High Resolution Time Estimation among Remote Machines Using Kalman Filtering

Leandro Fabrício Auler and Roberto d'Amore

Instituto Tecnológico de Aeronáutica

`lauler@ita.br`, `damore@ita.br`

Abstract - *Clock synchronization among remote machines is becoming important in commercial and instrumentation areas. Many clock adjustment and control protocols are available. The obtained precision, however, is disturbed by the unpredictable delays over the network.*

This work proposes a new approach to estimate time among machines interconnected by computer networks, which minimizes the disturbance introduced by the packets travel time. The methodology employs a high resolution clock and a stochastic filter to reduce the error in time measurement carried introduced by the network.

As result, we obtained an increase of time estimation accuracy close to three orders of magnitude, in comparison with traditional estimation methods.

1 Introduction

The Internet advent has created new challenges to time synchronization due to the speed increase of operations performed around the world. Time and date are critical elements on commercial transactions. They establish necessary evidence of the moment when the transaction takes place. The synchronism lack may generate unsuitably profit or loss in cases of selling, buying, financial transactions and electronic auctions, when the costs are related to the time of the action. The importance of hour maintenance and clock synchronization in commercial operations become evident by the synchronism services supplied by the National Institute of Standard and Technology to enterprises [1] and the Time Service Department of U.S. Naval Observatory [2].

In scientific applications, synchronism has its importance to assure time measurements of simultaneous events in different places, e.g. geophysical measurements like earthquakes and explosions, or events resulted by solar stains. Nowadays, the idea of remote laboratories is growing, and the samples acquisition need the storage of exact time, as well as, the schedule of actions performed by different devices that are working in cooperation.

The industrial use of synchronization is still uncommon. A possible application, for example, may be expected to electrical distribution companies. By using computer networks, these enterprises would perform a vast number of remote services: supervising the quality of service, observing the wave shape at real time and detecting problems in distant places. These operations need a clock synchronization of 100μs in order to achieve the demands of regulation agencies

Each example listed before demands a different accuracy, but all of them need a good reliability in time estimation. There are many protocols used today to remotely adjust time and date. Among them, we can list: *Time Protocol* [3], *Daytime Protocol* [4], *Network Time Protocol* (NTP) [5] and *Simple Network Time Protocol* (SNTP) [6]. The first two protocols only send the time and date in case of a request, without taking care with the accuracy of the value and delays introduced by the network. The NTP is the most commonly used Internet time protocol and it is employed in circumstances where higher reliability and higher accuracy are needed. The SNTP is a simplified variation of the NTP. Typically, in the Global Internet, NTP reaches synchronization accuracy from few milliseconds to tens milliseconds [7]. This synchronization accuracy is proportional to the accuracy of the remote time estimation [8].

This work proposes a new approach to estimate time in remote machines, minimizing the error introduced by the network delays. Due to the large use of the NTP in the Internet, we propose a technique which allows the improvement of time estimation to this protocol instead of a new synchronism model. This technique can be applied to any kind of network, even the Internet. Tests have already shown improvement upper to two orders of magnitude in comparison to the conventional estimation method used by NTP.

The Section 2 shows the basic concepts and terminology employed on the text. It is followed by Section 3, which presents the methodology adopted to develop this work, and in Section 4, the proposal of the work will be explained. In Section 5 some tests performed over the present proposal are discussed. At last, Section 6 will present the conclusions.

2 Initial Considerations

When two clocks are compared, usually one of them is adopted as the time reference *t* and the other as the compared time. The time difference between these two clocks at some instant is known as *offset(t)*, which definition is given by (1), where *t2* is the compared clock time. *Offset* is sometimes referred as clock error.

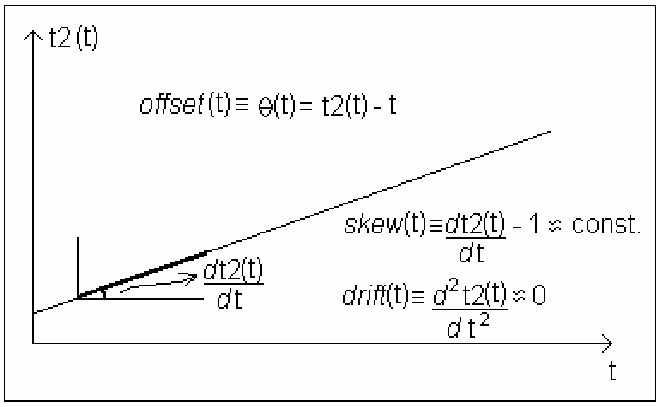
$$offset(t) \equiv \theta(t) = t2(t) - t \qquad (1)$$

