Experimental Evaluation of a Hybrid Approach for Deriving Service-time Bounds of Methods in Real-time Distributed Computing Objects

Juan A. Colmenares[1], K. H (Kane) Kim[1], and Doo-Hyun Kim[2]
[1]DREAM Laboratory. EECS Department. University of California, Irvine, USA.
jcolmena@uci.edu, khkim@uci.edu
[2]School of Internet and Multimedia Engineering. Konkuk University, Korea.
doohyun@konkuk.ac.kr

Abstract. Use of hybrid approaches that symbiotically combine analysis and measurements for deriving high-confidence tight service-time bounds (STBs) in real-time distributed computing (RTDC) applications represents a promising research area. A hybrid approach of this type was recently proposed for deriving STBs for methods in object-oriented RTDC applications. The approach combines analytical and measurement-based techniques to find a tight STB falling between the maximum measured service time and an analytically derived loose STB. A curve-fitting technique is applied to relate the measured data to the loose bound and also enables the estimation of the probability of the chosen STB not being exceeded at run time. Experimental research for checking the feasibility and potential problems of this type of hybrid approaches has been scarce. In this paper we report on the results of one case study aimed for validating the curve-fitting based hybrid approach mentioned above. The RTDC application dealt with in this experimental work is a relatively simple distributed video streaming application, called Televideo.

Keywords: service-time bound analysis, hybrid approach, curve fitting, time-triggered message-triggered objects.

1 Introduction

An essential requirement in real-time distributed computing (RTDC) is to obtain a high degree of assurance on the timeliness of critical actions taken by the systems. Previous research work on timing analysis has mostly covered single-node real-time programs involving no operating system (OS) or middleware service calls (e.g., [1]). But the timing analysis of RTDC applications involving substantial OS overhead and communication functions has not been sufficiently attempted.

A timely research issue is to derive *service-time bounds* (STBs) (or equivalently, guaranteed response times) for methods in object-oriented RTDC applications. When we use the term STB of an object-method M_i, we consider preemption-allowed executions of M_i and factor in both the actual execution times of M_i and the waiting times while M_i is in the active state.

We focus on a hybrid approach for deriving tight STBs for methods in object-oriented RTDC applications. In our approach, analysis as well as testing and measurements are important parts. The central idea in our hybrid approach is to find a

STB falling between the maximum measured service time and an analytically derived loose but safe bound. Specifically, service-time measurements and an analytically derived STB of a given object-method are used together for: 1) determining the safety margin to be added to the maximum observed service time in order to produce an acceptably tight and safe STB for the method, and 2) estimating the probability of this chosen STB not being exceeded at runtime.

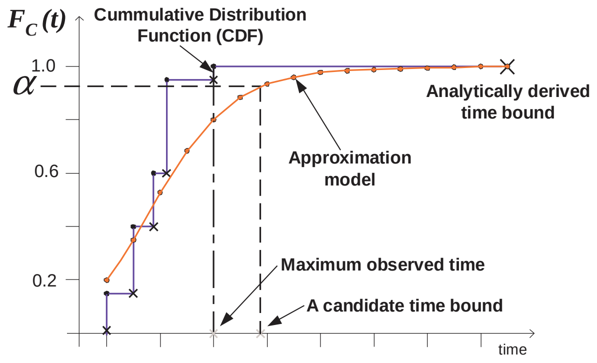


Fig. 1. Curve-fitting technique for deriving reasonably safe and tight time bounds (adapted from [2]).

A *curve-fitting technique* [2], illustrated in Fig. 1, plays a key role in relating the measured data to the analytically derived STB. This technique produces smooth approximation models that allow us to systematically determine tight and reasonably safe STBs. Note that time bounds produced analytically are considered *hard bounds* with a practically negligible probability of being violated at runtime. The reason is that methods for time bound analysis are frequently based on conservative assumptions and the derived bounds tend to have large error margins.

Conceptually, the approach presented in this paper can be seen as an extension of the segment-level hybrid approach in [2, 3] for deriving tight execution-time bounds (ETBs) for *program-segments* (i.e., non-interruptible code sequences not including OS and middleware calls) [4]. However, the analysis of service times of object-methods is a subject multiple times more complex than that of execution times of program-segments. In the analytical derivation of STBs for object-methods, we need to consider additional factors (e.g., concurrent method-executions, resource sharing, and communication activities) that influence the service times of methods.

