Measuring Distributed Durations with Stable Errors*

António Casimiro Pedro Martins Paulo Veríssimo Luís Rodrigues

Faculdade de Ciências da Universidade de Lisboa
Bloco C5, Campo Grande, 1749-016 Lisboa, Portugal
{casim,pmartins,pjv,ler}@di.fc.ul.pt

Abstract

The round-trip duration measurement technique is fundamental to solve many problems in asynchronous distributed systems. In essence, this technique provides the means for reading remote clocks with a known and bounded error. *Therefore, it is used as a fundamental building block in several clock synchronization algorithms. In general, the technique can be used to implement duration measurement services, such as the one of the* Timely Computing Base model. *In this paper we propose a new technique to measure distributed durations that minimizes the measurement error and is able to keep this error almost stable. The new technique can be used to improve the precision of remote clock reading in certain situations. We provide a protocol that implements this new technique and we present some evaluation results. The results clearly show that our solution is indeed better than existing ones.*

1 Introduction

The problem of remote clock reading has always been one of the fundamental problems in distributed computing, deriving from the need of synchronizing the clocks of all the nodes in the system.

Most proposed clock synchronization protocols require the reading of remote clocks. They are built on the principle that a node can synchronize its clock with the clock of a remote node if it knows the value displayed by the remote clock at a given instant. In practice, however, since remote clocks cannot be read instantaneously, when the local clock is synchronized to the remote one, they will already be apart. The difference corresponds to the time elapsed between the two events of reading the remote clock, and setting the local one. It is therefore necessary to take this time into account when synchronizing the clocks, which requires the use of some measurement technique.

We say that reading a remote clock is a *distributed action*, which takes a certain amount of time, which we call a *distributed duration*. Generically, a message exchanged between any two nodes in a distributed system can be considered a distributed action, to which is associated a certain real time duration that may be estimated with a bounded error. The ability to measure distributed durations with bounded and small errors is crucial not only for clock synchronization, but more generally in any distributed environment with timeliness requirements.

In this paper we propose a new technique for measuring distributed durations (or for reading remote clocks), that provides improved measurement errors when compared with the other existing techniques. In particular, it is able to provide better results than the original round-trip duration measurement technique [5] and some of its successors [7, 1]. We refer to this new technique as the **Improved Round-Trip Technique (IMP)** and propose a protocol that implements it. This protocol (and implicitly the improved technique) have been used, in particular, in the construction of the Duration Measurement service of the Timely Computing Base (TCB) model [12]. The TCB model provides a generic approach to the problem of partial synchrony, and the Duration Measurement service is one of the basic services that it provides in order to satisfy a wide range of applications with timeliness requirements.

The experiments that we performed using implementations of both the original and the improved round-trip techniques have clearly confirmed that it is possible to obtain better results with the later. We observed that unlike the previous existing solutions, the proposed solution is able to deliver almost stable measurement errors, which only increase due to the drift rate of local clocks.

The rest of the paper is organized as follows. In the next section we refer to related work in the area. The basic round-trip duration measurement technique is briefly presented in Section 3, just before we explain how it can be

*This work was partially supported by the EC, through project IST-2000-26031 (CORTEX), and by the FCT, though the Large-Scale Informatic Systems Laboratoty (LASIGE) and projects Praxis/P/EEI/12160/1998 (MICRA) and Praxis/P/EEI/14187/1998 (DEAR-COTS).

improved, in Section 4. The complete protocol is presented in Section 5 and the evaluation results appear in Section 6. Finally, Section 7 concludes the paper.

2 Related Work

The measurement of distributed durations is a generic problem of asynchronous distributed systems, which has been addressed in the context of other more specific problems, such as achieving clock synchronization or ensuring timely communication.

The variety of algorithms and solutions for clock synchronization that have been proposed during more than a decade is considerable (see surveys in [2] and [11]). Of all these algorithms, we are particularly interested in the category of probabilistic algorithms, in which it is necessary to obtain estimations of remote clock values, measuring the duration of reading actions. The seminal paper of Cristian about probabilistic clock synchronization [5] has first formally presented the round-trip duration measurement technique, on which several other works have built thereafter [1, 6, 7]. Probabilistic estimation methods have also been proposed in [3] and [10] and of particular relevance is the Network Time Protocol (NTP), widely used in the internet, that also uses a round-trip based method to obtain estimations of remote clock values [9].

These clock synchronization algorithms exploit the fact that message delivery delays in existing asynchronous networks, although exhibiting possibly very high delays, are typically around a small value, near the lower bound. Provided that a sufficient number of messages is transmitted, it is highly probable that some of those messages are fast messages, allowing good estimates of remote clock values. But as we show in this paper, this estimates can in certain situations be improved using the new technique we propose. Therefore, our contribution can lead to better probabilistic clock synchronization algorithms.

On the other hand, some services need to ensure the best possible estimation for *each* message delivery delay, independently of typical probabilistic distributions and observation intervals. This is the case of the Duration Measurement service, specified in the context of the Timely Computing Base (TCB) model [12]. We contribute with a new technique that is more appropriate to implement this service than previously existing ones.

3 The Round-Trip Technique

The round-trip duration measurement technique proposed by Cristian [5], used to read remote clock values, basically consists in the following. When a process p wants to read the clock of some process q, it measures on its local clock the round-trip delay elapsed between the sending of a request message m_1 to q, and the arrival of the reply m_2 (see Figure 1). This delay provides an upper bound for the time m_2 took to travel from q to p and allows to bound the reading error of q's clock.

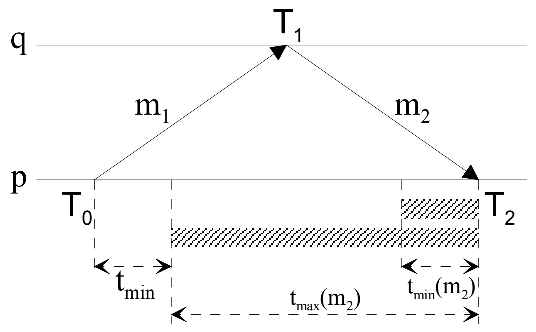

Figure 1. Round-Trip duration measurement using Cristian's technique.