Another very important parameter is denominated *skew*. It represents the rate which *offset* changes, or mathematically specking, it is the derivate of the *offset*, and is shown in Equation (2). A *skew* equal to zero means that both clocks have the same frequency. Temperature changes and electrical changes affect the frequency of the clocks. The frequency deviation of a clock is denominated *drift*, which can be defined as the second derivate of the *offset*, and is shown in Equation (3). For a crystal oscillator, the frequency deviation is often very small and can be ignored. For idealization purpose, the *drift* is considered zero through this. As a direct consequence of this idealization, the *skew* is considered constant. The first two parameters are better showed on Figure 1.

$$skew(t) \equiv \frac{\partial\theta(t)}{\partial t} = \frac{\partial t2(t)}{\partial t} - \frac{\partial t}{\partial t} = \frac{\partial t2(t)}{\partial t} - 1 = \phi - 1 \qquad (2)$$

$$drift(t) \equiv \frac{\partial^2 t2(t)}{\partial t^2} = \frac{\partial^2 \theta(t)}{\partial t^2} \qquad (3)$$

Another usual term used in time synchronization is *stratus*. It is a number attributed to a machine (which acts as a time reference) and indicates the quality of time supplied by this machine. For example, the NTP protocol attributes a *stratus* = 1 to machines which supply time directly from an atomic clocks or GPS, while *stratus* = 2 to machines which have them clocks synchronized with other *stratus* 1 machines.



t2 (t)
offset(t) ≡ θ(t) = t2(t) - t
skew(t) ≡ dt2(t)/dt - 1 ≈ const.
dt2(t)/dt
drift(t) ≡ d²t2(t)/dt² ≈ 0
t

Figure 1 - Main parameters used to compare two different clocks

To synchronize two clocks over a computer network, the *offset* estimation is required. The most usual method is by packet polling. It is performed by the transmission of a packet from a computer *a* (client) to a computer *b* (reference) and later returned to computer *a* [9]. During the packet travel, the instants of departure and arrival on each machine are storage in appropriate fields inside the packet. See Figure 2 to clarify.

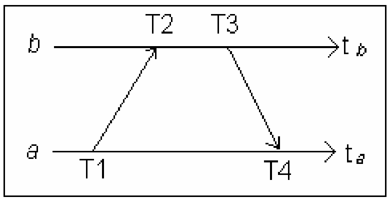


Figure 2 - Example of time polling between two machines interconnected by a packet-switched network.

Supposing that the time spent by a packet during its transmission is equal to its return (symmetric network), the *offset* of clock *b* (computer *b*) related to clock *a* (computer *a*) at some moment may be estimated by (4) [9]. In this equation, $\hat{\theta}$ is an estimator of θ. If the reference is changed from computer *a* to computer *b*, the Equation (4) may be rewritten as (5).

$$\hat{\theta}_b = \frac{(T2 - T1) + (T3 - T4)}{2} \tag{4}$$

$$\hat{\theta}_a = -\hat{\theta}_b = -\frac{(T2 - T1) + (T3 - T4)}{2} \tag{5}$$

This *offset* estimated between two clocks is satisfactory to achieve an accurate synchronism over symmetric networks without unpredictable delay variation (*Jitter*). Unfortunately, this is not the common case. Over long networks with intense traffic, as on the Global Internet, the influence of delay *jitter* may be large and it can’t be ignored. When Equation (4) is applied to time measurements performed by packets on the Internet, it gives an erroneous estimation. This erroneous estimation along the time may be modeled as a correct one plus a random noise or indetermination [10], see Figure 3. In these cases, it is required a more complex and reliable method to estimate the correct *offset*, considering the stochastic nature of the measurements. In practice, *offset* noise with hundreds of millisecond is often obtained when machines are separated by long distances and with elevated traffic over the network.