As shown in Fig. 2, our approach works at the method level but it relies on the program-segment-level hybrid approach for obtaining tight ETBs for basic code elements (i.e., program-segments, local service calls, and non-blocking communication service calls) contained in the methods under analysis. The tight ETBs obtained for the basic code elements of the subject object-methods are integrated during the analytical derivation of safe STBs for the methods.

Experimental work on deriving STBs of object-oriented RTDC applications has been scarce. This means that conceivable approaches for deriving STBs for object-methods in RTDC applications have not been much validated. In this paper we report on the results of one case study aimed for such validation. The RTDC application dealt with in this experimental research is a relatively simple distributed video streaming application, called *Televideo*. Deriving tight STBs of RTDC systems is, in general, a very complex problem. Thus, through this work we wanted to obtain better insight into the fundamental problem.

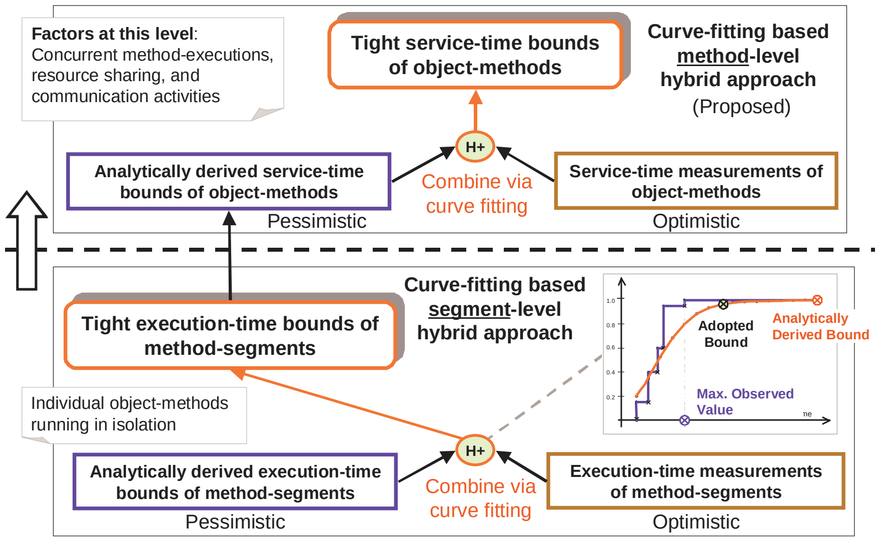

Factors at this level: Concurrent method-executions, resource sharing, and communication activities
Tight service-time bounds of object-methods
Curve-fitting based method-level hybrid approach (Proposed)
H+
Analytically derived service-time bounds of object-methods
Combine via curve fitting
Service-time measurements of object-methods
Pessimistic
Optimistic
Tight execution-time bounds of method-segments
Curve-fitting based segment-level hybrid approach
Adopted Bound
Analytically Derived Bound
Max. Observed Value
Individual object-methods running in isolation
H+
Analytically derived execution-time bounds of method-segments
Combine via curve fitting
Execution-time measurements of method-segments
Pessimistic
Optimistic

Fig. 2. General view of the proposed approach for deriving bounds for the service times of TMO-methods.

The Televideo application is structured as a network of *time-triggered message-triggered objects* (TMOs) and was developed according to the *TMO programming model* [5, 6]. The TMO model is a component-based programming scheme that relieves RTDC developers of the burden of dealing with low-level computing and communication abstractions. The TMO, the central element of this programming scheme, is a syntactically simple and natural but semantically major extension of the conventional object structure. The autonomous-action capability of TMO stems from the *time-triggered methods*, which are clearly separated from the *message-triggered methods* whose executions are activated by service request messages from clients. TMO allows developers to explicitly specify timing requirements in terms of global time in natural forms. The *TMO Support Middleware* (TMOSM) [7] is a middleware model that supports the execution of TMO-based applications.

The rest of the paper is structured as follows. Section 2 describes the Televideo application. The experimental setup is presented in Section 3. Section 4 presents the adopted ETBs for the object-methods in Televideo when each method runs in isolation; these time bounds are called *non-preemptive execution-time bounds* (NPETBs). Then, Section 5 discusses the derivation of STBs for the object-methods in Televideo. Section 6 provides a conclusion.