Assuming the minimum message transmission delay to be t_{min}, and the maximum drift rate of local clocks to be ρ, the real time duration for the transmission delay of m_2, $t(m_2)$, can be bounded as follows (the superscript RT stands for the **R**ound-**T**rip method):

$$t^{\mathrm{RT}}_{max}(m_2) = (T_2 - T_0)(1+\rho) - t_{min} \tag{1}$$

$$t^{\mathrm{RT}}_{min}(m_2) = t_{min} \tag{2}$$

Therefore, the transmission delay of m_2 can be estimated as the midpoint of this interval, with an associated error equivalent to half of the interval. The result is:

$$t^{\mathrm{RT}}(m_2) = \frac{(T_2 - T_0)(1+\rho)}{2} \tag{3}$$

$$\varepsilon^{\mathrm{RT}}(m_2) = \frac{(T_2 - T_0)(1+\rho)}{2} - t_{min} \tag{4}$$

In the above expressions, relative to the example of Figure 1, process q sends m_2 immediately when it receives m_1. However, note that in real settings one has to take into account the processing time spent by q to generate the reply. Therefore, a more generic approach consists in assuming that any request message m_k sent from p to q can be used to estimate $t(m_2)$ (see Figure 2). The estimation of $t(m_2)$ using any message pair $\langle m_k, m_2\rangle$, requires that process p knows all send and receive timestamps for that pair, that is, T_S, T_R, T_1 and T_2.

In the example illustrated in Figure 2 there is a message m_k that is used instead of m_1 to estimate $t(m_2)$. Although the minimum possible value for the transmission delay of

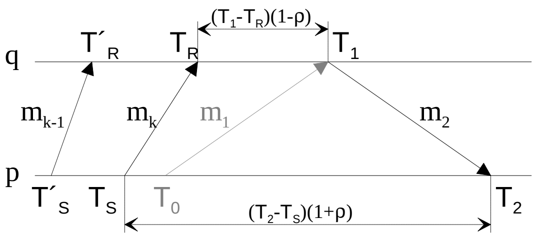

Figure 2. Choosing the optimal message pair in the round-trip duration measurement technique.

m_2 is always t_{min}, the expression for $t^{\mathrm{RT}}_{max}(m_2)$ can now be written as follows:

$$t^{\mathrm{RT}}_{max}(m_2) = (T_2 - T_S)(1+\rho) - (T_1 - T_R)(1-\rho) - t_{min} \tag{5}$$

This also affects the expressions that are used to estimate $t(m_2)$:

$$t^{\mathrm{RT}}(m_2) = \frac{(T_2 - T_S)(1+\rho) - (T_1 - T_R)(1-\rho)}{2} \tag{6}$$

$$\varepsilon^{\mathrm{RT}}(m_2) = \frac{(T_2 - T_S)(1+\rho) - (T_1 - T_R)(1-\rho)}{2} - t_{min} \tag{7}$$

Now assume that there is another message m_{k-1} that has also been sent by p to q before m_k. The basic round-trip duration measurement technique simply proposes to use the most recent message pair (in this case $\langle m_k, m_2\rangle$) to estimate $t(m_2)$. However, it is obviously possible to make a small optimization which consists in using the message pair ($\langle m_{k-1}, m_2\rangle$ or $\langle m_k, m_2\rangle$) that provides the best estimation.

This optimization has been presented in [7], as well as the criteria to decide which message pair is the optimal one. A similar result has also been presented in [1]. The criteria to decide which messages are "best" for estimation purposes is applied whenever a new message is received. For instance, when process q receives message m_k it has to decide whether m_k is "better" than m_{k-1} for the purpose of estimating the transmission delay of a subsequent message sent to p (m_2 in this example). This is done by comparing T_S and T_R with T'_S and T'_R. The condition for using m_k instead of m_{k-1} is the following:

$$\text{update: } (T_S - T'_S)(1+\rho) > (T_R - T'_R)(1-\rho) \tag{8}$$

4 Achieving a Stable Error

Given a message pair $\langle m_1, m_2\rangle$, we have seen that with the round-trip duration measurement technique, $t_{min}(m_2)$ is always t_{min}. In fact, since no assumption whatsoever is made about m_1, it is possible to have any upper bound for $t(m_1)$ and therefore the lower bound for $t(m_2)$ can be the lowest possible, that is, t_{min}. However, since it is possible to calculate upper bounds for received messages, the receiver of m_1 has already determined $t_{max}(m_1)$ when it sends m_2. Therefore, this value could be sent along with m_2, making the receiver of m_2 able to use it in its calculations and able to possibly obtain a lower bound for $t(m_2)$ higher than t_{min}. At best, the interval of variation of $t(m_2)$ would be reduced, yielding a more accurate estimation of $t(m_2)$.

This simplistic reasoning is sufficient to provide the intuition for the proposed improved technique. The basic idea is to estimate the transmission delay of received messages using not only the send and receive timestamps for the message pair, but also the estimated value for the delay of the first message of the pair. As we will see, the only drawback of the proposed improvement is that it requires more information to be transmitted between processes than in previous round-trip based solutions. Figure 3 illustrates the fundamental relations that are used in this improved duration measurement technique.

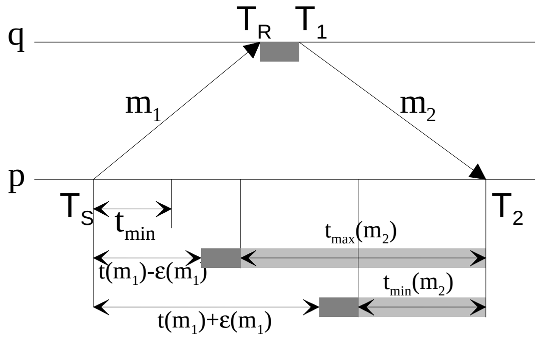

Figure 3. Round-Trip duration measurement using the improved technique.