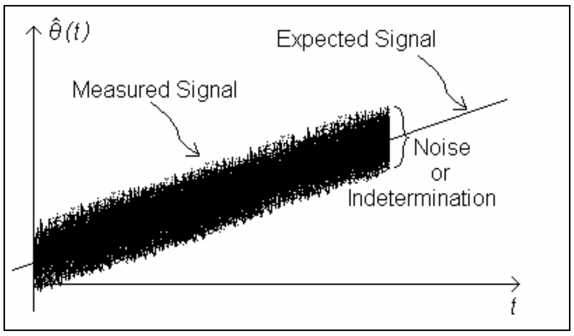


Figure 3 - Result of offset estimation by polling on the Internet on symetric paths and expected signat

3 Measurement and Test Methodologies

All implementations and tests were performed on Windows platform. The Windows system clock doesn't have a resolution proper for time measurement due to the fact that its time update is performed by timer interruption. On Windows 2000/XP, the time granularity is approximately 10ms, while on Windows 95/98/Me it is approximately 50ms. In order to avoid this limitation, a new high resolution clock was used. It was implemented to operate in kernel mode as a *Windows 2000 WDM device* Driver [11][12][13] and it is based on high resolution counters. In Pentium machines, these high resolution counters are denominated *Time Stamp Counter* and are implemented inside the microprocessor. As a result, a clock with 0.1µs of resolution was created. This clock implementation is very simple and due to the fact that it is based on a hardware counter, it doesn't affect the computer performance. The time/date is calculated only when there is a request and is a read-only clock; no adjust can be performed in usual operation. The estimation mechanism uses this high resolution clock as an assistant time clock (T_{assist}).

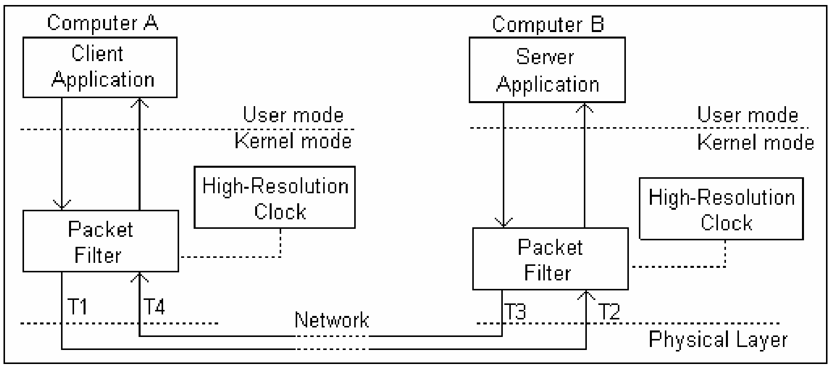


Figure 4 - Example showing how the packet filter and the high resolution timer acts on two machines

Besides the fact that this high resolution clock allows measurement of time with a low time granularity, a complex OS like Windows has many delay sources that may affect time measurement performed over network packets. In order to allow accurate time measurement, it was developed a network filter capable of intercepting packets directly from the network interface card. This filter intercepts the packets just before leaving the computer and just after arriving to the computer. After intercepting a packet, the filter fills a proper field with the current time (read from the assistant clock) and resumes the packet. The combination of the packet filter and the high resolution timer allows quality time measurements over local networks and also over long distance networks as the Global Internet. Figure 4 illustrates two computers with both Device Drivers.

The last point to be considered is the network topology used for the tests. The topology is composed by two parallel paths: an unpredictable delayed path (e.g.: Internet) and a direct high-speed path, as shown in Figure 5. We emulated the unpredictable path with a computer which delayed the packets. In this topology, the delayed path is used as the environment where the estimation techniques would be tested. The high-speed path is used to correctly evaluate the estimation performance.

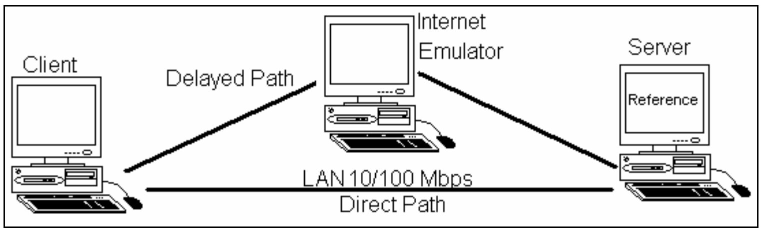


Figure 5 - Triangular topology used to test estimation.

4 Estimation Model

As already mentioned, we adopted a *drift* equal to zero. If we compare the *offset* between the assistant clock and a reference clock along the time and if we don't act in any of them, the expected *offset* is a straight line (see Figure 3). Once the clocks are in the same machine, the obtaining of this straight line is very easy. But if the clocks are in two separated machines connected by some communication network (e.g.: Internet), the determination of this straight line becomes difficult due to the stochastic nature of the network.