2 Televideo Application

Televideo is a TMO-structured application that allows two users in different nodes to establish a two-way video session. Each node continuously sends video frames captured by a local web camera to the other node, which in turn displays them on the screen. In addition, each node monitors, at the application level, the network performance associated with the incoming video stream and periodically sends a report with the collected performance metrics to the other node. Televideo has been implemented on Linux and uses the video codecs provided in FFmpeg.[1]

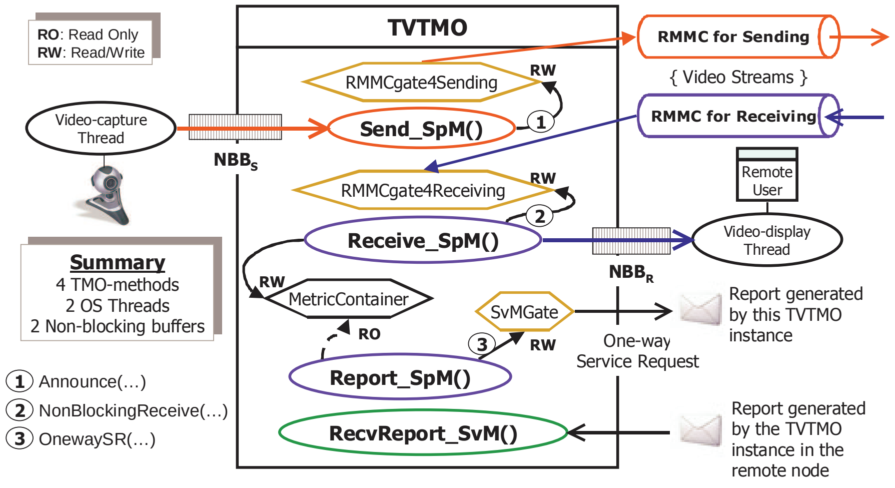


Fig. 3. Structure of the Televideo program running in each node.

Fig. 3 presents the structure of the Televideo program running in each node. The Televideo program consists of: 1) a TMO (called `TVTMO`), 2) two OS native threads (the *video-capture thread* and the *video-display thread*), and 3) two non-blocking buffers (NBB_S and NBB_R) [8, 9]. `TVTMO` is the central element of Televideo and contains three *time-triggered* or *spontaneous methods* (*SpMs*), and one *message-triggered* or *service method* (*SvM*).

SpMs and SvMs are collectively termed TMO-methods [5, 6]. Executions of SpMs are initiated during specified time windows. All time references in TMOs are global-time references and the SpM executions are thus triggered when the global time reaches specific values specified at design time. For example, the triggering times of an SpM may be specified as: `{FOR t = FROM 8:00am TO 11:55pm;EVERY 100ms; START-DURING (t, t+30ms); FINISH-BY (t+80ms)}.` This specification of the execution-time windows of an SpM is called the *Autonomous Activation Condition* (AAC) of the SpM.

Executions of SvMs, on the other hand, are activated by service-request messages from local or remote clients (e.g., other TMOs). For each SvM, system designers

[1] Available at http://ffmpeg.mplayerhq.hu

specify: 1) a guaranteed execution-duration bound (GEDB); 2) the maximum invocation rate (MIR), in terms of the maximum number of service requests (`MaxInvocations`) that will lead to the activation of the subject SvM per basic period (`BasicPeriod`); and 3) the maximum number of instances of the SvM that can execute concurrently (`PipelineDegree`).

As shown in Fig. 3, the TMO-methods of `TVTMO` are:

- `Send_SpM()`: Each execution-instance of this method reads from NBB_S a video frame captured by the local web camera, encodes it, and then sends the encoded frame to the `TVTMO` in the other node via a logical multicast channel, called *Real-time Multicast and Memory-replication Channel* (RMMC) [6].
- `Receive_SpM()`: Each execution-instance of this method reads from an RMMC an encoded video frame sent by the other node, decodes it, and inserts the decoded frame into NBB_R. It also collects the network performance metrics associated with the incoming frames and stores them in `MetricContainer`.
- `Report_SpM()`: Each execution-instance of this method reads the performance metrics from `MetricContainer`, creates a report, and sends the report via a one-way service-request message to the `RecvReport_SvM()` in the other node.
- `RecvReport_SvM()`: Each instance of this method reads the report contained as a parameter in the SvM-call, i.e., the service-request message that activated its execution, and displays the content on the console.