To estimate the transmission delay of some message m_2 sent from process q to p, another message m_1 must have been previously sent from p to q. The round-trip delay of $\langle m_1, m_2\rangle$ can be used to determine the upper bound for $t(m_2)$. It is also necessary to subtract the minimum (real) time spent by q before sending m_2 and the minimum possible delay of m_1, yielding the following expression:

$$\begin{aligned} t^{\mathrm{IMP}}_{max}(m_2) \quad = \quad & (T_2 - T_S)(1+\rho) - (T_1 - T_R)(1-\rho) - \\ & (t(m_1) - \varepsilon(m_1)) \end{aligned} \tag{9}$$

This expression is very similar to that of the original round-trip technique (expression (5)). The difference is that now the last term depends on the estimation of $t(m_1)$, with $t(m_1) - \varepsilon(m_1)$ possibly higher than t_{min}. Note that $t(m_1) - \varepsilon(m_1)$ cannot be lower than t_{min}, which guarantees that $t^{\mathrm{IMP}}_{max}(m_2)$ is not higher than $t^{\mathrm{RT}}_{max}(m_2)$. Although it may seem that $t_{max}(m_2)$ can now be lower, we show that this is unfortunately not true (see Appendix A). The upper bound of a measured duration is always the same no matter which technique, the original or the improved one, is used. The lower bounds, however, can be different.

The physical lower bound for $t(m_2)$ is obviously t_{min}. But it might also be higher than that, depending on the estimation of $t(m_1)$. Taking the lowest possible (real) time value for the round-trip duration, and subtracting the maximum (real) time spent by q before sending m_2 and the maximum possible delay of m_1, we obtain the following:

$$\begin{aligned} t^{\mathrm{IMP}}_{min}(m_2) &= MAX[t_{min}; (T_2 - T_S)(1-\rho) - \\ &\quad (T_1 - T_R)(1+\rho) - (t(m_1) + \varepsilon(m_1))] \end{aligned} \tag{10}$$

The expressions for the estimated transmission delay of m_2 follow directly from (9) and (10), assuming that the lower bound for $t(m_2)$ is higher than t_{min}:

$$\begin{aligned} t^{\mathrm{IMP}}(m_2) &= \frac{t_{max}(m_2) + t_{min}(m_2)}{2} \\ &= (T_2 - T_S) - (T_1 - T_R) - t(m_1) \end{aligned} \tag{11}$$

$$\begin{aligned} \varepsilon^{\mathrm{IMP}}(m_2) &= \frac{t_{max}(m_2) - t_{min}(m_2)}{2} \\ &= \rho(T_2 - T_S) + \rho(T_1 - T_R) + \varepsilon(m_1) \end{aligned} \tag{12}$$

However, if we use the above expressions when the lower bound of $t(m_2)$ is t_{min}, the result is that $t^{\mathrm{IMP}}(m_2) - \varepsilon^{\mathrm{IMP}}(m_2)$ will be lower than t_{min}, which is obviously impossible. If this happens, then we can obtain the correct estimation for $t(m_2)$ making the following simple transformation:

$$\begin{aligned} t'^{\mathrm{IMP}}(m_2) &= \frac{t_{max}(m_2) + t_{min}(m_2)}{2} \\ &= \frac{t^{\mathrm{IMP}}(m_2) + \varepsilon^{\mathrm{IMP}}(m_2) + t_{min}}{2} \end{aligned} \tag{13}$$

$$\begin{aligned} \varepsilon'^{\mathrm{IMP}}(m_2) &= \frac{t_{max}(m_2) - t_{min}(m_2)}{2} \\ &= \frac{t^{\mathrm{IMP}}(m_2) + \varepsilon^{\mathrm{IMP}}(m_2) - t_{min}}{2} \end{aligned} \tag{14}$$

Note that we have done these transformations simply using the knowledge that $t_{max}(m_2) = t(m_2) + \varepsilon(m_2)$ and that $t_{min}(m_2) = t_{min}$. Note also that in expression (12) it is quite evident how the estimation error is kept almost constant. In fact, each time a new estimation is made, the error just increases by an amount corresponding to the drift of local clocks during the (typically small) intervals of the round-trip measurement. This means that the error keeps increasing, after each consecutive measurement, until a message is received for which the estimated lower bound is t_{min}. Then, expressions (13) and (14) will bring the error again to a lower value. In the most extreme case, if we could assume perfect clock with $\rho = 0$, the error would never increase and would just be reduced upon the reception of “faster” messages. Therefore, we can say that the error we achieve is optimal given the assumptions that were made.

The improved technique described so far requires more information to be exchanged between processes than the original round-trip technique. Besides the timestamps T_S, T_R and T_1, the sender of m_2 also has to provide $t(m_1)$ and $\varepsilon(m_1)$. Furthermore, each process also has to keep more information than before, namely $t(m)$, $\varepsilon(m)$, T^m_S and T^m_R for the “best” message m received from each other process. These extra requirements are the trade-off for achieving best estimations.

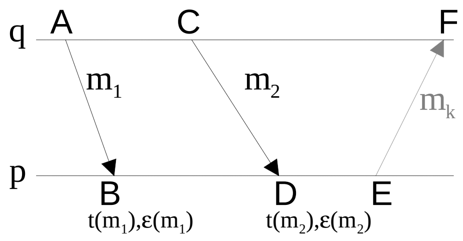

Figure 4. Comparing messages m_1 and m_2.

Just like in the round-trip technique, every received message is a potential “best” message. Hence, upon the reception of a new message it is necessary to determine whether this message is “better” than the currently “best” one. The criteria to consider a message better than another is strictly related to its potential to propagate smaller estimation errors. An old message with a very small associated error can be better than a new message with a large error.

The exact expression that must be used to compare two received messages is presented below (see Figure 4 for reference). In Appendix A we prove the correctness of this expression.

$$\text{Update: } \varepsilon(m_2) < (C - A)\rho + (D - B)\rho + \varepsilon(m_1) \tag{15}$$

Example:

The impact of using the improved technique instead of the original one is easily observed when the real transmission delay of a message is visibly higher than that of previous messages.

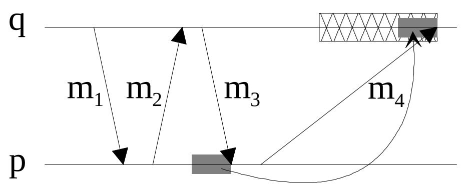

Figure 5. Example of improved transmission delay estimation.