In order to obtain an instantaneous estimation of *offset* between two clocks, Equation (4) can be used, which results in Equation (6). In this equation, T_{assist} is the average between $T1_{assist}$ and $T4_{assist}$. See Figure 6 for details.

But, this equation gives satisfactory results only over networks with symmetric paths. In order to make the *offset* model by time polling more realistic, we can improve it by adding a zero-mean Gaussian noise N(0, σ^2). Where, the variance (σ^2) depends on the network traffic intensity and on the network path characteristics. This new *offset* approach is alike Figure 3 and the main task consists on finding an estimation of the real *offset* by means of Equation (7), which allows predicting the remote time by Equation (8).

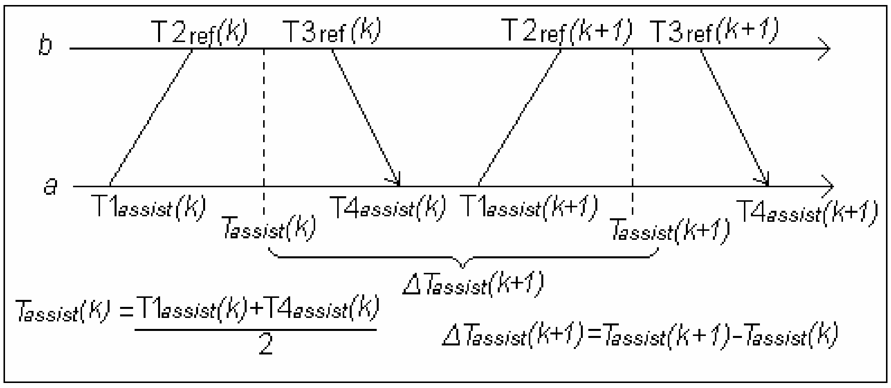


Figure 6 - Time polling between two clocks on different computers, a assistant clock and a reference clock.

$$\hat{\theta}(k) = \frac{(T2_{ref}(k) - T1_{assist}(k)) + (T3_{ref}(k) - T4_{assist}(k))}{2} \quad (6)$$

$$\hat{\theta}(k) = \hat{\theta}(k-1) + \Delta T.Skew \quad (7)$$

$$T_{ref}(k) = T_{assist}(k) + \hat{\theta}(k) \quad (8)$$

Once $T_{ref}(k)$ and *Skew* are found, the local system clock can be adjusted as a copy of the remote machine clock.

4.1 Kalman Filtering Approach

The Kalman filter is a generalization of the Wiener filter with the ability to accommodate vector signals and noise which, additionally, may be nonstationary. It is based on a state model that allows estimating the evolution of a signal embedded in noise. If the noise is Gaussian, Kalman filter is an optimal minimum mean square error (MMSE) estimator, and if not, it is the optimal linear minimum mean square error (LMMSE) estimator [14][15][16].

We start this section making a substitution of variables. We consider a discrete signal *x1(k)*, which corresponds to the real *offset* $\theta(k)$, and *x2(k)*, which corresponds to the *skew* for a given polling packet *k*. $\hat{\theta}(k)$ is the measured *offset* signal. A linear system may be modeled by

$$\begin{aligned} \begin{bmatrix} x1(k) \\ x2(k) \end{bmatrix} &= \begin{bmatrix} 1 & \Delta T \\ 0 & 1 \end{bmatrix} . \begin{bmatrix} x1(k-1) \\ x2(k-1) \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} .W(k) \\ \hat{\theta}(k) &= \begin{bmatrix} 1 & 0 \end{bmatrix} . \begin{bmatrix} x1(k) \\ x2(k) \end{bmatrix} + V(k) \end{aligned} \quad (9)$$

V(k) is the measurement noise and *W(k)* is the system noise. The interval ΔT between samples may vary, therefore ΔT is better replaced by $\Delta T(k)$. The matrixes of the model are named:

$$A = \begin{bmatrix} 1 & \Delta T(k) \\ 0 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad X = \begin{bmatrix} x1 \\ x2 \end{bmatrix}, \quad H = \begin{bmatrix} 1 & 0 \end{bmatrix} \tag{10}$$