The `MetricContainer` is a group of data members shared by `Receive_SpM()` and `Report_SpM()`. Such a group of data members in a TMO is called an *object-data-store segment* (ODSS) [5, 6]. An ODSS is a data-container unit that can be *automatically* locked to allow a TMO-method execution to have exclusive access. The TMO programming scheme defines several fundamental rules to orchestrate executions of TMO-method instances that have data-conflicts among themselves requiring exclusive access to ODSSs. The reader is referred to [10, 11] for a discussion on ODSSs and the concurrency rules in the TMO scheme.

`TVTMO` also contains *gate objects* [6] that provide call-paths to RMMCs and SvMs. The gate objects are treated as ODSSs with read-write access mode. `TVTMO` uses the gate objects to make *non-blocking communication service calls* only. These non-blocking calls return quickly without waiting for the completion of the activities of other TMOs and communication networks involved.

3 Experimental Setup

The experimental setup consists of two multi-core computers connected to an isolated 100-Mbps Ethernet switch. Both run Linux 2.6.25. One of the computers is equipped with a 2.4-GHz Intel Core 2 Quad CPU and 3GB of RAM. The results presented in this paper correspond to the Televideo program running in this computer, which is called the *test node*.

In each node, the *TMO Support Middleware* (TMOSM) [7] was configured to dedicate a core for the execution of TMO-methods. Moreover, a round-robin algorithm was incorporated into TMOSM to schedule the executions of the TMO-methods and the length of the round-robin time-slice was 3 ms in both nodes.

Communication among TMOSM instantiations running in the two distributed computing nodes was performed in a TDMA manner.

Televideo was configured with the following parameters: 1) frame size: 320 pixels $\times$ 240 pixels, 2) frame rate: 10 fps, 3) color depth: 24 bits, and 4) encoding format: MPEG4.

4 Adopted NPETBs for the TMO-methods in Televideo

As described in Section 2, the SpMs and SvM of `TVTMO` are *blocking-communication-call-free* (BCCF) methods. The most basic contribution of a BCCF TMO-method to its service time is the time the method takes to execute when it is neither affected by background activities nor subject to preemption. An ETB for a BCCF TMO-method M_i obtained under such execution conditions is called a *non-preemptive execution-time bound* (NPETB) for M_i.

Tight and reasonably safe NPETBs for the TMO-methods of `TVTMO` were obtained by applying the curve-fitting based hybrid approach originally proposed in [2] and using the measurement techniques discussed in [3]. This hybrid approach was slightly extended in order to consider, in addition to program-segments, the local service calls, non-blocking communication service calls, and application library calls present in the TMO-methods of `TVTMO`.

The adopted NPETBs for the TMO-methods of the Televideo program are the following: 11 ms for `Send_SpM()`, 2 ms for `Receive_SpM()`, 16 ms for `Report_SpM()`, and 19.1 ms for `RecvReport_SvM()`. The NPETBs for `Send_SpM()` and `Receive_SpM()` were the time values at which $F_c(t)$ values of the approximation models for the two SpMs became practically 1.0. For `Report_SpM()` and `RecvReport_SvM()` which executed less frequently than other TMO methods, we decided that it was sufficient to adopt NPETBs with estimated probabilities of not being exceeded at runtime equal to 90%.

Due to space limitation we do not discuss the derivation of the NPETBs and we refer the reader to [11] for the complete derivation analysis.

5 Derivation of GEDBs for the TMO-methods in Televideo

During the design of a TMO-based application, the part that requires the biggest care is the specification of the *guaranteed execution-duration bounds* (GEDBs) and other timing parameters for the TMO-methods. Often, in practice, the selection of those parameters may be achieved via iterative cycles. Table 1 contains the timing parameters for the TMO-methods of `TVTMO` that resulted from this iterative process.