For instance, in the example of Figure 5 there is a message, m_4, that is "slower" than the previous ones. When using the original round-trip technique to estimate $t(m_4)$, the estimation error will be nearly half of the round-trip delay of the pair $\langle m_3, m_4\rangle$. On the other hand, when using the improved round-trip this error is "inherited" from the estimation error of $t(m_3)$, which is at most half of the round-trip delay of the pair $\langle m_2, m_3\rangle$, clearly smaller than that of pair $\langle m_3, m_4\rangle$. The occurrence of messages with transmission delays higher than normal is thus completely irrelevant for the estimation errors obtained with the improved technique. The error is kept almost stable (see this effect in Figure 9, in Section 6).

The reader should note another interesting effect than can only be observed when the improved technique is used. Since the technique allows the best errors to be (optimally) preserved from a message to the consecutive one, the occurrence of a single message pair of two "fast" messages is sufficient to establish a small error that will be used in all subsequent estimations. This can be particularly interesting for systems where the resource utilization and the delays can be kept constantly high during long periods of time.

5 The Protocol

In this section we describe a protocol that uses the improved round-trip duration measurement technique. This protocol has been implemented as part of the distributed duration measurement service of our TCB prototype, developed for the Real-Time Linux operating system [13] (see Section 6).

The explanation is divided in three parts for simplicity. We first describe the constants and global variables that are used in the protocol (Figure 6). Then we describe the main loop of the protocol, presented in Figure 7 and finally we explain the more functional operations executed by the protocol, depicted in Figure 8.

```
// Constants
// $myid$ Id of executing process
// $\rho$ Maximum drift rate of local hardware clocks
// $t_{min}$ Minimum message delay
// Global Variables
// $ST$, $RT$, $DEL$, $ERR$ are arrays with entries for each process
// $ST(p)$ Send Timestamp of "best" message received from p
// $RT(p)$ Receive Timestamp of "best" message received from p
// $DEL(p)$ Delay of "best" message received from p
// $ERR(p)$ Error of "best" message received from p
// $del_m$ Estimated delay for received message $m$
// $err_m$ Error of estimation for received message $m$
// Global Function
// $C(t)$ Current hardware clock value
```

Figure 6. Constants and global variables.

Each process p runs an instantiation of this duration measurement service. They all have a different $myid$ value, but ρ and t_{min} are the same in all instances. Since it is necessary to keep informations regarding the best message received from every process, we use arrays ST, RT, DEL and ERR to store this information. The size of these arrays is N, where N is the number of processes in the system. Variables del_m and err_m keep the estimated delay and its associated error, that are returned to the user at the end of the main loop (Figure 7). We assume that it is possible to read the local clock value using function $C(t)$, which returns positive timestamps.

```
task Distributed_Duration_Measurement
for all p do
    $ST(p) \leftarrow -\infty$; $RT(p) \leftarrow 0$;
    $DEL(p) \leftarrow +\infty$; $ERR(p) \leftarrow +\infty$;
end do
loop
when request to send $\langle m\rangle$ using $st$ timestamp do
    broadcast $\langle m, st, ST, RT, DEL, ERR\rangle$;
end do
when $\langle m, st_m, ST_p, RT_p, DEL_p, ERR_p\rangle$ received from $p$ do
    $rt_m \leftarrow C(t)$;
    if $RT_p(myid) = 0$
        first_message ($p$,$st_m, rt_m$);
    else if $DEL_p(myid) = +\infty$
        second_message ($p$,$st_m, rt_m, ST_p, RT_p$);
    else
        normal_message ($p$,$st_m, rt_m, ST_p, RT_p, DEL_p, ERR_p$);
    end if
    deliver $\langle m\rangle$ with $(del_m, err_m)$;
end do
end loop
end task
```

Figure 7. Pseudo code for the main loop.

The main body of the protocol has an initialization block, followed by an execution loop. When the protocol starts, no messages have yet been received from other processes. Hence, we assume to have received imaginary initialization messages at instant 0, with send timestamp $-\infty$, and with infinite delays and errors. In the main loop we define two possible entry points, corresponding to application requests to send messages and to messages received from the network. In this protocol the send timestamp st is provided by the application since this is imposed by the interface of the TCB duration measurement service (see [12] for a complete description of this interface).

We use a broadcast service to avoid the need to specify a destination process. Nevertheless, when a message is sent it is necessary to include the four arrays and also the send timestamp. When no message is transmitted during a long period, the estimation error of received messages tends to increase due to the drift rate of local clocks. Therefore, in the implementation of the duration measurement service we added an additional action to force periodic (service specific) message transmissions. These periodic messages can be viewed as synchronization messages that prevent the errors to increase indefinitely. This action is not presented here since it is not strictly required by the improved technique.

The second entry point corresponds to messages received from other processes. Upon the reception of a message a receive timestamp is immediately obtained. Then, the message is processed accordingly to its type. Messages can be of three logical types:

- First messages – When the service at p initiates, it will start receiving messages from the other processes. These messages do not contain any information about messages sent by p to the other processes. They are simply *first messages* that will be used to estimate the delay of subsequent messages. They are identified by having $RT_p(myid) = 0$, which means that process p has never received a message from processor $myid$.

- Second messages – After sending its first message m to all other processes, process p will start receiving messages that have already been sent after their senders have received m. It is thus possible to estimate the delay for these *second messages* using the original round-trip technique. They can be identified because $DEL_p(myid) = +\infty$, that is, p has received a message from processor $myid$ but has not been able to estimate its delay.

- Normal messages – All other messages are received in a state that allows the improved technique to be applied. Therefore we call them *normal messages*.