From the previous model appears the typical vectored Kalman filter:

$$\begin{aligned} &\widehat{X}^-(k+1) = A.\widehat{X}^+(k) \\ &P^-(k+1) = A.P^+(k).A^T + B.Q.B^T \\ &K(k) = P^-(k).H^T.(C(k) + H.P^-(k).H^T)^{-1} \\ &\widehat{X}^+(k) = \widehat{X}^-(k) + K(k).(\widehat{\theta}(k) - H.\widehat{X}^-(k)) \\ &P^+(k) = (I - K(k).H).P^-(k) \end{aligned} \tag{11}$$

where *K(k)* is a 2x1 matrix, *P(k)* are 2x2 matrixes, *Q* and *C* are scalars. *Q* is the variance of *w(k)* and *C* is the variance of noise *v(k).* $\widehat{X}$ is the estimation of *X*, $\widehat{X}^-$ is the a priori estimation and $\widehat{X}^+$ is the correction of the estimation. Note that, the filter inputs are the measured *offset* and the measured interval Δ*T(k).*

In order to reduce the influence of time correlation present in the noise and neglected by the model, an approach called *artificial pseudo noise addition* [17] was used and tested. *Artificial pseudo noise addition* increases the filtering constant *K(k)* and allows the filter to converge faster. This approach replaces the second line of (11) with.

$$P^-(k+1) = A.P^+(k).A^T + B.Q.B^T + Q'(k) \tag{12}$$

where:

$$Q'(k) = A_{11}P_{12}^+A_{12}^T + A_{12}P_{12}^{+T}A_{11}^T + A_{12}P_{22}^{+T}A_{12}^T \tag{13}$$

The correct estimation of the noise variance *C* is very important in order to correctly tune the filter because it influences the determination of the filter gain matrix *K(k).* Unfortunately, due to the fact that the Internet traffic varies along the time, it causes unpredictable changes in the variance. Fortunately, the maximal error that a packet may contain can be estimated. As already mentioned, the *offset* error happens due to the time asymmetry between forward and backward directions covered by a given packet. So, we can deduct that the maximal error occurs with the maximal asymmetry, i.e., when the packet consume all its travel time in only one direction and consume no time in the other. Figure 7 shows an example to clarify.

It is not possible to know the real asymmetry of a packet travel, but it is possible to estimate the maximal asymmetry that a packet travel would contain. For *skew* equal to zero, this asymmetry may be precisely calculated by (14) and for small *skew*, the error introduced by the Equation can be disregarded (e.g.: *Skew* less than 1ms/s). Equation (15) gives the maximal variance of the one sampling packet from the asymmetry.

The usage of the dynamic $\sigma^2_{\tilde{\theta}_{\max}}$ as the C parameter has given improved results during practical tests in comparison with the traditional noise variance estimation. As already mentioned, the purpose of the filter is to achieve an equation like (7). This equation can be obtained by replacing $\hat{\theta}(k)$ with x1(k) and *skew* with x2(k) from the matrix $\hat{X}^{+}$(k). in (11).

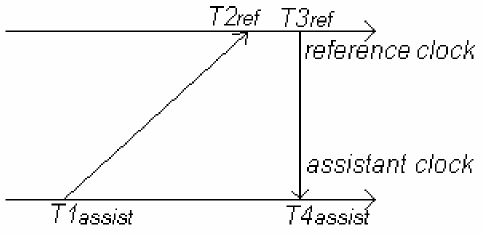


Figure 7 - Example of absolute asymmetry. The total travel time corresponds to the time consumed in one direction

$$\tilde{\theta}_{\max}(k) = \frac{(T4(k) - T1(k)) - (T3(k) - T2(k))}{2} \qquad (14)$$

$$\sigma^2_{\tilde{\theta}_{\max}}(k) = \tilde{\theta}^2(k) \qquad (15)$$

4.2 Paths Redundancy

When the Kalman filter is applied to samples acquired through only one path, the results may be unsatisfactory. This is due to the fact that the path length in each direction may vary unpredictably and when the routers are overloaded, a non-zero-mean asymmetry may occur for long periods. In order to reduce this problem, the estimation algorithms may use a strategy called paths redundancy. This technique consists in forcing the polling packets to voyage through different paths between two machines. To make this strategy possible, the alternative paths are constructed by means of assistant computers which act as routers. See Figure 8 for clarifying this concept. In this example, a computer A performs time measurements on a computer B. The measurements is performed through route 1 (the direct path), and also through alternative routes (routes 2 and 3 in this example). These alternative routes are forced by sending packets to assistant computers that transmits the packet to the correct destination.