To avoid buffer overflow, the periods of `Send_SpM()` and `Receive_SpM()` are shorter than the interval at which the video frames are produced (i.e., 1 frame every 100 ms or equivalently, 10 fps). Additionally, we assume that a network performance report needs to be generated every 700 ms; this interval ensures that multiple execution-instances of `RecvReport_SvM()` never run concurrently. For this reason, we configured `RecvReport_SvM()` with the parameters

PipelineDegree, BasicPeriod, and MaxInvocations shown in Table 1. The rest of the timing parameters, in particular BY (= LST + GEDB) of the SpMs and GEDB of the SvM, were selected based on and validated using the present iterative STB analysis illustrated below.

SpM	FROM	UNTIL	EVERY	EST	LST	BY
Send_SpM()	3 s	24 h	70 ms	0	6 ms	39 ms
Receive_SpM()	3 s	24 h	70 ms	54 ms	61 ms	66 ms
Report_SpM()	3 s	24 h	700 ms	0	6 ms	54 ms

EVERY: period, EST: earliest start time, LST: latest start time, BY: relative deadline

SvM	BasicPeriod	MaxInv.	PipelineDegree	GEDB
RecvReport_SvM()	650 ms	1	1	58 ms

GEDB: guaranteed execution duration bound and relative deadline

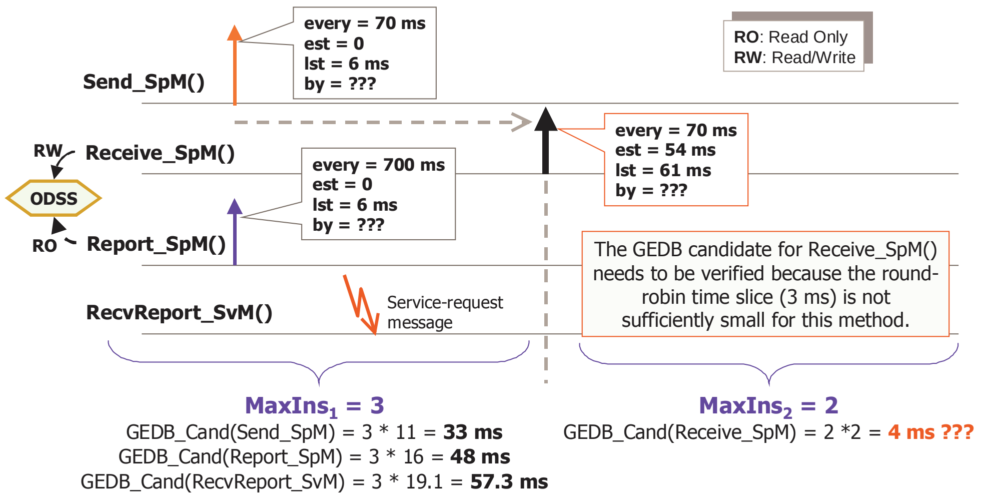
Table 1. Timing parameters for the TMO-methods of TVTMO.

We use a simple technique to analytically obtain loose GEDB candidates for the TMO-methods of TVTMO. We first determine the maximum number of TMO-method instances that can execute simultaneously during a certain interval; this quantity is denoted as *MaxIns*. This means that the CPU (which the TMO-methods were assigned to) may need to be multiplexed among the executions of *MaxIns* TMO-method instances. Since the round-robin scheduler is adopted and if we assume that the length of the round-robin time-slice is sufficiently small, then a loose GEDB candidate of a given TMO-method SxM_i can be obtained as follows:

$$GEDB\_Cand(SxM_i) = MaxIns \times NPETB(SxM_i) \qquad \textbf{(1)}$$

In general, this GEDB calculation is rather conservative and the produced GEDBs tend to be pessimistic. System designers can, however, tighten those bounds by using the curve-fitting based hybrid approach, as will be shown later in this section.

Given the timing parameters of Table 1, at most 3 TMO-method instances, one of Send_SpM(), one of Report_SpM(), and one of RecvReport_SvM(), can be in execution at the same time (see Fig. 4). Hence, a loose GEDB candidate for each of those 3 TMO-methods can be obtained by multiplying its NPETB by 3. In the case of Receive_SpM(), however, its EST value makes its execution to be overlapped with none but for a possible instance of RecvReport_SvM(). Thus, a GEDB candidate for Receive_SpM() could be initially obtained as $2 \times$ NPETB(Receive_SpM). Fig. 4 presents the preliminary GEDB candidates for the TMO-methods of TVTMO.