It is possible to receive several messages of the same type, and each message type is processed by a different function. However, all the processing functions assign values to del_m and err_m, which are returned to the application, along with the received message, in the end of the loop.

```
update_info ($p$,$del_m, err_m, st_m, rt_m$)
begin
    $ST(p) \leftarrow st_m$; $RT(p) \leftarrow rt_m$;
    $DEL(p) \leftarrow del_m$; $ERR(p) \leftarrow err_m$;
end

first_message ($p$,$st_m, rt_m$)
begin
    $del_m \leftarrow +\infty$;
    $err_m \leftarrow +\infty$;
    if $(st_m - ST(p))(1+\rho) > (rt_m - RT(p))(1-\rho)$
        update_info ($p$,$del_m, err_m, st_m, rt_m$);
    end if
end

second_message ($p$,$st_m, rt_m, ST_p, RT_p$)
begin
    $del_m \leftarrow ((rt_m - ST_p(myid))(1+\rho) -$
            $(st_m - RT_p(myid))(1-\rho) - t_{min})/2$;
    $err_m \leftarrow del_m$;
    if $err_m < ERR(p) + \rho(st_m - ST(p)) + \rho(rt_m - RT(p))$
        update_info ($p$,$del_m, err_m, st_m, rt_m$);
    end if
end

normal_message ($p$,$st_m, rt_m, ST_p, RT_p, DEL_p, ERR_p$)
begin
    $del_m \leftarrow (rt_m - ST_p(myid)) - (st_m - RT_p(myid)) -$
            $DEL_p(myid)$;
    $err_m \leftarrow ERR_p(myid) + \rho(rt_m - ST_p(myid)) +$
            $\rho(st_m - RT_p(myid))$;
    if $del_m < err_m + t_{min}$
        $correct\_del_m \leftarrow (del_m + err_m + t_{min})/2$;
        $correct\_err_m \leftarrow (del_m + err_m - t_{min})/2$;
        $del_m \leftarrow correct\_del_m$;
        $err_m \leftarrow correct\_err_m$;
    end if
    if $err_m < ERR(p) + \rho(st_m - ST(p)) + \rho(rt_m - RT(p))$
        update_info ($p$,$del_m, err_m, st_m, rt_m$);
    end if
end
```

Figure 8. Message processing functions.

Each of the message processing functions does two things: a) it estimates the transmission delay of messages; b) it updates, if necessary, the array entry of the process from which the message has been received. Estimating the message transmission delay of “first messages” is not possible, since there is not way to establish an upper bound for this delay. Therefore, the `first_message()` function simply assigns the $+\infty$ value to del_m and err_m. To determine is a new first message is better than a previous one, it is necessary to apply the expression (8) of the original round-trip technique. This is so because “first messages” will be paired with “second messages” to estimate the de-

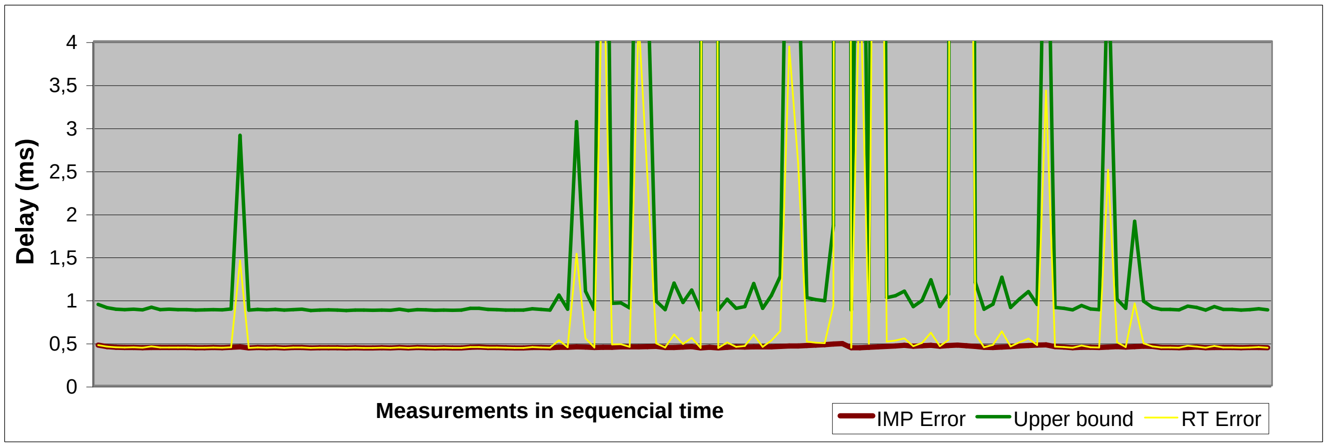

Figure 9. Measurement upper bounds and errors using the improved (IMP) and the original round-trip (RT) techniques.

lay of the later, and the original round-trip technique will be applied. The `update_info()` function is simply used to store the new information in the local arrays.

As just said, the transmission delay of "second messages" is estimated with the original round-trip technique (expressions (6) and (7)). However, the update decision is now based on the improved technique rule (expression (15).

Finally, the `normal_message()` function processes every other message, fully using the improved round-trip technique. del_m and err_m are obtained using expressions (11) and (12), and eventually expressions (13) and (14), if condition $del_m < err_m + t_{min}$ evaluates to true. The update part is equal to the one of function `second_message()`.

6 Evaluation Results

A distributed duration measurement service using the protocol just described, has been implemented as part of our RT-Linux TCB prototype [4]. Therefore, we were able to evaluate the effectiveness of the proposed improved round-trip technique when compared to its original version.

The tests were performed in our distributed systems laboratory, using Pentium based PCs connected to a 10 Mbit Ethernet LAN. To enhance the precision of our evaluation results, we have also used a special measurement tool that we developed, the Event Timestamping Tool [8], to externally measure the transmission delay of messages. Since our aim was to compare both techniques, we simply needed to generate sequences of messages pairs, for which only two machines were necessary.

Given that the TCB duration measurement service only implements the improved technique, for the comparisons we had to extend the service with an implementation of the original version of the round-trip technique, exactly as described in [7]. Both implementations run in parallel, using exactly the same input values, that is, the same send and receive timestamps. They both output pairs of $\langle del_m, err_m \rangle$ values that are used for the comparison.

The experiment consisted in the execution of two applications, a client and a server, respectively concerned with periodically sending a message, and replying to every received message. We have configured the client to send a message every 500ms. Therefore, in all the figures presented bellow the samples depicted in the X axis correspond to consecutive measurements obtained with intervals of about 500ms. Note that these intervals are affected not only by the jitter of the scheduling delay (of the client's sending task), but also by the jitter of the overall round-trip delay. The specific content of the messages is irrelevant for the experiment. Both applications were implemented on top of the TCB duration measurement service. Their only functionality, besides sending and receiving messages, was to store the returned pair of measurement values in a local file.

