different requests (No. of Jobs) marked on the x-axis. Although the fluctuations among the raw RTT are serious for the 1000 RawIAT samples being scrutinized, the convergence to the given Gamma function mean is always quick and accurate.

Other values plotted on Figures 6-8 are for comparison and investigation purposes:

a) RawIAT is the simulated raw RTT by the Data Generator.

b) CA_EstMean is the estimated mean RTT using equation (2.4) for the 1000 simulated RTT samples and it converges exactly to the given mean of the chosen Gamma function in Table 1.

c) Raw Mean is the mathematical average of raw RTT, namely, \( \left[ \sum_{i=1}^{1000} RTT_i \right] / 1000 \).

d) Typical Mean is the mode of the distribution for the 1000 RawIAT samples.
5. Preliminary Empirical Results

For limited validation test purposes, the M’RT prototype is implemented as a Java API in the in-house CApackage, which should be imported by the object/agent programs that want to invoke it for prediction.

The M’RT Java API was deployed in different empirical tests running on the Aglets mobile agent platform over a controlled PolyU Intranet environment that represents a PIN. The use of the Aglets is for the reason of scalability. The point is that the results obtained from the Aglets should be repeatable in a scalable manner when the same experiments are carried out over the open Internet. That is what we plan for the immediate future, namely, vigorous validation tests. One of the preliminary tests is to measure the difference in time delays between two well-known middleware products: the MPICH (version 1.1.2) for MPI, and the IONA’s Orbix 2.3c MT (CORBA 2.0 specification). The setup for this experiment is concisely illustrated in Figure 9.

In the setup in Figure 9, the driver agent makes requests, with random inter-arrival time (IAT) intervals, to ask the two local “Interfaces” to ping two separate remote server agents: one through the “CORBA-mediated” channel and the other through the “MPI-mediated” channel. The same ping, which is finally sent to the other side through the channels, is answered independently by the server agents, running on the same Sun machine. The answers are returned separately through the original channels. The “ping RTT” results are received by the local Interfaces, which pass them to the two independent dedicated M’RT APIs. The predicted mean values by the two APIs are then compared. The same experiment is repeated many times with different waveforms for the raw RTT. The results confirm that the M’RT always converged correctly. Figure 10 and Figure 11 are two sets of experimental results chosen for demonstration here, one set for MPI and the other for CORBA. The M’RT parameters in the two cases are: P=10, and F=14. The comparison of the two sets of results shows that the mean RTT is longer for the CORBA-mediated channel. This concurs with what had been observed in other similar experiments. It is a known fact the ORB (Object Request Broker) in CORBA is very slow. The M’RT actually enables the user to measure and gauge the ORB sluggishness, relative to other middleware (e.g. MPI) in particular. From all the experimental results gathered so far the MPI ping RTT is faster than the CORBA’s by at least 300%. That is why the CORBA is not exactly fit for some real-time applications that require astringent timeliness in a critical manner [16].
6. Conclusion

The focus of this paper is the development of a micro IEPM tool, namely, the M’RT, for real-time applications in a microscopic manner. The M’RT exists as an independent logical service provider and it does not require user intervention. Any client can invoke its service via message passing. The driving force behind the development of M’RT is the fact that there is no extant MIEPM tool that works on the integral basis. M’RT, which works on past history, is derived from the stable macro M’RT, which is an IEPM tool that always converges. In the mobile agent environment it helps agent migration decisions by directly measuring the mean RTT between agents quickly and accurately so that deleterious effects are avoided. For verification of the M’RT, the corresponding Petri-Net model is built. This model is first verified by exhaustive simulations and then it is converted into a Java API for limited validation tests. These tests were carried out in a controlled Intranet environment that behaves like a PIN. So far, the test results confirm that the M’RT has substantial potential in supporting real-time applications running on large networks such as the Internet, especially when mobile agents are involved. The next step in the immediate future is to validate the M’RT with more complex mobile-agent applications in the open Internet environments, in which unpredictable error events are the rule.

7. Acknowledgement

The authors thank the Hong Kong PolyU for the grants: GW110, APB89 and HZJ92.

8. References