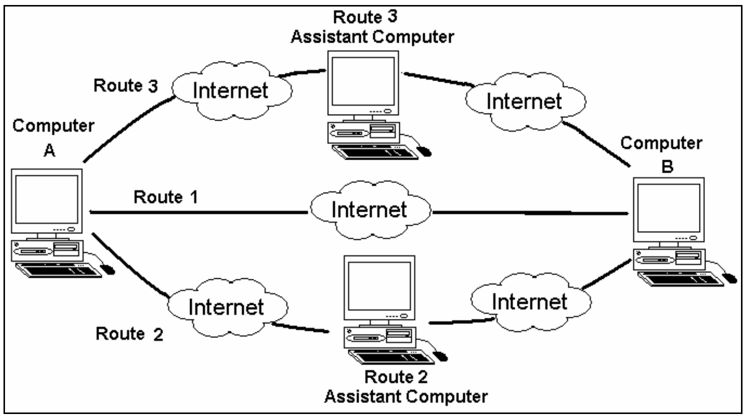


Figure 8 - Path redundancy concept

Each sample acquired through a different path is used to perform a partial *offset* estimation. Each partial estimation is combined with others to generate a final one. The final estimation is calculated by a weighted arithmetic average, where the weights are given by the inverses of the maximal variances. In the previous example, this average is performed by Equation (16), with σ^2_{route} different from 0.

$$\hat{x}1_B = \frac{\frac{1}{\sigma^2_{route1}}\hat{x}1_{route1} + \frac{1}{\sigma^2_{route2}}\hat{x}1_{route2} + \frac{1}{\sigma^2_{route3}}\hat{x}1_{route3}}{\frac{1}{\sigma^2_{route1}} + \frac{1}{\sigma^2_{route2}} + \frac{1}{\sigma^2_{route3}}} \qquad (16)$$

4.3 References Redundancy

The references redundancy is another strategy to improve the *offset* estimation and to increase the robustness of the system in situations when a reference computer becomes inaccessible. This strategy is an enhancement of the paths redundancy and is based on the simultaneous usage of more than one remote computer as reference. All this references must be previously synchronized among them. It acts similarly to the paths redundancy strategy, combining partial estimations using a weighted arithmetic average. The weights are calculated considering the variance and a reference quality parameter for each reference. This quality parameter may be the *stratus* of the reference. An example of the reference redundancy can be seen in Figure 9, where the final estimation is determined by (17) and the weight is determined by (18). Another important detail is the fact that all references must be synchronized among them. To reference computers using GPS or atomic clocks as time source, this synchronism is implicit.

$$\hat{x}1_B = \frac{\frac{W_{ref1}}{\sigma^2_{ref1}}\hat{x}1_{ref1} + \frac{W_{ref2}}{\sigma^2_{ref2}}\hat{x}1_{ref2}}{\frac{W_{ref1}}{\sigma^2_{ref1}} + \frac{W_{ref2}}{\sigma^2_{ref2}}} \qquad (17)$$

$$W_{ref} = \frac{1}{Stratus_{ref}} \qquad (18)$$

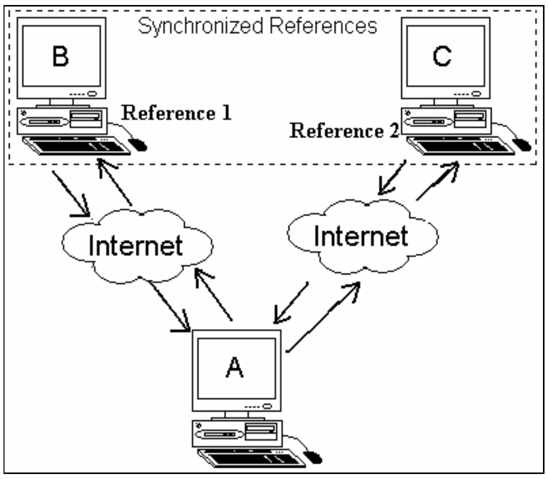


Figure 9 - Reference redundancy

A computer that uses both reference redundancy and path redundancy simultaneously may determine its final estimation by (19), which combines (16) and (17) together.

$$\widehat{x}1_B = \frac{\sum_{ref=1}^{n}\sum_{route=1}^{m(n)} \frac{1}{W_{ref}\,\sigma^2_{ref,route}}\,\widehat{x}1_{ref,route}}{\sum_{ref=1}^{n}\sum_{route=1}^{m(n)} \frac{1}{W_{ref}\,\sigma^2_{ref,route}}} \qquad (19)$$

5 Tests

In order to verify the viability of the proposed model, many tests were performed. The results of the main tests presented in this work are:

Test 1 - Offset estimation between two machines using population variance as C parameter in Kalman filter;

Test 2 - Offset estimation between two machines using pseudo-noise addition and population variance as C parameter in Kalman filter;

Test 3 - Offset estimation between two machines using time limited pseudo-noise addition and population variance as C parameter in Kalman filter;

Test 4 - Offset estimation between two machines using maximal variance per sample as C parameter in Kalman filter;

Test 5 - Offset estimation between two machines using pseudo-noise addition, population variance as C parameter in Kalman filter and path redundancy.