We had to set the values of some constants used in the protocol. The drift rate was set to $\rho = 5x10^{-6}$ and we assumed the minimum message transmission delay to be zero ($t_{min} = 0$). The size of the arrays was set to a value larger than two (the number of used processors).

Relatively to the conditions of the environment during the experiment, in particular the network and system loads, we have forced a scenario with almost no activity (typically idle), interleaved with short periods of intensive network utilization. This unstable behavior is perfectly visible in the figures presented below and, as expected, allows to clearly observing the improvements obtained with the proposed technique. To overload the network we simply used

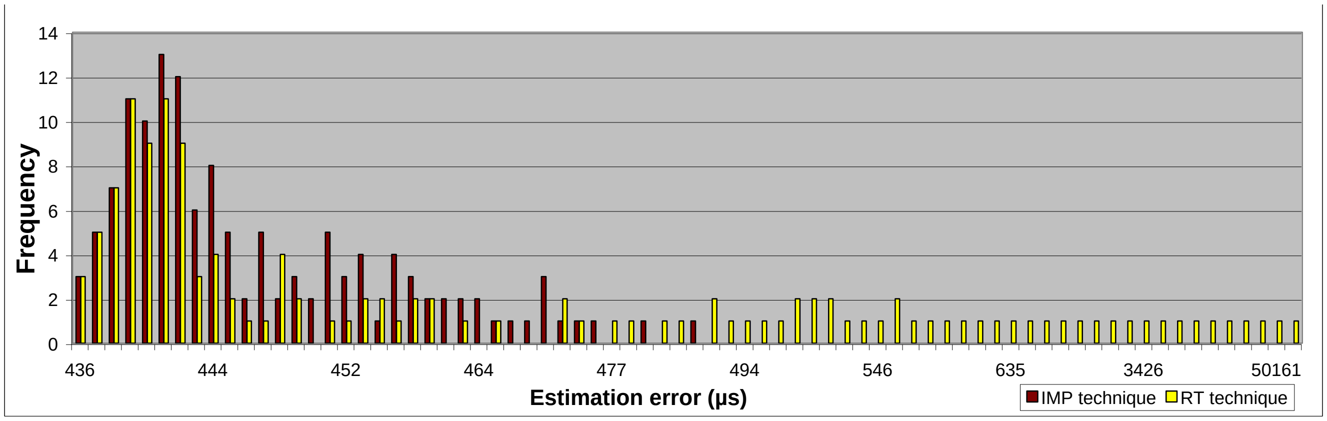

Figure 10. Distribution of estimation errors.

the Unix `ping` command, with the flood option enabled.

The applications were executed several times, just to compare the results and verify their coherence. However, for the purposes of this paper we have just selected a representative set of results, one where the differences between the two techniques can be observed.

In Figure 9 we compare the error values achieved by both techniques in a series of consecutive measurements. The upper bound values are also represented to provide an idea of the system behavior during the experiment. Higher upper bounds typically correspond to messages transmitted during periods of more intensive traffic. However, since the `ping` command was randomly issued, during randomly sized time intervals, we cannot guarantee a strict correspondence between higher delays and the increased traffic generated by the `ping` command.

The most important result that may be observed in this figure is the almost stable error achieved by the improved technique (IMP Error), in contrast with the variable error achieved by the other technique (RT Error). This result clearly confirms our expectations. Note that the RT Error line closely follows the Upper bound line, as dictated by the original round-trip technique (in this case, since t_{min} was set to zero, RT Error is always half of the Upper bound).

To more clearly observe, and compare, the dispersion of the measurement errors achieved with the two techniques, we analyzed the frequency of each observed error value, and present the result in Figure 10. Error values with no occurrences are obviously not depicted, which means that the X axis scale is irregular. The higher estimation error observed with the IMP technique was near 490μs, while with the RT technique we observed several error values above 500μs. This result shows that the improved technique can possibly be used to construct a distributed duration measurement service that ensures a bounded measurement error. Such a service can be very useful in network monitoring systems, since it allows the establishment of accuracy bounds for the observations.

The accuracy of the estimations obtained with the improved technique can be observed in Figure 11. In this figure we compare the real delay of message transmissions, with the delays that were estimated using both techniques. During the periods of “stability”, when the message transmission delays are low, both techniques seem to provide accurate estimations. However, when a higher delay occurs, only the IMP technique is able to provide accurate estimations.

7 Conclusions

We have presented a new technique to measure the duration of distributed actions and to read remote clocks. With this technique it is possible to achieve a lower reading error of remote clocks than with the well know round-trip duration measurement technique proposed by Cristian. Therefore, we have referred to it as the **Improved Round-Trip Technique**.

We described in detail the intuition that leads to this improved technique, and we provided the expressions that should be used to obtain the improved results. Besides reducing the error, and perhaps importantly, this technique keeps the estimation errors almost constant, independently of the real duration of the action that is being measured. This is an interesting feature for any distributed duration measurement service.

We also presented and described a protocol that implements the improved technique, which we have used in the implementation of the distributed duration measurement service of the Timely Computing Base (TCB). We were able to conduct an experiment to evaluate the effectiveness of the proposed technique. The results have shown a few interest-