RO: Read Only
RW: Read/Write
Send_SpM()
every = 70 ms
est = 0
lst = 6 ms
by = ???
RW
Receive_SpM()
ODSS
RO
Report_SpM()
every = 700 ms
est = 0
lst = 6 ms
by = ???
every = 70 ms
est = 54 ms
lst = 61 ms
by = ???
The GEDB candidate for Receive_SpM() needs to be verified because the round-robin time slice (3 ms) is not sufficiently small for this method.
RecvReport_SvM()
Service-request message
MaxIns1 = 3
GEDB_Cand(Send_SpM) = 3 * 11 = 33 ms
GEDB_Cand(Report_SpM) = 3 * 16 = 48 ms
GEDB_Cand(RecvReport_SvM) = 3 * 19.1 = 57.3 ms
MaxIns2 = 2
GEDB_Cand(Receive_SpM) = 2 *2 = 4 ms ???

Fig. 4. Initiation times and preliminary analytically derived GEDB candidates for the TMO-methods of TVTMO. MaxIns is the maximum number of TMO-method instances that can execute simultaneously at a give time.

TMO-method	GEDB Candidate	MMV	Ratio
`Send_SpM()`	33 ms	21.50 ms	1.53
`Receive_SpM()`	5 ms	3.47 ms	1.44
`Report_SpM()`	48 ms	25.19 ms	1.90
`RecvReport_SvM()`	57.3 ms	24.78 ms	2.31

MMV: Maximum Measured Value ∴ Ratio = (GEDB Candidate) / MMV

Table 2. Analytically derived GEDB candidates and maximum observed execution durations for the TMO-methods of TVTMO.

Note that the analytically derived GEDB candidate for `Receive_SpM()` (4 ms) needs to be verified because it is comparable with the length of the round-robin time-slice (3 ms). In other words, the assumption of having a round-robin time-slice sufficiently small does not hold for `Receive_SpM()`. The effect of time-slicing can cause an instance of `Receive_SpM()` to be delayed by 3 ms when the SpM becomes ready right after the time-slice taken by `RecvReport_SvM()` starts. Therefore, the tentative GEDB candidate obtained in Fig. 4 is not valid. Since NPETB(`Receive_SpM`) = 2 ms, then a valid GEDB candidate for `Receive_SpM()` is Max (2*2ms, 2ms + 3ms) = 5 ms. We also analyzed the consequences of a tick miss on the service time of `Send_SpM()` and confirmed that the corresponding GEDB candidate shown in Fig. 4 is valid. Table 2 contains the verified analytically derived GEDB candidates for the TMO-method of TVTMO.

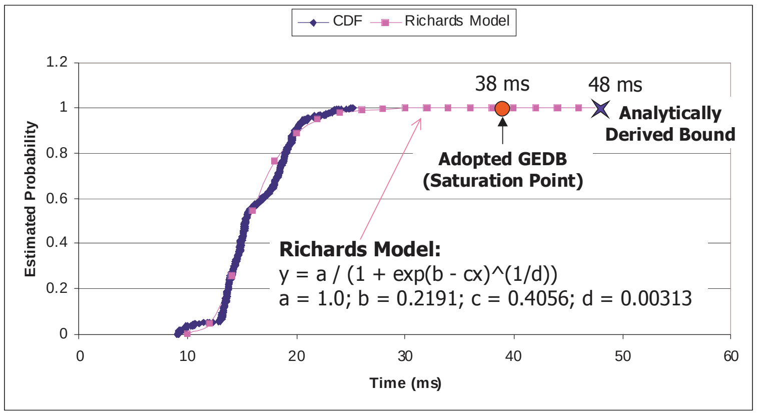


Fig. 5. Obtaining a tighter GEDB candidate for `Report_SpM()` via curve fitting.

We measured the execution durations of the TMO-methods of `TVTMO` in the node under test to validate the analytically derived GEDB candidates and explore the possibility of tightening those bounds using the curve-fitting technique. We collected 6000 execution-duration measurements for `Send_SpM()` and the same number for `Receive_SpM()`. In the case of `Report_SpM()` and `RecvReport_SvM()`, we collected 750 measurements for each of them. Table 2 presents the maximum observed execution durations of the TMO-methods of `TVTMO`.