As mentioned on section "*Measurement and Test Methodologies*", the tests were not performed over the real Internet, but over an emulated network with controlled traffic intensity which allowed a parallel measurement of remote time without delays. The traffic features used on this emulator were previously gotten from a measurement performed on the real Internet between Brazil and Austria during 48 hours. As result, the emulator created a traffic delay with exponential probability density function of 250ms average plus 200ms offset on each direction.

This probability density is applied in both directions of a polling packet travel, which generates a time measurement error with probability density as shown in Figure 10. This error has a variance of 1257ms^2 (it corresponds to a standard error of approximately 35,5ms).

All tests were performed during a period of 12 hours and with a sampling period of 1s. The four first tests intended to verify the Kalman filter as an estimator of *offset* between two machines through only one path. In the case of the last test presented, this sample period was used on both paths.

The following table shows a synthesis of the tests results.

Table 1 - Tests results

Test	Convergence time to achieve 1ms error [samples]	Standard error deviation after 30000 samples [ms]
1	20000	0,5
2	10000	0,35
3	10000	0,3
4	8000	0,1
5	3000	0,07

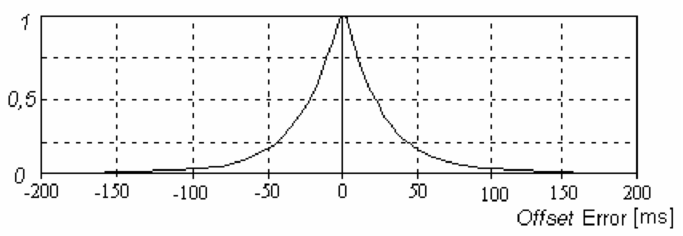


Figure 10 - Probability density of measurement error

The first test uses a constant noise variance of 1257ms^2 as parameter C of the Kalman filter (See Equation (11)). The pseudo-noise addition was introduced in the second experiment. With this addition, the convergence to zero mean improved.

The third test had as targets: accomplishment of convergence to zero mean and decrease of standard deviation on steady state. In this test, the pseudo-noise was added only in the first 15000 samples and then turned off. In the forth test, besides the addition of pseudo-noise, the concept of dynamic variance usage was analyzed. To estimate the dynamic variance, we used the squared of packet travel time, as already mentioned in previous section, but discounting a minimal travel time of a population inside a window of 5000 samples. In order to limit the influence of the dynamic variance on samples with very short travel time or with erroneous measurement, the variance was limited to 50ms^2 in its lower bound. The equation used to obtain the dynamic variance was.

$$\sigma^2 = \max((50ms)^2, (T_{travel} - T_{\min})^2)$$

As result, by using the dynamic variance, we obtained a steady state error with zero mean and standard deviation of approximately 0.1ms. This result can be seen in Figure 11.