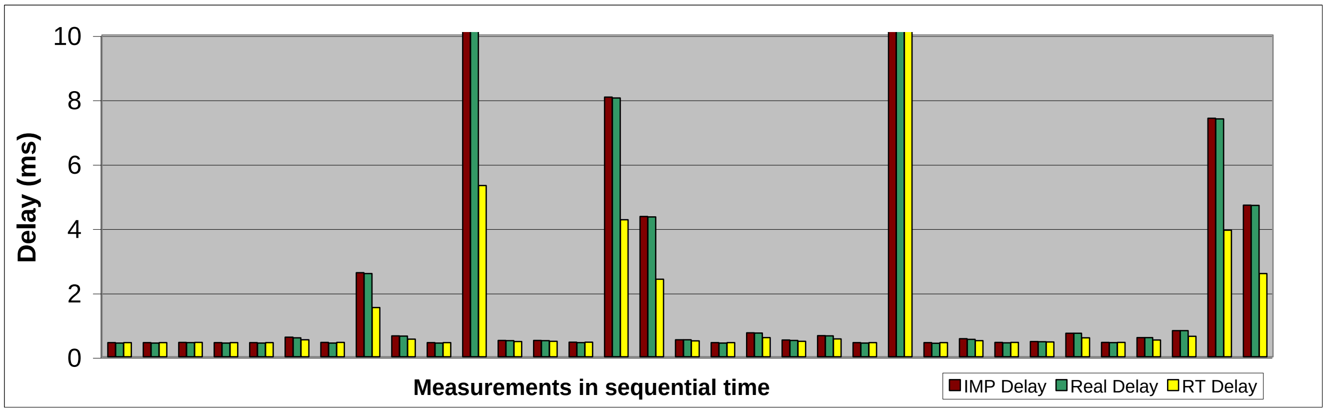

Figure 11. Estimated delays and real (measured) delay.

ing features of the improved technique:

- it is indeed able to provide readings with almost stable errors;
- it prevents measurement errors from depending on the measured delay (which might make them extremely high);
- its estimated delays are much closer to the real delays than the ones obtained with the original round-trip technique.

These positive characteristics render the improved round-trip duration measurement technique extremely effective and efficient, in short an excellent candidate for an embedded runtime supported service for distributed applications. This was the approach taken in our TCB prototype.

A Proofs

Theorem 1: *Given any message m, the upper bound determined for the message delivery delay of m using the improved technique is equal to the determined upper bound using the round-trip technique, $t^{\mathrm{IMP}}_{max}(m) = t^{\mathrm{RT}}_{max}(m)$.*
Proof:

We use Figure 12 to visually guide this proof. Consider the depicted sequence of messages. We will show that the theorem is valid for message m_{k+1}, and that it can be generalized to any other message.

We will start by showing that m_{k+1} can be the first message in some sequence that might possibly have $t^{\mathrm{IMP}}_{max} < t^{\mathrm{RT}}_{max}$ — all previous messages in the sequence are guaranteed to have the same upper bound using any of the techniques — and we will show that in spite of this possibility the upper bounds are nevertheless equal. Since the same reasoning can recursively be applied to the first subsequent message for which it might also be possible to have $t^{\mathrm{IMP}}_{max} < t^{\mathrm{RT}}_{max}$, the theorem can be generalized for every subsequent message.

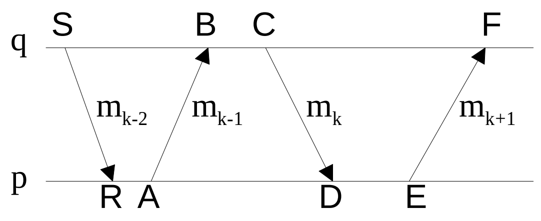

Figure 12. Upper bound preservation for m_{k+1}.

Let us assume that the message pair $\langle m_k, m_{k+1}\rangle$ is the one used to determine the upper bounds (m_k has been considered by p the "best" message so far). The equations that provide the upper bounds for m_{k+1} follow from (5) and (9):

$$t^{\mathrm{RT}}_{max}(m_{k+1}) = (F-C)(1+\rho) - (E-D)(1-\rho) - t_{min}$$

$$t^{\mathrm{IMP}}_{max}(m_{k+1}) = (F-C)(1+\rho) - (E-D)(1-\rho) - (t(m_k) - \varepsilon(m_k))$$

For $t(m_k) - \varepsilon(m_k) = t_{min}$ it immediately follows that $t^{\mathrm{IMP}}_{max}(m_{k+1}) = t^{\mathrm{RT}}_{max}(m_{k+1})$. We are left with the possibility that

$$t(m_k) - \varepsilon(m_k) > t_{min} \qquad (16)$$

In this case it is reasonable to believe that $t^{\mathrm{IMP}}_{max}(m_{k+1})$ may be lower than $t^{\mathrm{RT}}_{max}(m_{k+1})$, which we will prove to be untrue.

Now observe that for inequality (16) to be possible, it is necessary to consider the existence of a message pair preceding m_k in the same sequence. We assume this message pair to be $\langle m_{k-1}, m_{k-2}\rangle$, as depicted in Figure 12. Furthermore, we assume these two messages to be the first ones in the sequence, so that m_{k+1} is clearly the first message for which $t^{\mathrm{IMP}}_{max} < t^{\mathrm{RT}}_{max}$ may be possible. With these assumptions it is also clear that the theorem holds for all messages preceding m_{k+1}. Let us write the upper bound for $t(m_{k-1})$ (applying (5)):

$$t_{max}(m_{k-1}) = (B-S)(1+\rho) - (A-R)(1-\rho) - t_{min} \tag{17}$$

Given that $t^{\mathrm{IMP}}_{min}(m_k) = t(m_k) - \varepsilon(m_k) > t_{min}$, we can use expression (10) to write the following:

$$\begin{aligned}
&t^{\mathrm{IMP}}_{min}(m_k) > t_{min} \Leftrightarrow \\
&\quad\Leftrightarrow \quad (D-A)(1-\rho) - (C-B)(1+\rho) - \\
&\qquad\quad (t(m_{k-1}) + \varepsilon(m_{k-1})) > t_{min} \\
&\quad\Leftrightarrow \quad (D-A)(1-\rho) - (C-B)(1+\rho) - \\
&\qquad\quad t_{max}(m_{k-1}) > t_{min} \\
&\quad\Leftrightarrow \quad (D-A)(1-\rho) - (C-B)(1+\rho) - \\
&\qquad\quad (B-S)(1+\rho) + (A-R)(1-\rho) + t_{min} > t_{min} \\
&\quad\Leftrightarrow \quad (D-R)(1-\rho) - (C-S)(1+\rho) + t_{min} > t_{min} \\
&\quad\Leftrightarrow \quad (C-S)(1+\rho) < (D-R)(1-\rho)
\end{aligned}$$