The ratios in Table 2 clearly suggest that we can obtain tighter GEDB candidates using the curve-fitting technique for `Report_SpM()` and `RecvReport_SvM()`. Fig. 5 shows: 1) the *cumulative distribution function* (CDF) of the measured execution-duration times, 2) the analytically derived GEDB candidate, 3) the approximation model that best fits the CDF among a collection of smooth models, and 4) the newly adopted GEDB for `Report_SpM()`, which is equal to 38 ms (ratio = 1.51). Using the same technique we also obtained a GEDB equal to 34 ms (ratio = 1.37) for `RecvReport_SvM()`. Thus, the hybrid approach produced tighter yet reasonably safe GEDBs for both TMO-methods.

6 Conclusion

In this paper we reported an experimental evaluation of a curve-fitting based hybrid approach for deriving service-time bounds of object-methods in RTDC applications. The hybrid approach was applied to a relatively simple TMO-structured multimedia application, called Televideo. In this case study, the approach turned out to be effective in tightening the service-time bounds for the object-methods in the subject application. The effectiveness of the hybrid approach must, however, be further evaluated under more complex application scenarios. This is the subject of our ongoing and future research efforts.

Acknowledgments

The research reported here is supported in part by the NSF under Grant Numbers 03-26606 (ITR) & 05-24050 (CNS), by the ETRI, Korea, by the Ministry of Knowledge Economy, Korea, under the ITRC support program supervised by the IITA (IITA-2008-C1090-0804-0015), and by the Konkuk Univ WCU project, No, R33-2008-000-10068-0 sponsored by the MEST, Korea. Juan A. Colmenares also thanks the Univ of Zulia (Venezuela) for supporting his participation. No part of this paper represents the views and opinions of the sponsors mentioned above.

References

1. Wilhelm, R., Engblom, J., Ermedahl, A., Holsti, N., et al.: The worst-case execution time problem - Overview of methods and survey of tools, *ACM Trans. on Embedded Computing Systems*, 7(3), pp. 1-53 (2008)
2. Im, C., Kim, K. H.: A hybrid approach in TADE for derivation of execution time bounds of program-segments in distributed real-time embedded computing. In: 9th IEEE Int'l Symposium on Object and Component-Oriented Real-Time Distributed Computing, pp. 408-418 (2006)
3. Colmenares, J. A., Im, C., Kim, K. H., et al.: Measurement techniques in a hybrid approach for deriving tight execution-time bounds of program segments in fully-featured processors. In: 14th IEEE Real-Time and Embedded Technology and Applications Symposium, pp. 68-79 (2008)
4. Kim, K. H., Choi, L., Kim, M. H.: Issues in realization of an execution time analyzer for distributed real-time objects. In: 3rd IEEE Symposium on Application-Specific Systems and Software Engineering Technology, pp. 171-178 (2000)
5. Kim, K. H.: Object structures for real-time systems and simulators. IEEE Computer, 30 (8), pp. 62-70 (1997)
6. Kim, K. H.: APIs for real-time distributed object programming, IEEE Computer, 33 (6), pp. 72-80 (2000)
7. Jenks, S. F., Kim, K., et al.: A middleware model supporting time-triggered message-triggered objects for standard Linux systems. Real-Time Systems, 36 (1), pp. 75-99 (2007)
8. Kim, K. H.: A non-blocking buffer mechanism for real-time event message communication. Real-Time Systems, 32 (3), pp. 197-211 (2006)
9. Kim, K. H., Colmenares, J. A., and Rim, K.-W.: Efficient adaptations of the non-blocking buffer for event message communication between real-time threads. In: 10th IEEE Int’l Symposium on Object/Component/Service-Oriented Real-time Distributed Computing, pp. 29-40 (2007)
10. Kim, K. H., Colmenares, J. A.: Maximizing concurrency and analyzable timing behavior in component-oriented real-time distributed computing application systems. J. Computing Science and Engineering, 1(1), pp. 56-73 (2007)
11. Colmenares, J. A.: Derivation of service-time bounds of methods in time-triggered message-triggered objects. PhD thesis, University of California, Irvine (2009)