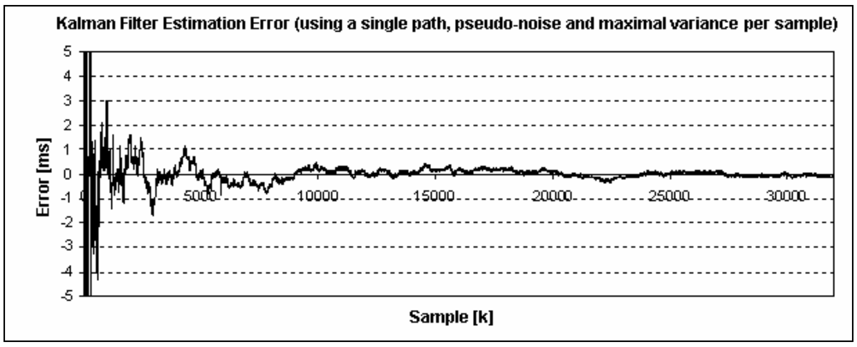


Figure 11 - – Kalman Filter using dynamic variance

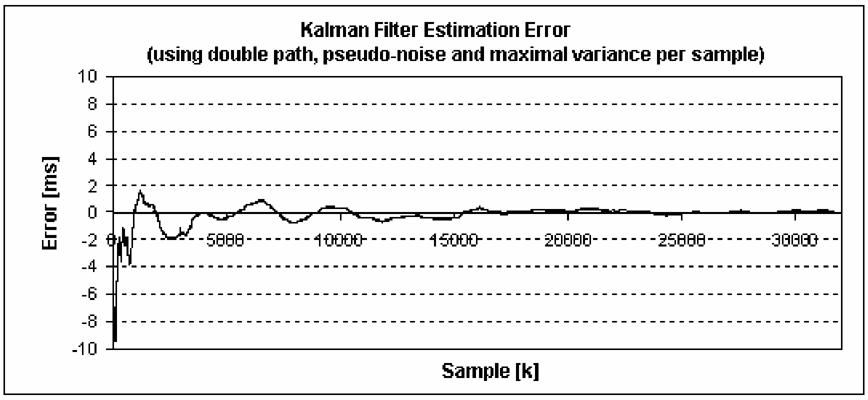


Figure 12– Kalman Filter using redundancy

The last test presented in this paper considered the paths redundancy, besides the pseudo-noise addition and dynamic variance. The test used two paths and each of them emulated the Internet by using different seeds to initialize their pseudo-random generators. For this test, it was used the same noise vector as used by the previous tests. This technique doubled the total network bandwidth consumed by the system. The standard deviation decreases in comparison to the single path technique, as it can be seen in Figure 12.

6 Conclusions

The tests performed with the proposed model achieved time estimation with standard error of approximately 100μs for an input signal with standard error of approximately 35ms without the usage of path redundancy. Considering that the NTP uses the result of Equation

(6) as *offset* estimation, we can conclude that the improvement was better than two orders of magnitude in relation to NTP estimation method [18] and the inclusion of two paths of redundancy almost doubled the precision. This shows the viability to reduce *offset* error by increasing the number of redundancy paths. The appropriately choice of these paths is very important. The reference redundancy wasn’t explicitly tested, but, in cases where the references are perfectly synchronized among them, this redundancy is similar to the path redundancy without the assistant computers, which turns it superior to the paths redundancy.

7 ACKNOWLEDGMENT

The authors would like to acknowledge the contribution to this work from Dr. David Fernandes and Dr. Elder Moreira Hemerly for their valuable comments, and the FAPESP Tidia program.

8 References

[1] National Institute of Standards and Technology, http://tf.nist.gov/; visited in June, 2005.

[2] Time Service Department of U.S. Naval Observatory, http://tycho.usno.navy.mil/; visited in June, 2005.

[3] RFC868 “Time Protocol”, RFC-Editor, www.rfc-editor.org.

[4] RFC867 “Daytime Protocol”, RFC-Editor, www.rfc-editor.org.

[5] RFC1305 “Network Time Protocol”, RFC-Editor, www.rfc-editor.org.

[6] RFC1769 “Simple Network Time Protocol”, RFC-Editor, www.rfc-editor.org.

[7] Precision Synchronization of Computer Network Clocks - http://www.ee.udel.edu/~mills/database/papers/fine.pdf.

[8] Measured Performance of the Network Time Protocol in the Internet System - http://www.faqs.org/rfc/rfc1128.pdf.

[9] Mills, D. L., “Internet Time Synchronization: The Network Time Protocol” IEEE Transaction on Communication, Vol. 39, No. 10, October 1991.

[10] Berthaud J. “Time Synchronization Over Networks Using Convex Clousures”. IEEE/ACM Transaction on Networking, 2000.

[11] Microsoft Corporation 1985 – 2000 , (2000) “Microsoft Windows 2000 Driver Development Kit”.

[12] Oney, W., “Programming the Microsoft Windows Driver Model”, Microsoft Press, 1999.

[13] Baker, A.; Lozano, J., “The Windows 2000 Device Driver Book” 2nd Edition, Prentice Hall PTR, 2001.

[14] Kay, S. M. “Fundamentals of Statistical Signal Processing”, Prentice Hall International, Inc. 1993.

[15] Maybeck, P. S. “Stochastic models, estimation and control”, Vol. 1, Academic Press Inc. 1982.

[16] Bozic, S. M. “Digital and Kalman Filtering”, Edward Arnold, 1979.

[17] Maybeck, P. S. “Stochastic models, estimation and control”, Vol. 2, Academic Press Inc. 1982.

[18] Butner, S. E., Vahey S., “Nanosecond-scale Event Synchronization over Local-area Networks” 27th Annual IEEE Conference on Local Computer Networks, 2002.