On the other hand, from (8) we know that m_k is only considered a "best" message than m_{k-2} if $(C-S)(1+\rho) > (D-R)(1-\rho)$. So we must conclude that our initial assumption that the message pair $\langle m_k, m_{k+1}\rangle$ can be used to determine the upper bound of m_{k+1} is in contradiction with the fact expressed in (16). We must therefore prove that the theorem still holds when the message pair $\langle m_{k-2}, m_{k+1}\rangle$ is used to obtain $t^{\mathrm{RT}}_{max}(m_{k+1})$. In this case we have:

$$t^{\mathrm{RT}}_{max}(m_{k+1}) = (F-S)(1+\rho) - (E-R)(1-\rho) - t_{min}$$

which we must compare with $t^{\mathrm{IMP}}_{max}(m_{k+1})$:

$$\begin{aligned}
&t^{\mathrm{IMP}}_{max}(m_{k+1}) = \\
&\quad = \quad (F-C)(1+\rho) - (E-D)(1-\rho) - \\
&\qquad\quad (t(m_k) - \varepsilon(m_k)) \\
&\quad = \quad (F-C)(1+\rho) - (E-D)(1-\rho) - \\
&\qquad\quad ((D-R)(1-\rho) - (C-S)(1+\rho) + t_{min}) \\
&\quad = \quad (F-S)(1+\rho) - (E-R)(1-\rho) - t_{min}
\end{aligned}$$

It follows that $t^{\mathrm{IMP}}_{max}(m_{k+1}) = t^{\mathrm{RT}}_{max}(m_{k+1})$, which completes our proof.

□

Theorem 2: *Given any two messages, m_1 and m_2, the former sent at A and received at B with estimation error of ε_{m_1}, and the later sent at C and received at D with estimation error of ε_{m_2}, m_2 is considered to be "best" for the accuracy of the improved round-trip technique if $\varepsilon_{m_2} < \varepsilon_{m_1} + \rho(C-A) + \rho(D-B)$*

Proof:

To compare messages m_1 and m_2 we analyze their impact on the estimation of $t(m_k)$ for a subsequent message m_k (see Figure 4). The "best" message is the one that allows $t(m_k)$ to be estimated with a smaller error $\varepsilon(m_k)$.

Applying expression (12) to the round-trip pairs $\langle m_1, m_k\rangle$ and $\langle m_2, m_k\rangle$ we obtain the following:

$$\begin{aligned}
\varepsilon_{m_1}(m_k) &= \varepsilon(m_1) + \rho(F-A) + \rho(E-B) \\
&= \varepsilon(m_1) + \rho(F-C) + \rho(C-A) + \\
&\quad\ \rho(E-D) + \rho(D-B)
\end{aligned}$$

$$\varepsilon_{m_2}(m_k) = \varepsilon(m_2) + \rho(F-C) + \rho(E-D)$$

Hence, m_2 is better than m_1 if:

$$\begin{aligned}
&\varepsilon_{m_2}(m_k) < \varepsilon_{m_1}(m_k) \Rightarrow \\
&\quad\Rightarrow \quad \varepsilon(m_2) + \rho(F-C) + \rho(E-D) < \\
&\qquad\quad \varepsilon(m_1) + \rho(F-C) + \rho(C-A) + \\
&\qquad\quad \rho(E-D) + \rho(D-B) \\
&\quad\Leftrightarrow \quad \varepsilon(m_2) < \varepsilon(m_1) + \rho(C-A) + \rho(D-B)
\end{aligned}$$

□

References

[1] G. Alari and A. Ciuffoletti. Implementing a probabilistic clock synchronization algorithm. *Journal of Real-Time Systems*, 13(1):25–46, July 1997.

[2] E. Anceaume and I. Puaut. Performance evaluation of clock synchronization algorithms. Technical Report PI-1208, IRISA, Campus de Beaulieu, 35042 Rennes Cedex, France, October 1998.

[3] K. Arvind. Probabilistic clock synchronization in distributed systems. *IEEE Transactions on Parallel and Distributed Systems*, 5(5):474–487, May 1994.

[4] A. Casimiro, P. Martins, and P. Veríssimo. How to build a timely computing base using real-time linux. In *Proceedings of the 2000 IEEE Intl. Workshop on Factory Communication Systems*, pages 127–134, Porto, Portugal, September 2000.

[5] F. Cristian. Probabilistic clock synchronization. *Distributed Computing*, 3(3):146–158, 1989.

[6] F. Cristian and C. Fetzer. Probabilistic internal clock synchronization. In *Proceedings of the 13th Symposium on Reliable Distributed Systems*, pages 22–31, Dana Point, California, USA, October 1994. IEEE Computer Society Press.

[7] C. Fetzer and F. Cristian. A fail-aware datagram service. In *Proceedings of the 2nd Annual Workshop on Fault-Tolerant Parallel and Distributed Systems*, Geneva, Switzerland, April 1997.

[8] P. Martins and A. Casimiro. Event timestamping tool: a simple pc based kernel to timestamp distributed events. DI/FCUL TR 00–4, Department of Computer Science, University of Lisbon, July 2000.

[9] D. L. Mills. Internet time synchronization: the network time protocol. *IEEE Transactions on Communications*, 39(10):1482–1493, October 1991.

[10] A. Olson and K. G. Shin. Probabilistic clock synchronization in large distributed systems. *IEEE Transactions on Computers*, 43(9):1106–1112, September 1994.

[11] F. B. Schneider. Understanding protocols for Byzantine clock synchronization. Technical Report TR 87-859, Cornell University, Dept. of Computer Science, Upson Hall, Ithaca, NY 14853, August 1987.

[12] P. Veríssimo, A. Casimiro, and C. Fetzer. The timely computing base: Timely actions in the presence of uncertain timeliness. In *Proceedings of the International Conference on Dependable Systems and Networks*, pages 533–542, New York City, USA, June 2000. IEEE Computer Society Press.

[13] V. Yodaiken and M. Barabanov. A real-time linux. In *Proceedings of the USENIX conference*, 1997. http://rtlinux.cs.nmt.edu/rtlinux/papers/usenix.ps.gz.